UNIVERSITY OF THE PHILIPPINES MANILA

COLLEGE OF ARTS AND SCIENCES

DEPARTMENT OF PHYSICAL SCIENCES AND MATHEMATICS

NAVIGATION DECISION SUPPORT SYSTEM BASED ON REAL-TIME TRAFFIC CONDITIONS AND TRAFFIC INCIDENTS USING CROWDSOURCING

A special problem in partial fulfilment

of the requirements for the degree of

Bachelor of Science in Computer Science

Submitted by:

Mauro Felipe M. Magpili

June 2014

Permission is given for the following people to have access on this SP:

Available to general public	YES
Available only after consultation with author/SP adviser	NO
Available only to those bound by confidentiality agreement	NO

ACCEPTANCE SHEET

The Special Problem entitled "Navigation Decision Support System Based on Real-Time Traffic Conditions and Traffic Incidents Using Crowdsourcing" prepared and submitted by Mauro Felipe Magpili in partial fulfillment of the requirements for the degree of Bachelor of Science in Computer Science has been examined and is recommended for acceptance.

_____

Geoffrey A. Solano, M.Sc
Adviser

EXAMINERS:

	Approved	**Disapproved**
1. Gregorio B. Baes, Ph.D. (candidate)	_____	_____
2. Avegail D. Carpio, M.Sc.	_____	_____
3. Richard Bryan L. Chua, M.Sc.	_____	_____
4. Aldrich Colin K. Co, M.Sc. (candidate)	_____	_____
5. Ma. Sheila A. Magboo, M.Sc.	_____	_____
6. Vincent Peter C. Magboo, M.D., M.Sc.	_____	_____
7. Bernie B. Terrado, M.Sc. (candidate)	_____	_____

Accepted and approved as partial fulfillment of the requirements for the degree of Bachelor of Science in Computer Science.

_____ _____

Ma. Sheila A. Magboo, M.Sc.
Unit Head
Mathematical and Computing Sciences Unit
Department of Physical Sciences
and Mathematics

Marcelina B. Lirazan, Ph.D.
Chair
Department of Physical Sciences
and Mathematics

_____

Alex C. Gonzaga, Ph.D., Dr.Eng.
Dean
College of Arts and Sciences

Abstract

Transportation and traffic problems have always been a major concern in developing countries due to its impact to the economy. Traffic conditions and traffic incidents, such as road accidents, flooded streets and closed roads, are some of key informations on deciding which route to take during travel. However, no system has been implemented to share this information among motorists that is accessible on the road. This study created a real-time navigation decision support system for motorists to help them make informed decisions while travelling. The system takes advantage of current technologies such as Global Positioning System, Graphical Information System, and Android in mapping traffic incidents and traffic conditions and uses crowdsourcing to collect the necessary data. This opens up future studies for long-range analysis of continuous long-range data, specifically traffic incidents and road conditions.

Contents

List of Figures

List of Tables

I. INTRODUCTION

A. Background of the Study

Transportation and traffic problems have always been important and recurring problems. These need to be addressed because of their impact on the economy. Consequently these problems affect the price of goods, tourism, profit of business and human time resources. [12] According to the National Center for Transportation Studies (NCTS) traffic congestion in Metro Manila amounts to 100 billion pesos of financial lost yearly. [35]In an attempt to solve these problems, various infrastructure and policies have been implemented. [12]

According to Philippine Atmospheric, Geophysical and Astronomical Services Administration (PAGASA), an average of 20 typhoons hit Philippines every year which is geographically located at side of the Pacific Ocean. This causes heavy street floods in Metro Manila that have been an existing problem for years.

According to the Department of Health (DOH), road accidents are the fourth leading cause of death in the Philippines. Furthermore, the DOH states that if the trend continues road accidents can become the leading cause of death in the Philippines in 2020. It is also important to note that the statistics gathered by the Metropolitan Manila Development Authority (MMDA) and Philippine National Police-Highway Patrol Group (PNP-HPG) shows that at least two vehicles in a major highway in the Philippines get involved on an accident every hour. All in all, fifty road accidents occur every day. As mentioned in House Bill No. 5436, these are not small numbers in a country where 78 percent of the population rely on buses, jeepneys, taxis, trains and tricycles for mobility. [32]

Traffic estimation has been one of the challenging problems for the transportation community due to the limited deployment of sensing infrastructure with high maintenance cost incurred by the government. Thus collecting reliable traffic data has been one of the challenges in traffic data assimilation. [18]

The DOTC in partnership with the MMDA, Cebu City Government and the World Bank lunched the first Philippine Transit App challenge. According to the talk which took place on July 2, 2013, "we need to empower the public with better decision making abilities which will allow everyone to have smoother trips".

B. Statement of the Problem

Transportation and traffic problems have always been key problems of developing countries due to its impact on the economy. [12]

Traffic is more than just red lines on a map according to the official website of Waze, a community based navigation system. [32] Traffic incidents such as flooded roads, road accidents, closed roads and road conditions also affect the traffic. Smart phone technology has yet not been used in the Philippines to gather this data.

As follows, how to validate the collected data will also be an important issue to ensure the accuracy of reports. It will also necessary to notify motorist about current traffic condition specifically the average speed along a road segment to help users on navigation decisions. Maintaining privacy on collected traffic data will be another problem.

Furthermore it will also be important to share this information to the public in timely a manner. In the case of road accident, it must also be reported to the appropriate authorities for immediate response.

1. Due to the dynamic and changing road conditions and traffic incidents, this information must be accessible on the road as it is a critical factor in making travel decisions.

C. Objectives

General Objectives

To develop a decision support and graphical information system that aims to

1. collect reports and data about key information (closed roads, flooded roads, road accident, traffic conditions) on road conditions
2. verify the collected data and reports
    a. remove outliers from traffic data set
    b. users of the system can verify reported traffic incidents
3. combine the collected data specifically location data to compute for the average speed of cars on a road segment
4. share key information regarding road condition among the users of the system
5. provide communication between the public and the appropriate authorities about road accidents
6. provide a system that is accessible on road

Specific Objectives

The system has four main users as follows:

1. Motorist
2. Traffic Investigator
3. Traffic Operations Center Officer
4. System Administrator

The users have the respective functionalities as follows:

Motorist

Provide features that will allows a Motorist to

1. provide/input the necessary details regarding reports only if there are no reports with the same type, e.g. road accident, road closed or flooded street, within a 10 m radius
    a. Classify the reports into the following types
        i. Flooded road
        ii. Road accident
        iii. Road closed
    b. describe the nature of the flooded road and road accident reports
        i. classify a flooded road report according to
            1. Not passable to light vehicles
            2. Not passable to all types of vehicle
        ii. classify a road accident report according to
            1. Head-on Collision
            2. Rear Collision

3. Collision with animals
4. Rollover
5. Run-off road collisions
6. Car collision involving pedestrians

c. attach additional pertinent information as follows:

i. Location, determined by latitude and longitude points, of a report using GPS

ii. Attach a picture of a report

2. View details regarding road accidents and flooded street report as follows

a. Time of report

b. Latest time of reports

c. Validity of a reports based on

i. Indicating if verified by system

ii. Number of users of verified it

iii. Photos

3. Verify nearby reports by answering “is there road closed/flooded road/road accident up ahead?”

4. Provide a map containing

a. traffic incident reports that are valid within the time frame

b. traffic conditions of roads bounded by a virtual tripline

Traffic Investigator

Provide features that allow a Traffic Investigator to

1. View the necessary details to respond to a road accidents by
    a. View a list of road accident assigned to a traffic investigator
    b. access a visual representation of the location of road accidents assigned to
2. Update a road accident report by changing to the appropriate road status as follows:
    a. Responding
    b. Resolved

Traffic Operations Center Officer

Provide features that allow a MMDA Traffic Operations Officer to

1. Monitor the road accident reports and traffic investigator by
    a. Providing the current locations of Traffic Investigator on Google Maps
    b. Providing a list of road accident reports
    c. Providing details regarding the road accidents reports
2. Maintain the accuracy of road accident, road closed and flooded street reports by
    a. Updating the status of reports
    b. Deleting a report
3. Assign a traffic investigator to a road accident report only if the road accident is within the traffic investigator's jurisdiction and the traffic investigator is online.

4. View a list of users involved in a road accident report
5. Maintain the accuracy of closed roads report by:
    a. Adding a traffic incident report
    b. Delete a traffic incident report
    c. Update a traffic incident report
6. Set up zone trip lines
7. Set up jurisdiction bounds

System Administrator

Provide features that allow a System Administrator to:

1. Create the following accounts of the system:
    a. Road Accident Monitoring Officer
    b. Traffic Investigator
    c. System Administrator

D. Significance of the study

Taking advantage of crowdsourcing allows access to a large pool of data. This allows a better representation and real time information about actual road condition.

Shared resource regarding flooded streets, traffic condition and current accidents allows motorists to prepare for the route to the destination. This allows better coordination among motorists, help traffic conditions to improve and help avoid road stress.

With the location data gathered, reconstructing the state of the highways is possible and is essential to gauge the speed and traffic volume in a given highway. Unlike the inductive loop

detectors that count road occupancy and provide unreliable speed measurements at a single point on the highway, GPS based cell phone provides an accurate position and speed information at low maintenance cost.

With the application of crowdsourcing through mobile devices, a system for motorist to exchange key information regarding current road conditions on the road can be created. With this collaborative tool and visual representation of traffic incidents and traffic conditions which can be accessed on the road will greatly help motorist make informed decision. Shared real time information regarding traffic condition, road accidents, flooded streets and road closed will allow motorist to avoid this areas.

The implementation of this kind of system would reduce traffic because motorist would likely avoid going to flooded areas that can cause a chain of motorist stucked in one street. The system takes advantage of mobile technology currently present.

Immediate action in road accidents is critical in saving lives. Fast communication and coordination among key person or rescuers are essential. A system that provides direct communication to the key person and share this information among users of the system would be solving the problem in parallel.

The DOTC in partnership with the MMDA, Cebu City Government and the World Bank lunched the first Philippine Transit App challenge. According to the talk which took place on July 2, 2013, “we need to empower the public with better decision making abilities which will allow everyone to have smoother trips”.

E. Scope and limitation

1. Accuracy of reports submitted are validated by fellow users
2. The system accesses Google Maps through GPRS, WIFI, 3G or HSDPA since the Term of Services specified by the Google Map API does not allow pre-fetching, or cache.
3. Streets are provided by Google Maps. Those not found in Google Maps API will not be supported.
4. Accuracy of the map and streets is determined by the Google Map API.
5. The latitude, longitude, and addresses generated are generated from the GPS receiver.
6. Reporting flooded streets, roads closed and roads accident cover the Metro Manila Area
7. The web server must support PHP 5.1.0 or higher to work.
8. The Smartphone used has at least android 2.3 installed as an operating system
9. The Smartphone used is equipped by GPS receiver.
10. MinPts and Eps are specified by a traffic expert for the outlier detection to work.
11. Accounts in the system can't be deleted due to restrictions in the design of the database

F. Assumption

1. Data from Google Maps API are accurate
2. User evaluation of reports are accurate
3. There are enough users using the application so that the data will be correctly populated
4. The error in measure of GPS σ is correct as used in the system.
5. Traffic data has a normal distribution
6. MinPts and Eps will be specified by an expert.

II. REVIEW OF RELATED LITERATURE

Wireless technology is now widely available Philippines. According to a study, one in two Filipinos is a mobile phone subscriber.[20] The current trend now in the market where the rising affordability in Smartphones show that the country is one of the fastest growing Smartphone market in the Southeast Asia region.[11] According to Mobile Life 2013 an annual study about the behavior of mobile users by TNS, 53% of Filipinos own a Smartphone. The usage of smart phones to browse the internet increased from 32% in 2012 to 45% in 2013. [36] This suggests that the mobile internet usage will continue to rise in the following years.

According to a study made in Bangladesh regarding disaster information management, mobile technology may be used to coordinate information or disseminate predisaster warnings and post disaster announcements. [17]

Due to the ease of integration of GPS, a lot of systems have been made to take advantage of this feature. For instance in agriculture, GPS technology is used to monitor the application of fertilizer and pesticides. In aviation, GPS has been used by pilots for en route navigation for airport approaches. In Environment, GPS helps survey disaster areas by mapping the movement of environmental phenomena, such as forest fires, oil spills, or hurricanes. In ground transportation, GPS helps with automatic vehicle location and in vehicle navigations which allow a vehicle to be shown on an electrical map, allowing drivers to keep track of their current location and look up to the destination. In surveying, GPS can be used to get precise information.[24] In Sharjah, UAE, an online GPRS-Sensor array for air pollution monitoring was

implemented and tested. The pollution server used Google Maps to display real-time pollutants level and location in large urban areas. [1]

The Geographical Information System is widely used tool for visualization of data. GIS system has been used to visualize accident data and analysis of hot spots in highways. With the development in Smartphone technology, the integration of maps is commonly used in the mobile platform to help get the information across the users. Google Maps and OpenStreetMap are platforms that can be used for mapping information. [3]

Crowdsourcing (CS) has been used to solve a multitude of problems by enlisting a crowd of humans to do specific tasks. Some examples of such systems are Linux, Ushahidi's crowdmap, Mechanical Turk-based systems and Waze.[4]

In the Philippines, several successful Crowdsourcing systems have been implemented. Disaster Relief in Laguna; A Geographical Information System Through Crowdsourcing on Facebook is a system made to collect disaster's information and post the disaster system to the wall of a user. The said system also allows sharing of information among friends and show recent added disaster within a time frame of 2 days. [16]

Another Crowdsourcing system in the Philippines is a cross-language tool using mobile devices to organize the EMR content for building standardize medical terminologies.[13]

Twitter is used as a Croudsourcing tool to collect data regarding flooded streets. The media, in turn, distribute this data through television news and radios.

It also important to mention the Crowdsourcing Act of 2013 (Senate Bill No. 73) was filled by Senator Teofisto Guingona III. The Senate Bill aims to allow the public to participate in the law making process by allowing a collective feedback from the public.[35]

The following are the advantages of the proposed system compared with the current system in place.

In comparison with Twitter as a crowdsourcing tool to collect data regarding flooded streets the proposed system is specifically designed for crowdsourcing. The dissemination of information from Twitter to the public is done by the media. Often time one has to wait for segment in the T.V. or radio news regarding the said information. The proposed system will be much better because this information is readily accessible anywhere and anytime. Flooded streets can also be reported as they are seen. The reports are better represented using maps. The reports can also be verified by other users of the system.

In comparison with nababaha.com, the proposed system has its advantages. As mentioned in their website, these flood hazard maps do not show the current flooding in areas. In contrast to the system, the flood reports are real time and updated. It also important to note that nababaha.com only maps flooded areas which do not necessarily translate to actual flooded roads within an area. [29] The system solves this problem by concentrating on the flooded roads rather than flooded areas.

On another note, the committee on Public Order and Safety under Representative Amado Espino Jr. (2nd District, Pangasinan) has approve a bill institutionalizing the universal emergency assistance 117 which is a hotline number for emergencies. [32] According to the official website of the MMDA in the event of a vehicular accident, the 136 hotline maybe use to ask for enforcers/investigators to be present on the scene. [33] Road accidents are reported through this hotline numbers. However in a news report written by Perseus Echeminda from Philstar, during the labor strike in the PLDT had rendered the 117 hotline unreachable during the New Year's Eve celebration. A number of residents complained that their distress call for emergency assistance could not get through. Later a patrol 117 operator admitted that the PLDT strike has triggered some technical problems. [34] It is also important to mention that the proposed system does not just notify the appropriate authorities on an event of a road accident, but it also shares this information among the users of the system. This inform the users of the system even before they reach the accident location and therefore allowing them to avoid, prepare and adjust their current route to destination and avoid causing heavy traffic.

Next the current traffic state estimation in Metro Manila will be discussed. Metro Manila has been using legacy loop detectors which measures the speeds of cars passing at certain point in the road to gather traffic data. Several intersections in Metro Manila have been using this for the past years. It is more prone to error and component malfunction because the loop detector system is already outdated. [35]

CCTVs and IP cameras are installed in major roads to gather traffic data. Video analytic software has not yet been implemented in the cameras. [35] As such traffic is only measure based on how the people interpret the live videos.

Recently, cellular based highway traffic monitoring has shown promise for obtaining cheap and reliable real time traffic information, without the high infrastructure and maintenance costs incurred by the government. Using cellular based highway traffic monitoring will also allow collection of data at high penetration rate. GPS based cell phone can provide accurate position and speed information. In contrast to Loop Detectors that can only produce occupancy counts and limited quality of speed measurements at a single point on a highway. [18]

For road constructions and roads closed, the current method of warning motorist about this is through road signs. These do not give enough time for the motorist to prepare and avoid these roads. The signs are only seen as a motorist reaches the road. Compared to the proposed system that warns you even before you reach the road, this will give you ample time to better prepare for the route and actual avoid closed roads.

Four key problems are posed by crowdsourcing systems namely: How to recruit contributors,What can they do, How to combine their contributions, and How to manage abuse.[4]

It is essential to understand how the “crowd” can be motivated to participate in exchanging information among the group. Several solutions were suggested on how to recruit contributions like offering a monetary reward, an example of this is in 2007 Cisco offered an

online idea competition to collect ideas for innovative IT solutions, versus non-monetary rewards. [14]

Consider the phenomena for the sociology of disaster called “convergence”. According to disaster sociologist, during the time of disaster and mass emergencies people will spontaneously converge on location to offer assistance in warning, response and relief and where people who are remote to the event, provide and seek information through social media.[15] If we find a way for people to help without converging to one location, this would greatly increase the accessibility of people to offer assistant.

Due to the nature of the system, community of users helping each other, it is the most practical solution for users to use the system. Access to the important information would only be granted if you are a user of the system. The difficulty to get informed and to inform other people would be solved.[6]

Limiting what information they can feed to the system would avoid the abuses on usage. The system uses another layer of crowdsourcing in which the users of the system will verify the reports.

Information gathered from crowd resources will be organized with use of database to store the information. Reports regarding flooded streets, accidents and road closed would be shared across the users and mapping the key locations using Google Maps. Detailed list of features would be further explained on the objectives.

Traffic data, specifically traffic congestion along a road, would come automatically from users using GPS. We combine this data to reconstruct the highway system, known as data assimilation. This will further be discuss in the theoretical framework.[22]

Ushahidi, CS system designed to collect information from disaster scene and visualizing data for relief decision making. They allowed users to verify a report by clicking a verification button. It leaves the verification problem to the crowds to get a collective feedback. In supplement to this; photos, videos and other comments were used as a way to provide credibility to the reports.[5]

In traffic data, consider when car stops for others reason that does not involve the traffic. In example, consider if a car stops at the side road to eat. This would affect the reconstruction of traffic condition. To address this, we will implement an outlier detection specifically DBSCAN that will be further discuss in the theoretical framework.[9]

GPS based system in gathering traffic data presents privacy problems. This kind of system requires the cars to send their positions raising privacy concerns. Virtual trip lines (VTLs) may be use to address this problem. VTLs are geographic markers stored in the mobile phone client which trigger a position and location update. [7]

Real-time information is critical when it come to disaster systems. With the rise of sale in mobile devices due to rapidly growing affordability and availability of such devices, the

implementation of the collection of data through mobile devices is the perfect instrument to use. Thus the integration of mobile devices in crowdsourcing will be a powerful tool in harnessing real-time information from “crowds” or people resource. With the help of geolocation-aware mobiles and other crowdsourcing system in particular the Google Map, as a visualization tool, we can take advantage of the participation of the masses to achieve a common goal. A community of users aimed in helping each other with collective information or data sets of location of streets which are not passable by vehicles. [8]

III. Theoretical Framework

A. Mobile Technology

Mobile technology is the technology used for cellular information. It allows people to use portable device for communication. It is now very commonly used in the Philippines since it is made more affordable to the public. According to a study, one out of two Filipinos is a mobile phone subscriber[20] and there are two billion mobile users subscriber in the world.

B. Android

Android is one of the popular mobile platform that powers hundreds of mobile devices in more than 190 countries in the world. It's the largest installed base of any mobile platform. Android has more than 300 hardware, software and carrier partners which is the reason why it is one of the fastest growing mobile platforms. It gives you single application model that lets you deploy your apps broadly to hundreds of millions of users. It also give you tools for creating apps that look great and take advantage of hardware capabilities available in each device.[25]

C. Global Positioning System (GPS)

Global Positioning System is a satellite-based navigation system composed of 24 satellites placed into the orbit by the U.S. Department of Defence. Sometime in 1980s it was opened to the public. The GPS satellites rotate around the earth two times a day in a precise orbit and transmit signal information towards the earth. The GPS receiver takes this information and uses it to triangulate the user's exact location. The GPS receiver compares the time difference in

the signal to calculate how far the satellite is. Using several more satellites to get its distance away from the GPS receiver, the user's position can be determined.[28]

D. Yii Framework

Yii is a high-performance, using lazy technique extensively, PHP framework best for developing Web 2.0 applications. It is a free, open-source Web application development framework that promotes clean, DRY design and encourages rapid development. It helps ensure an extremely efficient, extensible, and maintainable end product. It adapts Model-View-Controller design patter and Database Access Object to access and model database data. [26]

E. Appcelerator Titanium SDK

Titanium is an open source JavaScript-based SDK with over 5000 APIs for iOS, Android, Windows, Blackberry and HTML5 that uses an Appcelerator Alloy MVC framework. It has an integrated mobile backend as a service (MBaaS).

F. Speed computation

To compute for current speed of an automobile, consider an automobile moving along path P and travelling the distance Δl during the time interval $\Delta t = t_2 - t_1$ [2]

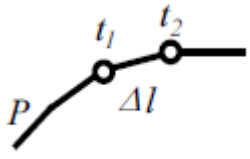


Figure 1 path P

The speed v of an automobile is defined as [2]

$$v = \lim_{\Delta t \to 0} \frac{\Delta l}{\Delta t} = \frac{dl}{dt}$$

The average speed V_a is [2]

$$\text{Va} = \frac{\int v dt}{T} = \frac{\int dl}{T} = \frac{L}{T} \quad (1)$$

Where

L = distance

T = time

In real time measurements, each discrete GPS speed sample V_{kD} contains an error of $V_{ke.}$ Therefore the true speed V_k [2]

$$V_k = V_{kD} - V_{ke} \qquad (2)$$

Hence from equation 1 and equation 2, for a discrete series of speed samples V_{kD} acquired at N uniform time intervals over the time T we approximate the integral to equation (2) V_a , [2]

$$Va = \frac{\sum_{k=1}^{N} \text{Vkd}}{N} - \frac{\sum_{k=1}^{N} \text{Vke}}{N} \qquad (3)$$

Using the trapezoidal rule to compute for $V_{a,}$ [2]

$$V_a = = \frac{\sum_{k=1}^{N} V_{kD}}{N} - \frac{V_{0D} + V_{ND}}{2N}$$

We can compute for the error of measurements by [2]

$\sigma_{va} = \frac{\sigma\sqrt{\sum_{k=1}^{N} N_k}}{\sum_{k=1}^{N} N_k}$, where σ is specified by the GPS manufacturer [2]

We can compute for claimed average speed of an automobile by, [2]

$$V_{claimed} = V_a - c\sigma_{va}$$

By adopting c=2 we can claim with 97.725% confidence interval that the claimed average has been achieved. [2]

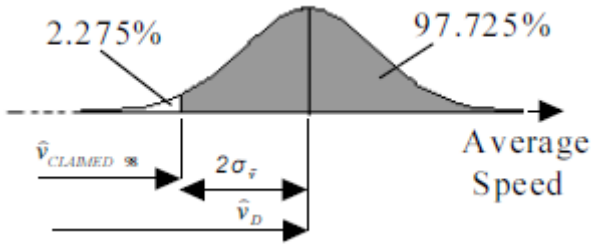


Figure 2 Average Speed Confidence Interval

G. estimated space mean speed (eSMS)

To compute for the average speed of the of a road segement by combining all the calculated average speed of the vehicles given by the formula,

$$eSMS = 1/\left(\frac{1}{m}\sum_{i=1}^{m}\frac{1}{\overline{v}_i}\right)$$

H. Density Based Spatial Clustering of Applications with Noise (DBSCAN)

Definition 1 (16) The *Eps ¡ neighborhood of a point p, denoted by NEps(p), is* defined by *NEps(p) = fq 2P j dist(p,q) <= Eps g.*

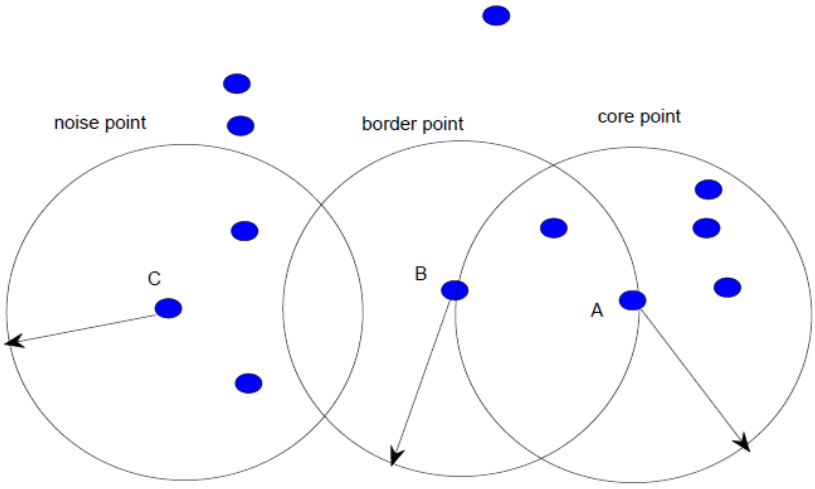


Figure 2.4: Illustrates DBSCAN's key concepts: core (A), border (B), and noise (C) points

Definition 2: (3) The *density-based approach is an approach that regards clusters as* regions in the data space in which the objects are dense and separated by regions of low object density (outliers).

Definition 3: (16) Considering points *p1 and p2 from Figure 2.5, p1 is directly density reachable from p2 if*

1. Points are close enough to each other such that dist(*p1,p2)< Eps, as measured* using Euclidean distance or using any other distance measure.

2. There are at least *MinP ts points in its neighborhood*

Definition 4: (16) A point *p1 is density reachable from a point p2 wrt. Eps and MinP ts if there is a chain of points p1, ..., pn, such that pi+1 is directly* density-reachable from *pi.*

Definition 5: (16) A point *p0 is density-connected to a point pn wrt. Eps and MinP ts* if there is a point *q such that both p0 and pn are density-reachable from q wrt. Eps* and *MinP ts.*

DBSCAN Algorithm

0. Select the values of *Eps and MinP ts for a data set P to be clustered.*

1. Start with an arbitrary point *p and retrieve all points density-reachable.*

2. If *p is a core point that contains at most MinP ts points*

2.1 A cluster is formed,

2.2 Otherwise, label *p as an outlier.*

3. A new unvisited point is retrieved and processed leading to the discovery of further clusters of core points.

4. Repeat step 3 until all the points have been visited.

5. Label any points not belonging to any cluster as outlier

I. Virtual Trip Lines (VTLs)

Virtual Trip Lines are geographic markers stored in a client's mobile phone which trigger recording location and speed data when a probe vehicle's trajectory intersects a trip line. More specifically it is defined by where vtlid is the virual trip line ID, x1, y1, x2 and y2 where (x, y) are the coordinates of the two line endpoints.. This addresses privacy concerns due to the nature of the system recording location data to reconstruct the traffic conditions. This also provides an advantage by providing an alternative to map matching algorithms that are computationally expensive.

IV. Design and Implementation

A. Platform Architecture

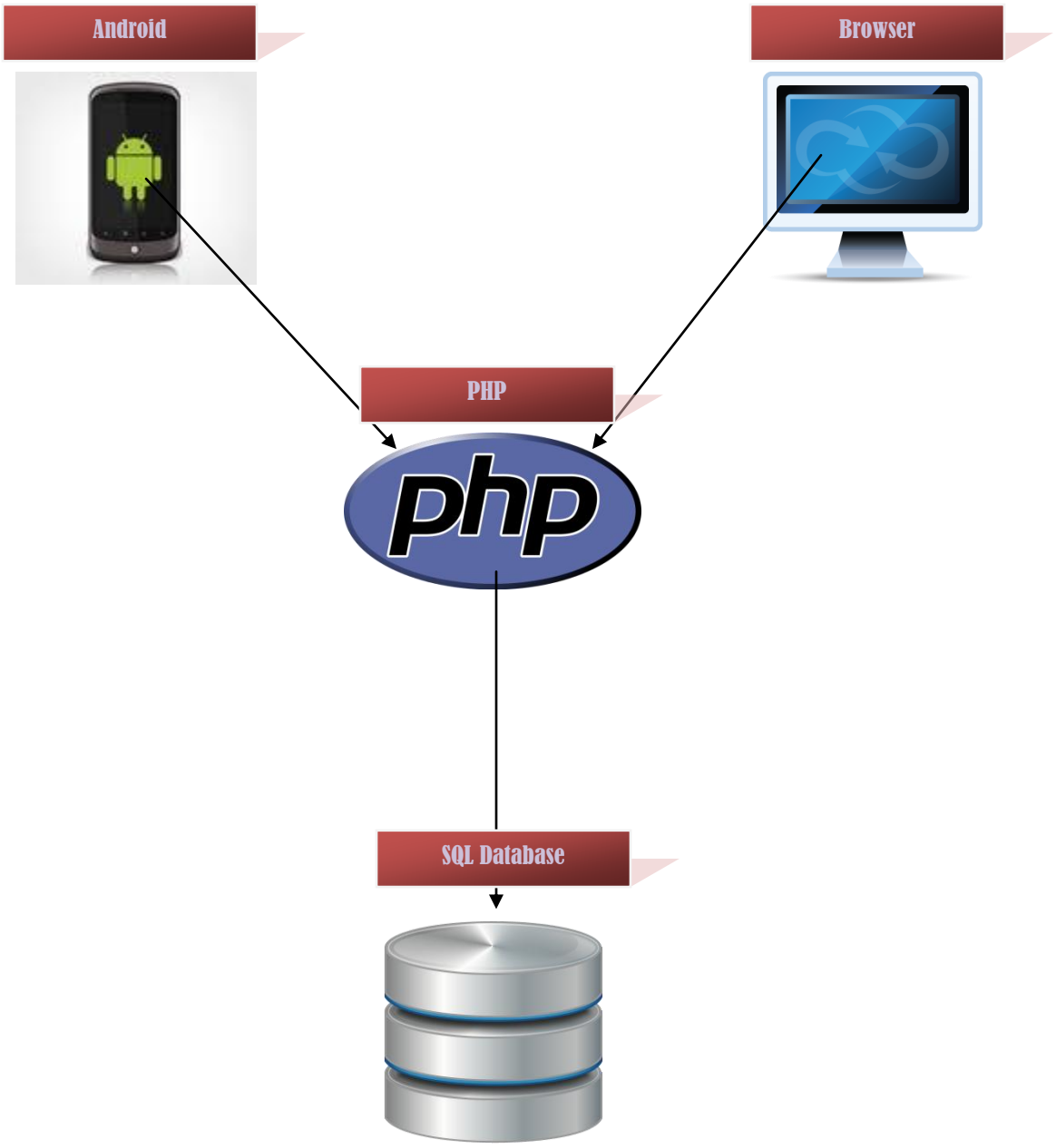


Figure 3 Platform Architechture

B. Use Case

Motorist

The crowd resource is the source of data for traffic incidence and traffic conditions. They are also the consumer of these data for navigation decisions on the road. They also verify traffic incident reports.

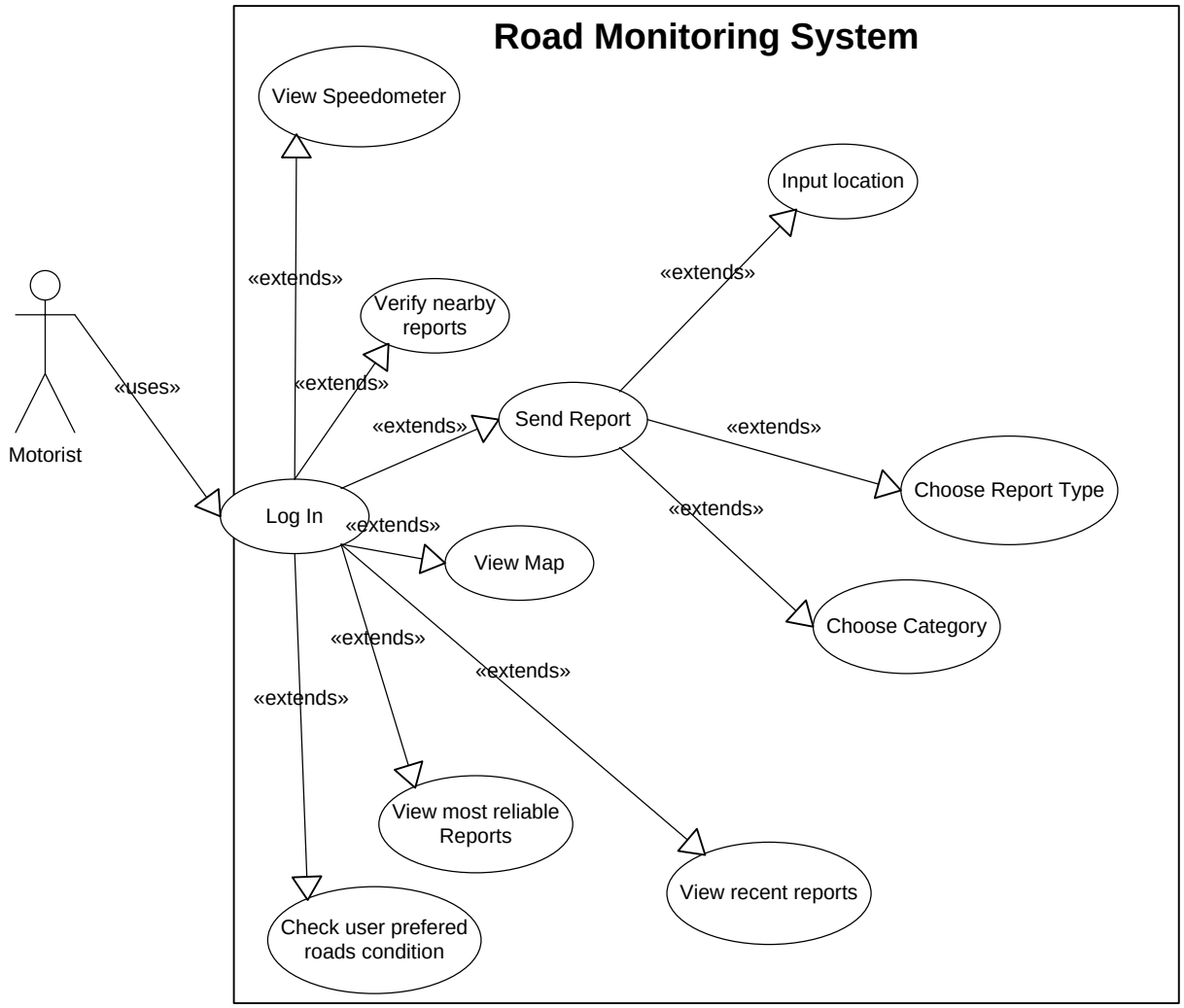


Figure 4 Use Case Motorist

System Administrator

System administrators are responsible for maintaining the system. They are responsible for creating accounts for other System Administrator, Road Accident Monitoring Officer and Traffic Investigator. They are also responsible for deletion of account of a Motorist, System Administrator, Road Accident Monitoring Officer, Traffic Investigator, Off-site Construction Officer and On-site Construction officer.

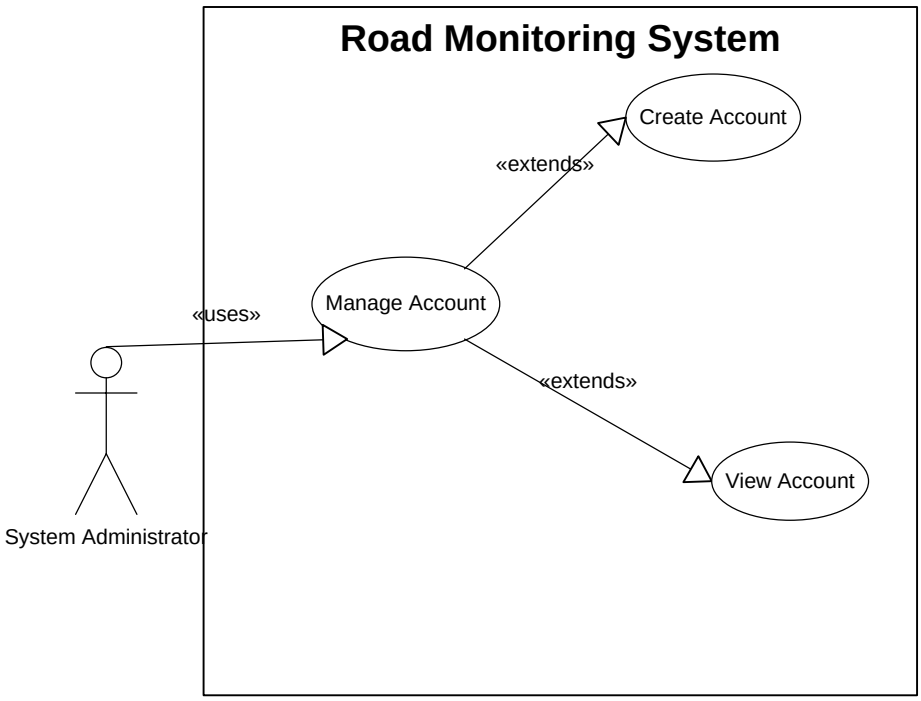


Figure 5 System Administrator

Traffic Operations Center Officer

Road Accident Monitoring Officers coordinate accident reports with Traffic Investigator. They see the most recent and most reliable accident reports and direct Traffic Investigator to which accident to act in response.

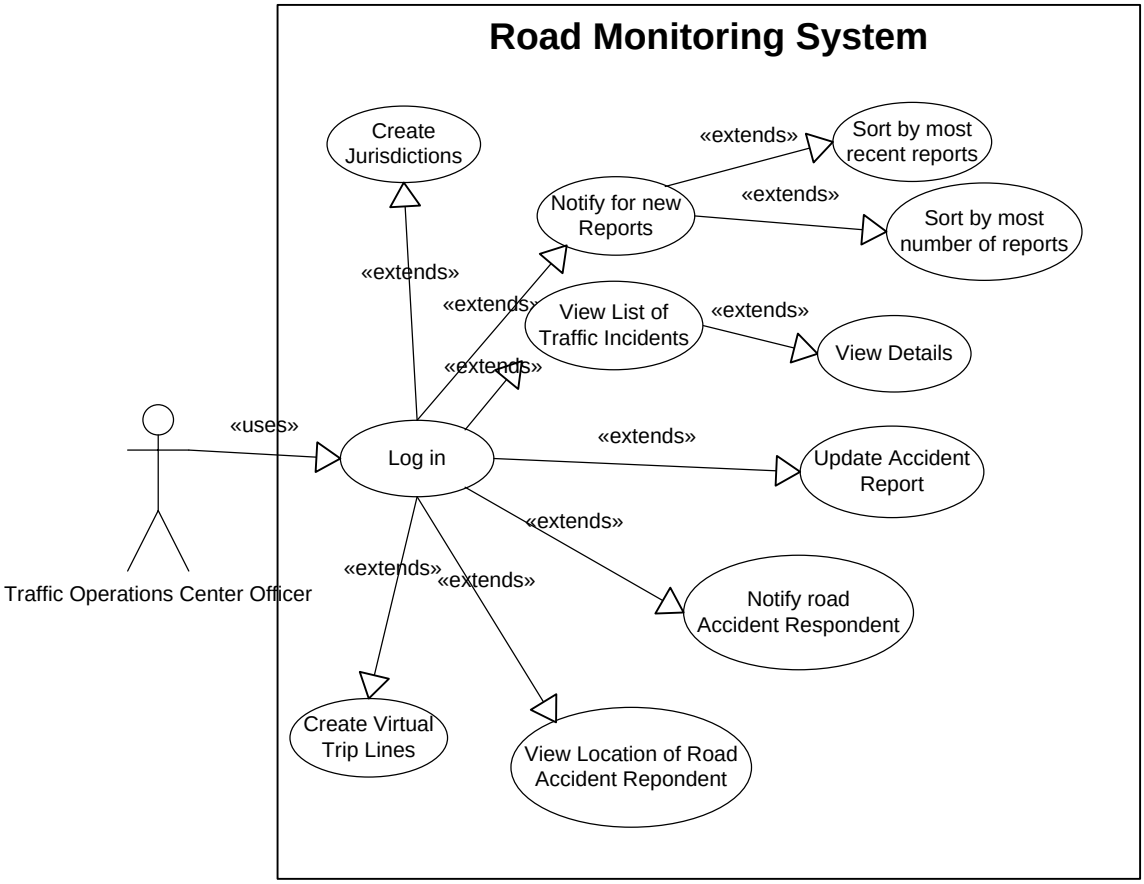


Figure 6 User Case Traffic Operations Center Officer

Traffic Investigator

Traffic Investigator are users who respond or go to the actual location of the accident and perform the necessary actions needed.

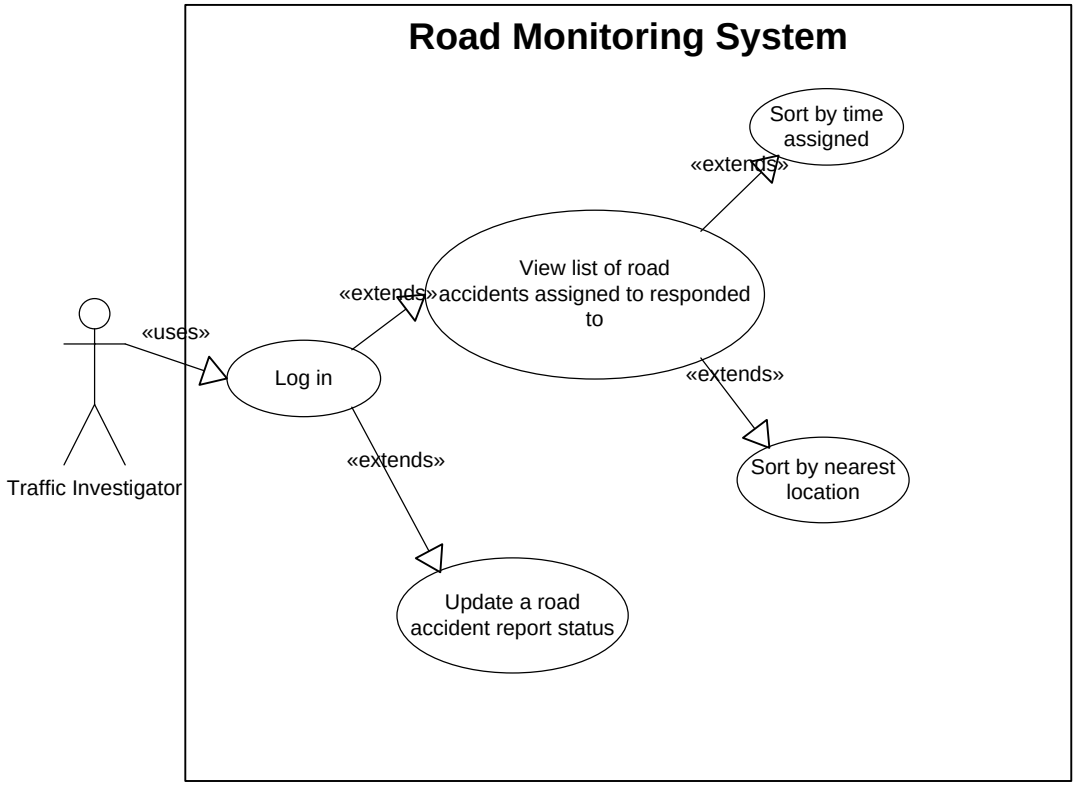


Figure 7 Use Case Traffic Investigator

D. Entity Relationship Diagram

Accounts

ra – road_accident

fs - flooded_street

rrr - road_restoration_request

os – on_site

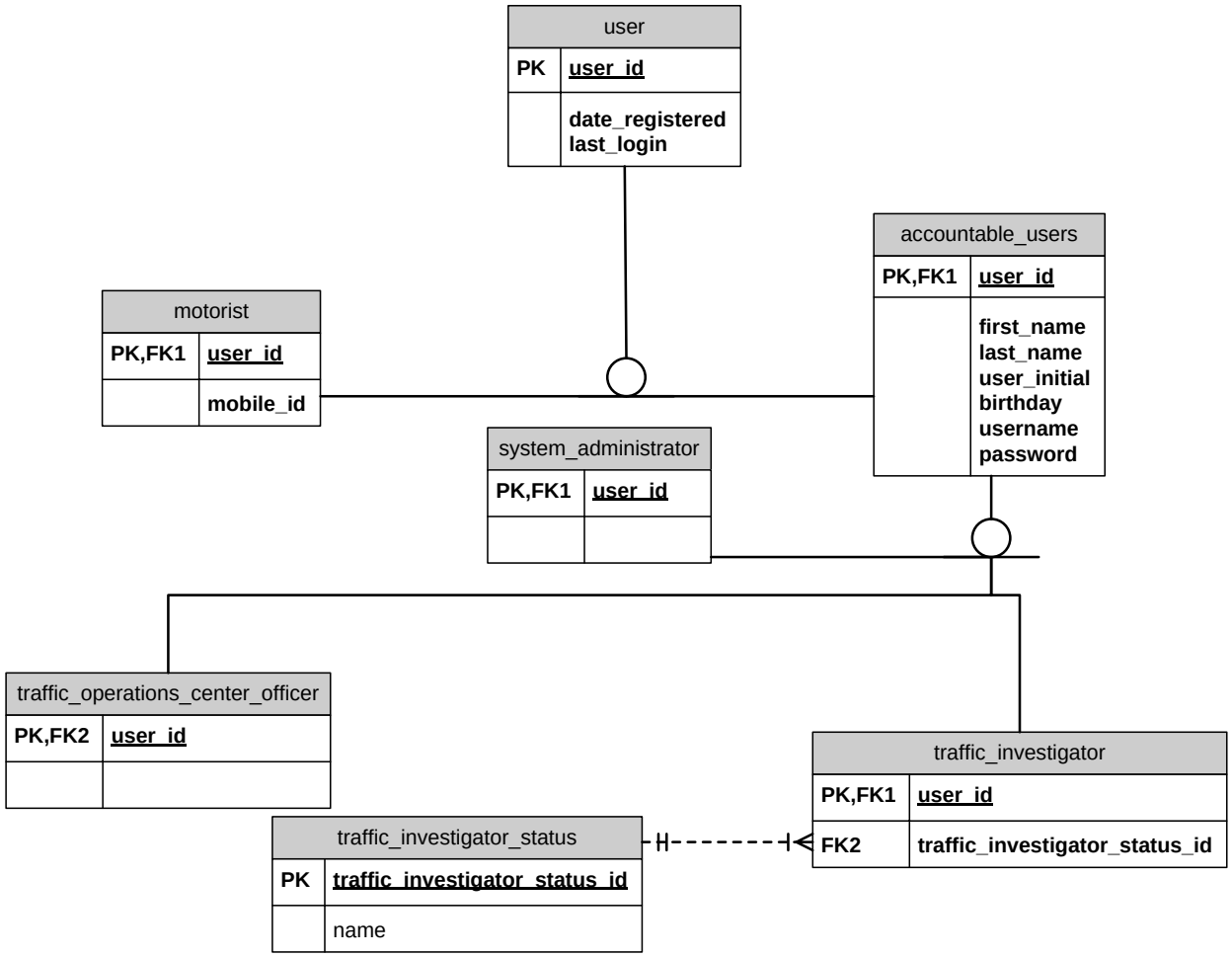


Figure 8 Entity Relationship Diagram Accounts

Report Types

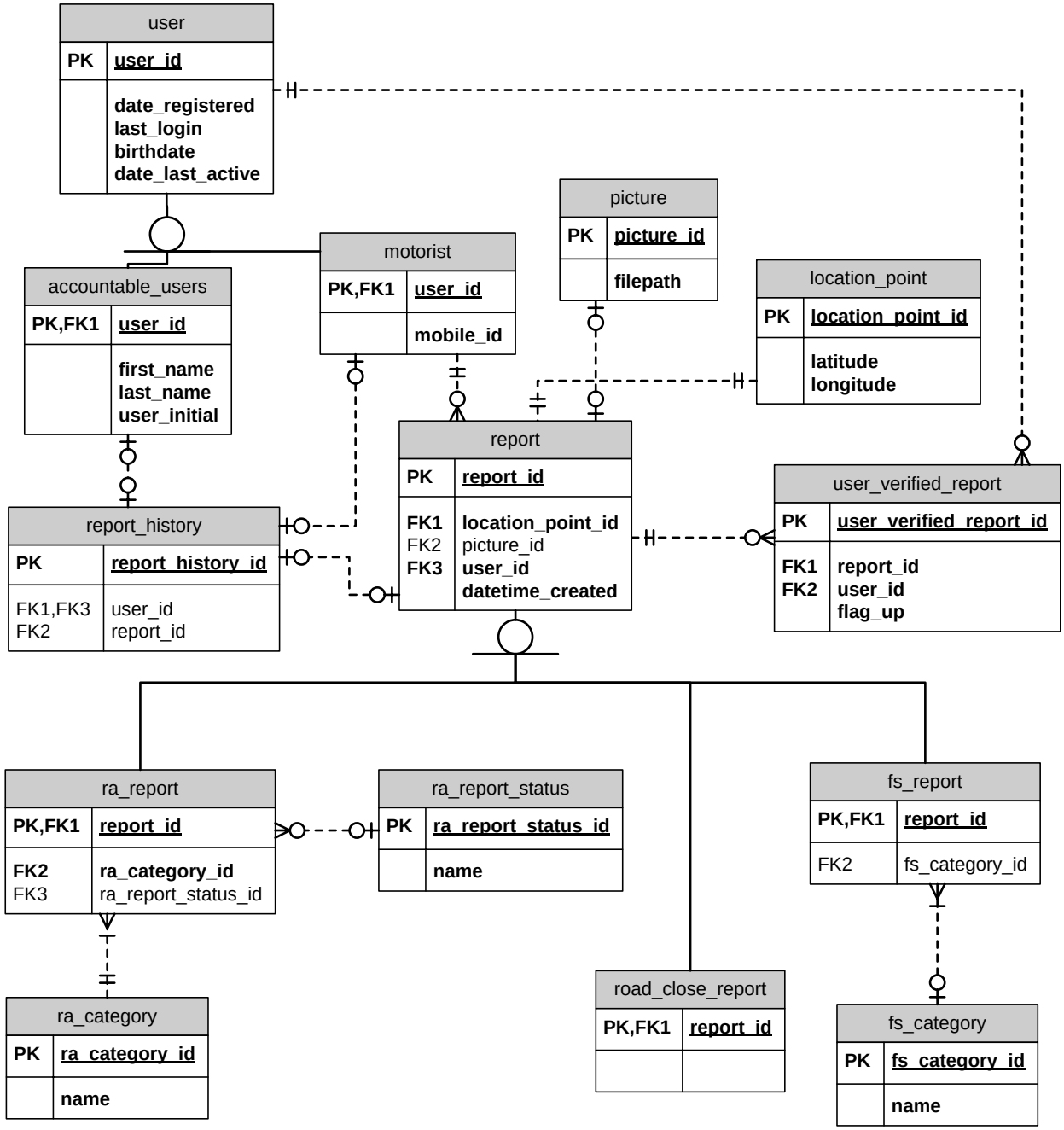


Figure 9 Entity Relationship Diagram Report Types

Traffic Data

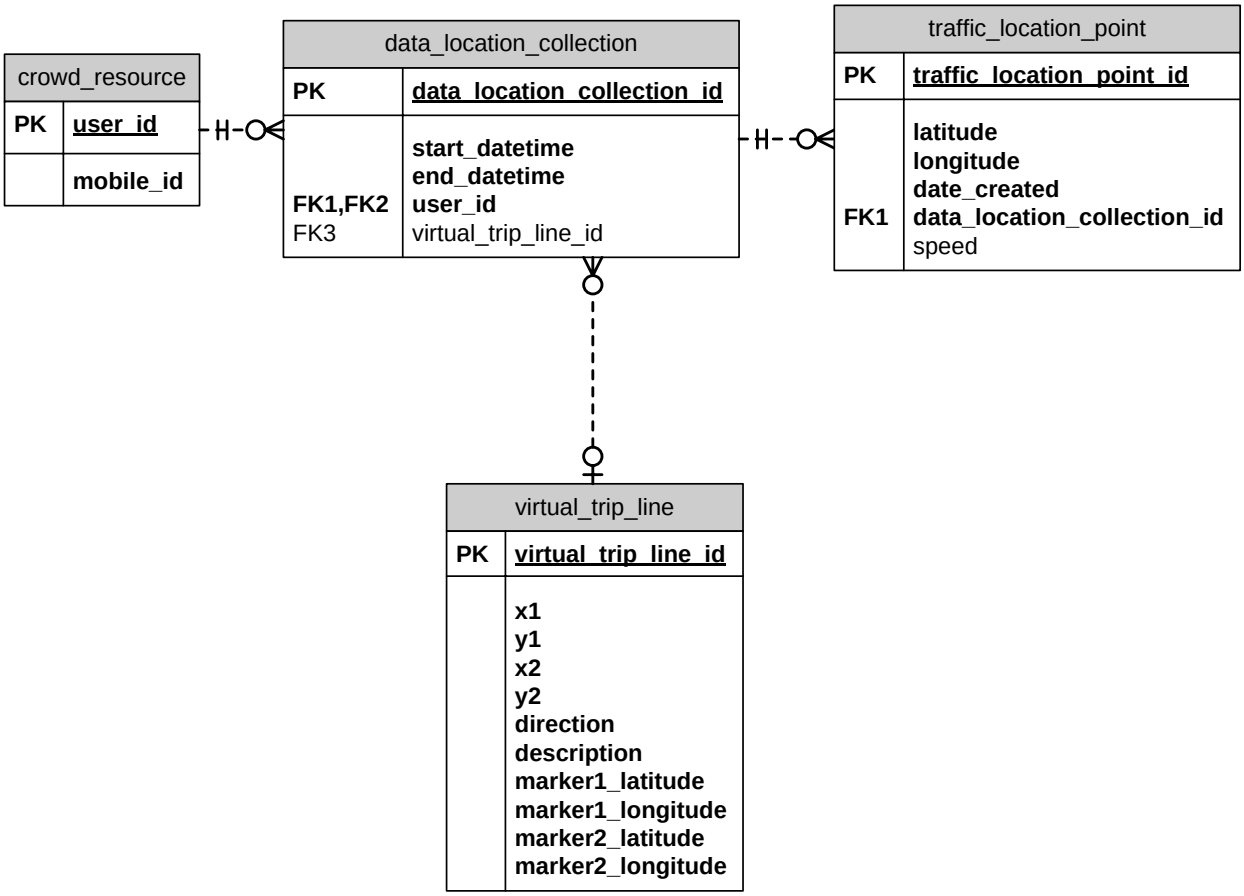


Figure 10 Entity Relationship Diagram Traffic Data

Road Accident Report

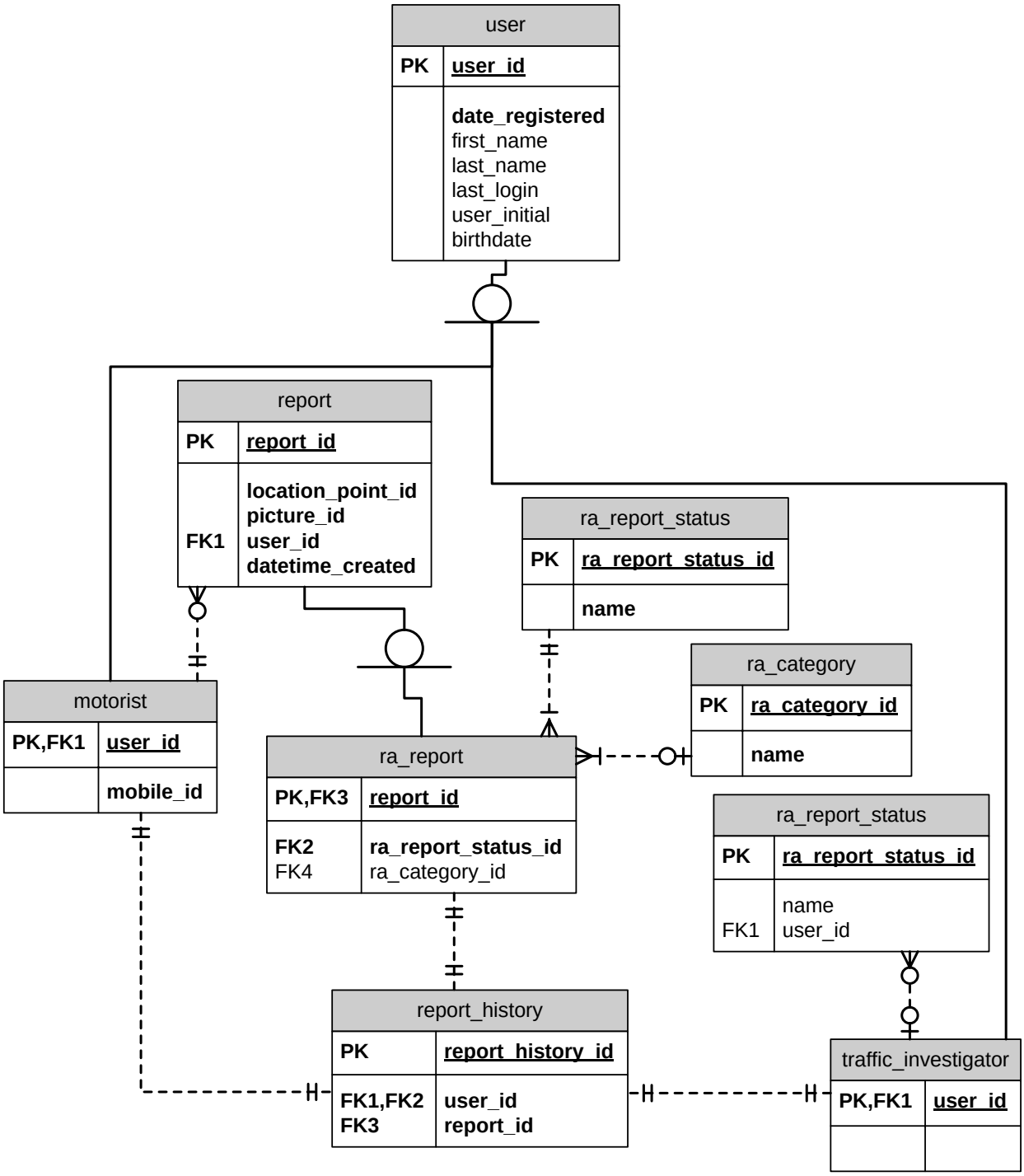


Figure 11 Entity Relationship Diagram Road Accident Report

Flood Report

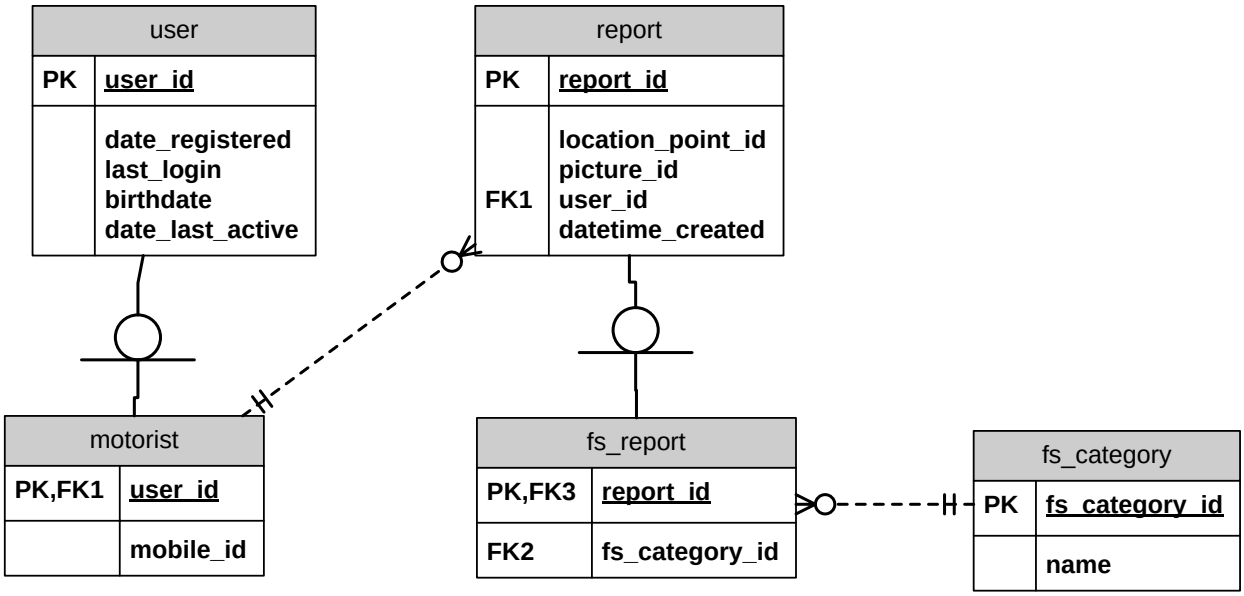


Figure 12 Entity Relationship Diagram Flooded Reports

Road Closed

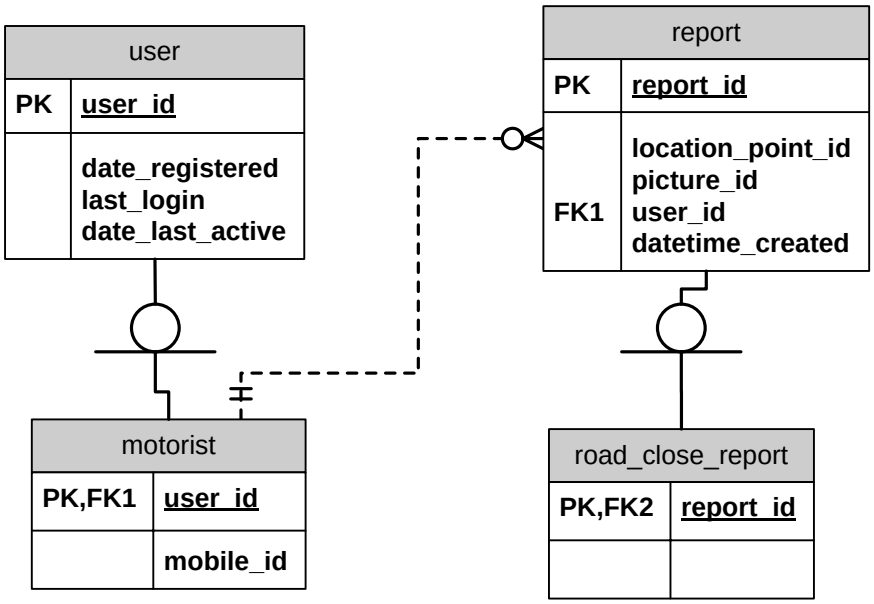


Figure 13 Entity Relationship Diagram Road Closed

Data Dictionary

Table 1 user

Field	Type	Description
user_id	INTEGER	Primary key for users
date_registered	DATETIME	Date of registration
last_login	DATETIME	Date and time when the user was last active

Table 2 motorist

Field	Type	Description
user_id	INTEGER	Foreign and primary key for crowd_resource
mobile_id	TINYTEXT	Unique identifier for mobile devices

Table 3 accountable_user

Field	Type	Description
user_id	INTEGER	Primary key for reports
first_name	TINYTEXT	First name of the user
last_name	TINYTEXT	Last name of the user
user_initial	TINYTEXT	Middle initial of the user
birthdate	DATE	birthdate of the user
username	TINYTEXT	Unique name used to login
password	TINYTEXT	Encrypted user password

Table 4 system_administrator

Field	Type	Description
user_id	INTEGER	system_administrator

Table 5 traffic_investigator

Field	Type	Description
user_id	INTEGER	Primary key for ra_respondent
rar_status_id	INTEGER	Used to indentify the status of the road accident respondent

Table 6 traffic Operations_center_officer

Field	Type	Description
user_id	INTEGER	Primary key for ra_monitoring_officer

Table 7 traffic_investigator_status

Field	Type	Description
rar_status_id	INTEGER	Primary key for ra_respondent_status
name	TINYTXT	Used to identify location of the report

Table 8 Report

Field	Type	Description
report_id	INTEGER	Primary key for reports
location_point_id	DATETIME	Foreign key. Identifies location of the report
picture_id	TINYTEXT	Foreign key. Identifies the picture attached to the report
user_id	TINYTEXT	Foreign key. Identifies who submitted the report
datetime_created	TINYTEXT	Date and time of creation of the report

Table 9 picture

Field	Type	Description
picture_id	INTEGER	Primary key for Picture
filepath	TINYTXT	contain the path where the picture is stored in the server

Table 10 location_point

Field	Type	Description
location_point_id	INTEGER	location_point
latitude	DOUBLE	coordinates of the location
longitude	DOUBLE	coordinates of the location

Table 11 report_history

Field	Type	Description
report_history_id	INTEGER	Primary key for report_history
user_id	DOUBLE	Foreign key. Indentifies the users who in involved with the report
report_id	DOUBLE	Foreign key. Identifies the report involved

Table 12 ra_report

Field	Type	Description
report_id	INTEGER	Primary key for road accident reports
ra_category_id	INTEGER	Foreign key. Category of the road accident report
ra_status_id	INTEGER	Foreign key. Current status of the road accident report

Table 13 fs_report

Field	Type	Description
report_id	INTEGER	Primary key for flooded street reports
fs_category_id	INTEGER	Foreign key. Category of the flooded street report

Table 14 road_close_report

Field	Type	Description

report_id	INTEGER	Primary key for road closed reports

Table 15 ra_category

Field	Type	Description
ra_category_id	INTEGER	Primary key for road accident categories
name	INTEGER	Category name

Table 16 ra_report_status

Field	Type	Description
ra_report_status_id	INTEGER	Primary key for road accident status
name	INTEGER	Status name

Table 17 fs_category

Field	Type	Description
fs_category_id	INTEGER	Primary key for flooded street categories
name	INTEGER	Category name

Table 18 data_location_collection

Field	Type	Description
data_location_collection_id	INTEGER	Primary key for flooded street categories
start_datetime	DATETIME	Data and time of recording data
end_datetime	DATETIME	Data and time of recording data

trip_line_zone_id	INTEGER	Foreign key. Area of recording data
user_id	INTEGER	Foreign key. Source of data

Table 19 trip_line_zone

Field	Type	Description
trip_line_zone	INTEGER	Primary key for flooded street categories
lower_right_boundary_point_id	INTEGER	boundaries of the trip line
upper_right_boundary_point_id	INTEGER	boundaries of the trip line
lower_left_boundary_point_id	INTEGER	boundaries of the trip line
upper_left_boundary_point_id	INTEGER	boundaries of the trip line

Context Diagram

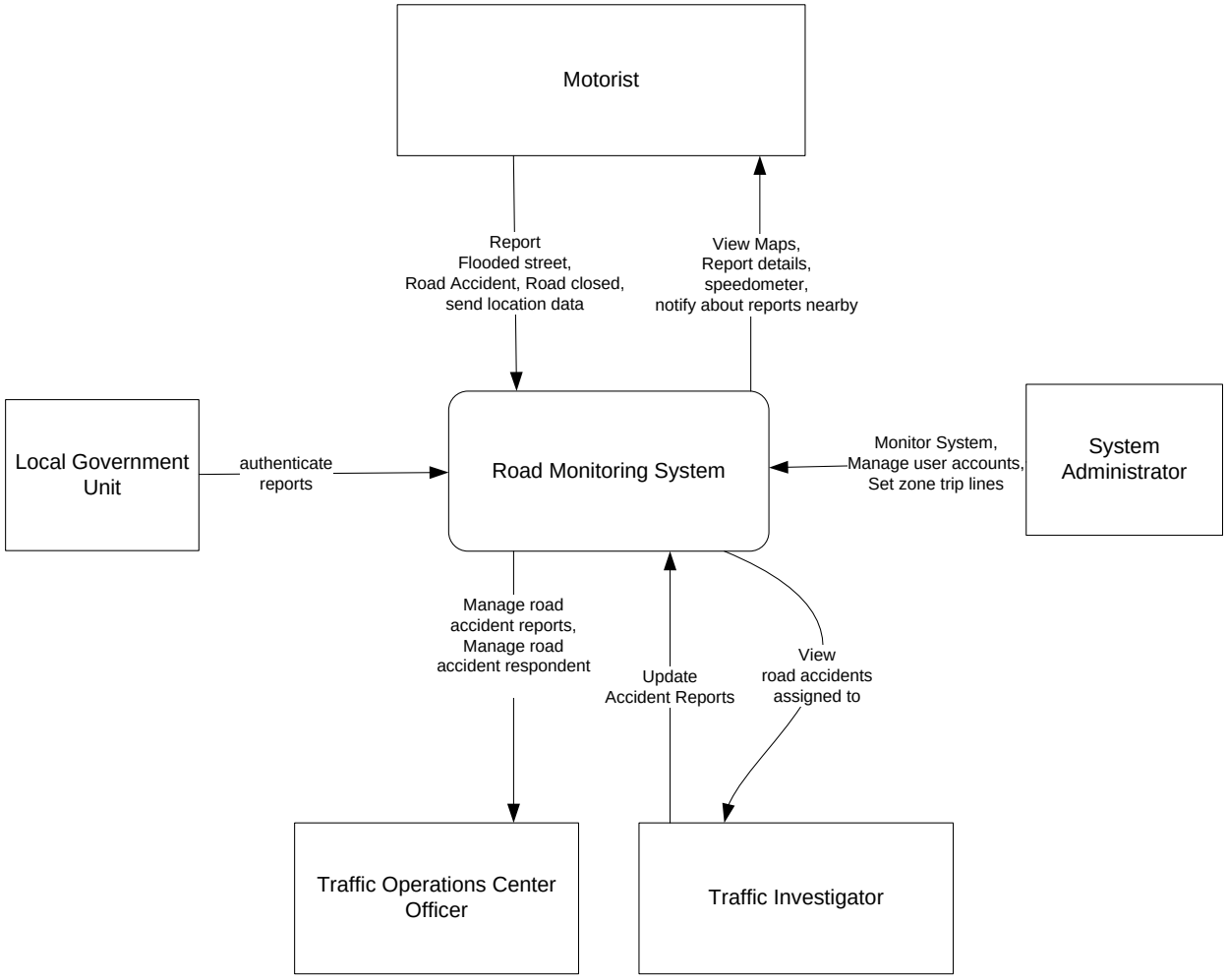


Figure 14 Context Diagram

Reports will be classified according to

- Flooded Streets
- Road Accidents
- Closed Roads

In addition specific reports will be classified according to the nature of the report

Flooded streets will be classified according to

- Not passable to light vehicles
- Not passable to all type of vehicles

Road Accident will be classified to

- Car Collision
- Collision with animals
- Rollover
- Run-off-road collisions
- Accidents involving pedestrian

Top Level Process (Motorist)

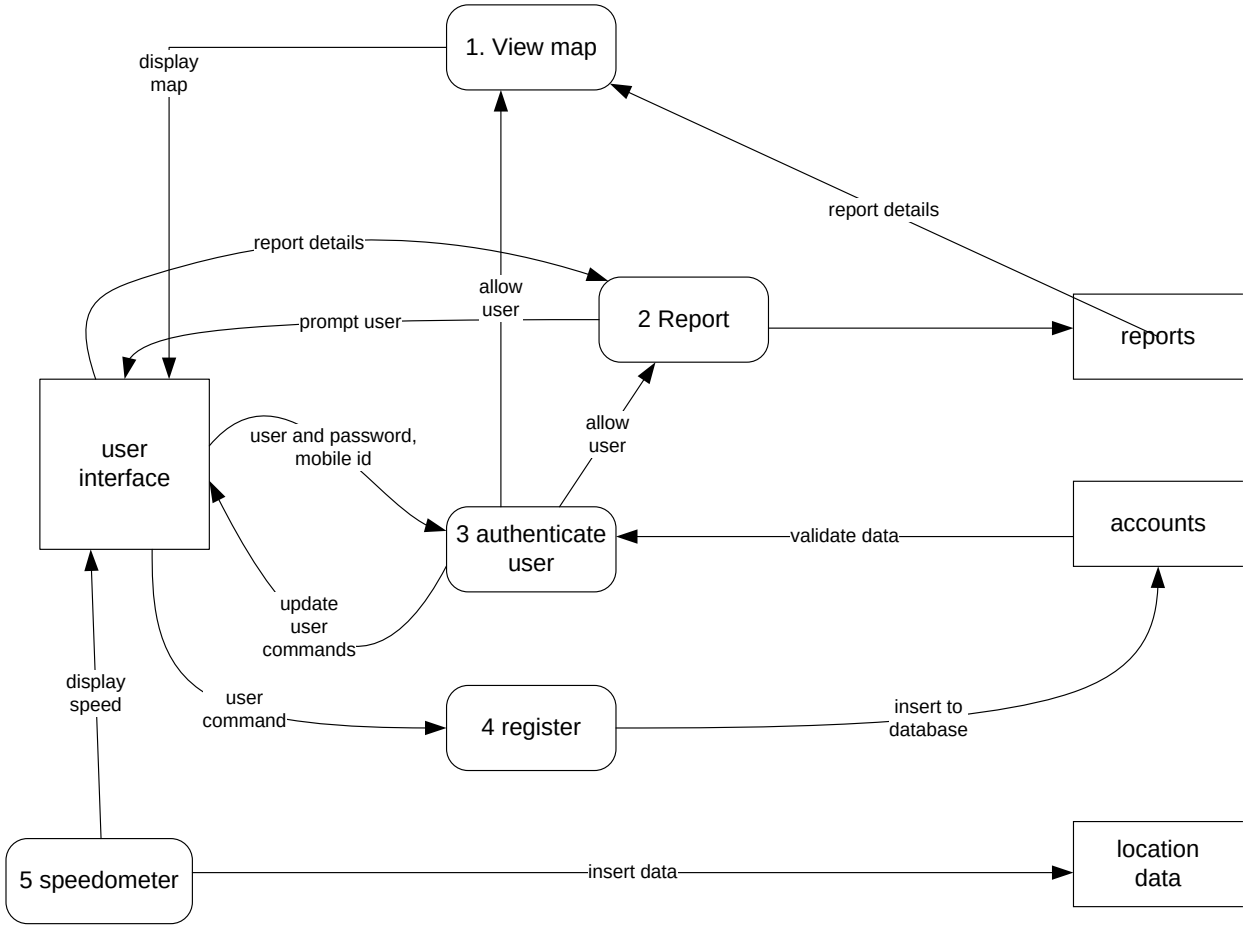
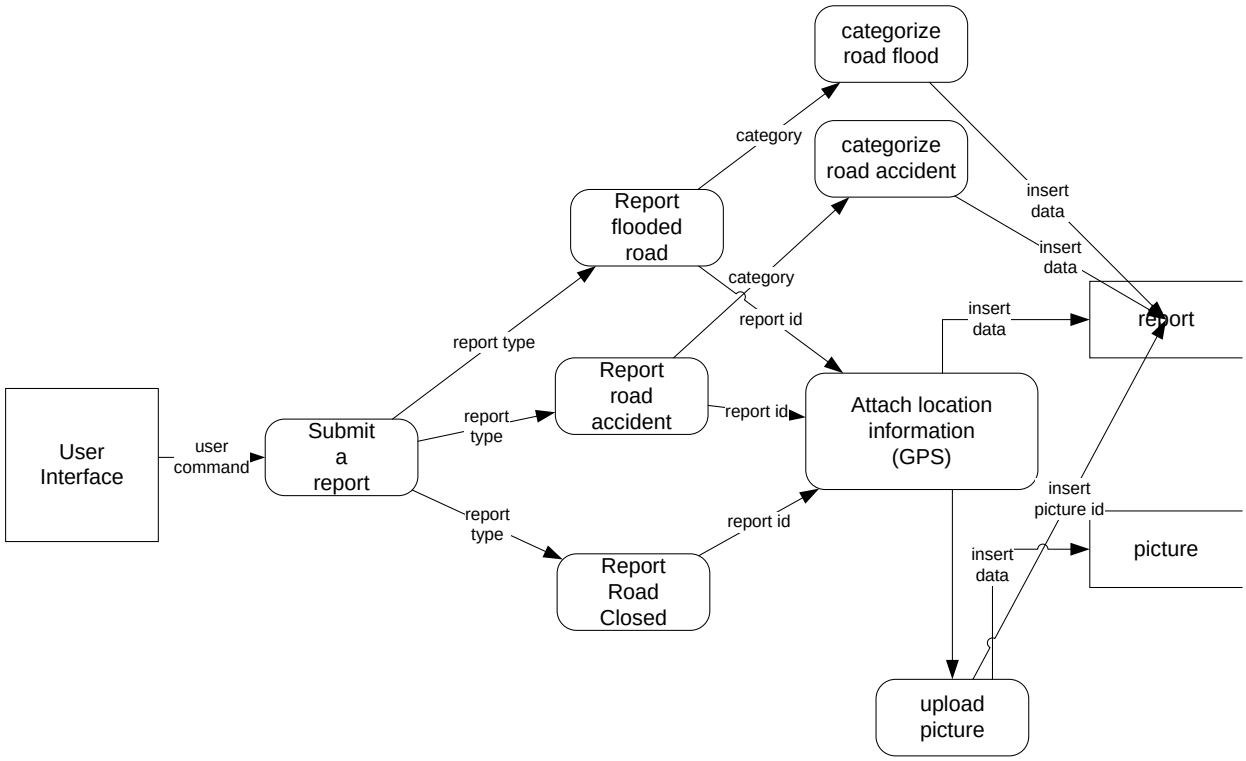


Figure 15 Top Level Process Motorist

2 Report sub explosion



3 authenticate user sub explosion

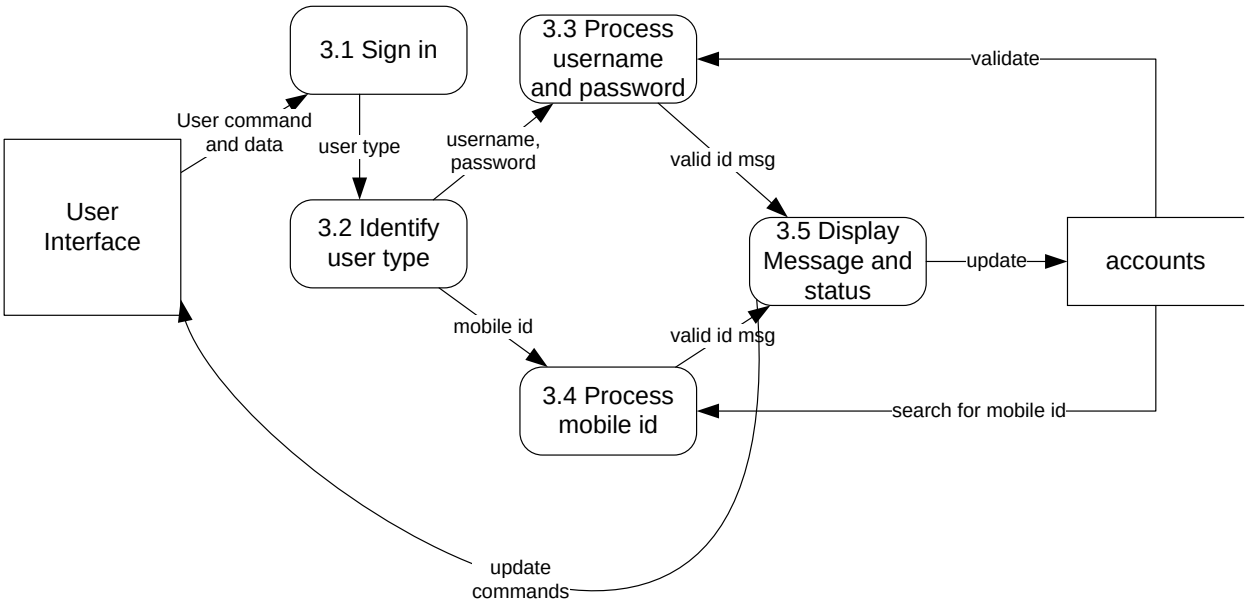


Figure 16 user authentication

Top Level Process (Traffic)

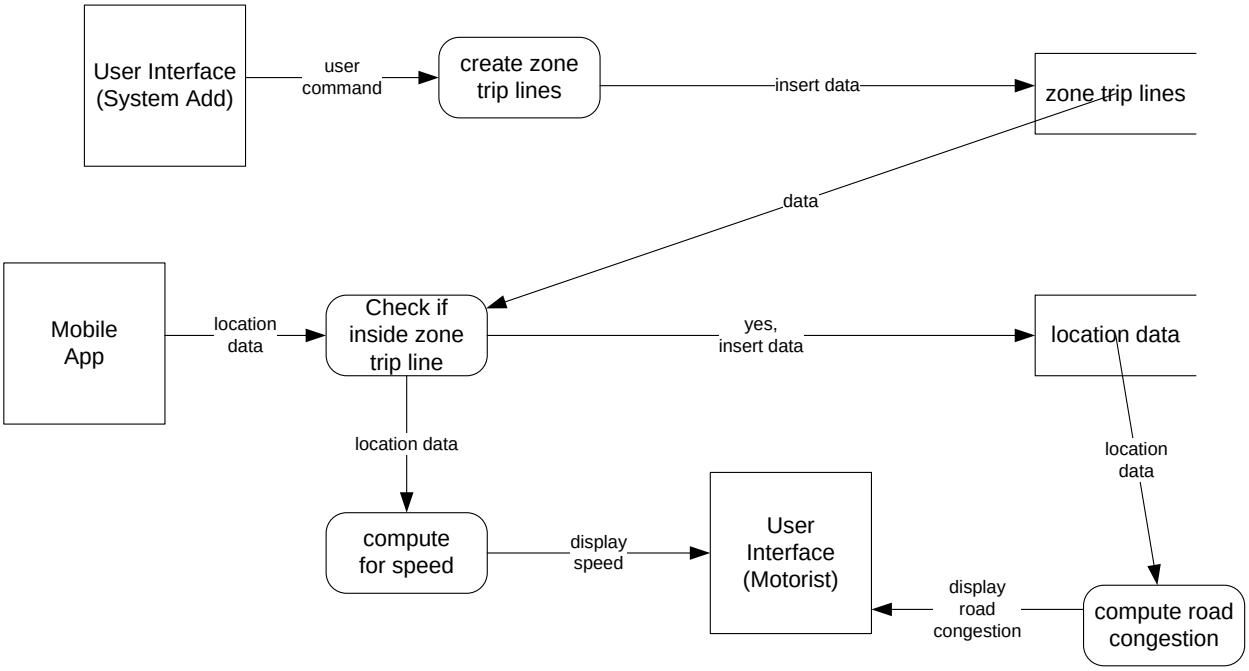


Figure 17 Top Level Process Traffic

Top Level Process (Traffic Investigator)

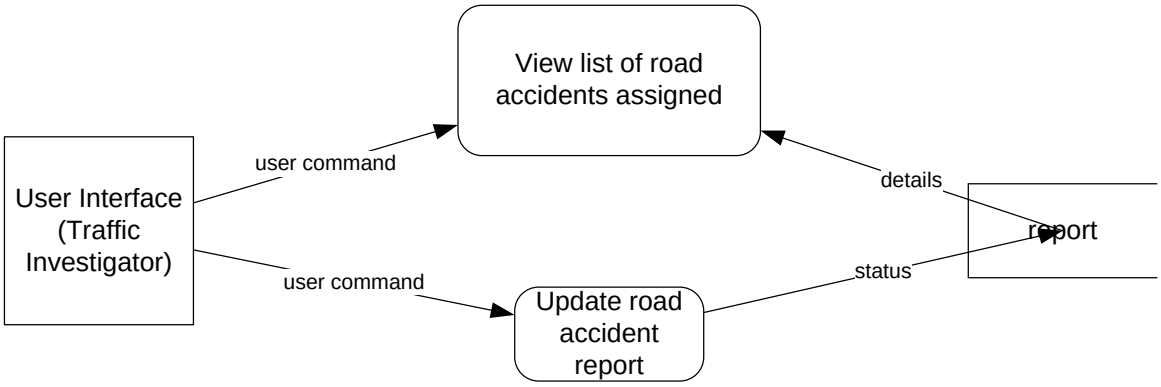


Figure 18 Top Level Process Traffic Investigator

Top Level Process (System Administrator)

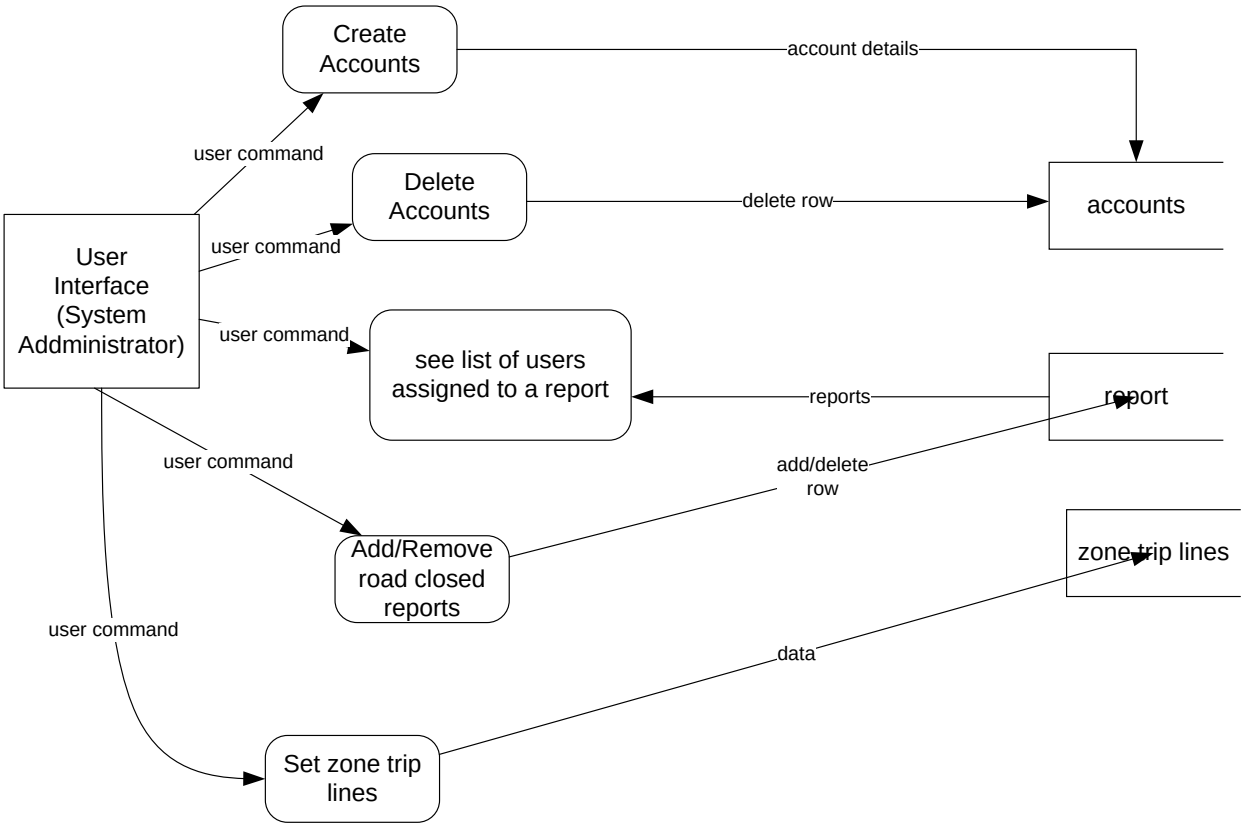


Figure 19 Top Level Process System Administrator

Top Level Process (Road Accident Monitoring Officer)

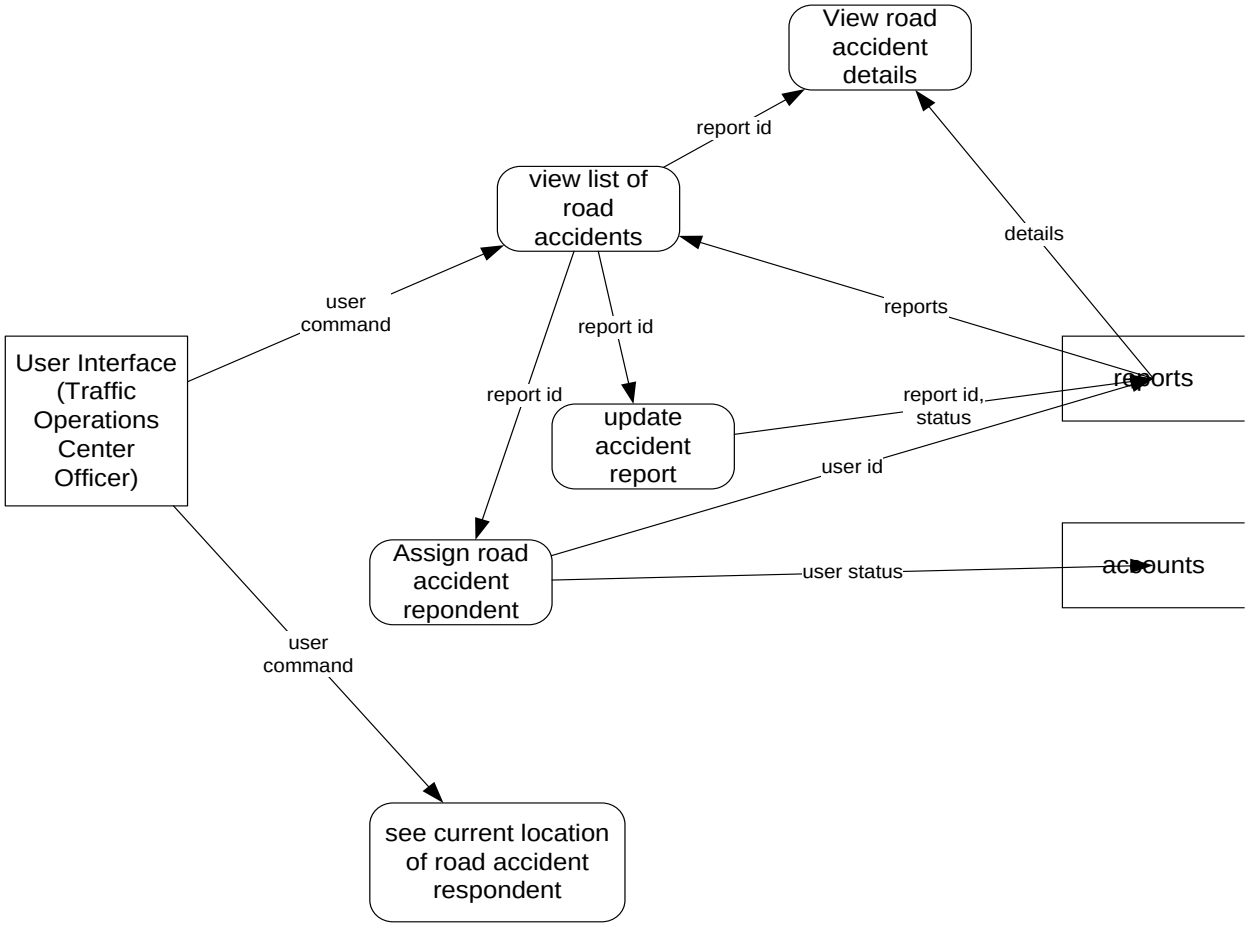


Figure 20 Top Level Process Traffic Operations Center Officer

V. Results

The system that has been developed is composed of three major component: a mobile application for Motorists, a mobile application for traffic investigators and a web application for traffic operations center officers and system administrators. The purpose of the mobile application for motorist is to gather traffic data and traffic incidents and share this information with fellow users. Further, the purpose of the mobile application for traffic investigators is to get the necessary information regarding road accidents as well as update the status of road accidents reports to notify the users of the system. The web application for traffic operations center officers and system administrator is mainly used to maintain accuracy of reports, assign traffic investigators to a road accident, manage accounts and traffic incidents as well as input necessary data for the whole system to function.

A. Mobile Application for Motorist

The three primary features of this component are (1) the Map Tab which contains map with all the traffic incidents and traffic data; (2) the Report Tab which allow the user to input reports to the database and (3) background services that updates and checks the database.

Map Tab

This tab contains all the information regarding a report as well as the corresponding location on the map.

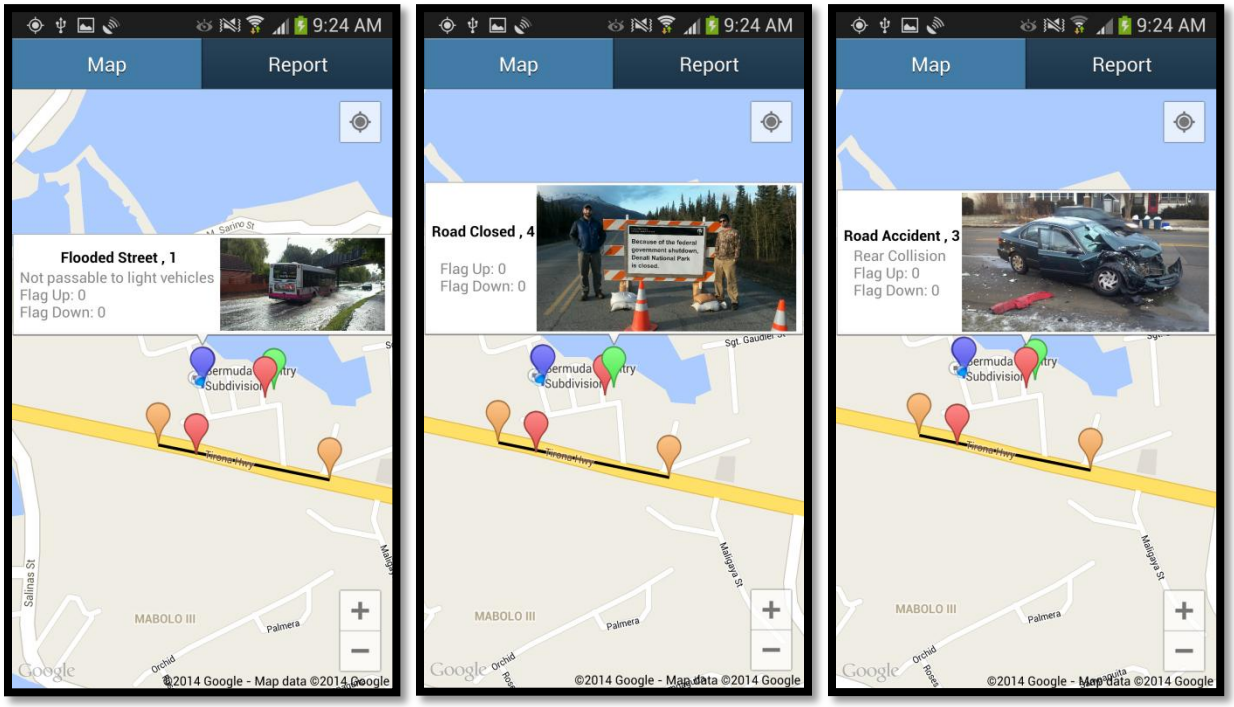


Figure 21 Traffic Incident Map

Report Tab

This tab allows the user to input traffic incident reports to the database

Figure 22 Reports Menu

Clicking this icon opens a new window to add road accident report to the database

Clicking this icon opens a new window to add road close report to the database

Clicking this icon opens a new window to add flooded street report to the database

The icons above open the corresponding windows and asks the user to input the necessary details to insert data in the database

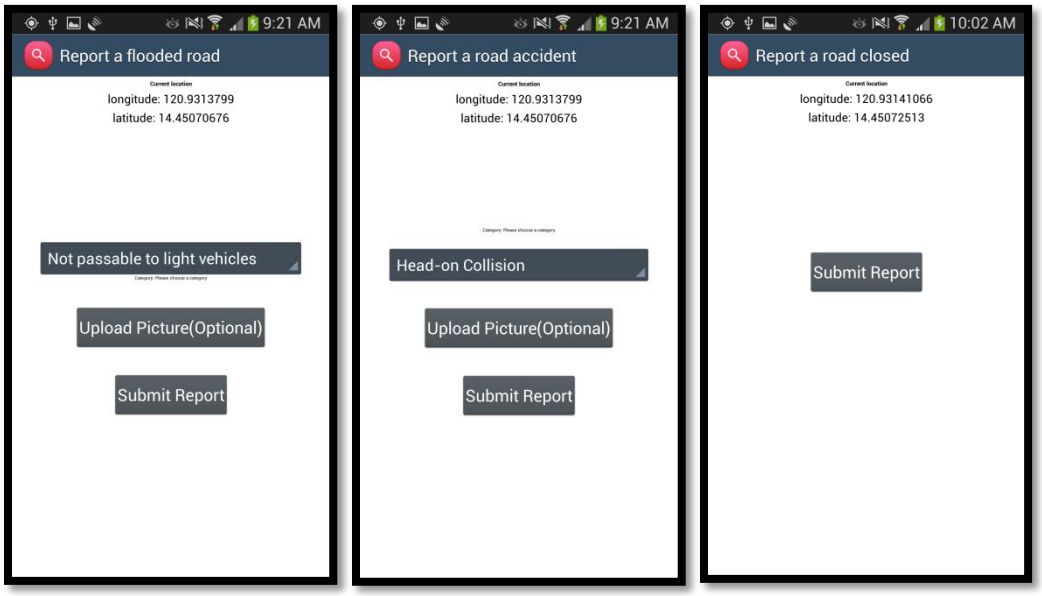


Figure 23 Report Window

Background Services

There are three background services that runs ones the application starts. These are (1) GPS service that continually checks for latitude and longitude points; (2) verify reports service which checks for nearby reports to verify; and (3) the VTL service which checks if the user is inside a virtual trip line.

If the VTL service detects that the user is inside a virtual trip line, the system starts collecting location and speed data from the GPS receiver of the phone. It then continues checking if the user is still on the virtual trip line through the collection of data. Once the user is outside the virtual trip line, the system stops collecting location and speed data and repeat checking if the user enters a trip line.

If the verify report service detects a nearby report to verify, it opens a window asking the user to verify the respective report. Note that it only detect reports that are not verified by traffic operation center officers and is still within the timeframe for which a report is considered valid.

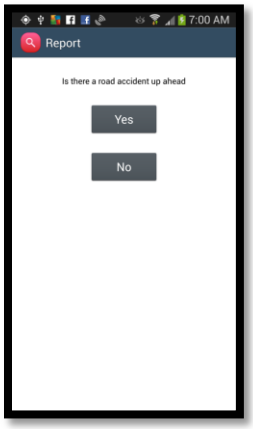


Figure 24 Verification Window

B. Mobile Application for Traffic Investigator

Once the traffic investigator app is opened, the user is asked to log in.

Once successfully authenticated the database records that the traffic investigator is online and would be available to be assigned on road accidents

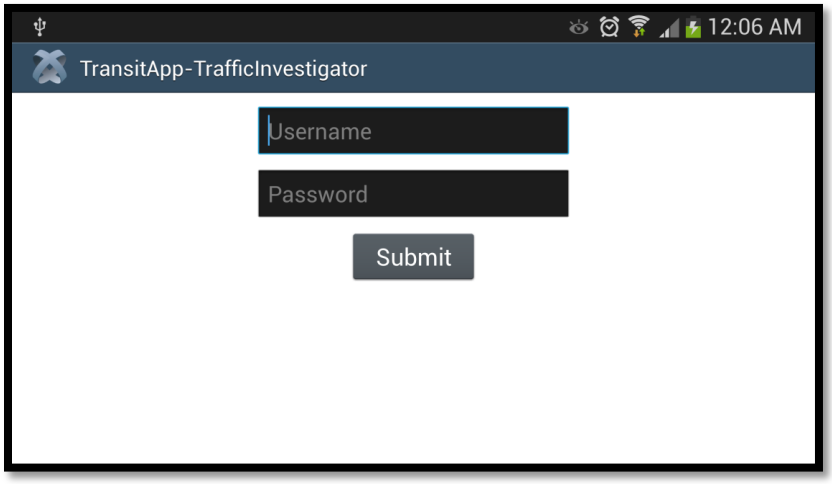


Figure 25 Login Window

Once successfully authenticated, the user is prompted with a menu allowing him to:

1. View assigned road accidents to the traffic investigator
2. View road accidents assigned to the users on the map
3. Logout

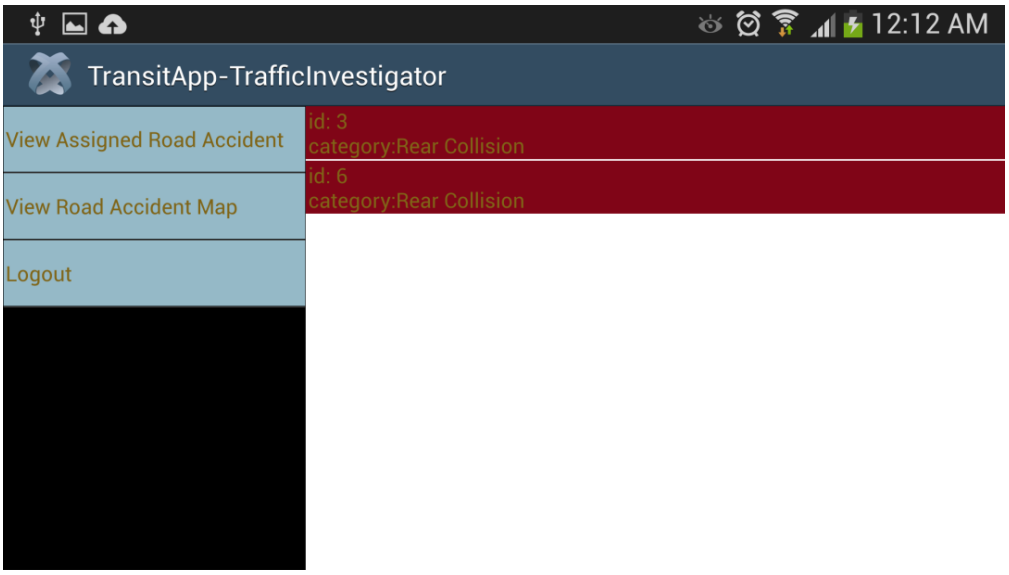


Figure 26 List of road assigned to traffic investigator

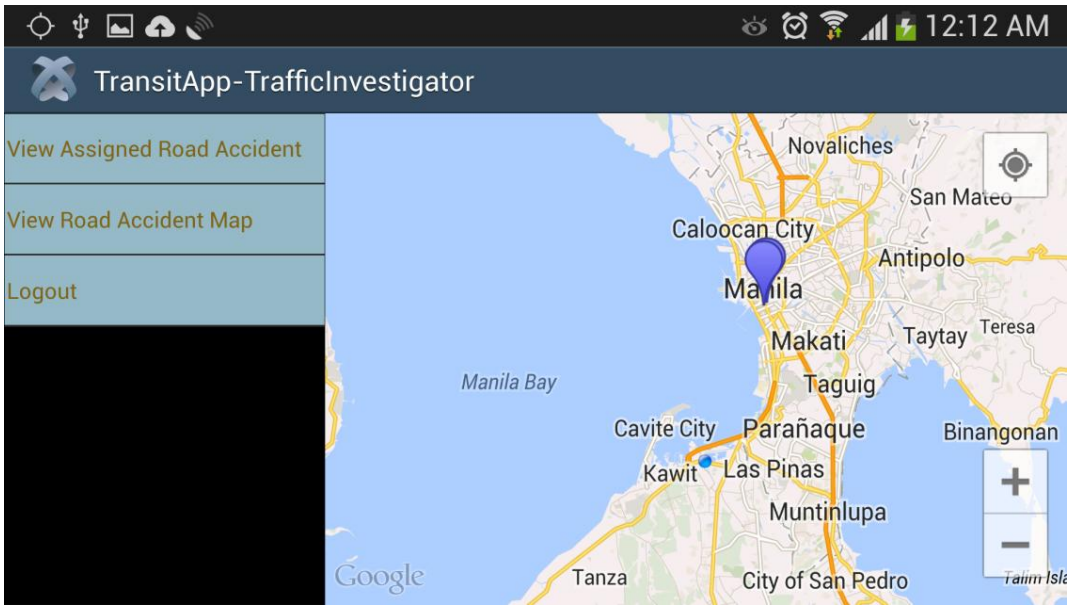


Figure 27 Road Accident Map

Update a road accident report by changing to the appropriate road status as follows:

a. Responding

b. Resolved

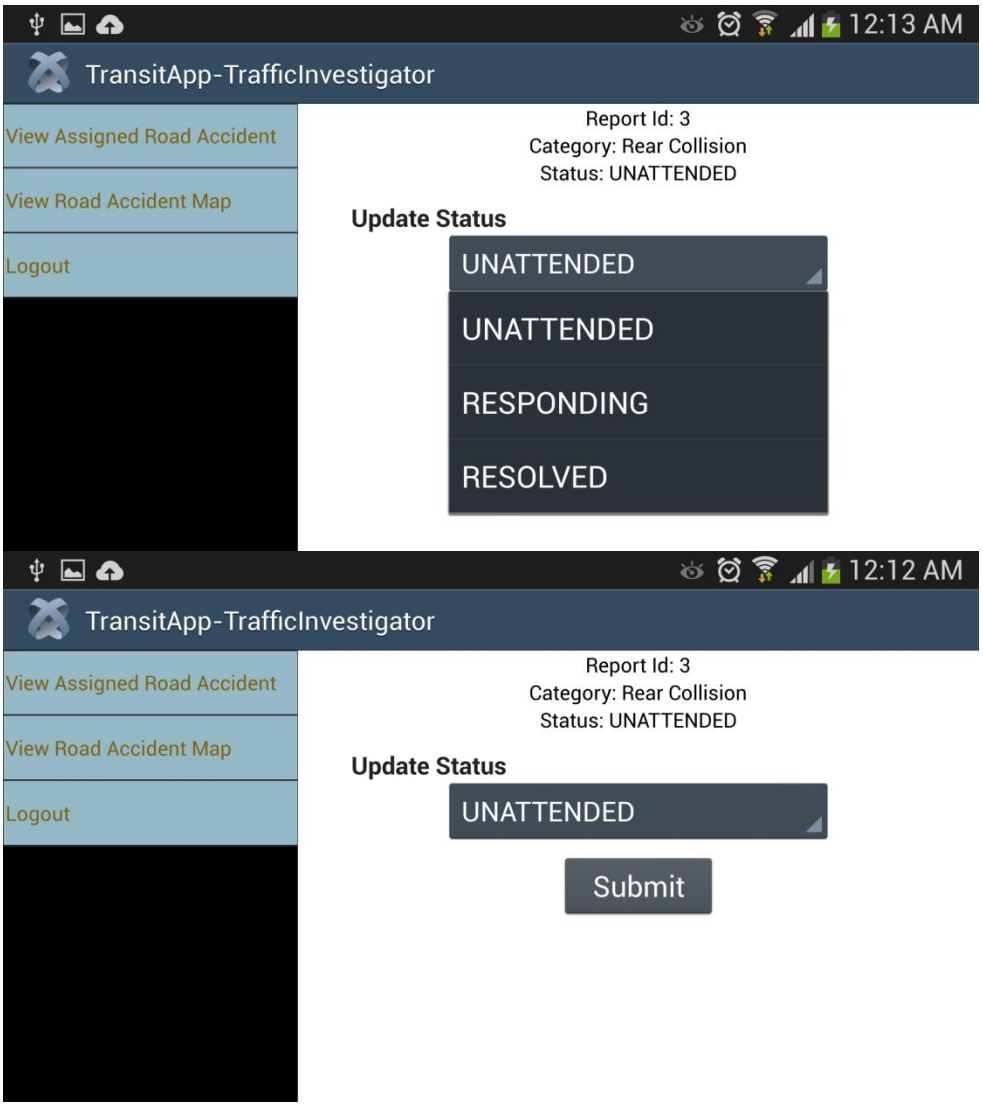


Figure 28 Update Road Accident Report

If the user logouts, the system updates the database and won't let traffic operation center officer assign them to a road accident.

C. Web Application for traffic operations centre officer and system administrator

Traffic Operations Centre Officer

View the location of road accident, road closed and flooded streets on a map and a list. The detail of road accidents, road closed and flooded streets may be viewed by clicking on the list provided at the bottom. Road accident, road closed or flooded can also be created on this view found on the side menu.

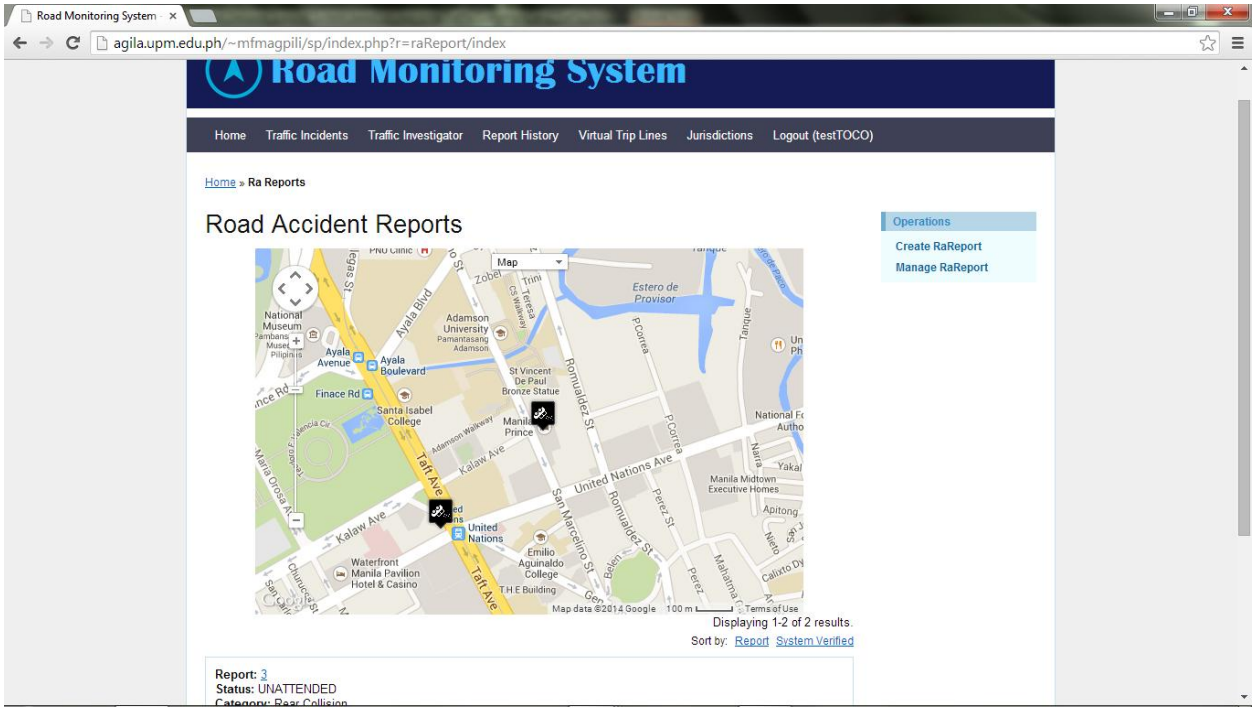


Figure 29 Road Accident Report Map

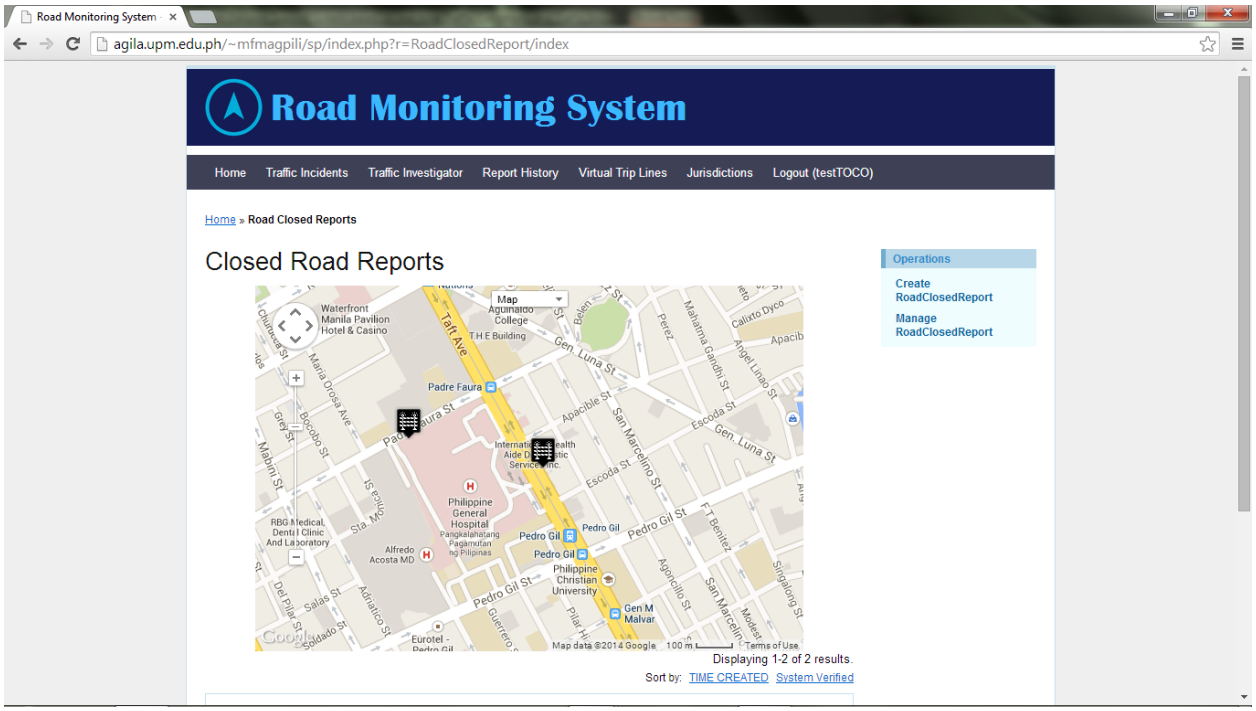


Figure 30 Closed Road Report Map

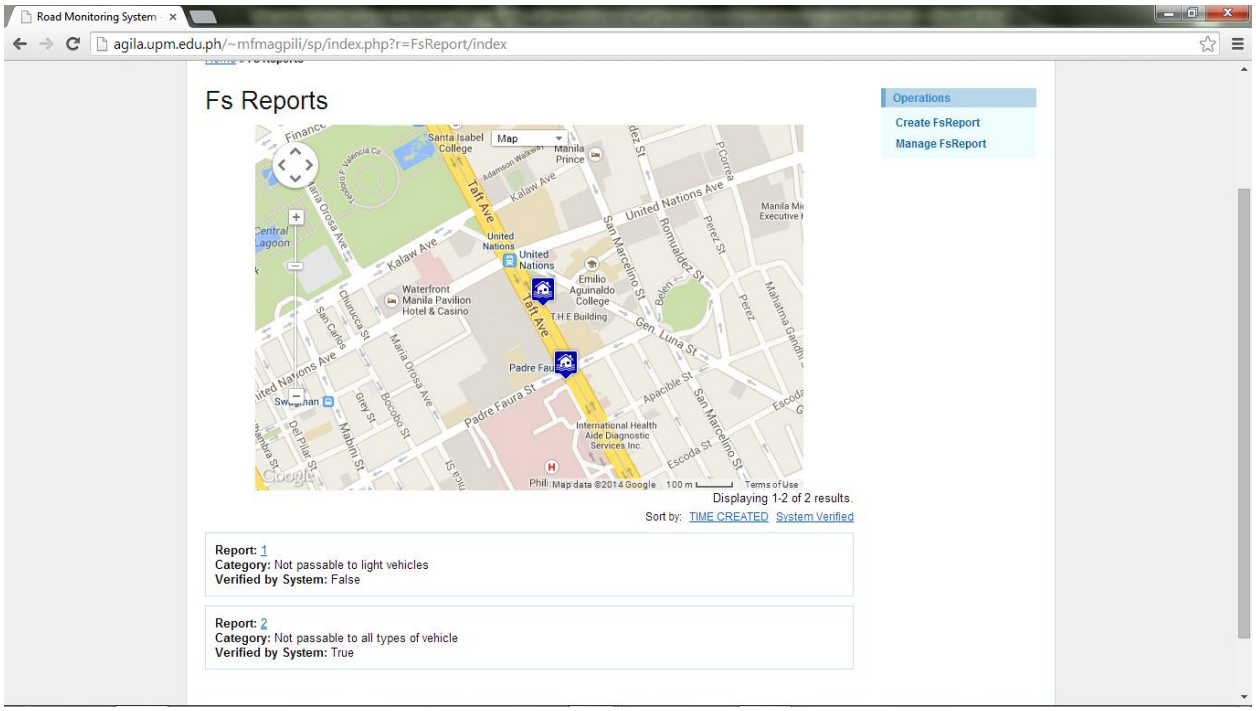


Figure 31 Flooded Street Report Map

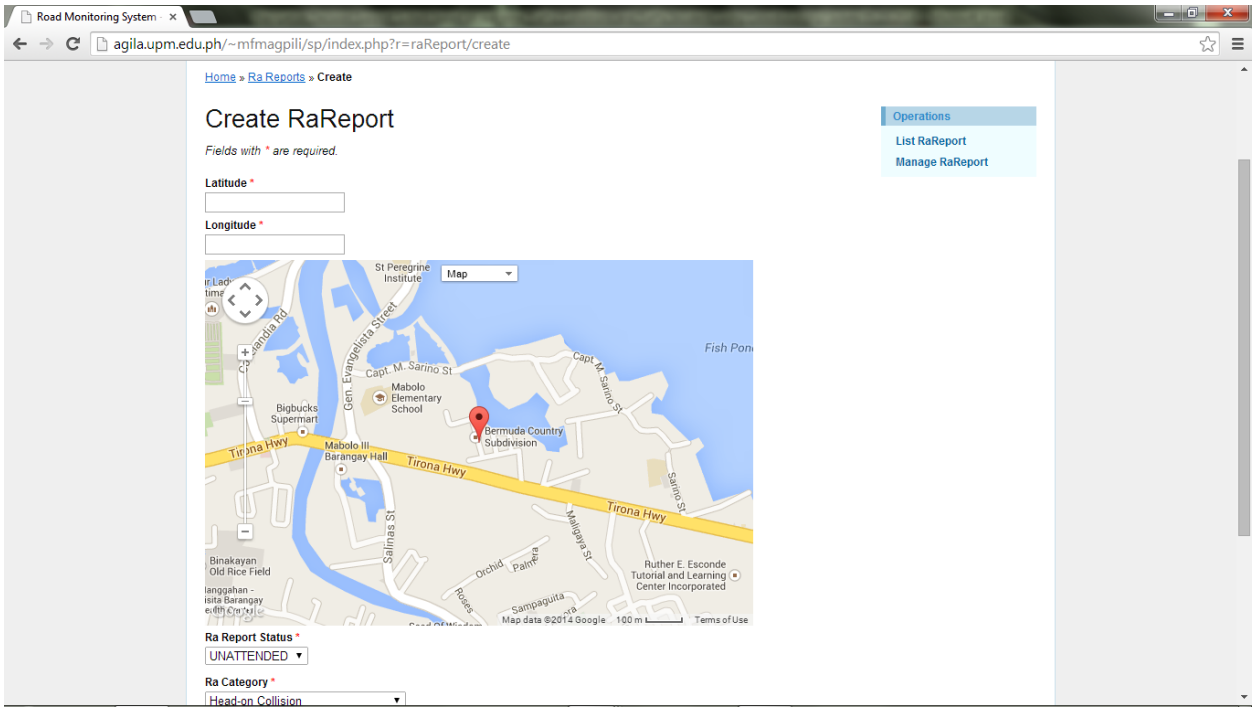


Figure 32 Road Accident Form

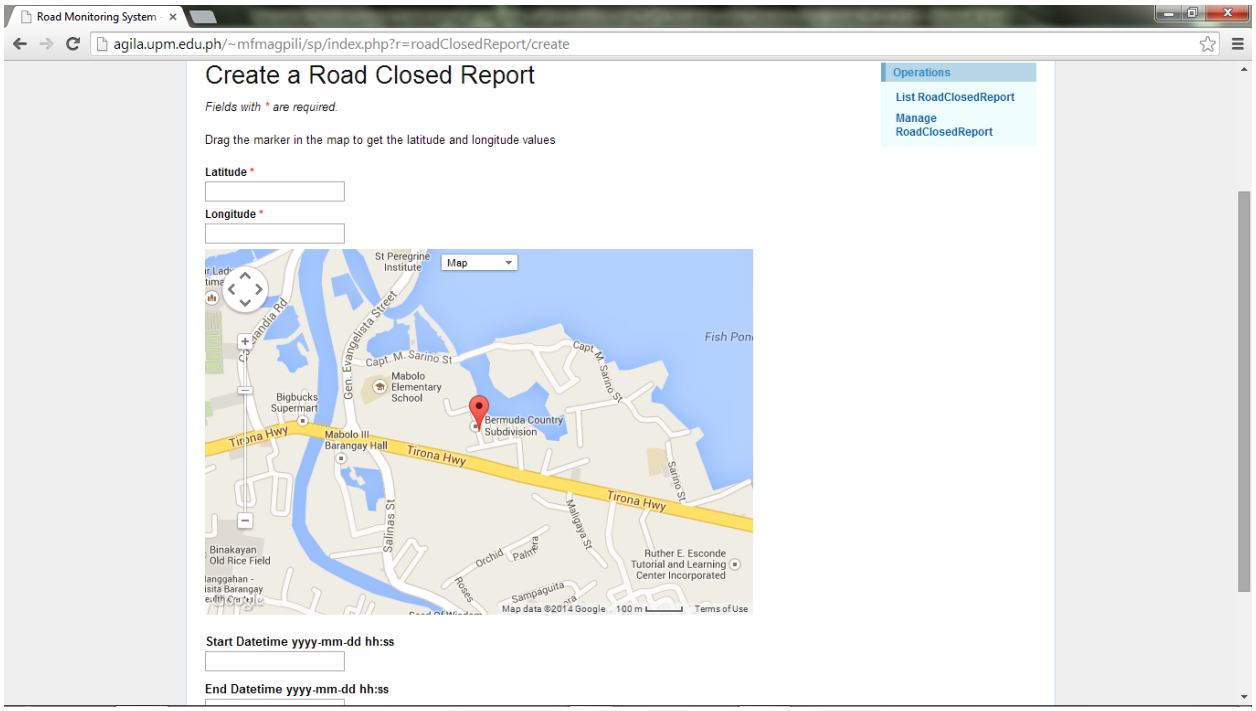


Figure 33 Road Closed Form

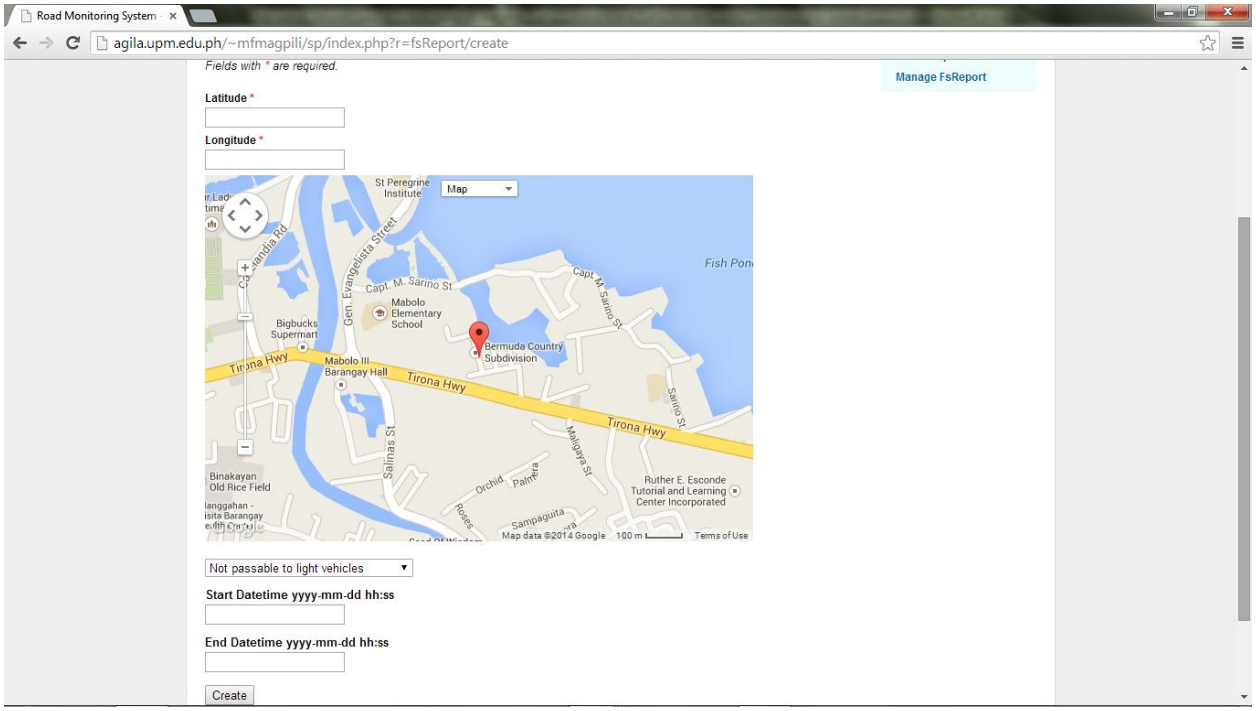


Figure 34 Flooded Street Form

Once the report id is clicked on the list generated by the system, the detail regarding a road accident, road closed or flooded street is shown to the user. The user may also verify, update or delete a report on the side menu. In the special case of road accident, there is an added option of

assigning a traffic investigato

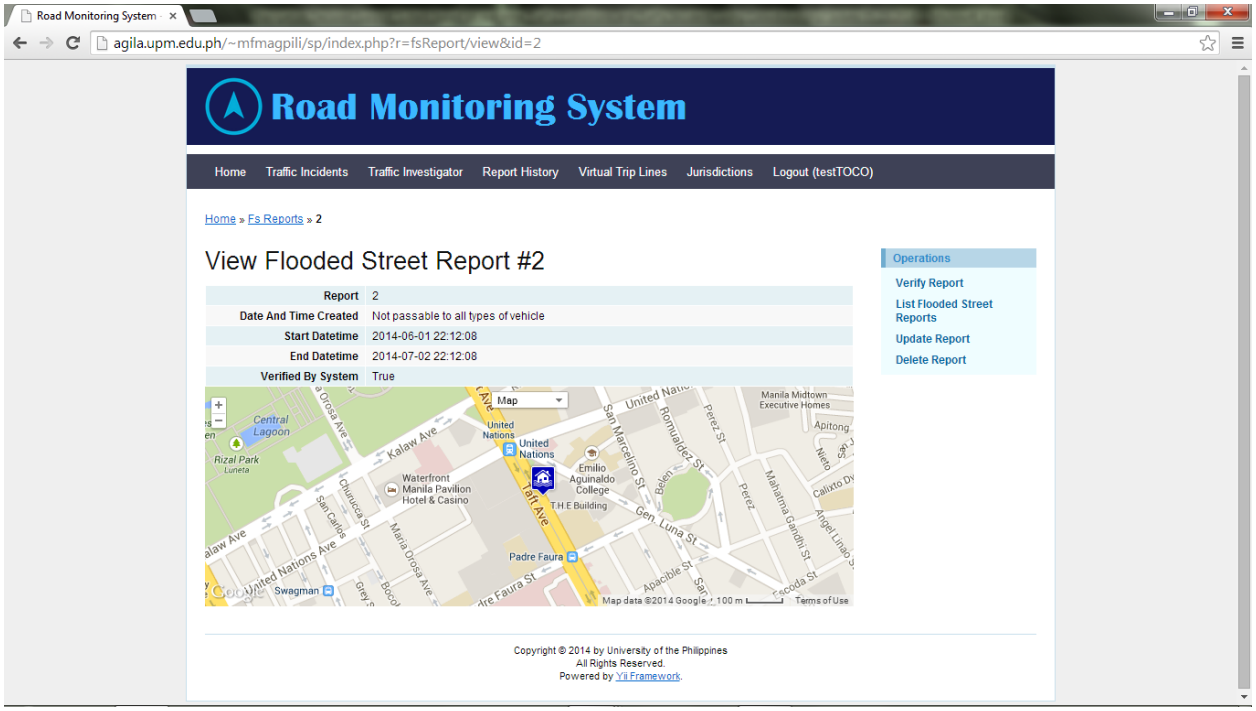


Figure 35 Flooded Street View

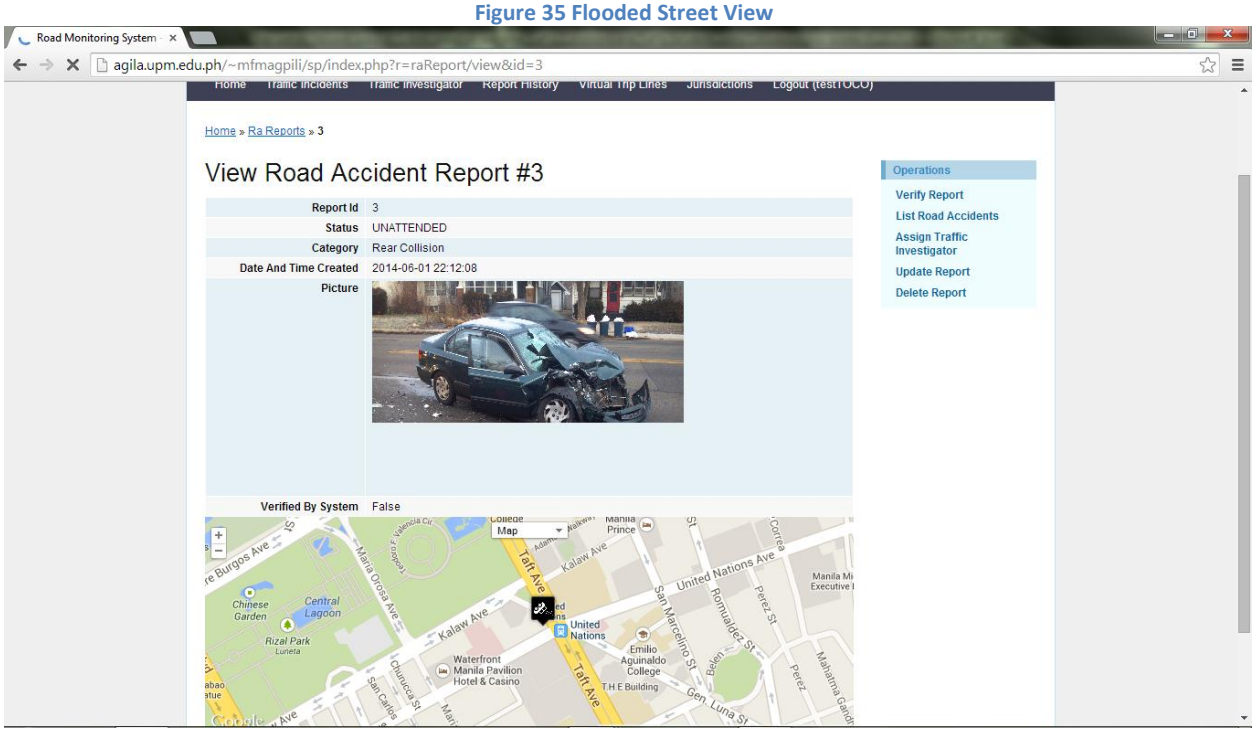


Figure 36Road Accident View

Location of the traffic investigator is shown in the map together with a list of all traffic investigators below the map. Details of the traffic investigator will be shown once opened by the user on list

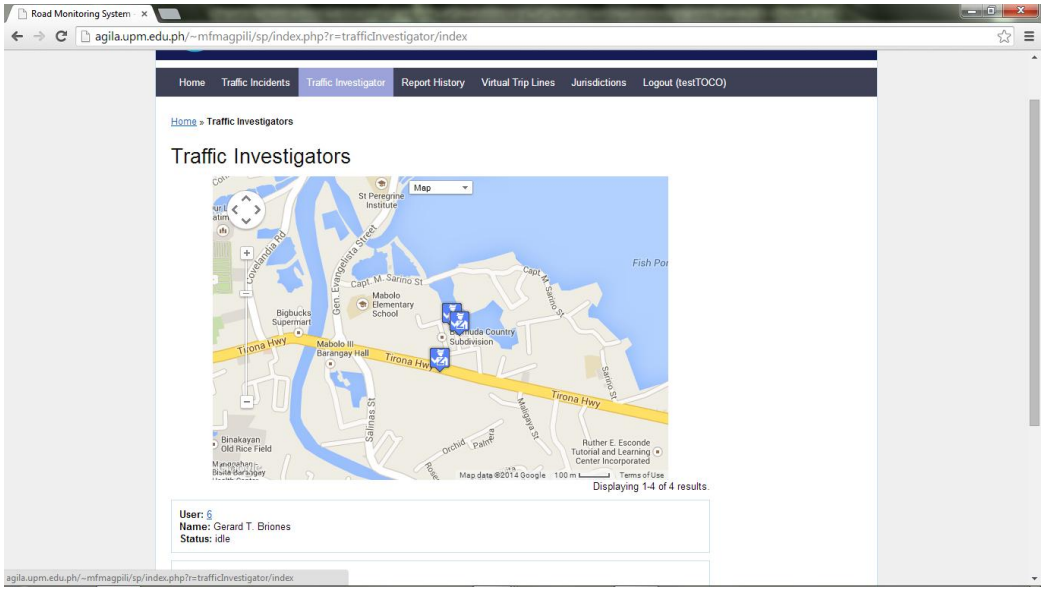


Figure 37 Traffic Investigator Map

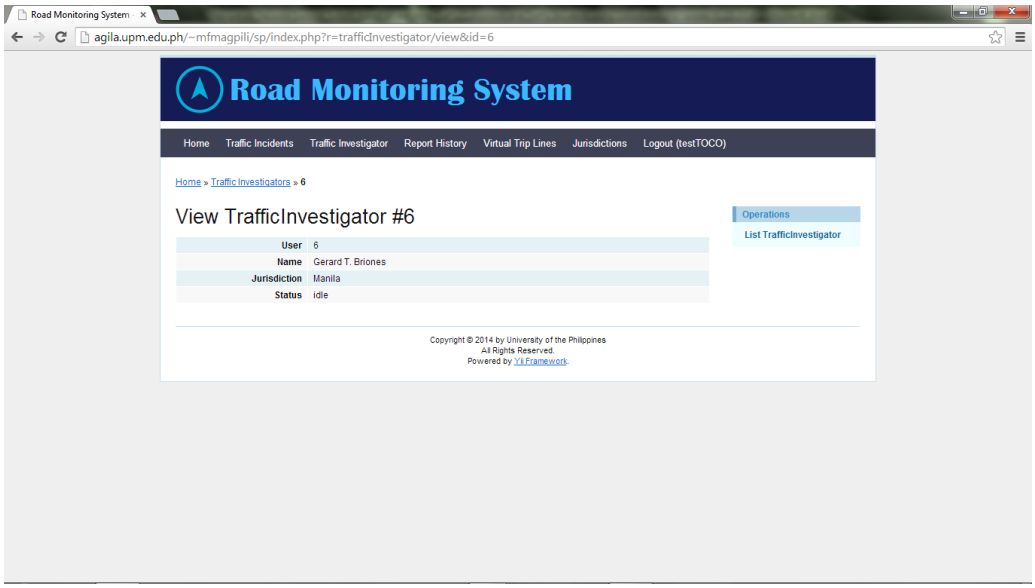


Figure 38Traffic Investigator View

The user can view the list of virtual trip lines as well as the detail by click on the link provided in the list. The user can also create a virtual trip line on this page. Once on the detail page, the user can delete or update the virtual trip lin

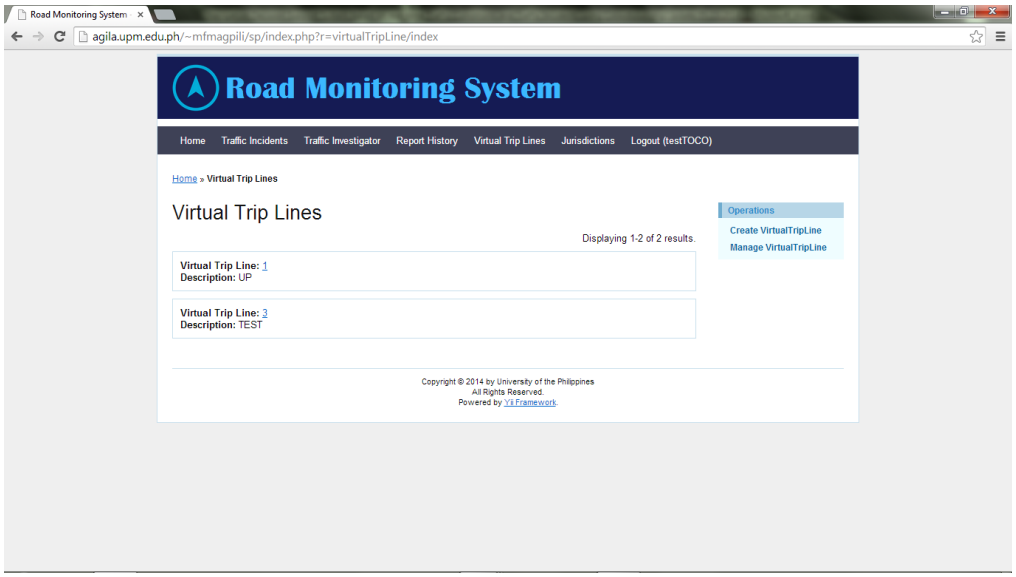


Figure 39 Virtual Trip Line List

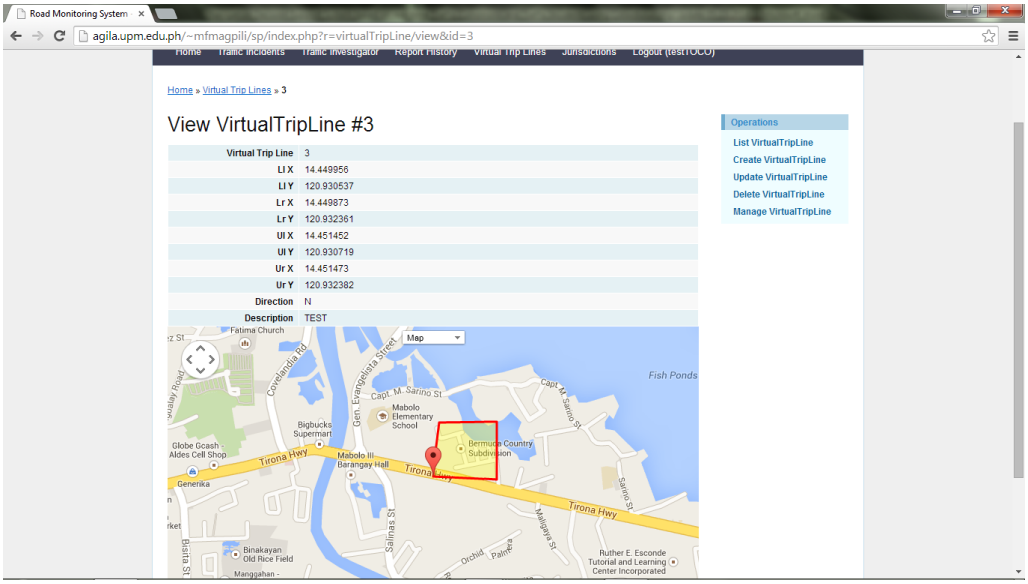


Figure 40 Virtual Trip Line View

The user can view the list of jurisdiction as well as view the detail of a jurisdiction by click on the link provided in the list. On the same page, the user can also create a new

jurisdiction. Once on the detail page, the user can update the jurisdiction.

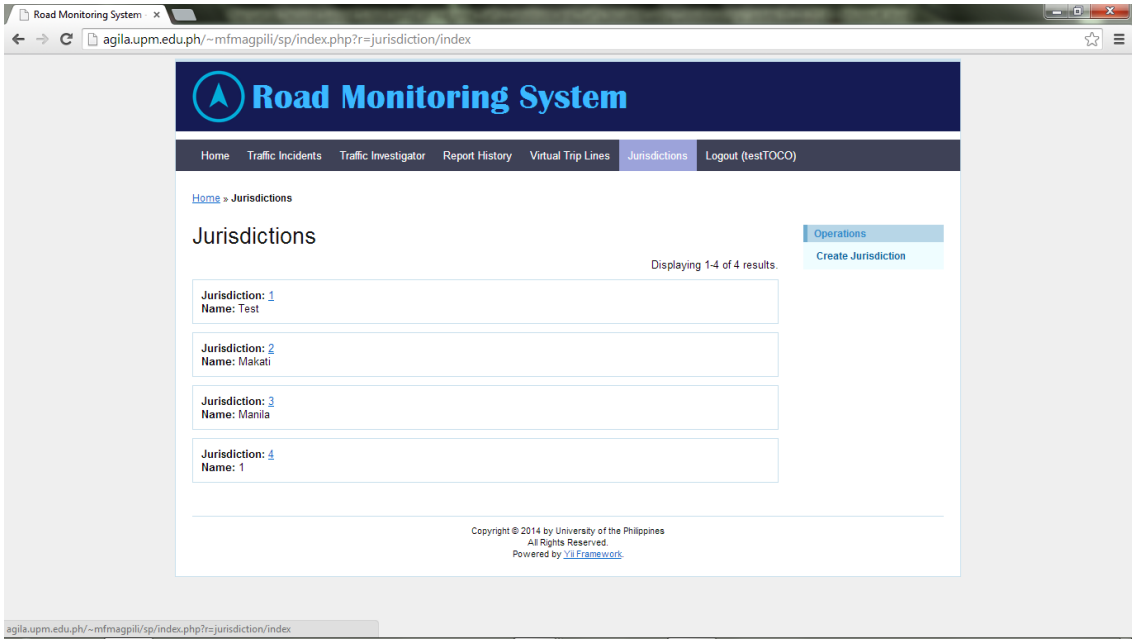


Figure 41 Juridiction List

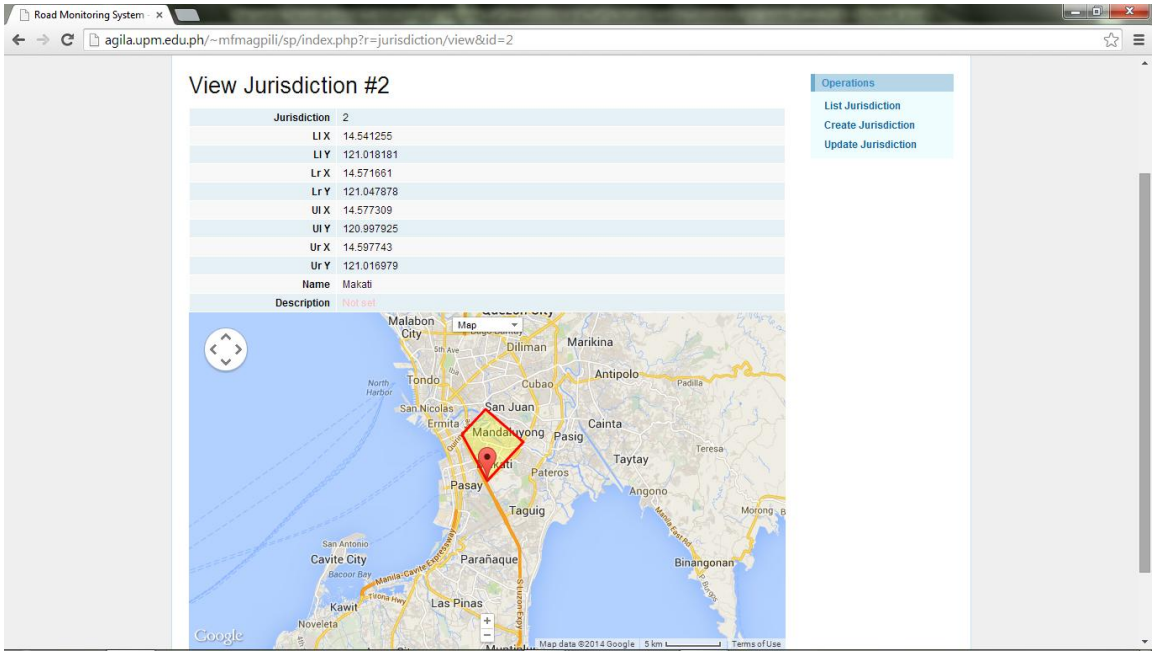


Figure 42 Jurisdiction View

System Administrator

Once successfully authenticated, a system administrator can create accounts for the following accounts of the system:

a. Traffic Operation Centre Officer

b. Traffic Investigator

c. System Administrator

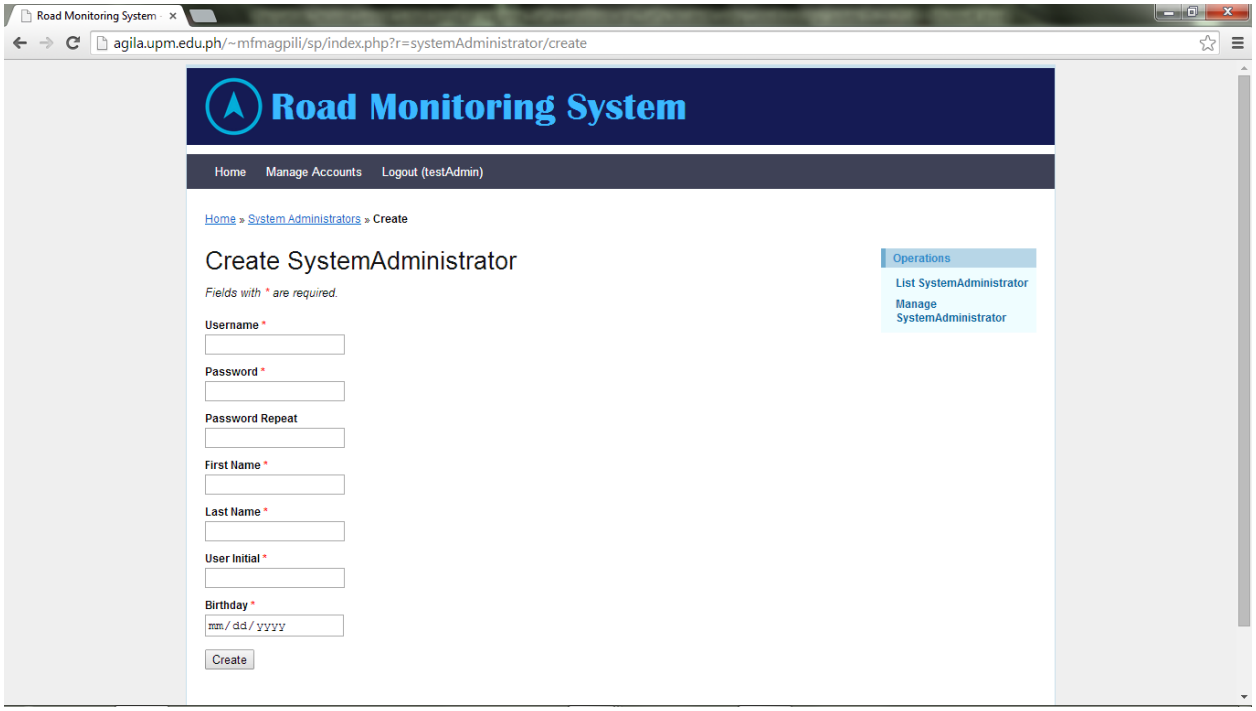


Figure 43 System Administrator Form

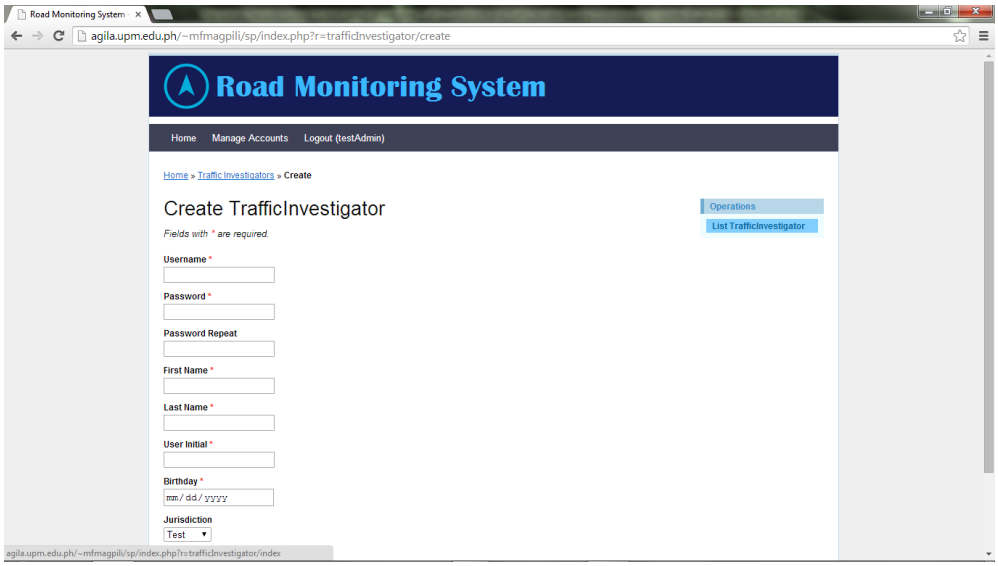


Figure 44 Traffic Investigator Form

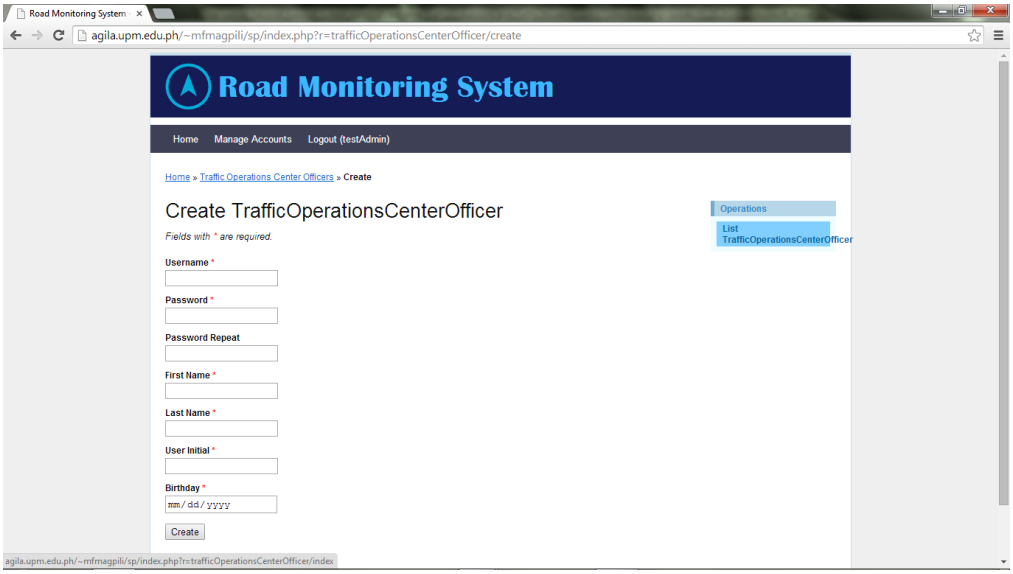


Figure 45Traffic Operation Center Form

View a list of users involved in a road accident report

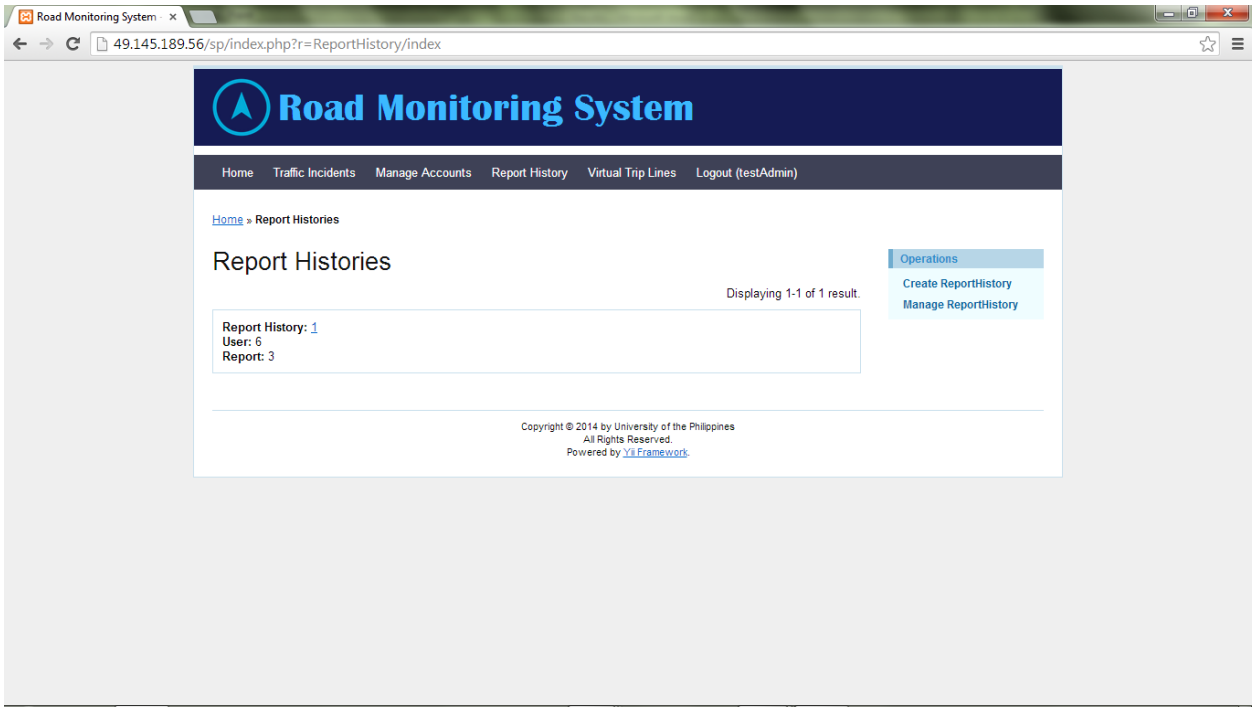


Figure 46 Report History

VI. Discussion

The Navigation Decision Support System based on real-time Traffic Condition and Traffic Incidents was implemented using Appcelerator for the mobile application and Yii framework for the web application. The system's aim is to improve traffic conditions by empowering the users to make informed decision when travelling on road. As such, a powerful system was introduced that can be accessed virtually anywhere there is internet access. Global Positioning System was used to pinpoint location of traffic incidents as accurate to plus/ minus 1 meter. The Global Positioning System is also used to record how fast the motorist was travelling which allowed the system to estimate traffic conditions only on roads monitored by virtual trip line.

The system also used Google Maps to show the locations of the traffic incidents as well the traffic condition computed by the system.

Outliers on traffic data were detected using DBSCAN algorithm which only uses 2 parameters to detect outliers in the data. Due to its simplicity, the implementation of this algorithm was easy but there was a difficulty in choosing the right parameters as these greatly influenced the effectiveness of the algorithm. During the testing phase, it was also noted that the algorithm was slow taking 30 seconds to computes 2000+ data points on a single road segments.

It should also noted that the effectiveness of the system depended amount of users using the system, as well as the amount and quality of data that is present in the system. The more users using the system, the closer the traffic estimation is to the actual traffic condition. Also the more users there are, the more likely a traffic incident will be reported or verified by fellow users

of the system. As such it would be important to find a way to increase the amount of users and user presence to populate the database with the necessary data.

VII. Conclusion

The Navigation Decision Support System based on real-time traffic conditions and traffic incidents using Crowdsourcing shows that it is possible to implement similar types of systems in the Philippines as shown in the results section. Still, further development to the system should be explored including (1) a more efficient detection of outliers in the traffic data; (2) a more intelligent algorithm on verifying the obtained data from crowdsourcing; as well as more advance features as discussed in the next section.

VIII. Recommendation

One difficulty in the system was the outlier detection which suggests that a different techniques or algorithm for finding outliers in the data is needed. This is due to the fact that output of the DBSCAN was too slow for a real time system that monitors a large number of roads.

The system would be more useful by adding a generic traffic incident that would allow users to share any traffic incident that they feel affects the traffic system. Although generating analysis report from a generic traffic incident may be difficult.

It is also useful to notify users about the conditions of the most frequent road segments they go through. This would allow the user to prepare before they start their travel.

Estimating traffic conditions can be further improved by using more advance algorithms such as the proposed algorithm in the paper Ensemble Kalman Filtering Approach to Highway Traffic Estimation rather than just computing for the average speed of all the data points. It is also suggested in the paper of mentioned algorithm, it is possible to forecast traffic states using this method.

The system can be further extended by computing the travel time from source to destination of a motorist since we can already obtain actual traffic conditions and alert users if there is enough time to reach the destination on a given appointment especially if we can sync this system with another system that contain the appointments of the user.

Lastly, a long-range analysis and report module that would analyze continuous long-term data for traffic incidents and road conditions obtained from the system would be useful. This would be helpful for traffic engineers as well as the regional and local governments for

transportation infrastructure planning. Detecting roads with heavy traffic persistent heavy traffic may suggest the alternative routes (i.e. roads) must be developed to ease traffic. It can also be useful in evaluating new traffic policies that has been implemented by the government. For example the manila government implemented a new traffic route from motorist coming from Cavite and Las Pinas Area. Comparing the status of the traffic system before the policy has been implemented and after the policy has been implemented would allow us to better evaluate the successes and effectiveness of the various traffic policies.

IX. Bibliography

[1] Al-Ali, A., Zualkernan, I., & Aloul, F. (October 2010). A Mobile GPRS-Sensors Array for Air Pollution Monitoring

[2] Chalko, T. J. (2007). High Accuracy speed measurement using GPS.

[3] Erdogan, S., Yilmaz, I., Baybura, T., Gullu, M. (2008). Geographical information systems aided traffic accident analysis system case study: city of Afyonkarahisar.

[4] Doan, A., Ramkrishnan, R., & Halevy, A. (2011). Crowdsourcing Systems on the World-Wide Web

[5] Gao, H., & Barbier, G. (2011). Harnessing the Crowdsourcing Power of Social Media for Disaster Relief. *Arizona State University*

[6] Heinzelman, J., & Waters, C. (2010). Crowdsourcing Crisis Information in Disaster-Affected Haiti

[7] Hoh, B., Iwuchukwu, T., Jacobson, Q., Work, D., Bayen, A. M., Herrera, J. C., Gruteser, M. (May 2012). Enhancing Privacy and Accuracy in Probe Vehicle-Based Traffic Monitoring via Virtual Trip Lines. *IEEE TRANSACTIONS ON MOBILE COMPUTING, VOL. 11, NO. 5, MAY 2012*

[8] Kamel, M. N., Crowley, D. N., Breslin, J. G., Burtner, G. S. R., Pike, W. A., Jezierski, E., & Chuang, K. S. (2011). Crowdsourcing, citizen sensing and sensor web technologies for public and environmental health surveillance and crisis management: trends, OGC standards and application examples

[9] Kware, S. S. (2009). Detection of Outliers in Time Series Data. *Marquette University*

[10] Magtoto, J., Roque A. (2012). Real-time traffic data collection and dissemination from an Android Smartphone using proportional computation and freesim as a practical

transportation system in Metro Manila. *TENCON 2012 - 2012 IEEE Region 10 Conference,*1-5

[11] Rao, M. (2012). Mobile Southeast Asia Report 2012.

[12] Regidor, J. R. F. (2004). Analyzing Impacts of Transportation Infrastructure and Policies on Traffic Flow in Metro Manila Using Advanced Tools and Techniques. *University of the Philippines-Diliman*

[13] Samiento, R. F., Liu, F., Fontelo, P. (2012). A Cross-Language Mobile Resource for Accessing Medline/PubMed based on an Open-Source, Crowdsourced Controlled Medical Vocubulary for the Philippines. *Journal of Mobile Technolgy in Medicine*

[14] Schweitzer, F. M., Buchinger, W., Gassmann, O., & Obrist, M. (May-June 2012). Crowdsourcing Leveraging Innovation through Online Idea Competitions

[15] Starbird, K. (2011). Digital Volunteerism During Disaster: Crowdsourcing Information Processing

[16] Tolentino, K. A. S., & Hermocilla J. A. C. (2012). Disaster Relief in Laguna: A Geographical Information System Through Crowdsourcing on Facebook. *Institute of Computer Science, University of the Philippines Los Baños*

[17] Tomas, K., Filip, M. & Antonin, S. (2008). Mobile approach trends, and technologies in modern information systems. *7th WSEAS International Conference on Applied Computer and Applied Computational Science,* Hangzhou, China pp 716-720

[18] Work, D. B., Blandin, O. T., Bayen, A. M., Iwuchukwu, T. & Tracton, K. (2008). An Ensemble Kalman Filtering Approach to Highway Traffic Estimation Using GPS Enabled Mobile Devices. *47th IEEE Conference on Decision and Control Cancun, Mexico*

[19] MMDA opens high tech flood control information system Retrieved August 20 2012 from http://www.manilatimes.net/~manilati/index.php/technology/3714-mmda-opens-high-tech-flood-control-information-center

[20] Asi Pacific Telecom Research Ltd (June 1 2009) Telecommunications in the Philippines.

[21] WAP. Retrieved August 20 2012 from http://www.webopedia.com/TERM/W/WAP.html

[22] SMART. Retrieved August 20 2012 from http://smart.com.ph/corporate/support/FAQs/gprs.htm

[23] Introduction to General Packet Radio Service (GPRS). Retrieve August 20 2012 from http://www.gsmfavorites.com/documents/gprs/

[24] Germin Ltd (July 2008). GPS Beginner's Guide.

[25] Android, the world's most popular mobile platform. Retrieved August 29 2012 from http://developer.android.com/about/index.html

[26] About Yii| Yii Framework. Retrieved August 29 2012 from http://www.yiiframework.com/about/

[27] Overview | Geographic Information Systems. Retrieved June 8 2013 from http://www.esri.com/what-is-gis/overview#overview_panel

[28] Garmin | What is GPS? Retrieved June 8 2013 from http://www8.garmin.com/aboutGPS/

[29] Philippine Hazard Map. Retrieved June 28 2013 from http://nababaha.com/

[30] Philippines Hit Hard by Typhoon Bopha, Donations Welcome | Embassy of the Philippines Retrieved July 20 2013 from view-source:http://www.philembassy.no/news-item/philippines-hit-hard-by-typhoon-bopha-donations-welcome

[31] Philippine Transit App Challenge | Create apps to solve transit problems in Philippine Retrieved July 20 2013 from http://philippine-transit.hackathome.com/

[32] House of Representatives - 15th Congress of the Philippines Retrieved July 22 from http://www.congress.gov.ph/committees/commnews/commnews_det.php?newsid=633

[33] The Official Website of the Metropolitan Manila Development Authority Retrieved July 22, 2013 from http://www.mmda.gov.ph/faq.html#page-10

[34] PLDT strike cripples DILG Patrol 117; emergency hotline | Headlines, News, The Philippine Star | philstar.com Retrieved July 22, 2013 from http://www.philstar.com/headlines/190073/pldt-strike-cripples-dilg%C2%92s-%C2%91patrol-117%C2%92-emergency-hotline

[35] Press Release - Sen. Guingona files Crowdsourcing Bill anew in 16th Congress Retrieved July 28, 2013 from http://www.senate.gov.ph/press_release/2013/0704_guingona1.asp

[36] PHL mobile usage getting more sophisticated, survey says - Yahoo! News Philippines Retrieved August 24, 2013 from view-source:http://ph.news.yahoo.com/phl-mobile-usage-getting-more-sophisticated-survey-says-085917527.html

X. Appendix

```
<?php

class AccountableUserController extends Controller
{
	/**
	 * @var string the default layout for the views. Defaults to
'//layouts/column2', meaning
	 * using two-column layout. See
'protected/views/layouts/column2.php'.
	 */
	public $layout='//layouts/column2';

	/**
	 * @return array action filters
	 */
	public function filters()
	{
		return array(
			'accessControl', // perform access control for CRUD
operations
			'postOnly + delete', // we only allow deletion via POST
request
		);
	}

	/**
	 * Specifies the access control rules.
	 * This method is used by the 'accessControl' filter.
	 * @return array access control rules
	 */
	public function accessRules()
	{
		return array(
			array('allow',  // allow all users to perform 'index' and
'view' actions

	'actions'=>array('index','view','authenticateUserMobile'),
					'users'=>array('*'),
			),
			array('allow', // allow authenticated user to perform
'create' and 'update' actions
					'actions'=>array('create','update'),
					'users'=>array('@'),
			),
			array('allow', // allow admin user to perform 'admin' and
'delete' actions
					'actions'=>array('admin','delete'),
					'users'=>array('admin'),
			),
			array('deny',  // deny all users
					'users'=>array('*'),
			),
```

```
		);
	}

	/**
	 * Displays a particular model.
	 * @param integer $id the ID of the model to be displayed
	 */
	public function actionView($id)
	{
		$this->render('view',array(
			'model'=>$this->loadModel($id),
		));
	}

	/**
	 * Creates a new model.
	 * If creation is successful, the browser will be redirected to the
'view' page.
	 */
	public function actionCreate()
	{
		$model=new AccountableUser;

		// Uncomment the following line if AJAX validation is needed
		// $this->performAjaxValidation($model);

		if(isset($_POST['AccountableUser']))
		{
			$model->attributes=$_POST['AccountableUser'];
			if($model->save())
				$this->redirect(array('view','id'=>$model-
>USER_ID));
		}

		$this->render('create',array(
			'model'=>$model,
		));
	}

	/**
	 * Updates a particular model.
	 * If update is successful, the browser will be redirected to the
'view' page.
	 * @param integer $id the ID of the model to be updated
	 */
	public function actionUpdate($id)
	{
		$model=$this->loadModel($id);

		// Uncomment the following line if AJAX validation is needed
		// $this->performAjaxValidation($model);

		if(isset($_POST['AccountableUser']))
		{
```

```
                $model->attributes=$_POST['AccountableUser'];
                if($model->save())
                      $this->redirect(array('view','id'=>$model-
>USER_ID));
            }

            $this->render('update',array(
                  'model'=>$model,
            ));
      }

      /**
       * Deletes a particular model.
       * If deletion is successful, the browser will be redirected to the
'admin' page.
       * @param integer $id the ID of the model to be deleted
       */
      public function actionDelete($id)
      {
            $this->loadModel($id)->delete();

            // if AJAX request (triggered by deletion via admin grid
view), we should not redirect the browser
            if(!isset($_GET['ajax']))
                  $this->redirect(isset($_POST['returnUrl']) ?
$_POST['returnUrl'] : array('admin'));
      }

      /**
       * Lists all models.
       */
      public function actionIndex()
      {
            $dataProvider=new CActiveDataProvider('AccountableUser');
            $this->render('index',array(
                  'dataProvider'=>$dataProvider,
            ));
      }

      /**
       * Manages all models.
       */
      public function actionAdmin()
      {
            $model=new AccountableUser('search');
            $model->unsetAttributes();  // clear any default values
            if(isset($_GET['AccountableUser']))
                  $model->attributes=$_GET['AccountableUser'];

            $this->render('admin',array(
                  'model'=>$model,
            ));
      }
```

```
	/**
	 * Performs authentication for the traffic investigator mobile app
	 * @param AccountableUser $model the model to be validated
	 */
	public function actionAuthenticateUserMobile()
	{


		$username = $_POST['username'];
		$password = $_POST['password'];

		//$username = 'testTI';
		//$password = '1234';

		$user=AccountableUser::model()->findByAttributes(array('USERNAME'=>$username));


		if($user===null){
			echo 2;
		}else{
			$trafficInvestigatorModel = TrafficInvestigator::model()->findByPk($user->USER_ID);
			if($trafficInvestigatorModel==null){
				echo 2;
			}else{
				if($user->PASSWORD!=$user->encrypt($password)){
					echo 1;
				}else{
					$trafficInvestigatorModel->IS_ACTIVE = 1;

					if($trafficInvestigatorModel->save()){
						$result = array(
							'id'=>$user->USER_ID
						);
						echo json_encode($result);
					}else{
						echo 'error';
					}


				}
			}


		}

	}

	/**
	 * Returns the data model based on the primary key given in the GET variable.
	 * If the data model is not found, an HTTP exception will be raised.
	 * @param integer $id the ID of the model to be loaded
```

```
	 * @return AccountableUser the loaded model
	 * @throws CHttpException
	 */
	public function loadModel($id)
	{
		$model=AccountableUser::model()->findByPk($id);
		if($model===null)
			throw new CHttpException(404,'The requested page does not
exist.');
		return $model;
	}

	/**
	 * Performs the AJAX validation.
	 * @param AccountableUser $model the model to be validated
	 */
	protected function performAjaxValidation($model)
	{
		if(isset($_POST['ajax']) && $_POST['ajax']==='accountable-
user-form')
		{
			echo CActiveForm::validate($model);
			Yii::app()->end();
		}
	}

}

<?php

class DataLocationCollectionController extends Controller
{
	/**
	 * @var string the default layout for the views. Defaults to
'//layouts/column2', meaning
	 * using two-column layout. See
'protected/views/layouts/column2.php'.
	 */
	public $layout='//layouts/column2';

	/**
	 * @return array action filters
	 */
	public function filters()
	{
		return array(
			'accessControl', // perform access control for CRUD
operations
			'postOnly + delete', // we only allow deletion via POST
request
		);
	}

	/**
```

```
	 * Specifies the access control rules.
	 * This method is used by the 'accessControl' filter.
	 * @return array access control rules
	 */
	public function accessRules()
	{
		return array(
			array('allow',  // allow all users to perform 'index' and
'view' actions

	'actions'=>array('index','view','CreateNewDataMobile'),
				'users'=>array('*'),
			),
			array('allow', // allow authenticated user to perform
'create' and 'update' actions
				'actions'=>array('create','update'),
				'users'=>array('@'),
			),
			array('allow', // allow admin user to perform 'admin' and
'delete' actions
				'actions'=>array('admin','delete'),
				'users'=>array('admin'),
			),
			array('deny',  // deny all users
				'users'=>array('*'),
			),
		);
	}

	/**
	 * Displays a particular model.
	 * @param integer $id the ID of the model to be displayed
	 */
	public function actionView($id)
	{
		$this->render('view',array(
			'model'=>$this->loadModel($id),
		));
	}

	/**
	 * Creates a new model.
	 * If creation is successful, the browser will be redirected to the
'view' page.
	 */
	public function actionCreate()
	{
		$model=new DataLocationCollection;

		// Uncomment the following line if AJAX validation is needed
		// $this->performAjaxValidation($model);

		if(isset($_POST['DataLocationCollection']))
		{
```

```
                $model->attributes=$_POST['DataLocationCollection'];
                if($model->save())
                      $this->redirect(array('view','id'=>$model-
>DATA_LOCATION_COLLECTION_ID));
            }

            $this->render('create',array(
                  'model'=>$model,
            ));
      }

      /**
       * Updates a particular model.
       * If update is successful, the browser will be redirected to the
'view' page.
       * @param integer $id the ID of the model to be updated
       */
      public function actionUpdate($id)
      {
            $model=$this->loadModel($id);

            // Uncomment the following line if AJAX validation is needed
            // $this->performAjaxValidation($model);

            if(isset($_POST['DataLocationCollection']))
            {
                  $model->attributes=$_POST['DataLocationCollection'];
                  if($model->save())
                        $this->redirect(array('view','id'=>$model-
>DATA_LOCATION_COLLECTION_ID));
            }

            $this->render('update',array(
                  'model'=>$model,
            ));
      }

      /**
       * Deletes a particular model.
       * If deletion is successful, the browser will be redirected to the
'admin' page.
       * @param integer $id the ID of the model to be deleted
       */
      public function actionDelete($id)
      {
            $this->loadModel($id)->delete();

            // if AJAX request (triggered by deletion via admin grid
view), we should not redirect the browser
            if(!isset($_GET['ajax']))
                  $this->redirect(isset($_POST['returnUrl']) ?
$_POST['returnUrl'] : array('admin'));
      }
```

```
	/**
	 * Lists all models.
	 */
	public function actionIndex()
	{
		$dataProvider=new
CActiveDataProvider('DataLocationCollection');
		$this->render('index',array(
			'dataProvider'=>$dataProvider,
		));
	}

	/**
	 * Manages all models.
	 */
	public function actionAdmin()
	{
		$model=new DataLocationCollection('search');
		$model->unsetAttributes();  // clear any default values
		if(isset($_GET['DataLocationCollection']))
			$model->attributes=$_GET['DataLocationCollection'];

		$this->render('admin',array(
			'model'=>$model,
		));
	}

	/**
	 * Returns the data model based on the primary key given in the GET
variable.
	 * If the data model is not found, an HTTP exception will be raised.
	 * @param integer $id the ID of the model to be loaded
	 * @return DataLocationCollection the loaded model
	 * @throws CHttpException
	 */
	public function loadModel($id)
	{
		$model=DataLocationCollection::model()->findByPk($id);
		if($model===null)
			throw new CHttpException(404,'The requested page does not
exist.');
		return $model;
	}

	/**
	 * Performs the AJAX validation.
	 * @param DataLocationCollection $model the model to be validated
	 */
	protected function performAjaxValidation($model)
	{
		if(isset($_POST['ajax']) && $_POST['ajax']==='data-location-
collection-form')
		{
			echo CActiveForm::validate($model);
```

```
                Yii::app()->end();
            }
        }

        public function actionCreateNewDataMobile()
        {
            $mobileId = $_POST['mobileId'];
            $vTLiD = $_POST['vtlid'];

            $user = Motorist::model()-
>findByAttributes(array('MOBILE_ID'=>$mobileId));

            $model = new DataLocationCollection;

            $model->USER_ID = $user->USER_ID;
            $model->VIRTUAL_TRIP_LINE_ID = $vTLiD;
            $model->START_DATETIME = new CDbExpression('NOW()');
            $model->END_DATETIME = new CDbExpression('NOW()');

            if($model->save()){
                echo $model->DATA_LOCATION_COLLECTION_ID;
            }else{
                echo 'fail';
            }
            //echo 'hello from datalocationcollection';
            /*$statusName = $_POST['statusName'];
            $reportId = $_POST['id'];
            //$statusName = 'RESOLVED';
            //$reportId = 6;

            $raReportModel = RaReport::model()->findByPk($reportId);
            $raReportStatusModel = RaReportStatus::model()-
>findByAttributes(array('NAME'=>$statusName));

            $raReportModel->RA_REPORT_STATUS_ID = $raReportStatusModel-
>RA_REPORT_STATUS_ID;

            $raReportModel->save();
            */
        }
}

<?php
class DBScan {

        private $_temp;
        private $_minPts = 20;
        private $_eps = 8;
        private $cluster = array();
        private $visited = array();

        protected function Compute_Euclidian_Distance($_x1, $_y1, $_x2,
$_y2){
```

```
            return sqrt(pow(($_x1-$_x2),2) + pow(($_y1 - $_y2),2));
      }

      protected function Get_Neighbors($data,$pIndex,$eps){
            $data_length = count($data);
            $listOfNeighbors = array();
            for($i = 0; $i < $data_length; $i++){
                  if($i!=$pIndex){
                        //not the same point

                        //prepare data
                        $_x1 = $data[$pIndex][0];
                        $_y1 = $data[$pIndex][1];
                        $_x2 = $data[$i][0];
                        $_y2 = $data[$i][1];

                        if($this-
>Compute_Euclidian_Distance($_x1,$_y1,$_x2,$_y2) < $eps){
                              array_push($listOfNeighbors,$i);
                        }

                  }
            }

            return $listOfNeighbors;
      }

      protected function Expand_Cluster($data, $pIndex,
$neighbors,$minPts,$eps){
            $data_length = count($neighbors);
            $clusterPoints = array();
            array_push($clusterPoints, $pIndex);

            for($i = 0; $i < count($neighbors); $i++){
                  if($this->visited[$neighbors[$i]]==0){
                        $this->visited[$neighbors[$i]] = 1;
                        $neighborsDelta = $this->Get_Neighbors($data,
$neighbors[$i],$eps);

                        if(count($neighborsDelta) <= $minPts){
                              $neighbors =
array_merge($neighbors,$neighborsDelta);
                              //$neighbors = array_unique($neighbors);
                              array_push($clusterPoints,$neighbors[$i]);
                        }
                  }
            }
            array_push($this->cluster, $clusterPoints);
      }


      public function Get_Outliers($data,$minPts,$eps){
            $cluster_id = 0;
            $outlier = array();
```

```
            $data_length = count($data);


            for($i = 0; $i < $data_length; $i++){
                  array_push($this->visited, 0);
            }
            //$this->visited[2117];

            for($i = 0; $i < $data_length; $i++){
                  if($this->visited[$i] == 0){
                        $this->visited[$i] = 1;
                        $listOfNeighbors = $this-
>Get_Neighbors($data,$i,$eps);
                        if(count($listOfNeighbors) <= $minPts){
                              array_push($outlier, $i);
                        }else{
                              $cluster_id = $cluster_id + 1;
                              $this->Expand_Cluster($data, $i,
$listOfNeighbors,$minPts,$eps);
                        }
                  }
            }


            $countCluster = count($this->cluster);
            //echo 'cluster count: '.$countCluster.'<br/>';
            return $outlier;
      }

      public function test(){
            $testData = array(
                  array(1,1),
                  array(2,2),
                  array(3,3),
                  array(4,4),
                  array(10,10),
                  array(1,1),
                  array(1,2),
                  array(1,3),
                  array(2,1),
                  array(2,2),
                  array(2,3),
                  array(100,100),
            );

            $outliers = $this->Get_Outliers($testData);



            for($i = 0; $i < count($outliers); $i++){
                  echo $outliers[$i].', ';
            }

      }
```

```
}
?>
<?php

class FsCategoryController extends Controller
{
	/**
	 * @var string the default layout for the views. Defaults to
'//layouts/column2', meaning
	 * using two-column layout. See
'protected/views/layouts/column2.php'.
	 */
	public $layout='//layouts/column2';

	/**
	 * @return array action filters
	 */
	public function filters()
	{
		return array(
			'accessControl', // perform access control for CRUD
operations
			'postOnly + delete', // we only allow deletion via POST
request
		);
	}

	/**
	 * Specifies the access control rules.
	 * This method is used by the 'accessControl' filter.
	 * @return array access control rules
	 */
	public function accessRules()
	{
		return array(
			array('allow',  // allow all users to perform 'index' and
'view' actions
				'actions'=>array('index','view'),
				'users'=>array('*'),
			),
			array('allow', // allow authenticated user to perform
'create' and 'update' actions

	'actions'=>array('create','update','admin','delete'),
				'users'=>array('@'),
			),
			array('allow', // allow admin user to perform 'admin' and
'delete' actions
				//'actions'=>array('admin','delete'),
				//'users'=>array('admin'),
			),
			array('deny',  // deny all users
				'users'=>array('*'),
```

```
                ),
            );
        }

        /**
         * Displays a particular model.
         * @param integer $id the ID of the model to be displayed
         */
        public function actionView($id)
        {
            $this->render('view',array(
                'model'=>$this->loadModel($id),
            ));
        }

        /**
         * Creates a new model.
         * If creation is successful, the browser will be redirected to the
'view' page.
         */
        public function actionCreate()
        {
            $model=new FsCategory;

            // Uncomment the following line if AJAX validation is needed
            // $this->performAjaxValidation($model);

            if(isset($_POST['FsCategory']))
            {
                $model->attributes=$_POST['FsCategory'];
                if($model->save())
                    $this->redirect(array('view','id'=>$model-
>FS_CATEGORY_ID));
            }

            $this->render('create',array(
                'model'=>$model,
            ));
        }

        /**
         * Updates a particular model.
         * If update is successful, the browser will be redirected to the
'view' page.
         * @param integer $id the ID of the model to be updated
         */
        public function actionUpdate($id)
        {
            $model=$this->loadModel($id);

            // Uncomment the following line if AJAX validation is needed
            // $this->performAjaxValidation($model);

            if(isset($_POST['FsCategory']))
```

```
            {
                  $model->attributes=$_POST['FsCategory'];
                  if($model->save())
                        $this->redirect(array('view','id'=>$model-
>FS_CATEGORY_ID));
            }

            $this->render('update',array(
                  'model'=>$model,
            ));
      }

      /**
       * Deletes a particular model.
       * If deletion is successful, the browser will be redirected to the
'admin' page.
       * @param integer $id the ID of the model to be deleted
       */
      public function actionDelete($id)
      {
            $this->loadModel($id)->delete();

            // if AJAX request (triggered by deletion via admin grid
view), we should not redirect the browser
            if(!isset($_GET['ajax']))
                  $this->redirect(isset($_POST['returnUrl']) ?
$_POST['returnUrl'] : array('admin'));
      }

      /**
       * Lists all models.
       */
      public function actionIndex()
      {
            $dataProvider=new CActiveDataProvider('FsCategory');
            $this->render('index',array(
                  'dataProvider'=>$dataProvider,
            ));
      }

      /**
       * Manages all models.
       */
      public function actionAdmin()
      {
            $model=new FsCategory('search');
            $model->unsetAttributes();  // clear any default values
            if(isset($_GET['FsCategory']))
                  $model->attributes=$_GET['FsCategory'];

            $this->render('admin',array(
                  'model'=>$model,
            ));
      }
```

```
	/**
	 * Returns the data model based on the primary key given in the GET
variable.
	 * If the data model is not found, an HTTP exception will be raised.
	 * @param integer $id the ID of the model to be loaded
	 * @return FsCategory the loaded model
	 * @throws CHttpException
	 */
	public function loadModel($id)
	{
		$model=FsCategory::model()->findByPk($id);
		if($model===null)
			throw new CHttpException(404,'The requested page does not
exist.');
		return $model;
	}

	/**
	 * Performs the AJAX validation.
	 * @param FsCategory $model the model to be validated
	 */
	protected function performAjaxValidation($model)
	{
		if(isset($_POST['ajax']) && $_POST['ajax']==='fs-category-
form')
		{
			echo CActiveForm::validate($model);
			Yii::app()->end();
		}
	}
}


<?php

class FsReportController extends Controller
{
	/**
	 * @var string the default layout for the views. Defaults to
'//layouts/column2', meaning
	 * using two-column layout. See
'protected/views/layouts/column2.php'.
	 */
	public $layout='//layouts/column2';
	private $_fsCategory = null;

	protected function loadFsReportCategory() {
		$this->_fsCategory=FsCategory::model()->findAll();
	}

	public function filterGetFsCategoryList($filterChain)
	{
		$this->loadFsReportCategory();
```

```
		$filterChain->run();
	}
	/**
	 * @return array action filters
	 */
	public function filters()
	{
		return array(
			'accessControl', // perform access control for CRUD
operations
			'postOnly + delete', // we only allow deletion via POST
request
		);
	}

	/**
	 * Specifies the access control rules.
	 * This method is used by the 'accessControl' filter.
	 * @return array access control rules
	 */
	public function accessRules()
	{
		return array(
			array('allow',  // allow all users to perform 'index' and
'view' actions

	'actions'=>array('index','view','CreateFsReportMobile','Test'),
				'users'=>array('*'),
			),
			array('allow', // allow authenticated user to perform
'create' and 'update' actions

	'actions'=>array('create','update','admin','delete'),
				'users'=>array('@'),
			),
			array('allow', // allow admin user to perform 'admin' and
'delete' actions
				//'actions'=>array('admin','delete'),
				//'users'=>array('admin'),
			),
			array('deny',  // deny all users
				'users'=>array('*'),
			),
		);
	}

	/**
	 * Displays a particular model.
	 * @param integer $id the ID of the model to be displayed
	 */
	public function actionView($id)
	{
		$reportModel = Report::model()->findByPk($id);
```

```
            $locationPointModel = LocationPoint::model()-
>findByPk($reportModel->LOCATION_POINT_ID);

            $this->render('view',array(
                  'model'=>$this->loadModel($id),
                  'locationPoint'=>$locationPointModel,
            ));
      }

      /**
       * Creates a new model.
       * If creation is successful, the browser will be redirected to the
'view' page.
       */
      public function actionCreate()
      {
            $modelFsReport=new FsReport;
            $modelLocationPoint=new LocationPoint;
            $modelReport=new Report;

            // Uncomment the following line if AJAX validation is needed
            // $this->performAjaxValidation($model);

            if(isset($_POST['LocationPoint']))
            {
                  $modelLocationPoint->attributes=$_POST['LocationPoint'];
                  if($modelLocationPoint->validate()){
                        if($modelLocationPoint->save()){
                              $modelReport->attributes=$_POST['Report'];
                              if($modelReport-
>END_DATETIME==''||$modelReport->START_DATETIME==''){
                                    $modelReport->END_DATETIME=new
CDbExpression('ADDDATE(NOW(),INTERVAL 2 HOUR)');
                                    $modelReport->START_DATETIME=new
CDbExpression('NOW()');
                              }
                              $modelReport->USER_ID=Yii::app()->user->id;
                              $modelReport-
>LOCATION_POINT_ID=$modelLocationPoint->LOCATION_POINT_ID;
                              $modelReport->DATETIME_CREATED = new
CDbExpression('NOW()');
                              $modelReport->IS_VERIFIED_BY_SYSTEM = 1;
                              if($modelReport->save()){
                                    $modelFsReport-
>attributes=$_POST['FsReport'];
                                    $modelFsReport->REPORT_ID=$modelReport-
>REPORT_ID;
                                    if($modelFsReport->save()){
                                          $this-
>redirect(array('view','id'=>$modelFsReport->REPORT_ID));
                                    }

                              }
                        }
```

```
                  }
            }

            $this->render('create',array(
                  'modelFsReport'=>$modelFsReport,
                  'modelLocationPoint'=>$modelLocationPoint,
                  'modelReport'=>$modelReport,
            ));
      }

      /**
       * Updates a particular model.
       * If update is successful, the browser will be redirected to the
'view' page.
       * @param integer $id the ID of the model to be updated
       */
      public function actionUpdate($id)
      {
            $modelFsReport=$this->loadModel($id);
            $modelReport = Report::model()->findByPk($id);
            $modelLocationPoint = LocationPoint::model()-
>findByPk($modelReport->LOCATION_POINT_ID);
            // Uncomment the following line if AJAX validation is needed
            // $this->performAjaxValidation($model);

            if(isset($_POST['LocationPoint']))
            {
                  $modelLocationPoint->attributes=$_POST['LocationPoint'];
                  if($modelLocationPoint->validate()){
                        if($modelLocationPoint->save()){
                              $modelReport->attributes=$_POST['Report'];
                              if($modelReport-
>END_DATETIME==''||$modelReport->START_DATETIME==''){
                                    $modelReport->END_DATETIME=new
CDbExpression('ADDDATE(NOW(),INTERVAL 2 HOUR)');
                                    $modelReport->START_DATETIME=new
CDbExpression('NOW()');
                              }
                              $modelReport->USER_ID=Yii::app()->user->id;
                              $modelReport-
>LOCATION_POINT_ID=$modelLocationPoint->LOCATION_POINT_ID;
                              $modelReport->DATETIME_CREATED = new
CDbExpression('NOW()');
                              $modelReport->IS_VERIFIED_BY_SYSTEM = 1;
                              if($modelReport->save()){
                                    $modelFsReport-
>attributes=$_POST['FsReport'];
                                    $modelFsReport->REPORT_ID=$modelReport-
>REPORT_ID;
                                    if($modelFsReport->save()){
                                          $this-
>redirect(array('view','id'=>$modelFsReport->REPORT_ID));
                                    }
```

```
                        }
                    }
                }
            }

            $this->render('update',array(
                'model'=>$modelFsReport,
                'modelReport'=>$modelReport,
                'modelLocationPoint'=>$modelLocationPoint ,
            ));
        }

        /**
         * Deletes a particular model.
         * If deletion is successful, the browser will be redirected to the
'admin' page.
         * @param integer $id the ID of the model to be deleted
         */
        public function actionDelete($id)
        {
            $this->loadModel($id)->delete();
            Report::model()->findByPk($id)->delete();

            // if AJAX request (triggered by deletion via admin grid
view), we should not redirect the browser
            if(!isset($_GET['ajax']))
                $this->redirect(isset($_POST['returnUrl']) ?
$_POST['returnUrl'] : array('index'));
        }

        /**
         * Lists all models.
         */
        public function actionIndex()
        {
            $dataProvider=new CActiveDataProvider('Report',array(
                'criteria'=>array(
                    'with'=>array('fsReport'),
                    'join' =>'JOIN fs_report fsr ON
fsr.REPORT_ID=t.REPORT_ID',
                    'condition'=>'t.START_DATETIME < NOW() AND NOW() <
t.END_DATETIME'
                )
            ));

            //$data = FsReport::model()->findAll();
            $ReportArray = Report::model()->findAll(array(
                    'with'=>array('fsReport'),
                    'join' =>'JOIN fs_report fsr ON
fsr.REPORT_ID=t.REPORT_ID',
                    'condition'=>'t.START_DATETIME < NOW() AND NOW() <
t.END_DATETIME'
            ));
            $this->render('index',array(
```

```
                'dataProvider'=>$dataProvider,
                'floodedStreets'=>$ReportArray
        ));
    }

    /**
     * Manages all models.
     */
    public function actionAdmin()
    {
        $model=new FsReport('search');
        $model->unsetAttributes();  // clear any default values
        if(isset($_GET['FsReport']))
            $model->attributes=$_GET['FsReport'];

        $this->render('admin',array(
            'model'=>$model,
        ));
    }

    /**
     * Returns the data model based on the primary key given in the GET
variable.
     * If the data model is not found, an HTTP exception will be raised.
     * @param integer $id the ID of the model to be loaded
     * @return FsReport the loaded model
     * @throws CHttpException
     */
    public function loadModel($id)
    {
        $model=FsReport::model()->findByPk($id);
        if($model===null)
            throw new CHttpException(404,'The requested page does not
exist.');
        return $model;
    }

    /**
     * Performs the AJAX validation.
     * @param FsReport $model the model to be validated
     */
    protected function performAjaxValidation($model)
    {
        if(isset($_POST['ajax']) && $_POST['ajax']==='fs-report-form')
        {
            echo CActiveForm::validate($model);
            Yii::app()->end();
        }
    }

    public function getCategoryName($id){
        $reportModel = FsCategory::model()->findByPk($id);
        return $reportModel->NAME;
    }
```

```
	/*
	Provide necessary data for traffic investigator mobile app

	*/
	public function actionCreateFsReportMobile()
	{

		$mobileID = $_POST['id'];
		$category = $_POST['category'];
		$lat = $_POST['lat'];
		$long = $_POST['longitude'];
		$isImageUploaded = $_POST['isImageUploaded'];
		/*
		$mobileID = '177d16e5d6265252';
		$category = 'Not passable to light vehicles';
		$lat = '14.450396';
		$long = '120.931723';
		$lat = '14.449627';
		$long = '120.922651';
		$isImageUploaded = 0;
		*/
		//echo $category;
		//echo $isImageUploaded;
		//echo ';

		//check if report already exist
		$ReportArray = Report::model()->findAll(array(
				'with'=>array('fsReport'),
				'join' =>'JOIN fs_report fsr ON
fsr.REPORT_ID=t.REPORT_ID',
				'condition'=>'t.START_DATETIME < NOW() AND NOW() <
t.END_DATETIME'
		));
		$arrayLength = count($ReportArray);
		for($i = 0; $i < $arrayLength; $i++){
			$currReport = $ReportArray[$i];
			//$modelReport = Report::model()->findByPk($currFsReport-
>REPORT_ID);
			$modelLocationPoint = LocationPoint::model()-
>findByPk($currReport->LOCATION_POINT_ID);
			//echo 'id: '.$currReport->REPORT_ID.'<br/>';
			//echo 'dist: '.$this-
>computeDistance($modelLocationPoint->LATITUDE,$modelLocationPoint-
>LONGITUDE,$lat,$long).'<br/>';
			if($this->computeDistance($modelLocationPoint-
>LATITUDE,$modelLocationPoint->LONGITUDE,$lat,$long) < 10){
				echo 'report already exist';
				return;
			}
		}
		//end check if report already exist

		$locationPoint=new LocationPoint;
```

```
            $locationPoint->LATITUDE = $lat;
            $locationPoint->LONGITUDE = $long;
            if($locationPoint->save()){

                  $picture = new Picture;

                  if($isImageUploaded==1){
                        if($picture->save()){
                              //echo $picture->PICTURE_ID;
                        }else{
                              echo 'fail saving picture';
                        }



                        ////upload image////

                              //get file type
                              $ImageType = @explode('/',
$_FILES['media']['type']);

                              //create unique name
                              $randName = $picture->PICTURE_ID;
                              $newName = $randName.'.'. $ImageType[1];
                              //upload dir
                              $target = 'images/';
                              $target = $target . $newName;
                              //move the file

      if(move_uploaded_file($_FILES['media']['tmp_name'], $target)){
                                    //do nothing
                              }else{
                                    echo 'fail uploading image';
                              }

                        ////end upload image////
                  }

                  $user = Motorist::model()-
>findByAttributes(array('MOBILE_ID'=>$mobileID));

                  $report = new Report;
                  if($isImageUploaded==1){
                        $report->PICTURE_ID = $picture->PICTURE_ID;
                  }

                  $report->LOCATION_POINT_ID = $locationPoint-
>LOCATION_POINT_ID;
                  $report->USER_ID = $user->USER_ID;
                  $report->DATETIME_CREATED = new CDbExpression('NOW()');
                  $report->END_DATETIME=new
CDbExpression('ADDDATE(NOW(),INTERVAL 1 HOUR)');
                  $report->START_DATETIME=new CDbExpression('NOW()');
```

```
                if($report->save()){

                      $fsCategory = FsCategory::model()-
>findByAttributes(Array('NAME'=>$category));
                      $fsReport =new FsReport;
                      $fsReport->REPORT_ID = $report->REPORT_ID;
                      $fsReport->FS_CATEGORY_ID = $fsCategory-
>FS_CATEGORY_ID;
                      if($fsReport->save()){
                            echo 'success';
                      }else{
                            echo ' fs fail';
                      }


                }else{
                      echo 'fail saving report';
                }


          }
     }

     public function actionTest(){
          $fsCategory = FsCategory::model()-
>findByAttributes(Array('NAME'=>'Not passable to light vehicles'));

                      echo $fsCategory->FS_CATEGORY_ID;
     }

     /**
      * get all the rows in FS_CATEGORY table
      *
      */
     public function getFsCategoryList(){
          $this->_fsCategory=FsCategory::model()->findAll();
          $statusArray = CHtml::listData($this->_fsCategory,
'FS_CATEGORY_ID', 'NAME');
          return $statusArray;
     }
     /**
      * get a single name from FS_CATEGORY table
      *
      */
     public function getRaCategoryName($ID){
          return RaCategory::model()->findByPk($ID)->NAME;
     }

     /**
     *     Set Report to isVerifiedBySystem
     *
     *
     */
```

```
	public function actionVerifyReport($id){
		$reportModel = Report::model()->findByPk($id);
		$reportModel->IS_VERIFIED_BY_SYSTEM = 1;
		$reportModel->save();
		$this->redirect(array('view','id'=>$id));
	}

	public function computeDistance($lat1, $long1, $lat2, $long2){
		$R = 6371; // Radius of the earth in km
		$dLat = deg2rad($lat2-$lat1);  // deg2rad below
		$dLon = deg2rad($long2-$long1);
		$a =
			sin($dLat/2) * sin($dLat/2) +
			cos(deg2rad($lat1)) * cos(deg2rad($lat2)) *
			sin($dLon/2) * sin($dLon/2);

		$c = 2 * atan2(sqrt($a), sqrt(1-$a));
		$d = ($R * $c)*1000; // Distance in m
		return $d;
	}

}

<?php

class JurisdictionController extends Controller
{
	/**
	 * @var string the default layout for the views. Defaults to
'//layouts/column2', meaning
	 * using two-column layout. See
'protected/views/layouts/column2.php'.
	 */
	public $layout='//layouts/column2';

	/**
	 * @return array action filters
	 */
	public function filters()
	{
		return array(
			'accessControl', // perform access control for CRUD
operations
			'postOnly + delete', // we only allow deletion via POST
request
		);
	}

	/**
	 * Specifies the access control rules.
	 * This method is used by the 'accessControl' filter.
	 * @return array access control rules
	 */
```

```
	public function accessRules()
	{
		return array(
			array('allow',  // allow all users to perform 'index' and
'view' actions
				'actions'=>array('index','view'),
				'users'=>array('*'),
			),
			array('allow', // allow authenticated user to perform
'create' and 'update' actions

	'actions'=>array('create','update','admin','delete'),
				'users'=>array('@'),
			),
			array('allow', // allow admin user to perform 'admin' and
'delete' actions
				//'actions'=>array('admin','delete'),
				//'users'=>array('admin'),
			),
			array('deny',  // deny all users
				'users'=>array('*'),
			),
		);
	}

	/**
	 * Displays a particular model.
	 * @param integer $id the ID of the model to be displayed
	 */
	public function actionView($id)
	{
		$this->render('view',array(
			'model'=>$this->loadModel($id),
		));
	}

	/**
	 * Creates a new model.
	 * If creation is successful, the browser will be redirected to the
'view' page.
	 */
	public function actionCreate()
	{
		$model=new Jurisdiction;

		// Uncomment the following line if AJAX validation is needed
		// $this->performAjaxValidation($model);

		if(isset($_POST['Jurisdiction']))
		{
			$model->attributes=$_POST['Jurisdiction'];
			if($model->save())
				$this->redirect(array('view','id'=>$model-
>JURISDICTION_ID));
```

```
            }

            $this->render('create',array(
                  'model'=>$model,
            ));
      }

      /**
       * Updates a particular model.
       * If update is successful, the browser will be redirected to the
'view' page.
       * @param integer $id the ID of the model to be updated
       */
      public function actionUpdate($id)
      {
            $model=$this->loadModel($id);

            // Uncomment the following line if AJAX validation is needed
            // $this->performAjaxValidation($model);

            if(isset($_POST['Jurisdiction']))
            {
                  $model->attributes=$_POST['Jurisdiction'];
                  if($model->save())
                        $this->redirect(array('view','id'=>$model-
>JURISDICTION_ID));
            }

            $this->render('update',array(
                  'model'=>$model,
            ));
      }

      /**
       * Deletes a particular model.
       * If deletion is successful, the browser will be redirected to the
'admin' page.
       * @param integer $id the ID of the model to be deleted
       */
      public function actionDelete($id)
      {
            $this->loadModel($id)->delete();

            // if AJAX request (triggered by deletion via admin grid
view), we should not redirect the browser
            if(!isset($_GET['ajax']))
                  $this->redirect(isset($_POST['returnUrl']) ?
$_POST['returnUrl'] : array('admin'));
      }

      /**
       * Lists all models.
       */
      public function actionIndex()
```

```
	{
		$dataProvider=new CActiveDataProvider('Jurisdiction');
		$this->render('index',array(
			'dataProvider'=>$dataProvider,
		));
	}

	/**
	 * Manages all models.
	 */
	public function actionAdmin()
	{
		$model=new Jurisdiction('search');
		$model->unsetAttributes();  // clear any default values
		if(isset($_GET['Jurisdiction']))
			$model->attributes=$_GET['Jurisdiction'];

		$this->render('admin',array(
			'model'=>$model,
		));
	}

	/**
	 * Returns the data model based on the primary key given in the GET
variable.
	 * If the data model is not found, an HTTP exception will be raised.
	 * @param integer $id the ID of the model to be loaded
	 * @return Jurisdiction the loaded model
	 * @throws CHttpException
	 */
	public function loadModel($id)
	{
		$model=Jurisdiction::model()->findByPk($id);
		if($model===null)
			throw new CHttpException(404,'The requested page does not
exist.');
		return $model;
	}

	/**
	 * Performs the AJAX validation.
	 * @param Jurisdiction $model the model to be validated
	 */
	protected function performAjaxValidation($model)
	{
		if(isset($_POST['ajax']) && $_POST['ajax']==='jurisdiction-
form')
		{
			echo CActiveForm::validate($model);
			Yii::app()->end();
		}
	}
}
```

```
<?php

class LocationPointController extends Controller
{
	/**
	 * @var string the default layout for the views. Defaults to
'//layouts/column2', meaning
	 * using two-column layout. See
'protected/views/layouts/column2.php'.
	 */
	public $layout='//layouts/column2';

	/**
	 * @return array action filters
	 */
	public function filters()
	{
		return array(
			'accessControl', // perform access control for CRUD
operations
			'postOnly + delete', // we only allow deletion via POST
request
		);
	}

	/**
	 * Specifies the access control rules.
	 * This method is used by the 'accessControl' filter.
	 * @return array access control rules
	 */
	public function accessRules()
	{
		return array(
			array('allow',  // allow all users to perform 'index' and
'view' actions
				'actions'=>array('index','view'),
				'users'=>array('*'),
			),
			array('allow', // allow authenticated user to perform
'create' and 'update' actions
				'actions'=>array('create','update'),
				'users'=>array('@'),
			),
			array('allow', // allow admin user to perform 'admin' and
'delete' actions
				'actions'=>array('admin','delete'),
				'users'=>array('admin'),
			),
			array('deny',  // deny all users
				'users'=>array('*'),
			),
		);
	}
```

```
	/**
	 * Displays a particular model.
	 * @param integer $id the ID of the model to be displayed
	 */
	public function actionView($id)
	{
		$this->render('view',array(
			'model'=>$this->loadModel($id),
		));
	}

	/**
	 * Creates a new model.
	 * If creation is successful, the browser will be redirected to the
'view' page.
	 */
	public function actionCreate()
	{
		$model=new LocationPoint;

		// Uncomment the following line if AJAX validation is needed
		// $this->performAjaxValidation($model);

		if(isset($_POST['LocationPoint']))
		{
			$model->attributes=$_POST['LocationPoint'];
			if($model->save())
				$this->redirect(array('view','id'=>$model-
>LOCATION_POINT_ID));
		}

		$this->render('create',array(
			'model'=>$model,
		));
	}

	/**
	 * Updates a particular model.
	 * If update is successful, the browser will be redirected to the
'view' page.
	 * @param integer $id the ID of the model to be updated
	 */
	public function actionUpdate($id)
	{
		$model=$this->loadModel($id);

		// Uncomment the following line if AJAX validation is needed
		// $this->performAjaxValidation($model);

		if(isset($_POST['LocationPoint']))
		{
			$model->attributes=$_POST['LocationPoint'];
			if($model->save())
```

```
                        $this->redirect(array('view','id'=>$model-
>LOCATION_POINT_ID));
            }

            $this->render('update',array(
                  'model'=>$model,
            ));
      }

      /**
       * Deletes a particular model.
       * If deletion is successful, the browser will be redirected to the
'admin' page.
       * @param integer $id the ID of the model to be deleted
       */
      public function actionDelete($id)
      {
            $this->loadModel($id)->delete();

            // if AJAX request (triggered by deletion via admin grid
view), we should not redirect the browser
            if(!isset($_GET['ajax']))
                  $this->redirect(isset($_POST['returnUrl']) ?
$_POST['returnUrl'] : array('admin'));
      }

      /**
       * Lists all models.
       */
      public function actionIndex()
      {
            $dataProvider=new CActiveDataProvider('LocationPoint');
            $this->render('index',array(
                  'dataProvider'=>$dataProvider,
            ));
      }

      /**
       * Manages all models.
       */
      public function actionAdmin()
      {
            $model=new LocationPoint('search');
            $model->unsetAttributes();  // clear any default values
            if(isset($_GET['LocationPoint']))
                  $model->attributes=$_GET['LocationPoint'];

            $this->render('admin',array(
                  'model'=>$model,
            ));
      }

      /**
```

```
	 * Returns the data model based on the primary key given in the GET
variable.
	 * If the data model is not found, an HTTP exception will be raised.
	 * @param integer $id the ID of the model to be loaded
	 * @return LocationPoint the loaded model
	 * @throws CHttpException
	 */
	public function loadModel($id)
	{
		$model=LocationPoint::model()->findByPk($id);
		if($model===null)
			throw new CHttpException(404,'The requested page does not
exist.');
		return $model;
	}

	/**
	 * Performs the AJAX validation.
	 * @param LocationPoint $model the model to be validated
	 */
	protected function performAjaxValidation($model)
	{
		if(isset($_POST['ajax']) && $_POST['ajax']==='location-point-
form')
		{
			echo CActiveForm::validate($model);
			Yii::app()->end();
		}
	}
}

<?php

class MotoristController extends Controller
{
	/**
	 * @var string the default layout for the views. Defaults to
'//layouts/column2', meaning
	 * using two-column layout. See
'protected/views/layouts/column2.php'.
	 */
	public $layout='//layouts/column2';

	/**
	 * @return array action filters
	 */
	public function filters()
	{
		return array(
			'accessControl', // perform access control for CRUD
operations
			'postOnly + delete', // we only allow deletion via POST
request
		);
```

```
	}

	/**
	 * Specifies the access control rules.
	 * This method is used by the 'accessControl' filter.
	 * @return array access control rules
	 */
	public function accessRules()
	{
		return array(
			array('allow',  // allow all users to perform 'index' and
'view' actions

	'actions'=>array('index','view','CheckIfUserIsRegistered','RegisterU
ser'),
				'users'=>array('*'),
			),
			array('allow', // allow authenticated user to perform
'create' and 'update' actions
				'actions'=>array('create','update'),
				'users'=>array('@'),
			),
			array('allow', // allow admin user to perform 'admin' and
'delete' actions
				'actions'=>array('admin','delete'),
				'users'=>array('admin'),
			),
			array('deny',  // deny all users
				'users'=>array('*'),
			),
		);
	}

	/**
	 * Displays a particular model.
	 * @param integer $id the ID of the model to be displayed
	 */
	public function actionView($id)
	{
		$this->render('view',array(
			'model'=>$this->loadModel($id),
		));
	}

	/**
	 * Creates a new model.
	 * If creation is successful, the browser will be redirected to the
'view' page.
	 */
	public function actionCreate()
	{
		$model=new Motorist;

		// Uncomment the following line if AJAX validation is needed
```

```
            // $this->performAjaxValidation($model);

            if(isset($_POST['Motorist']))
            {
                  $model->attributes=$_POST['Motorist'];
                  if($model->save())
                        $this->redirect(array('view','id'=>$model-
>USER_ID));
            }

            $this->render('create',array(
                  'model'=>$model,
            ));
      }

      /**
       * Updates a particular model.
       * If update is successful, the browser will be redirected to the
'view' page.
       * @param integer $id the ID of the model to be updated
       */
      public function actionUpdate($id)
      {
            $model=$this->loadModel($id);

            // Uncomment the following line if AJAX validation is needed
            // $this->performAjaxValidation($model);

            if(isset($_POST['Motorist']))
            {
                  $model->attributes=$_POST['Motorist'];
                  if($model->save())
                        $this->redirect(array('view','id'=>$model-
>USER_ID));
            }

            $this->render('update',array(
                  'model'=>$model,
            ));
      }

      /**
       * Deletes a particular model.
       * If deletion is successful, the browser will be redirected to the
'admin' page.
       * @param integer $id the ID of the model to be deleted
       */
      public function actionDelete($id)
      {
            $this->loadModel($id)->delete();

            // if AJAX request (triggered by deletion via admin grid
view), we should not redirect the browser
            if(!isset($_GET['ajax']))
```

```
                $this->redirect(isset($_POST['returnUrl']) ?
$_POST['returnUrl'] : array('admin'));
      }

      /**
       * Lists all models.
       */
      public function actionIndex()
      {
            $dataProvider=new CActiveDataProvider('Motorist');
            $this->render('index',array(
                  'dataProvider'=>$dataProvider,
            ));
      }

      /**
       * Manages all models.
       */
      public function actionAdmin()
      {
            $model=new Motorist('search');
            $model->unsetAttributes();  // clear any default values
            if(isset($_GET['Motorist']))
                  $model->attributes=$_GET['Motorist'];

            $this->render('admin',array(
                  'model'=>$model,
            ));
      }

      /**
       * Returns the data model based on the primary key given in the GET
variable.
       * If the data model is not found, an HTTP exception will be raised.
       * @param integer $id the ID of the model to be loaded
       * @return Motorist the loaded model
       * @throws CHttpException
       */
      public function loadModel($id)
      {
            $model=Motorist::model()->findByPk($id);
            if($model===null)
                  throw new CHttpException(404,'The requested page does not
exist.');
            return $model;
      }

      /**
       * Performs the AJAX validation.
       * @param Motorist $model the model to be validated
       */
      protected function performAjaxValidation($model)
      {
            if(isset($_POST['ajax']) && $_POST['ajax']==='motorist-form')
```

```
            {
                  echo CActiveForm::validate($model);
                  Yii::app()->end();
            }
      }

      public function actionRegisterUser(){
            $mobileID = $_POST['id'];
            //$mobileID = '177d16e5d62652523';
            $motoristModel=new Motorist;
            $userModel = new Users;
            $userModel->DATE_REGISTERED = new CDbExpression('NOW()');
            $userModel->LAST_LOGIN = new CDbExpression('NOW()');
            if($userModel->save()){
                  $motoristModel->MOBILE_ID = $mobileID;
                  $motoristModel->USER_ID = $userModel->USER_ID;
                  if($motoristModel->save()){
                        echo 1;
                  }else{
                        echo 0;
                  }

            }else{
                  echo 0;
            }

      }

      public function actionCheckIfUserIsRegistered(){
            $mobileID = $_POST['id'];
            //$mobileID = '177d16e5d62652523';

            $motoristModel = Motorist::model()-
>findByAttributes(array('MOBILE_ID'=>$mobileID));
            if(count($motoristModel)==0){
                  echo 0;
            }else{
                  echo 1;
            }

      }
}

<?php

class PictureController extends Controller
{
      /**
       * @var string the default layout for the views. Defaults to
'//layouts/column2', meaning
       * using two-column layout. See
'protected/views/layouts/column2.php'.
       */
      public $layout='//layouts/column2';
```

```
	/**
	 * @return array action filters
	 */
	public function filters()
	{
		return array(
			'accessControl', // perform access control for CRUD
operations
			'postOnly + delete', // we only allow deletion via POST
request
		);
	}

	/**
	 * Specifies the access control rules.
	 * This method is used by the 'accessControl' filter.
	 * @return array access control rules
	 */
	public function accessRules()
	{
		return array(
			array('allow',  // allow all users to perform 'index' and
'view' actions
				'actions'=>array('index','view'),
				'users'=>array('*'),
			),
			array('allow', // allow authenticated user to perform
'create' and 'update' actions
				'actions'=>array('create','update'),
				'users'=>array('@'),
			),
			array('allow', // allow admin user to perform 'admin' and
'delete' actions
				'actions'=>array('admin','delete'),
				'users'=>array('admin'),
			),
			array('deny',  // deny all users
				'users'=>array('*'),
			),
		);
	}

	/**
	 * Displays a particular model.
	 * @param integer $id the ID of the model to be displayed
	 */
	public function actionView($id)
	{
		$this->render('view',array(
			'model'=>$this->loadModel($id),
		));
	}
```

```
	/**
	 * Creates a new model.
	 * If creation is successful, the browser will be redirected to the
'view' page.
	 */
	public function actionCreate()
	{
		$model=new Picture;

		// Uncomment the following line if AJAX validation is needed
		// $this->performAjaxValidation($model);

		if(isset($_POST['Picture']))
		{
			$model->attributes=$_POST['Picture'];
			if($model->save())
				$this->redirect(array('view','id'=>$model-
>PICTURE_ID));
		}

		$this->render('create',array(
			'model'=>$model,
		));
	}

	/**
	 * Updates a particular model.
	 * If update is successful, the browser will be redirected to the
'view' page.
	 * @param integer $id the ID of the model to be updated
	 */
	public function actionUpdate($id)
	{
		$model=$this->loadModel($id);

		// Uncomment the following line if AJAX validation is needed
		// $this->performAjaxValidation($model);

		if(isset($_POST['Picture']))
		{
			$model->attributes=$_POST['Picture'];
			if($model->save())
				$this->redirect(array('view','id'=>$model-
>PICTURE_ID));
		}

		$this->render('update',array(
			'model'=>$model,
		));
	}

	/**
	 * Deletes a particular model.
```

```
	 * If deletion is successful, the browser will be redirected to the
'admin' page.
	 * @param integer $id the ID of the model to be deleted
	 */
	public function actionDelete($id)
	{
		$this->loadModel($id)->delete();

		// if AJAX request (triggered by deletion via admin grid
view), we should not redirect the browser
		if(!isset($_GET['ajax']))
			$this->redirect(isset($_POST['returnUrl']) ?
$_POST['returnUrl'] : array('admin'));
	}

	/**
	 * Lists all models.
	 */
	public function actionIndex()
	{
		$dataProvider=new CActiveDataProvider('Picture');
		$this->render('index',array(
			'dataProvider'=>$dataProvider,
		));
	}

	/**
	 * Manages all models.
	 */
	public function actionAdmin()
	{
		$model=new Picture('search');
		$model->unsetAttributes();  // clear any default values
		if(isset($_GET['Picture']))
			$model->attributes=$_GET['Picture'];

		$this->render('admin',array(
			'model'=>$model,
		));
	}

	/**
	 * Returns the data model based on the primary key given in the GET
variable.
	 * If the data model is not found, an HTTP exception will be raised.
	 * @param integer $id the ID of the model to be loaded
	 * @return Picture the loaded model
	 * @throws CHttpException
	 */
	public function loadModel($id)
	{
		$model=Picture::model()->findByPk($id);
		if($model===null)
```

```
                throw new CHttpException(404,'The requested page does not
exist.');
            return $model;
      }

      /**
       * Performs the AJAX validation.
       * @param Picture $model the model to be validated
       */
      protected function performAjaxValidation($model)
      {
            if(isset($_POST['ajax']) && $_POST['ajax']==='picture-form')
            {
                  echo CActiveForm::validate($model);
                  Yii::app()->end();
            }
      }
}

<?php

class RaReportController extends Controller
{
      private $_raReportStatus = null;
      private $_raReportCategory = null;

      protected function loadRaReportStatus() {
            $this->_raReportStatus=RaReportStatus::model()->findAll();
      }

      protected function loadRaReportCategory() {
            $this->_raReportCategory=RaCategory::model()->findAll();
      }

      public function filterGetRaReportStatus($filterChain)
      {
            $this->loadRaReportStatus();
            $filterChain->run();
      }

      public function filterGetRaReportCategory($filterChain)
      {
            $this->loadRaReportCategory();
            $filterChain->run();
      }

      /**
       * @var string the default layout for the views. Defaults to
'//layouts/column2', meaning
       * using two-column layout. See
'protected/views/layouts/column2.php'.
       */
      public $layout='//layouts/column2';
```

```
	/**
	 * @return array action filters
	 */
	public function filters()
	{
		return array(
			'accessControl', // perform access control for CRUD
operations
			'postOnly + delete', // we only allow deletion via POST
request
			'GetRaReportStatus + create, update, view',
			'GetRaReportCategory + create, update, view',
		);
	}

	/**
	 * Specifies the access control rules.
	 * This method is used by the 'accessControl' filter.
	 * @return array access control rules
	 */
	public function accessRules()
	{
		return array(
			array('allow',  // allow all users to perform 'index' and
'view' actions
				'actions'=>array('index','view'),
				'users'=>array('*'),
			),
			array('allow', // allow authenticated user to perform
'create' and 'update' actions

	'actions'=>array('create','update','assignTrafficInvestigator','admi
n','delete'),
				'users'=>array('@'),
			),
			array('allow', // allow admin user to perform 'admin' and
'delete' actions
				/*'actions'=>array('admin','delete'),
				'users'=>array('admin'),*/
			),
			array('deny',  // deny all users
				'users'=>array('*'),
			),
		);
	}

	/**
	 * Displays a particular model.
	 * @param integer $id the ID of the model to be displayed
	 */
	public function actionView($id)
	{
		if(!Yii::app()->user->checkAccess('readRoadAccident')){
```

```
                throw new CHttpException(403, 'You are not authorize to
access this page');
            }

            //get location point from database

            $RaReportModel = $this->loadModel($id);
            $reportModel = Report::model()->findByPk($RaReportModel-
>REPORT_ID);
            $locationPointModel = LocationPoint::model()-
>findByPk($reportModel->LOCATION_POINT_ID);


            $this->render('view',array(
                  'model'=>$RaReportModel,
                  'locationPoint'=>$locationPointModel,
                  'report'=>$reportModel,
            ));
      }

      /**
       * Creates a new model.
       * If creation is successful, the browser will be redirected to the
'view' page.
       */
      public function actionCreate()
      {
            if(!Yii::app()->user->checkAccess('createRoadAccident')){
                  throw new CHttpException(403, 'You are not authorize to
access this page');
            }

            $modelRaReport=new RaReport;
            $modelLocationPoint=new LocationPoint;
            $modelReport=new Report;
            // Uncomment the following line if AJAX validation is needed
            // $this->performAjaxValidation($model);

            if(isset($_POST['LocationPoint']))
            {
                  $modelLocationPoint->attributes=$_POST['LocationPoint'];
                  if($modelLocationPoint->validate()){
                        if($modelLocationPoint->save()){
                              $modelReport->attributes=$_POST['Report'];
                              if($modelReport-
>END_DATETIME==''||$modelReport->START_DATETIME==''){
                                    $modelReport->END_DATETIME=new
CDbExpression('ADDDATE(NOW(),INTERVAL 2 HOUR)');
                                    $modelReport->START_DATETIME=new
CDbExpression('NOW()');
                              }
                              $modelReport->USER_ID=Yii::app()->user->id;
                              $modelReport-
>LOCATION_POINT_ID=$modelLocationPoint->LOCATION_POINT_ID;
```

```
                              $modelReport->DATETIME_CREATED = new
CDbExpression('NOW()');
                              $modelReport->IS_VERIFIED_BY_SYSTEM = 1;
                              if($modelReport->save()){
                                    $modelRaReport-
>attributes=$_POST['RaReport'];
                                    $modelRaReport->REPORT_ID=$modelReport-
>REPORT_ID;
                                    if($modelRaReport->save()){
                                          $this-
>redirect(array('view','id'=>$modelRaReport->REPORT_ID));
                                    }

                              }
                        }
                  }
            }

            $this->render('create',array(
                  'modelRaReport'=>$modelRaReport,
                  'modelLocationPoint'=>$modelLocationPoint,
                  'modelReport'=>$modelReport,
            ));
      }

      /**
       * Updates a particular model.
       * If update is successful, the browser will be redirected to the
'view' page.
       * @param integer $id the ID of the model to be updated
       */
      public function actionUpdate($id)
      {
            if(!Yii::app()->user->checkAccess('updateRoadAccident')){
                  throw new CHttpException(403, 'You are not authorize to
access this page');
            }

            $model=$this->loadModel($id);

            // Uncomment the following line if AJAX validation is needed
            // $this->performAjaxValidation($model);

            if(isset($_POST['RaReport']))
            {
                  $model->attributes=$_POST['RaReport'];
                  if($model->save())
                        $this->redirect(array('view','id'=>$model-
>REPORT_ID));
            }

            $this->render('update',array(
                  'model'=>$model,
            ));
```

```
	}

	/**
	 * Deletes a particular model.
	 * If deletion is successful, the browser will be redirected to the
'admin' page.
	 * @param integer $id the ID of the model to be deleted
	 */
	public function actionDelete($id)
	{
		if(!Yii::app()->user->checkAccess('deleteRoadAccident')){
			throw new CHttpException(403, 'You are not authorize to
access this page');
		}

		$this->loadModel($id)->delete();
		Report::model()->findByPk($id)->delete();

		// if AJAX request (triggered by deletion via admin grid
view), we should not redirect the browser
		if(!isset($_GET['ajax']))
			$this->redirect(isset($_POST['returnUrl']) ?
$_POST['returnUrl'] : array('index'));
	}

	/**
	 * Lists all models.
	 */
	public function actionIndex()
	{
		if(!Yii::app()->user->checkAccess('readRoadAccident')){
			throw new CHttpException(403, 'You are not authorize to
access this page');
		}

		$ReportArray = Report::model()->findAll(array(
				'with'=>array('raReport'),
				'join' =>'JOIN ra_report rar ON
rar.REPORT_ID=t.REPORT_ID',
				'condition'=>'t.START_DATETIME < NOW() AND NOW() <
t.END_DATETIME'
		));

		$dataProvider=new CActiveDataProvider('Report',array(
			'criteria'=>array(
				'with'=>array('raReport'),
				'join' =>'JOIN ra_report rar ON
rar.REPORT_ID=t.REPORT_ID',
				'condition'=>'t.START_DATETIME < NOW() AND NOW() <
t.END_DATETIME'
			)

		));
```

```
		$this->render('index',array(
			'dataProvider'=>$dataProvider,
			'roadAccidents'=>$ReportArray,

		));
	}

	/**
	 * Manages all models.
	 */
	public function actionAdmin()
	{
		$model=new RaReport('search');
		$model->unsetAttributes();  // clear any default values
		if(isset($_GET['RaReport']))
			$model->attributes=$_GET['RaReport'];

		$this->render('admin',array(
			'model'=>$model,
		));
	}

	/**
	 * Returns the data model based on the primary key given in the GET
variable.
	 * If the data model is not found, an HTTP exception will be raised.
	 * @param integer $id the ID of the model to be loaded
	 * @return RaReport the loaded model
	 * @throws CHttpException
	 */
	public function loadModel($id)
	{
		$model=RaReport::model()->findByPk($id);
		if($model===null)
			throw new CHttpException(404,'The requested page does not
exist.');
		return $model;
	}

	/**
	 * Performs the AJAX validation.
	 * @param RaReport $model the model to be validated
	 */
	protected function performAjaxValidation($model)
	{
		if(isset($_POST['ajax']) && $_POST['ajax']==='ra-report-form')
		{
			echo CActiveForm::validate($model);
			Yii::app()->end();
		}
	}


	/**
```

```
       * get all the rows in RA_REPORT_STATUS table
       *
       */
      public function getRaReportStatusList(){
            $statusArray = CHtml::listData($this->_raReportStatus,
'RA_REPORT_STATUS_ID', 'NAME');
            return $statusArray;
      }

      /**
       * get all the rows in RA_REPORT_CATEGORY table
       *
       */
      public function getRaReportCategoryList(){
            $statusArray = CHtml::listData($this->_raReportCategory,
'RA_CATEGORY_ID', 'NAME');
            return $statusArray;
      }

      /**
       * get a single name from RA_REPORT_STATUS table
       *
       */
      public function getRaReportStatusName($RA_REPORT_STATUS_ID){
            return $this->_raReportStatus=RaReportStatus::model()-
>findByPk($RA_REPORT_STATUS_ID)->NAME;
      }

      /**
       * get a single name from RA_REPORT_STATUS table
       *
       */
      public function getRaCategoryName($RA_CATEGORY_ID){
            return $this->_raReportStatus=RaCategory::model()-
>findByPk($RA_CATEGORY_ID)->NAME;
      }

      public function actionAssignTrafficInvestigator($id)
      {
            $this->redirect('index.php?r=reportHistory/create&id='.$id);
      }


      /*
      Provide necessary data for traffic investigator mobile app

      */
      public function actionGetAllRaReportsMobile()
      {
            $userId = $_POST['id'];
            //$userId = '6';
            $reportHistoryModel = ReportHistory::model()-
>findAllByAttributes(array('USER_ID'=>$userId));
```

```
            $length = count($reportHistoryModel);
            $result = array();

            for($i = 0; $i < $length; $i++){

                  $currReportHistory = $reportHistoryModel[$i];
                  $reportModel = Report::model()-
>findByPk($currReportHistory->REPORT_ID);
                  $raReportModel = RaReport::model()-
>findByPk($reportModel->REPORT_ID);
                  $locationPointModel = LocationPoint::model()-
>findByPk($reportModel->LOCATION_POINT_ID);
                  $raCategoryModel = RaCategory::model()-
>findByPk($raReportModel->RA_CATEGORY_ID);

                  $result[] = array(
                        'id'=>$reportModel->REPORT_ID,
                        'lat'=>$locationPointModel->LATITUDE,
                        'lng'=>$locationPointModel->LONGITUDE
                  );
            }

            echo json_encode($result);
      }

      /*
      Provide necessary data for traffic investigator mobile app

      */
      public function actionGetRaReportsAssignedToUser()
      {
            $userId = $_POST['id'];
            //$userId = 6;
            $reportHistoryModel = ReportHistory::model()-
>findAllByAttributes(array('USER_ID'=>$userId));
            $length = count($reportHistoryModel);
            $result = array();


            for($i = 0; $i < $length; $i++){
                  $currReportHistory = $reportHistoryModel[$i];

                  $reportModel = Report::model()-
>findByPk($currReportHistory->REPORT_ID);
                  $raReportModel = RaReport::model()-
>findByPk($reportModel->REPORT_ID);
                  $locationPointModel = LocationPoint::model()-
>findByPk($reportModel->LOCATION_POINT_ID);
                  $raCategoryModel = RaCategory::model()-
>findByPk($raReportModel->RA_CATEGORY_ID);

                  $result[] = array(
                        'id'=>$reportModel->REPORT_ID,
                        'lat'=>$locationPointModel->LATITUDE,
```

```
                        'lng'=>$locationPointModel->LONGITUDE,
                        'status'=>$raCategoryModel->NAME
                  );
            }
            echo json_encode($result);
      }

      /*
      Provide necessary data for traffic investigator mobile app

      */
      public function actionGetRaReportDetails()
      {
            $reportId = $_POST['id'];
            //$reportId = '3';

            $reportModel = Report::model()->findByPk($reportId);
            $raReportModel = RaReport::model()->findByPk($reportModel-
>REPORT_ID);
            $locationPointModel = LocationPoint::model()-
>findByPk($reportModel->LOCATION_POINT_ID);
            $raCategoryModel = RaCategory::model()-
>findByPk($raReportModel->RA_CATEGORY_ID);
            $raStatusModel = RaReportStatus::model()-
>findByPk($raReportModel->RA_REPORT_STATUS_ID);

            $result = array(
                  'id'=>$reportModel->REPORT_ID,
                  'lat'=>$locationPointModel->LATITUDE,
                  'lng'=>$locationPointModel->LONGITUDE,
                  'address'=>'NOT YET IMPLEMENTED',
                  'status'=>$raStatusModel->NAME,
                  'category'=>$raCategoryModel->NAME,
            );

            echo json_encode($result);
      }

      /*
      Provide necessary data for traffic investigator mobile app

      */
      public function actionUpdateRaReportStatusMobile()
      {
            $statusName = $_POST['statusName'];
            $reportId = $_POST['id'];
            //$statusName = 'RESOLVED';
            //$reportId = 6;

            $raReportModel = RaReport::model()->findByPk($reportId);
            $raReportStatusModel = RaReportStatus::model()-
>findByAttributes(array('NAME'=>$statusName));
```

```
            $raReportModel->RA_REPORT_STATUS_ID = $raReportStatusModel-
>RA_REPORT_STATUS_ID;

            $raReportModel->save();

      }

      /*
      Provide necessary data for traffic investigator mobile app

      */
      public function actionCreateRaReportMobile()
      {

            $mobileID = $_POST['id'];
            $category = $_POST['category'];
            $lat = $_POST['lat'];
            $long = $_POST['longitude'];
            $isImageUploaded = $_POST['isImageUploaded'];

            /*$mobileID = '177d16e5d6265252';
            $category = 'Head-on Collision';
            //$lat = '14.450396';
            //$long = '120.931723';
            $lat = '14.449627';
            $long = '120.922651';
            $isImageUploaded = 0;
            */
            //check if report already exist
            $ReportArray = Report::model()->findAll(array(
                        'with'=>array('raReport'),
                        'join' =>'JOIN ra_report rar ON
rar.REPORT_ID=t.REPORT_ID',
                        'condition'=>'t.START_DATETIME < NOW() AND NOW() <
t.END_DATETIME'
            ));
            $arrayLength = count($ReportArray);
            for($i = 0; $i < $arrayLength; $i++){
                  $currReport = $ReportArray[$i];
                  //$modelReport = Report::model()->findByPk($currFsReport-
>REPORT_ID);
                  $modelLocationPoint = LocationPoint::model()-
>findByPk($currReport->LOCATION_POINT_ID);
                  //echo 'id: '.$currReport->REPORT_ID.'<br/>';
                  //echo 'dist: '.$this-
>computeDistance($modelLocationPoint->LATITUDE,$modelLocationPoint-
>LONGITUDE,$lat,$long).'<br/>';
                  if($this->computeDistance($modelLocationPoint-
>LATITUDE,$modelLocationPoint->LONGITUDE,$lat,$long) < 10){
                        echo 'report already exist';
                        return;
                  }
            }
            //end check if report already exist
```

```
            $locationPoint=new LocationPoint;

            $locationPoint->LATITUDE = $lat;
            $locationPoint->LONGITUDE = $long;
            if($locationPoint->save()){

                  $picture = new Picture;

                  if($isImageUploaded==1){
                        if($picture->save()){
                              //echo $picture->PICTURE_ID;
                        }else{
                              echo 'fail';
                        }



                        ////upload image////

                              //get file type
                              $ImageType = @explode('/',
$_FILES['media']['type']);

                              //create unique name
                              $randName = $picture->PICTURE_ID;
                              $newName = $randName.'.'. $ImageType[1];
                              //upload dir
                              $target = 'images/';
                              $target = $target . $newName;
                              //move the file

      if(move_uploaded_file($_FILES['media']['tmp_name'], $target)){
                                    //do nothing
                              }else{
                                    //echo 'fail';
                              }

                        ////end upload image////
                  }

                  $user = Motorist::model()-
>findByAttributes(array('MOBILE_ID'=>$mobileID));
                  $report = new Report;
                  if($isImageUploaded==1){
                        $report->PICTURE_ID = $picture->PICTURE_ID;
                  }
                  $report->LOCATION_POINT_ID = $locationPoint-
>LOCATION_POINT_ID;
                  $report->USER_ID = $user->USER_ID;
                  $report->DATETIME_CREATED = new CDbExpression('NOW()');
                  $report->END_DATETIME=new
CDbExpression('ADDDATE(NOW(),INTERVAL 1 HOUR)');
                  $report->START_DATETIME=new CDbExpression('NOW()');
```

```
                if($report->save()){
                      $raCategory = RaCategory::model()-
>findByAttributes(Array('NAME'=>$category));
                      $raReport =new RaReport;
                      $raReport->REPORT_ID = $report->REPORT_ID;
                      $raReport->RA_REPORT_STATUS_ID = 1;
                      $raReport->RA_CATEGORY_ID = $raCategory-
>RA_CATEGORY_ID;
                      if($raReport->save()){
                            echo 'success';
                      }else{
                            echo 'fail';
                      }
                }else{
                      echo 'fail';
                }


            }
      }

      /**
      *     Set Report to isVerifiedBySystem
      *
      *
      */

      public function actionVerifyReport($id){
            $reportModel = Report::model()->findByPk($id);
            $reportModel->IS_VERIFIED_BY_SYSTEM = 1;
            $reportModel->save();
            $this->redirect(array('view','id'=>$id));
      }

      public function computeDistance($lat1, $long1, $lat2, $long2){
            $R = 6371; // Radius of the earth in km
            $dLat = deg2rad($lat2-$lat1);  // deg2rad below
            $dLon = deg2rad($long2-$long1);
            $a =
                  sin($dLat/2) * sin($dLat/2) +
                  cos(deg2rad($lat1)) * cos(deg2rad($lat2)) *
                  sin($dLon/2) * sin($dLon/2);

            $c = 2 * atan2(sqrt($a), sqrt(1-$a));
            $d = ($R * $c)*1000; // Distance in m
            return $d;
      }

}

<?php

class RbacController extends Controller
```

```
{
	/**
	 * @var string the default layout for the views. Defaults to
'//layouts/column2', meaning
	 * using two-column layout. See
'protected/views/layouts/column2.php'.
	 */
	public $layout='//layouts/column2';

	/**
	 * @return array action filters
	 */
	public function filters()
	{
		return array(
			'accessControl', // perform access control for CRUD
operations
			'postOnly + delete', // we only allow deletion via POST
request
		);
	}

	/**
	 * Specifies the access control rules.
	 * This method is used by the 'accessControl' filter.
	 * @return array access control rules
	 */
	public function accessRules()
	{
		return array(
			array('allow',  // allow all users to perform 'index' and
'view' actions

	'actions'=>array('index','view','userAccessControl','create'),
					'users'=>array('*'),
			),
			array('allow', // allow authenticated user to perform
'create' and 'update' actions
					'actions'=>array('update'),
					'users'=>array('@'),
			),
			array('allow', // allow admin user to perform 'admin' and
'delete' actions
					'actions'=>array('admin','delete'),
					'users'=>array('admin'),
			),
			array('deny',  // deny all users
					'users'=>array('*'),
			),
		);
	}

	/**
	 * Displays a particular model.
```

```
	 * @param integer $id the ID of the model to be displayed
	 */
	public function actionView($id)
	{
		$this->render('view',array(
			'model'=>$this->loadModel($id),
		));
	}

	/**
	 * Creates a new model.
	 * If creation is successful, the browser will be redirected to the
'view' page.
	 */
	public function actionCreate()
	{
		echo 'auth';

		$auth=Yii::app()->authManager;

		$auth->clearAll();

		$auth->createOperation('createTrafficInvestigator','create a
traffic investigator user');
		$auth->createOperation('readTrafficInvestigator','read a
traffic investigator user');
		$auth->createOperation('updateTrafficInvestigator','update a
traffic investigator user');
		$auth->createOperation('deleteTrafficInvestigator','delete a
traffic investigator user');

		$auth->createOperation('createSystemAdministrator','create a
traffic investigator user');
		$auth->createOperation('readSystemAdministrator','read a
traffic investigator user');
		$auth->createOperation('updateSystemAdministrator','update a
traffic investigator user');
		$auth->createOperation('deleteSystemAdministrator','delete a
traffic investigator user');

		$auth-
>createOperation('createTrafficOperationsCenterOfficer','create a traffic
investigator user');
		$auth-
>createOperation('readTrafficOperationsCenterOfficer','read a traffic
investigator user');
		$auth-
>createOperation('updateTrafficOperationsCenterOfficer','update a traffic
investigator user');
		$auth-
>createOperation('deleteTrafficOperationsCenterOfficer','delete a traffic
investigator user');
```

```
            $auth->createOperation('createVirtualTripLines','create a
traffic investigator user');
            $auth->createOperation('readVirtualTripLines','read a traffic
investigator user');
            $auth->createOperation('updateVirtualTripLines','update a
traffic investigator user');
            $auth->createOperation('deleteVirtualTripLines','delete a
traffic investigator user');

            $auth->createOperation('createRoadAccident','create a traffic
investigator user');
            $auth->createOperation('readRoadAccident','read a traffic
investigator user');
            $auth->createOperation('updateRoadAccident','update a traffic
investigator user');
            $auth->createOperation('deleteRoadAccident','delete a traffic
investigator user');

            $auth->createOperation('createReportHistory','assign a traffic
investigator to a road accident');
            $auth->createOperation('readReportHistory','view traffic
investigators assigned to a road accident');
            //$auth->createOperation('updateTrafficInvestigator','update a
post');
            //$auth->createOperation('deleteTrafficInvestigator','delete a
post');

            $auth->createOperation('createRoadClosed','create a traffic
investigator user');
            $auth->createOperation('readRoadClosed','read a traffic
investigator user');
            $auth->createOperation('updateRoadClosed','update a traffic
investigator user');
            $auth->createOperation('deleteRoadClosed','delete a traffic
investigator user');

            $task=$auth->createTask('manageAccounts','managing user
accounts');
            $task->addChild('createTrafficInvestigator');
            $task->addChild('readTrafficInvestigator');
            $task->addChild('updateTrafficInvestigator');
            $task->addChild('deleteTrafficInvestigator');
            $task->addChild('createSystemAdministrator');
            $task->addChild('readSystemAdministrator');
            $task->addChild('updateSystemAdministrator');
            $task->addChild('deleteSystemAdministrator');
            $task->addChild('createTrafficOperationsCenterOfficer');
            $task->addChild('readTrafficOperationsCenterOfficer');
            $task->addChild('updateTrafficOperationsCenterOfficer');
            $task->addChild('deleteTrafficOperationsCenterOfficer');

            $task=$auth->createTask('manageTrafficIncidents');
            $task->addChild('createRoadAccident');
            $task->addChild('readRoadAccident');
```

```
            $task->addChild('updateRoadAccident');
            $task->addChild('deleteRoadAccident');
            $task->addChild('createRoadClosed');
            $task->addChild('readRoadClosed');
            $task->addChild('updateRoadClosed');
            $task->addChild('deleteRoadClosed');

            $role=$auth->createRole('systemAdmin');
            $role->addChild('manageAccounts');



            $role=$auth->createRole('trafficInvestigator');


            $task=$auth->createTask('viewRoadAccidentMenu');
            $task->addChild('readRoadAccident');

            $role=$auth->createRole('trafficOperationsCenterOfficer');
            $role->addChild('viewRoadAccidentMenu');
            $role->addChild('createReportHistory');
            $role->addChild('readRoadAccident');
            $role->addChild('updateRoadAccident');
            $role->addChild('deleteRoadAccident');
            $role->addChild('createRoadClosed');
            $role->addChild('readRoadClosed');
            $role->addChild('updateRoadClosed');
            $role->addChild('deleteRoadClosed');
            $role->addChild('readTrafficInvestigator');
            $role->addChild('readReportHistory');
            $role->addChild('createVirtualTripLines');
            $role->addChild('readVirtualTripLines');
            $role->addChild('updateVirtualTripLines');
            $role->addChild('deleteVirtualTripLines');
            $role->addChild('manageTrafficIncidents');
            $task->addChild('readTrafficInvestigator');

            $role=$auth->createRole('superUser');
            $role->addChild('systemAdmin');
            $role->addChild('trafficOperationsCenterOfficer');

            $auth->assign('superUser',4);
            $auth->assign('systemAdmin',5);
            $auth->assign('trafficInvestigator',6);
            $auth->assign('trafficOperationsCenterOfficer',7);
      }
}

<?php

class ReportController extends Controller
{
      /**
```

```
	 * @var string the default layout for the views. Defaults to
'//layouts/column2', meaning
	 * using two-column layout. See
'protected/views/layouts/column2.php'.
	 */
	public $layout='//layouts/column2';

	/**
	 * @return array action filters
	 */
	public function filters()
	{
		return array(
			'accessControl', // perform access control for CRUD
operations
			'postOnly + delete', // we only allow deletion via POST
request
		);
	}

	/**
	 * Specifies the access control rules.
	 * This method is used by the 'accessControl' filter.
	 * @return array access control rules
	 */
	public function accessRules()
	{
		return array(
			array('allow',  // allow all users to perform 'index' and
'view' actions

	'actions'=>array('index','view','GetReportsForMapAnnotation'),
				'users'=>array('*'),
			),
			array('allow', // allow authenticated user to perform
'create' and 'update' actions
				'actions'=>array('create','update'),
				'users'=>array('@'),
			),
			array('allow', // allow admin user to perform 'admin' and
'delete' actions
				'actions'=>array('admin','delete'),
				'users'=>array('admin'),
			),
			array('deny',  // deny all users
				'users'=>array('*'),
			),
		);
	}

	/**
	 * Displays a particular model.
	 * @param integer $id the ID of the model to be displayed
	 */
```

```
	public function actionView($id)
	{
		$this->render('view',array(
			'model'=>$this->loadModel($id),
		));
	}

	/**
	 * Creates a new model.
	 * If creation is successful, the browser will be redirected to the
'view' page.
	 */
	public function actionCreate()
	{
		$model=new Report;

		// Uncomment the following line if AJAX validation is needed
		// $this->performAjaxValidation($model);

		if(isset($_POST['Report']))
		{
			$model->attributes=$_POST['Report'];
			if($model->save())
				$this->redirect(array('view','id'=>$model-
>REPORT_ID));
		}

		$this->render('create',array(
			'model'=>$model,
		));
	}

	/**
	 * Updates a particular model.
	 * If update is successful, the browser will be redirected to the
'view' page.
	 * @param integer $id the ID of the model to be updated
	 */
	public function actionUpdate($id)
	{
		$model=$this->loadModel($id);

		// Uncomment the following line if AJAX validation is needed
		// $this->performAjaxValidation($model);

		if(isset($_POST['Report']))
		{
			$model->attributes=$_POST['Report'];
			if($model->save())
				$this->redirect(array('view','id'=>$model-
>REPORT_ID));
		}

		$this->render('update',array(
```

```
                'model'=>$model,
            ));
      }

      /**
       * Deletes a particular model.
       * If deletion is successful, the browser will be redirected to the
'admin' page.
       * @param integer $id the ID of the model to be deleted
       */
      public function actionDelete($id)
      {
            $this->loadModel($id)->delete();

            // if AJAX request (triggered by deletion via admin grid
view), we should not redirect the browser
            if(!isset($_GET['ajax']))
                  $this->redirect(isset($_POST['returnUrl']) ?
$_POST['returnUrl'] : array('admin'));
      }

      /**
       * Lists all models.
       */
      public function actionIndex()
      {
            $dataProvider=new CActiveDataProvider('Report');
            $this->render('index',array(
                  'dataProvider'=>$dataProvider,
            ));
      }

      /**
       * Manages all models.
       */
      public function actionAdmin()
      {
            $model=new Report('search');
            $model->unsetAttributes();  // clear any default values
            if(isset($_GET['Report']))
                  $model->attributes=$_GET['Report'];

            $this->render('admin',array(
                  'model'=>$model,
            ));
      }

      /**
       * Returns the data model based on the primary key given in the GET
variable.
       * If the data model is not found, an HTTP exception will be raised.
       * @param integer $id the ID of the model to be loaded
       * @return Report the loaded model
       * @throws CHttpException
```

```
	 */
	public function loadModel($id)
	{
		$model=Report::model()->findByPk($id);
		if($model===null)
			throw new CHttpException(404,'The requested page does not
exist.');
		return $model;
	}

	/**
	 * Performs the AJAX validation.
	 * @param Report $model the model to be validated
	 */
	protected function performAjaxValidation($model)
	{
		if(isset($_POST['ajax']) && $_POST['ajax']==='report-form')
		{
			echo CActiveForm::validate($model);
			Yii::app()->end();
		}
	}

	/*
	Provide necessary data for traffic investigator mobile app

	*/
	public function actionUpdateRaReportStatusMobile()
	{
		$statusName = $_POST['statusName'];
		$reportId = $_POST['id'];
		//$statusName = 'RESOLVED';
		//$reportId = 6;

		$raReportModel = RaReport::model()->findByPk($reportId);
		$raReportStatusModel = RaReportStatus::model()-
>findByAttributes(array('NAME'=>$statusName));

		$raReportModel->RA_REPORT_STATUS_ID = $raReportStatusModel-
>RA_REPORT_STATUS_ID;

		$raReportModel->save();

	}

	public function actionGetReportsForMapAnnotation(){
		$raReport = RaReport::Model()->findAll();
		foreach($raReport as $data){

		}

	}
}
```

```
<?php

class ReportHistoryController extends Controller
{
	/**
	 * @var string the default layout for the views. Defaults to
'//layouts/column2', meaning
	 * using two-column layout. See
'protected/views/layouts/column2.php'.
	 */
	public $layout='//layouts/column2';

	/**
	 * @return array action filters
	 */
	public function filters()
	{
		return array(
			'accessControl', // perform access control for CRUD
operations
			'postOnly + delete', // we only allow deletion via POST
request
		);
	}

	/**
	 * Specifies the access control rules.
	 * This method is used by the 'accessControl' filter.
	 * @return array access control rules
	 */
	public function accessRules()
	{
		return array(
			array('allow',  // allow all users to perform 'index' and
'view' actions
				'actions'=>array('index','view'),
				'users'=>array('*'),
			),
			array('allow', // allow authenticated user to perform
'create' and 'update' actions
				'actions'=>array('create','update','test'),
				'users'=>array('@'),
			),
			array('allow', // allow admin user to perform 'admin' and
'delete' actions
				//'actions'=>array('admin','delete'),
				//'users'=>array('admin'),
			),
			array('deny',  // deny all users
				'users'=>array('*'),
			),
		);
	}
```

```
	/**
	 * Displays a particular model.
	 * @param integer $id the ID of the model to be displayed
	 */
	public function actionView($id)
	{
		$this->render('view',array(
			'model'=>$this->loadModel($id),
		));
	}

	/**
	 * Creates a new model.
	 * If creation is successful, the browser will be redirected to the
'view' page.
	 */
	public function actionCreate($id)
	{
		$model=new ReportHistory;
		$model->REPORT_ID = $id;
		// Uncomment the following line if AJAX validation is needed
		// $this->performAjaxValidation($model);

		if(!Yii::app()->user->checkAccess('createReportHistory')){
			throw new CHttpException(403, 'You are not authorize to
access this page');
		}

		if(isset($_POST['ReportHistory']))
		{
			$model->attributes=$_POST['ReportHistory'];
			if($model->save()){

				$reportModel = RaReport::model()->findByPk($id);
				$reportModel->UPDATE_TIME = new
CDbExpression('NOW()');
				$reportModel->save();
				$this->redirect(array('view','id'=>$model-
>REPORT_HISTORY_ID));

			}
		}

		$this->render('create',array(
			'model'=>$model,
		));
	}

	/**
	 * Updates a particular model.
	 * If update is successful, the browser will be redirected to the
'view' page.
	 * @param integer $id the ID of the model to be updated
	 */
```

```
	public function actionUpdate($id)
	{
		$model=$this->loadModel($id);

		// Uncomment the following line if AJAX validation is needed
		// $this->performAjaxValidation($model);

		if(isset($_POST['ReportHistory']))
		{
			$model->attributes=$_POST['ReportHistory'];
			if($model->save())
				$this->redirect(array('view','id'=>$model-
>REPORT_HISTORY_ID));
		}

		$this->render('update',array(
			'model'=>$model,
		));
	}

	/**
	 * Deletes a particular model.
	 * If deletion is successful, the browser will be redirected to the
'admin' page.
	 * @param integer $id the ID of the model to be deleted
	 */
	public function actionDelete($id)
	{
		$this->loadModel($id)->delete();

		// if AJAX request (triggered by deletion via admin grid
view), we should not redirect the browser
		if(!isset($_GET['ajax']))
			$this->redirect(isset($_POST['returnUrl']) ?
$_POST['returnUrl'] : array('admin'));
	}

	/**
	 * Lists all models.
	 */
	public function actionIndex()
	{
		$dataProvider=new CActiveDataProvider('ReportHistory');
		$this->render('index',array(
			'dataProvider'=>$dataProvider,
		));
	}

	/**
	 * Manages all models.
	 */
	public function actionAdmin()
	{
		$model=new ReportHistory('search');
```

```
		$model->unsetAttributes();  // clear any default values
		if(isset($_GET['ReportHistory']))
			$model->attributes=$_GET['ReportHistory'];

		$this->render('admin',array(
			'model'=>$model,
		));
	}

	/**
	 * Returns the data model based on the primary key given in the GET
variable.
	 * If the data model is not found, an HTTP exception will be raised.
	 * @param integer $id the ID of the model to be loaded
	 * @return ReportHistory the loaded model
	 * @throws CHttpException
	 */
	public function loadModel($id)
	{
		$model=ReportHistory::model()->findByPk($id);
		if($model===null)
			throw new CHttpException(404,'The requested page does not
exist.');
		return $model;
	}

	/**
	 * Performs the AJAX validation.
	 * @param ReportHistory $model the model to be validated
	 */
	protected function performAjaxValidation($model)
	{
		if(isset($_POST['ajax']) && $_POST['ajax']==='report-history-
form')
		{
			echo CActiveForm::validate($model);
			Yii::app()->end();
		}
	}

	public function actionTest(){
		$this->getTrafficInvestigatorList(6);
	}

	public function getTrafficInvestigatorList($id){
		$modelReport = Report::model()->findByPk($id);
		$modelLocationPoint = LocationPoint::model()-
>findByPk($modelReport->LOCATION_POINT_ID);

		$model=TrafficInvestigator::model()->findAll();
		$modelLength = count($model);
		$trafficInvestigatorArray = Array();

		for($i = 0; $i < $modelLength; $i++){
```

```
                $temp = AccountableUser::model()->findByPk($model[$i]-
>USER_ID);
                $modelJurisdiction = Jurisdiction::model()-
>findByPk($model[$i]->JURISDICTION_ID);
                //echo 'user: '.$model[$i]->USER_ID.'<br/>';
                //echo 'jurisdiction id: '.$modelJurisdiction-
>JURISDICTION_ID.'<br/>';
                if($model[$i]->IS_ACTIVE==1&&$this-
>isInsideRectangle($modelJurisdiction->LL_X,$modelJurisdiction-
>LL_Y,$modelJurisdiction->LR_X,$modelJurisdiction-
>LR_Y,$modelJurisdiction->UL_X,$modelJurisdiction-
>UL_Y,$modelJurisdiction->UR_X,$modelJurisdiction-
>UR_Y,$modelLocationPoint->LATITUDE,$modelLocationPoint->LONGITUDE)){
                        array_push($trafficInvestigatorArray,$temp);
                }

            }

            //echo $data[0]->USER_ID.$data[0]->LAST_NAME;

            $temp = CHtml::listData($trafficInvestigatorArray, 'USER_ID',
'LAST_NAME');

            return $temp;
      }

      public function
isInsideRectangle($LLX,$LLY,$LRX,$LRY,$ULX,$ULY,$URX,$URY,$x,$y){
            $LOWERBOUND = -1;
            $UPPERBOUND = -1;
            $LEFTBOUND = -1;
            $RIGHTBOUND = -1;

            //check lower bound
            //$a = ($LLY-$LRY);

            $a = -($LLY - $LRY);
            $b = $LLX-$LRX;
            $c = -($a*$LRX + $b*$LRY);
            $d = $a*$x + $b*$y + $c;

            if($d<0){
                  $LOWERBOUND = 0;

            }else{
                  //return false;
                  $LOWERBOUND = 1;
            }

            //check upper bound
            $a = -($ULY - $URY);
            $b = $ULX-$URX;
            $c = -($a*$URX + $b*$URY);
            $d = $a*$x + $b*$y + $c;
```

```
            if($d<0){
                  $UPPERBOUND = 1;

            }else{
                  //return false;
                  $UPPERBOUND = 0;
            }

            //check left bound
            $a = -($ULY - $LLY);
            $b = $ULX-$LLX;
            $c = -($a*$LLX + $b*$LLY);
            $d = $a*$x + $b*$y + $c;
            if($d<0){
                  $LEFTBOUND = 0;

            }else{
                  //return false;
                  $LEFTBOUND = 1;
            }

            //check right bound
            $a = -($URY - $LRY);
            $b = $URX-$LRX;
            $c = -($a*$LRX + $b*$LRY);
            $d = $a*$x + $b*$y + $c;
            if($d<0){
                  $RIGHTBOUND = 1;

            }else{
                  //return false;
                  $RIGHTBOUND = 0;
            }

            //echo $LOWERBOUND.'<br/>';
            //echo $UPPERBOUND.'<br/>';
            //echo $LEFTBOUND.'<br/>';
            //echo $RIGHTBOUND.'<br/>';


      if($LOWERBOUND==1&&$UPPERBOUND==1&&$LEFTBOUND==1&&$RIGHTBOUND==1){
                  return true;
            }else{
                  return false;
            }
      }
}

<?php

class RoadClosedReportController extends Controller
{
      /**
```

```
	 * @var string the default layout for the views. Defaults to
'//layouts/column2', meaning
	 * using two-column layout. See
'protected/views/layouts/column2.php'.
	 */
	public $layout='//layouts/column2';

	/**
	 * @return array action filters
	 */
	public function filters()
	{
		return array(
			'accessControl', // perform access control for CRUD
operations
			'postOnly + delete', // we only allow deletion via POST
request
		);
	}

	/**
	 * Specifies the access control rules.
	 * This method is used by the 'accessControl' filter.
	 * @return array access control rules
	 */
	public function accessRules()
	{
		return array(
			array('allow',  // allow all users to perform 'index' and
'view' actions
				'actions'=>array('index','view','verifyReport'),
				'users'=>array('*'),
			),
			array('allow', // allow authenticated user to perform
'create' and 'update' actions

	'actions'=>array('create','update','delete','admin'),
				'users'=>array('@'),
			),
			array('allow', // allow admin user to perform 'admin' and
'delete' actions
				//'actions'=>array('admin','delete'),
				//'users'=>array('admin'),
			),
			array('deny',  // deny all users
				'users'=>array('*'),
			),
		);
	}

	/**
	 * Displays a particular model.
	 * @param integer $id the ID of the model to be displayed
	 */
```

```
	public function actionView($id)
	{

		if(!Yii::app()->user->checkAccess('readRoadClosed')){
			throw new CHttpException(403, 'You are not authorize to
access this page');
		}

		$reportModel = Report::model()->findByPk($id);
		$locationPointModel = LocationPoint::model()-
>findByPk($reportModel->LOCATION_POINT_ID);

		$this->render('view',array(
			'model'=>$this->loadModel($id),
			'locationPoint'=>$locationPointModel,
		));
	}

	/**
	 * Creates a new model.
	 * If creation is successful, the browser will be redirected to the
'view' page.
	 */
	public function actionCreate()
	{
		if(!Yii::app()->user->checkAccess('createRoadClosed')){
			throw new CHttpException(403, 'You are not authorize to
access this page');
		}

		$modelRoadClosedReport=new RoadClosedReport;
		$modelLocationPoint=new LocationPoint;
		$modelReport=new Report;
		// Uncomment the following line if AJAX validation is needed
		// $this->performAjaxValidation($model);

		if(isset($_POST['LocationPoint']))
		{

			$modelLocationPoint->attributes=$_POST['LocationPoint'];
			if($modelLocationPoint->validate()){
				if($modelLocationPoint->save()){
					$modelReport->attributes=$_POST['Report'];
					if($modelReport-
>END_DATETIME==''||$modelReport->START_DATETIME==''){
						$modelReport->END_DATETIME=new
CDbExpression('ADDDATE(NOW(),INTERVAL 2 HOUR)');
						$modelReport->START_DATETIME=new
CDbExpression('NOW()');
					}
					$modelReport->USER_ID=Yii::app()->user->id;
					$modelReport-
>LOCATION_POINT_ID=$modelLocationPoint->LOCATION_POINT_ID;
```

```
                              $modelReport->DATETIME_CREATED = new
CDbExpression('NOW()');
                              $modelReport->IS_VERIFIED_BY_SYSTEM = 1;



                              if($modelReport->save()){
                                    $modelRoadClosedReport-
>REPORT_ID=$modelReport->REPORT_ID;
                                    if($modelRoadClosedReport->save()){
                                          $this-
>redirect(array('view','id'=>$modelRoadClosedReport->REPORT_ID));
                                    }

                              }
                        }
                  }


            }

            $this->render('create',array(
                  'modelReport'=>$modelReport,
                  'modelRoadClosedReport'=>$modelRoadClosedReport,
                  'modelLocationPoint'=>$modelLocationPoint,
            ));
      }

      /**
       * Updates a particular model.
       * If update is successful, the browser will be redirected to the
'view' page.
       * @param integer $id the ID of the model to be updated
       */
      public function actionUpdate($id)
      {
            if(!Yii::app()->user->checkAccess('updateRoadClosed')){
                  throw new CHttpException(403, 'You are not authorize to
access this page');
            }

            $model=$this->loadModel($id);
            $modelReport = Report::model()->findByPk($id);
            $modelLocationPoint = LocationPoint::model()-
>findByPk($modelReport->LOCATION_POINT_ID);

            // Uncomment the following line if AJAX validation is needed
            // $this->performAjaxValidation($model);

            if(isset($_POST['LocationPoint']))
            {

                  $modelLocationPoint->attributes=$_POST['LocationPoint'];
                  if($modelLocationPoint->validate()){
```

```
                        if($modelLocationPoint->save()){
                              $modelReport->attributes=$_POST['Report'];
                              if($modelReport-
>END_DATETIME==''||$modelReport->START_DATETIME==''){
                                    $modelReport->END_DATETIME=new
CDbExpression('ADDDATE(NOW(),INTERVAL 2 HOUR)');
                                    $modelReport->START_DATETIME=new
CDbExpression('NOW()');
                              }
                              $modelReport-
>LOCATION_POINT_ID=$modelLocationPoint->LOCATION_POINT_ID;
                              $modelReport->DATETIME_CREATED = new
CDbExpression('NOW()');
                              $modelReport->IS_VERIFIED_BY_SYSTEM = 1;

                              if($modelReport->save()){

                                    $this-
>redirect(array('view','id'=>$model->REPORT_ID));


                              }
                        }
                  }

            }

            $this->render('update',array(
                  'model'=>$model,
                  'modelReport'=>$modelReport,
                  'modelLocationPoint'=>$modelLocationPoint ,
            ));
      }

      /**
       * Deletes a particular model.
       * If deletion is successful, the browser will be redirected to the
'admin' page.
       * @param integer $id the ID of the model to be deleted
       */
      public function actionDelete($id)
      {
            if(!Yii::app()->user->checkAccess('deleteRoadClosed')){
                  throw new CHttpException(403, 'You are not authorize to
access this page');
            }


            $this->loadModel($id)->delete();
            $reportModel = Report::model()->findByPk($id)->delete();

            // if AJAX request (triggered by deletion via admin grid
view), we should not redirect the browser
            if(!isset($_GET['ajax']))
```

```
                $this->redirect(isset($_POST['returnUrl']) ?
$_POST['returnUrl'] : array('index'));
      }

      /**
       * Lists all models.
       */
      public function actionIndex()
      {
            if(!Yii::app()->user->checkAccess('readRoadClosed')){
                  throw new CHttpException(403, 'You are not authorize to
access this page');
            }

            $data = RoadClosedReport::model()->findAll();

            $ReportArray = Report::model()->findAll(array(
                        'with'=>array('raReport'),
                        'join' =>'JOIN road_closed_report rac ON
rac.REPORT_ID=t.REPORT_ID',
                        'condition'=>'t.START_DATETIME < NOW() AND NOW() <
t.END_DATETIME'
            ));

            $dataProvider=new CActiveDataProvider('Report',array(
                  'criteria'=>array(
                        'with'=>array('roadClosedReport'),
                        'join' =>'JOIN road_closed_report rcr ON
rcr.REPORT_ID=t.REPORT_ID',
                        'condition'=>'t.START_DATETIME < NOW() AND NOW() <
t.END_DATETIME'
                  )

            ));
            $this->render('index',array(
                  'dataProvider'=>$dataProvider,
                  'roadClosedReportArray'=>$ReportArray,
            ));
      }

      /**
       * Manages all models.
       */
      public function actionAdmin()
      {
            if(!Yii::app()->user->checkAccess('readRoadClosed')){
                  throw new CHttpException(403, 'You are not authorize to
access this page');
            }

            $model=new RoadClosedReport('search');
            $model->unsetAttributes();  // clear any default values
            if(isset($_GET['RoadClosedReport']))
                  $model->attributes=$_GET['RoadClosedReport'];
```

```
            $this->render('admin',array(
                  'model'=>$model,
            ));
      }

      /**
       * Returns the data model based on the primary key given in the GET
variable.
       * If the data model is not found, an HTTP exception will be raised.
       * @param integer $id the ID of the model to be loaded
       * @return RoadClosedReport the loaded model
       * @throws CHttpException
       */
      public function loadModel($id)
      {
            $model=RoadClosedReport::model()->findByPk($id);
            if($model===null)
                  throw new CHttpException(404,'The requested page does not
exist.');
            return $model;
      }

      /**
       * Performs the AJAX validation.
       * @param RoadClosedReport $model the model to be validated
       */
      protected function performAjaxValidation($model)
      {
            if(isset($_POST['ajax']) && $_POST['ajax']==='road-closed-
report-form')
            {
                  echo CActiveForm::validate($model);
                  Yii::app()->end();
            }
      }
      /**
      *     Set Report to isVerifiedBySystem
      *
      *
      */

      public function actionVerifyReport($id){
            $reportModel = Report::model()->findByPk($id);
            $reportModel->IS_VERIFIED_BY_SYSTEM = 1;
            $reportModel->save();
            $this->redirect(array('view','id'=>$id));
      }

      public function actionCreateRcReportMobile(){

            $mobileID = $_POST['id'];
            $lat = $_POST['lat'];
            $long = $_POST['longitude'];
```

```
            /*
            $mobileID = '177d16e5d6265252';
            //$lat = '14.450396';
            //$long = '120.931723';
            $lat = '14.449627';
            $long = '120.922651';
            */
            //check if report already exist
            $ReportArray = Report::model()->findAll(array(
                        'with'=>array('raReport'),
                        'join' =>'JOIN road_closed_report rac ON
rac.REPORT_ID=t.REPORT_ID',
                        'condition'=>'t.START_DATETIME < NOW() AND NOW() <
t.END_DATETIME'
            ));
            $arrayLength = count($ReportArray);
            for($i = 0; $i < $arrayLength; $i++){
                  $currReport = $ReportArray[$i];
                  //$modelReport = Report::model()->findByPk($currFsReport-
>REPORT_ID);
                  $modelLocationPoint = LocationPoint::model()-
>findByPk($currReport->LOCATION_POINT_ID);
                  //echo 'id: '.$currReport->REPORT_ID.'<br/>';
                  //echo 'dist: '.$this-
>computeDistance($modelLocationPoint->LATITUDE,$modelLocationPoint-
>LONGITUDE,$lat,$long).'<br/>';
                  if($this->computeDistance($modelLocationPoint-
>LATITUDE,$modelLocationPoint->LONGITUDE,$lat,$long) < 10){
                        echo 'report already exist';
                        return;
                  }
            }
            //end check if report already exist

            $locationPoint=new LocationPoint;

            $locationPoint->LATITUDE = $lat;
            $locationPoint->LONGITUDE = $long;
            if($locationPoint->save()){
                  $user = Motorist::model()-
>findByAttributes(array('MOBILE_ID'=>$mobileID));

                  $report = new Report;
                  $report->LOCATION_POINT_ID = $locationPoint-
>LOCATION_POINT_ID;
                  $report->USER_ID = $user->USER_ID;
                  $report->DATETIME_CREATED = new CDbExpression('NOW()');
                  $report->END_DATETIME=new
CDbExpression('ADDDATE(NOW(),INTERVAL 1 HOUR)');
                  $report->START_DATETIME=new CDbExpression('NOW()');

                  if($report->save()){
                        $rcReport =new RoadClosedReport;
```

```
                        $rcReport->REPORT_ID = $report->REPORT_ID;
                        if($rcReport->save()){
                              echo 'success';
                        }else{
                              echo 'fail';
                        }
                  }else{
                        echo 'fail';
                  }
            }
      }

      public function computeDistance($lat1, $long1, $lat2, $long2){
            $R = 6371; // Radius of the earth in km
            $dLat = deg2rad($lat2-$lat1);  // deg2rad below
            $dLon = deg2rad($long2-$long1);
            $a =
                  sin($dLat/2) * sin($dLat/2) +
                  cos(deg2rad($lat1)) * cos(deg2rad($lat2)) *
                  sin($dLon/2) * sin($dLon/2);

            $c = 2 * atan2(sqrt($a), sqrt(1-$a));
            $d = ($R * $c)*1000; // Distance in m
            return $d;
      }
}

<?php

class SystemAdministratorController extends Controller
{
      /**
       * @var string the default layout for the views. Defaults to
'//layouts/column2', meaning
       * using two-column layout. See
'protected/views/layouts/column2.php'.
       */
      public $layout='//layouts/column2';

      /**
       * @return array action filters
       */
      public function filters()
      {
            return array(
                  'accessControl', // perform access control for CRUD
operations
                  'postOnly + delete', // we only allow deletion via POST
request
            );
      }

      /**
       * Specifies the access control rules.
```

```
	 * This method is used by the 'accessControl' filter.
	 * @return array access control rules
	 */
	public function accessRules()
	{
		return array(
			array('allow',  // allow all users to perform 'index' and
'view' actions
				'actions'=>array('index','view'),
				'users'=>array('*'),
			),
			array('allow', // allow authenticated user to perform
'create' and 'update' actions

	'actions'=>array('create','update','admin','delete'),
				'users'=>array('@'),
			),
			array('allow', // allow admin user to perform 'admin' and
'delete' actions
				//'actions'=>array(),
				//'users'=>array('admin'),
			),
			array('deny',  // deny all users
				'users'=>array('*'),
			),
		);
	}

	/**
	 * Displays a particular model.
	 * @param integer $id the ID of the model to be displayed
	 */
	public function actionView($id)
	{

		if(!Yii::app()->user->checkAccess('manageAccounts')){
			throw new CHttpException(403, 'You are not authorize to
access this page');
		}

		$this->render('view',array(
			'model'=>$this->loadModel($id),
		));
	}

	/**
	 * Creates a new model.
	 * If creation is successful, the browser will be redirected to the
'view' page.
	 */
	public function actionCreate()
	{

		if(!Yii::app()->user->checkAccess('manageAccounts')){
```

```
                throw new CHttpException(403, 'You are not authorize to
access this page');
            }

            $modelSystemAdministrator=new SystemAdministrator;
            $modelAccountableUser=new AccountableUser;
            $modelUsers= new Users;

            // Uncomment the following line if AJAX validation is needed
            // $this->performAjaxValidation($model);

            if(isset($_POST['AccountableUser']))
            {
                  $modelAccountableUser-
>attributes=$_POST['AccountableUser'];

                  $attributes =
array('FIRST_NAME','LAST_NAME','USER_INITIAL','PASSWORD','BIRTHDAY','USERN
AME');

                  if($modelAccountableUser->validate($attributes)){
                        $modelUsers->DATE_REGISTERED = new
CDbExpression('NOW()');
                        $modelUsers->LAST_LOGIN = new
CDbExpression('NOW()');
                        if($modelUsers->save()){
                              $modelAccountableUser->USER_ID = $modelUsers-
>USER_ID;
                              if($modelAccountableUser->save(false)){
                                    $modelSystemAdministrator->USER_ID =
$modelUsers->USER_ID;
                                    if($modelSystemAdministrator->save()){
                                          $auth=Yii::app()->authManager;
                                          $auth-
>assign('systemAdmin',$modelUsers->USER_ID);
                                          $this-
>redirect(array('view','id'=>$modelSystemAdministrator->USER_ID));
                                    }
                              }
                        }
                  }
            }

            $this->render('create',array(
                  'modelSystemAdministrator'=>$modelSystemAdministrator,
                  'modelAccountableUser'=>$modelAccountableUser,
                  'modelUsers'=>$modelUsers,
            ));
      }

      /**
       * Updates a particular model.
       * If update is successful, the browser will be redirected to the
'view' page.
```

```
	 * @param integer $id the ID of the model to be updated
	 */
	public function actionUpdate($id)
	{
		if(!Yii::app()->user->checkAccess('manageAccounts')){
			throw new CHttpException(403, 'You are not authorize to
access this page');
		}
		$modelSystemAdministrator=$this->loadModel($id);;
		$modelAccountableUser= AccountableUser::model()-
>findByPk($id);
		$modelUsers= Users::model()->findByPk($id);
		//$model=$this->loadModel($id);

		// Uncomment the following line if AJAX validation is needed
		// $this->performAjaxValidation($model);

		if(isset($_POST['SystemAdministrator']))
		{
			$model->attributes=$_POST['SystemAdministrator'];
			if($model->save())
				$this->redirect(array('view','id'=>$model-
>USER_ID));
		}

		$this->render('update',array(
			'modelSystemAdministrator'=>$modelSystemAdministrator,
			'modelAccountableUser'=>$modelAccountableUser,
			'modelUsers'=>$modelUsers,
		));
	}

	/**
	 * Deletes a particular model.
	 * If deletion is successful, the browser will be redirected to the
'admin' page.
	 * @param integer $id the ID of the model to be deleted
	 */
	public function actionDelete($id)
	{
		if(!Yii::app()->user->checkAccess('manageAccounts')){
			throw new CHttpException(403, 'You are not authorize to
access this page');
		}

		SystemAdministrator::model()->deleteByPk($id);
		AccountableUser::model()->deleteByPk($id);
		Users::model()->deleteByPk($id);

		// if AJAX request (triggered by deletion via admin grid
view), we should not redirect the browser
		if(!isset($_GET['ajax']))
			$this->redirect(isset($_POST['returnUrl']) ?
$_POST['returnUrl'] : array('admin'));
```

```
	}

	/**
	 * Lists all models.
	 */
	public function actionIndex()
	{
		if(!Yii::app()->user->checkAccess('manageAccounts')){
			throw new CHttpException(403, 'You are not authorize to
access this page');
		}

		$data = AccountableUser::model()->findAll();

		$dataProvider=new CActiveDataProvider('SystemAdministrator');
		$this->render('index',array(
			'dataProvider'=>$dataProvider,
			'usersDetails'=>$data,
		));
	}

	/**
	 * Manages all models.
	 */
	public function actionAdmin()
	{
		if(!Yii::app()->user->checkAccess('manageAccounts')){
			throw new CHttpException(403, 'You are not authorize to
access this page');
		}

		$model=new SystemAdministrator('search');
		$model->unsetAttributes();  // clear any default values
		if(isset($_GET['SystemAdministrator']))
			$model->attributes=$_GET['SystemAdministrator'];

		$this->render('admin',array(
			'model'=>$model,
		));
	}

	/**
	 * Manages all models.
	 */
	public function actionTimer()
	{
		/*if(!Yii::app()->user->checkAccess('manageAccounts')){
			throw new CHttpException(403, 'You are not authorize to
access this page');
		}
		*/
		$model=new SystemAdministrator('search');

		// current time
```

```
            echo date('h:i:s') . "\n";

            // sleep for 10 seconds
            sleep(10);

            // wake up !
            echo date('h:i:s') . "\n";


            /*
            $this->render('updateTimer',array(
                  'model'=>$model,
            ));
            */


      }


      /**
       * Returns the data model based on the primary key given in the GET
variable.
       * If the data model is not found, an HTTP exception will be raised.
       * @param integer $id the ID of the model to be loaded
       * @return SystemAdministrator the loaded model
       * @throws CHttpException
       */
      public function loadModel($id)
      {
            $model=SystemAdministrator::model()->findByPk($id);
            if($model===null)
                  throw new CHttpException(404,'The requested page does not
exist.');
            return $model;
      }

      /**
       * Performs the AJAX validation.
       * @param SystemAdministrator $model the model to be validated
       */
      protected function performAjaxValidation($model)
      {
            if(isset($_POST['ajax']) && $_POST['ajax']==='system-
administrator-form')
            {
                  echo CActiveForm::validate($model);
                  Yii::app()->end();
            }
      }

      public function getName($id){
            $userDetails = AccountableUser::model()->findByPk($id);
            $firstName = $userDetails->FIRST_NAME;
            $lastName = $userDetails->LAST_NAME;
```

```
            $initial = $userDetails->USER_INITIAL;
            return $firstName.' '.$initial.'. '.$lastName;
      }
}

<?php

class TrafficInvestigatorController extends Controller
{
      /**
       * @var string the default layout for the views. Defaults to
'//layouts/column2', meaning
       * using two-column layout. See
'protected/views/layouts/column2.php'.
       */
      public $layout='//layouts/column2';

      /**
       * @return array action filters
       */
      public function filters()
      {
            return array(
                  'accessControl', // perform access control for CRUD
operations
                  'postOnly + delete', // we only allow deletion via POST
request
            );
      }

      /**
       * Specifies the access control rules.
       * This method is used by the 'accessControl' filter.
       * @return array access control rules
       */
      public function accessRules()
      {
            return array(
                  array('allow',  // allow all users to perform 'index' and
'view' actions

      'actions'=>array('index','view','auth','UpdateCoordinatesMobile','Lo
goutMobile'),
                        'users'=>array('*'),
                  ),
                  array('allow', // allow authenticated user to perform
'create' and 'update' actions
                        'actions'=>array('create','update'),
                        'users'=>array('@'),
                  ),
                  array('allow', // allow admin user to perform 'admin' and
'delete' actions
                        'actions'=>array('admin','delete'),
                        'users'=>array('admin'),
```

```
                ),
                array('deny',  // deny all users
                      'users'=>array('*'),
                ),
          );
     }

     /**
      * Displays a particular model.
      * @param integer $id the ID of the model to be displayed
      */
     public function actionView($id)
     {

          if(!Yii::app()->user->checkAccess('readTrafficInvestigator')){
                throw new CHttpException(403, 'You are not authorize to
access this page');
          }

          $this->render('view',array(
                'model'=>$this->loadModel($id),
          ));
     }

     /**
      * Creates a new model.
      * If creation is successful, the browser will be redirected to the
'view' page.
      */
     public function actionCreate()
     {

          if(!Yii::app()->user->checkAccess('manageAccounts')){
                throw new CHttpException(403, 'You are not authorize to
access this page');
          }

          $modelTrafficInvestigator=new TrafficInvestigator;
          $modelAccountableUser=new AccountableUser;
          $modelUsers= new Users;

          // Uncomment the following line if AJAX validation is needed
          // $this->performAjaxValidation($model);

          if(isset($_POST['AccountableUser']))
          {

                $modelAccountableUser-
>attributes=$_POST['AccountableUser'];

                $attributes =
array('FIRST_NAME','LAST_NAME','USER_INITIAL','PASSWORD','BIRTHDAY','USERN
AME');
```

```
                if($modelAccountableUser->validate($attributes)){
                        $modelTrafficInvestigator-
>TRAFFIC_INVESTIGATOR_STATUS_ID = 1;
                        $modelUsers->DATE_REGISTERED = new
CDbExpression('NOW()');
                        $modelUsers->LAST_LOGIN = new
CDbExpression('NOW()');
                        if($modelUsers->save()){
                              $modelAccountableUser->USER_ID = $modelUsers-
>USER_ID;
                              if($modelAccountableUser->save(false)){
                                    //echo $_POST['TrafficInvestigator']-
>JURISDICTION_ID;
                                    $modelTrafficInvestigator-
>attributes=$_POST['TrafficInvestigator'];
                                    $modelTrafficInvestigator->USER_ID =
$modelUsers->USER_ID;
                                    $modelTrafficInvestigator->IS_ACTIVE =
0;
                                    if($modelTrafficInvestigator->save()){
                                          $auth=Yii::app()->authManager;
                                          $auth-
>assign('trafficInvestigator',$modelUsers->USER_ID);
                                          $this-
>redirect(array('view','id'=>$modelTrafficInvestigator->USER_ID));
                                    }
                              }
                        }
                  }
            }

            $this->render('create',array(
                  'modelTrafficInvestigator'=>$modelTrafficInvestigator,
                  'modelAccountableUser'=>$modelAccountableUser,
                  'modelUsers'=>$modelUsers,
            ));
      }

      /**
       * Updates a particular model.
       * If update is successful, the browser will be redirected to the
'view' page.
       * @param integer $id the ID of the model to be updated
       */
      public function actionUpdate($id)
      {

            if(!Yii::app()->user->checkAccess('manageAccounts')){
                  throw new CHttpException(403, 'You are not authorize to
access this page');
            }
            $modelTrafficInvestigator=$this->loadModel($id);
            $modelAccountableUser= AccountableUser::model()-
>findByPk($id);
```

```
            $modelUsers= Users::model()->findByPk($id);
            //$model=$this->loadModel($id);

            // Uncomment the following line if AJAX validation is needed
            // $this->performAjaxValidation($model);

            if(isset($_POST['TrafficInvestigator']))
            {
                  $model->attributes=$_POST['TrafficInvestigator'];
                  if($model->save())
                        $this->redirect(array('view','id'=>$model-
>USER_ID));
            }

            $this->render('update',array(
                  'modelTrafficInvestigator'=>$modelTrafficInvestigator,
                  'modelAccountableUser'=>$modelAccountableUser,
                  'modelUsers'=>$modelUsers,
            ));
      }

      /**
       * Deletes a particular model.
       * If deletion is successful, the browser will be redirected to the
'admin' page.
       * @param integer $id the ID of the model to be deleted
       */
      public function actionDelete($id)
      {

            if(!Yii::app()->user->checkAccess('manageAccounts')){
                  throw new CHttpException(403, 'You are not authorize to
access this page');
            }

            TrafficInvestigator::model()->deleteByPk($id);
            AccountableUser::model()->deleteByPk($id);
            Users::model()->deleteByPk($id);

            // if AJAX request (triggered by deletion via admin grid
view), we should not redirect the browser
            if(!isset($_GET['ajax']))
                  $this->redirect(isset($_POST['returnUrl']) ?
$_POST['returnUrl'] : array('admin'));
      }

      /**
       * Lists all models.
       */
      public function actionIndex()
      {
```

```
            if(!Yii::app()->user-
>checkAccess('manageAccounts')&&!Yii::app()->user-
>checkAccess('readTrafficInvestigator')){
                  throw new CHttpException(403, 'You are not authorize to
access this page');
            }

            $data = TrafficInvestigator::model()->findAll();

            $dataProvider=new CActiveDataProvider('TrafficInvestigator');
            $this->render('index',array(
                  'dataProvider'=>$dataProvider,
                  'trafficInvestigators'=>$data,
            ));
      }

      /**
       * Manages all models.
       */
      public function actionAdmin()
      {

            if(!Yii::app()->user->checkAccess('manageAccounts')){
                  throw new CHttpException(403, 'You are not authorize to
access this page');
            }

            $model=new TrafficInvestigator('search');
            $model->unsetAttributes();  // clear any default values
            if(isset($_GET['TrafficInvestigator']))
                  $model->attributes=$_GET['TrafficInvestigator'];

            $this->render('admin',array(
                  'model'=>$model,
            ));
      }

      /**
       * Returns the data model based on the primary key given in the GET
variable.
       * If the data model is not found, an HTTP exception will be raised.
       * @param integer $id the ID of the model to be loaded
       * @return TrafficInvestigator the loaded model
       * @throws CHttpException
       */
      public function loadModel($id)
      {
            $model=TrafficInvestigator::model()->findByPk($id);
            if($model===null)
                  throw new CHttpException(404,'The requested page does not
exist.');
            return $model;
      }
```

```
	/**
	 * Performs the AJAX validation.
	 * @param TrafficInvestigator $model the model to be validated
	 */
	protected function performAjaxValidation($model)
	{
		if(isset($_POST['ajax']) && $_POST['ajax']==='traffic-
investigator-form')
		{
			echo CActiveForm::validate($model);
			Yii::app()->end();
		}
	}

	public function getStatusName($id){
		return TrafficInvestigatorStatus::model()->findByPk($id)-
>NAME;

	}

	public function getName($id){
		$userDetails = AccountableUser::model()->findByPk($id);
		$firstName = $userDetails->FIRST_NAME;
		$lastName = $userDetails->LAST_NAME;
		$initial = $userDetails->USER_INITIAL;
		return $firstName.' '.$initial.'. '.$lastName;
	}

	public function actionUpdateCoordinatesMobile()
	{


		$userId = $_POST['id'];
		$latitude = $_POST['latitude'];
		$longitude = $_POST['longitude'];
		//$userId = 6;
		//$latitude = 14.44975;
		//$longitude = 120.930945;

		$tiModel=TrafficInvestigator::model()->findByPk($userId);
		$tiModel->CURRENT_LAT = $latitude;
		$tiModel->CURRENT_LNG = $longitude;

		if($tiModel->save()){
			echo 1;
		}else{
			echo 0;
		}

	}

	public function actionLogoutMobile(){
		$userId = $_POST['id'];
		//$userId = 6;
```

```
            $tiModel=TrafficInvestigator::model()->findByPk($userId);
            $tiModel->IS_ACTIVE = 0;

            if($tiModel->save()){
                  echo 1;
            }else{
                  echo 0;
            }
      }

      public function getJurisdictions(){
            $jurisdictionArray = Jurisdiction::model()->findAll();
            $statusArray = CHtml::listData($jurisdictionArray,
'JURISDICTION_ID', 'NAME');
            return $statusArray;
      }

      public function getJurisdictionName($id){
            return $jurisdictionArray = Jurisdiction::model()-
>findByPk($id)->NAME;
      }
}

<?php

class TrafficInvestigatorStatusController extends Controller
{
      /**
       * @var string the default layout for the views. Defaults to
'//layouts/column2', meaning
       * using two-column layout. See
'protected/views/layouts/column2.php'.
       */
      public $layout='//layouts/column2';

      /**
       * @return array action filters
       */
      public function filters()
      {
            return array(
                  'accessControl', // perform access control for CRUD
operations
                  'postOnly + delete', // we only allow deletion via POST
request
            );
      }

      /**
       * Specifies the access control rules.
       * This method is used by the 'accessControl' filter.
       * @return array access control rules
       */
      public function accessRules()
```

```
	{
		return array(
			array('allow',  // allow all users to perform 'index' and 
'view' actions
				'actions'=>array('index','view'),
				'users'=>array('*'),
			),
			array('allow', // allow authenticated user to perform 
'create' and 'update' actions
				'actions'=>array('create','update'),
				'users'=>array('@'),
			),
			array('allow', // allow admin user to perform 'admin' and 
'delete' actions
				'actions'=>array('admin','delete'),
				'users'=>array('admin'),
			),
			array('deny',  // deny all users
				'users'=>array('*'),
			),
		);
	}

	/**
	 * Displays a particular model.
	 * @param integer $id the ID of the model to be displayed
	 */
	public function actionView($id)
	{
		$this->render('view',array(
			'model'=>$this->loadModel($id),
		));
	}

	/**
	 * Creates a new model.
	 * If creation is successful, the browser will be redirected to the 
'view' page.
	 */
	public function actionCreate()
	{
		$model=new TrafficInvestigatorStatus;

		// Uncomment the following line if AJAX validation is needed
		// $this->performAjaxValidation($model);

		if(isset($_POST['TrafficInvestigatorStatus']))
		{
			$model->attributes=$_POST['TrafficInvestigatorStatus'];
			if($model->save())
				$this->redirect(array('view','id'=>$model-
>TRAFFIC_INVESTIGATOR_STATUS_ID));
		}
```

```
            $this->render('create',array(
                  'model'=>$model,
            ));
      }

      /**
       * Updates a particular model.
       * If update is successful, the browser will be redirected to the
'view' page.
       * @param integer $id the ID of the model to be updated
       */
      public function actionUpdate($id)
      {
            $model=$this->loadModel($id);

            // Uncomment the following line if AJAX validation is needed
            // $this->performAjaxValidation($model);

            if(isset($_POST['TrafficInvestigatorStatus']))
            {
                  $model->attributes=$_POST['TrafficInvestigatorStatus'];
                  if($model->save())
                        $this->redirect(array('view','id'=>$model-
>TRAFFIC_INVESTIGATOR_STATUS_ID));
            }

            $this->render('update',array(
                  'model'=>$model,
            ));
      }

      /**
       * Deletes a particular model.
       * If deletion is successful, the browser will be redirected to the
'admin' page.
       * @param integer $id the ID of the model to be deleted
       */
      public function actionDelete($id)
      {
            $this->loadModel($id)->delete();

            // if AJAX request (triggered by deletion via admin grid
view), we should not redirect the browser
            if(!isset($_GET['ajax']))
                  $this->redirect(isset($_POST['returnUrl']) ?
$_POST['returnUrl'] : array('admin'));
      }

      /**
       * Lists all models.
       */
      public function actionIndex()
      {
```

```
            $dataProvider=new
CActiveDataProvider('TrafficInvestigatorStatus');
            $this->render('index',array(
                  'dataProvider'=>$dataProvider,
            ));
      }

      /**
       * Manages all models.
       */
      public function actionAdmin()
      {
            $model=new TrafficInvestigatorStatus('search');
            $model->unsetAttributes();  // clear any default values
            if(isset($_GET['TrafficInvestigatorStatus']))
                  $model->attributes=$_GET['TrafficInvestigatorStatus'];

            $this->render('admin',array(
                  'model'=>$model,
            ));
      }

      /**
       * Returns the data model based on the primary key given in the GET
variable.
       * If the data model is not found, an HTTP exception will be raised.
       * @param integer $id the ID of the model to be loaded
       * @return TrafficInvestigatorStatus the loaded model
       * @throws CHttpException
       */
      public function loadModel($id)
      {
            $model=TrafficInvestigatorStatus::model()->findByPk($id);
            if($model===null)
                  throw new CHttpException(404,'The requested page does not
exist.');
            return $model;
      }

      /**
       * Performs the AJAX validation.
       * @param TrafficInvestigatorStatus $model the model to be validated
       */
      protected function performAjaxValidation($model)
      {
            if(isset($_POST['ajax']) && $_POST['ajax']==='traffic-
investigator-status-form')
            {
                  echo CActiveForm::validate($model);
                  Yii::app()->end();
            }
      }
}
```

```
<?php

class TrafficLocationPointController extends Controller
{
	/**
	 * @var string the default layout for the views. Defaults to
'//layouts/column2', meaning
	 * using two-column layout. See
'protected/views/layouts/column2.php'.
	 */
	public $layout='//layouts/column2';

	/**
	 * @return array action filters
	 */
	public function filters()
	{
		return array(
			'accessControl', // perform access control for CRUD
operations
			'postOnly + delete', // we only allow deletion via POST
request
		);
	}

	/**
	 * Specifies the access control rules.
	 * This method is used by the 'accessControl' filter.
	 * @return array access control rules
	 */
	public function accessRules()
	{
		return array(
			array('allow',  // allow all users to perform 'index' and
'view' actions

	'actions'=>array('index','view','CreateNewDataMobile'),
					'users'=>array('*'),
			),
			array('allow', // allow authenticated user to perform
'create' and 'update' actions
					'actions'=>array('create','update'),
					'users'=>array('@'),
			),
			array('allow', // allow admin user to perform 'admin' and
'delete' actions
					'actions'=>array('admin','delete'),
					'users'=>array('admin'),
			),
			array('deny',  // deny all users
					'users'=>array('*'),
			),
		);
	}
```

```
	/**
	 * Displays a particular model.
	 * @param integer $id the ID of the model to be displayed
	 */
	public function actionView($id)
	{
		$this->render('view',array(
			'model'=>$this->loadModel($id),
		));
	}

	/**
	 * Creates a new model.
	 * If creation is successful, the browser will be redirected to the
'view' page.
	 */
	public function actionCreate()
	{
		$model=new TrafficLocationPoint;

		// Uncomment the following line if AJAX validation is needed
		// $this->performAjaxValidation($model);

		if(isset($_POST['TrafficLocationPoint']))
		{
			$model->attributes=$_POST['TrafficLocationPoint'];
			if($model->save())
				$this->redirect(array('view','id'=>$model-
>TRAFFIC_LOCATION_POINT_ID));
		}

		$this->render('create',array(
			'model'=>$model,
		));
	}

	/**
	 * Updates a particular model.
	 * If update is successful, the browser will be redirected to the
'view' page.
	 * @param integer $id the ID of the model to be updated
	 */
	public function actionUpdate($id)
	{
		$model=$this->loadModel($id);

		// Uncomment the following line if AJAX validation is needed
		// $this->performAjaxValidation($model);

		if(isset($_POST['TrafficLocationPoint']))
		{
			$model->attributes=$_POST['TrafficLocationPoint'];
			if($model->save())
```

```
                        $this->redirect(array('view','id'=>$model-
>TRAFFIC_LOCATION_POINT_ID));
            }

            $this->render('update',array(
                  'model'=>$model,
            ));
      }

      /**
       * Deletes a particular model.
       * If deletion is successful, the browser will be redirected to the
'admin' page.
       * @param integer $id the ID of the model to be deleted
       */
      public function actionDelete($id)
      {
            $this->loadModel($id)->delete();

            // if AJAX request (triggered by deletion via admin grid
view), we should not redirect the browser
            if(!isset($_GET['ajax']))
                  $this->redirect(isset($_POST['returnUrl']) ?
$_POST['returnUrl'] : array('admin'));
      }

      /**
       * Lists all models.
       */
      public function actionIndex()
      {
            $dataProvider=new CActiveDataProvider('TrafficLocationPoint');
            $this->render('index',array(
                  'dataProvider'=>$dataProvider,
            ));
      }

      /**
       * Manages all models.
       */
      public function actionAdmin()
      {
            $model=new TrafficLocationPoint('search');
            $model->unsetAttributes();  // clear any default values
            if(isset($_GET['TrafficLocationPoint']))
                  $model->attributes=$_GET['TrafficLocationPoint'];

            $this->render('admin',array(
                  'model'=>$model,
            ));
      }

      /**
```

```
	 * Returns the data model based on the primary key given in the GET
variable.
	 * If the data model is not found, an HTTP exception will be raised.
	 * @param integer $id the ID of the model to be loaded
	 * @return TrafficLocationPoint the loaded model
	 * @throws CHttpException
	 */
	public function loadModel($id)
	{
		$model=TrafficLocationPoint::model()->findByPk($id);
		if($model===null)
			throw new CHttpException(404,'The requested page does not
exist.');
		return $model;
	}

	/**
	 * Performs the AJAX validation.
	 * @param TrafficLocationPoint $model the model to be validated
	 */
	protected function performAjaxValidation($model)
	{
		if(isset($_POST['ajax']) && $_POST['ajax']==='traffic-
location-point-form')
		{
			echo CActiveForm::validate($model);
			Yii::app()->end();
		}
	}

	public function actionCreateNewDataMobile()
	{


		$id = $_POST['id'];
		$lat = $_POST['latitude'];
		$lon = $_POST['longitude'];
		$speed = $_POST['speed'];


		/*
		$id = 2;
		$lat = 1;
		$lon = 1;
		$speed = 1;
		echo '{'.$lat.'.}';
		*/
		$model = new TrafficLocationPoint;

		$model->LATITUDE = $lat;
		$model->LONGITUDE = $lon;
		$model->DATE_CREATED = new CDbExpression('NOW()');
		$model->DATA_LOCATION_COLLECTION_ID = $id;
		$model->SPEED = $speed;
```

```
            if($model->save()){
                  echo 'success.';
            }else{
                  echo 'fail.';
            }

      }
}

<?php

class TrafficOperationsCenterOfficerController extends Controller
{
      /**
       * @var string the default layout for the views. Defaults to
'//layouts/column2', meaning
       * using two-column layout. See
'protected/views/layouts/column2.php'.
       */
      public $layout='//layouts/column2';

      /**
       * @return array action filters
       */
      public function filters()
      {
            return array(
                  'accessControl', // perform access control for CRUD
operations
                  'postOnly + delete', // we only allow deletion via POST
request
            );
      }

      /**
       * Specifies the access control rules.
       * This method is used by the 'accessControl' filter.
       * @return array access control rules
       */
      public function accessRules()
      {
            return array(
                  array('allow',  // allow all users to perform 'index' and
'view' actions
                        'actions'=>array('index','view'),
                        'users'=>array('*'),
                  ),
                  array('allow', // allow authenticated user to perform
'create' and 'update' actions
                        'actions'=>array('create','update'),
                        'users'=>array('@'),
                  ),
```

```
                array('allow', // allow admin user to perform 'admin' and
'delete' actions
                      'actions'=>array('admin','delete'),
                      'users'=>array('admin'),
                ),
                array('deny',  // deny all users
                      'users'=>array('*'),
                ),
          );
     }

     /**
      * Displays a particular model.
      * @param integer $id the ID of the model to be displayed
      */
     public function actionView($id)
     {
          $this->render('view',array(
                'model'=>$this->loadModel($id),
          ));
     }

     /**
      * Creates a new model.
      * If creation is successful, the browser will be redirected to the
'view' page.
      */
     public function actionCreate()
     {
          if(!Yii::app()->user->checkAccess('manageAccounts')){
                throw new CHttpException(403, 'You are not authorize to
access this page');
          }

          $modelTrafficOperationsCenterOfficer=new
TrafficOperationsCenterOfficer;
          $modelAccountableUser=new AccountableUser;
          $modelUsers= new Users;


          // Uncomment the following line if AJAX validation is needed
          // $this->performAjaxValidation($model);

          if(isset($_POST['AccountableUser']))
          {
                $modelAccountableUser-
>attributes=$_POST['AccountableUser'];

                $attributes =
array('FIRST_NAME','LAST_NAME','USER_INITIAL','PASSWORD','BIRTHDAY','USERN
AME');

                if($modelAccountableUser->validate($attributes)){
```

```
                        $modelUsers->DATE_REGISTERED = new
CDbExpression('NOW()');
                        $modelUsers->LAST_LOGIN = new
CDbExpression('NOW()');
                        if($modelUsers->save()){
                              $modelAccountableUser->USER_ID = $modelUsers-
>USER_ID;
                              if($modelAccountableUser->save()){
                                    $modelTrafficOperationsCenterOfficer-
>USER_ID = $modelUsers->USER_ID;

      if($modelTrafficOperationsCenterOfficer->save()){
                                          $auth=Yii::app()->authManager;
                                          $auth-
>assign('trafficOperationsCenterOfficer',$modelUsers->USER_ID);
                                          $this-
>redirect(array('view','id'=>$modelTrafficOperationsCenterOfficer-
>USER_ID));
                                    }
                              }
                        }
                  }
            }

            $this->render('create',array(

      'modelTrafficOperationsCenterOfficer'=>$modelTrafficOperationsCenter
Officer,
                  'modelAccountableUser'=>$modelAccountableUser,
                  'modelUsers'=>$modelUsers,
            ));
      }

      /**
       * Updates a particular model.
       * If update is successful, the browser will be redirected to the
'view' page.
       * @param integer $id the ID of the model to be updated
       */
      public function actionUpdate($id)
      {

            if(!Yii::app()->user->checkAccess('manageAccounts')){
                  throw new CHttpException(403, 'You are not authorize to
access this page');
            }
            $modelTrafficOperationsCenterOfficer=$this->loadModel($id);
            $modelAccountableUser= AccountableUser::model()-
>findByPk($id);
            $modelUsers= Users::model()->findByPk($id);
            //$model=$this->loadModel($id);

            // Uncomment the following line if AJAX validation is needed
            // $this->performAjaxValidation($model);
```

```
            if(isset($_POST['TrafficOperationsCenterOfficer']))
            {
                  $model-
>attributes=$_POST['TrafficOperationsCenterOfficer'];
                  if($model->save())
                        $this->redirect(array('view','id'=>$model-
>USER_ID));
            }

            $this->render('update',array(

      'modelTrafficOperationsCenterOfficer'=>$modelTrafficOperationsCenter
Officer,
                  'modelAccountableUser'=>$modelAccountableUser,
                  'modelUsers'=>$modelUsers,
            ));
      }

      /**
       * Deletes a particular model.
       * If deletion is successful, the browser will be redirected to the
'admin' page.
       * @param integer $id the ID of the model to be deleted
       */
      public function actionDelete($id)
      {
            if(!Yii::app()->user->checkAccess('manageAccounts')){
                  throw new CHttpException(403, 'You are not authorize to
access this page');
            }

            TrafficOperationsCenterOfficer::model()->deleteByPk($id);
            AccountableUser::model()->deleteByPk($id);
            Users::model()->deleteByPk($id);

            // if AJAX request (triggered by deletion via admin grid
view), we should not redirect the browser
            if(!isset($_GET['ajax']))
                  $this->redirect(isset($_POST['returnUrl']) ?
$_POST['returnUrl'] : array('admin'));
      }

      /**
       * Lists all models.
       */
      public function actionIndex()
      {
            if(!Yii::app()->user->checkAccess('manageAccounts')){
                  throw new CHttpException(403, 'You are not authorize to
access this page');
            }
```

```
            $dataProvider=new
CActiveDataProvider('TrafficOperationsCenterOfficer');
            $this->render('index',array(
                  'dataProvider'=>$dataProvider,
            ));
      }

      /**
       * Manages all models.
       */
      public function actionAdmin()
      {
            if(!Yii::app()->user->checkAccess('manageAccounts')){
                  throw new CHttpException(403, 'You are not authorize to
access this page');
            }

            $model=new TrafficOperationsCenterOfficer('search');
            $model->unsetAttributes();  // clear any default values
            if(isset($_GET['TrafficOperationsCenterOfficer']))
                  $model-
>attributes=$_GET['TrafficOperationsCenterOfficer'];

            $this->render('admin',array(
                  'model'=>$model,
            ));
      }

      /**
       * Returns the data model based on the primary key given in the GET
variable.
       * If the data model is not found, an HTTP exception will be raised.
       * @param integer $id the ID of the model to be loaded
       * @return TrafficOperationsCenterOfficer the loaded model
       * @throws CHttpException
       */
      public function loadModel($id)
      {
            $model=TrafficOperationsCenterOfficer::model()->findByPk($id);
            if($model===null)
                  throw new CHttpException(404,'The requested page does not
exist.');
            return $model;
      }

      /**
       * Performs the AJAX validation.
       * @param TrafficOperationsCenterOfficer $model the model to be
validated
       */
      protected function performAjaxValidation($model)
      {
            if(isset($_POST['ajax']) && $_POST['ajax']==='traffic-
operations-center-officer-form')
```

```
            {
                  echo CActiveForm::validate($model);
                  Yii::app()->end();
            }
      }

      public function getName($id){
            $userDetails = AccountableUser::model()->findByPk($id);
            $firstName = $userDetails->FIRST_NAME;
            $lastName = $userDetails->LAST_NAME;
            $initial = $userDetails->USER_INITIAL;
            return $firstName.' '.$initial.'. '.$lastName;
      }
}

<?php

class UsersController extends Controller
{
      /**
       * @var string the default layout for the views. Defaults to
'//layouts/column2', meaning
       * using two-column layout. See
'protected/views/layouts/column2.php'.
       */
      public $layout='//layouts/column2';

      /**
       * @return array action filters
       */
      public function filters()
      {
            return array(
                  'accessControl', // perform access control for CRUD
operations
                  'postOnly + delete', // we only allow deletion via POST
request
            );
      }

      /**
       * Specifies the access control rules.
       * This method is used by the 'accessControl' filter.
       * @return array access control rules
       */
      public function accessRules()
      {
            return array(
                  array('allow',  // allow all users to perform 'index' and
'view' actions
                        'actions'=>array('index','view'),
                        'users'=>array('*'),
                  ),
```

```
                array('allow', // allow authenticated user to perform
'create' and 'update' actions
                      'actions'=>array('create','update'),
                      'users'=>array('@'),
                ),
                array('allow', // allow admin user to perform 'admin' and
'delete' actions
                      'actions'=>array('admin','delete'),
                      'users'=>array('admin'),
                ),
                array('deny',  // deny all users
                      'users'=>array('*'),
                ),
          );
     }

     /**
      * Displays a particular model.
      * @param integer $id the ID of the model to be displayed
      */
     public function actionView($id)
     {
          $this->render('view',array(
                'model'=>$this->loadModel($id),
          ));
     }

     /**
      * Creates a new model.
      * If creation is successful, the browser will be redirected to the
'view' page.
      */
     public function actionCreate()
     {
          $model=new Users;

          // Uncomment the following line if AJAX validation is needed
          // $this->performAjaxValidation($model);

          if(isset($_POST['Users']))
          {
                $model->attributes=$_POST['Users'];
                if($model->save())
                      $this->redirect(array('view','id'=>$model-
>USER_ID));
          }

          $this->render('create',array(
                'model'=>$model,
          ));
     }

     /**
      * Updates a particular model.
```

```
	 * If update is successful, the browser will be redirected to the
'view' page.
	 * @param integer $id the ID of the model to be updated
	 */
	public function actionUpdate($id)
	{
		$model=$this->loadModel($id);

		// Uncomment the following line if AJAX validation is needed
		// $this->performAjaxValidation($model);

		if(isset($_POST['Users']))
		{
			$model->attributes=$_POST['Users'];
			if($model->save())
				$this->redirect(array('view','id'=>$model-
>USER_ID));
		}

		$this->render('update',array(
			'model'=>$model,
		));
	}

	/**
	 * Deletes a particular model.
	 * If deletion is successful, the browser will be redirected to the
'admin' page.
	 * @param integer $id the ID of the model to be deleted
	 */
	public function actionDelete($id)
	{
		$this->loadModel($id)->delete();

		// if AJAX request (triggered by deletion via admin grid
view), we should not redirect the browser
		if(!isset($_GET['ajax']))
			$this->redirect(isset($_POST['returnUrl']) ?
$_POST['returnUrl'] : array('admin'));
	}

	/**
	 * Lists all models.
	 */
	public function actionIndex()
	{
		$dataProvider=new CActiveDataProvider('Users');
		$this->render('index',array(
			'dataProvider'=>$dataProvider,
		));
	}

	/**
	 * Manages all models.
```

```
	 */
	public function actionAdmin()
	{
		$model=new Users('search');
		$model->unsetAttributes();  // clear any default values
		if(isset($_GET['Users']))
			$model->attributes=$_GET['Users'];

		$this->render('admin',array(
			'model'=>$model,
		));
	}

	/**
	 * Returns the data model based on the primary key given in the GET
variable.
	 * If the data model is not found, an HTTP exception will be raised.
	 * @param integer $id the ID of the model to be loaded
	 * @return Users the loaded model
	 * @throws CHttpException
	 */
	public function loadModel($id)
	{
		$model=Users::model()->findByPk($id);
		if($model===null)
			throw new CHttpException(404,'The requested page does not
exist.');
		return $model;
	}

	/**
	 * Performs the AJAX validation.
	 * @param Users $model the model to be validated
	 */
	protected function performAjaxValidation($model)
	{
		if(isset($_POST['ajax']) && $_POST['ajax']==='users-form')
		{
			echo CActiveForm::validate($model);
			Yii::app()->end();
		}
	}
}

<?php

class VirtualTripLineController extends Controller
{
	/**
	 * @var string the default layout for the views. Defaults to
'//layouts/column2', meaning
	 * using two-column layout. See
'protected/views/layouts/column2.php'.
	 */
```

```
	public $layout='//layouts/column2';

	/**
	 * @return array action filters
	 */
	public function filters()
	{
		return array(
			'accessControl', // perform access control for CRUD
operations
			'postOnly + delete', // we only allow deletion via POST
request
		);
	}

	/**
	 * Specifies the access control rules.
	 * This method is used by the 'accessControl' filter.
	 * @return array access control rules
	 */
	public function accessRules()
	{
		return array(
			array('allow',  // allow all users to perform 'index' and
'view' actions

	'actions'=>array('index','view','GetTrafficDataMobile','CheckIfInVTL
'),
				'users'=>array('*'),
			),
			array('allow', // allow authenticated user to perform
'create' and 'update' actions

	'actions'=>array('create','update','admin','delete'),
				'users'=>array('@'),
			),
			array('allow', // allow admin user to perform 'admin' and
'delete' actions
				//'actions'=>array('admin','delete'),
				//'users'=>array('admin'),
			),
			array('deny',  // deny all users
				'users'=>array('*'),
			),
		);
	}

	/**
	 * Displays a particular model.
	 * @param integer $id the ID of the model to be displayed
	 */
	public function actionView($id)
	{
		if(!Yii::app()->user->checkAccess('readVirtualTripLines')){
```

```
                throw new CHttpException(403, 'You are not authorize to
access this page');
            }

            $this->render('view',array(
                  'model'=>$this->loadModel($id),
            ));
      }

      /**
       * Creates a new model.
       * If creation is successful, the browser will be redirected to the
'view' page.
       */
      public function actionCreate()
      {
            if(!Yii::app()->user->checkAccess('createVirtualTripLines')){
                  throw new CHttpException(403, 'You are not authorize to
access this page');
            }

            $model=new VirtualTripLine;

            // Uncomment the following line if AJAX validation is needed
            // $this->performAjaxValidation($model);

            if(isset($_POST['VirtualTripLine']))
            {
                  $model->attributes=$_POST['VirtualTripLine'];
                  if($model->save())
                        $this->redirect(array('view','id'=>$model-
>VIRTUAL_TRIP_LINE_ID));
            }

            $this->render('create',array(
                  'model'=>$model,
            ));
      }

      /**
       * Updates a particular model.
       * If update is successful, the browser will be redirected to the
'view' page.
       * @param integer $id the ID of the model to be updated
       */
      public function actionUpdate($id)
      {

            if(!Yii::app()->user->checkAccess('updateVirtualTripLines')){
                  throw new CHttpException(403, 'You are not authorize to
access this page');
            }

            $model=$this->loadModel($id);
```

```
            // Uncomment the following line if AJAX validation is needed
            // $this->performAjaxValidation($model);

            if(isset($_POST['VirtualTripLine']))
            {
                  $model->attributes=$_POST['VirtualTripLine'];
                  if($model->save())
                        $this->redirect(array('view','id'=>$model-
>VIRTUAL_TRIP_LINE_ID));
            }

            $this->render('update',array(
                  'model'=>$model,
            ));
      }

      /**
       * Deletes a particular model.
       * If deletion is successful, the browser will be redirected to the
'admin' page.
       * @param integer $id the ID of the model to be deleted
       */
      public function actionDelete($id)
      {

            if(!Yii::app()->user->checkAccess('deleteVirtualTripLines')){
                  throw new CHttpException(403, 'You are not authorize to
access this page');
            }

            $this->loadModel($id)->delete();

            // if AJAX request (triggered by deletion via admin grid
view), we should not redirect the browser
            if(!isset($_GET['ajax']))
                  $this->redirect(isset($_POST['returnUrl']) ?
$_POST['returnUrl'] : array('index'));
      }

      /**
       * Lists all models.
       */
      public function actionIndex()
      {
            if(!Yii::app()->user->checkAccess('readVirtualTripLines')){
                  throw new CHttpException(403, 'You are not authorize to
access this page');
            }

            $dataProvider=new CActiveDataProvider('VirtualTripLine');
            $this->render('index',array(
                  'dataProvider'=>$dataProvider,
            ));
```

```
	}

	/**
	 * Manages all models.
	 */
	public function actionAdmin()
	{

		if(!Yii::app()->user->checkAccess('readVirtualTripLine')){
			throw new CHttpException(403, 'You are not authorize to
access this page');
		}

		$model=new VirtualTripLine('search');
		$model->unsetAttributes();  // clear any default values
		if(isset($_GET['VirtualTripLine']))
			$model->attributes=$_GET['VirtualTripLine'];

		$this->render('admin',array(
			'model'=>$model,
		));
	}

	/**
	 * Returns the data model based on the primary key given in the GET
variable.
	 * If the data model is not found, an HTTP exception will be raised.
	 * @param integer $id the ID of the model to be loaded
	 * @return VirtualTripLine the loaded model
	 * @throws CHttpException
	 */
	public function loadModel($id)
	{
		$model=VirtualTripLine::model()->findByPk($id);
		if($model===null)
			throw new CHttpException(404,'The requested page does not
exist.');
		return $model;
	}

	/**
	 * Performs the AJAX validation.
	 * @param VirtualTripLine $model the model to be validated
	 */
	protected function performAjaxValidation($model)
	{
		if(isset($_POST['ajax']) && $_POST['ajax']==='virtual-trip-
line-form')
		{
			echo CActiveForm::validate($model);
			Yii::app()->end();
		}
	}
```

```
	protected function setLowestValueToZero($dataArray){
		$min = $dataArray[0][0];
		foreach($dataArray as $data){
			if($min > $data[0]){
				$min = $data[0];
			}
		}

		$size = count($dataArray);
		for($i = 0; $i < $size; $i++){
			$dataArray[$i][0] = $dataArray[$i][0] - $min;
		}

		return $dataArray;
	}


	public function actionGetTrafficDataMobile(){
		include 'DBScan.php';
		$resultData = array();
		$virtualTripLine = VirtualTripLine::model()->findAll();

		foreach ($virtualTripLine as $data){
			$speedData = array();
			$dataLocationCollection =
DataLocationCollection::model()-
>findAllByAttributes(array('VIRTUAL_TRIP_LINE_ID'=>$data-
>VIRTUAL_TRIP_LINE_ID));
			foreach($dataLocationCollection as $data2){

				$qry = '
				SELECT *
				FROM traffic_location_point
				WHERE DATE_CREATED BETWEEN NOW() - INTERVAL 1 DAY
AND NOW()
				AND DATA_LOCATION_COLLECTION_ID =
'.$data2['DATA_LOCATION_COLLECTION_ID'];
				$trafficLocationPoint =
TrafficLocationPoint::model()->findAllBySql($qry);
				foreach($trafficLocationPoint as $data3){
					$speedData[]=array(

	'trafficId'=>$data3['TRAFFIC_LOCATION_POINT_ID'],
						'speed'=>$data3['SPEED'],

	'dataCollectionId'=>$data2['DATA_LOCATION_COLLECTION_ID'],
						'id'=>$data2['VIRTUAL_TRIP_LINE_ID'],
						'dateCreated'=>$data3['DATE_CREATED']
					);
				}


			}
```

```
                  //prepare data for dbscan

                  $dataForDBSCAN = array();
                  foreach($speedData as $speed){

      $timestamp=CDateTimeParser::parse($speed['dateCreated'],'yyyy-MM-dd
hh:mm:ss');
                        $temp = array($timestamp,$speed['speed']);
                        array_push($dataForDBSCAN,$temp);

                  }

                  if(count($dataForDBSCAN) != 0){
                        $dataForDBSCAN = $this-
>setLowestValueToZero($dataForDBSCAN);
                  }else{
                        //no data
                  }

                  //end prepare data for dbscan

                  //start dbscan
                  /*
                  $dbScan = new DBScan();
                  $outliers;

                  if(count($dataForDBSCAN) != 0){
                        $outliers = $dbScan-
>Get_Outliers($dataForDBSCAN,1,8);
                        //remove outliers from data
                        foreach($outliers as $data){
                              echo 'outlier'.$data.'<br/>';
                              unset($dataForDBSCAN[$data]);
                        }

                  }else{
                        //no data
                  }
                  */
                  //end dbscan


                  $averageSpeed = 0;
                  foreach($dataForDBSCAN as $temp){
                        $averageSpeed = $averageSpeed + $temp[1];
                  }

                  if(count($dataForDBSCAN) != 0){
                        $averageSpeed =
$averageSpeed/count($dataForDBSCAN);
                  }else{
                        $averageSpeed = null;
                  }
                  $resultData[]=array(
```

```
                        'vtlid'=>$data->VIRTUAL_TRIP_LINE_ID,
                        'averageSpeed'=>$averageSpeed,
                        'marker1_latitude'=>$data->marker1_latitude,
                        'marker1_longitude'=>$data->marker1_longitude,
                        'marker2_latitude'=>$data->marker2_latitude,
                        'marker2_longitude'=>$data->marker2_longitude
                  );
            }

            echo json_encode($resultData);
      }

      public function actionCheckIfInVTL(){
            //echo 'checkin VTL';
            //14.4507418,120.9315033
            //$vtlID = $_POST['vtlid'];
            //$latitude = $_POST['lat'];
            //$longitude = $_POST['lon'];
            //echo $latitude.','.$longitude;
            $vtlID = 3;
            $latitude = '14.4507418';
            $longitude = '120.9315033';
            $vtlModel = VirtualTripLine::model()->findByPk($vtlID);
            if($this->isInsideRectangle($vtlModel->LL_X,$vtlModel-
>LL_Y,$vtlModel->LR_X,$vtlModel->LR_Y,$vtlModel->UL_X,$vtlModel-
>UL_Y,$vtlModel->UR_X,$vtlModel->UR_Y,$latitude,$longitude)){
                  echo 1;
            }else{
                  echo 0;
            }
      }

      public function
isInsideRectangle($LLX,$LLY,$LRX,$LRY,$ULX,$ULY,$URX,$URY,$x,$y){
            $LOWERBOUND = -1;
            $UPPERBOUND = -1;
            $LEFTBOUND = -1;
            $RIGHTBOUND = -1;

            //check lower bound
            //$a = ($LLY-$LRY);

            $a = -($LLY - $LRY);
            $b = $LLX-$LRX;
            $c = -($a*$LRX + $b*$LRY);
            $d = $a*$x + $b*$y + $c;

            if($d<0){
                  $LOWERBOUND = 0;

            }else{
                  //return false;
                  $LOWERBOUND = 1;
            }
```

```
            //check upper bound
            $a = -($ULY - $URY);
            $b = $ULX-$URX;
            $c = -($a*$URX + $b*$URY);
            $d = $a*$x + $b*$y + $c;
            if($d<0){
                  $UPPERBOUND = 1;

            }else{
                  //return false;
                  $UPPERBOUND = 0;
            }

            //check left bound
            $a = -($ULY - $LLY);
            $b = $ULX-$LLX;
            $c = -($a*$LLX + $b*$LLY);
            $d = $a*$x + $b*$y + $c;
            if($d<0){
                  $LEFTBOUND = 0;

            }else{
                  //return false;
                  $LEFTBOUND = 1;
            }

            //check right bound
            $a = -($URY - $LRY);
            $b = $URX-$LRX;
            $c = -($a*$LRX + $b*$LRY);
            $d = $a*$x + $b*$y + $c;
            if($d<0){
                  $RIGHTBOUND = 1;

            }else{
                  //return false;
                  $RIGHTBOUND = 0;
            }


      if($LOWERBOUND==1&&$UPPERBOUND==1&&$LEFTBOUND==1&&$RIGHTBOUND==1){
                  return true;
            }else{
                  return false;
            }
      }

}

<?php

/**
 * This is the model class for table "accountable_users".
```

```
 *
 * The followings are the available columns in table 'accountable_users':
 * @property integer $USER_ID
 * @property string $FIRST_NAME
 * @property string $LAST_NAME
 * @property string $USER_INITIAL
 * @property string $BIRTHDAY
 * @property string $USERNAME
 * @property string $PASSWORD
 *
 * The followings are the available model relations:
 * @property Users $uSER
 * @property SystemAdministrator $systemAdministrator
 * @property TrafficInvestigator $trafficInvestigator
 * @property TrafficOperationsCenterOfficer
$trafficOperationsCenterOfficer
 */
class AccountableUser extends CActiveRecord
{

	public $PASSWORD_repeat;

	/**
	 * Returns the static model of the specified AR class.
	 * @param string $className active record class name.
	 * @return AccountableUser the static model class
	 */
	public static function model($className=__CLASS__)
	{
		return parent::model($className);
	}

	/**
	 * @return string the associated database table name
	 */
	public function tableName()
	{
		return 'accountable_users';
	}

	/**
	 * @return array validation rules for model attributes.
	 */
	public function rules()
	{
		// NOTE: you should only define rules for those attributes
that
		// will receive user inputs.
		return array(
			array('USER_ID, FIRST_NAME, LAST_NAME, USER_INITIAL,
BIRTHDAY, USERNAME, PASSWORD', 'required'),
			array('USER_ID', 'numerical', 'integerOnly'=>true),
			array('USERNAME', 'length', 'max'=>200),
			array('USERNAME', 'unique'),
```

```
                array('PASSWORD', 'compare'),
                array('PASSWORD_repeat', 'safe'),
                // The following rule is used by search().
                // Please remove those attributes that should not be
searched.
                array('USER_ID, FIRST_NAME, LAST_NAME, USER_INITIAL,
BIRTHDAY, USERNAME, PASSWORD', 'safe', 'on'=>'search'),
          );
     }

     /**
      * @return array relational rules.
      */
     public function relations()
     {
          // NOTE: you may need to adjust the relation name and the
related
          // class name for the relations automatically generated below.
          return array(
                'uSER' => array(self::BELONGS_TO, 'Users', 'USER_ID'),
                'systemAdministrator' => array(self::HAS_ONE,
'SystemAdministrator', 'USER_ID'),
                'trafficInvestigator' => array(self::HAS_ONE,
'TrafficInvestigator', 'USER_ID'),
                'trafficOperationsCenterOfficer' => array(self::HAS_ONE,
'TrafficOperationsCenterOfficer', 'USER_ID'),
          );
     }

     /**
      * @return array customized attribute labels (name=>label)
      */
     public function attributeLabels()
     {
          return array(
                'USER_ID' => 'User',
                'FIRST_NAME' => 'First Name',
                'LAST_NAME' => 'Last Name',
                'USER_INITIAL' => 'User Initial',
                'BIRTHDAY' => 'Birthday',
                'USERNAME' => 'Username',
                'PASSWORD' => 'Password',
          );
     }

     /**
      * Retrieves a list of models based on the current search/filter
conditions.
      * @return CActiveDataProvider the data provider that can return the
models based on the search/filter conditions.
      */
     public function search()
     {
```

```
            // Warning: Please modify the following code to remove
attributes that
            // should not be searched.

            $criteria=new CDbCriteria;

            $criteria->compare('USER_ID',$this->USER_ID);
            $criteria->compare('FIRST_NAME',$this->FIRST_NAME,true);
            $criteria->compare('LAST_NAME',$this->LAST_NAME,true);
            $criteria->compare('USER_INITIAL',$this->USER_INITIAL,true);
            $criteria->compare('BIRTHDAY',$this->BIRTHDAY,true);
            $criteria->compare('USERNAME',$this->USERNAME,true);
            $criteria->compare('PASSWORD',$this->PASSWORD,true);

            return new CActiveDataProvider($this, array(
                  'criteria'=>$criteria,
            ));
      }

      public function beforeSave(){
         if(parent::beforeSave()){
                  $this->PASSWORD = $this->encrypt($this->PASSWORD); // if
you save dates as INT
                  return true;
         }
         return false;
      }
      /*
      protected function afterValidate(){
            parent::afterValidate();
            $this->PASSWORD=$this->encrypt($this->PASSWORD);
      }
      */
      public function encrypt($value){
            return md5($value);
      }

}
<?php

/**
 * ContactForm class.
 * ContactForm is the data structure for keeping
 * contact form data. It is used by the 'contact' action of
'SiteController'.
 */
class ContactForm extends CFormModel
{
      public $name;
      public $email;
      public $subject;
      public $body;
      public $verifyCode;
```

```
	/**
	 * Declares the validation rules.
	 */
	public function rules()
	{
		return array(
			// name, email, subject and body are required
			array('name, email, subject, body', 'required'),
			// email has to be a valid email address
			array('email', 'email'),
			// verifyCode needs to be entered correctly
			array('verifyCode', 'captcha',
'allowEmpty'=>!CCaptcha::checkRequirements()),
		);
	}

	/**
	 * Declares customized attribute labels.
	 * If not declared here, an attribute would have a label that is
	 * the same as its name with the first letter in upper case.
	 */
	public function attributeLabels()
	{
		return array(
			'verifyCode'=>'Verification Code',
		);
	}
}
<?php

/**
 * This is the model class for table "data_location_collection".
 *
 * The followings are the available columns in table
'data_location_collection':
 * @property integer $DATA_LOCATION_COLLECTION_ID
 * @property string $START_DATETIME
 * @property string $END_DATETIME
 * @property integer $USER_ID
 * @property integer $VIRTUAL_TRIP_LINE_ID
 *
 * The followings are the available model relations:
 * @property VirtualTripLine $vIRTUALTRIPLINE
 * @property TrafficLocationPoint[] $trafficLocationPoints
 */
class DataLocationCollection extends CActiveRecord
{
	/**
	 * Returns the static model of the specified AR class.
	 * @param string $className active record class name.
	 * @return DataLocationCollection the static model class
	 */
	public static function model($className=__CLASS__)
	{
```

```
		return parent::model($className);
	}

	/**
	 * @return string the associated database table name
	 */
	public function tableName()
	{
		return 'data_location_collection';
	}

	/**
	 * @return array validation rules for model attributes.
	 */
	public function rules()
	{
		// NOTE: you should only define rules for those attributes
that
		// will receive user inputs.
		return array(
			array('START_DATETIME, END_DATETIME, USER_ID,
VIRTUAL_TRIP_LINE_ID', 'required'),
			array('USER_ID, VIRTUAL_TRIP_LINE_ID', 'numerical',
'integerOnly'=>true),
			// The following rule is used by search().
			// Please remove those attributes that should not be
searched.
			array('DATA_LOCATION_COLLECTION_ID, START_DATETIME,
END_DATETIME, USER_ID, VIRTUAL_TRIP_LINE_ID', 'safe', 'on'=>'search'),
		);
	}

	/**
	 * @return array relational rules.
	 */
	public function relations()
	{
		// NOTE: you may need to adjust the relation name and the
related
		// class name for the relations automatically generated below.
		return array(
			'vIRTUALTRIPLINE' => array(self::BELONGS_TO,
'VirtualTripLine', 'VIRTUAL_TRIP_LINE_ID'),
			'trafficLocationPoints' => array(self::HAS_MANY,
'TrafficLocationPoint', 'DATA_LOCATION_COLLECTION_ID'),
		);
	}

	/**
	 * @return array customized attribute labels (name=>label)
	 */
	public function attributeLabels()
	{
		return array(
```

```
                  'DATA_LOCATION_COLLECTION_ID' => 'Data Location
Collection',
                  'START_DATETIME' => 'Start Datetime',
                  'END_DATETIME' => 'End Datetime',
                  'USER_ID' => 'User',
                  'VIRTUAL_TRIP_LINE_ID' => 'Virtual Trip Line',
            );
      }

      /**
       * Retrieves a list of models based on the current search/filter
conditions.
       * @return CActiveDataProvider the data provider that can return the
models based on the search/filter conditions.
       */
      public function search()
      {
            // Warning: Please modify the following code to remove
attributes that
            // should not be searched.

            $criteria=new CDbCriteria;

            $criteria->compare('DATA_LOCATION_COLLECTION_ID',$this-
>DATA_LOCATION_COLLECTION_ID);
            $criteria->compare('START_DATETIME',$this-
>START_DATETIME,true);
            $criteria->compare('END_DATETIME',$this->END_DATETIME,true);
            $criteria->compare('USER_ID',$this->USER_ID);
            $criteria->compare('VIRTUAL_TRIP_LINE_ID',$this-
>VIRTUAL_TRIP_LINE_ID);

            return new CActiveDataProvider($this, array(
                  'criteria'=>$criteria,
            ));
      }
}
<?php

/**
 * This is the model class for table "fs_category".
 *
 * The followings are the available columns in table 'fs_category':
 * @property integer $FS_CATEGORY_ID
 * @property string $NAME
 *
 * The followings are the available model relations:
 * @property FsReport[] $fsReports
 */
class FsCategory extends CActiveRecord
{
      /**
       * Returns the static model of the specified AR class.
       * @param string $className active record class name.
```

```
	 * @return FsCategory the static model class
	 */
	public static function model($className=__CLASS__)
	{
		return parent::model($className);
	}

	/**
	 * @return string the associated database table name
	 */
	public function tableName()
	{
		return 'fs_category';
	}

	/**
	 * @return array validation rules for model attributes.
	 */
	public function rules()
	{
		// NOTE: you should only define rules for those attributes
that
		// will receive user inputs.
		return array(
			array('NAME', 'required'),
			// The following rule is used by search().
			// Please remove those attributes that should not be
searched.
			array('FS_CATEGORY_ID, NAME', 'safe', 'on'=>'search'),
		);
	}

	/**
	 * @return array relational rules.
	 */
	public function relations()
	{
		// NOTE: you may need to adjust the relation name and the
related
		// class name for the relations automatically generated below.
		return array(
			'fsReports' => array(self::HAS_MANY, 'FsReport',
'FS_CATEGORY_ID'),
		);
	}

	/**
	 * @return array customized attribute labels (name=>label)
	 */
	public function attributeLabels()
	{
		return array(
			'FS_CATEGORY_ID' => 'Fs Category',
			'NAME' => 'Name',
```

```
		));
	}

	/**
	 * Retrieves a list of models based on the current search/filter
conditions.
	 * @return CActiveDataProvider the data provider that can return the
models based on the search/filter conditions.
	 */
	public function search()
	{
		// Warning: Please modify the following code to remove
attributes that
		// should not be searched.

		$criteria=new CDbCriteria;

		$criteria->compare('FS_CATEGORY_ID',$this->FS_CATEGORY_ID);
		$criteria->compare('NAME',$this->NAME,true);

		return new CActiveDataProvider($this, array(
			'criteria'=>$criteria,
		));
	}
}
<?php

/**
 * This is the model class for table "fs_report".
 *
 * The followings are the available columns in table 'fs_report':
 * @property integer $REPORT_ID
 * @property integer $FS_CATEGORY_ID
 *
 * The followings are the available model relations:
 * @property Report $rEPORT
 * @property FsCategory $fSCATEGORY
 */
class FsReport extends CActiveRecord
{
	/**
	 * Returns the static model of the specified AR class.
	 * @param string $className active record class name.
	 * @return FsReport the static model class
	 */
	public static function model($className=__CLASS__)
	{
		return parent::model($className);
	}

	/**
	 * @return string the associated database table name
	 */
	public function tableName()
```

```
	{
		return 'fs_report';
	}

	/**
	 * @return array validation rules for model attributes.
	 */
	public function rules()
	{
		// NOTE: you should only define rules for those attributes
that
		// will receive user inputs.
		return array(
			array('REPORT_ID, FS_CATEGORY_ID', 'required'),
			array('REPORT_ID, FS_CATEGORY_ID', 'numerical',
'integerOnly'=>true),
			// The following rule is used by search().
			// Please remove those attributes that should not be
searched.
			array('REPORT_ID, FS_CATEGORY_ID', 'safe',
'on'=>'search'),
		);
	}

	/**
	 * @return array relational rules.
	 */
	public function relations()
	{
		// NOTE: you may need to adjust the relation name and the
related
		// class name for the relations automatically generated below.
		return array(
			'rEPORT' => array(self::BELONGS_TO, 'Report',
'REPORT_ID'),
			'fSCATEGORY' => array(self::BELONGS_TO, 'FsCategory',
'FS_CATEGORY_ID'),
		);
	}

	/**
	 * @return array customized attribute labels (name=>label)
	 */
	public function attributeLabels()
	{
		return array(
			'REPORT_ID' => 'Report',
			'FS_CATEGORY_ID' => 'Fs Category',
		);
	}

	/**
	 * Retrieves a list of models based on the current search/filter
conditions.
```

```
	 * @return CActiveDataProvider the data provider that can return the
models based on the search/filter conditions.
	 */
	public function search()
	{
		// Warning: Please modify the following code to remove
attributes that
		// should not be searched.

		$criteria=new CDbCriteria;

		$criteria->compare('REPORT_ID',$this->REPORT_ID);
		$criteria->compare('FS_CATEGORY_ID',$this->FS_CATEGORY_ID);

		return new CActiveDataProvider($this, array(
			'criteria'=>$criteria,
		));
	}
}
<?php

/**
 * This is the model class for table "jurisdiction".
 *
 * The followings are the available columns in table 'jurisdiction':
 * @property integer $JURISDICTION_ID
 * @property double $LL_X
 * @property double $LL_Y
 * @property double $LR_X
 * @property double $LR_Y
 * @property double $UL_X
 * @property double $UL_Y
 * @property double $UR_X
 * @property double $UR_Y
 * @property string $NAME
 * @property string $DESCRIPTION
 *
 * The followings are the available model relations:
 * @property TrafficInvestigator[] $trafficInvestigators
 */
class Jurisdiction extends CActiveRecord
{
	/**
	 * Returns the static model of the specified AR class.
	 * @param string $className active record class name.
	 * @return Jurisdiction the static model class
	 */
	public static function model($className=__CLASS__)
	{
		return parent::model($className);
	}

	/**
	 * @return string the associated database table name
```

```
	 */
	public function tableName()
	{
		return 'jurisdiction';
	}

	/**
	 * @return array validation rules for model attributes.
	 */
	public function rules()
	{
		// NOTE: you should only define rules for those attributes
that
		// will receive user inputs.
		return array(
			array('LL_X, LL_Y, LR_X, LR_Y, UL_X, UL_Y, UR_X, UR_Y,
NAME', 'required'),
			array('LL_X, LL_Y, LR_X, LR_Y, UL_X, UL_Y, UR_X, UR_Y',
'numerical'),
			array('DESCRIPTION', 'safe'),
			// The following rule is used by search().
			// Please remove those attributes that should not be
searched.
			array('JURISDICTION_ID, LL_X, LL_Y, LR_X, LR_Y, UL_X,
UL_Y, UR_X, UR_Y, NAME, DESCRIPTION', 'safe', 'on'=>'search'),
		);
	}

	/**
	 * @return array relational rules.
	 */
	public function relations()
	{
		// NOTE: you may need to adjust the relation name and the
related
		// class name for the relations automatically generated below.
		return array(
			'trafficInvestigators' => array(self::HAS_MANY,
'TrafficInvestigator', 'JURISDICTION_ID'),
		);
	}

	/**
	 * @return array customized attribute labels (name=>label)
	 */
	public function attributeLabels()
	{
		return array(
			'JURISDICTION_ID' => 'Jurisdiction',
			'LL_X' => 'Ll X',
			'LL_Y' => 'Ll Y',
			'LR_X' => 'Lr X',
			'LR_Y' => 'Lr Y',
			'UL_X' => 'Ul X',
```

```
                'UL_Y' => 'Ul Y',
                'UR_X' => 'Ur X',
                'UR_Y' => 'Ur Y',
                'NAME' => 'Name',
                'DESCRIPTION' => 'Description',
            );
      }

      /**
       * Retrieves a list of models based on the current search/filter
conditions.
       * @return CActiveDataProvider the data provider that can return the
models based on the search/filter conditions.
       */
      public function search()
      {
            // Warning: Please modify the following code to remove
attributes that
            // should not be searched.

            $criteria=new CDbCriteria;

            $criteria->compare('JURISDICTION_ID',$this->JURISDICTION_ID);
            $criteria->compare('LL_X',$this->LL_X);
            $criteria->compare('LL_Y',$this->LL_Y);
            $criteria->compare('LR_X',$this->LR_X);
            $criteria->compare('LR_Y',$this->LR_Y);
            $criteria->compare('UL_X',$this->UL_X);
            $criteria->compare('UL_Y',$this->UL_Y);
            $criteria->compare('UR_X',$this->UR_X);
            $criteria->compare('UR_Y',$this->UR_Y);
            $criteria->compare('NAME',$this->NAME,true);
            $criteria->compare('DESCRIPTION',$this->DESCRIPTION,true);

            return new CActiveDataProvider($this, array(
                  'criteria'=>$criteria,
            ));
      }
}
<?php

/**
 * This is the model class for table "location_point".
 *
 * The followings are the available columns in table 'location_point':
 * @property integer $LOCATION_POINT_ID
 * @property double $LATITUDE
 * @property double $LONGITUDE
 *
 * The followings are the available model relations:
 * @property Report[] $reports
 */
class LocationPoint extends CActiveRecord
{
```

```
	/**
	 * Returns the static model of the specified AR class.
	 * @param string $className active record class name.
	 * @return LocationPoint the static model class
	 */
	public static function model($className=__CLASS__)
	{
		return parent::model($className);
	}

	/**
	 * @return string the associated database table name
	 */
	public function tableName()
	{
		return 'location_point';
	}

	/**
	 * @return array validation rules for model attributes.
	 */
	public function rules()
	{
		// NOTE: you should only define rules for those attributes
that
		// will receive user inputs.
		return array(
			array('LATITUDE, LONGITUDE', 'required'),
			array('LATITUDE, LONGITUDE', 'numerical'),
			// The following rule is used by search().
			// Please remove those attributes that should not be
searched.
			array('LOCATION_POINT_ID, LATITUDE, LONGITUDE', 'safe',
'on'=>'search'),
		);
	}

	/**
	 * @return array relational rules.
	 */
	public function relations()
	{
		// NOTE: you may need to adjust the relation name and the
related
		// class name for the relations automatically generated below.
		return array(
			'reports' => array(self::HAS_MANY, 'Report',
'LOCATION_POINT_ID'),
		);
	}

	/**
	 * @return array customized attribute labels (name=>label)
	 */
```

```
	public function attributeLabels()
	{
		return array(
			'LOCATION_POINT_ID' => 'Location Point',
			'LATITUDE' => 'Latitude',
			'LONGITUDE' => 'Longitude',
		);
	}

	/**
	 * Retrieves a list of models based on the current search/filter
conditions.
	 * @return CActiveDataProvider the data provider that can return the
models based on the search/filter conditions.
	 */
	public function search()
	{
		// Warning: Please modify the following code to remove
attributes that
		// should not be searched.

		$criteria=new CDbCriteria;

		$criteria->compare('LOCATION_POINT_ID',$this-
>LOCATION_POINT_ID);
		$criteria->compare('LATITUDE',$this->LATITUDE);
		$criteria->compare('LONGITUDE',$this->LONGITUDE);

		return new CActiveDataProvider($this, array(
			'criteria'=>$criteria,
		));
	}
}
<?php

/**
 * LoginForm class.
 * LoginForm is the data structure for keeping
 * user login form data. It is used by the 'login' action of
'SiteController'.
 */
class LoginForm extends CFormModel
{
	public $username;
	public $password;
	public $rememberMe;

	private $_identity;

	/**
	 * Declares the validation rules.
	 * The rules state that username and password are required,
	 * and password needs to be authenticated.
	 */
```

```
	public function rules()
	{
		return array(
			// username and password are required
			array('username, password', 'required'),
			// rememberMe needs to be a boolean
			array('rememberMe', 'boolean'),
			// password needs to be authenticated
			array('password', 'authenticate'),
		);
	}

	/**
	 * Declares attribute labels.
	 */
	public function attributeLabels()
	{
		return array(
			'rememberMe'=>'Remember me next time',
		);
	}

	/**
	 * Authenticates the password.
	 * This is the 'authenticate' validator as declared in rules().
	 */
	public function authenticate($attribute,$params)
	{
		if(!$this->hasErrors())
		{
			$this->_identity=new UserIdentity($this->username,$this-
>password);
			if(!$this->_identity->authenticate())
				$this->addError('password','Incorrect username or
password.');
		}
	}

	/**
	 * Logs in the user using the given username and password in the
model.
	 * @return boolean whether login is successful
	 */
	public function login()
	{
		if($this->_identity===null)
		{
			$this->_identity=new UserIdentity($this->username,$this-
>password);
			$this->_identity->authenticate();
		}
		if($this->_identity->errorCode===UserIdentity::ERROR_NONE)
		{
			$duration=$this->rememberMe ? 3600*24*30 : 0; // 30 days
```

```
                Yii::app()->user->login($this->_identity,$duration);
                Users::model()->updateByPk($this->_identity-
>id,array('LAST_LOGIN'=>new CDbExpression('NOW()')));
                return true;
            }
            else
                return false;
      }
}

<?php

/**
 * This is the model class for table "motorist".
 *
 * The followings are the available columns in table 'motorist':
 * @property integer $USER_ID
 * @property string $MOBILE_ID
 *
 * The followings are the available model relations:
 * @property Users $uSER
 * @property UserVerifiedReport[] $userVerifiedReports
 */
class Motorist extends CActiveRecord
{
      /**
       * Returns the static model of the specified AR class.
       * @param string $className active record class name.
       * @return Motorist the static model class
       */
      public static function model($className=__CLASS__)
      {
            return parent::model($className);
      }

      /**
       * @return string the associated database table name
       */
      public function tableName()
      {
            return 'motorist';
      }

      /**
       * @return array validation rules for model attributes.
       */
      public function rules()
      {
            // NOTE: you should only define rules for those attributes
that
            // will receive user inputs.
            return array(
                  array('USER_ID, MOBILE_ID', 'required'),
                  array('USER_ID', 'numerical', 'integerOnly'=>true),
```

```
                // The following rule is used by search().
                // Please remove those attributes that should not be
searched.
                array('USER_ID, MOBILE_ID', 'safe', 'on'=>'search'),
            );
      }

      /**
       * @return array relational rules.
       */
      public function relations()
      {
            // NOTE: you may need to adjust the relation name and the
related
            // class name for the relations automatically generated below.
            return array(
                  'uSER' => array(self::BELONGS_TO, 'Users', 'USER_ID'),
                  'userVerifiedReports' => array(self::HAS_MANY,
'UserVerifiedReport', 'USER_ID'),
            );
      }

      /**
       * @return array customized attribute labels (name=>label)
       */
      public function attributeLabels()
      {
            return array(
                  'USER_ID' => 'User',
                  'MOBILE_ID' => 'Mobile',
            );
      }

      /**
       * Retrieves a list of models based on the current search/filter
conditions.
       * @return CActiveDataProvider the data provider that can return the
models based on the search/filter conditions.
       */
      public function search()
      {
            // Warning: Please modify the following code to remove
attributes that
            // should not be searched.

            $criteria=new CDbCriteria;

            $criteria->compare('USER_ID',$this->USER_ID);
            $criteria->compare('MOBILE_ID',$this->MOBILE_ID,true);

            return new CActiveDataProvider($this, array(
                  'criteria'=>$criteria,
            ));
      }
```

```
}
<?php

/**
 * This is the model class for table "picture".
 *
 * The followings are the available columns in table 'picture':
 * @property integer $PICTURE_ID
 *
 * The followings are the available model relations:
 * @property Report[] $reports
 */
class Picture extends CActiveRecord
{
	/**
	 * Returns the static model of the specified AR class.
	 * @param string $className active record class name.
	 * @return Picture the static model class
	 */
	public static function model($className=__CLASS__)
	{
		return parent::model($className);
	}

	/**
	 * @return string the associated database table name
	 */
	public function tableName()
	{
		return 'picture';
	}

	/**
	 * @return array validation rules for model attributes.
	 */
	public function rules()
	{
		// NOTE: you should only define rules for those attributes
that
		// will receive user inputs.
		return array(
			// The following rule is used by search().
			// Please remove those attributes that should not be
searched.
			array('PICTURE_ID', 'safe', 'on'=>'search'),
		);
	}

	/**
	 * @return array relational rules.
	 */
	public function relations()
	{
```

```
            // NOTE: you may need to adjust the relation name and the
related
            // class name for the relations automatically generated below.
            return array(
                  'reports' => array(self::HAS_MANY, 'Report',
'PICTURE_ID'),
            );
      }

      /**
       * @return array customized attribute labels (name=>label)
       */
      public function attributeLabels()
      {
            return array(
                  'PICTURE_ID' => 'Picture',
            );
      }

      /**
       * Retrieves a list of models based on the current search/filter
conditions.
       * @return CActiveDataProvider the data provider that can return the
models based on the search/filter conditions.
       */
      public function search()
      {
            // Warning: Please modify the following code to remove
attributes that
            // should not be searched.

            $criteria=new CDbCriteria;

            $criteria->compare('PICTURE_ID',$this->PICTURE_ID);

            return new CActiveDataProvider($this, array(
                  'criteria'=>$criteria,
            ));
      }
}
<?php

/**
 * This is the model class for table "ra_category".
 *
 * The followings are the available columns in table 'ra_category':
 * @property integer $RA_CATEGORY_ID
 * @property string $NAME
 *
 * The followings are the available model relations:
 * @property RaReport[] $raReports
 */
class RaCategory extends CActiveRecord
{
```

```
	/**
	 * Returns the static model of the specified AR class.
	 * @param string $className active record class name.
	 * @return RaCategory the static model class
	 */
	public static function model($className=__CLASS__)
	{
		return parent::model($className);
	}

	/**
	 * @return string the associated database table name
	 */
	public function tableName()
	{
		return 'ra_category';
	}

	/**
	 * @return array validation rules for model attributes.
	 */
	public function rules()
	{
		// NOTE: you should only define rules for those attributes
that
		// will receive user inputs.
		return array(
			array('NAME', 'required'),
			// The following rule is used by search().
			// Please remove those attributes that should not be
searched.
			array('RA_CATEGORY_ID, NAME', 'safe', 'on'=>'search'),
		);
	}

	/**
	 * @return array relational rules.
	 */
	public function relations()
	{
		// NOTE: you may need to adjust the relation name and the
related
		// class name for the relations automatically generated below.
		return array(
			'raReports' => array(self::HAS_MANY, 'RaReport',
'RA_CATEGORY_ID'),
		);
	}

	/**
	 * @return array customized attribute labels (name=>label)
	 */
	public function attributeLabels()
	{
```

```
            return array(
                  'RA_CATEGORY_ID' => 'Ra Category',
                  'NAME' => 'Name',
            );
      }

      /**
       * Retrieves a list of models based on the current search/filter
conditions.
       * @return CActiveDataProvider the data provider that can return the
models based on the search/filter conditions.
       */
      public function search()
      {
            // Warning: Please modify the following code to remove
attributes that
            // should not be searched.

            $criteria=new CDbCriteria;

            $criteria->compare('RA_CATEGORY_ID',$this->RA_CATEGORY_ID);
            $criteria->compare('NAME',$this->NAME,true);

            return new CActiveDataProvider($this, array(
                  'criteria'=>$criteria,
            ));
      }
}
<?php

/**
 * This is the model class for table "ra_report".
 *
 * The followings are the available columns in table 'ra_report':
 * @property integer $REPORT_ID
 * @property integer $RA_REPORT_STATUS_ID
 * @property integer $RA_CATEGORY_ID
 *
 * The followings are the available model relations:
 * @property Report $rEPORT
 * @property RaReportStatus $rAREPORTSTATUS
 * @property RaCategory $rACATEGORY
 */
class RaReport extends CActiveRecord
{
      /**
       * Returns the static model of the specified AR class.
       * @param string $className active record class name.
       * @return RaReport the static model class
       */
      public static function model($className=__CLASS__)
      {
            return parent::model($className);
      }
```

```
	/**
	 * @return string the associated database table name
	 */
	public function tableName()
	{
		return 'ra_report';
	}

	/**
	 * @return array validation rules for model attributes.
	 */
	public function rules()
	{
		// NOTE: you should only define rules for those attributes
that
		// will receive user inputs.
		return array(
			array('REPORT_ID, RA_REPORT_STATUS_ID, RA_CATEGORY_ID',
'required'),
			array('REPORT_ID, RA_REPORT_STATUS_ID, RA_CATEGORY_ID',
'numerical', 'integerOnly'=>true),
			// The following rule is used by search().
			// Please remove those attributes that should not be
searched.
			array('REPORT_ID, RA_REPORT_STATUS_ID, RA_CATEGORY_ID',
'safe', 'on'=>'search'),
		);
	}

	/**
	 * @return array relational rules.
	 */
	public function relations()
	{
		// NOTE: you may need to adjust the relation name and the
related
		// class name for the relations automatically generated below.
		return array(
			'rEPORT' => array(self::BELONGS_TO, 'Report',
'REPORT_ID'),
			'rAREPORTSTATUS' => array(self::BELONGS_TO,
'RaReportStatus', 'RA_REPORT_STATUS_ID'),
			'rACATEGORY' => array(self::BELONGS_TO, 'RaCategory',
'RA_CATEGORY_ID'),
		);
	}

	/**
	 * @return array customized attribute labels (name=>label)
	 */
	public function attributeLabels()
	{
		return array(
```

```
                'REPORT_ID' => 'Report',
                'RA_REPORT_STATUS_ID' => 'Ra Report Status',
                'RA_CATEGORY_ID' => 'Ra Category',
                'UPDATE_TIME' => 'Updated Time',
            );
      }

      /**
       * Retrieves a list of models based on the current search/filter
conditions.
       * @return CActiveDataProvider the data provider that can return the
models based on the search/filter conditions.
       */
      public function search()
      {
            // Warning: Please modify the following code to remove
attributes that
            // should not be searched.

            $criteria=new CDbCriteria;

            $criteria->compare('REPORT_ID',$this->REPORT_ID);
            $criteria->compare('RA_REPORT_STATUS_ID',$this-
>RA_REPORT_STATUS_ID);
            $criteria->compare('RA_CATEGORY_ID',$this->RA_CATEGORY_ID);

            return new CActiveDataProvider($this, array(
                  'criteria'=>$criteria,
            ));
      }

}
<?php

/**
 * This is the model class for table "ra_report_status".
 *
 * The followings are the available columns in table 'ra_report_status':
 * @property integer $RA_REPORT_STATUS_ID
 * @property string $NAME
 *
 * The followings are the available model relations:
 * @property RaReport[] $raReports
 */
class RaReportStatus extends CActiveRecord
{
      /**
       * Returns the static model of the specified AR class.
       * @param string $className active record class name.
       * @return RaReportStatus the static model class
       */
      public static function model($className=__CLASS__)
      {
            return parent::model($className);
```

```
	}

	/**
	 * @return string the associated database table name
	 */
	public function tableName()
	{
		return 'ra_report_status';
	}

	/**
	 * @return array validation rules for model attributes.
	 */
	public function rules()
	{
		// NOTE: you should only define rules for those attributes
that
		// will receive user inputs.
		return array(
			array('NAME', 'required'),
			// The following rule is used by search().
			// Please remove those attributes that should not be
searched.
			array('RA_REPORT_STATUS_ID, NAME', 'safe',
'on'=>'search'),
		);
	}

	/**
	 * @return array relational rules.
	 */
	public function relations()
	{
		// NOTE: you may need to adjust the relation name and the
related
		// class name for the relations automatically generated below.
		return array(
			'raReports' => array(self::HAS_MANY, 'RaReport',
'RA_REPORT_STATUS_ID'),
		);
	}

	/**
	 * @return array customized attribute labels (name=>label)
	 */
	public function attributeLabels()
	{
		return array(
			'RA_REPORT_STATUS_ID' => 'Ra Report Status',
			'NAME' => 'Name',
		);
	}

	/**
```

```
       * Retrieves a list of models based on the current search/filter
conditions.
       * @return CActiveDataProvider the data provider that can return the
models based on the search/filter conditions.
       */
      public function search()
      {
            // Warning: Please modify the following code to remove
attributes that
            // should not be searched.

            $criteria=new CDbCriteria;

            $criteria->compare('RA_REPORT_STATUS_ID',$this-
>RA_REPORT_STATUS_ID);
            $criteria->compare('NAME',$this->NAME,true);

            return new CActiveDataProvider($this, array(
                  'criteria'=>$criteria,
            ));
      }
}
<?php

/**
 * This is the model class for table "report".
 *
 * The followings are the available columns in table 'report':
 * @property integer $REPORT_ID
 * @property integer $LOCATION_POINT_ID
 * @property integer $PICTURE_ID
 * @property integer $USER_ID
 * @property string $DATETIME_CREATED
 * @property integer $IS_VERIFIED_BY_SYSTEM
 * @property string $START_DATETIME
 * @property string $END_DATETIME
 *
 * The followings are the available model relations:
 * @property FsReport $fsReport
 * @property RaReport $raReport
 * @property LocationPoint $lOCATIONPOINT
 * @property Picture $pICTURE
 * @property Users $uSER
 * @property ReportHistory[] $reportHistories
 * @property RoadClosedReport $roadClosedReport
 * @property UserVerifiedReport[] $userVerifiedReports
 */
class Report extends CActiveRecord
{
      /**
       * Returns the static model of the specified AR class.
       * @param string $className active record class name.
       * @return Report the static model class
       */
```

```
	public static function model($className=__CLASS__)
	{
		return parent::model($className);
	}

	/**
	 * @return string the associated database table name
	 */
	public function tableName()
	{
		return 'report';
	}

	/**
	 * @return array validation rules for model attributes.
	 */
	public function rules()
	{
		// NOTE: you should only define rules for those attributes
that
		// will receive user inputs.
		return array(
			array('LOCATION_POINT_ID, USER_ID, DATETIME_CREATED,
START_DATETIME, END_DATETIME', 'required'),
			array('LOCATION_POINT_ID, PICTURE_ID, USER_ID,
IS_VERIFIED_BY_SYSTEM', 'numerical', 'integerOnly'=>true),
			// The following rule is used by search().
			// Please remove those attributes that should not be
searched.
			array('REPORT_ID, LOCATION_POINT_ID, PICTURE_ID, USER_ID,
DATETIME_CREATED, IS_VERIFIED_BY_SYSTEM, START_DATETIME, END_DATETIME',
'safe', 'on'=>'search'),
		);
	}

	/**
	 * @return array relational rules.
	 */
	public function relations()
	{
		// NOTE: you may need to adjust the relation name and the
related
		// class name for the relations automatically generated below.
		return array(
			'fsReport' => array(self::HAS_ONE, 'FsReport',
'REPORT_ID'),
			'raReport' => array(self::HAS_ONE, 'RaReport',
'REPORT_ID'),
			'lOCATIONPOINT' => array(self::BELONGS_TO,
'LocationPoint', 'LOCATION_POINT_ID'),
			'pICTURE' => array(self::BELONGS_TO, 'Picture',
'PICTURE_ID'),
			'uSER' => array(self::BELONGS_TO, 'Users', 'USER_ID'),
```

```
                'reportHistories' => array(self::HAS_MANY,
'ReportHistory', 'REPORT_ID'),
                'roadClosedReport' => array(self::HAS_ONE,
'RoadClosedReport', 'REPORT_ID'),
                'userVerifiedReports' => array(self::HAS_MANY,
'UserVerifiedReport', 'REPORT_ID'),
            );
      }

      /**
       * @return array customized attribute labels (name=>label)
       */
      public function attributeLabels()
      {
            return array(
                'REPORT_ID' => 'Report',
                'LOCATION_POINT_ID' => 'Location Point',
                'PICTURE_ID' => 'Picture',
                'USER_ID' => 'User',
                'DATETIME_CREATED' => 'Datetime Created',
                'IS_VERIFIED_BY_SYSTEM' => 'Is Verified By System',
                'START_DATETIME' => 'Start Datetime',
                'END_DATETIME' => 'End Datetime',
            );
      }

      /**
       * Retrieves a list of models based on the current search/filter
conditions.
       * @return CActiveDataProvider the data provider that can return the
models based on the search/filter conditions.
       */
      public function search()
      {
            // Warning: Please modify the following code to remove
attributes that
            // should not be searched.

            $criteria=new CDbCriteria;

            $criteria->compare('REPORT_ID',$this->REPORT_ID);
            $criteria->compare('LOCATION_POINT_ID',$this-
>LOCATION_POINT_ID);
            $criteria->compare('PICTURE_ID',$this->PICTURE_ID);
            $criteria->compare('USER_ID',$this->USER_ID);
            $criteria->compare('DATETIME_CREATED',$this-
>DATETIME_CREATED,true);
            $criteria->compare('IS_VERIFIED_BY_SYSTEM',$this-
>IS_VERIFIED_BY_SYSTEM);
            $criteria->compare('START_DATETIME',$this-
>START_DATETIME,true);
            $criteria->compare('END_DATETIME',$this->END_DATETIME,true);

            return new CActiveDataProvider($this, array(
```

```
                'criteria'=>$criteria,
            ));
      }
}
<?php

/**
 * This is the model class for table "report_history".
 *
 * The followings are the available columns in table 'report_history':
 * @property integer $REPORT_HISTORY_ID
 * @property integer $USER_ID
 * @property integer $REPORT_ID
 *
 * The followings are the available model relations:
 * @property TrafficInvestigator $uSER
 * @property Report $rEPORT
 */
class ReportHistory extends CActiveRecord
{
      /**
       * Returns the static model of the specified AR class.
       * @param string $className active record class name.
       * @return ReportHistory the static model class
       */
      public static function model($className=__CLASS__)
      {
            return parent::model($className);
      }

      /**
       * @return string the associated database table name
       */
      public function tableName()
      {
            return 'report_history';
      }

      /**
       * @return array validation rules for model attributes.
       */
      public function rules()
      {
            // NOTE: you should only define rules for those attributes
that
            // will receive user inputs.
            return array(
                  array('USER_ID, REPORT_ID', 'required'),
                  array('USER_ID, REPORT_ID', 'numerical',
'integerOnly'=>true),
                  // The following rule is used by search().
                  // Please remove those attributes that should not be
searched.
```

```
                array('REPORT_HISTORY_ID, USER_ID, REPORT_ID', 'safe',
'on'=>'search'),
            );
      }

      /**
       * @return array relational rules.
       */
      public function relations()
      {
            // NOTE: you may need to adjust the relation name and the
related
            // class name for the relations automatically generated below.
            return array(
                  'uSER' => array(self::BELONGS_TO, 'TrafficInvestigator',
'USER_ID'),
                  'rEPORT' => array(self::BELONGS_TO, 'Report',
'REPORT_ID'),
            );
      }

      /**
       * @return array customized attribute labels (name=>label)
       */
      public function attributeLabels()
      {
            return array(
                  'REPORT_HISTORY_ID' => 'Report History',
                  'USER_ID' => 'User',
                  'REPORT_ID' => 'Report',
            );
      }

      /**
       * Retrieves a list of models based on the current search/filter
conditions.
       * @return CActiveDataProvider the data provider that can return the
models based on the search/filter conditions.
       */
      public function search()
      {
            // Warning: Please modify the following code to remove
attributes that
            // should not be searched.

            $criteria=new CDbCriteria;

            $criteria->compare('REPORT_HISTORY_ID',$this-
>REPORT_HISTORY_ID);
            $criteria->compare('USER_ID',$this->USER_ID);
            $criteria->compare('REPORT_ID',$this->REPORT_ID);

            return new CActiveDataProvider($this, array(
                  'criteria'=>$criteria,
```

```
            ));
      }
}
<?php

/**
 * This is the model class for table "road_closed_report".
 *
 * The followings are the available columns in table 'road_closed_report':
 * @property integer $REPORT_ID
 */
class RoadClosedReport extends CActiveRecord
{
      /**
       * Returns the static model of the specified AR class.
       * @param string $className active record class name.
       * @return RoadClosedReport the static model class
       */
      public static function model($className=__CLASS__)
      {
            return parent::model($className);
      }

      /**
       * @return string the associated database table name
       */
      public function tableName()
      {
            return 'road_closed_report';
      }

      /**
       * @return array validation rules for model attributes.
       */
      public function rules()
      {
            // NOTE: you should only define rules for those attributes
that
            // will receive user inputs.
            return array(
                  array('REPORT_ID', 'required'),
                  array('REPORT_ID', 'numerical', 'integerOnly'=>true),
                  // The following rule is used by search().
                  // Please remove those attributes that should not be
searched.
                  array('REPORT_ID', 'safe', 'on'=>'search'),
            );
      }

      /**
       * @return array relational rules.
       */
      public function relations()
      {
```

```
            // NOTE: you may need to adjust the relation name and the
related
            // class name for the relations automatically generated below.
            return array(
            );
      }

      /**
       * @return array customized attribute labels (name=>label)
       */
      public function attributeLabels()
      {
            return array(
                  'REPORT_ID' => 'Report',
            );
      }

      /**
       * Retrieves a list of models based on the current search/filter
conditions.
       * @return CActiveDataProvider the data provider that can return the
models based on the search/filter conditions.
       */
      public function search()
      {
            // Warning: Please modify the following code to remove
attributes that
            // should not be searched.

            $criteria=new CDbCriteria;

            $criteria->compare('REPORT_ID',$this->REPORT_ID);

            return new CActiveDataProvider($this, array(
                  'criteria'=>$criteria,
            ));
      }
}
<?php

/**
 * This is the model class for table "system_administrator".
 *
 * The followings are the available columns in table
'system_administrator':
 * @property integer $USER_ID
 *
 * The followings are the available model relations:
 * @property AccountableUsers $uSER
 */
class SystemAdministrator extends CActiveRecord
{
      /**
       * Returns the static model of the specified AR class.
```

```
	 * @param string $className active record class name.
	 * @return SystemAdministrator the static model class
	 */
	public static function model($className=__CLASS__)
	{
		return parent::model($className);
	}

	/**
	 * @return string the associated database table name
	 */
	public function tableName()
	{
		return 'system_administrator';
	}

	/**
	 * @return array validation rules for model attributes.
	 */
	public function rules()
	{
		// NOTE: you should only define rules for those attributes
that
		// will receive user inputs.
		return array(
			array('USER_ID', 'required'),
			array('USER_ID', 'numerical', 'integerOnly'=>true),
			// The following rule is used by search().
			// Please remove those attributes that should not be
searched.
			array('USER_ID', 'safe', 'on'=>'search'),
		);
	}

	/**
	 * @return array relational rules.
	 */
	public function relations()
	{
		// NOTE: you may need to adjust the relation name and the
related
		// class name for the relations automatically generated below.
		return array(
			'uSER' => array(self::BELONGS_TO, 'AccountableUsers',
'USER_ID'),
		);
	}

	/**
	 * @return array customized attribute labels (name=>label)
	 */
	public function attributeLabels()
	{
		return array(
```

```
                'USER_ID' => 'User',
            );
      }

      /**
       * Retrieves a list of models based on the current search/filter
conditions.
       * @return CActiveDataProvider the data provider that can return the
models based on the search/filter conditions.
       */
      public function search()
      {
            // Warning: Please modify the following code to remove
attributes that
            // should not be searched.

            $criteria=new CDbCriteria;

            $criteria->compare('USER_ID',$this->USER_ID);

            return new CActiveDataProvider($this, array(
                  'criteria'=>$criteria,
            ));
      }
}
<?php

abstract class TimeActiveRecord extends CActiveRecord{

      /**
      prepares create_time and update_time
      */

      protected function beforeValidate(){
            $this->create_time = $this->update_time = new
CDbExpression('NOW()');
            $this->update_user_id = Yii::app()->user_id;

            return parent::beforeValidate();
      }


}


?>
<?php

/**
 * This is the model class for table "traffic_location_point".
 *
 * The followings are the available columns in table
'traffic_location_point':
 * @property integer $TRAFFIC_LOCATION_POINT_ID
```

```
 * @property double $LATITUDE
 * @property double $LONGITUDE
 * @property string $DATE_CREATED
 * @property integer $DATA_LOCATION_COLLECTION_ID
 * @property double $SPEED
 *
 * The followings are the available model relations:
 * @property DataLocationCollection $dATALOCATIONCOLLECTION
 */
class Tra extends CActiveRecord
{
	/**
	 * Returns the static model of the specified AR class.
	 * @param string $className active record class name.
	 * @return Tra the static model class
	 */
	public static function model($className=__CLASS__)
	{
		return parent::model($className);
	}

	/**
	 * @return string the associated database table name
	 */
	public function tableName()
	{
		return 'traffic_location_point';
	}

	/**
	 * @return array validation rules for model attributes.
	 */
	public function rules()
	{
		// NOTE: you should only define rules for those attributes
that
		// will receive user inputs.
		return array(
			array('LATITUDE, LONGITUDE, DATE_CREATED,
DATA_LOCATION_COLLECTION_ID, SPEED', 'required'),
			array('DATA_LOCATION_COLLECTION_ID', 'numerical',
'integerOnly'=>true),
			array('LATITUDE, LONGITUDE, SPEED', 'numerical'),
			// The following rule is used by search().
			// Please remove those attributes that should not be
searched.
			array('TRAFFIC_LOCATION_POINT_ID, LATITUDE, LONGITUDE,
DATE_CREATED, DATA_LOCATION_COLLECTION_ID, SPEED', 'safe',
'on'=>'search'),
		);
	}

	/**
	 * @return array relational rules.
```

```
	 */
	public function relations()
	{
		// NOTE: you may need to adjust the relation name and the
related
		// class name for the relations automatically generated below.
		return array(
			'dATALOCATIONCOLLECTION' => array(self::BELONGS_TO,
'DataLocationCollection', 'DATA_LOCATION_COLLECTION_ID'),
		);
	}

	/**
	 * @return array customized attribute labels (name=>label)
	 */
	public function attributeLabels()
	{
		return array(
			'TRAFFIC_LOCATION_POINT_ID' => 'Traffic Location Point',
			'LATITUDE' => 'Latitude',
			'LONGITUDE' => 'Longitude',
			'DATE_CREATED' => 'Date Created',
			'DATA_LOCATION_COLLECTION_ID' => 'Data Location
Collection',
			'SPEED' => 'Speed',
		);
	}

	/**
	 * Retrieves a list of models based on the current search/filter
conditions.
	 * @return CActiveDataProvider the data provider that can return the
models based on the search/filter conditions.
	 */
	public function search()
	{
		// Warning: Please modify the following code to remove
attributes that
		// should not be searched.

		$criteria=new CDbCriteria;

		$criteria->compare('TRAFFIC_LOCATION_POINT_ID',$this-
>TRAFFIC_LOCATION_POINT_ID);
		$criteria->compare('LATITUDE',$this->LATITUDE);
		$criteria->compare('LONGITUDE',$this->LONGITUDE);
		$criteria->compare('DATE_CREATED',$this->DATE_CREATED,true);
		$criteria->compare('DATA_LOCATION_COLLECTION_ID',$this-
>DATA_LOCATION_COLLECTION_ID);
		$criteria->compare('SPEED',$this->SPEED);

		return new CActiveDataProvider($this, array(
			'criteria'=>$criteria,
		));
```

```
	}
}
<?php

/**
 * This is the model class for table "traffic_investigator".
 *
 * The followings are the available columns in table
'traffic_investigator':
 * @property integer $USER_ID
 * @property integer $TRAFFIC_INVESTIGATOR_STATUS_ID
 * @property integer $JURISDICTION_ID
 * @property double $CURRENT_LAT
 * @property double $CURRENT_LNG
 * @property integer $IS_ACTIVE
 *
 * The followings are the available model relations:
 * @property ReportHistory[] $reportHistories
 * @property AccountableUsers $uSER
 * @property Jurisdiction $jURISDICTION
 * @property TrafficInvestigatorStatus $tRAFFICINVESTIGATORSTATUS
 */
class TrafficInvestigator extends CActiveRecord
{
	/**
	 * Returns the static model of the specified AR class.
	 * @param string $className active record class name.
	 * @return TrafficInvestigator the static model class
	 */
	public static function model($className=__CLASS__)
	{
		return parent::model($className);
	}

	/**
	 * @return string the associated database table name
	 */
	public function tableName()
	{
		return 'traffic_investigator';
	}

	/**
	 * @return array validation rules for model attributes.
	 */
	public function rules()
	{
		// NOTE: you should only define rules for those attributes
that
		// will receive user inputs.
		return array(
			array('USER_ID, JURISDICTION_ID', 'required'),
			array('USER_ID, TRAFFIC_INVESTIGATOR_STATUS_ID,
JURISDICTION_ID, IS_ACTIVE', 'numerical', 'integerOnly'=>true),
```

```
                array('CURRENT_LAT, CURRENT_LNG', 'numerical'),
                // The following rule is used by search().
                // Please remove those attributes that should not be
searched.
                array('USER_ID, TRAFFIC_INVESTIGATOR_STATUS_ID,
JURISDICTION_ID, CURRENT_LAT, CURRENT_LNG, IS_ACTIVE', 'safe',
'on'=>'search'),
            );
      }

      /**
       * @return array relational rules.
       */
      public function relations()
      {
            // NOTE: you may need to adjust the relation name and the
related
            // class name for the relations automatically generated below.
            return array(
                'reportHistories' => array(self::HAS_MANY,
'ReportHistory', 'USER_ID'),
                'uSER' => array(self::BELONGS_TO, 'AccountableUsers',
'USER_ID'),
                'jURISDICTION' => array(self::BELONGS_TO, 'Jurisdiction',
'JURISDICTION_ID'),
                'tRAFFICINVESTIGATORSTATUS' => array(self::BELONGS_TO,
'TrafficInvestigatorStatus', 'TRAFFIC_INVESTIGATOR_STATUS_ID'),
            );
      }

      /**
       * @return array customized attribute labels (name=>label)
       */
      public function attributeLabels()
      {
            return array(
                'USER_ID' => 'User',
                'TRAFFIC_INVESTIGATOR_STATUS_ID' => 'Traffic Investigator
Status',
                'JURISDICTION_ID' => 'Jurisdiction',
                'CURRENT_LAT' => 'Current Lat',
                'CURRENT_LNG' => 'Current Lng',
                'IS_ACTIVE' => 'Is Active',
            );
      }

      /**
       * Retrieves a list of models based on the current search/filter
conditions.
       * @return CActiveDataProvider the data provider that can return the
models based on the search/filter conditions.
       */
      public function search()
      {
```

```
            // Warning: Please modify the following code to remove
attributes that
            // should not be searched.

            $criteria=new CDbCriteria;

            $criteria->compare('USER_ID',$this->USER_ID);
            $criteria->compare('TRAFFIC_INVESTIGATOR_STATUS_ID',$this-
>TRAFFIC_INVESTIGATOR_STATUS_ID);
            $criteria->compare('JURISDICTION_ID',$this->JURISDICTION_ID);
            $criteria->compare('CURRENT_LAT',$this->CURRENT_LAT);
            $criteria->compare('CURRENT_LNG',$this->CURRENT_LNG);
            $criteria->compare('IS_ACTIVE',$this->IS_ACTIVE);

            return new CActiveDataProvider($this, array(
                  'criteria'=>$criteria,
            ));
      }
}
<?php

/**
 * This is the model class for table "traffic_investigator_status".
 *
 * The followings are the available columns in table
'traffic_investigator_status':
 * @property integer $TRAFFIC_INVESTIGATOR_STATUS_ID
 * @property string $NAME
 *
 * The followings are the available model relations:
 * @property TrafficInvestigator[] $trafficInvestigators
 */
class TrafficInvestigatorStatus extends CActiveRecord
{
      /**
       * Returns the static model of the specified AR class.
       * @param string $className active record class name.
       * @return TrafficInvestigatorStatus the static model class
       */
      public static function model($className=__CLASS__)
      {
            return parent::model($className);
      }

      /**
       * @return string the associated database table name
       */
      public function tableName()
      {
            return 'traffic_investigator_status';
      }

      /**
       * @return array validation rules for model attributes.
```

```
	 */
	public function rules()
	{
		// NOTE: you should only define rules for those attributes
that
		// will receive user inputs.
		return array(
			array('NAME', 'required'),
			// The following rule is used by search().
			// Please remove those attributes that should not be
searched.
			array('TRAFFIC_INVESTIGATOR_STATUS_ID, NAME', 'safe',
'on'=>'search'),
		);
	}

	/**
	 * @return array relational rules.
	 */
	public function relations()
	{
		// NOTE: you may need to adjust the relation name and the
related
		// class name for the relations automatically generated below.
		return array(
			'trafficInvestigators' => array(self::HAS_MANY,
'TrafficInvestigator', 'TRAFFIC_INVESTIGATOR_STATUS_ID'),
		);
	}

	/**
	 * @return array customized attribute labels (name=>label)
	 */
	public function attributeLabels()
	{
		return array(
			'TRAFFIC_INVESTIGATOR_STATUS_ID' => 'Traffic Investigator
Status',
			'NAME' => 'Name',
		);
	}

	/**
	 * Retrieves a list of models based on the current search/filter
conditions.
	 * @return CActiveDataProvider the data provider that can return the
models based on the search/filter conditions.
	 */
	public function search()
	{
		// Warning: Please modify the following code to remove
attributes that
		// should not be searched.
```

```
            $criteria=new CDbCriteria;

            $criteria->compare('TRAFFIC_INVESTIGATOR_STATUS_ID',$this-
>TRAFFIC_INVESTIGATOR_STATUS_ID);
            $criteria->compare('NAME',$this->NAME,true);

            return new CActiveDataProvider($this, array(
                  'criteria'=>$criteria,
            ));
      }
}
<?php

/**
 * This is the model class for table "traffic_location_point".
 *
 * The followings are the available columns in table
'traffic_location_point':
 * @property integer $TRAFFIC_LOCATION_POINT_ID
 * @property double $LATITUDE
 * @property double $LONGITUDE
 * @property string $DATE_CREATED
 * @property integer $DATA_LOCATION_COLLECTION_ID
 * @property double $SPEED
 *
 * The followings are the available model relations:
 * @property DataLocationCollection $dATALOCATIONCOLLECTION
 */
class TrafficLocationPoint extends CActiveRecord
{
      /**
       * Returns the static model of the specified AR class.
       * @param string $className active record class name.
       * @return TrafficLocationPoint the static model class
       */
      public static function model($className=__CLASS__)
      {
            return parent::model($className);
      }

      /**
       * @return string the associated database table name
       */
      public function tableName()
      {
            return 'traffic_location_point';
      }

      /**
       * @return array validation rules for model attributes.
       */
      public function rules()
      {
```

```
            // NOTE: you should only define rules for those attributes
that
            // will receive user inputs.
            return array(
                  array('LATITUDE, LONGITUDE, DATE_CREATED,
DATA_LOCATION_COLLECTION_ID, SPEED', 'required'),
                  array('DATA_LOCATION_COLLECTION_ID', 'numerical',
'integerOnly'=>true),
                  array('LATITUDE, LONGITUDE, SPEED', 'numerical'),
                  // The following rule is used by search().
                  // Please remove those attributes that should not be
searched.
                  array('TRAFFIC_LOCATION_POINT_ID, LATITUDE, LONGITUDE,
DATE_CREATED, DATA_LOCATION_COLLECTION_ID, SPEED', 'safe',
'on'=>'search'),
            );
      }

      /**
       * @return array relational rules.
       */
      public function relations()
      {
            // NOTE: you may need to adjust the relation name and the
related
            // class name for the relations automatically generated below.
            return array(
                  'dATALOCATIONCOLLECTION' => array(self::BELONGS_TO,
'DataLocationCollection', 'DATA_LOCATION_COLLECTION_ID'),
            );
      }

      /**
       * @return array customized attribute labels (name=>label)
       */
      public function attributeLabels()
      {
            return array(
                  'TRAFFIC_LOCATION_POINT_ID' => 'Traffic Location Point',
                  'LATITUDE' => 'Latitude',
                  'LONGITUDE' => 'Longitude',
                  'DATE_CREATED' => 'Date Created',
                  'DATA_LOCATION_COLLECTION_ID' => 'Data Location
Collection',
                  'SPEED' => 'Speed',
            );
      }

      /**
       * Retrieves a list of models based on the current search/filter
conditions.
       * @return CActiveDataProvider the data provider that can return the
models based on the search/filter conditions.
       */
```

```
	public function search()
	{
		// Warning: Please modify the following code to remove
attributes that
		// should not be searched.

		$criteria=new CDbCriteria;

		$criteria->compare('TRAFFIC_LOCATION_POINT_ID',$this-
>TRAFFIC_LOCATION_POINT_ID);
		$criteria->compare('LATITUDE',$this->LATITUDE);
		$criteria->compare('LONGITUDE',$this->LONGITUDE);
		$criteria->compare('DATE_CREATED',$this->DATE_CREATED,true);
		$criteria->compare('DATA_LOCATION_COLLECTION_ID',$this-
>DATA_LOCATION_COLLECTION_ID);
		$criteria->compare('SPEED',$this->SPEED);

		return new CActiveDataProvider($this, array(
			'criteria'=>$criteria,
		));
	}
}
<?php

/**
 * This is the model class for table "traffic_operations_center_officer".
 *
 * The followings are the available columns in table
'traffic_operations_center_officer':
 * @property integer $USER_ID
 *
 * The followings are the available model relations:
 * @property AccountableUsers $uSER
 */
class TrafficOperationsCenterOfficer extends CActiveRecord
{
	/**
	 * Returns the static model of the specified AR class.
	 * @param string $className active record class name.
	 * @return TrafficOperationsCenterOfficer the static model class
	 */
	public static function model($className=__CLASS__)
	{
		return parent::model($className);
	}

	/**
	 * @return string the associated database table name
	 */
	public function tableName()
	{
		return 'traffic_operations_center_officer';
	}
```

```
	/**
	 * @return array validation rules for model attributes.
	 */
	public function rules()
	{
		// NOTE: you should only define rules for those attributes
that
		// will receive user inputs.
		return array(
			array('USER_ID', 'required'),
			array('USER_ID', 'numerical', 'integerOnly'=>true),
			// The following rule is used by search().
			// Please remove those attributes that should not be
searched.
			array('USER_ID', 'safe', 'on'=>'search'),
		);
	}

	/**
	 * @return array relational rules.
	 */
	public function relations()
	{
		// NOTE: you may need to adjust the relation name and the
related
		// class name for the relations automatically generated below.
		return array(
			'uSER' => array(self::BELONGS_TO, 'AccountableUsers',
'USER_ID'),
		);
	}

	/**
	 * @return array customized attribute labels (name=>label)
	 */
	public function attributeLabels()
	{
		return array(
			'USER_ID' => 'User',
		);
	}

	/**
	 * Retrieves a list of models based on the current search/filter
conditions.
	 * @return CActiveDataProvider the data provider that can return the
models based on the search/filter conditions.
	 */
	public function search()
	{
		// Warning: Please modify the following code to remove
attributes that
		// should not be searched.
```

```
            $criteria=new CDbCriteria;

            $criteria->compare('USER_ID',$this->USER_ID);

            return new CActiveDataProvider($this, array(
                  'criteria'=>$criteria,
            ));
      }
}
<?php

/**
 * This is the model class for table "users".
 *
 * The followings are the available columns in table 'users':
 * @property integer $USER_ID
 * @property string $DATE_REGISTERED
 * @property string $LAST_LOGIN
 *
 * The followings are the available model relations:
 * @property AccountableUsers $accountableUsers
 * @property Motorist $motorist
 * @property Report[] $reports
 * @property ReportHistory[] $reportHistories
 * @property UserVerifiedReport[] $userVerifiedReports
 */
class Users extends CActiveRecord
{
      /**
       * Returns the static model of the specified AR class.
       * @param string $className active record class name.
       * @return Users the static model class
       */
      public static function model($className=__CLASS__)
      {
            return parent::model($className);
      }

      /**
       * @return string the associated database table name
       */
      public function tableName()
      {
            return 'users';
      }

      /**
       * @return array validation rules for model attributes.
       */
      public function rules()
      {
            // NOTE: you should only define rules for those attributes
that
            // will receive user inputs.
```

```
            return array(
                  array('DATE_REGISTERED, LAST_LOGIN', 'required'),
                  // The following rule is used by search().
                  // Please remove those attributes that should not be
searched.
                  array('USER_ID, DATE_REGISTERED, LAST_LOGIN', 'safe',
'on'=>'search'),
            );
      }

      /**
       * @return array relational rules.
       */
      public function relations()
      {
            // NOTE: you may need to adjust the relation name and the
related
            // class name for the relations automatically generated below.
            return array(
                  'accountableUsers' => array(self::HAS_ONE,
'AccountableUsers', 'USER_ID'),
                  'motorist' => array(self::HAS_ONE, 'Motorist',
'USER_ID'),
                  'reports' => array(self::HAS_MANY, 'Report', 'USER_ID'),
                  'reportHistories' => array(self::HAS_MANY,
'ReportHistory', 'USER_ID'),
                  'userVerifiedReports' => array(self::HAS_MANY,
'UserVerifiedReport', 'USER_ID'),
            );
      }

      /**
       * @return array customized attribute labels (name=>label)
       */
      public function attributeLabels()
      {
            return array(
                  'USER_ID' => 'User',
                  'DATE_REGISTERED' => 'Date Registered',
                  'LAST_LOGIN' => 'Last Login',
            );
      }

      /**
       * Retrieves a list of models based on the current search/filter
conditions.
       * @return CActiveDataProvider the data provider that can return the
models based on the search/filter conditions.
       */
      public function search()
      {
            // Warning: Please modify the following code to remove
attributes that
            // should not be searched.
```

```
            $criteria=new CDbCriteria;

            $criteria->compare('USER_ID',$this->USER_ID);
            $criteria->compare('DATE_REGISTERED',$this-
>DATE_REGISTERED,true);
            $criteria->compare('LAST_LOGIN',$this->LAST_LOGIN,true);

            return new CActiveDataProvider($this, array(
                  'criteria'=>$criteria,
            ));
      }
}
<?php

/**
 * This is the model class for table "virtual_trip_line".
 *
 * The followings are the available columns in table 'virtual_trip_line':
 * @property integer $VIRTUAL_TRIP_LINE_ID
 * @property double $LL_X
 * @property double $LL_Y
 * @property double $LR_X
 * @property double $LR_Y
 * @property double $UL_X
 * @property double $UL_Y
 * @property double $UR_X
 * @property double $UR_Y
 * @property double $marker1_latitude
 * @property double $marker1_longitude
 * @property double $marker2_latitude
 * @property double $marker2_longitude
 * @property string $DIRECTION
 * @property string $DESCRIPTION
 *
 * The followings are the available model relations:
 * @property DataLocationCollection[] $dataLocationCollections
 */
class VirtualTripLine extends CActiveRecord
{
      /**
       * Returns the static model of the specified AR class.
       * @param string $className active record class name.
       * @return VirtualTripLine the static model class
       */
      public static function model($className=__CLASS__)
      {
            return parent::model($className);
      }

      /**
       * @return string the associated database table name
       */
      public function tableName()
```

```
	{
		return 'virtual_trip_line';
	}

	/**
	 * @return array validation rules for model attributes.
	 */
	public function rules()
	{
		// NOTE: you should only define rules for those attributes
that
		// will receive user inputs.
		return array(
			array('LL_X, LL_Y, LR_X, LR_Y, UL_X, UL_Y, UR_X, UR_Y,
marker1_latitude, marker1_longitude, marker2_latitude, marker2_longitude,
DIRECTION, DESCRIPTION', 'required'),
			array('LL_X, LL_Y, LR_X, LR_Y, UL_X, UL_Y, UR_X, UR_Y,
marker1_latitude, marker1_longitude, marker2_latitude, marker2_longitude',
'numerical'),
			// The following rule is used by search().
			// Please remove those attributes that should not be
searched.
			array('VIRTUAL_TRIP_LINE_ID, LL_X, LL_Y, LR_X, LR_Y,
UL_X, UL_Y, UR_X, UR_Y, marker1_latitude, marker1_longitude,
marker2_latitude, marker2_longitude, DIRECTION, DESCRIPTION', 'safe',
'on'=>'search'),
		);
	}

	/**
	 * @return array relational rules.
	 */
	public function relations()
	{
		// NOTE: you may need to adjust the relation name and the
related
		// class name for the relations automatically generated below.
		return array(
			'dataLocationCollections' => array(self::HAS_MANY,
'DataLocationCollection', 'VIRTUAL_TRIP_LINE_ID'),
		);
	}

	/**
	 * @return array customized attribute labels (name=>label)
	 */
	public function attributeLabels()
	{
		return array(
			'VIRTUAL_TRIP_LINE_ID' => 'Virtual Trip Line',
			'LL_X' => 'Ll X',
			'LL_Y' => 'Ll Y',
			'LR_X' => 'Lr X',
			'LR_Y' => 'Lr Y',
```

```
                'UL_X' => 'Ul X',
                'UL_Y' => 'Ul Y',
                'UR_X' => 'Ur X',
                'UR_Y' => 'Ur Y',
                'marker1_latitude' => 'Marker1 Latitude',
                'marker1_longitude' => 'Marker1 Longitude',
                'marker2_latitude' => 'Marker2 Latitude',
                'marker2_longitude' => 'Marker2 Longitude',
                'DIRECTION' => 'Direction',
                'DESCRIPTION' => 'Description',
            );
      }

      /**
       * Retrieves a list of models based on the current search/filter
conditions.
       * @return CActiveDataProvider the data provider that can return the
models based on the search/filter conditions.
       */
      public function search()
      {
            // Warning: Please modify the following code to remove
attributes that
            // should not be searched.

            $criteria=new CDbCriteria;

            $criteria->compare('VIRTUAL_TRIP_LINE_ID',$this-
>VIRTUAL_TRIP_LINE_ID);
            $criteria->compare('LL_X',$this->LL_X);
            $criteria->compare('LL_Y',$this->LL_Y);
            $criteria->compare('LR_X',$this->LR_X);
            $criteria->compare('LR_Y',$this->LR_Y);
            $criteria->compare('UL_X',$this->UL_X);
            $criteria->compare('UL_Y',$this->UL_Y);
            $criteria->compare('UR_X',$this->UR_X);
            $criteria->compare('UR_Y',$this->UR_Y);
            $criteria->compare('marker1_latitude',$this-
>marker1_latitude);
            $criteria->compare('marker1_longitude',$this-
>marker1_longitude);
            $criteria->compare('marker2_latitude',$this-
>marker2_latitude);
            $criteria->compare('marker2_longitude',$this-
>marker2_longitude);
            $criteria->compare('DIRECTION',$this->DIRECTION,true);
            $criteria->compare('DESCRIPTION',$this->DESCRIPTION,true);

            return new CActiveDataProvider($this, array(
                  'criteria'=>$criteria,
            ));
      }
}
<?php
```

```
// This is the configuration for yiic console application.
// Any writable CConsoleApplication properties can be configured here.
return array(
      'basePath'=>dirname(__FILE__).DIRECTORY_SEPARATOR.'..',
      'name'=>'My Console Application',

      // preloading 'log' component
      'preload'=>array('log'),

      // application components
      'components'=>array(
            'db'=>array(
                  'connectionString' =>
'sqlite:'.dirname(__FILE__).'/../data/testdrive.db',
            ),
            // uncomment the following to use a MySQL database
            /*
            'db'=>array(
                  'connectionString' =>
'mysql:host=localhost;dbname=testdrive',
                  'emulatePrepare' => true,
                  'username' => 'root',
                  'password' => '',
                  'charset' => 'utf8',
            ),
            */
            'log'=>array(
                  'class'=>'CLogRouter',
                  'routes'=>array(
                        array(
                              'class'=>'CFileLogRoute',
                              'levels'=>'error, warning',
                        ),
                  ),
            ),
      ),
);
<?php

// uncomment the following to define a path alias
// Yii::setPathOfAlias('local','path/to/local-folder');

// This is the main Web application configuration. Any writable
// CWebApplication properties can be configured here.
return array(
      'basePath'=>dirname(__FILE__).DIRECTORY_SEPARATOR.'..',
      'name'=>'Road Monitoring System',
      //'theme'=>'trond',
      // preloading 'log' component
      'preload'=>array('log'),

      // autoloading model and component classes
      'import'=>array(
```

```
            'application.models.*',
            'application.components.*',
      ),

      'modules'=>array(
            // uncomment the following to enable the Gii tool
            'gii'=>array(
                  'class'=>'system.gii.GiiModule',
                  'password'=>'1234',
                  // If removed, Gii defaults to localhost only. Edit
carefully to taste.
                  'ipFilters'=>false,
            ),
      ),

      // application components
      'components'=>array(
            'user'=>array(
                  // enable cookie-based authentication
                  'allowAutoLogin'=>true,
            ),
            // uncomment the following to enable URLs in path-format
            /*
            'urlManager'=>array(
                  'urlFormat'=>'path',
                  'rules'=>array(
                        '<controller:\w+>/<id:\d+>'=>'<controller>/view',

      '<controller:\w+>/<action:\w+>/<id:\d+>'=>'<controller>/<action>',

      '<controller:\w+>/<action:\w+>'=>'<controller>/<action>',
                  ),
            ),
            */
            /*
            'db'=>array(
                  'connectionString' =>
'sqlite:'.dirname(__FILE__).'/../data/testdrive.db',
            ),
            */
            // uncomment the following to use a MySQL database
            /*
            'db'=>array(
                  'connectionString' =>
'mysql:host=agila.upm.edu.ph;dbname=NavigateSupport',
                  'emulatePrepare' => true,
                  'username' => 'NavigateSupport',
                  'password' => 'aXVDEh2yDDCMTc5v',
                  'charset' => 'utf8',
            ),
            */
            /*
            'db'=>array(
```

```
                'connectionString' =>
'mysql:host=localhost;dbname=NavigateSupport',
                'emulatePrepare' => true,
                'username' => 'NavigateSupport',
                'password' => 'aXVDEh2yDDCMTc5v',
                'charset' => 'utf8',
            ),
            */
            'db'=>array(
                'connectionString' =>
'mysql:host=localhost;dbname=sp_test',
                'emulatePrepare' => true,
                'username' => '',
                'password' => '',
                'charset' => 'utf8',
            ),


            'authManager'=>array(
                'class'=>'CDbAuthManager',
                'connectionID'=>'db',

            ),

            'errorHandler'=>array(
                // use 'site/error' action to display errors
                'errorAction'=>'site/error',
            ),
            'log'=>array(
                'class'=>'CLogRouter',
                'routes'=>array(
                    array(
                        'class'=>'CFileLogRoute',
                        'levels'=>'error, warning',
                    ),
                    // uncomment the following to show log messages on
web pages
                    /*
                    array(
                        'class'=>'CWebLogRoute',
                    ),
                    */
                ),
            ),
        ),

        // application-level parameters that can be accessed
        // using Yii::app()->params['paramName']
        'params'=>array(
            // this is used in contact page
            'adminEmail'=>'webmaster@example.com',
        ),
);
<?php
```

```
return CMap::mergeArray(
      require(dirname(__FILE__).'/main.php'),
      array(
            'components'=>array(
                  'fixture'=>array(
                        'class'=>'system.test.CDbFixtureManager',
                  ),
                  /* uncomment the following to provide test database
connection
                  'db'=>array(
                        'connectionString'=>'DSN for test database',
                  ),
                  */
            ),
      )
);

<?php
/* @var $this AccountableUserController */
/* @var $model AccountableUser */
/* @var $form CActiveForm */
?>

<div class="form">

<?php $form=$this->beginWidget('CActiveForm', array(
      'id'=>'accountable-user-form',
      'enableAjaxValidation'=>false,
)); ?>

      <p class="note">Fields with <span class="required">*</span> are
required.</p>

      <?php echo $form->errorSummary($model); ?>

      <div class="row">
            <?php echo $form->labelEx($model,'USER_ID'); ?>
            <?php echo $form->textField($model,'USER_ID'); ?>
            <?php echo $form->error($model,'USER_ID'); ?>
      </div>

      <div class="row">
            <?php echo $form->labelEx($model,'FIRST_NAME'); ?>
            <?php echo $form-
>textArea($model,'FIRST_NAME',array('rows'=>6, 'cols'=>50)); ?>
            <?php echo $form->error($model,'FIRST_NAME'); ?>
      </div>

      <div class="row">
            <?php echo $form->labelEx($model,'LAST_NAME'); ?>
            <?php echo $form->textArea($model,'LAST_NAME',array('rows'=>6,
'cols'=>50)); ?>
            <?php echo $form->error($model,'LAST_NAME'); ?>
```

```
	</div>

	<div class="row">
		<?php echo $form->labelEx($model,'USER_INITIAL'); ?>
		<?php echo $form-
>textArea($model,'USER_INITIAL',array('rows'=>6, 'cols'=>50)); ?>
		<?php echo $form->error($model,'USER_INITIAL'); ?>
	</div>

	<div class="row">
		<?php echo $form->labelEx($model,'BIRTHDAY'); ?>
		<?php echo $form->textField($model,'BIRTHDAY'); ?>
		<?php echo $form->error($model,'BIRTHDAY'); ?>
	</div>

	<div class="row">
		<?php echo $form->labelEx($model,'USERNAME'); ?>
		<?php echo $form->textArea($model,'USERNAME',array('rows'=>6,
'cols'=>50)); ?>
		<?php echo $form->error($model,'USERNAME'); ?>
	</div>

	<div class="row">
		<?php echo $form->labelEx($model,'PASSWORD'); ?>
		<?php echo $form->textArea($model,'PASSWORD',array('rows'=>6,
'cols'=>50)); ?>
		<?php echo $form->error($model,'PASSWORD'); ?>
	</div>

	<div class="row buttons">
		<?php echo CHtml::submitButton($model->isNewRecord ? 'Create'
: 'Save'); ?>
	</div>

<?php $this->endWidget(); ?>

</div><!-- form -->
<?php
/* @var $this AccountableUserController */
/* @var $model AccountableUser */
/* @var $form CActiveForm */
?>

<div class="wide form">

<?php $form=$this->beginWidget('CActiveForm', array(
	'action'=>Yii::app()->createUrl($this->route),
	'method'=>'get',
)); ?>

	<div class="row">
		<?php echo $form->label($model,'USER_ID'); ?>
		<?php echo $form->textField($model,'USER_ID'); ?>
	</div>
```

```
	<div class="row">
		<?php echo $form->label($model,'FIRST_NAME'); ?>
		<?php echo $form-
>textArea($model,'FIRST_NAME',array('rows'=>6, 'cols'=>50)); ?>
	</div>

	<div class="row">
		<?php echo $form->label($model,'LAST_NAME'); ?>
		<?php echo $form->textArea($model,'LAST_NAME',array('rows'=>6,
'cols'=>50)); ?>
	</div>

	<div class="row">
		<?php echo $form->label($model,'USER_INITIAL'); ?>
		<?php echo $form-
>textArea($model,'USER_INITIAL',array('rows'=>6, 'cols'=>50)); ?>
	</div>

	<div class="row">
		<?php echo $form->label($model,'BIRTHDAY'); ?>
		<?php echo $form->textField($model,'BIRTHDAY'); ?>
	</div>

	<div class="row">
		<?php echo $form->label($model,'USERNAME'); ?>
		<?php echo $form->textArea($model,'USERNAME',array('rows'=>6,
'cols'=>50)); ?>
	</div>

	<div class="row">
		<?php echo $form->label($model,'PASSWORD'); ?>
		<?php echo $form->textArea($model,'PASSWORD',array('rows'=>6,
'cols'=>50)); ?>
	</div>

	<div class="row buttons">
		<?php echo CHtml::submitButton('Search'); ?>
	</div>

<?php $this->endWidget(); ?>

</div><!-- search-form -->
<?php
/* @var $this AccountableUserController */
/* @var $data AccountableUser */
?>

<div class="view">

	<b><?php echo CHtml::encode($data->getAttributeLabel('USER_ID'));
?>:</b>
	<?php echo CHtml::link(CHtml::encode($data->USER_ID), array('view',
'id'=>$data->USER_ID)); ?>
```

```
	<br />

	<b><?php echo CHtml::encode($data->getAttributeLabel('FIRST_NAME'));
?>:</b>
	<?php echo CHtml::encode($data->FIRST_NAME); ?>
	<br />

	<b><?php echo CHtml::encode($data->getAttributeLabel('LAST_NAME'));
?>:</b>
	<?php echo CHtml::encode($data->LAST_NAME); ?>
	<br />

	<b><?php echo CHtml::encode($data-
>getAttributeLabel('USER_INITIAL')); ?>:</b>
	<?php echo CHtml::encode($data->USER_INITIAL); ?>
	<br />

	<b><?php echo CHtml::encode($data->getAttributeLabel('BIRTHDAY'));
?>:</b>
	<?php echo CHtml::encode($data->BIRTHDAY); ?>
	<br />

	<b><?php echo CHtml::encode($data->getAttributeLabel('USERNAME'));
?>:</b>
	<?php echo CHtml::encode($data->USERNAME); ?>
	<br />

	<b><?php echo CHtml::encode($data->getAttributeLabel('PASSWORD'));
?>:</b>
	<?php echo CHtml::encode($data->PASSWORD); ?>
	<br />


</div>
<?php
/* @var $this AccountableUserController */
/* @var $model AccountableUser */

$this->breadcrumbs=array(
	'Accountable Users'=>array('index'),
	'Manage',
);

$this->menu=array(
	array('label'=>'List AccountableUser', 'url'=>array('index')),
	array('label'=>'Create AccountableUser', 'url'=>array('create')),
);

Yii::app()->clientScript->registerScript('search', "
$('.search-button').click(function(){
	$('.search-form').toggle();
	return false;
});
$('.search-form form').submit(function(){
```

```
      $('#accountable-user-grid').yiiGridView('update', {
            data: $(this).serialize()
      });
      return false;
});
");
?>

<h1>Manage Accountable Users</h1>

<p>
You may optionally enter a comparison operator (<b>&lt;</b>, <b>&lt;=</b>,
<b>&gt;</b>, <b>&gt;=</b>, <b>&lt;&gt;</b>
or <b>=</b>) at the beginning of each of your search values to specify how
the comparison should be done.
</p>

<?php echo CHtml::link('Advanced Search','#',array('class'=>'search-
button')); ?>
<div class="search-form" style="display:none">
<?php $this->renderPartial('_search',array(
      'model'=>$model,
)); ?>
</div><!-- search-form -->

<?php $this->widget('zii.widgets.grid.CGridView', array(
      'id'=>'accountable-user-grid',
      'dataProvider'=>$model->search(),
      'filter'=>$model,
      'columns'=>array(
            'USER_ID',
            'FIRST_NAME',
            'LAST_NAME',
            'USER_INITIAL',
            'BIRTHDAY',
            'USERNAME',
            /*
            'PASSWORD',
            */
            array(
                  'class'=>'CButtonColumn',
            ),
      ),
)); ?>

<?php
/* @var $this AccountableUserController */
/* @var $model AccountableUser */

$this->breadcrumbs=array(
      'Accountable Users'=>array('index'),
      'Create',
);
```

```
$this->menu=array(
      array('label'=>'List AccountableUser', 'url'=>array('index')),
      array('label'=>'Manage AccountableUser', 'url'=>array('admin')),
);
?>

<h1>Create AccountableUser</h1>

<?php echo $this->renderPartial('_form', array('model'=>$model)); ?>
<?php
/* @var $this AccountableUserController */
/* @var $dataProvider CActiveDataProvider */

$this->breadcrumbs=array(
      'Accountable Users',
);

$this->menu=array(
      array('label'=>'Create AccountableUser', 'url'=>array('create')),
      array('label'=>'Manage AccountableUser', 'url'=>array('admin')),
);
?>

<h1>Accountable Users</h1>

<?php $this->widget('zii.widgets.CListView', array(
      'dataProvider'=>$dataProvider,
      'itemView'=>'_view',
)); ?>

<?php
/* @var $this AccountableUserController */
/* @var $model AccountableUser */

$this->breadcrumbs=array(
      'Accountable Users'=>array('index'),
      $model->USER_ID=>array('view','id'=>$model->USER_ID),
      'Update',
);

$this->menu=array(
      array('label'=>'List AccountableUser', 'url'=>array('index')),
      array('label'=>'Create AccountableUser', 'url'=>array('create')),
      array('label'=>'View AccountableUser', 'url'=>array('view',
'id'=>$model->USER_ID)),
      array('label'=>'Manage AccountableUser', 'url'=>array('admin')),
);
?>

<h1>Update AccountableUser <?php echo $model->USER_ID; ?></h1>

<?php echo $this->renderPartial('_form', array('model'=>$model)); ?>
<?php
/* @var $this AccountableUserController */
```

```
/* @var $model AccountableUser */

$this->breadcrumbs=array(
      'Accountable Users'=>array('index'),
      $model->USER_ID,
);

$this->menu=array(
      array('label'=>'List AccountableUser', 'url'=>array('index')),
      array('label'=>'Create AccountableUser', 'url'=>array('create')),
      array('label'=>'Update AccountableUser', 'url'=>array('update',
'id'=>$model->USER_ID)),
      array('label'=>'Delete AccountableUser', 'url'=>'#',
'linkOptions'=>array('submit'=>array('delete','id'=>$model-
>USER_ID),'confirm'=>'Are you sure you want to delete this item?')),
      array('label'=>'Manage AccountableUser', 'url'=>array('admin')),
);
?>

<h1>View AccountableUser #<?php echo $model->USER_ID; ?></h1>

<?php $this->widget('zii.widgets.CDetailView', array(
      'data'=>$model,
      'attributes'=>array(
            'USER_ID',
            'FIRST_NAME',
            'LAST_NAME',
            'USER_INITIAL',
            'BIRTHDAY',
            'USERNAME',
            'PASSWORD',
      ),
)); ?>

<?php
/* @var $this FsReportController */
/* @var $model FsReport */
/* @var $form CActiveForm */
?>

<div class="form">

<?php $form=$this->beginWidget('CActiveForm', array(
      'id'=>'fs-report-form',
      'enableAjaxValidation'=>false,
)); ?>

      <p class="note">Fields with <span class="required">*</span> are
required.</p>

      <?php echo CHtml::errorSummary(array($modelReport,$modelFsReport));
?>

      <?php
```

```
	/* MAP */

	// init the map
	Yii::import('ext.jquery-gmap.*');
	$gmap = new EGmap3Widget();

	$options = array(
		'scaleControl' => true,
		'streetViewControl' => false,
		'zoom' => 16,
		'center' => array(0, 0),
		'mapTypeId' => EGmap3MapTypeId::ROADMAP,
		'mapTypeControlOptions' => array(
			'style' => EGmap3MapTypeControlStyle::DROPDOWN_MENU,
			'position' => EGmap3ControlPosition::TOP_CENTER,
		),
	);

	$gmap->setOptions($options);
	$gmap->setSize(600, 400);

	// create the marker
	$marker = new EGmap3Marker(array(
		'title' => 'Draggable address marker 1',
		'draggable' => true,
	));
	$marker->latLng = array(14.4505957,120.9314332);
	$marker->centerOnMap();

	// set the marker to relay its position information a model
	$marker->capturePosition(
		 // the model object
		 $modelLocationPoint,
		 // model's latitude property name
		 'LATITUDE',
		 // model's longitude property name
		 'LONGITUDE',
		 // Options set :
		 //   show the fields, defaults to hidden fields
		 //   update the fields during the marker drag event
		 array('visible','drag')
	);
	$gmap->add($marker);
	$gmap->renderMap();

	?>
	<br/>
	<?php
		//echo $form->textField($model,'RA_REPORT_STATUS_ID');
		echo $form->dropDownList($modelFsReport,'FS_CATEGORY_ID',
$this->getFsCategoryList());
	?>

	<br/>
```

```
	<div class="row">
		<?php echo '<b>Start Datetime yyyy-mm-dd hh:ss <br/></b>'; ?>
		<?php echo $form->textField($modelReport,'START_DATETIME'); ?>
		<?php echo $form->error($modelReport,'START_DATETIME'); ?>
	</div>

	<div class="row">
		<?php echo '<b>End Datetime yyyy-mm-dd hh:ss <br/></b>'; ?>
		<?php echo $form->textField($modelReport,'END_DATETIME'); ?>
		<?php echo $form->error($modelReport,'END_DATETIME'); ?>
	</div>


	<div class="row buttons">
		<?php echo CHtml::submitButton($modelReport->isNewRecord ?
'Create' : 'Save'); ?>
	</div>

<?php $this->endWidget(); ?>

</div><!-- form -->
<?php
/* @var $this FsReportController */
/* @var $model FsReport */
/* @var $form CActiveForm */
?>

<div class="wide form">

<?php $form=$this->beginWidget('CActiveForm', array(
	'action'=>Yii::app()->createUrl($this->route),
	'method'=>'get',
)); ?>

	<div class="row">
		<?php echo $form->label($model,'REPORT_ID'); ?>
		<?php echo $form->textField($model,'REPORT_ID'); ?>
	</div>

	<div class="row">
		<?php echo $form->label($model,'FS_CATEGORY_ID'); ?>
		<?php echo $form->textField($model,'FS_CATEGORY_ID'); ?>
	</div>

	<div class="row buttons">
		<?php echo CHtml::submitButton('Search'); ?>
	</div>

<?php $this->endWidget(); ?>

</div><!-- search-form -->
<?php
/* @var $this FsReportController */
/* @var $data FsReport */
```

```
$reportModel = report::model()->findByPk($data->REPORT_ID);
$fsReportModel = fsReport::model()->findByPk($data->REPORT_ID);
$locationPointModel = locationPoint::model()->findByPk($data->REPORT_ID);
?>

<div class="view">

	<b><?php echo CHtml::encode($data->getAttributeLabel('REPORT_ID'));
?>:</b>
	<?php echo CHtml::link(CHtml::encode($data->REPORT_ID),
array('view', 'id'=>$data->REPORT_ID)); ?>
	<br />

	<b><?php echo CHtml::encode('Category'); ?>:</b>
	<?php echo CHtml::encode($this->getCategoryName($fsReportModel-
>FS_CATEGORY_ID)); ?>
	<br />

	<b><?php echo CHtml::encode('Verified by System'); ?>:</b>
	<?php
	if($reportModel->IS_VERIFIED_BY_SYSTEM==1){
		echo CHtml::encode('True');
	}else{
		echo CHtml::encode('False');
	}
	?>
	<br />


</div>
<?php
/* @var $this FsReportController */
/* @var $model FsReport */

$this->breadcrumbs=array(
	'Fs Reports'=>array('index'),
	'Manage',
);

$this->menu=array(
	array('label'=>'List FsReport', 'url'=>array('index')),
	array('label'=>'Create FsReport', 'url'=>array('create')),
);

Yii::app()->clientScript->registerScript('search', "
$('.search-button').click(function(){
	$('.search-form').toggle();
	return false;
});
$('.search-form form').submit(function(){
	$('#fs-report-grid').yiiGridView('update', {
		data: $(this).serialize()
	});
	return false;
```

```
});
");
?>

<h1>Manage Fs Reports</h1>

<p>
You may optionally enter a comparison operator (<b>&lt;</b>, <b>&lt;=</b>,
<b>&gt;</b>, <b>&gt;=</b>, <b>&lt;&gt;</b>
or <b>=</b>) at the beginning of each of your search values to specify how
the comparison should be done.
</p>

<?php echo CHtml::link('Advanced Search','#',array('class'=>'search-
button')); ?>
<div class="search-form" style="display:none">
<?php $this->renderPartial('_search',array(
      'model'=>$model,
)); ?>
</div><!-- search-form -->

<?php $this->widget('zii.widgets.grid.CGridView', array(
      'id'=>'fs-report-grid',
      'dataProvider'=>$model->search(),
      'filter'=>$model,
      'columns'=>array(
            'REPORT_ID',
            'FS_CATEGORY_ID',
            array(
                  'class'=>'CButtonColumn',
            ),
      ),
)); ?>

<?php
/* @var $this FsReportController */
/* @var $model FsReport */

$this->breadcrumbs=array(
      'Fs Reports'=>array('index'),
      'Create',
);

$this->menu=array(
      array('label'=>'List FsReport', 'url'=>array('index')),
      array('label'=>'Manage FsReport', 'url'=>array('admin')),
);
?>

<h1>Create FsReport</h1>

<?php echo $this->renderPartial('_form', array(

      'modelReport'=>$modelReport,
```

```
	'modelFsReport'=>$modelFsReport,

	'modelLocationPoint'=>$modelLocationPoint,
                                                                ));
?>
<?php
/* @var $this FsReportController */
/* @var $dataProvider CActiveDataProvider */

$this->breadcrumbs=array(
	'Fs Reports',
);

$this->menu=array(
	array('label'=>'Create FsReport', 'url'=>array('create')),
	array('label'=>'Manage FsReport', 'url'=>array('admin')),
);
?>

<h1>Fs Reports</h1>

<?php
/**
 * EGmap3 Yii extension example view file.
 *
 * You can copy this file or its contents into your Yii
 * application for testing.
 *
 */
Yii::import('ext.jquery-gmap.*');

$gmap = new EGmap3Widget();
$gmap->setSize(600, 400);

// base options
$options = array(
	'scaleControl' => true,
	'streetViewControl' => false,
	'zoom' => 16,
	'center' => array(0, 0),
	'mapTypeId' => EGmap3MapTypeId::ROADMAP,
	'mapTypeControlOptions' => array(
		'style' => EGmap3MapTypeControlStyle::DROPDOWN_MENU,
		'position' => EGmap3ControlPosition::TOP_CENTER,
	),
);
$gmap->setOptions($options);

$length = count($floodedStreets);
for($i =0; $i < $length; $i++){
	$reportModel = $floodedStreets[$i];
	$fsModel = FsReport::model()->findByPk($reportModel->REPORT_ID);
```

```
      $locationPointModel = LocationPoint::model()->findByPk($reportModel-
>LOCATION_POINT_ID);

      // marker with custom icon
      $marker = new EGmap3Marker(array(
                  'title' => $reportModel->REPORT_ID,
                  'icon' => array(
                        'url' =>
'http://agila.upm.edu.ph/~mfmagpili/sp/icons/flood.png',
                        'anchor' => array('x' => 1, 'y' => 36),
                  //'anchor' => new EGmap3Point(5,5),
                  )
            ));

      // set marker position by address
      $marker->latLng = array($locationPointModel-
>LATITUDE,$locationPointModel->LONGITUDE);

      // data associated with the marker
      $marker->data = $reportModel->REPORT_ID.'<br/>'.$this-
>getCategoryName($fsModel->FS_CATEGORY_ID);

      // add a Javascript event on marker click
      $js = "function(marker, event, data){
                  var map = $(this).gmap3('get'),
                  infowindow = $(this).gmap3({action:'get',
name:'infowindow'});
                  if (infowindow){
                        infowindow.open(map, marker);
                        infowindow.setContent(data);
                  } else {
                        $(this).gmap3({action:'addinfowindow',
anchor:marker, options:{content: data}});
                  }
      }";
      $marker->addEvent('click', $js);

      // center the map on the marker
      $marker->centerOnMap();

      $gmap->add($marker);
}

echo "<center>";
$gmap->renderMap();
echo "</center>";
?>




<?php $this->widget('zii.widgets.CListView', array(
      'dataProvider'=>$dataProvider,
      'itemView'=>'_view',
```

```
      'sortableAttributes'=>array(
        'REPORT_ID'=>'TIME CREATED',
           'IS_VERIFIED_BY_SYSTEM'=>'System Verified',
    ),
)); ?>

<?php
/* @var $this FsReportController */
/* @var $model FsReport */

$this->breadcrumbs=array(
      'Fs Reports'=>array('index'),
      $model->REPORT_ID=>array('view','id'=>$model->REPORT_ID),
      'Update',
);

$this->menu=array(
      array('label'=>'List FsReport', 'url'=>array('index')),
      array('label'=>'Create FsReport', 'url'=>array('create')),
      array('label'=>'View FsReport', 'url'=>array('view', 'id'=>$model-
>REPORT_ID)),
      array('label'=>'Manage FsReport', 'url'=>array('admin')),
);
?>

<h1>Update FsReport <?php echo $model->REPORT_ID; ?></h1>

<?php echo $this->renderPartial('_form',
array('modelFsReport'=>$model,'modelReport'=>$modelReport,'modelLocationPo
int'=>$modelLocationPoint)); ?>
<?php
/**
 * EGmap3 Yii extension example view file.
 *
 * You can copy this file or its contents into your Yii
 * application for testing.
 *
 */
Yii::import('ext.jquery-gmap.*');

$gmap = new EGmap3Widget();
$gmap->setSize('100%', 240);

// base options
$options = array(
      'scaleControl' => true,
      'streetViewControl' => false,
      'zoom' => 16,
      'center' => array(0, 0),
      'mapTypeId' => EGmap3MapTypeId::ROADMAP,
      'mapTypeControlOptions' => array(
            'style' => EGmap3MapTypeControlStyle::DROPDOWN_MENU,
            'position' => EGmap3ControlPosition::TOP_CENTER,
      ),
```

```
);
$gmap->setOptions($options);

// marker with custom icon
$marker = new EGmap3Marker(array(
                'title' => 'hello',
                'icon' => array(
                      'url' =>
'http://agila.upm.edu.ph/~mfmagpili/sp/icons/flood.png',
                      'anchor' => array('x' => 1, 'y' => 36),
                //'anchor' => new EGmap3Point(5,5),
                )
           ));

// set marker position by address
$lat = $locationPoint->LATITUDE;
$lng = $locationPoint->LONGITUDE;
$marker->latLng = array($lat,$lng);

// data associated with the marker
$marker->data = 'test data !';

// add a Javascript event on marker click
$js = "function(marker, event, data){
        var map = $(this).gmap3('get'),
        infowindow = $(this).gmap3({action:'get', name:'infowindow'});
        if (infowindow){
            infowindow.open(map, marker);
            infowindow.setContent(data);
        } else {
            $(this).gmap3({action:'addinfowindow', anchor:marker,
options:{content: data}});
        }
}";
$marker->addEvent('click', $js);

// center the map on the marker
$marker->centerOnMap();

$gmap->add($marker);
?>

<?php
/* @var $this FsReportController */
/* @var $model FsReport */

$this->breadcrumbs=array(
      'Fs Reports'=>array('index'),
      $model->REPORT_ID,
);

$this->menu=array(
      array('label'=>'Verify Report',
'url'=>array('VerifyReport','id'=>$model->REPORT_ID),
```

```
'visible'=>Yii::app()->user-
>checkAccess('trafficOperationsCenterOfficer')),
      array('label'=>'List Flooded Street Reports', 'url'=>array('index'),
'visible'=>Yii::app()->user-
>checkAccess('trafficOperationsCenterOfficer')),
      //array('label'=>'Create FsReport', 'url'=>array('create')),
      array('label'=>'Update Report', 'url'=>array('update', 'id'=>$model-
>REPORT_ID), 'visible'=>Yii::app()->user-
>checkAccess('trafficOperationsCenterOfficer')),
      array('label'=>'Delete Report', 'url'=>'#',
'linkOptions'=>array('submit'=>array('delete','id'=>$model-
>REPORT_ID),'confirm'=>'Are you sure you want to delete this item?'),
'visible'=>Yii::app()->user-
>checkAccess('trafficOperationsCenterOfficer')),
      //array('label'=>'Manage FsReport', 'url'=>array('admin')),
);

$categoryName = FsCategory::model()->findByPk($model->FS_CATEGORY_ID)-
>NAME;
?>

<h1>View Flooded Street Report #<?php echo $model->REPORT_ID; ?></h1>

<?php
$reportModel = report::model()->findByPk($model->REPORT_ID);
$isVerifiedBySystem;
if($reportModel->IS_VERIFIED_BY_SYSTEM==1){
      $isVerifiedBySystem = 'True';
}else{
      $isVerifiedBySystem = 'False';
}

$this->widget('zii.widgets.CDetailView', array(
      'data'=>$model,
      'attributes'=>array(
            'REPORT_ID',
            array(
                  'name'=>'Date and Time Created',
                  'value'=>CHtml::encode($categoryName)
            ),
            array(
            'name'=>CHtml::encode('START DATETIME'),
            'value'=>CHtml::encode($reportModel->START_DATETIME)
            ),
            array(
            'name'=>CHtml::encode('END DATETIME'),
            'value'=>CHtml::encode($reportModel->END_DATETIME)
            ),
            array(
            'name'=>CHtml::encode('Verified by System'),
            'value'=>CHtml::encode($isVerifiedBySystem)
            ),
      ),
)); ?>
```

```
<?php
$gmap->renderMap();
?>
<?php
/* @var $this JurisdictionController */
/* @var $model Jurisdiction */
/* @var $form CActiveForm */
?>

<div class="form">

<?php $form=$this->beginWidget('CActiveForm', array(
      'id'=>'jurisdiction-form',
      'enableAjaxValidation'=>false,
)); ?>

      <p class="note">Fields with <span class="required">*</span> are
required.</p>

      <?php echo $form->errorSummary($model); ?>
<?php
      /* MAP */

      // init the map
      Yii::import('ext.jquery-gmap.*');
      $gmap = new EGmap3Widget();
      $temp;
      $options = array(
            'scaleControl' => true,
            'streetViewControl' => false,
            'zoom' => 16,
            'center' => array(0, 0),
            'mapTypeId' => EGmap3MapTypeId::ROADMAP,
            'mapTypeControlOptions' => array(
                  'style' => EGmap3MapTypeControlStyle::DROPDOWN_MENU,
                  'position' => EGmap3ControlPosition::TOP_CENTER,
            ),
      );

      $gmap->setOptions($options);
      $gmap->setSize('100%', 200);

      // create the marker
      $marker = new EGmap3Marker(array(
            'title' => 'Draggable address marker 1',
            'draggable' => true,
      ));
      $marker->latLng = array(14.4505957,120.9314332);
      $marker->centerOnMap();

      // set the marker to relay its position information a model
      $marker->displayPosition(
             // the model object
```

```
        $model,
        // model's latitude property name
        'LL_X',
        // model's longitude property name
        'LL_Y',
        // Options set :
        //   show the fields, defaults to hidden fields
        //   update the fields during the marker drag event
        array('visible','drag')
);


$gmap->add($marker);
$gmap->renderMap();
?>

<div class="row">
      <?php echo $form->labelEx($model,'LL_X'); ?>
      <?php echo $form->textField($model,'LL_X'); ?>
      <?php echo $form->error($model,'LL_X'); ?>
</div>

<div class="row">
      <?php echo $form->labelEx($model,'LL_Y'); ?>
      <?php echo $form->textField($model,'LL_Y'); ?>
      <?php echo $form->error($model,'LL_Y'); ?>
</div>

<div class="row">
      <?php echo $form->labelEx($model,'LR_X'); ?>
      <?php echo $form->textField($model,'LR_X'); ?>
      <?php echo $form->error($model,'LR_X'); ?>
</div>

<div class="row">
      <?php echo $form->labelEx($model,'LR_Y'); ?>
      <?php echo $form->textField($model,'LR_Y'); ?>
      <?php echo $form->error($model,'LR_Y'); ?>
</div>

<div class="row">
      <?php echo $form->labelEx($model,'UL_X'); ?>
      <?php echo $form->textField($model,'UL_X'); ?>
      <?php echo $form->error($model,'UL_X'); ?>
</div>

<div class="row">
      <?php echo $form->labelEx($model,'UL_Y'); ?>
      <?php echo $form->textField($model,'UL_Y'); ?>
      <?php echo $form->error($model,'UL_Y'); ?>
</div>

<div class="row">
      <?php echo $form->labelEx($model,'UR_X'); ?>
```

```
            <?php echo $form->textField($model,'UR_X'); ?>
            <?php echo $form->error($model,'UR_X'); ?>
      </div>

      <div class="row">
            <?php echo $form->labelEx($model,'UR_Y'); ?>
            <?php echo $form->textField($model,'UR_Y'); ?>
            <?php echo $form->error($model,'UR_Y'); ?>
      </div>

      <div class="row">
            <?php echo $form->labelEx($model,'NAME'); ?>
            <?php echo $form->textArea($model,'NAME',array('rows'=>6,
'cols'=>50)); ?>
            <?php echo $form->error($model,'NAME'); ?>
      </div>

      <div class="row">
            <?php echo $form->labelEx($model,'DESCRIPTION'); ?>
            <?php echo $form-
>textArea($model,'DESCRIPTION',array('rows'=>6, 'cols'=>50)); ?>
            <?php echo $form->error($model,'DESCRIPTION'); ?>
      </div>

      <div class="row buttons">
            <?php echo CHtml::submitButton($model->isNewRecord ? 'Create'
: 'Save'); ?>
      </div>

<?php $this->endWidget(); ?>

</div><!-- form -->
<?php
/* @var $this JurisdictionController */
/* @var $model Jurisdiction */
/* @var $form CActiveForm */
?>

<div class="wide form">

<?php $form=$this->beginWidget('CActiveForm', array(
      'action'=>Yii::app()->createUrl($this->route),
      'method'=>'get',
)); ?>

      <div class="row">
            <?php echo $form->label($model,'JURISDICTION_ID'); ?>
            <?php echo $form->textField($model,'JURISDICTION_ID'); ?>
      </div>

      <div class="row">
            <?php echo $form->label($model,'LL_X'); ?>
            <?php echo $form->textField($model,'LL_X'); ?>
      </div>
```

```
	<div class="row">
		<?php echo $form->label($model,'LL_Y'); ?>
		<?php echo $form->textField($model,'LL_Y'); ?>
	</div>

	<div class="row">
		<?php echo $form->label($model,'LR_X'); ?>
		<?php echo $form->textField($model,'LR_X'); ?>
	</div>

	<div class="row">
		<?php echo $form->label($model,'LR_Y'); ?>
		<?php echo $form->textField($model,'LR_Y'); ?>
	</div>

	<div class="row">
		<?php echo $form->label($model,'UL_X'); ?>
		<?php echo $form->textField($model,'UL_X'); ?>
	</div>

	<div class="row">
		<?php echo $form->label($model,'UL_Y'); ?>
		<?php echo $form->textField($model,'UL_Y'); ?>
	</div>

	<div class="row">
		<?php echo $form->label($model,'UR_X'); ?>
		<?php echo $form->textField($model,'UR_X'); ?>
	</div>

	<div class="row">
		<?php echo $form->label($model,'UR_Y'); ?>
		<?php echo $form->textField($model,'UR_Y'); ?>
	</div>

	<div class="row">
		<?php echo $form->label($model,'NAME'); ?>
		<?php echo $form->textArea($model,'NAME',array('rows'=>6,
'cols'=>50)); ?>
	</div>

	<div class="row">
		<?php echo $form->label($model,'DESCRIPTION'); ?>
		<?php echo $form-
>textArea($model,'DESCRIPTION',array('rows'=>6, 'cols'=>50)); ?>
	</div>

	<div class="row buttons">
		<?php echo CHtml::submitButton('Search'); ?>
	</div>

<?php $this->endWidget(); ?>
```

```
</div><!-- search-form -->
<?php
/* @var $this JurisdictionController */
/* @var $data Jurisdiction */
?>

<div class="view">

	<b><?php echo CHtml::encode($data-
>getAttributeLabel('JURISDICTION_ID')); ?>:</b>
	<?php echo CHtml::link(CHtml::encode($data->JURISDICTION_ID),
array('view', 'id'=>$data->JURISDICTION_ID)); ?>
	<br />

	<b><?php echo CHtml::encode($data->getAttributeLabel('NAME'));
?>:</b>
	<?php echo CHtml::encode($data->NAME); ?>
	<br />

	<?php /*
	<b><?php echo CHtml::encode($data->getAttributeLabel('UR_X'));
?>:</b>
	<?php echo CHtml::encode($data->UR_X); ?>
	<br />

	<b><?php echo CHtml::encode($data->getAttributeLabel('UR_Y'));
?>:</b>
	<?php echo CHtml::encode($data->UR_Y); ?>
	<br />

	<b><?php echo CHtml::encode($data-
>getAttributeLabel('DESCRIPTION')); ?>:</b>
	<?php echo CHtml::encode($data->DESCRIPTION); ?>
	<br />

	*/ ?>

</div>
<?php
/* @var $this JurisdictionController */
/* @var $model Jurisdiction */

$this->breadcrumbs=array(
	'Jurisdictions'=>array('index'),
	'Manage',
);

$this->menu=array(
	array('label'=>'List Jurisdiction', 'url'=>array('index')),
	array('label'=>'Create Jurisdiction', 'url'=>array('create')),
);

Yii::app()->clientScript->registerScript('search', "
$('.search-button').click(function(){
```

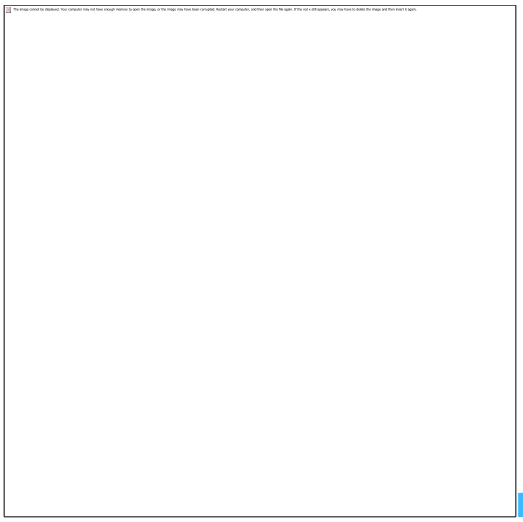
```
	$('.search-form').toggle();
	return false;
});
$('.search-form form').submit(function(){
	$('#jurisdiction-grid').yiiGridView('update', {
		data: $(this).serialize()
	});
	return false;
});
");
?>

<h1>Manage Jurisdictions</h1>

<p>
You may optionally enter a comparison operator (<b>&lt;</b>, <b>&lt;=</b>,
<b>&gt;</b>, <b>&gt;=</b>, <b>&lt;&gt;</b>
or <b>=</b>) at the beginning of each of your search values to specify how
the comparison should be done.
</p>

<?php echo CHtml::link('Advanced Search','#',array('class'=>'search-
button')); ?>
<div class="search-form" style="display:none">
<?php $this->renderPartial('_search',array(
	'model'=>$model,
)); ?>
</div><!-- search-form -->

<?php $this->widget('zii.widgets.grid.CGridView', array(
	'id'=>'jurisdiction-grid',
	'dataProvider'=>$model->search(),
	'filter'=>$model,
	'columns'=>array(
		'JURISDICTION_ID',
		'LL_X',
		'LL_Y',
		'LR_X',
		'LR_Y',
		'UL_X',
		/*
		'UL_Y',
		'UR_X',
		'UR_Y',
		'NAME',
		'DESCRIPTION',
		*/
		array(
			'class'=>'CButtonColumn',
		),
	),
)); ?>

<?php
```

```
/* @var $this JurisdictionController */
/* @var $model Jurisdiction */

$this->breadcrumbs=array(
      'Jurisdictions'=>array('index'),
      'Create',
);

$this->menu=array(
      array('label'=>'List Jurisdiction', 'url'=>array('index')),
      //array('label'=>'Manage Jurisdiction', 'url'=>array('admin')),
);
?>

<h1>Create Jurisdiction</h1>

<?php echo $this->renderPartial('_form', array('model'=>$model)); ?>
<?php
/* @var $this JurisdictionController */
/* @var $dataProvider CActiveDataProvider */

$this->breadcrumbs=array(
      'Jurisdictions',
);

$this->menu=array(
      array('label'=>'Create Jurisdiction', 'url'=>array('create')),
      //array('label'=>'Manage Jurisdiction', 'url'=>array('admin')),
);
?>

<h1>Jurisdictions</h1>

<?php $this->widget('zii.widgets.CListView', array(
      'dataProvider'=>$dataProvider,
      'itemView'=>'_view',
)); ?>

<?php
/* @var $this JurisdictionController */
/* @var $model Jurisdiction */

$this->breadcrumbs=array(
      'Jurisdictions'=>array('index'),
      $model->NAME=>array('view','id'=>$model->JURISDICTION_ID),
      'Update',
);

$this->menu=array(
      array('label'=>'List Jurisdiction', 'url'=>array('index')),
      //array('label'=>'Create Jurisdiction', 'url'=>array('create')),
      array('label'=>'View Jurisdiction', 'url'=>array('view',
'id'=>$model->JURISDICTION_ID)),
      //array('label'=>'Manage Jurisdiction', 'url'=>array('admin')),
```

```
);
?>

<h1>Update Jurisdiction <?php echo $model->JURISDICTION_ID; ?></h1>

<?php echo $this->renderPartial('_form', array('model'=>$model)); ?>
<?php
/* @var $this JurisdictionController */
/* @var $model Jurisdiction */

$this->breadcrumbs=array(
      'Jurisdictions'=>array('index'),
      $model->NAME,
);

$this->menu=array(
      array('label'=>'List Jurisdiction', 'url'=>array('index')),
      array('label'=>'Create Jurisdiction', 'url'=>array('create')),
      array('label'=>'Update Jurisdiction', 'url'=>array('update',
'id'=>$model->JURISDICTION_ID)),
      //array('label'=>'Delete Jurisdiction', 'url'=>'#',
'linkOptions'=>array('submit'=>array('delete','id'=>$model-
>JURISDICTION_ID),'confirm'=>'Are you sure you want to delete this
item?')),
      //array('label'=>'Manage Jurisdiction', 'url'=>array('admin')),
);
?>

<h1>View Jurisdiction #<?php echo $model->JURISDICTION_ID; ?></h1>

<?php $this->widget('zii.widgets.CDetailView', array(
      'data'=>$model,
      'attributes'=>array(
            'JURISDICTION_ID',
            'LL_X',
            'LL_Y',
            'LR_X',
            'LR_Y',
            'UL_X',
            'UL_Y',
            'UR_X',
            'UR_Y',
            'NAME',
            'DESCRIPTION',
      ),
)); ?>
<?php

/**
 * EGmap3 Yii extension example view file.
 *
 * You can copy this file or its contents into your Yii
 * application for testing.
 *
```

```
 */
Yii::import('ext.jquery-gmap.*');

$gmap = new EGmap3Widget();
$gmap->setSize('100%', 400);

// base options
$options = array(
      'scaleControl' => true,
      'streetViewControl' => false,
      'zoom' => 16,
      'center' => array(0, 0),
      'mapTypeId' => EGmap3MapTypeId::ROADMAP,
      'mapTypeControlOptions' => array(
            'style' => EGmap3MapTypeControlStyle::DROPDOWN_MENU,
            'position' => EGmap3ControlPosition::TOP_CENTER,
      ),
      'zoomControlOptions' => array(
            'style' => EGmap3ZoomControlStyle::SMALL,
            'position' => EGmap3ControlPosition::BOTTOM_CENTER
      ),
);
$gmap->setOptions($options);

$marker = new EGmap3Marker(array(
      'title' => 'Draggable address marker 1',
      'draggable' => true,
));
$marker->latLng = array($model->LL_X, $model->LL_Y);
$marker->centerOnMap();
$gmap->add($marker);

$polyOptions = array(
      'fillColor' => 'yellow',
      'strokeColor' => 'red'
);
/*
$rectangleOptions = array_merge(
            $polyOptions, array('bounds' => array(
                                                                  $model->X1,
                                                                  $model->Y1,
                                                                  $model->X2,
                                                                  $model->Y2

            ))
);
$rectangle = new EGmap3Rectangle($rectangleOptions);
*/
$polygon = new EGmap3Polygon($polyOptions);
$polygon->paths = array(
      array($model->LL_X, $model->LL_Y),
      array($model->LR_X, $model->LR_Y),
      array($model->UR_X, $model->UR_Y),
      array($model->UL_X, $model->UL_Y),
);
```

```
$gmap->add($polygon);

// adding a route automatically centers it on the map, overriding all
other
// centering calls
/*
$route = new EGmap3Route(array(
                  'origin' => 'New York, NY',
                  'destination' => 'Acapulco, México',
            ));
$gmap->add($route);
*/
$gmap->renderMap();
?>

<?php /* @var $this Controller */ ?>
<?php $this->beginContent('//layouts/main'); ?>
<div id="content">
      <?php echo $content; ?>
</div><!-- content -->
<?php $this->endContent(); ?>
<?php /* @var $this Controller */ ?>
<?php $this->beginContent('//layouts/main'); ?>
<div class="span-19">
      <div id="content">
            <?php echo $content; ?>
      </div><!-- content -->
</div>
<div class="span-5 last">
      <div id="sidebar">
      <?php
            $this->beginWidget('zii.widgets.CPortlet', array(
                  'title'=>'Operations',
            ));
            $this->widget('zii.widgets.CMenu', array(
                  'items'=>$this->menu,
                  'htmlOptions'=>array('class'=>'operations'),
            ));
            $this->endWidget();
      ?>
      </div><!-- sidebar -->
</div>
<?php $this->endContent(); ?>
```



name); ?>

widget('zii.widgets.CMenu',array('activeCssClass'=>'active', 'activateParents'=>true, 'items'=>array(array('label'=>'Home', 'url'=>array('/site/index')), array('label'=>'DBSCAN', 'url'=>array('/site/page', 'view'=>'about'),'visible'=>Yii::app()->user->isGuest), array('label'=>'Traffic Incidents', 'url'=>array('/company/index'), 'items'=>array(array('label'=>'Road Accidents', 'url'=>array('raReport/index'), 'visible'=>Yii::app()->user->checkAccess('readRoadAccident')), array('label'=>'Flooded Roads', 'url'=>array('/company/aboutUs')), array('label'=>'Closed Roads', 'url'=>array('RoadClosedReport/index'), 'visible'=>Yii::app()->user->checkAccess('readRoadClosed')),),), array('label'=>'Manage Accounts', 'url'=>array('/company/index'), 'items'=>array(array('label'=>'Administrator', 'url'=>array('/company/index')), array('label'=>'Operations Center Officer', 'url'=>array('/company/aboutUs')), array('label'=>'Traffic Investigator', 'url'=>array('/company/careers')),),), array('label'=>'Contact', 'url'=>array('/site/contact'),'visible'=>Yii::app()->user->isGuest), //array('label'=>'Road Accidents', 'url'=>array('raReport/index'), 'visible'=>Yii::app()->user->checkAccess('readRoadAccident')), //array('label'=>'Closed Roads', 'url'=>array('RoadClosedReport/index'), 'visible'=>Yii::app()->user->checkAccess('readRoadClosed')), array('label'=>'Virtual Trip Lines', 'url'=>array('/virtualTripLine/index'), 'visible'=>Yii::app()->user->checkAccess('readVirtualTripLines')), array('label'=>'Login', 'url'=>array('/site/login'), 'visible'=>Yii::app()->user->isGuest), array('label'=>'System Administrator', 'url'=>array('systemAdministrator/index'), 'visible'=>Yii::app()->user->checkAccess('manageAccounts')), array('label'=>'Traffic Investigator', 'url'=>array('trafficInvestigator/index'), 'visible'=>Yii::app()->user->checkAccess('manageAccounts')), array('label'=>'Operation Center Officer', 'url'=>array('TrafficOperationsCenterOfficer/index'), 'visible'=>Yii::app()->user->checkAccess('manageAccounts')), array('label'=>'Logout ('.Yii::app()->user->name.')', 'url'=>array('/site/logout'), 'visible'=>!Yii::app()->user->isGuest),),)); */ $cs=Yii::app()->getClientScript(); $scriptUrl = Yii::app()->getAssetManager()->publish(Yii::getPathOfAlias('application.extensions.eflatmenu.resources')); $cs->registerCssFile($scriptUrl . '/eflatmenu.css'); $cs->registerCssFile($scriptUrl . '/font-awesome/font-awesome.css'); $cs->registerScriptFile($scriptUrl . '/eflatmenu.js'); $this->widget('application.extensions.eflatmenu.EFlatMenu', array('items'=>array(array('label'=>'Home', 'url'=>array('/site/index')), array('label'=>'DBSCAN', 'url'=>array('/site/page', 'view'=>'about'),'visible'=>Yii::app()->user->isGuest), array('label'=>'Traffic Incidents', 'url'=>array('#'), 'visible'=>Yii::app()->user->checkAccess('manageTrafficIncidents'), 'items'=>array(array('label'=>'Road Accidents', 'url'=>array('raReport/index'), 'visible'=>Yii::app()->user->checkAccess('readRoadAccident')),

array('label'=>'Closed Roads', 'url'=>array('RoadClosedReport/index'), 'visible'=>Yii::app()->user->checkAccess('readRoadClosed')), array('label'=>'Flooded Roads', 'url'=>array('FsReport/index'), 'visible'=>Yii::app()->user->checkAccess('trafficOperationsCenterOfficer')),),), array('label'=>'Manage Accounts', 'url'=>array('#'), 'visible'=>Yii::app()->user->checkAccess('manageAccounts'), 'items'=>array(array('label'=>'System Administrator', 'url'=>array('systemAdministrator/index'), 'visible'=>Yii::app()->user->checkAccess('manageAccounts')), array('label'=>'Traffic Investigator', 'url'=>array('trafficInvestigator/index'), 'visible'=>Yii::app()->user->checkAccess('manageAccounts')), array('label'=>'Operation Center Officer', 'url'=>array('TrafficOperationsCenterOfficer/index'), 'visible'=>Yii::app()->user->checkAccess('manageAccounts')),),), array('label'=>'Traffic Investigator', 'url'=>array('trafficInvestigator/index'), 'visible'=>Yii::app()->user->checkAccess('trafficOperationsCenterOfficer')), //array('label'=>'Road Accidents', 'url'=>array('raReport/index'), 'visible'=>Yii::app()->user->checkAccess('viewRoadAccidentMenu')), array('label'=>'Report History', 'url'=>array('ReportHistory/index'), 'visible'=>Yii::app()->user->checkAccess('readReportHistory')), array('label'=>'Contact', 'url'=>array('/site/contact'),'visible'=>Yii::app()->user->isGuest), //array('label'=>'Road Accidents', 'url'=>array('raReport/index'), 'visible'=>Yii::app()->user->checkAccess('readRoadAccident')), //array('label'=>'Closed Roads', 'url'=>array('RoadClosedReport/index'), 'visible'=>Yii::app()->user->checkAccess('readRoadClosed')), array('label'=>'Virtual Trip Lines', 'url'=>array('/virtualTripLine/index'), 'visible'=>Yii::app()->user->checkAccess('readVirtualTripLines')), array('label'=>'Jurisdictions', 'url'=>array('/jurisdiction/index'), 'visible'=>Yii::app()->user->checkAccess('trafficOperationsCenterOfficer')), array('label'=>'Login', 'url'=>array('/site/login'), 'visible'=>Yii::app()->user->isGuest), //array('label'=>'System Administrator', 'url'=>array('systemAdministrator/index'), 'visible'=>Yii::app()->user->checkAccess('manageAccounts')), //array('label'=>'Traffic Investigator', 'url'=>array('trafficInvestigator/index'), 'visible'=>Yii::app()->user->checkAccess('manageAccounts')), //array('label'=>'Operation Center Officer', 'url'=>array('TrafficOperationsCenterOfficer/index'), 'visible'=>Yii::app()->user->checkAccess('manageAccounts')), array('label'=>'Logout ('.Yii::app()->user->name.')', 'url'=>array('/site/logout'), 'visible'=>!Yii::app()->user->isGuest),),)); ?>

breadcrumbs)):?> widget('zii.widgets.CBreadcrumbs', array('links'=>$this->breadcrumbs,)); ?>


```
<?php
$this->widget('zii.widgets.CMenu',array(
  'activeCssClass'=>'active',
  'activateParents'=>true,
  'items'=>array(
    array(
      'label'=>'Company',
      'url'=>array('/company/index'),
      'linkOptions'=>array('id'=>'menuCompany'),
      'itemOptions'=>array('id'=>'itemCompany'),
      'items'=>array(
        array('label'=>'Our Mission', 'url'=>array('/company/index')),
        array('label'=>'About Us', 'url'=>array('/company/aboutUs')),
        array('label'=>'Careers', 'url'=>array('/company/careers')),
        array('label'=>'Contact Us', 'url'=>array('/company/contactUs')),
```

```
        array('label'=>'Store Locator',
'url'=>array('/company/storeLocator')),
      ),
    ),
    array(
      'label'=>'Blog',
      'url'=>array('/blog/post/index'),
      'linkOptions'=>array('id'=>'menuBlog')
    ),
    array(
      'label'=>'Change',
      'url'=>array('/change/index'),
      'linkOptions'=>array('id'=>'menuChange'),
      'itemOptions'=>array('id'=>'itemChange'),
      'items'=>array(
        array('label'=>'Community Involvement',
'url'=>array('/change/index')),
        array('label'=>'Eco Responsibility',
'url'=>array('/change/ecoPolicy')),
        array('label'=>'Responsibility',
'url'=>array('/change/responsibility')),
      ),
    ),
    array(
      'label'=>'Shop',
      'url'=>array('/shop'),
      'linkOptions'=>array('id'=>'menuBuy')
    ),
  ),
));

?>
<?php
/* @var $this RaReportController */
/* @var $model RaReport */
/* @var $form CActiveForm */
?>

<div class="form">

<?php $form=$this->beginWidget('CActiveForm', array(
      'id'=>'ra-report-form',
      'enableAjaxValidation'=>false,
)); ?>

      <p class="note">Fields with <span class="required">*</span> are
required.</p>

      <?php echo CHtml::errorSummary(array($modelReport,$modelRaReport));
?>

<?php
      /* MAP */
```

```
	// init the map
	Yii::import('ext.jquery-gmap.*');
	$gmap = new EGmap3Widget();

	$options = array(
		'scaleControl' => true,
		'streetViewControl' => false,
		'zoom' => 16,
		'center' => array(0, 0),
		'mapTypeId' => EGmap3MapTypeId::ROADMAP,
		'mapTypeControlOptions' => array(
			'style' => EGmap3MapTypeControlStyle::DROPDOWN_MENU,
			'position' => EGmap3ControlPosition::TOP_CENTER,
		),
	);

	$gmap->setOptions($options);
	$gmap->setSize(600, 400);

	// create the marker
	$marker = new EGmap3Marker(array(
		'title' => 'Draggable address marker 1',
		'draggable' => true,
	));
	$marker->latLng = array(14.4505957,120.9314332);
	$marker->centerOnMap();

	// set the marker to relay its position information a model
	$marker->capturePosition(
		// the model object
		$modelLocationPoint,
		// model's latitude property name
		'LATITUDE',
		// model's longitude property name
		'LONGITUDE',
		// Options set :
		//   show the fields, defaults to hidden fields
		//   update the fields during the marker drag event
		array('visible','drag')
	);
	$gmap->add($marker);
	$gmap->renderMap();
	?>

	<div class="row">
		<?php echo $form-
>labelEx($modelRaReport,'RA_REPORT_STATUS_ID'); ?>

		<?php
		//echo $form->textField($model,'RA_REPORT_STATUS_ID');
		echo $form->dropDownList($modelRaReport,'RA_REPORT_STATUS_ID',
$this->getRaReportStatusList());
		?>
```

```
            <?php echo $form->error($modelRaReport,'RA_REPORT_STATUS_ID');
?>
      </div>

      <div class="row">
            <?php echo $form->labelEx($modelRaReport,'RA_CATEGORY_ID'); ?>

            <?php
            //echo $form->textField($model,'RA_REPORT_STATUS_ID');
            echo $form->dropDownList($modelRaReport,'RA_CATEGORY_ID',
$this->getRaReportCategoryList());
            ?>

            <?php echo $form->error($modelRaReport,'RA_CATEGORY_ID'); ?>
      </div>


      <br/>
      <div class="row">
            <?php echo '<b>Start Datetime yyyy-mm-dd hh:ss <br/></b>'; ?>
            <?php echo $form->textField($modelReport,'START_DATETIME'); ?>
            <?php echo $form->error($modelReport,'START_DATETIME'); ?>
      </div>

      <div class="row">
            <?php echo '<b>End Datetime yyyy-mm-dd hh:ss <br/></b>'; ?>
            <?php echo $form->textField($modelReport,'END_DATETIME'); ?>
            <?php echo $form->error($modelReport,'END_DATETIME'); ?>
      </div>


      <div class="row buttons">
            <?php echo CHtml::submitButton($modelReport->isNewRecord ?
'Create' : 'Save'); ?>
      </div>

<?php $this->endWidget(); ?>

</div><!-- form -->
<?php
/* @var $this RaReportController */
/* @var $model RaReport */
/* @var $form CActiveForm */
?>

<div class="wide form">

<?php $form=$this->beginWidget('CActiveForm', array(
      'action'=>Yii::app()->createUrl($this->route),
      'method'=>'get',
)); ?>

      <div class="row">
            <?php echo $form->label($model,'REPORT_ID'); ?>
```

```
            <?php echo $form->textField($model,'REPORT_ID'); ?>
      </div>

      <div class="row">
            <?php echo $form->label($model,'RA_REPORT_STATUS_ID'); ?>
            <?php echo $form->textField($model,'RA_REPORT_STATUS_ID'); ?>
      </div>

      <div class="row">
            <?php echo $form->label($model,'RA_CATEGORY_ID'); ?>
            <?php echo $form->textField($model,'RA_CATEGORY_ID'); ?>
      </div>

      <div class="row buttons">
            <?php echo CHtml::submitButton('Search'); ?>
      </div>

<?php $this->endWidget(); ?>

</div><!-- search-form -->
<?php
/* @var $this RaReportController */
/* @var $data RaReport */
$reportModel = report::model()->findByPk($data->REPORT_ID);
$raReportModel = raReport::model()->findByPk($data->REPORT_ID);
$locationPointModel = locationPoint::model()->findByPk($data->REPORT_ID);
?>

<div class="view">

      <b><?php echo CHtml::encode($data->getAttributeLabel('REPORT_ID'));
?>:</b>
      <?php echo CHtml::link(CHtml::encode($data->REPORT_ID),
array('view', 'id'=>$data->REPORT_ID)); ?>
      <br />

      <b><?php echo CHtml::encode($data->getAttributeLabel('STATUS'));
?>:</b>
      <?php echo CHtml::encode($this-
>getRaReportStatusName($raReportModel->RA_REPORT_STATUS_ID)); ?>
      <br />

      <b><?php echo CHtml::encode($data->getAttributeLabel('CATEGORY'));
?>:</b>
      <?php echo CHtml::encode($this->getRaCategoryName($raReportModel-
>RA_CATEGORY_ID)); ?>
      <br />

      <b><?php echo CHtml::encode('Verified by System'); ?>:</b>
      <?php
      if($reportModel->IS_VERIFIED_BY_SYSTEM==1){
            echo CHtml::encode('True');
      }else{
            echo CHtml::encode('False');
```

```
	}
	?>
	<br />

	<?php
	//Print_r($this->getRaReportStatusList());

	?>

</div>
<?php
/* @var $this RaReportController */
/* @var $model RaReport */

$this->breadcrumbs=array(
	'Ra Reports'=>array('index'),
	'Manage',
);

$this->menu=array(
	array('label'=>'List RaReport', 'url'=>array('index')),
	array('label'=>'Create RaReport', 'url'=>array('create')),
);

Yii::app()->clientScript->registerScript('search', "
$('.search-button').click(function(){
	$('.search-form').toggle();
	return false;
});
$('.search-form form').submit(function(){
	$('#ra-report-grid').yiiGridView('update', {
		data: $(this).serialize()
	});
	return false;
});
");
?>

<h1>Manage Ra Reports</h1>

<p>
You may optionally enter a comparison operator (<b>&lt;</b>, <b>&lt;=</b>,
<b>&gt;</b>, <b>&gt;=</b>, <b>&lt;&gt;</b>
or <b>=</b>) at the beginning of each of your search values to specify how
the comparison should be done.
</p>

<?php echo CHtml::link('Advanced Search','#',array('class'=>'search-
button')); ?>
<div class="search-form" style="display:none">
<?php $this->renderPartial('_search',array(
	'model'=>$model,
)); ?>
</div><!-- search-form -->
```

```
<?php $this->widget('zii.widgets.grid.CGridView', array(
      'id'=>'ra-report-grid',
      'dataProvider'=>$model->search(),
      'filter'=>$model,
      'columns'=>array(
            'REPORT_ID',
            'RA_REPORT_STATUS_ID',
            'RA_CATEGORY_ID',
            array(
                  'class'=>'CButtonColumn',
            ),
      ),
)); ?>

<?php
/* @var $this RaReportController */
/* @var $model RaReport */

$this->breadcrumbs=array(
      'Ra Reports'=>array('index'),
      'Create',
);

$this->menu=array(
      array('label'=>'List RaReport', 'url'=>array('index')),
      array('label'=>'Manage RaReport', 'url'=>array('admin')),
);
?>

<h1>Create RaReport</h1>

<?php echo $this->renderPartial('_form', array(

      'modelReport'=>$modelReport,

      'modelRaReport'=>$modelRaReport,

      'modelLocationPoint'=>$modelLocationPoint,
                                                ));
?>
<?php
/* @var $this RaReportController */
/* @var $dataProvider CActiveDataProvider */

$this->breadcrumbs=array(
      'Ra Reports',
);

$this->menu=array(
      array('label'=>'Create RaReport', 'url'=>array('create'),
'visible'=>Yii::app()->user-
>checkAccess('trafficOperationsCenterOfficer')),
```

```
      array('label'=>'Manage RaReport', 'url'=>array('admin'),
'visible'=>Yii::app()->user-
>checkAccess('trafficOperationsCenterOfficer')),
);
?>

<h1>Road Accident Reports</h1>

<?php
/**
 * EGmap3 Yii extension example view file.
 *
 * You can copy this file or its contents into your Yii
 * application for testing.
 *
 */
Yii::import('ext.jquery-gmap.*');

$gmap = new EGmap3Widget();
$gmap->setSize(600, 400);

// base options
$options = array(
      'scaleControl' => true,
      'streetViewControl' => false,
      'zoom' => 16,
      'center' => array(0, 0),
      'mapTypeId' => EGmap3MapTypeId::ROADMAP,
      'mapTypeControlOptions' => array(
            'style' => EGmap3MapTypeControlStyle::DROPDOWN_MENU,
            'position' => EGmap3ControlPosition::TOP_CENTER,
      ),
);
$gmap->setOptions($options);

$length = count($roadAccidents);
for($i =0; $i < $length; $i++){
      $reportModel = $roadAccidents[$i];

      $roadAccidentModel = RaReport::model()->findByPk($reportModel-
>REPORT_ID);
      $locationPointModel = LocationPoint::model()->findByPk($reportModel-
>LOCATION_POINT_ID);

      // marker with custom icon
      $marker = new EGmap3Marker(array(
                  'title' => $reportModel->REPORT_ID,
                  'icon' => array(
                        'url' => 'http://google-maps-
icons.googlecode.com/files/accident2.png',
                        'anchor' => array('x' => 1, 'y' => 36),
                  //'anchor' => new EGmap3Point(5,5),
                  )
            ));
```

```
      // set marker position by address
      $marker->latLng = array($locationPointModel-
>LATITUDE,$locationPointModel->LONGITUDE);

      // data associated with the marker
      $marker->data = $reportModel->REPORT_ID.'<br/>'.$this-
>getRaReportStatusName($roadAccidentModel-
>RA_REPORT_STATUS_ID).'<br/>'.$this->getRaCategoryName($roadAccidentModel-
>RA_CATEGORY_ID);

      // add a Javascript event on marker click
      $js = "function(marker, event, data){
                  var map = $(this).gmap3('get'),
                  infowindow = $(this).gmap3({action:'get',
name:'infowindow'});
                  if (infowindow){
                        infowindow.open(map, marker);
                        infowindow.setContent(data);
                  } else {
                        $(this).gmap3({action:'addinfowindow',
anchor:marker, options:{content: data}});
                  }
      }";
      $marker->addEvent('click', $js);

      // center the map on the marker
      $marker->centerOnMap();

      $gmap->add($marker);
}

echo "<center>";
$gmap->renderMap();
echo "</center>";
?>

<?php $this->widget('zii.widgets.CListView', array(
      'dataProvider'=>$dataProvider,
      'itemView'=>'_view',
      'sortableAttributes'=>array(
        'REPORT_ID',
            'IS_VERIFIED_BY_SYSTEM'=>'System Verified',
    ),
)); ?>

<?php
/* @var $this RaReportController */
/* @var $model RaReport */

$this->breadcrumbs=array(
      'Ra Reports'=>array('index'),
      $model->REPORT_ID=>array('view','id'=>$model->REPORT_ID),
      'Update',
```

```
);

$this->menu=array(
	array('label'=>'List RaReport', 'url'=>array('index')),
	array('label'=>'Create RaReport', 'url'=>array('create')),
	array('label'=>'View RaReport', 'url'=>array('view', 'id'=>$model-
>REPORT_ID)),
	array('label'=>'Manage RaReport', 'url'=>array('admin')),
);
?>

<h1>Update RaReport <?php echo $model->REPORT_ID; ?></h1>

<div class="form">

<?php $form=$this->beginWidget('CActiveForm', array(
	'id'=>'ra-report-form',
	'enableAjaxValidation'=>false,
)); ?>

	<p class="note">Fields with <span class="required">*</span> are
required.</p>

	<?php echo $form->errorSummary($model); ?>

	<div class="row">
		<?php echo $form->labelEx($model,'RA_REPORT_STATUS_ID'); ?>

		<?php
		//echo $form->textField($model,'RA_REPORT_STATUS_ID');
		echo $form->dropDownList($model,'RA_REPORT_STATUS_ID', $this-
>getRaReportStatusList());

		?>

		<?php echo $form->error($model,'RA_REPORT_STATUS_ID'); ?>
	</div>

	<div class="row buttons">
		<?php echo CHtml::submitButton($model->isNewRecord ? 'Create'
: 'Save'); ?>
	</div>

<?php $this->endWidget(); ?>

</div><!-- form -->
<h1>View Road Accident Report #<?php echo $model->REPORT_ID; ?></h1>
<center>
<?php
/**
 * EGmap3 Yii extension example view file.
 *
 * You can copy this file or its contents into your Yii
 * application for testing.
```

```
 *
 */
Yii::import('ext.jquery-gmap.*');

$gmap = new EGmap3Widget();
$gmap->setSize('100%', 240);

// base options
$options = array(
      'scaleControl' => true,
      'streetViewControl' => false,
      'zoom' => 16,
      'center' => array(0, 0),
      'mapTypeId' => EGmap3MapTypeId::ROADMAP,
      'mapTypeControlOptions' => array(
            'style' => EGmap3MapTypeControlStyle::DROPDOWN_MENU,
            'position' => EGmap3ControlPosition::TOP_CENTER,
      ),
);
$gmap->setOptions($options);

// marker with custom icon
$marker = new EGmap3Marker(array(
                  'title' => $model->REPORT_ID,
                  'icon' => array(
                        'url' => 'http://google-maps-
icons.googlecode.com/files/accident2.png',                        'anchor' =>
array('x' => 1, 'y' => 36),
                  //'anchor' => new EGmap3Point(5,5),
                  )
           ));

// set marker position by address
$lat = $locationPoint->LATITUDE;
$lng = $locationPoint->LONGITUDE;
$marker->latLng = array($lat,$lng);

// data associated with the marker
$marker->data = 'test data !';

// add a Javascript event on marker click
$js = "function(marker, event, data){
        var map = $(this).gmap3('get'),
        infowindow = $(this).gmap3({action:'get', name:'infowindow'});
        if (infowindow){
            infowindow.open(map, marker);
            infowindow.setContent(data);
        } else {
            $(this).gmap3({action:'addinfowindow', anchor:marker,
options:{content: data}});
        }
}";
$marker->addEvent('click', $js);
```

```
// center the map on the marker
$marker->centerOnMap();

$gmap->add($marker);
?>
</center>

<?php
/* @var $this RaReportController */
/* @var $model RaReport */

$this->breadcrumbs=array(
      'Ra Reports'=>array('index'),
      $model->REPORT_ID,
);
?>

<?php
$this->menu=array(
      array('label'=>'Verify Report',
'url'=>array('VerifyReport','id'=>$model->REPORT_ID),
'visible'=>Yii::app()->user-
>checkAccess('trafficOperationsCenterOfficer')),
      array('label'=>'List Road Accidents', 'url'=>array('index'),
'visible'=>Yii::app()->user-
>checkAccess('trafficOperationsCenterOfficer')),
      //array('label'=>'Create RaReport', 'url'=>array('create'),
'visible'=>Yii::app()->user-
>checkAccess('trafficOperationsCenterOfficer')),
      array('label'=>'Assign Traffic Investigator',
'url'=>array('AssignTrafficInvestigator', 'id'=>$model->REPORT_ID)),
      array('label'=>'Update Report', 'url'=>array('update', 'id'=>$model-
>REPORT_ID)),
      array('label'=>'Delete Report', 'url'=>'#',
'linkOptions'=>array('submit'=>array('delete','id'=>$model-
>REPORT_ID),'confirm'=>'Are you sure you want to delete this item?')),
      //array('label'=>'Manage RaReport', 'url'=>array('admin'),
'visible'=>Yii::app()->user->checkAccess('systemAdmin')),
);
?>

<?php
$reportModel = report::model()->findByPk($model->REPORT_ID);
$isVerifiedBySystem;
if($reportModel->IS_VERIFIED_BY_SYSTEM==1){
      $isVerifiedBySystem = 'True';
}else{
      $isVerifiedBySystem = 'False';
}

if($model->REPORT_ID < 5){
      $this->widget('zii.widgets.CDetailView', array(
            'data'=>$model,
            'attributes'=>array(
```

```
                array(
                'name'=>'Report ID',
                'value'=>CHtml::encode($model->REPORT_ID)
                ),
                array(
                'name'=>'STATUS',
                'value'=>CHtml::encode($model->rAREPORTSTATUS->NAME)
                ),
                array(
                'name'=>'CATEGORY',
                'value'=>CHtml::encode($model->rACATEGORY->NAME)
                ),
                array(
                'name'=>'Date and Time Created',
                'value'=>CHtml::encode($model->rEPORT->DATETIME_CREATED)
                ),
                /*
                array(
                'name'=>'Location',
                'value'=>CHtml::encode($marker->address)
                ),
                */
                array(
                'label'=>'Picture',
                'type'=>'raw',

      'value'=>html_entity_decode(CHtml::image('images/'.$report-
>PICTURE_ID.'.jpg','alt',array('width'=>341,'height'=>232))),
                ),
                array(
                'name'=>'Verified by System',
                'value'=>CHtml::encode($isVerifiedBySystem)
                )
          ),
      ));
}else if($report->PICTURE_ID != null){
      $this->widget('zii.widgets.CDetailView', array(
            'data'=>$model,
            'attributes'=>array(
                array(
                'name'=>'Report ID',
                'value'=>CHtml::encode($model->REPORT_ID)
                ),
                array(
                'name'=>'STATUS',
                'value'=>CHtml::encode($model->rAREPORTSTATUS->NAME)
                ),
                array(
                'name'=>'CATEGORY',
                'value'=>CHtml::encode($model->rACATEGORY->NAME)
                ),
                array(
                'name'=>'Date and Time Created',
                'value'=>CHtml::encode($model->rEPORT->DATETIME_CREATED)
```

```
                ),
                /*
                array(
                'name'=>'Location',
                'value'=>CHtml::encode($marker->address)
                ),
                */
                array(
                'name'=>CHtml::encode('START DATETIME'),
                'value'=>CHtml::encode($reportModel->START_DATETIME)
                ),
                array(
                'name'=>CHtml::encode('END DATETIME'),
                'value'=>CHtml::encode($reportModel->END_DATETIME)
                ),
                array(
                'name'=>'Verified by System',
                'value'=>CHtml::encode($isVerifiedBySystem)
                ),
                array(
                'label'=>'Picture',
                'type'=>'raw',

      'value'=>html_entity_decode(CHtml::image('images/'.$report-
>PICTURE_ID.'.jpeg','alt',array('width'=>341,'height'=>232))),
                )


           ),
      ));
}else{
      $this->widget('zii.widgets.CDetailView', array(
           'data'=>$model,
           'attributes'=>array(
                array(
                'name'=>'Report ID',
                'value'=>CHtml::encode($model->REPORT_ID)
                ),
                array(
                'name'=>'STATUS',
                'value'=>CHtml::encode($model->rAREPORTSTATUS->NAME)
                ),
                array(
                'name'=>'CATEGORY',
                'value'=>CHtml::encode($model->rACATEGORY->NAME)
                ),
                array(
                'name'=>'Date and Time Created',
                'value'=>CHtml::encode($model->rEPORT->DATETIME_CREATED)
                ),
                /*
                array(
                'name'=>'Location',
                'value'=>CHtml::encode($marker->address)
```

```
                ),
                */
                array(
                'name'=>CHtml::encode('START DATETIME'),
                'value'=>CHtml::encode($reportModel->START_DATETIME)
                ),
                array(
                'name'=>CHtml::encode('END DATETIME'),
                'value'=>CHtml::encode($reportModel->END_DATETIME)
                ),
                array(
                'name'=>'Verified by System',
                'value'=>CHtml::encode($isVerifiedBySystem)
                ),
                array(
                'label'=>'Picture',
                'value'=>'no picture',
                )

            ),
      ));
}

 ?>

<?php
$gmap->renderMap();
?>
<?php
/* @var $this ReportHistoryController */
/* @var $model ReportHistory */
/* @var $form CActiveForm */
?>

<div class="form">

<?php $form=$this->beginWidget('CActiveForm', array(
      'id'=>'report-history-form',
      'enableAjaxValidation'=>false,
)); ?>

      <p class="note">Fields with <span class="required">*</span> are
required.</p>

      <?php echo $form->errorSummary($model); ?>

      <div class="row">
            <?php echo $form->labelEx($model,'USER_ID'); ?>
            <?php
                  echo $form->dropDownList($model,'USER_ID', $this-
>getTrafficInvestigatorList($model->REPORT_ID));
            ?>
            <?php echo $form->error($model,'USER_ID'); ?>
      </div>
```

```
	<div class="row">
		<?php

		?>
	</div>

	<div class="row buttons">
		<?php echo CHtml::submitButton($model->isNewRecord ? 'Create'
: 'Save'); ?>
	</div>

<?php $this->endWidget(); ?>

</div><!-- form -->
<?php
/* @var $this ReportHistoryController */
/* @var $model ReportHistory */
/* @var $form CActiveForm */
?>

<div class="wide form">

<?php $form=$this->beginWidget('CActiveForm', array(
	'action'=>Yii::app()->createUrl($this->route),
	'method'=>'get',
)); ?>

	<div class="row">
		<?php echo $form->label($model,'REPORT_HISTORY_ID'); ?>
		<?php echo $form->textField($model,'REPORT_HISTORY_ID'); ?>
	</div>

	<div class="row">
		<?php echo $form->label($model,'USER_ID'); ?>
		<?php echo $form->textField($model,'USER_ID'); ?>
	</div>

	<div class="row">
		<?php echo $form->label($model,'REPORT_ID'); ?>
		<?php echo $form->textField($model,'REPORT_ID'); ?>
	</div>

	<div class="row buttons">
		<?php echo CHtml::submitButton('Search'); ?>
	</div>

<?php $this->endWidget(); ?>

</div><!-- search-form -->
<?php
/* @var $this ReportHistoryController */
/* @var $data ReportHistory */
$userModel = AccountableUser::model()->findByPk($data->USER_ID);
```

```
$userName = $userModel->FIRST_NAME.' '.$userModel->USER_INITIAL.'
'.$userModel->LAST_NAME;


?>

<div class="view">

	<b><?php echo CHtml::encode('History ID'); ?>:</b>
	<?php
	//echo CHtml::link(CHtml::encode($data->REPORT_HISTORY_ID),
array('view', 'id'=>$data->REPORT_HISTORY_ID));
	echo CHtml::encode(CHtml::encode($data->REPORT_HISTORY_ID));
	?>
	<br />

	<b><?php echo CHtml::encode('Report ID'); ?>:</b>
	<?php echo CHtml::encode($data->REPORT_ID); ?>
	<br />

	<b><?php echo CHtml::encode($data->getAttributeLabel('Assigned
to')); ?>:</b>
	<?php echo CHtml::encode($userName); ?>
	<br />




</div>
<?php
/* @var $this ReportHistoryController */
/* @var $model ReportHistory */

$this->breadcrumbs=array(
	'Report Histories'=>array('index'),
	'Manage',
);

$this->menu=array(
	array('label'=>'List ReportHistory', 'url'=>array('index')),
	array('label'=>'Create ReportHistory', 'url'=>array('create')),
);

Yii::app()->clientScript->registerScript('search', "
$('.search-button').click(function(){
	$('.search-form').toggle();
	return false;
});
$('.search-form form').submit(function(){
	$('#report-history-grid').yiiGridView('update', {
		data: $(this).serialize()
	});
	return false;
});
```

```
");
?>

<h1>Manage Report Histories</h1>

<p>
You may optionally enter a comparison operator (<b>&lt;</b>, <b>&lt;=</b>,
<b>&gt;</b>, <b>&gt;=</b>, <b>&lt;&gt;</b>
or <b>=</b>) at the beginning of each of your search values to specify how
the comparison should be done.
</p>

<?php echo CHtml::link('Advanced Search','#',array('class'=>'search-
button')); ?>
<div class="search-form" style="display:none">
<?php $this->renderPartial('_search',array(
      'model'=>$model,
)); ?>
</div><!-- search-form -->

<?php $this->widget('zii.widgets.grid.CGridView', array(
      'id'=>'report-history-grid',
      'dataProvider'=>$model->search(),
      'filter'=>$model,
      'columns'=>array(
            'REPORT_HISTORY_ID',
            'USER_ID',
            'REPORT_ID',
            array(
                  'class'=>'CButtonColumn',
            ),
      ),
)); ?>

<?php
/* @var $this ReportHistoryController */
/* @var $model ReportHistory */

$this->breadcrumbs=array(
      'Report Histories'=>array('index'),
      'Create',
);

$this->menu=array(
      array('label'=>'List ReportHistory', 'url'=>array('index'),
'visible'=>Yii::app()->user->checkAccess('systemAdmin')),
      array('label'=>'Manage ReportHistory', 'url'=>array('admin'),
'visible'=>Yii::app()->user->checkAccess('systemAdmin')),
);

$reportID = $_GET['id'];
?>
```

```
<h1>Assign Traffic Investigator to road accident # <?php echo $reportID
?></h1>

<?php echo $this->renderPartial('_form', array('model'=>$model)); ?>
<?php
/* @var $this ReportHistoryController */
/* @var $dataProvider CActiveDataProvider */

$this->breadcrumbs=array(
      'Report Histories',
);

$this->menu=array(
      //array('label'=>'Create ReportHistory', 'url'=>array('create')),
      //array('label'=>'Manage ReportHistory', 'url'=>array('admin')),
);
?>

<h1>Report Histories</h1>

<?php $this->widget('zii.widgets.CListView', array(
      'dataProvider'=>$dataProvider,
      'itemView'=>'_view',
)); ?>

<?php
/* @var $this ReportHistoryController */
/* @var $model ReportHistory */

$this->breadcrumbs=array(
      'Report Histories'=>array('index'),
      $model->REPORT_HISTORY_ID=>array('view','id'=>$model-
>REPORT_HISTORY_ID),
      'Update',
);

$this->menu=array(
      array('label'=>'List ReportHistory', 'url'=>array('index')),
      array('label'=>'Create ReportHistory', 'url'=>array('create')),
      array('label'=>'View ReportHistory', 'url'=>array('view',
'id'=>$model->REPORT_HISTORY_ID)),
      array('label'=>'Manage ReportHistory', 'url'=>array('admin')),
);
?>

<h1>Update ReportHistory <?php echo $model->REPORT_HISTORY_ID; ?></h1>

<?php echo $this->renderPartial('_form', array('model'=>$model)); ?>
<?php
/* @var $this ReportHistoryController */
/* @var $model ReportHistory */

$this->breadcrumbs=array(
      'Report Histories'=>array('index'),
```

```
	$model->REPORT_HISTORY_ID,
);

$this->menu=array(
	array('label'=>'List ReportHistory', 'url'=>array('index')),
	array('label'=>'Create ReportHistory', 'url'=>array('create')),
	array('label'=>'Update ReportHistory', 'url'=>array('update',
'id'=>$model->REPORT_HISTORY_ID)),
	array('label'=>'Delete ReportHistory', 'url'=>'#',
'linkOptions'=>array('submit'=>array('delete','id'=>$model-
>REPORT_HISTORY_ID),'confirm'=>'Are you sure you want to delete this
item?')),
	array('label'=>'Manage ReportHistory', 'url'=>array('admin')),
);
?>

<h1>View ReportHistory #<?php echo $model->REPORT_HISTORY_ID; ?></h1>

<?php $this->widget('zii.widgets.CDetailView', array(
	'data'=>$model,
	'attributes'=>array(
		'REPORT_HISTORY_ID',
		'USER_ID',
		'REPORT_ID',
	),
)); ?>

<?php
/* @var $this RoadClosedReportController */
/* @var $model RoadClosedReport */
/* @var $form CActiveForm */
?>

<div class="form">

<?php $form=$this->beginWidget('CActiveForm', array(
	'id'=>'road-closed-report-form',
	'enableAjaxValidation'=>false,
)); ?>

	<p class="note">Fields with <span class="required">*</span> are
required.</p>
	<p>Drag the marker in the map to get the latitude and longitude
values</p>

	<?php echo
CHtml::errorSummary(array($modelReport,$modelRoadClosedReport)); ?>

	<?php
	/* MAP */

	// init the map
	Yii::import('ext.jquery-gmap.*');
	$gmap = new EGmap3Widget();
```

```
$options = array(
      'scaleControl' => true,
      'streetViewControl' => false,
      'zoom' => 16,
      'center' => array(0, 0),
      'mapTypeId' => EGmap3MapTypeId::ROADMAP,
      'mapTypeControlOptions' => array(
            'style' => EGmap3MapTypeControlStyle::DROPDOWN_MENU,
            'position' => EGmap3ControlPosition::TOP_CENTER,
      ),
);

$gmap->setOptions($options);
$gmap->setSize(600, 400);

// create the marker
$marker = new EGmap3Marker(array(
      'title' => 'Draggable address marker 1',
      'draggable' => true,
));
$marker->latLng = array(14.4505957,120.9314332);
$marker->centerOnMap();

// set the marker to relay its position information a model
$marker->capturePosition(
       // the model object
       $modelLocationPoint,
       // model's latitude property name
       'LATITUDE',
       // model's longitude property name
       'LONGITUDE',
       // Options set :
       //   show the fields, defaults to hidden fields
       //   update the fields during the marker drag event
       array('visible','drag')
);
$gmap->add($marker);
$gmap->renderMap();
?>
<br/>
<div class="row">
      <?php echo '<b>Start Datetime yyyy-mm-dd hh:ss <br/></b>'; ?>
      <?php echo $form->textField($modelReport,'START_DATETIME'); ?>
      <?php echo $form->error($modelReport,'START_DATETIME'); ?>
</div>

<div class="row">
      <?php echo '<b>End Datetime yyyy-mm-dd hh:ss <br/></b>'; ?>
      <?php echo $form->textField($modelReport,'END_DATETIME'); ?>
      <?php echo $form->error($modelReport,'END_DATETIME'); ?>
</div>
```

```
	<div class="row buttons">
		<?php echo CHtml::submitButton($modelReport->isNewRecord ?
'Create' : 'Save'); ?>
	</div>

<?php $this->endWidget(); ?>

</div><!-- form -->
<?php
/* @var $this RoadClosedReportController */
/* @var $model RoadClosedReport */
/* @var $form CActiveForm */
?>

<div class="wide form">

<?php $form=$this->beginWidget('CActiveForm', array(
	'action'=>Yii::app()->createUrl($this->route),
	'method'=>'get',
)); ?>

	<div class="row">
		<?php echo $form->label($model,'REPORT_ID'); ?>
		<?php echo $form->textField($model,'REPORT_ID'); ?>
	</div>

	<div class="row buttons">
		<?php echo CHtml::submitButton('Search'); ?>
	</div>

<?php $this->endWidget(); ?>

</div><!-- search-form -->
<?php
/* @var $this RoadClosedReportController */
/* @var $data RoadClosedReport */

$reportModel = report::model()->findByPk($data->REPORT_ID);
$locationPointModel = locationPoint::model()->findByPk($data->REPORT_ID);

?>

<div class="view">

	<b><?php echo CHtml::encode('Report ID'); ?>:</b>
	<?php echo CHtml::link(CHtml::encode($data->REPORT_ID),
array('view', 'id'=>$data->REPORT_ID)); ?>
	<br />

	<b><?php echo CHtml::encode('Address'); ?>:</b>
	<?php echo CHtml::encode($locationPointModel->LATITUDE.',
'.$locationPointModel->LONGITUDE); ?>
	<br />
```

```
	<b><?php echo CHtml::encode('Time Created'); ?>:</b>
	<?php echo CHtml::encode($reportModel->DATETIME_CREATED); ?>
	<br />

	<b><?php echo CHtml::encode('Verified by System'); ?>:</b>
	<?php
	if($reportModel->IS_VERIFIED_BY_SYSTEM==1){
		echo CHtml::encode('True');
	}else{
		echo CHtml::encode('False');
	}
	?>
	<br />
</div>
<?php
/* @var $this RoadClosedReportController */
/* @var $model RoadClosedReport */

$this->breadcrumbs=array(
	'Road Closed Reports'=>array('index'),
	'Manage',
);

$this->menu=array(
	array('label'=>'List RoadClosedReport', 'url'=>array('index')),
	array('label'=>'Create RoadClosedReport', 'url'=>array('create')),
);

Yii::app()->clientScript->registerScript('search', "
$('.search-button').click(function(){
	$('.search-form').toggle();
	return false;
});
$('.search-form form').submit(function(){
	$('#road-closed-report-grid').yiiGridView('update', {
		data: $(this).serialize()
	});
	return false;
});
");
?>

<h1>Manage Road Closed Reports</h1>

<p>
You may optionally enter a comparison operator (<b>&lt;</b>, <b>&lt;=</b>,
<b>&gt;</b>, <b>&gt;=</b>, <b>&lt;&gt;</b>
or <b>=</b>) at the beginning of each of your search values to specify how
the comparison should be done.
</p>

<?php echo CHtml::link('Advanced Search','#',array('class'=>'search-
button')); ?>
<div class="search-form" style="display:none">
```

```
<?php $this->renderPartial('_search',array(
      'model'=>$model,
)); ?>
</div><!-- search-form -->

<?php $this->widget('zii.widgets.grid.CGridView', array(
      'id'=>'road-closed-report-grid',
      'dataProvider'=>$model->search(),
      'filter'=>$model,
      'columns'=>array(
            'REPORT_ID',
            array(
                  'class'=>'CButtonColumn',
            ),
      ),
)); ?>

<?php
/* @var $this RoadClosedReportController */
/* @var $model RoadClosedReport */

$this->breadcrumbs=array(
      'Road Closed Reports'=>array('index'),
      'Create',
);

$this->menu=array(
      array('label'=>'List RoadClosedReport', 'url'=>array('index')),
      array('label'=>'Manage RoadClosedReport', 'url'=>array('admin')),
);
?>

<h1>Create a Road Closed Report</h1>

<?php echo $this->renderPartial('_form', array(
                                    'modelReport'=>$modelReport,

      'modelRoadClosedReport'=>$modelRoadClosedReport,

      'modelLocationPoint'=>$modelLocationPoint,
                                 ));
?>
<?php
/* @var $this RoadClosedReportController */
/* @var $dataProvider CActiveDataProvider */

$this->breadcrumbs=array(
      'Road Closed Reports',
);

$this->menu=array(
      array('label'=>'Create RoadClosedReport', 'url'=>array('create'),
'visible'=>Yii::app()->user-
>checkAccess('trafficOperationsCenterOfficer')),
```

```
      array('label'=>'Manage RoadClosedReport', 'url'=>array('admin'),
'visible'=>Yii::app()->user-
>checkAccess('trafficOperationsCenterOfficer')),
);
?>

<h1>Closed Road Reports</h1>

<?php
/**
 * EGmap3 Yii extension example view file.
 *
 * You can copy this file or its contents into your Yii
 * application for testing.
 *
 */
Yii::import('ext.jquery-gmap.*');

$gmap = new EGmap3Widget();
$gmap->setSize(600, 400);

// base options
$options = array(
      'scaleControl' => true,
      'streetViewControl' => false,
      'zoom' => 16,
      'center' => array(0, 0),
      'mapTypeId' => EGmap3MapTypeId::ROADMAP,
      'mapTypeControlOptions' => array(
            'style' => EGmap3MapTypeControlStyle::DROPDOWN_MENU,
            'position' => EGmap3ControlPosition::TOP_CENTER,
      ),
);
$gmap->setOptions($options);

$length = count($roadClosedReportArray);
for($i =0; $i < $length; $i++){
      $roadClosedReport = $roadClosedReportArray[$i];
      $reportModel = Report::model()->findByPk($roadClosedReport-
>REPORT_ID);
      $locationPointModel = LocationPoint::model()->findByPk($reportModel-
>LOCATION_POINT_ID);

      // marker with custom icon
      $marker = new EGmap3Marker(array(
                  'title' => 'hello',
                  'icon' => array(
                        'url' => 'http://google-maps-
icons.googlecode.com/files/closedroad.png',//'http://google-maps-
icons.googlecode.com/files/accident2.png'
                        'anchor' => array('x' => 1, 'y' => 36),
                  //'anchor' => new EGmap3Point(5,5),
                  )
            ));
```

```
      // set marker position by address
      $marker->latLng = array($locationPointModel-
>LATITUDE,$locationPointModel->LONGITUDE);

      // data associated with the marker
      $marker->data = $reportModel->REPORT_ID;

      // add a Javascript event on marker click
      $js = "function(marker, event, data){
                  var map = $(this).gmap3('get'),
                  infowindow = $(this).gmap3({action:'get',
name:'infowindow'});
                  if (infowindow){
                        infowindow.open(map, marker);
                        infowindow.setContent(data);
                  } else {
                        $(this).gmap3({action:'addinfowindow',
anchor:marker, options:{content: data}});
                  }
      }";
      $marker->addEvent('click', $js);

      // center the map on the marker
      $marker->centerOnMap();

      $gmap->add($marker);
}

echo "<center>";
$gmap->renderMap();
echo "</center>";
?>

<?php $this->widget('zii.widgets.CListView', array(
      'dataProvider'=>$dataProvider,
      'itemView'=>'_view',
      'sortableAttributes'=>array(
        'REPORT_ID'=>'TIME CREATED',
           'IS_VERIFIED_BY_SYSTEM'=>'System Verified',
    ),
)); ?>

<?php
/* @var $this RoadClosedReportController */
/* @var $model RoadClosedReport */

$this->breadcrumbs=array(
      'Road Closed Reports'=>array('index'),
      $model->REPORT_ID=>array('view','id'=>$model->REPORT_ID),
      'Update',
);

$this->menu=array(
```

```
	array('label'=>'List RoadClosedReport', 'url'=>array('index')),
	array('label'=>'Create RoadClosedReport', 'url'=>array('create')),
	array('label'=>'View RoadClosedReport', 'url'=>array('view',
'id'=>$model->REPORT_ID)),
	array('label'=>'Manage RoadClosedReport', 'url'=>array('admin')),
);
?>

<h1>Update RoadClosedReport <?php echo $model->REPORT_ID; ?></h1>

<?php echo $this->renderPartial('_form',
array('modelRoadClosedReport'=>$model,'modelReport'=>$modelReport,'modelLo
cationPoint'=>$modelLocationPoint)); ?>
<?php
/**
 * EGmap3 Yii extension example view file.
 *
 * You can copy this file or its contents into your Yii
 * application for testing.
 *
 */
Yii::import('ext.jquery-gmap.*');

$gmap = new EGmap3Widget();
$gmap->setSize('100%', 240);

// base options
$options = array(
	'scaleControl' => true,
	'streetViewControl' => false,
	'zoom' => 16,
	'center' => array(0, 0),
	'mapTypeId' => EGmap3MapTypeId::ROADMAP,
	'mapTypeControlOptions' => array(
		'style' => EGmap3MapTypeControlStyle::DROPDOWN_MENU,
		'position' => EGmap3ControlPosition::TOP_CENTER,
	),
);
$gmap->setOptions($options);

// marker with custom icon
$marker = new EGmap3Marker(array(
			'title' => 'hello',
			'icon' => array(
				'url' => 'http://google-maps-
icons.googlecode.com/files/closedroad.png',//'http://google-maps-
icons.googlecode.com/files/accident2.png'
				'anchor' => array('x' => 1, 'y' => 36),
			//'anchor' => new EGmap3Point(5,5),
			)
		));

// set marker position by address
$lat = $locationPoint->LATITUDE;
```

```
$lng = $locationPoint->LONGITUDE;
$marker->latLng = array($lat,$lng);

// data associated with the marker
$marker->data = 'test data !';

// add a Javascript event on marker click
$js = "function(marker, event, data){
        var map = $(this).gmap3('get'),
        infowindow = $(this).gmap3({action:'get', name:'infowindow'});
        if (infowindow){
            infowindow.open(map, marker);
            infowindow.setContent(data);
        } else {
            $(this).gmap3({action:'addinfowindow', anchor:marker,
options:{content: data}});
        }
}";
$marker->addEvent('click', $js);

// center the map on the marker
$marker->centerOnMap();

$gmap->add($marker);
?>

<?php
/* @var $this RoadClosedReportController */
/* @var $model RoadClosedReport */

$this->breadcrumbs=array(
      'Road Closed Reports'=>array('index'),
      $model->REPORT_ID,
);

$this->menu=array(
      array('label'=>'Verify Report',
'url'=>array('VerifyReport','id'=>$model->REPORT_ID),
'visible'=>Yii::app()->user-
>checkAccess('trafficOperationsCenterOfficer')),
      array('label'=>'List Road Closed Report', 'url'=>array('index'),
'visible'=>Yii::app()->user-
>checkAccess('trafficOperationsCenterOfficer')),
      //array('label'=>'Create Report', 'url'=>array('create')),
      array('label'=>'Update Report', 'url'=>array('update', 'id'=>$model-
>REPORT_ID), 'visible'=>Yii::app()->user-
>checkAccess('trafficOperationsCenterOfficer')),
      array('label'=>'Delete Report', 'url'=>'#',
'linkOptions'=>array('submit'=>array('delete','id'=>$model-
>REPORT_ID),'confirm'=>'Are you sure you want to delete this item?'),
'visible'=>Yii::app()->user-
>checkAccess('trafficOperationsCenterOfficer')),
      //array('label'=>'Manage Road Closed Report',
'url'=>array('admin')),
```

```
);
?>

<h1>View RoadClosedReport #<?php echo $model->REPORT_ID; ?></h1>

<?php


$reportModel = report::model()->findByPk($model->REPORT_ID);
$isVerifiedBySystem;
if($reportModel->IS_VERIFIED_BY_SYSTEM==1){
      $isVerifiedBySystem = 'True';
}else{
      $isVerifiedBySystem = 'False';
}


$this->widget('zii.widgets.CDetailView', array(
      'data'=>$model,
      'attributes'=>array(
            array(
            'name'=>CHtml::encode('Report ID'),
            'value'=>CHtml::encode($model->REPORT_ID)
            ),
            array(
            'name'=>CHtml::encode('START DATETIME'),
            'value'=>CHtml::encode($reportModel->START_DATETIME)
            ),
            array(
            'name'=>CHtml::encode('END DATETIME'),
            'value'=>CHtml::encode($reportModel->END_DATETIME)
            ),
            array(
            'name'=>CHtml::encode('Verified by System'),
            'value'=>CHtml::encode($isVerifiedBySystem)
            ),
            /*array(
            'name'=>CHtml::encode('Address'),
            'value'=>CHtml::encode('')
            ),
            array(
            'name'=>CHtml::encode('LATITUDE'),
            'value'=>CHtml::encode($locationPoint->LATITUDE)
            ),
            array(
            'name'=>CHtml::encode('LONGITUDE'),
            'value'=>CHtml::encode($locationPoint->LONGITUDE)
            ),*/
      ),
)); ?>

<?php
$gmap->renderMap();
?>
```

```
<?php
/* @var $this SiteController */
/* @var $model ContactForm */
/* @var $form CActiveForm */

$this->pageTitle=Yii::app()->name . ' - Contact Us';
$this->breadcrumbs=array(
      'Contact',
);
?>

<h1>Contact Us</h1>

<?php if(Yii::app()->user->hasFlash('contact')): ?>

<div class="flash-success">
      <?php echo Yii::app()->user->getFlash('contact'); ?>
</div>

<?php else: ?>

<p>
If you have business inquiries or other questions, please fill out the
following form to contact us. Thank you.
</p>

<div class="form">

<?php $form=$this->beginWidget('CActiveForm', array(
      'id'=>'contact-form',
      'enableClientValidation'=>true,
      'clientOptions'=>array(
            'validateOnSubmit'=>true,
      ),
)); ?>

      <p class="note">Fields with <span class="required">*</span> are
required.</p>

      <?php echo $form->errorSummary($model); ?>

      <div class="row">
            <?php echo $form->labelEx($model,'name'); ?>
            <?php echo $form->textField($model,'name'); ?>
            <?php echo $form->error($model,'name'); ?>
      </div>

      <div class="row">
            <?php echo $form->labelEx($model,'email'); ?>
            <?php echo $form->textField($model,'email'); ?>
            <?php echo $form->error($model,'email'); ?>
      </div>

      <div class="row">
```

```
            <?php echo $form->labelEx($model,'subject'); ?>
            <?php echo $form-
>textField($model,'subject',array('size'=>60,'maxlength'=>128)); ?>
            <?php echo $form->error($model,'subject'); ?>
      </div>

      <div class="row">
            <?php echo $form->labelEx($model,'body'); ?>
            <?php echo $form->textArea($model,'body',array('rows'=>6,
'cols'=>50)); ?>
            <?php echo $form->error($model,'body'); ?>
      </div>

      <?php if(CCaptcha::checkRequirements()): ?>
      <div class="row">
            <?php echo $form->labelEx($model,'verifyCode'); ?>
            <div>
            <?php $this->widget('CCaptcha'); ?>
            <?php echo $form->textField($model,'verifyCode'); ?>
            </div>
            <div class="hint">Please enter the letters as they are shown
in the image above.
            <br/>Letters are not case-sensitive.</div>
            <?php echo $form->error($model,'verifyCode'); ?>
      </div>
      <?php endif; ?>

      <div class="row buttons">
            <?php echo CHtml::submitButton('Submit'); ?>
      </div>

<?php $this->endWidget(); ?>

</div><!-- form -->

<?php endif; ?>
<?php
/* @var $this SiteController */
/* @var $error array */

$this->pageTitle=Yii::app()->name . ' - Error';
$this->breadcrumbs=array(
      'Error',
);
?>

<h2>Error <?php echo $code; ?></h2>

<div class="error">
<?php echo CHtml::encode($message); ?>
</div>
<?php
/* @var $this SiteController */
```

```
$this->pageTitle=Yii::app()->name;
?>

<h1>Welcome to <i><?php echo CHtml::encode(Yii::app()->name); ?></i></h1>
<?php
/* @var $this SiteController */
/* @var $model LoginForm */
/* @var $form CActiveForm  */

$this->pageTitle=Yii::app()->name . ' - Login';
$this->breadcrumbs=array(
      'Login',
);
?>

<h1>Login</h1>

<p>Please fill out the following form with your login credentials:</p>

<div class="form">
<?php $form=$this->beginWidget('CActiveForm', array(
      'id'=>'login-form',
      'enableClientValidation'=>true,
      'clientOptions'=>array(
            'validateOnSubmit'=>true,
      ),
)); ?>

      <p class="note">Fields with <span class="required">*</span> are
required.</p>

      <div class="row">
            <?php echo $form->labelEx($model,'username'); ?>
            <?php echo $form->textField($model,'username'); ?>
            <?php echo $form->error($model,'username'); ?>
      </div>

      <div class="row">
            <?php echo $form->labelEx($model,'password'); ?>
            <?php echo $form->passwordField($model,'password'); ?>
            <?php echo $form->error($model,'password'); ?>
      </div>

      <div class="row rememberMe">
            <?php echo $form->checkBox($model,'rememberMe'); ?>
            <?php echo $form->label($model,'rememberMe'); ?>
            <?php echo $form->error($model,'rememberMe'); ?>
      </div>

      <div class="row buttons">
            <?php echo CHtml::submitButton('Login'); ?>
      </div>

<?php $this->endWidget(); ?>
```

```
</div><!-- form -->

<?php
/* @var $this SystemAdministratorController */
/* @var $model SystemAdministrator */
/* @var $form CActiveForm */
?>

<div class="form">

<?php $form=$this->beginWidget('CActiveForm', array(
      'id'=>'system-administrator-form',
      'enableAjaxValidation'=>false,
)); ?>

      <p class="note">Fields with <span class="required">*</span> are
required.</p>

      <?php echo
CHtml::errorSummary(array($modelUsers,$modelAccountableUser,$modelSystemAd
ministrator)); ?>

      <div class="row">
            <?php echo $form->labelEx($modelAccountableUser,'USERNAME');
?>
            <?php echo $form-
>textField($modelAccountableUser,'USERNAME',array('rows'=>6, 'cols'=>50));
?>
            <?php echo $form->error($modelAccountableUser,'USERNAME'); ?>
      </div>

      <div class="row">
            <?php echo $form->labelEx($modelAccountableUser,'PASSWORD');
?>
            <?php echo $form-
>passwordField($modelAccountableUser,'PASSWORD',array('rows'=>6,
'cols'=>50)); ?>
            <?php echo $form->error($modelAccountableUser,'PASSWORD'); ?>
      </div>

      <div class="row">
            <?php echo $form->labelEx($modelAccountableUser,'PASSWORD
REPEAT'); ?>
            <?php echo $form-
>passwordField($modelAccountableUser,'PASSWORD_repeat',array('rows'=>6,
'cols'=>50)); ?>
            <?php echo $form-
>error($modelAccountableUser,'PASSWORD_repeat'); ?>
      </div>

      <div class="row">
            <?php echo $form->labelEx($modelAccountableUser,'FIRST_NAME');
?>
```

```
            <?php echo $form-
>textField($modelAccountableUser,'FIRST_NAME',array('rows'=>6,
'cols'=>50)); ?>
            <?php echo $form->error($modelAccountableUser,'FIRST_NAME');
?>
      </div>

      <div class="row">
            <?php echo $form->labelEx($modelAccountableUser,'LAST_NAME');
?>
            <?php echo $form-
>textField($modelAccountableUser,'LAST_NAME',array('rows'=>6,
'cols'=>50)); ?>
            <?php echo $form->error($modelAccountableUser,'LAST_NAME'); ?>
      </div>

      <div class="row">
            <?php echo $form-
>labelEx($modelAccountableUser,'USER_INITIAL'); ?>
            <?php echo $form-
>textField($modelAccountableUser,'USER_INITIAL',array('rows'=>6,
'cols'=>50)); ?>
            <?php echo $form->error($modelAccountableUser,'USER_INITIAL');
?>
      </div>

      <div class="row">
            <?php echo $form->labelEx($modelAccountableUser,'BIRTHDAY');
?>
            <?php echo $form->DateField($modelAccountableUser,'BIRTHDAY');
?>
            <?php echo $form->error($modelAccountableUser,'BIRTHDAY'); ?>
      </div>

      <div class="row buttons">
            <?php echo CHtml::submitButton($modelUsers->isNewRecord ?
'Create' : 'Save'); ?>
      </div>
<?php $this->endWidget(); ?>

</div><!-- form -->
<?php
/* @var $this SystemAdministratorController */
/* @var $model SystemAdministrator */
/* @var $form CActiveForm */
?>

<div class="wide form">

<?php $form=$this->beginWidget('CActiveForm', array(
      'action'=>Yii::app()->createUrl($this->route),
      'method'=>'get',
)); ?>
```

```
	<div class="row">
		<?php echo $form->label($model,'USER_ID'); ?>
		<?php echo $form->textField($model,'USER_ID'); ?>
	</div>

	<div class="row buttons">
		<?php echo CHtml::submitButton('Search'); ?>
	</div>

<?php $this->endWidget(); ?>

</div><!-- search-form -->
<?php
/* @var $this SystemAdministratorController */
/* @var $data SystemAdministrator */
?>

<div class="view">

	<b><?php echo CHtml::encode('User Id'); ?>:</b>
	<?php echo CHtml::link(CHtml::encode($data->USER_ID), array('view',
'id'=>$data->USER_ID)); ?>
	<br />

	<b><?php echo CHtml::encode('Name'); ?>:</b>
	<?php echo CHtml::encode(CHtml::encode($this->getName($data-
>USER_ID))); ?>
	<br />

</div>
<?php
/* @var $this SystemAdministratorController */
/* @var $model SystemAdministrator */

$this->breadcrumbs=array(
	'System Administrators'=>array('index'),
	'Manage',
);

$this->menu=array(
	array('label'=>'List SystemAdministrator', 'url'=>array('index')),
	array('label'=>'Create SystemAdministrator',
'url'=>array('create')),
);

Yii::app()->clientScript->registerScript('search', "
$('.search-button').click(function(){
	$('.search-form').toggle();
	return false;
});
$('.search-form form').submit(function(){
	$('#system-administrator-grid').yiiGridView('update', {
		data: $(this).serialize()
	});
```

```
      return false;
});
");
?>

<h1>Manage System Administrators</h1>

<p>
You may optionally enter a comparison operator (<b>&lt;</b>, <b>&lt;=</b>,
<b>&gt;</b>, <b>&gt;=</b>, <b>&lt;&gt;</b>
or <b>=</b>) at the beginning of each of your search values to specify how
the comparison should be done.
</p>

<?php echo CHtml::link('Advanced Search','#',array('class'=>'search-
button')); ?>
<div class="search-form" style="display:none">
<?php $this->renderPartial('_search',array(
      'model'=>$model,
)); ?>
</div><!-- search-form -->

<?php $this->widget('zii.widgets.grid.CGridView', array(
      'id'=>'system-administrator-grid',
      'dataProvider'=>$model->search(),
      'filter'=>$model,
      'columns'=>array(
            'USER_ID',
            array(
                  'class'=>'CButtonColumn',
            ),
      ),
)); ?>

<?php
/* @var $this SystemAdministratorController */
/* @var $model SystemAdministrator */

$this->breadcrumbs=array(
      'System Administrators'=>array('index'),
      'Create',
);

$this->menu=array(
      array('label'=>'List SystemAdministrator', 'url'=>array('index')),
      array('label'=>'Manage SystemAdministrator', 'url'=>array('admin')),
);
?>

<h1>Create SystemAdministrator</h1>

<?php echo $this->renderPartial('_form',
array('modelSystemAdministrator'=>$modelSystemAdministrator,
```

```
'modelAccountableUser'=>$modelAccountableUser,

'modelUsers'=>$modelUsers,)
                                                    );
?>
<?php
/* @var $this SystemAdministratorController */
/* @var $dataProvider CActiveDataProvider */

$this->breadcrumbs=array(
      'System Administrators',
);

$this->menu=array(
      array('label'=>'Create SystemAdministrator',
'url'=>array('create')),
      array('label'=>'Manage SystemAdministrator', 'url'=>array('admin')),
);
?>

<h1>System Administrators</h1>

<?php $this->widget('zii.widgets.CListView', array(
      'dataProvider'=>$dataProvider,
      'itemView'=>'_view',
)); ?>

<?php
/* @var $this SystemAdministratorController */
/* @var $model SystemAdministrator */

$this->breadcrumbs=array(
      'System Administrators'=>array('index'),
      $modelUsers->USER_ID=>array('view','id'=>$modelUsers->USER_ID),
      'Update',
);

$this->menu=array(
      array('label'=>'List SystemAdministrator', 'url'=>array('index')),
      //array('label'=>'Create SystemAdministrator',
'url'=>array('create')),
      //array('label'=>'View SystemAdministrator', 'url'=>array('view',
'id'=>$modelUsers->USER_ID)),
      //array('label'=>'Manage SystemAdministrator',
'url'=>array('admin')),
);
?>

<h1>Update SystemAdministrator <?php echo $modelUsers->USER_ID; ?></h1>

<?php echo $this->renderPartial('_form',
array('modelSystemAdministrator'=>$modelSystemAdministrator,
```

```
'modelAccountableUser'=>$modelAccountableUser,

'modelUsers'=>$modelUsers,)
                                                    );
?>
<?php
/* @var $this SystemAdministratorController */
/* @var $model SystemAdministrator */

$this->breadcrumbs=array(
      'System Administrators'=>array('index'),
      $model->USER_ID=>array('view','id'=>$model->USER_ID),
      'Update',
);

$this->menu=array(
      array('label'=>'List SystemAdministrator', 'url'=>array('index')),
      array('label'=>'Create SystemAdministrator',
'url'=>array('create')),
      array('label'=>'View SystemAdministrator', 'url'=>array('view',
'id'=>$model->USER_ID)),
      array('label'=>'Manage SystemAdministrator', 'url'=>array('admin')),
);
?>

<h1>Update Timer</h1>
<?php
/* @var $this SystemAdministratorController */
/* @var $model SystemAdministrator */
$modelAccountableUser = AccountableUser::model()->findByPk($model-
>USER_ID);
$name = $modelAccountableUser->FIRST_NAME.' '.$modelAccountableUser-
>USER_INITIAL.'. '.$modelAccountableUser->LAST_NAME;

$this->breadcrumbs=array(
      'System Administrators'=>array('index'),
      $model->USER_ID,
);

$this->menu=array(
      array('label'=>'List SystemAdministrator', 'url'=>array('index')),
      //array('label'=>'Create SystemAdministrator',
'url'=>array('create')),
      array('label'=>'Update SystemAdministrator', 'url'=>array('update',
'id'=>$model->USER_ID)),
      //array('label'=>'Delete SystemAdministrator', 'url'=>'#',
'linkOptions'=>array('submit'=>array('delete','id'=>$model-
>USER_ID),'confirm'=>'Are you sure you want to delete this item?')),
      //array('label'=>'Manage SystemAdministrator',
'url'=>array('admin')),
);
```

```
?>

<h1>View SystemAdministrator #<?php echo $model->USER_ID; ?></h1>

<?php $this->widget('zii.widgets.CDetailView', array(
      'data'=>$model,
      'attributes'=>array(
            'USER_ID',
            array(
                  'name'=>'NAME',
                  'value'=>$name
            ),
      ),
)); ?>

<?php
/* @var $this TrafficInvestigatorController */
/* @var $model TrafficInvestigator */
/* @var $form CActiveForm */
?>

<div class="form">

<?php $form=$this->beginWidget('CActiveForm', array(
      'id'=>'traffic-investigator-form',
      'enableAjaxValidation'=>false,
)); ?>

      <p class="note">Fields with <span class="required">*</span> are
required.</p>

      <?php echo
CHtml::errorSummary(array($modelUsers,$modelAccountableUser,$modelTrafficI
nvestigator)); ?>

      <div class="row">
            <?php echo $form->labelEx($modelAccountableUser,'USERNAME');
?>
            <?php echo $form-
>textField($modelAccountableUser,'USERNAME',array('rows'=>6, 'cols'=>50));
?>
            <?php echo $form->error($modelAccountableUser,'USERNAME'); ?>
      </div>

      <div class="row">
            <?php echo $form->labelEx($modelAccountableUser,'PASSWORD');
?>
            <?php echo $form-
>passwordField($modelAccountableUser,'PASSWORD',array('rows'=>6,
'cols'=>50)); ?>
            <?php echo $form->error($modelAccountableUser,'PASSWORD'); ?>
      </div>

      <div class="row">
```

```
            <?php echo $form->labelEx($modelAccountableUser,'PASSWORD
REPEAT'); ?>
            <?php echo $form-
>passwordField($modelAccountableUser,'PASSWORD_repeat',array('rows'=>6,
'cols'=>50)); ?>
            <?php echo $form-
>error($modelAccountableUser,'PASSWORD_repeat'); ?>
      </div>

      <div class="row">
            <?php echo $form->labelEx($modelAccountableUser,'FIRST_NAME');
?>
            <?php echo $form-
>textField($modelAccountableUser,'FIRST_NAME',array('rows'=>6,
'cols'=>50)); ?>
            <?php echo $form->error($modelAccountableUser,'FIRST_NAME');
?>
      </div>

      <div class="row">
            <?php echo $form->labelEx($modelAccountableUser,'LAST_NAME');
?>
            <?php echo $form-
>textField($modelAccountableUser,'LAST_NAME',array('rows'=>6,
'cols'=>50)); ?>
            <?php echo $form->error($modelAccountableUser,'LAST_NAME'); ?>
      </div>

      <div class="row">
            <?php echo $form-
>labelEx($modelAccountableUser,'USER_INITIAL'); ?>
            <?php echo $form-
>textField($modelAccountableUser,'USER_INITIAL',array('rows'=>6,
'cols'=>50)); ?>
            <?php echo $form->error($modelAccountableUser,'USER_INITIAL');
?>
      </div>

      <div class="row">
            <?php echo $form->labelEx($modelAccountableUser,'BIRTHDAY');
?>
            <?php echo $form->DateField($modelAccountableUser,'BIRTHDAY');
?>
            <?php echo $form->error($modelAccountableUser,'BIRTHDAY'); ?>
      </div>

      <div class="row">
      <?php
            echo $form->labelEx($modelTrafficInvestigator,'JURISDICTION');
            echo $form-
>dropDownList($modelTrafficInvestigator,'JURISDICTION_ID', $this-
>getJurisdictions());
      ?>
      </div>
```

```
	<div class="row buttons">
		<?php echo CHtml::submitButton($modelUsers->isNewRecord ?
'Create' : 'Save'); ?>
	</div>

<?php $this->endWidget(); ?>

</div><!-- form -->
<?php
/* @var $this TrafficInvestigatorController */
/* @var $model TrafficInvestigator */
/* @var $form CActiveForm */
?>

<div class="wide form">

<?php $form=$this->beginWidget('CActiveForm', array(
	'action'=>Yii::app()->createUrl($this->route),
	'method'=>'get',
)); ?>

	<div class="row">
		<?php echo $form->label($model,'USER_ID'); ?>
		<?php echo $form->textField($model,'USER_ID'); ?>
	</div>

	<div class="row">
		<?php echo $form-
>label($model,'TRAFFIC_INVESTIGATOR_STATUS_ID'); ?>
		<?php echo $form-
>textField($model,'TRAFFIC_INVESTIGATOR_STATUS_ID'); ?>
	</div>

	<div class="row buttons">
		<?php echo CHtml::submitButton('Search'); ?>
	</div>

<?php $this->endWidget(); ?>

</div><!-- search-form -->
<?php
/* @var $this TrafficInvestigatorController */
/* @var $data TrafficInvestigator */
?>

<div class="view">

	<b><?php echo CHtml::encode($data->getAttributeLabel('USER_ID'));
?>:</b>
	<?php echo CHtml::link(CHtml::encode($data->USER_ID), array('view',
'id'=>$data->USER_ID)); ?>
	<br />
```

```
	<b><?php echo CHtml::encode('Name'); ?>:</b>
	<?php echo CHtml::encode($this->getName($data->USER_ID)); ?>
	<br />

	<b><?php echo CHtml::encode('Status'); ?>:</b>
	<?php echo CHtml::encode($this->getStatusName($data-
>TRAFFIC_INVESTIGATOR_STATUS_ID)); ?>
	<br />


</div>
<?php
/* @var $this TrafficInvestigatorController */
/* @var $model TrafficInvestigator */

$this->breadcrumbs=array(
	'Traffic Investigators'=>array('index'),
	'Manage',
);

$this->menu=array(
	array('label'=>'List TrafficInvestigator', 'url'=>array('index')),
	array('label'=>'Create TrafficInvestigator',
'url'=>array('create')),
);

Yii::app()->clientScript->registerScript('search', "
$('.search-button').click(function(){
	$('.search-form').toggle();
	return false;
});
$('.search-form form').submit(function(){
	$('#traffic-investigator-grid').yiiGridView('update', {
		data: $(this).serialize()
	});
	return false;
});
");
?>

<h1>Manage Traffic Investigators</h1>

<p>
You may optionally enter a comparison operator (<b>&lt;</b>, <b>&lt;=</b>,
<b>&gt;</b>, <b>&gt;=</b>, <b>&lt;&gt;</b>
or <b>=</b>) at the beginning of each of your search values to specify how
the comparison should be done.
</p>

<?php echo CHtml::link('Advanced Search','#',array('class'=>'search-
button')); ?>
<div class="search-form" style="display:none">
<?php $this->renderPartial('_search',array(
	'model'=>$model,
```

```
)); ?>
</div><!-- search-form -->

<?php $this->widget('zii.widgets.grid.CGridView', array(
      'id'=>'traffic-investigator-grid',
      'dataProvider'=>$model->search(),
      'filter'=>$model,
      'columns'=>array(
            'USER_ID',
            'TRAFFIC_INVESTIGATOR_STATUS_ID',
            array(
                  'class'=>'CButtonColumn',
            ),
      ),
)); ?>

<?php
/* @var $this TrafficInvestigatorController */
/* @var $model TrafficInvestigator */

$this->breadcrumbs=array(
      'Traffic Investigators'=>array('index'),
      'Create',
);

$this->menu=array(
      array('label'=>'List TrafficInvestigator', 'url'=>array('index')),
      //array('label'=>'Manage TrafficInvestigator',
'url'=>array('admin')),
);
?>

<h1>Create TrafficInvestigator</h1>

<?php echo $this->renderPartial('_form',
array('modelTrafficInvestigator'=>$modelTrafficInvestigator,

'modelAccountableUser'=>$modelAccountableUser,

'modelUsers'=>$modelUsers,)
                                                );
?>
<?php
/* @var $this TrafficInvestigatorController */
/* @var $dataProvider CActiveDataProvider */

$this->breadcrumbs=array(
      'Traffic Investigators',
);

$this->menu=array(
      array('label'=>'Create TrafficInvestigator', 'url'=>array('create'),
'visible'=>Yii::app()->user->checkAccess('systemAdmin')),
```

```
      //array('label'=>'Manage TrafficInvestigator',
'url'=>array('admin'), 'visible'=>Yii::app()->user-
>checkAccess('systemAdmin')),
);
?>

<h1>Traffic Investigators</h1>

<?php
/**
 * EGmap3 Yii extension example view file.
 *
 * You can copy this file or its contents into your Yii
 * application for testing.
 *
 */
Yii::import('ext.jquery-gmap.*');

$gmap = new EGmap3Widget();
$gmap->setSize(600, 400);

// base options
$options = array(
      'scaleControl' => true,
      'streetViewControl' => false,
      'zoom' => 16,
      'center' => array(0, 0),
      'mapTypeId' => EGmap3MapTypeId::ROADMAP,
      'mapTypeControlOptions' => array(
            'style' => EGmap3MapTypeControlStyle::DROPDOWN_MENU,
            'position' => EGmap3ControlPosition::TOP_CENTER,
      ),
);
$gmap->setOptions($options);

$length = count($trafficInvestigators);
for($i =0; $i < $length; $i++){
      $trafficInvestigatorModel = $trafficInvestigators[$i];


      if($trafficInvestigatorModel->IS_ACTIVE==1){
            // marker with custom icon
            $marker = new EGmap3Marker(array(
                        'title' => $this-
>getName($trafficInvestigatorModel->USER_ID),
                        'icon' => array(
                              'url' => 'http://google-maps-
icons.googlecode.com/files/police2.png',//'http://google-maps-
icons.googlecode.com/files/accident2.png'
                              'anchor' => array('x' => 1, 'y' => 36),
                        //'anchor' => new EGmap3Point(5,5),
                        )
                  ));
```

```
            // set marker position by address
            $marker->latLng = array($trafficInvestigatorModel-
>CURRENT_LAT,$trafficInvestigatorModel->CURRENT_LNG);

            // data associated with the marker
            $marker->data = $trafficInvestigatorModel->USER_ID;

            // add a Javascript event on marker click
            $js = "function(marker, event, data){
                        var map = $(this).gmap3('get'),
                        infowindow = $(this).gmap3({action:'get',
name:'infowindow'});
                        if (infowindow){
                              infowindow.open(map, marker);
                              infowindow.setContent(data);
                        } else {
                              $(this).gmap3({action:'addinfowindow',
anchor:marker, options:{content: data}});
                        }
            }";
            $marker->addEvent('click', $js);

            // center the map on the marker
            $marker->centerOnMap();

            $gmap->add($marker);
      }



}

echo "<center>";
$gmap->renderMap();
echo "</center>";
?>


<?php $this->widget('zii.widgets.CListView', array(
      'dataProvider'=>$dataProvider,
      'itemView'=>'_view',
)); ?>

<?php
/* @var $this TrafficInvestigatorController */
/* @var $model TrafficInvestigator */

$this->breadcrumbs=array(
      'Traffic Investigators'=>array('index'),
      $modelUsers->USER_ID=>array('view','id'=>$modelUsers->USER_ID),
      'Update',
);

$this->menu=array(
```

```
      array('label'=>'List TrafficInvestigator', 'url'=>array('index')),
      //array('label'=>'Create TrafficInvestigator',
'url'=>array('create')),
      array('label'=>'View TrafficInvestigator', 'url'=>array('view',
'id'=>$modelUsers->USER_ID)),
      //array('label'=>'Manage TrafficInvestigator',
'url'=>array('admin')),
);
?>

<h1>Update TrafficInvestigator <?php echo $modelUsers->USER_ID; ?></h1>

<?php echo $this->renderPartial('_form',
array('modelTrafficInvestigator'=>$modelTrafficInvestigator,

'modelAccountableUser'=>$modelAccountableUser,

'modelUsers'=>$modelUsers,)
                                                    );
?>
<?php
/* @var $this TrafficInvestigatorController */
/* @var $model TrafficInvestigator */
$modelAccountableUser = AccountableUser::model()->findByPk($model-
>USER_ID);
$modelJurisdiction = Jurisdiction::model()->findByPk($model-
>JURISDICTION_ID);
$modelStatus = TrafficInvestigatorStatus::model()->findByPk($model-
>TRAFFIC_INVESTIGATOR_STATUS_ID);
$name = $modelAccountableUser->FIRST_NAME.' '.$modelAccountableUser-
>USER_INITIAL.'. '.$modelAccountableUser->LAST_NAME;
$jurisdiction = $modelJurisdiction->NAME;
$status = $modelStatus->NAME;
$this->breadcrumbs=array(
      'Traffic Investigators'=>array('index'),
      $model->USER_ID,
);

$this->menu=array(
      array('label'=>'List TrafficInvestigator', 'url'=>array('index')),
      //array('label'=>'Create TrafficInvestigator',
'url'=>array('create'), 'visible'=>Yii::app()->user-
>checkAccess('systemAdmin')),
      array('label'=>'Update TrafficInvestigator', 'url'=>array('update',
'id'=>$model->USER_ID), 'visible'=>Yii::app()->user-
>checkAccess('systemAdmin')),
      //array('label'=>'Delete TrafficInvestigator', 'url'=>'#',
'linkOptions'=>array('submit'=>array('delete','id'=>$model-
>USER_ID),'confirm'=>'Are you sure you want to delete this item?'),
'visible'=>Yii::app()->user->checkAccess('systemAdmin')),
      //array('label'=>'Manage TrafficInvestigator',
'url'=>array('admin'), 'visible'=>Yii::app()->user-
>checkAccess('systemAdmin')),
);
```

```
?>

<h1>View TrafficInvestigator #<?php echo $model->USER_ID; ?></h1>

<?php $this->widget('zii.widgets.CDetailView', array(
      'data'=>$model,
      'attributes'=>array(
            'USER_ID',
            array(
                  'name'=>'NAME',
                  'value'=>$name
            ),
            array(
                  'name'=>'JURISDICTION',
                  'value'=>$jurisdiction
            ),
            array(
                  'name'=>'STATUS',
                  'value'=>$status
            ),
      ),
)); ?>

<?php
/* @var $this TrafficOperationsCenterOfficerController */
/* @var $model TrafficOperationsCenterOfficer */
/* @var $form CActiveForm */
?>

<div class="form">

<?php $form=$this->beginWidget('CActiveForm', array(
      'id'=>'traffic-operations-center-officer-form',
      'enableAjaxValidation'=>false,
)); ?>

      <p class="note">Fields with <span class="required">*</span> are
required.</p>

      <?php echo
CHtml::errorSummary(array($modelUsers,$modelAccountableUser,$modelTrafficO
perationsCenterOfficer)); ?>

      <div class="row">
            <?php echo $form->labelEx($modelAccountableUser,'USERNAME');
?>
            <?php echo $form-
>textField($modelAccountableUser,'USERNAME',array('rows'=>6, 'cols'=>50));
?>
            <?php echo $form->error($modelAccountableUser,'USERNAME'); ?>
      </div>

      <div class="row">
```

```
            <?php echo $form->labelEx($modelAccountableUser,'PASSWORD');
?>
            <?php echo $form-
>passwordField($modelAccountableUser,'PASSWORD',array('rows'=>6,
'cols'=>50)); ?>
            <?php echo $form->error($modelAccountableUser,'PASSWORD'); ?>
      </div>

      <div class="row">
            <?php echo $form->labelEx($modelAccountableUser,'PASSWORD
REPEAT'); ?>
            <?php echo $form-
>passwordField($modelAccountableUser,'PASSWORD_repeat',array('rows'=>6,
'cols'=>50)); ?>
            <?php echo $form-
>error($modelAccountableUser,'PASSWORD_repeat'); ?>
      </div>

      <div class="row">
            <?php echo $form->labelEx($modelAccountableUser,'FIRST_NAME');
?>
            <?php echo $form-
>textField($modelAccountableUser,'FIRST_NAME',array('rows'=>6,
'cols'=>50)); ?>
            <?php echo $form->error($modelAccountableUser,'FIRST_NAME');
?>
      </div>

      <div class="row">
            <?php echo $form->labelEx($modelAccountableUser,'LAST_NAME');
?>
            <?php echo $form-
>textField($modelAccountableUser,'LAST_NAME',array('rows'=>6,
'cols'=>50)); ?>
            <?php echo $form->error($modelAccountableUser,'LAST_NAME'); ?>
      </div>

      <div class="row">
            <?php echo $form-
>labelEx($modelAccountableUser,'USER_INITIAL'); ?>
            <?php echo $form-
>textField($modelAccountableUser,'USER_INITIAL',array('rows'=>6,
'cols'=>50)); ?>
            <?php echo $form->error($modelAccountableUser,'USER_INITIAL');
?>
      </div>

      <div class="row">
            <?php echo $form->labelEx($modelAccountableUser,'BIRTHDAY');
?>
            <?php echo $form->DateField($modelAccountableUser,'BIRTHDAY');
?>
            <?php echo $form->error($modelAccountableUser,'BIRTHDAY'); ?>
      </div>
```

```
	<div class="row buttons">
		<?php echo CHtml::submitButton($modelUsers->isNewRecord ?
'Create' : 'Save'); ?>
	</div>

<?php $this->endWidget(); ?>

</div><!-- form -->
<?php
/* @var $this TrafficOperationsCenterOfficerController */
/* @var $model TrafficOperationsCenterOfficer */
/* @var $form CActiveForm */
?>

<div class="wide form">

<?php $form=$this->beginWidget('CActiveForm', array(
	'action'=>Yii::app()->createUrl($this->route),
	'method'=>'get',
)); ?>

	<div class="row">
		<?php echo $form->label($model,'USER_ID'); ?>
		<?php echo $form->textField($model,'USER_ID'); ?>
	</div>

	<div class="row buttons">
		<?php echo CHtml::submitButton('Search'); ?>
	</div>

<?php $this->endWidget(); ?>

</div><!-- search-form -->
<?php
/* @var $this TrafficOperationsCenterOfficerController */
/* @var $data TrafficOperationsCenterOfficer */
?>

<div class="view">

	<b><?php echo CHtml::encode($data->getAttributeLabel('USER_ID'));
?>:</b>
	<?php echo CHtml::link(CHtml::encode($data->USER_ID), array('view',
'id'=>$data->USER_ID)); ?>
	<br />

	<b><?php echo CHtml::encode($data->getAttributeLabel('USER_ID'));
?>:</b>
	<?php echo CHtml::encode($this->getName($data->USER_ID)); ?>
	<br />

</div>
<?php
```

```
/* @var $this TrafficOperationsCenterOfficerController */
/* @var $model TrafficOperationsCenterOfficer */

$this->breadcrumbs=array(
      'Traffic Operations Center Officers'=>array('index'),
      'Manage',
);

$this->menu=array(
      array('label'=>'List TrafficOperationsCenterOfficer',
'url'=>array('index')),
      array('label'=>'Create TrafficOperationsCenterOfficer',
'url'=>array('create')),
);

Yii::app()->clientScript->registerScript('search', "
$('.search-button').click(function(){
      $('.search-form').toggle();
      return false;
});
$('.search-form form').submit(function(){
      $('#traffic-operations-center-officer-grid').yiiGridView('update', {
            data: $(this).serialize()
      });
      return false;
});
");
?>

<h1>Manage Traffic Operations Center Officers</h1>

<p>
You may optionally enter a comparison operator (<b>&lt;</b>, <b>&lt;=</b>,
<b>&gt;</b>, <b>&gt;=</b>, <b>&lt;&gt;</b>
or <b>=</b>) at the beginning of each of your search values to specify how
the comparison should be done.
</p>

<?php echo CHtml::link('Advanced Search','#',array('class'=>'search-
button')); ?>
<div class="search-form" style="display:none">
<?php $this->renderPartial('_search',array(
      'model'=>$model,
)); ?>
</div><!-- search-form -->

<?php $this->widget('zii.widgets.grid.CGridView', array(
      'id'=>'traffic-operations-center-officer-grid',
      'dataProvider'=>$model->search(),
      'filter'=>$model,
      'columns'=>array(
            'USER_ID',
            array(
                  'class'=>'CButtonColumn',
```

```
            ),
      ),
)); ?>

<?php
/* @var $this TrafficOperationsCenterOfficerController */
/* @var $model TrafficOperationsCenterOfficer */

$this->breadcrumbs=array(
      'Traffic Operations Center Officers'=>array('index'),
      'Create',
);

$this->menu=array(
      array('label'=>'List TrafficOperationsCenterOfficer',
'url'=>array('index')),
      //array('label'=>'Manage TrafficOperationsCenterOfficer',
'url'=>array('admin')),
);
?>

<h1>Create TrafficOperationsCenterOfficer</h1>

<?php echo $this->renderPartial('_form',
array('modelTrafficOperationsCenterOfficer'=>$modelTrafficOperationsCenter
Officer,

'modelAccountableUser'=>$modelAccountableUser,

'modelUsers'=>$modelUsers,)
                                                    );
?>
<?php
/* @var $this TrafficOperationsCenterOfficerController */
/* @var $dataProvider CActiveDataProvider */

$this->breadcrumbs=array(
      'Traffic Operations Center Officers',
);

$this->menu=array(
      array('label'=>'Create TrafficOperationsCenterOfficer',
'url'=>array('create')),
      //array('label'=>'Manage TrafficOperationsCenterOfficer',
'url'=>array('admin')),
);
?>

<h1>Traffic Operations Center Officers</h1>

<?php $this->widget('zii.widgets.CListView', array(
      'dataProvider'=>$dataProvider,
      'itemView'=>'_view',
)); ?>
```

```
<?php
/* @var $this TrafficOperationsCenterOfficerController */
/* @var $model TrafficOperationsCenterOfficer */

$this->breadcrumbs=array(
      'Traffic Operations Center Officers'=>array('index'),
      $modelUsers->USER_ID=>array('view','id'=>$modelUsers->USER_ID),
      'Update',
);

$this->menu=array(
      array('label'=>'List TrafficOperationsCenterOfficer',
'url'=>array('index')),
      //array('label'=>'Create TrafficOperationsCenterOfficer',
'url'=>array('create')),
      array('label'=>'View TrafficOperationsCenterOfficer',
'url'=>array('view', 'id'=>$modelUsers->USER_ID)),
      //array('label'=>'Manage TrafficOperationsCenterOfficer',
'url'=>array('admin')),
);
?>

<h1>Update TrafficOperationsCenterOfficer <?php echo $modelUsers->USER_ID;
?></h1>

<?php echo $this->renderPartial('_form',
array('modelTrafficOperationsCenterOfficer'=>$modelTrafficOperationsCenter
Officer,

'modelAccountableUser'=>$modelAccountableUser,

'modelUsers'=>$modelUsers,)
                                                    );
?>
<?php
/* @var $this TrafficOperationsCenterOfficerController */
/* @var $model TrafficOperationsCenterOfficer */
$modelAccountableUser = AccountableUser::model()->findByPk($model-
>USER_ID);
$name = $modelAccountableUser->FIRST_NAME.' '.$modelAccountableUser-
>USER_INITIAL.'. '.$modelAccountableUser->LAST_NAME;
$this->breadcrumbs=array(
      'Traffic Operations Center Officers'=>array('index'),
      $model->USER_ID,
);

$this->menu=array(
      array('label'=>'List TrafficOperationsCenterOfficer',
'url'=>array('index')),
      //array('label'=>'Create TrafficOperationsCenterOfficer',
'url'=>array('create')),
      array('label'=>'Update TrafficOperationsCenterOfficer',
'url'=>array('update', 'id'=>$model->USER_ID)),
```

```
	//array('label'=>'Delete TrafficOperationsCenterOfficer',
'url'=>'#', 'linkOptions'=>array('submit'=>array('delete','id'=>$model-
>USER_ID),'confirm'=>'Are you sure you want to delete this item?')),
	//array('label'=>'Manage TrafficOperationsCenterOfficer',
'url'=>array('admin')),
);
?>

<h1>View TrafficOperationsCenterOfficer #<?php echo $model->USER_ID;
?></h1>

<?php $this->widget('zii.widgets.CDetailView', array(
	'data'=>$model,
	'attributes'=>array(
		'USER_ID',
		array(
			'name'=>'NAME',
			'value'=>$name
		),
	),
)); ?>

<?php
/* @var $this VirtualTripLineController */
/* @var $model VirtualTripLine */
/* @var $form CActiveForm */
?>

<div class="form">

<?php $form=$this->beginWidget('CActiveForm', array(
	'id'=>'virtual-trip-line-form',
	'enableAjaxValidation'=>false,
)); ?>

	<p class="note">Fields with <span class="required">*</span> are
required.</p>

	<?php echo $form->errorSummary($model); ?>

	<?php
	/* MAP */

	// init the map
	Yii::import('ext.jquery-gmap.*');
	$gmap = new EGmap3Widget();
	$temp;
	$options = array(
		'scaleControl' => true,
		'streetViewControl' => false,
		'zoom' => 16,
		'center' => array(0, 0),
		'mapTypeId' => EGmap3MapTypeId::ROADMAP,
		'mapTypeControlOptions' => array(
```

```
            'style' => EGmap3MapTypeControlStyle::DROPDOWN_MENU,
            'position' => EGmap3ControlPosition::TOP_CENTER,
      ),
);

$gmap->setOptions($options);
$gmap->setSize('100%', 200);

// create the marker
$marker = new EGmap3Marker(array(
      'title' => 'Draggable address marker 1',
      'draggable' => true,
));
$marker->latLng = array(14.4505957,120.9314332);
$marker->centerOnMap();

// set the marker to relay its position information a model
$marker->displayPosition(
       // the model object
       $model,
       // model's latitude property name
       'LL_X',
       // model's longitude property name
       'LL_Y',
       // Options set :
       //   show the fields, defaults to hidden fields
       //   update the fields during the marker drag event
       array('visible','drag')
);


$gmap->add($marker);
$gmap->renderMap();
?>

<div class="row">
      <?php echo $form->labelEx($model,'LL_X'); ?>
      <?php echo $form->textField($model,'LL_X'); ?>
      <?php echo $form->error($model,'LL_X'); ?>
</div>

<div class="row">
      <?php echo $form->labelEx($model,'LL_Y'); ?>
      <?php echo $form->textField($model,'LL_Y'); ?>
      <?php echo $form->error($model,'LL_Y'); ?>
</div>

<div class="row">
      <?php echo $form->labelEx($model,'LR_X'); ?>
      <?php echo $form->textField($model,'LR_X'); ?>
      <?php echo $form->error($model,'LR_X'); ?>
</div>

<div class="row">
```

```
      <?php echo $form->labelEx($model,'LR_Y'); ?>
      <?php echo $form->textField($model,'LR_Y'); ?>
      <?php echo $form->error($model,'LR_Y'); ?>
</div>

<div class="row">
      <?php echo $form->labelEx($model,'UL_X'); ?>
      <?php echo $form->textField($model,'UL_X'); ?>
      <?php echo $form->error($model,'UL_X'); ?>
</div>

<div class="row">
      <?php echo $form->labelEx($model,'UL_Y'); ?>
      <?php echo $form->textField($model,'UL_Y'); ?>
      <?php echo $form->error($model,'UL_Y'); ?>
</div>

<div class="row">
      <?php echo $form->labelEx($model,'UR_X'); ?>
      <?php echo $form->textField($model,'UR_X'); ?>
      <?php echo $form->error($model,'UR_X'); ?>
</div>

<div class="row">
      <?php echo $form->labelEx($model,'UR_Y'); ?>
      <?php echo $form->textField($model,'UR_Y'); ?>
      <?php echo $form->error($model,'UR_Y'); ?>
</div>

<div class="row">
      <?php echo $form->labelEx($model,'marker1_latitude'); ?>
      <?php echo $form->textField($model,'marker1_latitude'); ?>
      <?php echo $form->error($model,'marker1_latitude'); ?>
</div>

<div class="row">
      <?php echo $form->labelEx($model,'marker1_longitude'); ?>
      <?php echo $form->textField($model,'marker1_longitude'); ?>
      <?php echo $form->error($model,'marker1_longitude'); ?>
</div>

<div class="row">
      <?php echo $form->labelEx($model,'marker2_latitude'); ?>
      <?php echo $form->textField($model,'marker2_latitude'); ?>
      <?php echo $form->error($model,'marker2_latitude'); ?>
</div>

<div class="row">
      <?php echo $form->labelEx($model,'marker2_longitude'); ?>
      <?php echo $form->textField($model,'marker2_longitude'); ?>
      <?php echo $form->error($model,'marker2_longitude'); ?>
</div>
```

```
	<div class="row">
		<?php echo $form->labelEx($model,'DIRECTION'); ?>
		<?php echo $form->textArea($model,'DIRECTION',array('rows'=>6,
'cols'=>50)); ?>
		<?php echo $form->error($model,'DIRECTION'); ?>
	</div>

	<div class="row">
		<?php echo $form->labelEx($model,'DESCRIPTION'); ?>
		<?php echo $form-
>textArea($model,'DESCRIPTION',array('rows'=>6, 'cols'=>50)); ?>
		<?php echo $form->error($model,'DESCRIPTION'); ?>
	</div>

	<div class="row buttons">
		<?php echo CHtml::submitButton($model->isNewRecord ? 'Create'
: 'Save'); ?>
	</div>

<?php $this->endWidget(); ?>

</div><!-- form -->
<?php
/* @var $this VirtualTripLineController */
/* @var $model VirtualTripLine */
/* @var $form CActiveForm */
?>

<div class="wide form">

<?php $form=$this->beginWidget('CActiveForm', array(
	'action'=>Yii::app()->createUrl($this->route),
	'method'=>'get',
)); ?>

	<div class="row">
		<?php echo $form->label($model,'VIRTUAL_TRIP_LINE_ID'); ?>
		<?php echo $form->textField($model,'VIRTUAL_TRIP_LINE_ID'); ?>
	</div>

	<div class="row">
		<?php echo $form->label($model,'X1'); ?>
		<?php echo $form->textField($model,'X1'); ?>
	</div>

	<div class="row">
		<?php echo $form->label($model,'Y1'); ?>
		<?php echo $form->textField($model,'Y1'); ?>
	</div>

	<div class="row">
		<?php echo $form->label($model,'X2'); ?>
		<?php echo $form->textField($model,'X2'); ?>
	</div>
```

```
	<div class="row">
		<?php echo $form->label($model,'Y2'); ?>
		<?php echo $form->textField($model,'Y2'); ?>
	</div>

	<div class="row">
		<?php echo $form->label($model,'DIRECTION'); ?>
		<?php echo $form->textArea($model,'DIRECTION',array('rows'=>6,
'cols'=>50)); ?>
	</div>

	<div class="row">
		<?php echo $form->label($model,'DESCRIPTION'); ?>
		<?php echo $form-
>textArea($model,'DESCRIPTION',array('rows'=>6, 'cols'=>50)); ?>
	</div>

	<div class="row buttons">
		<?php echo CHtml::submitButton('Search'); ?>
	</div>

<?php $this->endWidget(); ?>

</div><!-- search-form -->
<?php
/* @var $this VirtualTripLineController */
/* @var $data VirtualTripLine */
?>

<div class="view">

	<b><?php echo CHtml::encode($data-
>getAttributeLabel('VIRTUAL_TRIP_LINE_ID')); ?>:</b>
	<?php echo CHtml::link(CHtml::encode($data->VIRTUAL_TRIP_LINE_ID),
array('view', 'id'=>$data->VIRTUAL_TRIP_LINE_ID)); ?>
	<br />

	<b><?php echo CHtml::encode($data-
>getAttributeLabel('DESCRIPTION')); ?>:</b>
	<?php echo CHtml::encode($data->DESCRIPTION); ?>
	<br />


</div>
<?php
/* @var $this VirtualTripLineController */
/* @var $model VirtualTripLine */

$this->breadcrumbs=array(
	'Virtual Trip Lines'=>array('index'),
	'Manage',
);
```

```
$this->menu=array(
	array('label'=>'List VirtualTripLine', 'url'=>array('index')),
	array('label'=>'Create VirtualTripLine', 'url'=>array('create')),
);

Yii::app()->clientScript->registerScript('search', "
$('.search-button').click(function(){
	$('.search-form').toggle();
	return false;
});
$('.search-form form').submit(function(){
	$('#virtual-trip-line-grid').yiiGridView('update', {
		data: $(this).serialize()
	});
	return false;
});
");
?>

<h1>Manage Virtual Trip Lines</h1>

<p>
You may optionally enter a comparison operator (<b>&lt;</b>, <b>&lt;=</b>,
<b>&gt;</b>, <b>&gt;=</b>, <b>&lt;&gt;</b>
or <b>=</b>) at the beginning of each of your search values to specify how
the comparison should be done.
</p>

<?php echo CHtml::link('Advanced Search','#',array('class'=>'search-
button')); ?>
<div class="search-form" style="display:none">
<?php $this->renderPartial('_search',array(
	'model'=>$model,
)); ?>
</div><!-- search-form -->

<?php $this->widget('zii.widgets.grid.CGridView', array(
	'id'=>'virtual-trip-line-grid',
	'dataProvider'=>$model->search(),
	'filter'=>$model,
	'columns'=>array(
		'VIRTUAL_TRIP_LINE_ID',
		'X1',
		'Y1',
		'X2',
		'Y2',
		'DIRECTION',
		/*
		'DESCRIPTION',
		*/
		array(
			'class'=>'CButtonColumn',
		),
	),
```

```
)); ?>

<?php
/* @var $this VirtualTripLineController */
/* @var $model VirtualTripLine */

$this->breadcrumbs=array(
      'Virtual Trip Lines'=>array('index'),
      'Create',
);

$this->menu=array(
      array('label'=>'List VirtualTripLine', 'url'=>array('index')),
      array('label'=>'Manage VirtualTripLine', 'url'=>array('admin')),
);
?>

<h1>Create Virtual Trip Line</h1>

<?php echo $this->renderPartial('_form', array('model'=>$model)); ?>
<?php
/* @var $this VirtualTripLineController */
/* @var $dataProvider CActiveDataProvider */

$this->breadcrumbs=array(
      'Virtual Trip Lines',
);

$this->menu=array(
      array('label'=>'Create VirtualTripLine', 'url'=>array('create')),
      array('label'=>'Manage VirtualTripLine', 'url'=>array('admin')),
);
?>

<h1>Virtual Trip Lines</h1>

<?php $this->widget('zii.widgets.CListView', array(
      'dataProvider'=>$dataProvider,
      'itemView'=>'_view',
)); ?>

<?php
/* @var $this VirtualTripLineController */
/* @var $model VirtualTripLine */

$this->breadcrumbs=array(
      'Virtual Trip Lines'=>array('index'),
      $model->VIRTUAL_TRIP_LINE_ID=>array('view','id'=>$model-
>VIRTUAL_TRIP_LINE_ID),
      'Update',
);

$this->menu=array(
      array('label'=>'List VirtualTripLine', 'url'=>array('index')),
```

```
	array('label'=>'Create VirtualTripLine', 'url'=>array('create')),
	array('label'=>'View VirtualTripLine', 'url'=>array('view',
'id'=>$model->VIRTUAL_TRIP_LINE_ID)),
	array('label'=>'Manage VirtualTripLine', 'url'=>array('admin')),
);
?>

<h1>Update VirtualTripLine <?php echo $model->VIRTUAL_TRIP_LINE_ID;
?></h1>

<?php echo $this->renderPartial('_form', array('model'=>$model)); ?>
<?php
/* @var $this VirtualTripLineController */
/* @var $model VirtualTripLine */

$this->breadcrumbs=array(
	'Virtual Trip Lines'=>array('index'),
	$model->VIRTUAL_TRIP_LINE_ID,
);

$this->menu=array(
	array('label'=>'List VirtualTripLine', 'url'=>array('index')),
	array('label'=>'Create VirtualTripLine', 'url'=>array('create')),
	array('label'=>'Update VirtualTripLine', 'url'=>array('update',
'id'=>$model->VIRTUAL_TRIP_LINE_ID)),
	array('label'=>'Delete VirtualTripLine', 'url'=>'#',
'linkOptions'=>array('submit'=>array('delete','id'=>$model-
>VIRTUAL_TRIP_LINE_ID),'confirm'=>'Are you sure you want to delete this
item?')),
	array('label'=>'Manage VirtualTripLine', 'url'=>array('admin')),
);
?>

<h1>View VirtualTripLine #<?php echo $model->VIRTUAL_TRIP_LINE_ID; ?></h1>

<?php $this->widget('zii.widgets.CDetailView', array(
	'data'=>$model,
	'attributes'=>array(
		'VIRTUAL_TRIP_LINE_ID',
		'LL_X',
		'LL_Y',
		'LR_X',
		'LR_Y',
		'UL_X',
		'UL_Y',
		'UR_X',
		'UR_Y',
		'DIRECTION',
		'DESCRIPTION',
	),
)); ?>

<?php
```

```
/**
 * EGmap3 Yii extension example view file.
 *
 * You can copy this file or its contents into your Yii
 * application for testing.
 *
 */
Yii::import('ext.jquery-gmap.*');

$gmap = new EGmap3Widget();
$gmap->setSize('100%', 400);

// base options
$options = array(
      'scaleControl' => true,
      'streetViewControl' => false,
      'zoom' => 16,
      'center' => array(0, 0),
      'mapTypeId' => EGmap3MapTypeId::ROADMAP,
      'mapTypeControlOptions' => array(
            'style' => EGmap3MapTypeControlStyle::DROPDOWN_MENU,
            'position' => EGmap3ControlPosition::TOP_CENTER,
      ),
      'zoomControlOptions' => array(
            'style' => EGmap3ZoomControlStyle::SMALL,
            'position' => EGmap3ControlPosition::BOTTOM_CENTER
      ),
);
$gmap->setOptions($options);

$marker = new EGmap3Marker(array(
      'title' => 'Draggable address marker 1',
      'draggable' => true,
));
$marker->latLng = array($model->LL_X, $model->LL_Y);
$marker->centerOnMap();
$gmap->add($marker);

$polyOptions = array(
      'fillColor' => 'yellow',
      'strokeColor' => 'red'
);
/*
$rectangleOptions = array_merge(
            $polyOptions, array('bounds' => array(
                                                                  $model->X1,
                                                                  $model->Y1,
                                                                  $model->X2,
                                                                  $model->Y2

            ))
);
$rectangle = new EGmap3Rectangle($rectangleOptions);
*/
$polygon = new EGmap3Polygon($polyOptions);
```

```
$polygon->paths = array(
      array($model->LL_X, $model->LL_Y),
      array($model->LR_X, $model->LR_Y),
      array($model->UR_X, $model->UR_Y),
      array($model->UL_X, $model->UL_Y),
);
$gmap->add($polygon);

// adding a route automatically centers it on the map, overriding all
other
// centering calls
/*
$route = new EGmap3Route(array(
                  'origin' => 'New York, NY',
                  'destination' => 'Acapulco, México',
           ));
$gmap->add($route);
*/
$gmap->renderMap();
?>

<?php
require ("dbConfig.php");
require ("tools.php");

$distanceFromReport = 999999999999;
$resultData = array();

/*
$currLat = 14.450638;
$currLong = 120.9314595;
*/

/*
$currLat = 14.450575;
$currLong = 120.932313;
*/


//$currLat = $_POST['lat'];
//$currLong = $_POST['longitude'];
//$mobileId = $_POST['mobileId'];
$currLat = 14.45072787;
$currLong = 120.93154966;
$mobileId = '177d16e5d6265252';

//get user id

$userID;
$qry = "SELECT USER_ID FROM motorist WHERE MOBILE_ID = '$mobileId';";

$result=mysql_query($qry) or die('THERE WAS AN ERROR IN GETTING THE
DATA');
```

```
while($data=mysql_fetch_array($result, MYSQL_ASSOC)){
      $userID = $data['USER_ID'];
}



$qry = "SELECT report.REPORT_ID, location_point.LATITUDE,
location_point.LONGITUDE
            FROM report
            inner join location_point
            ON report.LOCATION_POINT_ID=location_point.LOCATION_POINT_ID
            inner join road_closed_report
            ON report.REPORT_ID=road_closed_report.REPORT_ID
            WHERE report.IS_VERIFIED_BY_SYSTEM = 0
            AND report.START_DATETIME < NOW() AND NOW() <
report.END_DATETIME;
            ";

$result=mysql_query($qry) or die('THERE WAS AN ERROR IN GETTING THE
DATA');

while($data=mysql_fetch_array($result, MYSQL_ASSOC)){
      if(computeDistanceLatLong($currLat, $currLong, $data['LATITUDE'],
$data['LONGITUDE']) < $distanceFromReport){
            //echo $data['REPORT_ID'].'<br/>';
            //query if user already verified the report
            $reportId = $data['REPORT_ID'];
            $IsAlreadyReported;
            $qry2 = "SELECT * FROM user_verified_report WHERE
USER_ID=$userID AND $reportId=REPORT_ID;";

            $result2=mysql_query($qry2) or die('THERE WAS AN ERROR IN
GETTING THE DATA');
            $num_rows = mysql_num_rows($result2);

            $qry2 = "SELECT * FROM report WHERE USER_ID=$userID AND
$reportId=REPORT_ID;";

            $result2=mysql_query($qry2) or die('THERE WAS AN ERROR IN
GETTING THE DATA');
            $num_rows = $num_rows + mysql_num_rows($result2);


            if($num_rows == 0){
                  $resultData[]=array(
                        'type'=>'road closed',
                        'id'=>$data['REPORT_ID'],
                        'latitude'=>$data['LATITUDE'],
                        'longitude'=>$data['LONGITUDE'],
                        'type'=>'road closed'
                  );
            }
      }
```

```
}

$qry = "SELECT report.REPORT_ID, location_point.LATITUDE,
location_point.LONGITUDE
            FROM report
            inner join location_point
            ON report.LOCATION_POINT_ID=location_point.LOCATION_POINT_ID
            inner join ra_report
            ON report.REPORT_ID=ra_report.REPORT_ID
            WHERE report.IS_VERIFIED_BY_SYSTEM = 0
            AND report.START_DATETIME < NOW() AND NOW() <
report.END_DATETIME;
            ";

$result=mysql_query($qry) or die('THERE WAS AN ERROR IN GETTING THE
DATA');

while($data=mysql_fetch_array($result, MYSQL_ASSOC)){
      if(computeDistanceLatLong($currLat, $currLong, $data['LATITUDE'],
$data['LONGITUDE']) < $distanceFromReport){
            //query if user already verified the report
            $reportId = $data['REPORT_ID'];
            $IsAlreadyReported;
            $qry2 = "SELECT * FROM user_verified_report WHERE
USER_ID=$userID AND $reportId=REPORT_ID;";

            $result2=mysql_query($qry2) or die('THERE WAS AN ERROR IN
GETTING THE DATA');
            $num_rows = mysql_num_rows($result2);

            $qry2 = "SELECT * FROM report WHERE USER_ID=$userID AND
$reportId=REPORT_ID;";

            $result2=mysql_query($qry2) or die('THERE WAS AN ERROR IN
GETTING THE DATA');
            $num_rows = $num_rows + mysql_num_rows($result2);

            if($num_rows == 0){
                  $resultData[]=array(
                        'type'=>'road accident',
                        'id'=>$data['REPORT_ID'],
                        'latitude'=>$data['LATITUDE'],
                        'longitude'=>$data['LONGITUDE'],
                        'type'=>'road accident'
                  );
            }


      }

}

$qry = "SELECT report.REPORT_ID, location_point.LATITUDE,
location_point.LONGITUDE
```

```
            FROM report
            inner join location_point
            ON report.LOCATION_POINT_ID=location_point.LOCATION_POINT_ID
            inner join fs_report
            ON report.REPORT_ID=fs_report.REPORT_ID
            WHERE report.IS_VERIFIED_BY_SYSTEM = 0
            AND report.START_DATETIME < NOW() AND NOW() <
report.END_DATETIME;;
            ";

$result=mysql_query($qry) or die('THERE WAS AN ERROR IN GETTING THE
DATA');

while($data=mysql_fetch_array($result, MYSQL_ASSOC)){
      if(computeDistanceLatLong($currLat, $currLong, $data['LATITUDE'],
$data['LONGITUDE']) < $distanceFromReport){

            //query if user already verified the report
            $reportId = $data['REPORT_ID'];
            $IsAlreadyReported;
            $qry2 = "SELECT * FROM user_verified_report WHERE
USER_ID=$userID AND $reportId=REPORT_ID;";

            $result2=mysql_query($qry2) or die('THERE WAS AN ERROR IN
GETTING THE DATA');
            $num_rows = mysql_num_rows($result2);

            $qry2 = "SELECT * FROM report WHERE USER_ID=$userID AND
$reportId=REPORT_ID;";

            $result2=mysql_query($qry2) or die('THERE WAS AN ERROR IN
GETTING THE DATA');
            $num_rows = $num_rows + mysql_num_rows($result2);


            if($num_rows == 0){
                  //push data to array
                        $resultData[]=array(
                        'type'=>'flooded street',
                        'id'=>$data['REPORT_ID'],
                        'latitude'=>$data['LATITUDE'],
                        'longitude'=>$data['LONGITUDE'],
                        'type'=>'flooded street'
                  );
            }

      }

}

echo json_encode($resultData);

?>
<?php
```

```
    define('DB_HOST', 'localhost');
    define('DB_USER', 'NavigateSupport');
    define('DB_PASSWORD', 'aXVDEh2yDDCMTc5v');
    define('DB_DATABASE', 'NavigateSupport');
    $conn=mysql_pconnect(DB_HOST, DB_USER, DB_PASSWORD) or die ("Error
connecting to sp_test database");
    mysql_select_db(DB_DATABASE) or die ("Error: Cannot access $dbname
database");
?>
<?php
require ("dbConfig.php");


$qry = "SELECT * FROM  fs_category";
$result=mysql_query($qry) or die('THERE WAS AN ERROR IN GETTING THE
DATA');

$categoryFS = array();

while ($category=mysql_fetch_array($result, MYSQL_ASSOC)) {
      $categoryFS[]=array(
            'id'=>$category['FS_CATEGORY_ID'],
            'name'=>$category['NAME']
     );
}


echo json_encode($categoryFS);
?>
<?php
//load db config
require ("dbConfig.php");

//load tools
require ("tools.php");
//get data from mobile
$currLat = $_POST['lat'];
$currLong = $_POST['longitude'];
//$currLat = 14.45068;
//$currLong = 120.931728;


//variables
$distanceFromReport = 99999999;
$resultData = array();


//get road closed report
$qry = "SELECT report.REPORT_ID, location_point.LATITUDE,
location_point.LONGITUDE , report.PICTURE_ID, report.DATETIME_CREATED,
report.IS_VERIFIED_BY_SYSTEM, report.START_DATETIME, report.END_DATETIME
            FROM report
            inner join location_point
            ON report.LOCATION_POINT_ID=location_point.LOCATION_POINT_ID
```

```
            inner join road_closed_report
            ON report.REPORT_ID=road_closed_report.REPORT_ID
            WHERE report.START_DATETIME < NOW() AND NOW() <
report.END_DATETIME;
            ";

$result=mysql_query($qry) or die('THERE WAS AN ERROR IN GETTING THE
DATA');

while($data=mysql_fetch_array($result, MYSQL_ASSOC)){
      if(computeDistanceLatLong($currLat, $currLong, $data['LATITUDE'],
$data['LONGITUDE']) < $distanceFromReport){

            $qry = "SELECT * FROM user_verified_report WHERE REPORT_ID =
".$data['REPORT_ID']."
            AND FLAG_UP = 1;";
            $link = mysql_query($qry);
            $num_flag_up = mysql_num_rows($link);

            $qry = "SELECT * FROM user_verified_report WHERE REPORT_ID =
".$data['REPORT_ID']."
            AND FLAG_UP = 0;";
            $link = mysql_query($qry);
            $num_flag_down = mysql_num_rows($link);

            $resultData[]=array(
                  'type'=>'road closed',
                  'pictureId'=>$data['PICTURE_ID'],
             'id'=>$data['REPORT_ID'],
             'latitude'=>$data['LATITUDE'],
                  'longitude'=>$data['LONGITUDE'],
                  'type'=>'Road Closed',
                  'isVerifiedBySystem'=>$data['IS_VERIFIED_BY_SYSTEM'],
                  'classification'=>'',
                  'dateCreated'=>$data['DATETIME_CREATED'],
                  'flagUp'=>$num_flag_up,
                  'flagDown'=>$num_flag_down
            );
      }

}

//get road accident reports
$qry = "SELECT report.REPORT_ID, location_point.LATITUDE,
location_point.LONGITUDE, ra_category.NAME, report.PICTURE_ID,
report.DATETIME_CREATED, report.IS_VERIFIED_BY_SYSTEM
            FROM report
            inner join location_point
            ON report.LOCATION_POINT_ID=location_point.LOCATION_POINT_ID
            inner join ra_report
            ON report.REPORT_ID=ra_report.REPORT_ID
            inner join ra_category
            ON ra_report.RA_CATEGORY_ID = ra_category.RA_CATEGORY_ID
```

```
            WHERE report.START_DATETIME < NOW() AND NOW() <
report.END_DATETIME;
            ";

$result=mysql_query($qry) or die('THERE WAS AN ERROR IN GETTING THE
DATA');

while($data=mysql_fetch_array($result, MYSQL_ASSOC)){
      if(computeDistanceLatLong($currLat, $currLong, $data['LATITUDE'],
$data['LONGITUDE']) < $distanceFromReport){
            $qry = "SELECT * FROM user_verified_report WHERE REPORT_ID =
".$data['REPORT_ID']."
            AND FLAG_UP = 1;";
            $link = mysql_query($qry);
            $num_flag_up = mysql_num_rows($link);


            $qry = "SELECT * FROM user_verified_report WHERE REPORT_ID =
".$data['REPORT_ID']."
            AND FLAG_UP = 0;";
            $link = mysql_query($qry);
            $num_flag_down = mysql_num_rows($link);


            $resultData[]=array(
                  'type'=>'road accident',
                  'pictureId'=>$data['PICTURE_ID'],
             'id'=>$data['REPORT_ID'],
             'latitude'=>$data['LATITUDE'],
                  'longitude'=>$data['LONGITUDE'],
                  'type'=>'Road Accident',
                  'classification'=>$data['NAME'],
                  'isVerifiedBySystem'=>$data['IS_VERIFIED_BY_SYSTEM'],
                  'dateCreated'=>$data['DATETIME_CREATED'],
                  'flagUp'=>$num_flag_up,
                  'flagDown'=>$num_flag_down
            );
      }

}

//get flooded roads
$qry = "SELECT report.REPORT_ID, location_point.LATITUDE,
location_point.LONGITUDE, fs_category.NAME, report.PICTURE_ID,
report.DATETIME_CREATED, report.IS_VERIFIED_BY_SYSTEM
            FROM report
            inner join location_point
            ON report.LOCATION_POINT_ID=location_point.LOCATION_POINT_ID
            inner join fs_report
            ON report.REPORT_ID=fs_report.REPORT_ID
            inner join fs_category
            ON fs_report.FS_CATEGORY_ID=fs_category.FS_CATEGORY_ID
            WHERE report.START_DATETIME < NOW() AND NOW() <
report.END_DATETIME;
```

```
            ";

$result=mysql_query($qry) or die('THERE WAS AN ERROR IN GETTING THE
DATA');

while($data=mysql_fetch_array($result, MYSQL_ASSOC)){
      if(computeDistanceLatLong($currLat, $currLong, $data['LATITUDE'],
$data['LONGITUDE']) < $distanceFromReport){
            $qry = "SELECT * FROM user_verified_report WHERE REPORT_ID =
".$data['REPORT_ID']."
            AND FLAG_UP = 1;";
            $link = mysql_query($qry);
            $num_flag_up = mysql_num_rows($link);

            $qry = "SELECT * FROM user_verified_report WHERE REPORT_ID =
".$data['REPORT_ID']."
            AND FLAG_UP = 0;";
            $link = mysql_query($qry);
            $num_flag_down = mysql_num_rows($link);


            //push data to array
                  $resultData[]=array(
                  'type'=>'flooded street',
                  'pictureId'=>$data['PICTURE_ID'],
                  'id'=>$data['REPORT_ID'],
                  'latitude'=>$data['LATITUDE'],
                  'longitude'=>$data['LONGITUDE'],
                  'type'=>'Flooded Street',
                  'classification'=>$data['NAME'],
                  'isVerifiedBySystem'=>$data['IS_VERIFIED_BY_SYSTEM'],
                  'dateCreated'=>$data['DATETIME_CREATED'],
                  'flagUp'=>$num_flag_up,
                  'flagDown'=>$num_flag_down
            );
      }


}

echo json_encode($resultData);
?>
<?php
require ("dbConfig.php");


$qry = "SELECT * FROM  ra_category";
$result=mysql_query($qry) or die('THERE WAS AN ERROR IN GETTING THE
DATA');
$categoryRA = array();

while ($category=mysql_fetch_array($result, MYSQL_ASSOC)) {
      $categoryRA[]=array(
            'id'=>$category['RA_CATEGORY_ID'],
```

```
            'name'=>$category['NAME']
      );
}


echo json_encode($categoryRA);
?>
<?php

require ("dbConfig.php");
//load tools
require ("tools.php");

//$currLat = $_POST['lat'];
//$currLon = $_POST['longitude'];
//$mobileId = $_POST['mobileId'];

$mobileId = '177d16e5d6265252';
$currLat = 14.45035;
$currLon = 120.932066;
//$currLat = 12;
//$currLon = 123.931803;
//14.446547,120.9305295
//get user id

$userID;
$qry = "SELECT USER_ID FROM motorist WHERE MOBILE_ID = '$mobileId';";

$result=mysql_query($qry) or die('THERE WAS AN ERROR IN GETTING THE
DATA');

while($data=mysql_fetch_array($result, MYSQL_ASSOC)){
      $userID = $data['USER_ID'];
}

//get VTL user is inside
$response = null;

$qry =
      "
            SELECT *
            FROM virtual_trip_line;
      ";

$result=mysql_query($qry) or die('THERE WAS AN ERROR IN GETTING THE
DATA');

while($data=mysql_fetch_array($result, MYSQL_ASSOC)){
      $temp = $data['VIRTUAL_TRIP_LINE_ID'];

      if(isInsideRectangle($data['LL_X'],$data['LL_Y'],$data['LR_X'],$data
['LR_Y'],$data['UL_X'],$data['UL_Y'],$data['UR_X'],$data['UR_Y'],$currLat,
$currLon)){
            $response[]=array(
```

```
                'vtlid'=>$data['VIRTUAL_TRIP_LINE_ID'],
                'direction'=>$data['DIRECTION']
            );
      }


}

if($response == null){
      echo 0;
}else{
      echo json_encode($response);
}


?>
<?php
require ("dbConfig.php");


$mobileID = $_POST['id'];
$category = $_POST['category'];
$lat = $_POST['lat'];
$long = $_POST['longitude'];

/*
$mobileID = '177d16e5d6265252';
$category = 'Not passable to light vehicles';
$lat = 14.45068;
$long = 120.931728;
*/
$qry = "INSERT INTO location_point (LATITUDE, LONGITUDE)
            VALUES ('$lat','$long');";

$result=mysql_query($qry) or die('THERE WAS AN ERROR IN GETTING THE
DATA');

if(mysql_errno($conn) == 1062){
      echo "1st: 1062";
}else{
      echo "1st: 0";
}

// get location point id
$locationPointID;
$qry = "SELECT LOCATION_POINT_ID FROM location_point WHERE
LOCATION_POINT_ID = LAST_INSERT_ID();";

$result=mysql_query($qry) or die('THERE WAS AN ERROR IN GETTING THE
DATA');


while($data=mysql_fetch_array($result, MYSQL_ASSOC)){
      $locationPointID = $data['LOCATION_POINT_ID'];
```

```
}

echo "loc: $locationPointID \n";
//get user id
$userID;
$qry = "SELECT USER_ID FROM motorist WHERE MOBILE_ID = '$mobileID';";

$result=mysql_query($qry) or die('THERE WAS AN ERROR IN GETTING THE
DATA');

while($data=mysql_fetch_array($result, MYSQL_ASSOC)){
      $userID = $data['USER_ID'];
}

echo "id: $userID \n";
//insert to report
$qry = "INSERT INTO report (LOCATION_POINT_ID,USER_ID,DATETIME_CREATED,
START_DATETIME, END_DATETIME)
            VALUES
('$locationPointID','$userID',NOW(),NOW(),ADDDATE(NOW(),INTERVAL 1
HOUR));";

$result=mysql_query($qry) or die('THERE WAS AN ERROR IN GETTING THE
DATA');

if(mysql_errno($conn) == 1062){
      echo "2nd: 1062";
}else{
      echo "2nd: 0 \n";
}

//get report id
$reportID;

$qry = "SELECT REPORT_ID FROM report WHERE REPORT_ID = LAST_INSERT_ID();";

$result=mysql_query($qry) or die('THERE WAS AN ERROR IN GETTING THE
DATA');

while($data=mysql_fetch_array($result, MYSQL_ASSOC)){
      $reportID = $data['REPORT_ID'];
}

echo "report id: $reportID \n";
//get category id
$categoryID;

$qry = "SELECT FS_CATEGORY_ID FROM fs_category WHERE NAME = '$category';";

$result=mysql_query($qry) or die('THERE WAS AN ERROR IN GETTING THE
DATA');

while($data=mysql_fetch_array($result, MYSQL_ASSOC)){
      $categoryID = $data['FS_CATEGORY_ID'];
```

```
}

echo "category id: $categoryID \n";
//insert to fs report
$qry = "INSERT INTO fs_report (REPORT_ID, FS_CATEGORY_ID)
            VALUES (
            '$reportID',
            '$categoryID'
            );";

$result=mysql_query($qry) or die('THERE WAS AN ERROR IN GETTING THE
DATA');

if(mysql_errno($conn) == 1062){
      echo "3rd: " + mysql_errno($conn);
}else{
      echo "3rd: " + mysql_errno($conn);
}

?>
<?php
require ("dbConfig.php");


$mobileID = $_POST['id'];
$category = $_POST['category'];
$lat = $_POST['lat'];
$long = $_POST['longitude'];;


$qry = "INSERT INTO location_point (LATITUDE, LONGITUDE)
            VALUES ('$lat','$long');";

$result=mysql_query($qry) or die('THERE WAS AN ERROR IN GETTING THE
DATA');

if(mysql_errno($conn) == 1062){
      echo "1st: 1062 \n";
}else{
      echo "1st: 0 \n";
}

// get location point id
$locationPointID;
$qry = "SELECT LOCATION_POINT_ID FROM location_point WHERE
LOCATION_POINT_ID = LAST_INSERT_ID();";

$result=mysql_query($qry) or die('THERE WAS AN ERROR IN GETTING THE
DATA');


while($data=mysql_fetch_array($result, MYSQL_ASSOC)){
      $locationPointID = $data['LOCATION_POINT_ID'];
}
```

```
echo "loc: $locationPointID \n";
//get user id
$userID;
$qry = "SELECT USER_ID FROM motorist WHERE MOBILE_ID = '$mobileID';";

$result=mysql_query($qry) or die('THERE WAS AN ERROR IN GETTING THE
DATA');

while($data=mysql_fetch_array($result, MYSQL_ASSOC)){
      $userID = $data['USER_ID'];
}

echo "id: $userID \n";
//insert to report
$qry = "INSERT INTO report (LOCATION_POINT_ID,USER_ID,DATETIME_CREATED,
START_DATETIME, END_DATETIME)
           VALUES
('$locationPointID','$userID',NOW(),NOW(),ADDDATE(NOW(),INTERVAL 1
HOUR));";

$result=mysql_query($qry) or die('THERE WAS AN ERROR IN GETTING THE
DATA');

if(mysql_errno($conn) == 1062){
      echo "2nd: 1062";
}else{
      echo "2nd: 0 \n";
}

//get report id
$reportID;

$qry = "SELECT REPORT_ID FROM report WHERE REPORT_ID = LAST_INSERT_ID();";

$result=mysql_query($qry) or die('THERE WAS AN ERROR IN GETTING THE
DATA');

while($data=mysql_fetch_array($result, MYSQL_ASSOC)){
      $reportID = $data['REPORT_ID'];
}

echo "report id: $reportID \n";
//get category id
$categoryID;

$qry = "SELECT RA_CATEGORY_ID FROM ra_category WHERE NAME = '$category';";

$result=mysql_query($qry) or die('THERE WAS AN ERROR IN GETTING THE
DATA');

while($data=mysql_fetch_array($result, MYSQL_ASSOC)){
      $categoryID = $data['RA_CATEGORY_ID'];
}
```

```
echo "category id: $categoryID \n";
//insert to fs report
$qry = "INSERT INTO ra_report (REPORT_ID, RA_REPORT_STATUS_ID,
RA_CATEGORY_ID )
            VALUES (
            '$reportID',
            '1',
            '$categoryID'
            );";

$result=mysql_query($qry) or die('THERE WAS AN ERROR IN GETTING THE
DATA');

if(mysql_errno($conn) == 1062){
      echo "3rd: " + mysql_errno($conn);
}else{
      echo "3rd: " + mysql_errno($conn);
}

?>
<?php
require ("dbConfig.php");


$mobileID = $_POST['id'];
$lat = $_POST['lat'];
$long = $_POST['longitude'];;


$qry = "INSERT INTO location_point (LATITUDE, LONGITUDE)
            VALUES ('$lat','$long');";

$result=mysql_query($qry) or die('THERE WAS AN ERROR IN GETTING THE
DATA');

if(mysql_errno($conn) == 1062){
      echo "1st: 1062 \n";
}else{
      echo "1st: 0 \n";
}

// get location point id
$locationPointID;
$qry = "SELECT LOCATION_POINT_ID FROM location_point WHERE
LOCATION_POINT_ID = LAST_INSERT_ID();";

$result=mysql_query($qry) or die('THERE WAS AN ERROR IN GETTING THE
DATA');


while($data=mysql_fetch_array($result, MYSQL_ASSOC)){
      $locationPointID = $data['LOCATION_POINT_ID'];
}
```

```
echo "loc: $locationPointID \n";
//get user id
$userID;
$qry = "SELECT USER_ID FROM motorist WHERE MOBILE_ID = '$mobileID';";

$result=mysql_query($qry) or die('THERE WAS AN ERROR IN GETTING THE
DATA');

while($data=mysql_fetch_array($result, MYSQL_ASSOC)){
      $userID = $data['USER_ID'];
}

echo "id: $userID \n";
//insert to report
$qry = "INSERT INTO report (LOCATION_POINT_ID,USER_ID,DATETIME_CREATED,
START_DATETIME, END_DATETIME)
            VALUES
('$locationPointID','$userID',NOW(),NOW(),ADDDATE(NOW(),INTERVAL 1
HOUR));";

$result=mysql_query($qry) or die('THERE WAS AN ERROR IN GETTING THE
DATA');

if(mysql_errno($conn) == 1062){
      echo "2nd: 1062";
}else{
      echo "2nd: 0 \n";
}

//get report id
$reportID;

$qry = "SELECT REPORT_ID FROM report WHERE REPORT_ID = LAST_INSERT_ID();";

$result=mysql_query($qry) or die('THERE WAS AN ERROR IN GETTING THE
DATA');

while($data=mysql_fetch_array($result, MYSQL_ASSOC)){
      $reportID = $data['REPORT_ID'];
}

echo "report id: $reportID \n";

//insert to road closed report
$qry = "INSERT INTO road_closed_report (REPORT_ID)
            VALUES (
            '$reportID'
            );";

$result=mysql_query($qry) or die('THERE WAS AN ERROR IN GETTING THE
DATA');

if(mysql_errno($conn) == 1062){
```

```
      echo "3rd: " + mysql_errno($conn);
}else{
      echo "3rd: " + mysql_errno($conn);
}

?>
<?php
//get data from mobile
$lat = $_POST['lat'];
$longitude = $_POST['longitude'];
$vtlId = $_POST['vtlId'];
$mobileId = $_POST['mobileId'];
$speed = $_POST['speed'];

echo "lat: $lat \n";
echo "longitude:  $longitude\n";
echo "vtlId: $vtlId \n ";
echo "mobileId: $mobileId \n ";
echo "speed: $speed \n ";

//get user id

$userID;
$qry = "SELECT USER_ID FROM crowd_resource WHERE MOBILE_ID =
'$mobileId';";

$result=mysql_query($qry) or die('THERE WAS AN ERROR IN GETTING THE
DATA');

while($data=mysql_fetch_array($result, MYSQL_ASSOC)){
      $userID = $data['USER_ID'];
}

echo "userId: $userID \n ";


if($flag == 1){
      echo "yes";
}else if($flag == 1){
      echo "no";
}

//insert to database

$qry = "INSERT INTO traffic_location_point
(LATITUDE,LONGITUDE,DATE_CREATED,DATA_LOCATION_COLLECTION_ID,SPEED)
            VALUES ('$lat','$longitude',NOW(),'1',$speed);";

$result=mysql_query($qry) or die('THERE WAS AN ERROR IN GETTING THE
DATA');

echo "test";
?>
<?php
```

```
//load db config
require ("dbConfig.php");


//get data

$reportId = $_POST['reportId'];
$mobileId = $_POST['mobileId'];
$flag = $_POST['flag'];
/*
$reportId = 6;
$mobileId = '177d16e5d6265252';
$flag = 1;
*/

echo "reportId: $reportId \n";
echo "mobileId:  $mobileId\n";
echo "flag: $flag \n ";

//get user id

$userID;
$qry = "SELECT USER_ID FROM motorist WHERE MOBILE_ID = '$mobileId';";

$result=mysql_query($qry) or die('THERE WAS AN ERROR IN GETTING THE
DATA');

while($data=mysql_fetch_array($result, MYSQL_ASSOC)){
      $userID = $data['USER_ID'];
}

echo "userId: $userID \n ";


if($flag == 1){
      echo "yes";
}else if($flag == 1){
      echo "no";
}

//insert to database

$qry = "INSERT INTO user_verified_report (USER_ID,REPORT_ID,FLAG_UP)
            VALUES ('$userID','$reportId',$flag);";

$result=mysql_query($qry) or die('THERE WAS AN ERROR IN GETTING THE DATA
user_verified_report');

//update report
$qry = "UPDATE report
      SET END_DATETIME = ADDDATE(NOW(),INTERVAL 1 HOUR)
      WHERE REPORT_ID = ".$reportId."
;";
```

```
$result=mysql_query($qry) or die('THERE WAS AN ERROR IN GETTING THE DATA
update report');

?>
<?php
      echo hello;
?>
<?php
require ("tools.php");

//echo "Hi from server! \n";

//$temp =
computeDistanceLatLong(14.4553101,120.932313,14.450481,120.932205);

//echo $temp;

//isInsideRectangle(2,2,5,2,2,5,5,5,6,1);
//isInsideRectangle(14.449956,120.930537,14.449873,120.932361,14.451452,12
0.930719,14.451473,120.932382,14.450579,120.933498);
//isInsideRectangle(14.449956,120.930537,14.449873,120.932361,14.451452,12
0.930719,14.451473,120.932382,14.450953,120.931524);
$result =
isInsideRectangle(14.449956,120.930537,14.449873,120.932361,14.451452,120.
930719,14.451473,120.932382,14.450953,120.931524);
if($result == true){
      echo 'true';
}else{
      echo 'false';
}
?>
<?php


//compute the distance between 2 points in meters
function computeDistanceLatLong($lat1, $long1, $lat2, $long2){
      $R = 6371; // Radius of the earth in km
      $dLat = deg2rad($lat2-$lat1);  // deg2rad below
      $dLon = deg2rad($long2-$long1);
      $a =
            sin($dLat/2) * sin($dLat/2) +
            cos(deg2rad($lat1)) * cos(deg2rad($lat2)) *
            sin($dLon/2) * sin($dLon/2);

      $c = 2 * atan2(sqrt($a), sqrt(1-$a));
      $d = ($R * $c)*1000; // Distance in m
      return $d;
}


//check if inside rectangle
function isInsideRectangle($LLX,$LLY,$LRX,$LRY,$ULX,$ULY,$URX,$URY,$x,$y){
      $LOWERBOUND = -1;
      $UPPERBOUND = -1;
```

```
$LEFTBOUND = -1;
$RIGHTBOUND = -1;

//check lower bound
//$a = ($LLY-$LRY);

$a = -($LLY - $LRY);
$b = $LLX-$LRX;
$c = -($a*$LRX + $b*$LRY);
$d = $a*$x + $b*$y + $c;

if($d<0){
      $LOWERBOUND = 0;

}else{
      //return false;
      $LOWERBOUND = 1;
}

//check upper bound
$a = -($ULY - $URY);
$b = $ULX-$URX;
$c = -($a*$URX + $b*$URY);
$d = $a*$x + $b*$y + $c;
if($d<0){
      $UPPERBOUND = 1;

}else{
      //return false;
      $UPPERBOUND = 0;
}

//check left bound
$a = -($ULY - $LLY);
$b = $ULX-$LLX;
$c = -($a*$LLX + $b*$LLY);
$d = $a*$x + $b*$y + $c;
if($d<0){
      $LEFTBOUND = 0;

}else{
      //return false;
      $LEFTBOUND = 1;
}

//check right bound
$a = -($URY - $LRY);
$b = $URX-$LRX;
$c = -($a*$LRX + $b*$LRY);
$d = $a*$x + $b*$y + $c;
if($d<0){
      $RIGHTBOUND = 1;

}else{
```

```
            //return false;
            $RIGHTBOUND = 0;
      }

      if($LOWERBOUND==1&&$UPPERBOUND==1&&$LEFTBOUND==1&&$RIGHTBOUND==1){
            return true;
      }else{
            return false;
      }
}


?>
<?php
      require("dbConfig.php");



      //get file ext
      $ImageType = @explode('/', $_FILES['media']['type']);

      //create unique name
       $randName = md5(time());
       $newName = $randName.'.'. $ImageType[1];
       //upload dir
       $target = 'images/';
       $target = $target . $newName;
      //move the file
      if(move_uploaded_file($_FILES['media']['tmp_name'], $target)){
            echo 'success';
            exit();
      }else{
            echo 'fail';
      }

?>
/*
* A master detail view, utilizing a native table view component and
platform-specific UI and navigation.
* A starting point for a navigation-based application with hierarchical
data, or a stack of windows.
* Requires Titanium Mobile SDK 1.8.0+.
*
* In app.js, we generally take care of a few things:
* - Bootstrap the application with any data we need
* - Check for dependencies like device type, platform version or network
connection
* - Require and open our top-level UI component
*
*/

//bootstrap and check dependencies
if (Ti.version < 1.8 ) {
```

```
	alert('Sorry - this application template requires Titanium Mobile
SDK 1.8 or later');
}

// This is a single context application with mutliple windows in a stack
(function() {
	//determine platform and form factor and render approproate
components
	var osname = Ti.Platform.osname,
		version = Ti.Platform.version,
		height = Ti.Platform.displayCaps.platformHeight,
		width = Ti.Platform.displayCaps.platformWidth;

	//considering tablet to have one dimension over 900px - this is
imperfect, so you should feel free to decide
	//yourself what you consider a tablet form factor for android
	var isTablet = osname === 'ipad' || (osname === 'android' && (width
> 899 || height > 899));

	var Window;
	if (isTablet) {
		Window = require('ui/tablet/ApplicationWindow');
		Ti.Gesture.addEventListener('orientationchange', function(e) {

	Ti.Android.currentActivity.setRequestedOrientation(Ti.Android.SCREEN
_ORIENTATION_LANDSCAPE);
	  });
	}
	else {
		// iPhone and Mobile Web make use of the platform-specific
navigation controller,
		// all other platforms follow a similar UI pattern
		if (osname === 'iphone') {
			Window = require('ui/handheld/ios/ApplicationWindow');
		}
		else if (osname == 'mobileweb') {
			Window =
require('ui/handheld/mobileweb/ApplicationWindow');
		}
		else {
			Window =
require('ui/handheld/android/ApplicationWindow');
			Ti.Gesture.addEventListener('orientationchange',
function(e) {

Ti.Android.currentActivity.setRequestedOrientation(Ti.Android.SCREEN_ORIEN
TATION_LANDSCAPE);
		 });

		}
	}
	new Window().open();
})();
```

```
exports.ip = "http://agila.upm.edu.ph/~mfmagpili/sp/";
var latitude;
var longitude;

function displayConsole(msg){
      Titanium.API.warn('GPSComponent: ' + msg);
}

exports.initGPSComponent =  function(){

      //check geolocation accessibility
      /*
      if (Titanium.Geolocation.locationServicesEnabled === false)
      {
            Titanium.UI.createAlertDialog({title:'TransitApp',
message:'Your device has geo turned off - turn it on.'}).show();
      }else{
            if(Titanium.Geolocation.locationServicesAuthorization ==
Titanium.Geolocation.AUTHORIZATION_RESTRICTED) {
                  Ti.UI.createAlertDialog({
                        title:'TransitApp',
                        message:'Your system has disallowed Titanium from
running geolocation services.'
                  }).show();
            }
      }
      */

      longitude = 1;
      latitude = 2;
      /*
      Titanium.Geolocation.accuracy = Titanium.Geolocation.ACCURACY_BEST;
      Titanium.Geolocation.distanceFilter = 10;



      Titanium.Geolocation.getCurrentPosition(function(e){

            if (!e.success || e.error){
                  currentLocation.text = 'error: ' +
JSON.stringify(e.error);
                  Ti.API.info("Code translation:
"+translateErrorCode(e.code));
                  alert('error ' + JSON.stringify(e.error));
                  Ti.UI.createAlertDialog({
                        title:'Kitchen Sink',
                        message:'error ' + JSON.stringify(e.error)
                  }).show();

                  return;
            }

            longitude = e.coords.longitude;
            latitude = e.coords.latitude;
```

```
            var altitude = e.coords.altitude;
            var heading = e.coords.heading;
            var accuracy = e.coords.accuracy;
            var speed = e.coords.speed;
            var timestamp = e.coords.timestamp;

            Ti.UI.createAlertDialog({
                  title:'Kitchen Sink',
                  message:'longitude: ' + longitude + 'latitude: ' +
e.coords.latitude
            }).show();

            Titanium.API.info('Date: ' + new Date(timestamp) + ' long: ' +
longitude + ' lat: ' + latitude);


      });
      */
};

function setLatitude(value){
      this.latitude = value;
}

function setLongitude(value){
      this.longitude = value;
}


exports.init = function(map){
      Titanium.Geolocation.getCurrentPosition(function(e){
            this.latitude = '' + e.coords.latitude;
            this.longitude = '' + e.coords.longitude;
            exports.longitude = e.coords.longitude;
            exports.latitude = e.coords.latitude;

            map.latitude = e.coords.latitude;
            map.longitude = e.coords.longitude;

            /*
            Ti.UI.createAlertDialog({
                  title:'Kitchen Sink',
                  message:'longitude: ' + this.longitude + ' latitude: ' +
e.coords.latitude
            }).show();
             */
      });
};


exports.getLatitude =  function(){
      return this.latitude;
};
```

```
exports.getLongitude =  function(){
	return this.longitude;
};

exports.getLatLon = function(){
	var lat;
	var lon;
	var coordinate = new Object();
	Titanium.Geolocation.getCurrentPosition(function(e){
		this.latitude = '' + e.coords.latitude;
		this.longitude = '' + e.coords.longitude;
		exports.longitude = e.coords.longitude;
		exports.latitude = e.coords.latitude;

		coordinate.latitude = e.coords.latitude;
		coordinate.longitude = e.coords.longitude;

		//mapview.latitude = e.coords.latitude;
		//mapview.longitude = e.coords.longitude;

	});



	//displayConsole('latitude:' + coordinate.latitude);
	//displayConsole('longitude:' + coordinate.longitude);
	return coordinate;

};
user.id = 'empty';
//exports.user.lat = 14.4505957;
//exports.user.lng = 120.9314332;

exports.setId = function(anId){
	this.id = anId;
};

exports.getId = function(anId){
	return this.id;
};

function user(anId){
	this.id = anId;
};

module.exports.user;


var lbl = Ti.UI.createLabel({
	text:'Please select an item',
	height:'auto',
	width:'auto',
	color:'#000',
	font: {
```

```
                fontSize: 40
      }
});

var loadIndicator = Ti.UI.createActivityIndicator({
    message: 'loading data...',
    font : 'Arial',
    color: '#FFF'
});

var self = Ti.UI.createView({
            width : 'auto',
            backgroundColor : 'white',
            layout : 'vertical'
});

function displayConsole(msg){
      Ti.API.warn('DetailView: ' + msg);
}

function createViewEchoTest(){


      var db = require('model/database');
      var request = Titanium.Network.createHTTPClient();
      var url = db.ip +'index.php?r=example/test';
      request.open("POST", url);
      request.send({
            id : Titanium.Platform.id,
            category : 'temp',
            lat : 'lat',
            longitude : 'long',

      });

      request.onload = function(){
            lbl.text = request.responseText;
      };
}

function createViewTestUser(){


      var user = require('model/user');

      var txt = 'user  :'  + user.getId()  + '\n';

      lbl.text = txt;
}

function createViewMap(){
      lbl.text = 'loading data';
      var db = require('model/database');
      var request = Titanium.Network.createHTTPClient();
```

```
	var url = db.ip +'index.php?r=raReport/GetAllRaReportsMobile';
	var user = require('model/user');
	request.open("POST", url);
	request.send({
		id : user.getId(),
	});

	request.onload = function(){
		var user = require('model/user');
		//alert('server responded \n lat: ' + user.lat);
		displayConsole(this.responseText);
		var json = JSON.parse(this.responseText);
		var annotation;
		var gps = require('model/GPSComponent');
		var userCoordinate = gps.getLatLon();
		var Map = require('ti.map');
		var mapview = Map.createView({
			userLocation: true,
			animate: true,
			regionFit: true,
			mapType: Map.NORMAL_TYPE,
			pitchEnabled: true,
			region: {
				latitude: userCoordinate.latitude,
				longitude: userCoordinate.longitude,
				latitudeDelta:0.01,
				longitudeDelta:0.01
			},
			camera: {
				centerCoordinate: {
					latitude: userCoordinate.latitude,
					longitude: userCoordinate.longitude
				}
			}
		});

		for(i=0; i < json.length; i++){
			annotation = Map.createAnnotation({
					animate:true,
					pincolor: Map.ANNOTATION_BLUE
			});
			displayConsole(json[i].id + ', ' + json[i].lat + ', ' +
json[i].lng);
			annotation.subtitle = json[i].classification;
			annotation.latitude = json[i].lat;
			annotation.longitude = json[i].lng;
			annotation.title = json[i].type;
			mapview.addAnnotation(annotation);
		}
		self.removeAllChildren();
		self.add(mapview);
	};
}
```

```
function updateRaReportStatus(statusName,reportId){
	//alert(statusName);

	var db = require('model/database');
	var request = Titanium.Network.createHTTPClient();
	var url = db.ip +'index.php?r=RaReport/UpdateRaReportStatusMobile';
	request.open("POST", url);
	request.send({
		id : reportId,
		statusName : statusName,
	});

	request.onload = function(){
		//alert(request.responseText);

		var json = JSON.parse(request.responseText);
		getRaReportDetails(reportId);
		alert('Updated Successfully');
		//alert('id: ' + json.id + '\n' + 'status: ' + json.status +
'\n');
		//createViewRaReportDetials(json);
		//lbl.text = request.responseText;


	};
}

function createViewRaReportDetials(data){
	//alert('id: ' + data.id);

	var reportDetailId = Ti.UI.createLabel({
		text: 'Report Id: ' + data.id,
		height:'auto',
		width:'auto',
		color:'#000',
		font: {
			fontSize: 40
		}
	});

	var reportDetailAddress = Ti.UI.createLabel({
		text:'Address: ' + data.address,
		height:'auto',
		width:'auto',
		color:'#000',
		font: {
			fontSize: 40
		}
	});

	var reportDetailCategory = Ti.UI.createLabel({
		text:'Category: ' + data.category,
		height:'auto',
		width:'auto',
```

```
	color:'#000',
	font: {
		fontSize: 40
	}
});

var reportDetailStatus = Ti.UI.createLabel({
	text:'Status: ' + data.status,
	height:'auto',
	width:'auto',
	color:'#000',
	font: {
		fontSize: 40
	}
});

var forms = require('ui/common/forms');

var fields = [
	{ title:'Update Status', type:'picker', id:'state', data: [
		'UNATTENDED', 'RESPONDING', 'RESOLVED'
	] },
	{ title:'Submit', type:'submit', id:'updateRaReportStatus' }
];

var form = forms.createForm({
	style: forms.STYLE_HINT,
	fields: fields
});
form.addEventListener('updateRaReportStatus', function(e) {
	updateRaReportStatus(e.values.state, data.id);
	//alert();
});


/*
var picker = Ti.UI.createPicker({
  top:50
});

var data = [];
data[0]=Ti.UI.createPickerRow({title:'UNATTENDED'});
data[1]=Ti.UI.createPickerRow({title:'RESPONDING'});
data[2]=Ti.UI.createPickerRow({title:'RESOLVED'});

picker.add(data);
picker.selectionIndicator = true;

var button = Ti.UI.createButton({
	title: 'submit',
	height: '40dp',
	width: '100dp',
	top:'10dp'
});
```

```
	button.addEventListener('click', function(e) {
		alert(picker.getSelectedRow(0).title);
	});
	*/
	self.removeAllChildren();


	self.add(reportDetailId);
	//self.add(reportDetailAddress);
	self.add(reportDetailCategory);
	self.add(reportDetailStatus);
	self.add(form);
	/*
	self.add(picker);
	self.add(button);
	*/

}

function getRaReportDetails(id){
	//alert('id: ' + id);
	var db = require('model/database');
	var request = Titanium.Network.createHTTPClient();
	var url = db.ip +'index.php?r=RaReport/GetRaReportDetails';
	request.open("POST", url);
	request.send({
		id : id,
	});

	request.onload = function(){
		var json = JSON.parse(request.responseText);
		//alert('id: ' + json.id + '\n' + 'status: ' + json.status +
'\n');
		createViewRaReportDetials(json);
		//lbl.text = request.responseText;
	};
}

function createViewRaList(){
	var db = require('model/database');
	var user = require('model/user');
	var request = Titanium.Network.createHTTPClient();
	var url = db.ip +'index.php?r=RaReport/GetRaReportsAssignedToUser';
	request.open("POST", url);
	request.send({
		id : user.getId(),

	});

	request.onload = function(){
		//lbl.text = request.responseText;
		var json = JSON.parse(request.responseText);
		var tableData = new Array();
```

```
            if(json.length == 0){
                  lbl.text = 'There are no road accidents assigned to
user';
                  self.removeAllChildren();
                  self.add(lbl);
            }else{
                  for(i=0; i < json.length; i++){
                        var temp =  Object();
                        temp.value = json[i].id;
                        temp.title = 'id: ' + json[i].id + '\ncategory:' +
json[i].status;
                        temp.color = '#806517';
                        temp.latitude = json[i].lat;
                        temp.longitude = json[i].lng;
                        temp.hasChild = false;
                        temp.backgroundColor = '#800517';
                        temp.font =  {
                              fontSize: 40
                        };
                        tableData.push(temp);
                  }

                  var table = Ti.UI.createTableView({
                        data:tableData
                  });

                  self.removeAllChildren();
                  self.add(table);

                  //add behavior
                  table.addEventListener('click', function(e) {
                        getRaReportDetails(e.rowData.value);
                  });
            }

      };
}

function displayCorrectView(e){
      self.removeAllChildren();
      self.add(lbl);

      //lbl.text = e.name+': $'+e.value;

      if(e.value == '1'){
            logout();
      }else if(e.value == '2'){
            //lbl.text = 'menu 2 selected';
            createViewRaList();
      }else if(e.value == '3'){
            //
            createViewMap();
      }else if(e.value == '4'){
            createViewEchoTest();
```

```
	}else if(e.value == '5'){
		createViewTestUser();
	}
};

function logout(){
	var db = require('model/database');
	var gps = require('model/GPSComponent');
	var userCoordinate = gps.getLatLon();
	var user = require('model/user');
	var request = Titanium.Network.createHTTPClient();
	var url = db.ip +'index.php?r=TrafficInvestigator/LogoutMobile';
	request.open("POST", url);
	request.send({
		id : user.id,
	});

	request.onload = function(){
		displayConsole(request.responseText);
		Ti.UI.currentWindow.close();
	};
}

function DetailView() {


	self.add(lbl);

	self.addEventListener('itemSelected', function(e) {
		displayCorrectView(e);
	});

	return self;
};

module.exports = DetailView;

// "Constants"
exports.STYLE_HINT = 'hint';
exports.STYLE_LABEL = 'label';

exports.TYPE_DATE = 'date';
exports.TYPE_EMAIL = 'email';
exports.TYPE_NUMBER = 'number';
exports.TYPE_PASSWORD = 'password';
exports.TYPE_PHONE = 'phone';
exports.TYPE_PICKER = 'picker';
exports.TYPE_TEXT = 'text';
exports.TYPE_SUBMIT = 'submit';

var isAndroid = Ti.Platform.osname === 'android';
var textFieldDefaults = {
	height: '40dp',
	width: '250dp',
```

```
	top: '10dp',
	color: '#CFF',
	borderStyle:Titanium.UI.INPUT_BORDERSTYLE_ROUNDED
};
var keyboardMap = {};
keyboardMap[exports.TYPE_EMAIL] = Ti.UI.KEYBOARD_EMAIL;
keyboardMap[exports.TYPE_NUMBER] = Ti.UI.KEYBOARD_NUMBER_PAD;
keyboardMap[exports.TYPE_PASSWORD] = Ti.UI.KEYBOARD_DEFAULT;
keyboardMap[exports.TYPE_PHONE] = Ti.UI.KEYBOARD_NUMBER_PAD;
keyboardMap[exports.TYPE_TEXT] = Ti.UI.KEYBOARD_DEFAULT;

var handleStyle = function(form, textField, title) {
	if (form.fieldStyle === exports.STYLE_HINT && textField) {
		textField.hintText = title;
	} else {
		form.container.add(Ti.UI.createLabel({
			text: title,
			top: '10dp',
			left: '35dp',
			color: '#222',
			font: {
				fontSize: '16dp',
				fontWeight: 'bold'
			},
			height: 'auto',
			width: 'auto'
		}));
		if (textField) {
			textField.top = '5dp';
		}
	}
};

var setupPickerTextField = function(textField, pickerType, data) {
	textField.editable = false;
	textField.rightButton = Ti.UI.createButton({
		style: Ti.UI.iPhone.SystemButton.DISCLOSURE,
		transform: Ti.UI.create2DMatrix().rotate(90)
	});
	textField.rightButtonMode = Ti.UI.INPUT_BUTTONMODE_ALWAYS;

	textField.addEventListener('focus', function(e) {
		e.source.blur();
		require('extensions/semiModalPicker').createSemiModalPicker({
			textField: textField,
			value: textField.value,
			type: pickerType,
			data: data
		}).open({animated:true});
	});
};

var ctr = 1;
var addField = function(field, fieldRefs) {
```

```
	var title = field.title || ('field' + ctr++);
	var id = field.id || title;
	var type = field.type || exports.TYPE_TEXT;
	var form = this;
	var fieldObject = undefined;

	if (type === exports.TYPE_TEXT ||
		type === exports.TYPE_EMAIL ||
		type === exports.TYPE_NUMBER ||
		type === exports.TYPE_PHONE ||
		type === exports.TYPE_PASSWORD) {
		fieldObject = Ti.UI.createTextField(textFieldDefaults);
		fieldObject.keyboardType = keyboardMap[type];
		fieldObject.passwordMask = type === exports.TYPE_PASSWORD;
		fieldObject.isPicker = false;
		handleStyle(form, fieldObject, title);
	} else if (type === exports.TYPE_DATE) {
		if (isAndroid) {
			fieldObject = Ti.UI.createPicker({
				type: Ti.UI.PICKER_TYPE_DATE
			});
			handleStyle(form, undefined, title);
		} else {
			fieldObject = Ti.UI.createTextField(textFieldDefaults);
			handleStyle(form, fieldObject, title);
			setupPickerTextField(fieldObject,
Ti.UI.PICKER_TYPE_DATE);
		}
	} else if (type === exports.TYPE_PICKER) {
		if (isAndroid) {
			fieldObject = Ti.UI.createPicker({
				type: Ti.UI.PICKER_TYPE_PLAIN,
				width: '250dp'
			});
			handleStyle(form, undefined, title);
			for (var i in field.data) {

	fieldObject.add(Ti.UI.createPickerRow({title:field.data[i]}));
			}
			fieldObject.isPicker = true;
		} else {
			fieldObject = Ti.UI.createTextField(textFieldDefaults);
			handleStyle(form, fieldObject, title);
			setupPickerTextField(fieldObject,
Ti.UI.PICKER_TYPE_PLAIN, field.data);
		}
	} else if (type === exports.TYPE_SUBMIT) {
		var button = Ti.UI.createButton({
			title: title,
			height: '40dp',
			width: '100dp',
			top:'10dp'
		});
		button.addEventListener('click', function(e) {
```

```
                var values = {};
                for (var i in fieldRefs) {

                      if(fieldRefs[i].isPicker){
                            values[i] =
fieldRefs[i].getSelectedRow(0).title;
                      }else{
                            values[i] = fieldRefs[i].value;
                      }


                }
                form.fireEvent(id, {values:values});
          });
          form.container.add(button);
     }

     // Add our prepared UI component to the form
     if (fieldObject) {
          //alert(fieldObject.isPicker);
          form.container.add(fieldObject);
          fieldRefs[id] = fieldObject;
     }
};

var addFields = function(fields, fieldRefs) {
     for (var i in fields) {
          this.addField(fields[i], fieldRefs);
     }
};

exports.createForm = function(o) {
     var container = Ti.UI.createView({
          layout: 'vertical',
          height: 'auto'
     });
     var fieldRefs = {};
     var form = Ti.UI.createView({
          contentHeight: 'auto',
          contentWidth: 'auto',
          showVerticalScrollIndicator:true,
          showHorizontalScrollIndicator:true,

          // new stuff
          container: container,
          fieldStyle: o.style || exports.STYLE_HINT,
          addField: addField,
          addFields: addFields
     });

     form.addFields(o.fields, fieldRefs);
     form.add(container);

     // Add this so each field can be accessed directly, if necessary
```

```
	form.fieldRefs = fieldRefs;

	return form;
};
//Master View Component Constructor
function MasterView() {
	//create object instance, parasitic subclass of Observable
	var self = Ti.UI.createView({
		backgroundColor:'black',
		width: '100%'
	});

	//some dummy data for our table view
	var tableData = [
		{title:'View Assigned Road Accident', value:'2',
hasChild:false, color: '#806517', backgroundColor:'#95B9C7', font:
{fontSize: 40},height: 125},
		{title:'View Road Accident Map', value:'3', hasChild:false,
color: '#806517', backgroundColor:'#95B9C7', font: {fontSize: 40},height:
125},
		{title:'Logout', value:'1', hasChild:false, color: '#806517',
backgroundColor:'#95B9C7', font: {fontSize: 40}, height: 125},
		//{title:'Test echo', value:'4', hasChild:false, color:
'#806517', backgroundColor:'#95B9C7'},
		//{title:'Test user', value:'5', hasChild:false, color:
'#806517', backgroundColor:'#95B9C7'	},
	];

	var table = Ti.UI.createTableView({
		data:tableData
	});
	self.add(table);

	//add behavior
	table.addEventListener('click', function(e) {
		self.fireEvent('itemSelected', {
			name:e.rowData.title,
			value:e.rowData.value
		});
	});

	return self;
};

module.exports = MasterView;
exports.isCheckingVTL = 0;
exports.isDataLocationCollectionSet = 0;
exports.currVTLid = 0;
exports.isDataLocationCollectionSet = 0;
exports.currDataLocationCollectionid = 0;

exports.SetIsCheckingVTLTrue = function(){
	var module = require('tools/appStatusObserver');
	module.isCheckingVTL = 1;
```

```
};

exports.SetIsCheckingVTLFalse = function(){
      var module = require('tools/appStatusObserver');
      module.isCheckingVTL = 0;
};
exports.isVerifying = 0;

function displayConsole(msg){
      Ti.API.warn('checkForNearbyReportsToVerify: ' + msg);
}

function checkForNearbyReportsToVerify() {
      //displayConsole('contacting server');
      var map_view = require('ui/common/baseui/map_view');
      var db = require('tools/db');


      var request = Titanium.Network.createHTTPClient();
      var url = db.url + "checkNearbyReportsToVerify.php";
      var gps = require('tools/GPSComponent');
      var userCoordinate = gps.getLatLon();
      //displayConsole('lat long: ' + userCoordinate.latitude + ',' +
userCoordinate.longitude);
      request.open("POST", url);
      request.send({
            mobileId: Titanium.Platform.id,
            lat : userCoordinate.latitude,
            longitude : userCoordinate.longitude,
      });

      request.onload = function() {
            //displayConsole('server response: ' + this.responseText);
            var json = JSON.parse(this.responseText);
            var verifyReport =
require('tools/checkForNearbyReportsToVerify');


            if(json.length == 0){
                  //alert('do nothing');
            }else if(verifyReport.isVerifying==0){

                  //alert(this.responseText);
                  verifyReport.isVerifying = 1;
                  var createUI =
require('ui/common/createVerifyReportWindow');
                  createUI.createVerifyReportWindow(json);

            }

      };

      request.onerror = function() {
            //alert("error");
```

```
	};
}



function selfCalling(){
	//displayConsole('loop per 1 sec');
	checkForNearbyReportsToVerify();
	setTimeout(selfCalling, 1000);
}

exports.schuduleCheckNearbyReport = function(){
	//displayConsole('starting schedule checknearby report');
	selfCalling();
};


function displayConsole(msg){
	Ti.API.warn('checkVTL: ' + msg);
}

function checkIfStillInVTL(){
	var appStatus = require('tools/appStatusObserver');
	if(appStatus.isDataLocationCollectionSet = 1&&
appStatus.currVTLid!=0){
		displayConsole('Checkin if still in VTL');
		var db = require('tools/db');
		var gps = require('tools/GPSComponent');
		var userCoordinate = gps.getLatLon();

		var request = Titanium.Network.createHTTPClient();
		var url = db.urlYii +
"index.php?r=VirtualTripLine/CheckIfInVTL";
		var appStatus = require('tools/appStatusObserver');
		request.open("POST", url);
		request.send({
			vtlid : appStatus.currVTLid,
			lat: userCoordinate.latitude,
			lon: userCoordinate.longitude,
		});

		request.onload = function(){
			displayConsole(this.responseText);
			if(this.responseText == 1){
				displayConsole('still in vtl');
				// do nothing
			}else{
				//outside vlt
				displayConsole('outside vtl');
				appStatus.isDataLocationCollectionSet = 0;
				appStatus.currDataLocationCollectionid = 0;
				appStatus.currVTLid = 0;
				appStatus.isCheckingVTL = 0;
```

```
                }

            };

            request.onerror = function(){
                //currWin.close();
                //alert("Sorry we couldn't process your request");
            };
      }
}


function createDataLocationCollection(){
      displayConsole("createDataLocationCollection started");

      var db = require('tools/db');
      var request = Titanium.Network.createHTTPClient();
      var url = db.urlYii +
"index.php?r=DataLocationCollection/CreateNewDataMobile";
      var appStatus = require('tools/appStatusObserver');
      displayConsole(appStatus.currVTLid);
      request.open("POST", url);
      request.send({
            mobileId : Titanium.Platform.id,
            vtlid : appStatus.currVTLid,
      });

      request.onload = function(){
            appStatus.isDataLocationCollectionSet = 1;
            appStatus.currDataLocationCollectionid = this.responseText;
            //Ti.API.warn('CreateRoadAccidentView: ' + this.responseText);
            //currWin.close();
            //alert("Thank you for the reporting a flooded road");
            displayConsole('createDataLocation' + this.responseText);
      };

      request.onerror = function(){
            //currWin.close();
            //alert("Sorry we couldn't process your request");
      };
}

function checkIfInsideVTL() {

      var appStatus = require('tools/appStatusObserver');
      displayConsole('check if inside vtl');
      if(appStatus.isCheckingVTL == 0){
            displayConsole('iscCheckingVTL = 0');
            var db = require('tools/db');
            var gps = require('tools/GPSComponent');
            var userCoordinate = gps.getLatLon();
            displayConsole('lat lon: ' + userCoordinate.latitude + ',' +
userCoordinate.longitude);
            var request = Titanium.Network.createHTTPClient();
```

```
            var url = db.url + "getVTL.php";
            request.open("POST", url);
            request.send({
                  mobileId: Titanium.Platform.id,
                  lat : userCoordinate.latitude,
                  longitude : userCoordinate.longitude,
            });

            request.onload = function() {
                        //alert(this.responseText);
                        displayConsole(this.responseText);
                        if(this.responseText == 0){
                              displayConsole('check if inside vtl false');
                        }else{
                              displayConsole('check if inside vtl true');
                              appStatus.isCheckingVTL = 1;
                              appStatus.SetIsCheckingVTLTrue();
                              var data = JSON.parse(this.responseText);
                              appStatus.currVTLid = data[0].vtlid;
                              createDataLocationCollection();
                        }



            };

            request.onerror = function() {
                  //alert("error");
            };
      }

}

function selfCalling(){
      displayConsole('loop checkVTL');
      checkIfInsideVTL();
      checkIfStillInVTL();
      setTimeout(selfCalling, 1000);
}

exports.schuduleCheckVTL = function(){
      displayConsole('starting VTL services');
      selfCalling();
};
function deg2rad(deg) {
  return deg * (Math.PI/180)
}

exports.computeDistance = function(lat1, long1, lat2, long2) {
      var R = 6371;
      // Radius of the earth in km
      var dLat = deg2rad(lat2 - lat1);
      // deg2rad below
      var dLon = deg2rad(long2 - long1);
```

```
      var a = Math.sin(dLat / 2) * Math.sin(dLat / 2) +
Math.cos(deg2rad(lat1)) * Math.cos(deg2rad(lat2)) * Math.sin(dLon / 2) *
Math.sin(dLon / 2);
      var c = 2 * Math.atan2(Math.sqrt(a), Math.sqrt(1 - a));
      var d = R * c* 1000;
      // Distance in km
      return d;
}
exports.url = "http://agila.upm.edu.ph/~mfmagpili/mobilehandler/";
exports.urlYii = "http://agila.upm.edu.ph/~mfmagpili/sp/";

exports.getRoadAccidentCategory = function() {
      var category = ['Head-on Collision', 'Rear Collision', 'Collision
with animals', 'Rollover', 'run-off road collision', 'car collision
involving pedestrian'];
      return category;
};

function showCamera(){
      Ti.Media.showCamera({
            success:function(event){
                  Ti.API.warn('camera: Our type was ' + event.mediaType);
                  image = event.media;
            },
            cancel:function(){

            },
            error:function(error){

            }


      });

}


exports.getFloodedRoadCategory = function() {

      var xhr = Ti.Network.createHTTPClient({
            // function called when the response data is available

            onload : function(e) {

                  lbl.text = this.responseText;
                  Ti.API.info("Received text: " + this.responseText);
                  alert('success');
                  var category = new Array();
                  var data = [];
                  var json = JSON.parse(this.responseText);
                  var jsnrows = json.rows;
                  var result = '';
```

```
                  category = ['Not Passable to All type of Vehicles', 'Not
Passable to Light Vehicle'];
                  return category;
            },

            // function called when an error occurs, including a timeout
            onerror : function(e) {
                  lbl.text = 'err: ' + e.error;
                  Ti.API.info(e.error);
                  alert('error');
            },

            timeout : 5000
      });

      xhr.open('GET',
'http://112.204.56.96/mobilehandler/getFloodedRoadCategory.php');
      xhr.send();

      //var category = ['Not Passable to All type of Vehicles', 'Not
Passable to Light Vehicle'];

};

exports.getReports = function() {

      var reportArray = [];
      var report = new Object();

      report.type = 'flood';
      report.latitude = 14.450514;
      report.longitude = 120.931445;

      reportArray.push(report);

      report = new Object();
      report.type = 'road accident';
      report.latitude = 14.450283;
      report.longitude = 120.931721;

      reportArray.push(report);

      report = new Object();
      report.type = 'road closed';
      report.latitude = 14.449909;
      report.longitude = 120.932411;

      reportArray.push(report);

      /*

       */
      return reportArray;
};
```

```
var latitude;
var longitude;

function displayConsole(msg){
      Titanium.API.warn('GPSComponent: ' + msg);
}

exports.initGPSComponent =  function(){

      //check geolocation accessibility
      /*
      if (Titanium.Geolocation.locationServicesEnabled === false)
      {
            Titanium.UI.createAlertDialog({title:'TransitApp',
message:'Your device has geo turned off - turn it on.'}).show();
      }else{
            if(Titanium.Geolocation.locationServicesAuthorization ==
Titanium.Geolocation.AUTHORIZATION_RESTRICTED) {
                  Ti.UI.createAlertDialog({
                        title:'TransitApp',
                        message:'Your system has disallowed Titanium from
running geolocation services.'
                  }).show();
            }
      }
      */

      longitude = 1;
      latitude = 2;
      /*
      Titanium.Geolocation.accuracy = Titanium.Geolocation.ACCURACY_BEST;
      Titanium.Geolocation.distanceFilter = 10;



      Titanium.Geolocation.getCurrentPosition(function(e){

            if (!e.success || e.error){
                  currentLocation.text = 'error: ' +
JSON.stringify(e.error);
                  Ti.API.info("Code translation:
"+translateErrorCode(e.code));
                  alert('error ' + JSON.stringify(e.error));
                  Ti.UI.createAlertDialog({
                        title:'Kitchen Sink',
                        message:'error ' + JSON.stringify(e.error)
                  }).show();

                  return;
            }

            longitude = e.coords.longitude;
            latitude = e.coords.latitude;
            var altitude = e.coords.altitude;
```

```
            var heading = e.coords.heading;
            var accuracy = e.coords.accuracy;
            var speed = e.coords.speed;
            var timestamp = e.coords.timestamp;

            Ti.UI.createAlertDialog({
                  title:'Kitchen Sink',
                  message:'longitude: ' + longitude + 'latitude: ' +
e.coords.latitude
            }).show();

            Titanium.API.info('Date: ' + new Date(timestamp) + ' long: ' +
longitude + ' lat: ' + latitude);


      });
      */
};

function setLatitude(value){
      this.latitude = value;
}

function setLongitude(value){
      this.longitude = value;
}


exports.init = function(map){
      Titanium.Geolocation.getCurrentPosition(function(e){
            this.latitude = '' + e.coords.latitude;
            this.longitude = '' + e.coords.longitude;
            exports.longitude = e.coords.longitude;
            exports.latitude = e.coords.latitude;

            map.latitude = e.coords.latitude;
            map.longitude = e.coords.longitude;

            /*
            Ti.UI.createAlertDialog({
                  title:'Kitchen Sink',
                  message:'longitude: ' + this.longitude + ' latitude: ' +
e.coords.latitude
            }).show();
             */
      });
};


exports.getLatitude =  function(){
      return this.latitude;
};

exports.getLongitude =  function(){
```

```
	return this.longitude;
};

exports.getLatLon = function(){
	var lat;
	var lon;
	var coordinate = new Object();
	Titanium.Geolocation.getCurrentPosition(function(e){
		this.latitude = '' + e.coords.latitude;
		this.longitude = '' + e.coords.longitude;
		exports.longitude = e.coords.longitude;
		exports.latitude = e.coords.latitude;

		coordinate.latitude = e.coords.latitude;
		coordinate.longitude = e.coords.longitude;

		//mapview.latitude = e.coords.latitude;
		//mapview.longitude = e.coords.longitude;

	});



	//displayConsole('latitude:' + coordinate.latitude);
	//displayConsole('longitude:' + coordinate.longitude);
	return coordinate;

};
var readingValues = new Array();
var currValues = new Object();
var currentLat;
var currentLon;
var currentSpeed;

//continuelly monitor GPS readings
var x = 0;// just to measure how many readings has been done

function displayConsole(msg){
	Ti.API.warn('speedReadingsModel: ' + msg);
}

function sendTrafficDataToServer(id,lat,lon,speed){
	displayConsole('sending traffic data');

	var db = require('tools/db');
	var request = Titanium.Network.createHTTPClient();
	var url = db.urlYii +
"index.php?r=TrafficLocationPoint/CreateNewDataMobile";
	var appStatus = require('tools/appStatusObserver');
	//displayConsole(appStatus.currVTLid);
	request.open("POST", url);
	request.send({
		id : id,
		latitude : lat,
```

```
            longitude : lon,
            speed : speed,
      });

      request.onload = function(){
            displayConsole('server respose: ' + this.responseText);
      };

      request.onerror = function(){
            //currWin.close();
            //alert("Sorry we couldn't process your request");
      };
}

exports.getGpsReadings = function(){
      // WARNING: This will only work well outside, where the phone can
get a good GPS signal
      // Set up the geolocation code
      displayConsole('gps readings');
      Ti.Geolocation.Android.manualMode = true;
      //Ti.Geolocation.accuracy = Ti.Geolocation.ACCURACY_HIGH;
      Ti.Geolocation.purpose = 'To track how far you have traveled!';
      var lastLocation = null, totalFtTraveled = 0;


      gpsProvider = Ti.Geolocation.Android.createLocationProvider({
            name : Ti.Geolocation.PROVIDER_GPS
      });
      Ti.Geolocation.Android.addLocationProvider(gpsProvider);

      var gpsRule = Ti.Geolocation.Android.createLocationRule({
    provider: Ti.Geolocation.PROVIDER_GPS,
    // Updates should be no older than 5m
    maxAge: 101,
    // But  no more frequent than once per 10 seconds
    minAge: 99
      });
      Ti.Geolocation.Android.addLocationRule(gpsRule);

      /**
       * This function is called by the phone every time we have new
geolocation data. It writes out where the user currently is.
       * @param e An argument given to us by the phone with information
such as accuracy, latitude, and longitude.
       */
      function reportPosition(e) {
            if (!e.success || e.error) {
                  //label.text = 'error: ' + JSON.stringify(e.error);
            } else {
                  x = x + 1;
                  if (lastLocation != null) {
                        var lat1 = lastLocation.latitude, lon1 =
lastLocation.longitude;
```

```
                        var lat2 = e.coords.latitude, lon2 =
e.coords.longitude;
                        var kmTraveled = 3963.0 * Math.acos(Math.sin(lat1 /
57.2958) * Math.sin(lat2 / 57.2958) + Math.cos(lat1 / 57.2958) *
Math.cos(lat2 / 57.2958) * Math.cos(lon2 / 57.2958 - lon1 / 57.2958));
                        var ftTraveled = kmTraveled * 3280.8399;
                        totalFtTraveled += ftTraveled;
                  }
                  var speed = parseFloat((e.coords.speed*3.6)).toFixed(2);
                  currValues.latitude = e.coords.latitude + '';
                  currValues.longitude = e.coords.longitude + '';
                  currValues.speed =
parseFloat((e.coords.speed*3.6)).toFixed(2) + '';
                  currentLon = 'longitude: ' + e.coords.longitude;
                  currentLat = 'latitude: ' + e.coords.latitude;
                  currentSpeed = x + ' : ' +
parseFloat((e.coords.speed*3.6)).toFixed(2) + "kph";
                  lastLocation = e.coords;

      insertReadingValues(e.coords.latitude,e.coords.longitude,parseFloat(
(e.coords.speed*3.6)).toFixed(2));

                  var appStatus = require('tools/appStatusObserver');

                  if(appStatus.isCheckingVTL == 1 &&
appStatus.isDataLocationCollectionSet == 1){
                        displayConsole('ready to send traffic data');

      sendTrafficDataToServer(appStatus.currDataLocationCollectionid,e.coo
rds.latitude,e.coords.longitude,speed);
                  }else{
                        displayConsole('not ready to send traffic data');
                  }


            }


            var readingValues = new Object();
            readingValues.lat = currentLat;
            readingValues.lon = currentLon;
            readingValues.speed = currentSpeed;

            Ti.App.fireEvent('updateGpsReadings',{data:readingValues});
      }

      // This will get the location right now, and will get the phone
ready to listen for the user's current location.
      Ti.Geolocation.getCurrentPosition(reportPosition);
      // And this fires depending on the settings.
      Ti.Geolocation.addEventListener('location', reportPosition);

};
```

```
exports.readingValuesLength = function(){
      return readingValues.length;
};

exports.clearValues = function(){
      readingValues = new Array();
};

exports.getReadingValue = function(index){
      return readingValues[index];
};

function insertReadingValues(latitude,longitude,speed){
      var temp = new Object();
      temp.latitude = latitude;
      temp.longitude = longitude;
      temp.speed = speed;
      readingValues.push(temp);
}

exports.computeAverageSpeed = function(){
      var total = 0;
      for(var i = 0; i < readingValues.length; i++){
            total = total + parseFloat(readingValues[i].speed);
      }

      var result = total/readingValues.length;

      return result;
};

exports.getCurrValues = function(){
      return this.currValues;
};
var self = Ti.UI.createTabGroup({
      //activeTabBackgroundColor:'#E41B17',
      //activeTabBackgroundDisabledColor:'#800517',
      navBarHidden:true,
      tabsAtBottom:true
});

var activityIndicator = Ti.UI.createActivityIndicator({
      width: 'auto',
      lenght: 'auto',
      message: 'checking configuration'
      //style: Titanium.UI.ActivityIndicatorStyle.BIG

});

var loadingWin = Ti.UI.createWindow({
      title:'Map',
      background:"#fff"
});
```

```
var loadingWinView = Ti.UI.createView({
      background:"#fff"
});


var LoadingTab = Ti.UI.createTab({
      title:'Checking Resources',
      window:loadingWin
});

function displayConsole(msg){
      Ti.API.warn('ApplicationTabGroup: ' + msg);
}

function StartMonitoringServices(){
      var verifyReport = require('tools/checkForNearbyReportsToVerify');
      verifyReport.schuduleCheckNearbyReport();

      var checkVtl = require('tools/checkVTL');
      checkVtl.schuduleCheckVTL();

      displayConsole('starting speed readings services');
      var speedReadingService = require('tools/speedReadingsModel');
      speedReadingService.getGpsReadings();
}

function timer(){

}

function displayMainWindow(){
      StartMonitoringServices();
      var win2 = Ti.UI.createWindow({
            title:'Map',
            background:"#fff"
      });

      var tab2 = Ti.UI.createTab({
            title:'Map',
            window:win2
      });

      var MapTab = require('MapTab');
      /*self.addTab(new MapTab());
      */
      loadingWinView.removeAllChildren();

      var map = new MapTab();
      loadingWinView.add(map);
      LoadingTab.title = 'Map';
      //loadingWin.remove(loadingWinView);
      var view = Ti.UI.createView({
            length:'100%',
            width:'100%',
```

```
            background:"#white"
      });

      var label = Ti.UI.createLabel({
            text:'Temp'
      });
      view.add(label);

      var ReportTab = require('ReportTab');
      self.addTab(new ReportTab());

      // Code for opening Report window
      var ReportTabController = require('ui/common/ReportTabController');

      self.addEventListener('app:reportSelected', function(e) {
            //create detail view container
            var currView;
            var detailContainerWindow = Ti.UI.createWindow({
                  title:'Product Details',
                  navBarHidden:false,
                  backgroundColor:'#ffffff'
            });

            if(e.title=='Flooded Street'){
                  var createFloodedReportView =
require('ui/common/createFloodedReportView');
                  curView = new createFloodedReportView();
            }
            detailContainerWindow.add(currView);
            detailView.fireEvent('itemSelected',e);
            detailContainerWindow.open();
      });
}

function checkIfGPSCoordinatesIsReady(){
      activityIndicator.message = 'checking if coordinates are ready...';
      var gps = require('tools/GPSComponent');
      var userCoordinate = gps.getLatLon();
      for(var i = 0; userCoordinate.latitude ==null && i < 600; i++){
            displayConsole('no coordinates yet: ' + i);
            setTimeout(function(){timer();}, 1000);
      }
      if(i<600){
            setTimeout(function(){displayMainWindow();}, 1000);
      }else{
            activityIndicator.message = 'failed to get gps
coordinates...';
            //restart app
      }


}

function registerUser(){
```

```
	var db = require('tools/db');
	var url = db.urlYii + "index.php?r=Motorist/RegisterUser";


	activityIndicator.message = 'register user to database';
	var xhr = Ti.Network.createHTTPClient({

		// function called when the response data is available
		onload : function(e) {
			//Ti.API.warn("Received text: " + this.responseText);
			if(this.responseText == 0){
				//fail to create user
			}else{
				setTimeout(function(){checkGeolocationServices();},
1000);
			}

			Ti.App.fireEvent('loadingScreenReady');

		},

		// function called when an error occurs, including a timeout
		onerror : function(e) {
			activityIndicator.message = 'could not register user';
		},

		timeout : 5000
	});

	var db = require('tools/db');

	xhr.open('POST', url);
	xhr.send({
		id: Titanium.Platform.id,
	});
}

function checkIfUserIsRegistered(){

	var db = require('tools/db');
	var url = db.urlYii +
"index.php?r=Motorist/CheckIfUserIsRegistered";


	activityIndicator.message = 'checking if user is already
registered';
	var xhr = Ti.Network.createHTTPClient({

		// function called when the response data is available
		onload : function(e) {
			//Ti.API.warn("Received text: " + this.responseText);
			if(this.responseText == 0){
				setTimeout(function(){registerUser();}, 1000);
			}else{
```

```
                        setTimeout(function(){checkGeolocationServices();},
1000);
                }

                Ti.App.fireEvent('loadingScreenReady');

            },

            // function called when an error occurs, including a timeout
            onerror : function(e) {
                activityIndicator.message = 'could not check if user is
registered';
            },

            timeout : 5000
      });

      var db = require('tools/db');

      xhr.open('POST', url);
      xhr.send({
            id: Titanium.Platform.id,
      });
}

function connectToServer(){
      displayConsole('connecting to server');
      activityIndicator.message = 'checking if server is online';
      var xhr = Ti.Network.createHTTPClient({

            // function called when the response data is available
            onload : function(e) {
                //Ti.API.warn("Received text: " + this.responseText);
                Ti.App.fireEvent('loadingScreenReady');
                setTimeout(function(){checkIfUserIsRegistered();}, 1000);
                ;
            },

            // function called when an error occurs, including a timeout
            onerror : function(e) {
                activityIndicator.message = 'could not connect to
server';
            },

            timeout : 5000
      });

      var db = require('tools/db');

      xhr.open('GET', db.url + 'respond.php');
      xhr.send();
}

function checkNetworkState(){
```

```
	activityIndicator.message = 'checking network state...';
	if(Titanium.Network.networkType == Titanium.Network.NETWORK_NONE){
	  displayConsole('No Network');
	}else{
		displayConsole('Network Connection Available');
		setTimeout(function(){connectToServer();}, 1000);
	}
}

function checkGeolocationServices(){
	activityIndicator.message = 'checking geolocation services...';
	if(Ti.Geolocation.locationServicesEnabled == false){
		//
	}else{
		setTimeout(function(){checkIfGPSCoordinatesIsReady();}, 1000);
	}
}

function ApplicationTabGroup(Window) {
	//run monitoring services

	//create module instance

	self.addTab(LoadingTab);
	activityIndicator.show();

	loadingWin.add(loadingWinView);

	loadingWinView.add(activityIndicator);

	Ti.App.addEventListener('loadingScreenReady', function(e) {
		displayConsole('loading event fired');
	});

	checkNetworkState();
	return self;
};

module.exports = ApplicationTabGroup;

function displayConsole(msg){
	Ti.API.warn('createVerifyReportView: ' + msg);
}

exports.createVerifyReportWindow = function(reportData) {

	var db = require('tools/db');

	var detailContainerWindow = Ti.UI.createWindow({
		title : 'Report',
		navBarHidden : false,
		backgroundColor : '#ffffff'
	});
```

```
var self = Ti.UI.createView({
      width : 'auto',
      backgroundColor : 'white',
      layout : 'vertical'
});


var roadClosedQuestion = Ti.UI.createLabel({
      text : 'Is there a road closed up ahead',
      height : 'auto',
      width : 'auto',
      top : '5%',
      color : '#000',
      font: {
            fontSize: 40
      }
});

var floodedRoadQuestion = Ti.UI.createLabel({
      text : 'Is there a flooded road up ahead',
      height : 'auto',
      width : 'auto',
      top : '5%',
      color : '#000',
      font: {
            fontSize: 40
      }
});

var roadAccidentQuestion = Ti.UI.createLabel({
      text : 'Is there a road accident up ahead',
      height : 'auto',
      width : 'auto',
      top : '5%',
      color : '#000',
      font: {
            fontSize: 40
      }
});

var yes = Titanium.UI.createButton({
      title : 'Yes',
      width : '30%',
      top : '5%',
      height : 'auto'
});

var no = Titanium.UI.createButton({
      title : 'No',
      width : '30%',
      top : '5%',
      height : 'auto'
});
```

```
	//add even listeners to buttons


	yes.addEventListener('click', function(e) {

		var request = Titanium.Network.createHTTPClient();
		var url = db.url + "insertUserVerifiedReport.php";
		request.open("POST", url);
		request.send({
			reportId: reportData[0].id,
			mobileId : Titanium.Platform.id,
			flag : '1',
		});

		request.onload = function() {
			displayConsole(this.responseText);
			detailContainerWindow.close();
			var verifyReport =
require('tools/checkForNearbyReportsToVerify');
			verifyReport.isVerifying = 0;
			alert("Thank you for you feedback");
		};

		request.onerror = function() {
			alert("error");
		};
	});

	no.addEventListener('click', function(e) {
		var request = Titanium.Network.createHTTPClient();
		var url = db.url + "insertUserVerifiedReport.php";
		request.open("POST", url);
		request.send({
			reportId: reportData[0].id,
			mobileId : Titanium.Platform.id,
			flag : '0',
		});

		request.onload = function() {
			displayConsole(this.responseText);
			detailContainerWindow.close();
			var verifyReport =
require('tools/checkForNearbyReportsToVerify');
			verifyReport.isVerifying = 0;
			alert("Thank you for you feedback");
		};

		request.onerror = function() {
			alert("error");
		};

	});
```

```
	if(reportData[0].type == 'flooded street'){
		self.add(floodedRoadQuestion);
	}else if(reportData[0].type == 'road accident'){
		self.add(roadAccidentQuestion);
	}else if(reportData[0].type == 'road closed'){
		self.add(roadClosedQuestion);
	}else{
		alert('error at class createVerifyReportWindow');
	}


	self.add(yes);
	self.add(no);
	detailContainerWindow.add(self);
	detailContainerWindow.open();

};
var Map = require('ti.map');
var gps = require('tools/GPSComponent');
var userCoordinate = gps.getLatLon();

function displayConsole(msg){
	Titanium.API.warn('MapTab: ' + msg);
}


var mapview = Map.createView({
		userLocation: true,
		animate: true,
		regionFit: true,
		mapType: Map.NORMAL_TYPE,
		pitchEnabled: true,
		region: {
			latitude: userCoordinate.latitude,
			longitude: userCoordinate.longitude,
			latitudeDelta:0.01,
			longitudeDelta:0.01
		},
		camera: {
			centerCoordinate: {
				latitude: userCoordinate.latitude,
				longitude: userCoordinate.longitude
			}
		}
});

function getTrafficData(map){
	var map_view = require('ui/common/baseui/map_view');
	var db = require('tools/db');
	var gps = require('tools/GPSComponent');
	var userCoordinate = gps.getLatLon();
	var request = Titanium.Network.createHTTPClient();
```

```
	var url = db.urlYii +
"index.php?r=VirtualTripLine/GetTrafficDataMobile";
	request.open("POST", url);
	request.send({

	});

	request.onload = function() {
		displayConsole(this.responseText);
		var json = JSON.parse(this.responseText);
		for(i = 0; i < json.length; i++){

			var MapModule = require('ti.map');

			var pointsArray = new Array();
			var temp = new Object();
			temp.latitude = json[i].marker1_latitude;
			temp.longitude = json[i].marker1_longitude;
			pointsArray.push(temp);
			var temp = new Object();
			temp.latitude = json[i].marker2_latitude;
			temp.longitude = json[i].marker2_longitude;
			pointsArray.push(temp);
			var route = MapModule.createRoute({
				points: pointsArray
			});

			map.addRoute(route);

			var annotation = MapModule.createAnnotation({
						//image : '/images/river-2.png',
						pincolor: MapModule.ANNOTATION_ORANGE,
						draggable: true,
						latitude: json[i].marker1_latitude,
						longitude: json[i].marker1_longitude
			});
			if(json[i].averageSpeed!=null){
				annotation.title = 'Average Speed:' +
json[i].averageSpeed.toFixed(2);
			}else{
				annotation.title = 'Average Speed: No Data';
			}


			map.addAnnotation(annotation);
			var annotation = MapModule.createAnnotation({
						//image : '/images/river-2.png',
						pincolor: MapModule.ANNOTATION_ORANGE,
						draggable: true,
						latitude: json[i].marker2_latitude,
						longitude: json[i].marker2_longitude
			});
			map.addAnnotation(annotation);
			if(json[i].averageSpeed!=null){
```

```
                        annotation.title = 'Average Speed:' +
json[i].averageSpeed;
                  }else{
                        annotation.title = 'Average Speed: No Data';
                  }
            }

      };

      request.onerror = function() {
            alert("error on getting traffic data");
      };
}

function getReports(map) {
      var map_view = require('ui/common/baseui/map_view');
      var db = require('tools/db');

      var request = Titanium.Network.createHTTPClient();
      var url = db.url + "getReportsForMapAnnotations.php";
      request.open("POST", url);
      request.send({
            mobileId : Titanium.Platform.id,
            lat : userCoordinate.latitude,
            longitude : userCoordinate.longitude,
      });

      request.onload = function() {

            //alert(this.responseText);
            var MapModule = require('ti.map');
            var json = JSON.parse(this.responseText);
            var annotation;
            var annotationArray = new Array();

            for(i = 0; i < json.length; i++){

                  if(json[i].type == 'Flooded Street'){
                        annotation = MapModule.createAnnotation({
                              //image : '/images/river-2.png',
                              pincolor: MapModule.ANNOTATION_BLUE,
                              draggable: true
                        });
                  }else if(json[i].type == 'Road Accident'){
                        annotation = MapModule.createAnnotation({
                              //image : '/images/caraccudent.png',
                              pincolor: MapModule.ANNOTATION_RED,
                              draggable: true
                        });
                  }else{
                        annotation = MapModule.createAnnotation({
                              //image : '/images/closedroad.png',
                              pincolor: MapModule.ANNOTATION_GREEN ,
                              draggable: true
```

```
                        });
                  }

                  if(json[i].pictureId == null){
                        //Titanium.API.warn('MapTab: ' + 'getReport: ' +
'no picture');
                  }else{
                        if(json[i].id < 5){
                              var image = Ti.UI.createImageView({
                                    image: db.urlYii + 'images/' +
json[i].pictureId + '.jpg',
                                    width: '1%'
                              });

                              annotation.rightView = image;
                        }else{
                              var image = Ti.UI.createImageView({
                                    image: db.urlYii + 'images/' +
json[i].pictureId + '.jpeg',
                                    width: '1%'
                              });
                              annotation.rightView = image;
                        }

                        if(json[i].pictureId == null){
                        //Titanium.API.warn('MapTab: ' + 'getReport: ' +
'no picture');
                        }
                        //Titanium.API.warn('MapTab: ' + 'getReport: ' +
'picture available from' + db.urlYii + 'images/');
                  }

                  annotation.latitude = json[i].latitude;
                  annotation.longitude = json[i].longitude;
                  annotation.title = json[i].type;

                  if(json[i].isVerifiedBySystem == 1){
                        /*var image = Ti.UI.createImageView({
                                    image: db.urlYii +
'icons/verified.jpg',
                                    width: '10%'
                                    //image: db.urlYii + 'images/' +
json[i].pictureId + '.jpg',

                        });

                        annotation.leftView = image;
                        */
                        annotation.subtitle = json[i].classification +
'\nDate Created: ' + json[i].dateCreated + '\nReport has been verified by
the system';
                        //Titanium.API.warn('MapTab: ' + 'getReport: ' +
'no picture');
                  }else{
```

```
                        annotation.subtitle = json[i].classification +
'\nFlag Up: ' + json[i].flagUp + '\nFlag Down: ' + json[i].flagDown +
'\nDate Created: ' + json[i].dateCreated;
                  }

                  map.addAnnotation(annotation);
                  //annotationArray.push(annotation);

            }
            //map.annotation = annotationArray;
            /*
            var verifyReport =
require('tools/checkForNearbyReportsToVerify');

            if (json.length == 0) {
                  //alert('do nothing');
            } else if (verifyReport.isVerifying == 0) {

                  alert(this.responseText);
                  verifyReport.isVerifying = 1;
                  var createUI =
require('ui/common/createVerifyReportWindow');
                  createUI.createVerifyReportWindow(json);

            }
            */
      };

      request.onerror = function() {
            alert("error on getting map annotations");
      };
}

function selfCalling(){
      mapview.removeAllAnnotations();
      displayConsole('Updating Map');
      getReports(mapview);
      getTrafficData(mapview);
      setTimeout(selfCalling, 300000);
}

function scheduleMapUpdate(){

      selfCalling();
}


function MapTab() {
      //create module instance

      var win2 = Ti.UI.createWindow({
            title:'Map',
            background:"#fff"
      });
```

```
var view = Ti.UI.createView({
      background:"white"
});

var self = Ti.UI.createTab({
      title:'Map',
      window:win2
});

var label0 = Ti.UI.createLabel({
      text: "This is where the map will rise",
      color: "red"
});

win2.add(label0);

//create map view

var map_view = require('ui/common/baseui/map_view');
var mapviewMapTab = map_view.map;

var gps = require('tools/GPSComponent');
gps.init(mapviewMapTab);


mapviewMapTab.height = '100%';
mapviewMapTab.top = 0;
//alert("lat: " + map_view.map.longitude);


scheduleMapUpdate();
//get reports from database
//getReports(map);

view.add(mapview);
win2.add(view);
/*
var pointsArray = new Array();
var temp = new Object();
temp.latitude = 14.449819;
temp.longitude = 120.931484;
pointsArray.push(temp);
var temp = new Object();
temp.latitude = 14.449497;
temp.longitude = 120.932852;
pointsArray.push(temp);
var route = Map.createRoute({
      points: pointsArray
});

mapview.addRoute(route);
*/
return mapview;
```

```
};

module.exports = MapTab;
var detailContainerWindow = Ti.UI.createWindow({
      title : 'Report',
      navBarHidden : false,
      backgroundColor : '#ffffff'
});


function displayConsole(msg){
      Titanium.API.warn('MapTab: ' + msg);
}

function getDataFSCategory(){
      displayConsole('getting FS data');
      var xhr = Ti.Network.createHTTPClient({
            // function called when the response data is available

            onload : function(e) {
                  //Ti.API.info();
                  //alert('success');
                  displayConsole("Received text: " + this.responseText);
                  var category = new Array();
                  var data = new Array();
                  var json = JSON.parse(this.responseText);
                  var jsnrows = json.rows;
                  var result = '';

                  for ( i = 0; i < json.length; i++) {
                        category[i] = json[i].name + '';
                        result = result + json[i].name;
                        data[i] = Ti.UI.createPickerRow({
                              title : json[i].name + '',
                              value : json[i].id + ''
                        });
                  }

                  createFSRView(data);
            },

            // function called when an error occurs, including a timeout
            onerror : function(e) {
                  lbl.text = 'err: ' + e.error;
                  Ti.API.info(e.error);
                  alert('error');
            },

            timeout : 5000
      });

      var db = require('tools/db');

      xhr.open('GET', db.url + 'getFloodedRoadCategory.php');
```

```
	xhr.send();

	//alert('getDataFS end');
}

// create flooded street report form view
function createFSRView(data) {

	var createFloodedReportView =
require('ui/common/ReportViews/createFloodedReportView');
	var currView = new createFloodedReportView(data);

}

//get road accident categories
function getDataRACategory(){
	//alert('getDataFS');
	displayConsole('getting RA data');
	var xhr = Ti.Network.createHTTPClient({
		// function called when the response data is available

		onload : function(e) {
			displayConsole("Received text: " + this.responseText);
			//alert('success');

			var data = new Array();
			var json = JSON.parse(this.responseText);
			var jsnrows = json.rows;
			var result = '';

			for ( i = 0; i < json.length; i++) {
				result = result + json[i].name;
				data[i] = Ti.UI.createPickerRow({
					title : json[i].name + '',
					value : json[i].id + ''
				});
			}

			createRARView(data);
		},

		// function called when an error occurs, including a timeout
		onerror : function(e) {
			lbl.text = 'err: ' + e.error;
			Ti.API.info(e.error);
			alert('error');
		},

		timeout : 5000
	});


	var db = require('tools/db');
	xhr.open('GET', db.url + 'getRoadAccidentCategory.php');
```

```
	xhr.send();
}

function createRARView(data) {
	//alert('RAR View');
	var createRoadAccidentView =
require('ui/common/ReportViews/createRoadAccidentView');
	var currView = new createRoadAccidentView(data);

}

//create road accident report form view

function ReportTab() {
	//create module instance

	var win1 = Ti.UI.createWindow({
		layout: 'vertical',
		title:'Report',
		background:"#ffff"
	});

	var self = Ti.UI.createTab({
		title:'Report',
		window:win1
	});

	var div1 = Ti.UI.createView({
		height:'100%',
		layout: 'horizontal',
		backgroundColor:'white'
	});


		var div11 = Ti.UI.createView({
			layout:'vertical',
			width:'50%',
			backgroundColor:'white'
		});

			var image11 = Ti.UI.createImageView({
				top:'15%',
				width:'100%',
				image:'/images/AccidentMenu.gif'

			});
			var image21 = Ti.UI.createImageView({
				width:'100%',
				image:'/images/closedMenu.png'

			});

		div11.add(image11);
		div11.add(image21);
```

```
	var div12 = Ti.UI.createView({
		layout:'vertical',
		width:'50%',
		backgroundColor:'white'
	});

		var image12 = Ti.UI.createImageView({
			top:'15%',
			width:'100%',
			image:'/images/floodMenu.png'

		});
		var image22 = Ti.UI.createImageView({
			width:'100%',
			image:'/images/SettingsMenu.png'

		});

	div12.add(image12);
	div12.add(image22);

div1.add(div11);
div1.add(div12);

var div2 = Ti.UI.createView({
	height:'50%',
	layout: 'horizontal',
	backgroundColor:'white'
});


	var div21 = Ti.UI.createView({
		width:'50%',
		backgroundColor:'white'
	});


	var div22 = Ti.UI.createView({
		width:'50%',
		backgroundColor:'yellow'
	});

div2.add(div21);
div2.add(div21);
win1.add(div1);
win1.add(div2);

//Listeners
image11.addEventListener('click',function(e){
	getDataRACategory();
});

image12.addEventListener('click',function(e){
```

```
		getDataFSCategory();
	});

	image21.addEventListener('click',function(e){
		var createRoadClosedView =
require('ui/common/ReportViews/createRoadClosedView');
		var currView = new createRoadClosedView();
	});

	image22.addEventListener('click',function(e){
		var camera = require('ui/common/ReportViews/camera');
		var currView = new camera();
	});

	/*
	var tableData = [
				{title:'Flooded Street', hasChild:true, color:
'#000'},
				{title:'Road Accident', hasChild:true, color:
'#000'},
				{title:'Road Closed', hasChild:true, color:
'#000'},
				{title:'Remote Data', hasChild:true, color:
'#000'},
				{title:'Speedometer', hasChild:true, color:
'#000'},
				];


	var table = Ti.UI.createTableView({
		data:tableData
	});

	div.add(table);
	win1.add(div);

	table.addEventListener('click', function(e) {


		if(e.rowData.title=='Flooded Street'){

			getDataFSCategory();

		}else if(e.rowData.title=='Road Accident'){

			getDataRACategory();

		}else if(e.rowData.title=='Speedometer'){

			var createRoadHazardView =
require('ui/common/temp/speedometer');
			var currView = new createRoadHazardView();

		}else if(e.rowData.title=='Road Closed'){
```

```
                  var createRoadClosedView =
require('ui/common/ReportViews/createRoadClosedView');
                  var currView = new createRoadClosedView();
            }else if(e.rowData.title=='Remote Data'){

                  var remoteDataTest =
require('ui/common/temp/remoteDataTest');
                  var currView = new remoteDataTest();

            }else{
                  var lbl = Ti.UI.createLabel({
                        text:'/'+e.rowData.title+'/',
                        height:'auto',
                        width:'auto',
                        color:'#000'
                  });
                  detailContainerWindow.add(lbl);
                  detailContainerWindow.open();
            }


            self.fireEvent('app:reportSelected', {
                  name:e.rowData.title
            });
      });
      */
      return self;
};

module.exports = ReportTab;


var image;
var isImageUploaded = 0;

function displayConsole(msg){
      Titanium.API.warn('createFloodedReportView: ' + msg);
}


function showCamera(){
      Ti.Media.showCamera({
            success:function(event){
                  Ti.API.warn('camera: Our type was ' + event.mediaType);
                  //image = event.media;
                  isImageUploaded = 1;

                  var imageView = Titanium.UI.createImageView({
                   image:event.media,
                   width:256,
                   height:256
                });
                  image = imageView.toImage().media;
```

```
            },
            cancel:function(){

            },
            error:function(error){

            }


      });

}

function createFloodedReportView(data) {

      var currWin = Ti.UI.createWindow({
            title : 'Report a flooded road',
            navBarHidden : false,
            backgroundColor : '#ffffff'
      });

      var category = new Array();
      var db = require('tools/db');
      var opts;
      var dialog;

      var self = Ti.UI.createScrollView({
            width : 'auto',
            layout : 'vertical',
            backgroundColor : 'white'

      });

      var lbl = Ti.UI.createLabel({
            text : 'Flooded Report Underconstuction',
            height : 'auto',
            width : 'auto',
            top : '5%',
            color : '#000'
      });

      //Creating GPS Point View
      var createGPSPointView =
require('ui/common/ReportViews/createGPSPointsView');
      var gpsPointsView = new createGPSPointView();


      //CHoosing for category

      var buttonCategory = Titanium.UI.createButton({
            title : 'Choose Category',
            width : 'auto',
            top : '5%',
```

```
		height : 'auto'
	});

	var lebelCategory = Ti.UI.createLabel({
		text : 'Category: Please choose a category',
		height : 'auto',
		width : 'auto',
		color : '#000'
	});

	//create upload photo button;

	var buttonUpload = Titanium.UI.createButton({
		title : 'Upload Picture(Optional)',
		width : 'auto',
		top : '5%',
		height : 'auto'
	});

	//create submit button
	var buttonSubmit = Titanium.UI.createButton({
		title : 'Submit Report',
		width : 'auto',
		top : '5%',
		height : 'auto'
	});

	//eventlistners
	var picker = Ti.UI.createPicker({
		top : 50
	});

	picker.add(data);

	picker.selectionIndicator = true;

	picker.addEventListener('change', function(e) {
		//var dialog = Ti.UI.createOptionDialog(opts).show();
		//alert(picker.getSelectedRow(0).title);
	});

	var map_view = require('ui/common/baseui/map_view');

	buttonSubmit.addEventListener('click', function(e) {
		//var dialog = Ti.UI.createOptionDialog(opts).show();
		//alert('id: ' + Titanium.Platform.id + '\n' + 'category: ' +
picker.getSelectedRow(0).title + '\n' + 'latitude: ' +
map_view.map.latitude + '\n' + 'long: ' + map_view.map.latitude);

		var request = Titanium.Network.createHTTPClient();
		var url = db.urlYii +
"index.php?r=fsReport/CreateFsReportMobile";
		request.open("POST", url);
		request.send({
```

```
                media : image,
                id : Titanium.Platform.id,
                category : picker.getSelectedRow(0).title,
                lat : map_view.map.latitude,
                longitude : map_view.map.longitude,
                isImageUploaded : isImageUploaded,

          });

          request.onload = function() {
                //Ti.API.warn('CreateRoadAccidentView: ' +
this.responseText);
                currWin.close();
                if(this.responseText == 'report already exist'){
                      alert("Report already exist");
                }else{
                      alert("Thank you for the reporting a flooded
road");
                }
                displayConsole(this.responseText);
          };

          request.onerror = function(){
                currWin.close();
                alert("Sorry we couldn't process your request");
          };

     });

     buttonUpload.addEventListener('click', showCamera);

     currWin.addEventListener('close',function(e){
          isImageUploaded = 0;
     });

     //Add components to view
     self.add(gpsPointsView);
     self.add(picker);
     self.add(lebelCategory);
     self.add(buttonUpload);
     self.add(buttonSubmit);

     currWin.add(self);
     currWin.open();
};

module.exports = createFloodedReportView;
function createGPSPointsView() {

     //create and initialize GPSComponent to get lat and long values
     var longitudeValue = Ti.UI.createLabel({
          text:''
     });
     var longitudeValue = Ti.UI.createLabel({
```

```
      text:''
});

var self = Ti.UI.createView({
      height:'30%',
      backgroundColor:'white',
      layout:'vertical'
});

var map_view = require('ui/common/baseui/map_view');
/*var mapviewReport = map_view.map;

var gps = require('tools/GPSComponent');
gps.init(mapviewReport);

mapviewReport.height = '70%'
mapviewReport.removeAllAnnotations();
*/


var coordinateHeader = Ti.UI.createLabel({
      text:'Current location',
      height:'auto',
      width:'auto',
      color:'#000',
      top:10,
      font: {
            fontSize: 20,
            fontFamily: 'Helvetica',
            fontWeight: 'bold'

      },
      bold:'true'
});

//get and display longitude values



var longitudeLabel = Ti.UI.createLabel({
      text:'longitude: ' + map_view.map.longitude,
      height:'auto',
      width:'auto',
      top:10,
      font: {
            fontSize: 40,
            fontFamily: 'Helvetica'

      },
      color:'#000'
});


//Get and display latitude values
```

```
	var latitudeLabel = Ti.UI.createLabel({
		text:'latitude: ' + map_view.map.latitude,
		height:'auto',
		width:'auto',
		top:10,
		font: {
			fontSize: 40,
			fontFamily: 'Helvetica'

		},
		color:'#000'
	});
	//self.add(mapviewReport);
	self.add(coordinateHeader);
	self.add(longitudeLabel);
	self.add(latitudeLabel);

	return self;
};

module.exports = createGPSPointsView;
var image;
var isImageUploaded = 0;

function displayConsole(msg){
	Titanium.API.warn('createRoadAccidentView: ' + msg);
}

function showCamera(){
	Ti.Media.showCamera({
		success:function(event){
			Ti.API.warn('camera: Our type was ' + event.mediaType);
			image = event.media;
			isImageUploaded = 1;
		},
		cancel:function(){

		},
		error:function(error){

		}


	});

}

function createRoadAccidentView(data) {

	var currWin = Ti.UI.createWindow({
		title : 'Report a road accident',
```

```
            navBarHidden : false,
            backgroundColor : '#ffffff'
      });

      var db = require('tools/db');

      var self = Ti.UI.createView({
            width : 'auto',
            backgroundColor : 'white',
            layout : 'vertical'
      });

      // Creating GPS Point View
      var createGPSPointView =
require('ui/common/ReportViews/createGPSPointsView');

      var gpsPointsView = new createGPSPointView();

      // CHoosing for category
      // var categories = ['Head-on Collision', 'Rear Collision',
'Collision with
      // animals','Rollover','run-off road collision','car collision
involving
      // pedestrian'];

      var lebelCategory = Ti.UI.createLabel({
            text : 'Category: Please choose a category',
            height : 'auto',
            width : 'auto',
            color : '#000'
      });

      // create submit button
      var buttonSubmit = Titanium.UI.createButton({
            title : 'Submit Report',
            width : 'auto',
            top : '5%',
            height : 'auto'
      });

      var buttonUpload = Titanium.UI.createButton({
            title : 'Upload Picture(Optional)',
            width : 'auto',
            top : '5%',
            height : 'auto'
      });


      //picker for categories
      var picker = Ti.UI.createPicker({
            top : 50
      });
```

```
	picker.add(data);

	picker.selectionIndicator = true;

	picker.addEventListener('change', function(e) {
		//var dialog = Ti.UI.createOptionDialog(opts).show();
		//alert(picker.getSelectedRow(0).title);
	});

	//submit button function
	var map_view = require('ui/common/baseui/map_view');

	buttonSubmit.addEventListener('click', function(e) {
		/*alert(
			'id: ' +  Titanium.Platform.id + '\n' +
			'category: ' +  picker.getSelectedRow(0).title + '\n' +
			'latitude: ' +  map_view.map.latitude + '\n' +
			'long: ' +  map_view.map.longitude
		);*/

		var request = Titanium.Network.createHTTPClient();
		var url = db.urlYii +
"index.php?r=raReport/CreateRaReportMobile";
		request.open("POST", url);
		request.send({
			media : image,
			id : Titanium.Platform.id,
			category : picker.getSelectedRow(0).title,
			lat : map_view.map.latitude,
			longitude : map_view.map.longitude,
			isImageUploaded : isImageUploaded,

		});

		request.onload = function(){
			currWin.close();
			if(this.responseText == 'report already exist'){
				alert("Report already exist");
			}else{
				alert("Thank you for the reporting a road
accident");
			}
			displayConsole(this.responseText);
		};

		request.onerror = function(){
			currWin.close();
			alert("Sorry we couldn't process your request");
		};

	});

	buttonUpload.addEventListener('click', showCamera);
```

```
	// Add components to view
	self.add(gpsPointsView);
	self.add(lebelCategory);
	self.add(picker);
	self.add(buttonUpload);
	self.add(buttonSubmit);

	currWin.addEventListener('close',function(e){
		isImageUploaded = 0;
	});

	currWin.add(self);
	currWin.open({
		transition: Ti.UI.ANIMATION_CURVE_EASE_IN
	});

};

module.exports = createRoadAccidentView;
function createRoadClosedView() {
	var currWin = Ti.UI.createWindow({
		title : 'Report a road closed',
		navBarHidden : false,
		backgroundColor : '#ffffff'
	});


	var db = require('tools/db');

	var self = Ti.UI.createView({
		width : 'auto',
		backgroundColor : 'white',
		layout : 'vertical'
	});

	// Creating GPS Point View
	var createGPSPointView =
require('ui/common/ReportViews/createGPSPointsView');

	var gpsPointsView = new createGPSPointView();

	var buttonSubmit = Titanium.UI.createButton({
		title : 'Submit Report',
		width : 'auto',
		top : '5%',
		height : 'auto'
	});

	// Add components to view
	self.add(gpsPointsView);
	self.add(buttonSubmit);

	//submit button listener
	var map_view = require('ui/common/baseui/map_view');
```

```
	buttonSubmit.addEventListener('click', function(e) {

		var request = Titanium.Network.createHTTPClient();
		var url = db.urlYii +
"index.php?r=RoadClosedReport/CreateRcReportMobile";
		request.open("POST", url);
		request.send({
			id : Titanium.Platform.id,
			lat : map_view.map.latitude,
			longitude : map_view.map.longitude,
		});

		request.onload = function(){
			currWin.close();
			if(this.responseText == 'report already exist'){
				alert("Report already exist");
			}else{
				alert("Thank you for the reporting a road
accident");
			}
			displayConsole(this.responseText);
		};

		request.error = function(){
			alert("error");
		};
	});

	currWin.add(self);
	currWin.open();
};

module.exports = createRoadClosedView;
function createRoadHazardView() {

	var self = Ti.UI.createView({
		width:'auto',
		layout:'vertical',
		backgroundColor:'white'
	});

	var lbl = Ti.UI.createLabel({
		text:'Road Hazard Underconstuction',
		height:'auto',
		width:'auto',
		top:'5%',
		color:'#000'
	});

	var createGPSPointView =
require('ui/common/ReportViews/createGPSPointsView');

	var gpsPointsView = new createGPSPointView();
```

```
	self.add(gpsPointsView);
	self.add(lbl);

	return self;
};

module.exports = createRoadHazardView;

exports.generateLabel = function(){

}

exports.createCameraButton = function (){
	var button = Titanium.UI.createButton({
		title: 'Upload Picture',
		width: 'auto',
		top: '5%',
		height: 'auto'
	});

	return button;
}
var Map = require('ti.map');
var gps = require('tools/GPSComponent');
var userCoordinate = gps.getLatLon();

exports.map = Map.createView({
	top: 0,
	height: '70%',
	mapType: Titanium.Map.NORMAL_TYPE,

	region: {
			latitude: userCoordinate.latitude,
			longitude: userCoordinate.longitude,
			latitudeDelta:0.01,
			longitudeDelta:0.01
	},

		animate: true,
		regionFit: true,
		userLocation: true
});
```

XI. Acknowledgement

I would like to express my deepest gratitude to my family for supporting me throughout the whole process. Without them I would have likely cease my pursuit of finishing this requirement, the only and remaining requirement I need to finish my college degree in computer science. They have always been the source of my resilience, patience, and inspiration.

I would like thank one of the greatest proffessors in the University of the Philippines and my adviser Geoffrey Solano for guiding me during the creation of this project. Thank you for all the knowledge and experiences you shared with me. It is an honor and previledge to have you as one of my mentors. To the panel of both my defense and proposal, I would like to thank you for all the feedback that allowed the project to improve and move forward.

I would like to thank my mother for pursuing me to finish my degree in computer science. I've always noticed the compassion and hope you show towards your student. To actually be in the receiving side of that compassion is really remarkable specially the fact that I am your son.

To my super fabulous gorgoues sisters! Be careful what you wish for. I might just actually do it. I would also like to thank you for serving as my other advisers for this project. You helped me understand how to achieve my goals for this project. I would always remember "polishing the apple" where you exert 50% of you energy and time in improving your project by 5%. It would really help me manage my energy and time or have a really intelligent excuse to procrastinate.

Thank you for all the people who helped me finish this project.