UNIVERSITY OF THE PHILIPPINES MANILA

COLLEGE OF ARTS AND SCIENCES

DEPARTMENT OF PHYSICAL SCIENCES AND MATHEMATICS

ROX-RMS:

POINT OF SALE, INVENTORY MANAGEMENT, AND GIS-BASED REAL-TIME SALES MONITORING FOR RECREATIONAL OUTDOOR EXCHANGE (R.O.X)

A special problem in partial fulfillment

Of the requirements for the degree of

Bachelor of Science in Computer Science

Submitted by:

Roldan Real

May 2017

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

Available to the 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 "ROX-RMS: Point of Sale, Inventory Management, and GIS-Based Real-time Sales Monitoring for Recreational Outdoor eXchange (R.O.X.)" prepared and submitted by Roldan M. Real in partial fulfillment of the requirements for the degree of Bachelor of Science in Computer Science has been examined and is recommended for acceptance.

_____
Marvin John C. Ignacio, M.S. (candidate)
Adviser

EXAMINERS:

		Approved	**Disapproved**
1.	Gregorio B. Baes, Ph.D. (candidate)	_____	_____
2.	Avegail D. Carpio, M.S.	_____	_____
3.	Richard Bryann L. Chua, Ph.D. (candidate)	_____	_____
4.	Perlita E. Gasmen, M.S. (candidate)	_____	_____
5.	Ma. Sheila A. Magboo, M.S.	_____	_____
6.	Vincent Peter C. Magboo, M.D., M.S.	_____	_____

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

_____
Ma. Sheila A. Magboo, M.S.
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

_____
Leonardo R. Estacio Jr., Ph.D.
Dean
College of Arts and Sciences

Abstract

Recreational Outdoor eXchange (R.O.X.) is the biggest outdoor and recreation hub in the country, with eight branches located all over the Philippines. Currently each store's point of sale system is not integrated to each other, and without a central system that consolidates sales and other store transactions in real-time. This project aims to have create an integrated retail management system for the all the stores and to have real-time monitoring on the transactions done in the stores.

Keywords: real-time monitoring, retail management system, point of sale

Contents

List of Figures

List of Tables

I. Introduction

A. Background of the Study

Founded in 1985, the Primer Group of Companies is engaged in the retail and distribution of premium consumer brands in outdoor, travel, footwear, fashion, wellness and urban lifestyle. They have also diversified their portfolio into industrial products and services, venturing and creating a strong foothold in air-conditioning, ink manufacturing, creative design services, and silkscreen printing [1]. The Primer Group takes pride in developing unique retail concepts that define and respond to the ever evolving mood of the global retail landscape [2].

One of the many concept stores the company has developed is Recreational Outdoor eXchange, more commonly known as R.O.X. This concept store is the biggest outdoor sports and recreation hub in the Philippines – with three levels of all outdoor recreation gears and equipment for hiking, climbing, water sports, adventure travel, wellness, cycling, and scuba diving. They also organize adventure tour packages like hiking, camping, river rafting, kayaking, as wells as eco-tourism tours like bird-watching [3].

R.O.X. has eight stores all over the country [4]. These stores carry the biggest brands in outdoor recreation including The North Face, Columbia Sportswear, Mountain Hardware, Salomon, Sanuk, Osprey, Go Pro, Fox Racing, and many others [5].

R.O.X. has a central warehouse that dispatches all the items to be sold to each store. Each store is equipped with a point of sale system that does retail transactions. At the end of each business day, a report is generated by each store and send it to central office. The report contains the total amount of sales, number of invoices, total cash payments, and total debit and credit card payments. Cash

payments are deposited to a bank and debit and credit card payments are considered as account receivables [4].

Each store only sells items that are sent by the warehouse and maintains the inventory manually. Reconciliation of inventory is done at the end of each business day. Dispatching of items to the stores depends on how marketable the product is.

B. Statement of the Problem

Since each store only sends summarized reports to the central office at the end of each business day, reconciliation of inventory is also done at the end of the day. The reports lack other information like the time each transaction is made, thus no insights can be generated as to what time of the day the store is more profitable.

The point of sale system of each store is not integrated with each other. Each store cannot see the available items of other stores. This may affect profitability since customers will look for the items in another competing store instead of the store staff to recommend other branches where the item is available.

Since each store sends summary reports individually, consolidating and summarizing huge amount of data manually can lead to errors that translate to losses in money and time. Manual summarization of data is tedious and time-consuming. Doing daily, weekly, monthly, and yearly summaries could be more challenging when done manually.

When sales, transactions, and inventory data are not relayed in real-time, businessmen cannot formulate immediate business moves on how to improve the turn-out based on the summary of data received.

C. Objectives of the Study

General Objective:

The proposed system aims to provide Recreational Outdoor eXchange (R.O.X.) an alternative to traditional point of sale (POS) systems that lack important features. The system should have the features of a POS and should provide real-time reports about sales, transactions, and other information that help them see how business is doing in real time and in turn translates to better profitability.

Specific Objectives:

Different users have different access levels. Proprietor can

1. Log-in to the system
2. View/add/edit/enable/disable/change password system users
3. View/add/edit store branch
4. View/add/edit/ items
5. View/add/edit/search store items inventory
6. View sales and transaction reports for:
    (a) All store branches
    (b) An area
    (c) Specific store
        In between dates
7. Monitor the business in real-time based on:
    (a) Country (Summary of all stores in the country)
        i. Number of sales invoice
        ii. Total amount of sales
        iii. Top-selling items by:
            iii.i Quantity

iii.ii Amount

iv. Total number of Cash, Credit, Debit, and Voucher payments

v. Payment types

iii.i Number of Cash Payments

iii.ii. Number of Debit Card Payments

iii.iii. Number of Credit Card Payments

iii.iv. Number of Voucher Payments

(b) Area (Summary for all the stores in particular area)

i. Number of sales invoice

ii. Total amount of sales

iii. Top-selling items by:

iii.i Quantity

iii.ii Amount

iv. Total number of Cash, Credit, Debit, and Voucher payments

v. Payment types

iii.i Number of Cash Payments

iii.ii. Number of Debit Card Payments

iii.iii. Number of Credit Card Payments

iii.iv. Number of Voucher Payments

(c) Store (Under an area)

i. Number of sales invoice

ii. Total amount of sales

iii. Top-selling items by:

iii.i Quantity

iii.ii Amount

iv. Total number of Cash, Credit, Debit, and Voucher payments

v. Payment types

iii.i Number of Cash Payments

iii.ii. Number of Debit Card Payments

iii.iii. Number of Credit Card Payments

iii.iv. Number of Voucher Payments

Via an interactive map-based page which displays the geographical locations of each stores.

Store managers can

1. Log-in to the system
2. View/add/edit/enable/disable/change password store users
3. View/search available items inventory of other stores
4. Checkout items and print receipt
5. Return items and generate voucher

Store cashiers can

1. Log-in to the system
2. Checkout items and print receipt
3. Return items and generate voucher
4. View/search available items inventory of other stores

D. Significance of the Study

Real-time analytics helps the company to plan for better profitability of the business. Knowing the time of the day when low sales happen can help the business owners decide on what to do for that particular time of the day: have a promotion for that time of the day, limit the number of staff for the particular store, or minimize or set to low the usage of air-conditioning units and other electronic products.

Knowing the items available in another stores will help business profitability. When the customer asks for the item and is not available on the store where the customer is, store staff can search for that particular item in another branches and suggests to customer as to what branches the item is available.

The proposed system can serve as a highly useful and helpful tool for business owners, most especially, for specialty stores like R.O.X. The system is equipped with important features like POS, inventory management, and real-time monitoring and reporting, needed in most retail business.

The real-time monitoring page provides owners the opportunity to monitor the business from anywhere as long as connected to the internet. It provides a visual presentation of each area of the operations of one store and the whole country via an interactive map because it displays the area and location of the stores. Clicking the area (Metro Manila, Metro Cebu, etc.) displays the summary of sales, number of sales invoice, payment types, top-selling items per quantity, and top-selling items per amount for that particular area. Clicking the store displays information in real-time regarding its sales, number of sales invoice, payment types, top-selling items per quantity, and top-selling items per amount for that particular store.

While it may seem that the reference is only visual, the real time data offered is automatically consolidated which saves a large amount of time for business owners to evaluate how the business operations is going. With the substantial and immediate information being made available to the business owner, he/she can immediately attend to several issues on sales transactions, inventory management and retail management. In doing so, the real-time feedback of data and reports from each store and the map provide the owner the much needed response to strategize and strengthen his/her business.

Most importantly, the owner can also gauge the customers' behavior at a specific time of the day or specific week of the month. Consequently, a concept for a marketing plan targeting the low sales areas/stores can be drafted and put in motion to respond to weakening sales turn-over.

E. Scope and Limitation

Listed below are the scope and limitation of the proposed system:

1. The system is only intended for Recreational Outdoor eXchange, a specialty shop with specific target kind of customers but with high-ticket purchases.
2. It is substantially considered that real time monitoring and OpenStreet map rely on a stable and fast internet connection.
3. The transaction when it comes to the replenishment of stocks is done outside the system. This includes the coordination and ordering of supplies with the manufacturers or concessionaires.
4. Map coordinates per area (e.g. Luzon, Visayas, and Mindanao) are stored in a JSON file for faster reading. The store coordinates are stored in database.

5. Reports generated do not suggest to the proprietor on the next business move. Proprietor acts based on the data provided by the system.

F. Assumptions

Listed below are the assumptions of the proposed system:

1. Store has limited and identified customer base with high potential of purchases as this is strongly intended for specialty stores like R.O.X.
2. The use of the system is based on fast, reliable and strong internet connection.
3. ROX-RMS works on a physical store and not on any online shops.

II. Review of Related Literature

Small and Medium Sized Enterprises (SMEs) have a critical role in the economic development of various areas. Particularly, SMEs mainly constitute private businesses and are the major source of productivity growth, innovation, and employment generation. As discussed by Shaw (2012), SMEs are in a better position to exercise the best CSR practices, which is the basis for a more localized company structure. The more localized company structure is what promotes connectivity to the local communities, which leads to local employment generation. In particular, SMEs play a vital role in providing innovative solutions to various local economic problems. For instance, in the UK, SMEs have enhanced the accessibility of a healthy diet to local consumers (Shaw, 2012). According to Shaw (2012), the flexibility and innovativeness of SMEs promote tourist trade, which has a significant contribution to an area's GDP [6].

SMEs usually have an advantage over larger business corporations because the SMEs' management often has more knowledge of, and is closer to, the communities' needs. The constantly changing retail industry has diversified retail inventories over time, providing a wide range of services/products to the 'one-shop' shopper. As a result of the modern competitive retail environment that requires retail management to get to get the right product/service at the right place and time, retail managers are obligated to adapt to unique promotion, planning, merchandising and pricing techniques. The bases of modern SMEs retailing management are the mutually respectable retailers-suppliers partnerships and the Electronic Data Interface. Whereas the retailers-suppliers partnerships aim at enhancing retail operations efficiency and reliability, the Electronic Data Interface technology is part of the retail management system that targets SME retailers who seek to automate their store

operations including employees management, information security, marketing, customer management, inventory control/tracking as well as generation of customized reports (Aggarwal, 2009) [7].

ICT is widely utilized in many business organizations. Notably, Information and Communication Technology has provided new techniques for businesses to process, distribute, store and share information within their organization structures as well as with their stakeholders. Various studies have established that SMEs' adoption of ICT has a significant impact on business performance and marketing capabilities. The advent of the internet technology has not only enabled SMEs to venture into e-commerce but has also facilitated SMEs' effective marketing and management practices. Nonetheless, the adoption of Information Communication Technology among SMEs is barred by various technological factors such as complexity, observability, and trialability (Hartoyo, Daryanto & Arifin, 2015) [8]. Other factors that also influence SMEs' ICT adoption include owners' innovativeness/know-how, institutional intervention and competitive pressure. Similarly, technological platforms like e-commerce support and call centers have enabled retail companies to adapt to real-time business intelligence (Sahay & Ranjan, 2008) [9]. The e-commerce and call centers technological platforms are critical in obtaining timely analytical insights which lead to effectiveness in retail supply chain analytics/management. The availability of various inventory software in the concurrent technological business environment has enabled SMEs to conduct inventory management in the easiest and reliable manner possible [9].

The point of Sale systems are digitized cash registers that are traditionally designed to be used by retailers to ring up consumers' purchases. Often, the Point of Sale system data is used for marketing purposes. Besides, various time-consuming

administrative activities such as customer management, order tracking, stock control and ordering can be reduced using point of Sale (POS) system. POS systems are considered the electronic backbone of retail companies that facilitate chain digitization in value chains. According to Plomp, Rijn & Batenburg (2012), POS systems enable retail companies to adapt to automatic business-to-consumer and business-to-business collaboration, which enables effective retail management through timely sharing of business information with relevant stakeholders. Mainly, the POS system supports more retail management functionalities other than accounting including stock control, order tracking and customer management by establishing electronic business-to-consumer as well as business-to-business collaboration systems [10].

Every retail organization has warehouse unit(s) to stock merchandises. The sole purpose of keeping the stocks in warehouses is to ensure continuous replenishment of supply goods. Retailers keep a track of the stocked products and often make sure that there is surplus inventory to prevent being out of stock. In the retail industry, retailers are keen on maintaining the loyalty of every single consumer. In that case, they ensure effective inventory management to avoid leaving a negative impression on consumers as a result of empty shelves or unavailability of merchandise. During 'lead time' (the time needed for merchandise to reach retail store form suppliers' unit), retailers often ensure that they have ample stock to provide customers. Particularly, inventory management aid retailers to maintain supply during difficult times like transport strikes, curfews, and crises. Retailers can keep track of every merchandise in their warehouses by entering every Stock Keeping Unit (SKU) number in their master computers. The assignment of unique SKU numbers to products helps retailers to avoid unnecessary searches. With the advent of new technology, various software like the Vend software has been invented to aid retailers in automating their

inventory management. The software technology has eased different retail inventory management activities including quick creation of stocks by scanning products' barcodes, automatic generation of stocks whenever products beyond a certain customized threshold level, easy return of faulty/unsold merchandise to suppliers and automatic adjustment of inventories during product transfers. As articulated by Shah and Raykundaliya (2010), retailers should replenish smaller orders more often to avail sales promotional tool as a trade credit [11].

In the recent years, the implementation and design of Wireless Sensor Networks (WSN) have become a common area of research and application in businesses. The WSN constitute autonomous sensors that are designed to monitor environmental/physical conditions like pressure, temperature or sound and cooperatively pass their information/data through the network to the main location. The advantage of WSN is that it can be utilized with ease in an environment where a wired system cannot be used. In businesses, various types of WSN including Bluetooth, Wi-Fi, smart transducers, Personal Area Network as well as Winmax can be applied in remote monitoring. As elaborated by García et al. (2007) [12], the WSN technology is utilized in tracking the transportation of fruits in reefer containers along the European fruit supply chain. In that case, the WSN technology provides a real-time status update on the quality of fruits as they are being transported from point to point. On the other hand, Reddy and Sawant claim that the ZigBee WSN is used in controlling and monitoring the D.C motor parameters in industrial processes (2014) [13]. Moreover, as elaborated by Singhal and Gujra (2012), the Radio Frequency Identification (RFID) technology is used for remote, real-time monitoring of employee attendance in business organizations [14].

A GIS (Geographic information system) is designed to integrate data, software and hardware for capturing, analyzing, displaying and managing all kind of geographically referenced data/information. Mainly, the GIS allows retailers to visualize, interpret, question and understand data in different ways that reveal business patterns, trends as well as relationships in the form of charts, maps, globes and reports. According to Azaz (2011), various business organizations can integrate the GIS technology in their framework. Azaz argues that the GIS technology is contemporarily being used in many business management functions including facilities/sites management, marketing, logistics, planning as well as decision-making. Specifically, the GIS technology can assist retailers in locating the best site for building a warehouse. Alternatively, GIS can help the marketer to establish new prospects and identify geographical locations with many consumers. Ultimately, placing business data on a map using the GIS technology can enable retailers and business administrators to make informed decisions [15].

In a publication compiled by Smith, MacGregor and Johnson (2005), a method and a system for displaying and supporting product selection is described comprehensively. The system/method enables real-time monitoring of purchase transactions over the Internet using the World Wide Web. The method/system described by Smith, MacGregor and Johnson is designed to enable local organizations to have their merchandise displayed to customers on a computer monitor in a way that facilitates their identification by local consumers (2005). The system constitutes a search engine server that is designed to display search results in reference to coordinated distance from the customers. The search results are ranked in a specific way that entails product price, product availability and store location. With reference to retail store location, GPS coordinates are utilized to establish the distances from

the stores to consumers. In cases where the GPS coordinates are unavailable, a postal code database is used to determine postal codes that are adjacent to consumers (Smith, MacGregor & Johnson 2005). The sales transaction priorities are ordered in accordance to consumers' proximity to retail warehouses [16].

III. Theoretical Framework

Recreation Outdoor eXchange (R.O.X.)

The Philippines' top outdoor hub that offers world-class products and services. It also organizes activities that makes outdoor adventure easier and cheaper [3]. Currently has eight branches in the Philippines, R.O.X. carries the biggest names in outdoor sports and recreation, including: The North Face, Columbia Sportswear, Mountain Hardware, Salomon, Sanuk, Black Diamond, Osprey, Go Pro, among others.

Small and Medium-sized Enterprises (SMEs)

SMEs are defined by three keywords - small, single and local:

Small - SMEs are small in nature - either in terms of number of (a) employees - 10 persons for 'small' to 200 persons for 'medium', depending on the country's laws, (b) capital and assets - limited working capital and assets and (c) turnover - the overall turnover of the enterprise is small, compared to larger businesses.

Single - Most SMEs have a single owner who could also be the sole employee. While this may predominantly be the case, definitions set 250 to 500 employees as the limit for enterprises to be called an SME. The 'single' also refers to single products produced or service provided.

Local - SMEs are essentially local in nature - their market is usually localized to the area where they are located (same city, district or state); or may be 'local' in the sense that they operate from a place of residence - also called SOHO (Small Office Home Office) [17].

In the Philippines, SME is officially defined as any business activity or enterprise engaged in industry, agribusiness and/or services, whether single proprietorship, cooperative, partnership or corporation whose total assets, inclusive of those arising from loans but exclusive of the land on which the particular business entity's office,

plant and equipment are situated, must have value falling under the following categories:

Micro	**less than**	**P 1,500,001**
Small	**P1,500,001**	**P15,000,000**
Medium	**P15,000,001**	**P60,000,000**

However, the definition shall be subject to review and adjustment upon recommendation of sectorial organization(s) taking into account inflation and other economic indicators [18].

Point of Sale System

Also known as "point of purchase", it is the place where sales are made. On a macro level, a point of sale may be a mall, market or city. On a micro-level, retailers consider a point of sale to be the area surrounding the counter where customers pay [19].

A point-of-sale (POS) terminal is a computerized replacement for a cash register. Much more complex than the cash registers of even just a few years ago, the POS system can include the ability to record and track customer orders, process credit and debit cards, connect to other systems in a network, and manage inventory. Generally, a POS terminal has as its core a personal computer, which is provided with application-specific programs and I/O devices for the particular environment in which it will serve. A POS system for a restaurant, for example, is likely to have all menu items stored in a database that can be queried for information in a number of ways. POS terminals are used in most industries that have a point of sale such as a service desk, including restaurants, lodging, entertainment, and museums [20].

Browser-based web application

In a browser-based Web application, JavaScript instructions are contained within the Web page that is retrieved from a Web site. Combined with the HTML code that determines the visual layout and the CSS style sheet, the HTML, JavaScript and CSS are executed via the browser. In addition, processing at the server side is often widely performed to access databases and other networks. The data for a Web application may be stored locally or on the Web, or in both locations [21].

From a technical view-point, the web is a highly programmable environment that allows mass customization through the immediate deployment of a large and diverse range of applications, to millions of global users. Two important components of a modern website are flexible web browsers and web applications; both available to all and sundry at no expense [22].

Retail Management System

Retail management means running a store where merchandise is sold and Retail Management Information Systems include using hardware, software and procedures to manage activities like planning, inventory control, financial management, logistics and point of sale transactions [23].

It is essentially an integrated set of computerized applications that the retailer uses to operate their business. Retail management systems typically include Point of Sale (POS), Customer Relationship Management (CRM), Sales Order Management, Inventory Management, Purchasing & Receiving, Reporting and data driven Dashboard applications. Some offer e-commerce applications as part of their suite [24].

Inventory Management System

Inventory management is the process of efficiently overseeing the constant flow of units into and out of an existing inventory. This process usually involves controlling the transfer in of units in order to prevent the inventory from becoming too high, or dwindling to levels that could put the operation of the company into jeopardy [25]. It consists of business applications that track, manage and organize product sales, material purchases and other production processes [26].

Remote Monitoring and Management

Remote monitoring and management (RMM) is a collection of information technology tools that are loaded to client workstations and servers. These tools gather information regarding the applications and hardware operating in the client's location as well as supply activity reports to the IT service provider, allowing them to resolve any issues. RMM usually provides a set of IT management tools like trouble ticket tracking, remote desktop monitoring, support, and user information through a complete interface [27].

Geographic Information System (GIS)

A geographic information system (GIS) is a computer system for capturing, storing, checking, and displaying data related to positions on Earth's surface. GIS can show many different kinds of data on one map. This enables people to more easily see, analyze, and understand patterns and relationships [28]. The key word to this technology is Geography – this means that some portion of the data is spatial. In other words, data that is in some way referenced to locations on the earth [29]. GIS allows us to view, understand, question, interpret, and visualize our world in ways that reveal relationships, patterns, and trends in the form of maps, globes, reports, and charts [30].

Database Management Systems (RDBMS)

A DBMS is a system that enables the search and retrieval of information from a database. It controls how the data are stored and organized while addressing problems such as data integrity and security. It is used by an application that sends a request to which it responds by instructing the operating system to transfer the appropriate data.

A Relational Database Management System (RDBMS) is a DBMS that organizes its data into a series of tables which might be related by common fields. Its structure is made up of database tables, fields, and records. RDBMS also allows manipulation of data in the database tables with the use of relational operators [31].

One of the most popular open source RDBMS products is MySQL [32].

IV. Design and Implementation

A. Use Case Diagram

Top level Use Case Diagram

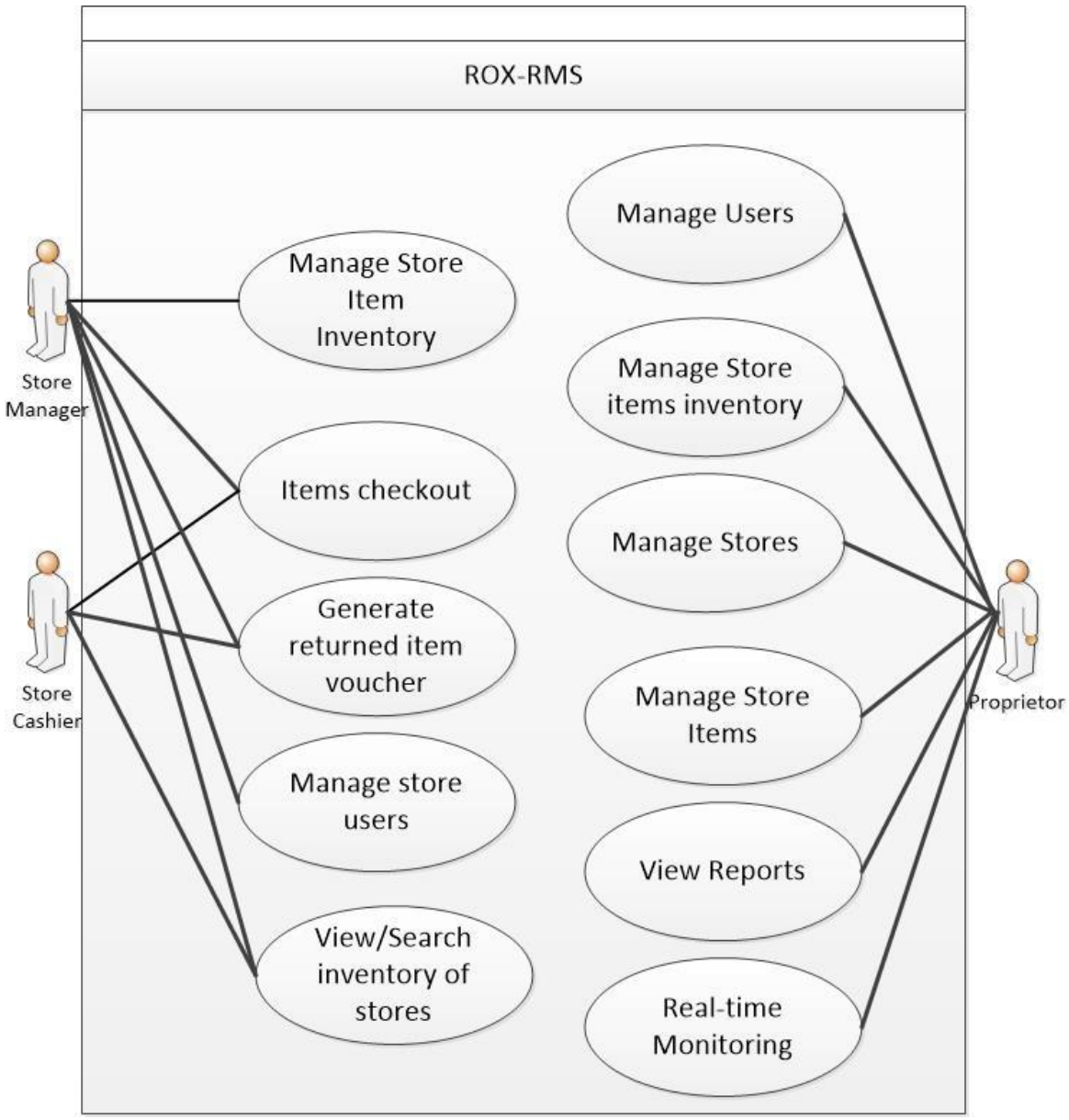


Figure 4. 2 Top Level Use Case Diagram for ROX-RMS

B. Activity Diagram

1. Manage Users

The Proprietor and Store Manager has the capability to add, edit, enable, or disable any system user. Store manager is only limited to the store he/she manages.

A. Add User

The activity diagram below (Figure 4.2a) illustrates how a Proprietor can add new user to the system.

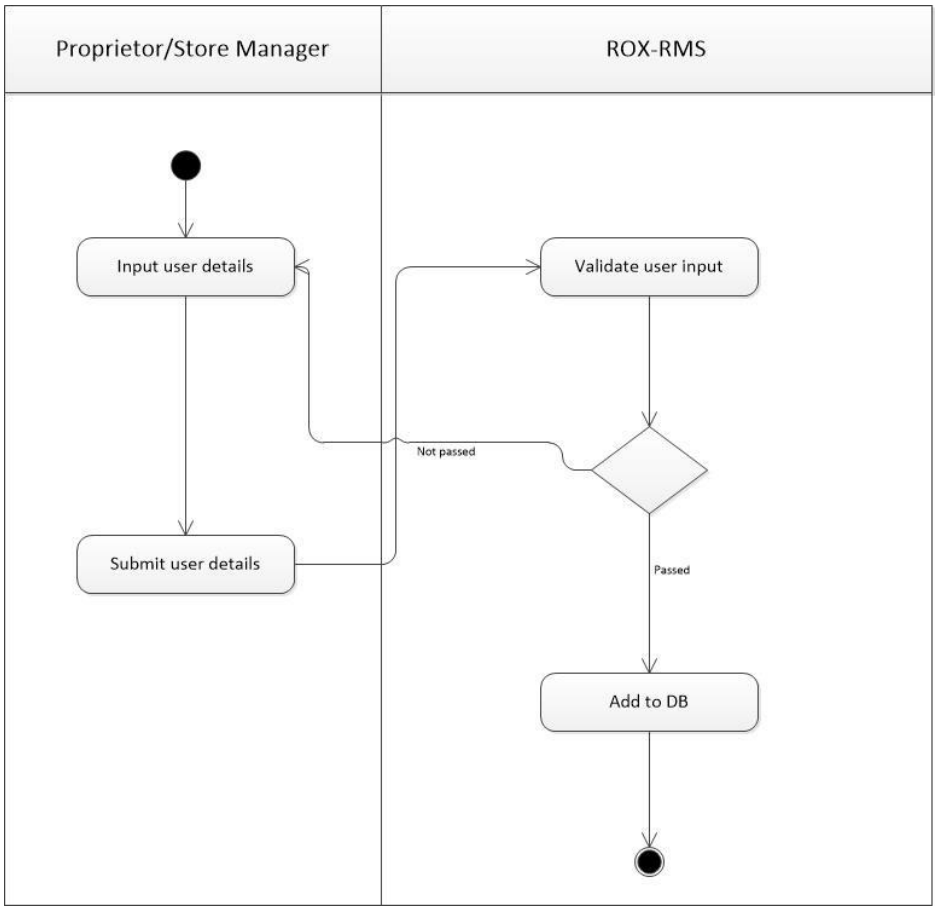


Figure 4. 3 Activity Diagram for Adding a user

B. Edit Details/Change User Password

The Proprietor and Store Manager can also edit details of a user. Store Manager is only limited to the store he/she manages. Figure 4.4 below illustrates this.

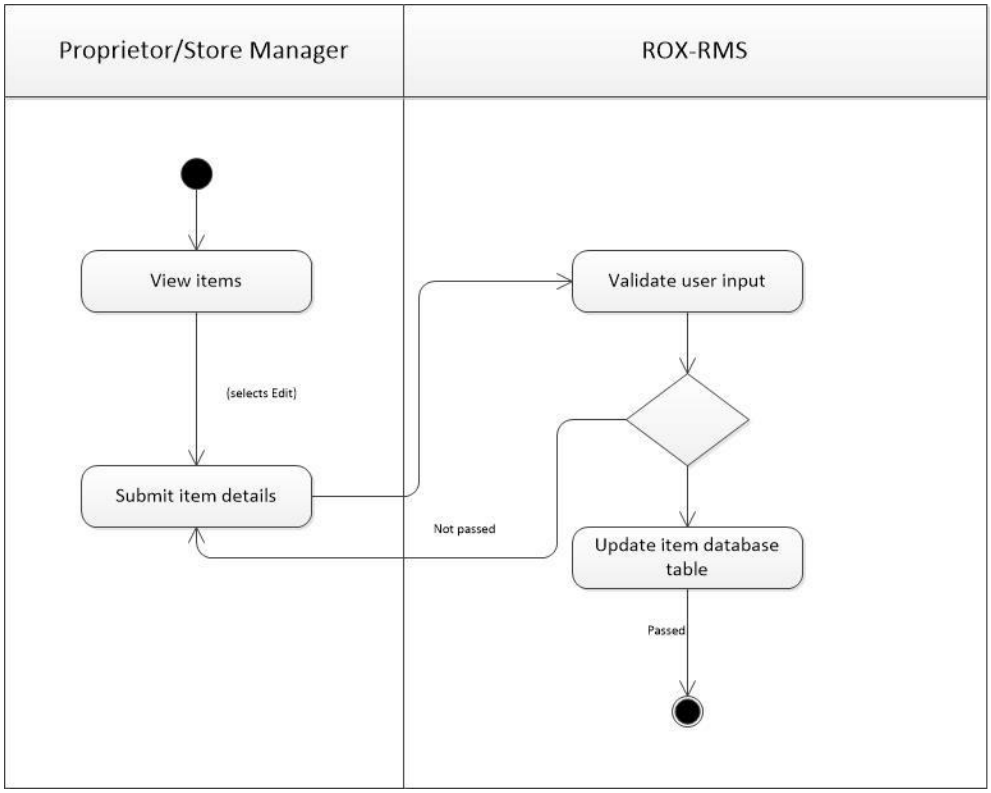


Figure 4. 4 Activity Diagram for editing user details/password

C. Enable/Disable User

The Proprietor and Store Manager can enable or disable a user. See Figure 4.5 below:

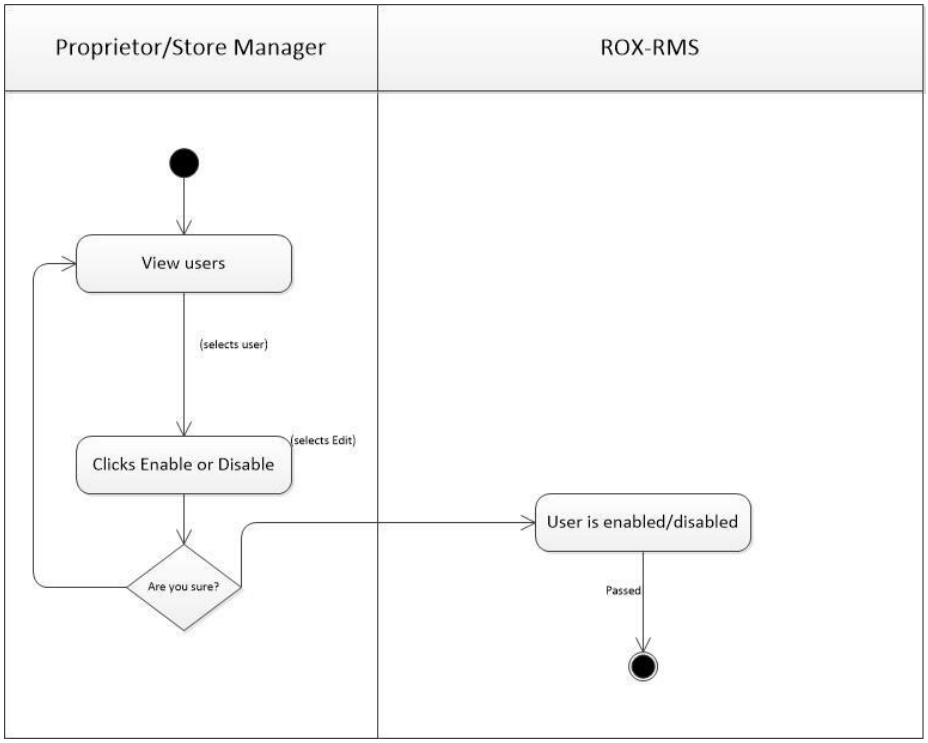


Figure 4. 5 Activity Diagram for enabling of disabling a user

2. Manage Stores

   A. Add Store

      Proprietor can add new store in the system. Figure 4.6 illustrates below:

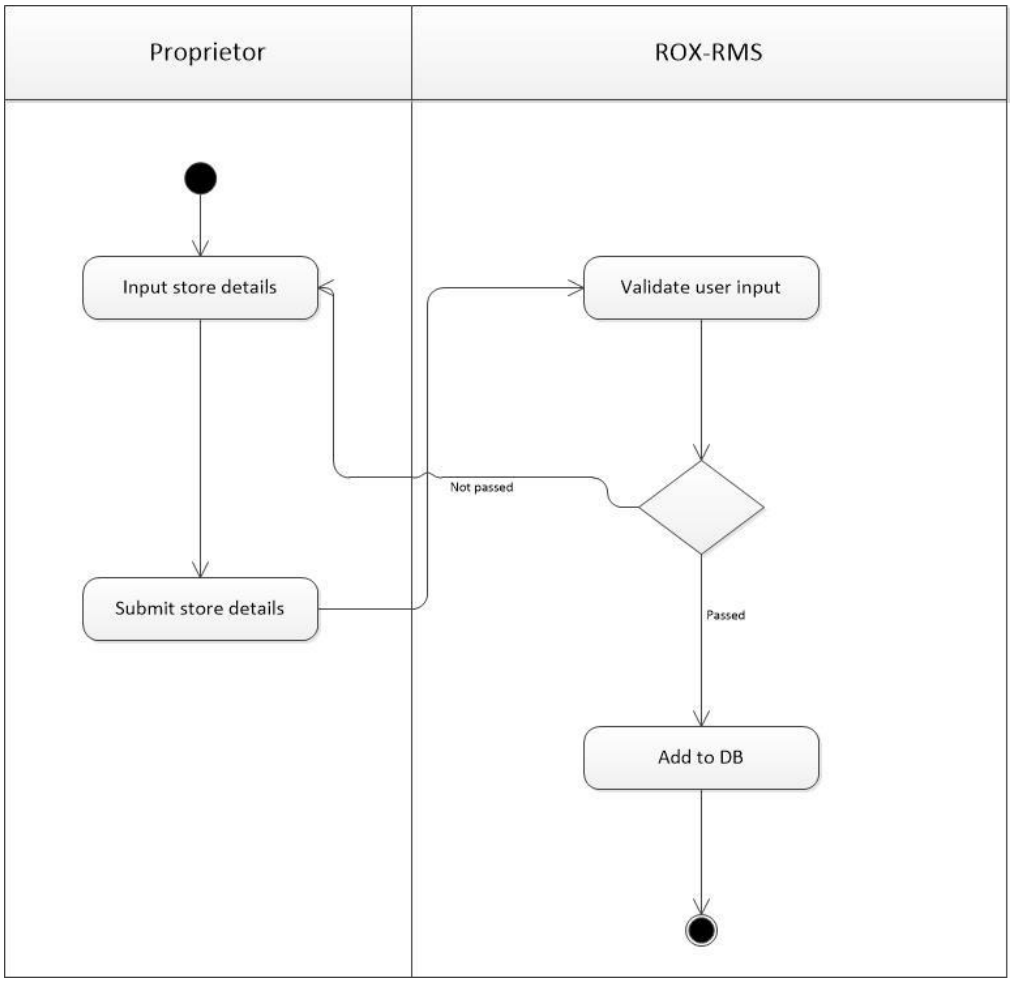


Figure 4.6 Activity Diagram for Adding a store

B. Edit Store Details

Proprietor can also Edit store details (Figure 4.7).

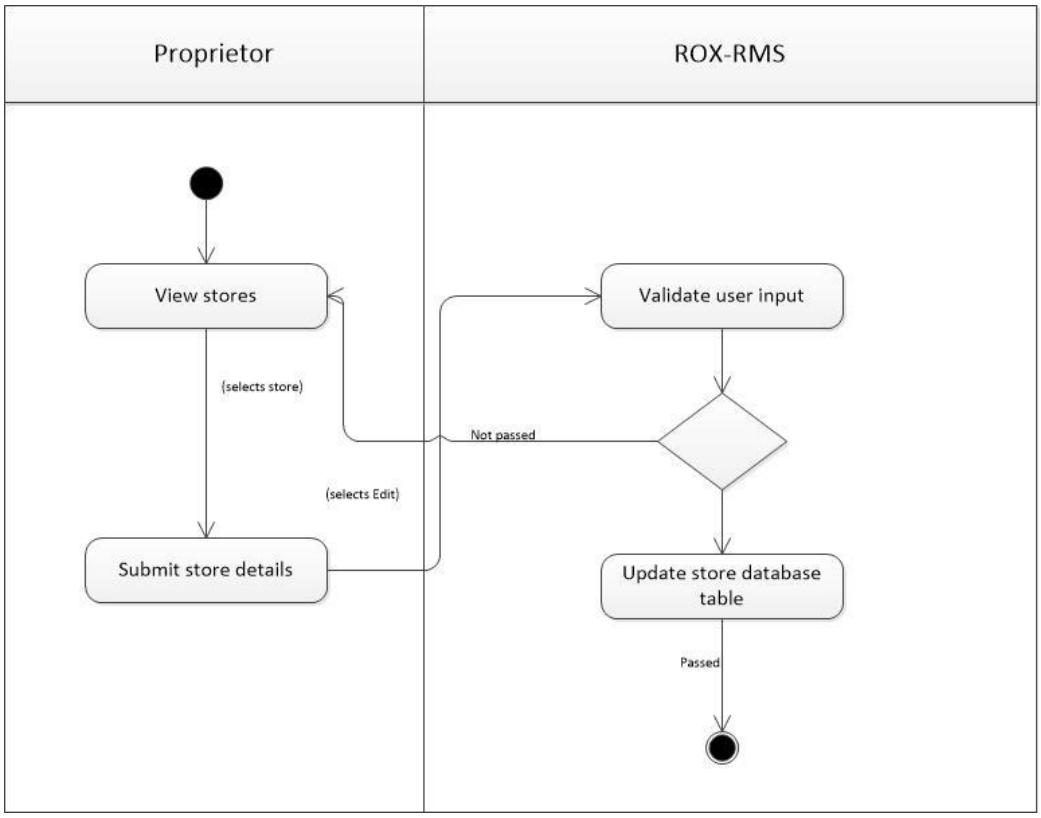


Figure 4.7 Activity Diagram for editing store details

3. Manage Items

   A. Add item

      Proprietor can add new item to be sold to all stores (Figure 4.8)

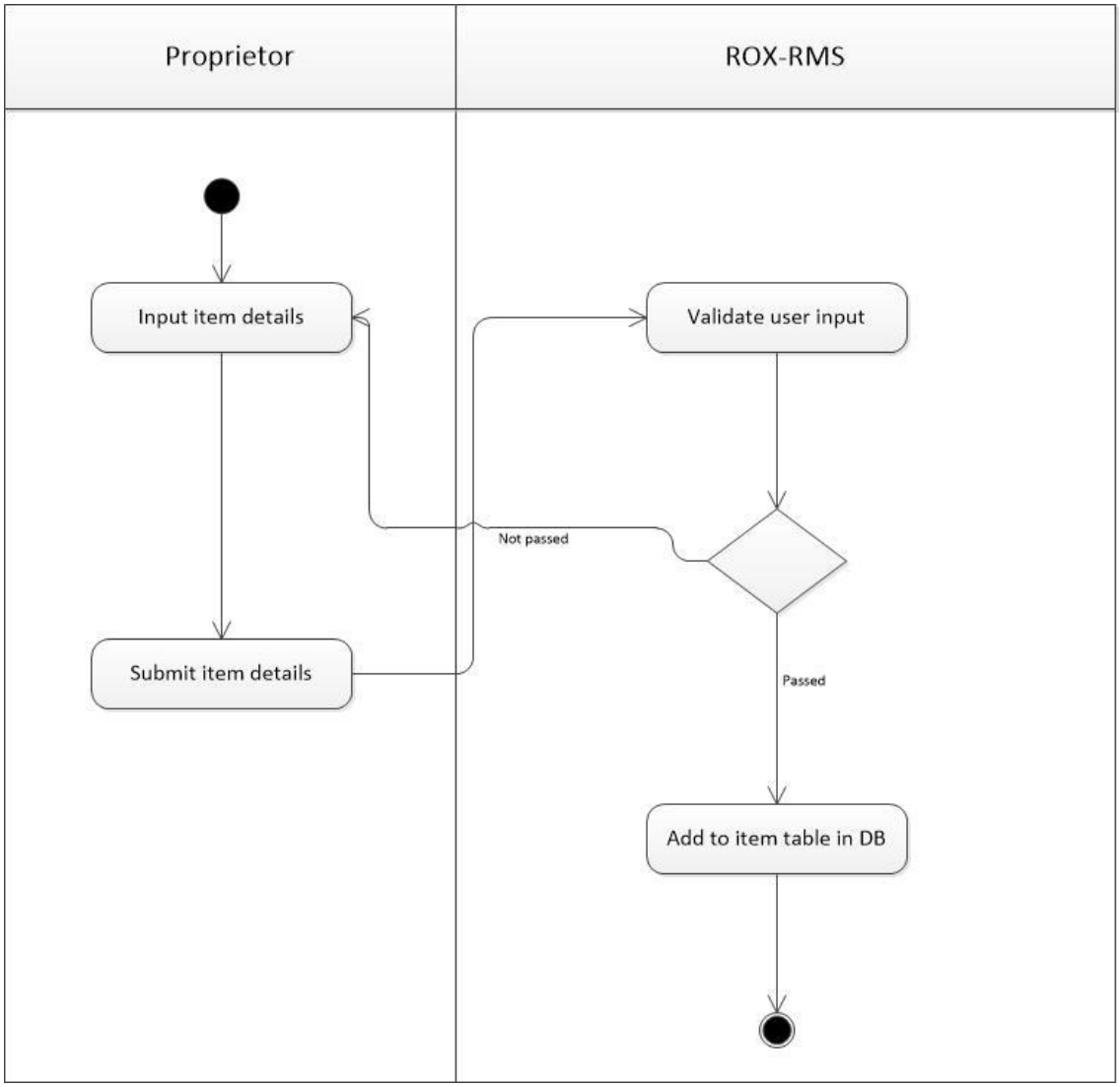


Figure 4.8 Activity Diagram for adding new store item inventory

B. Edit Item details

Proprietor can also edit item details as illustrated in Figure 4.9 below:

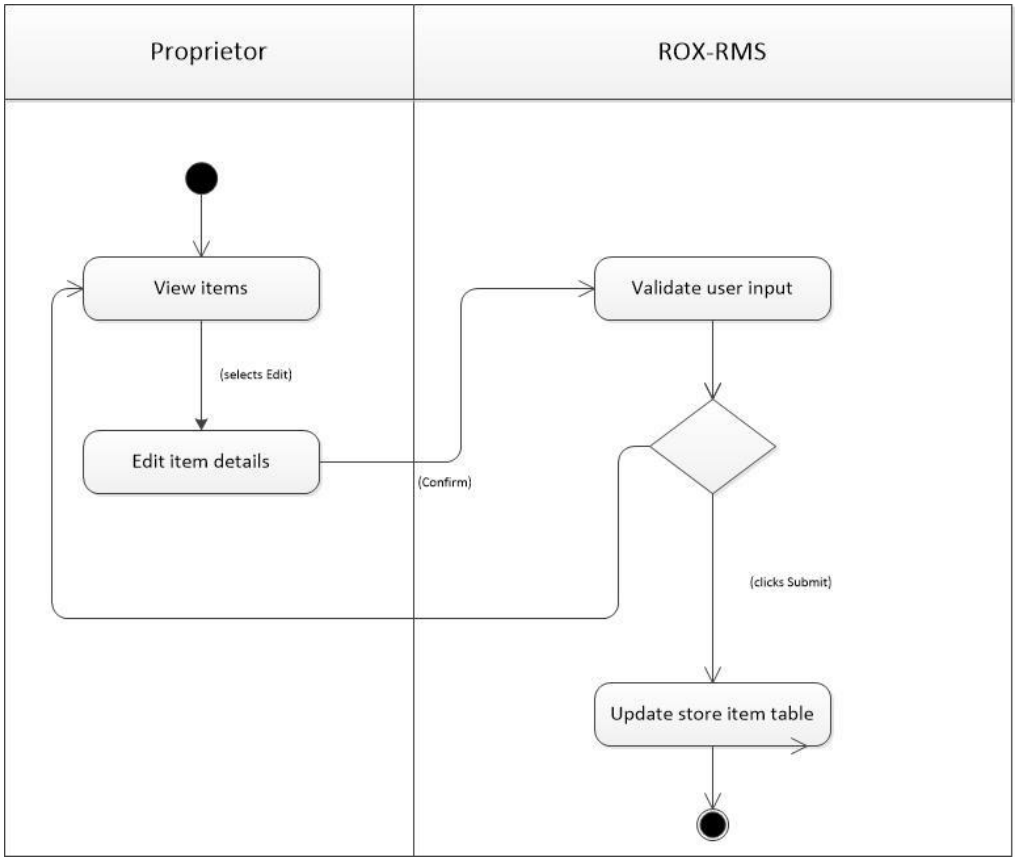


Figure 4.9 Activity Diagram for editing store item

4. View Reports

Proprietors can view reports based on the options selected (Figure 4.10).

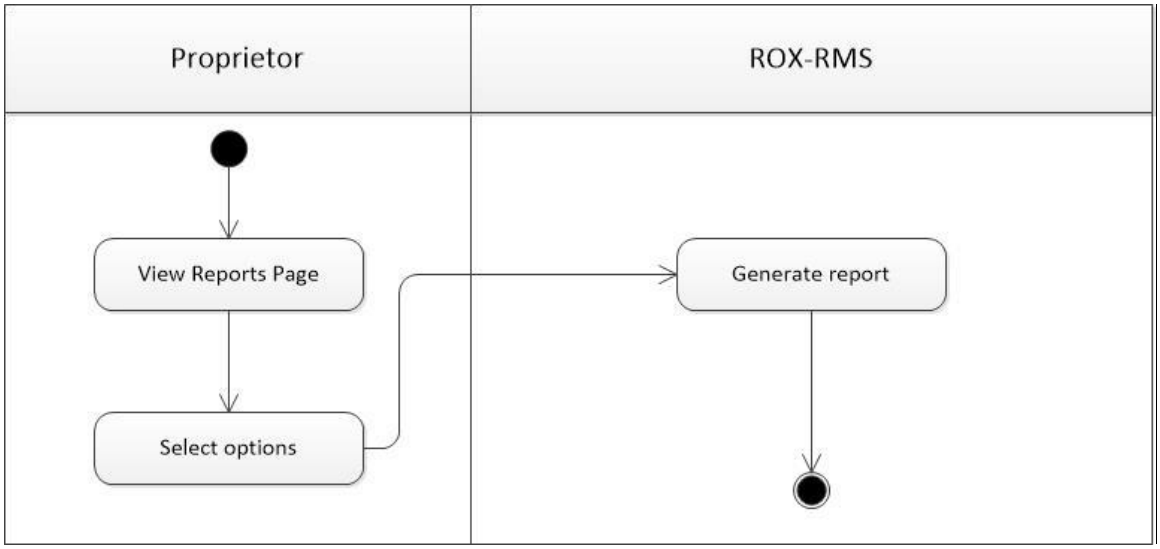


Figure 4.10 Activity Diagram for Viewing Reports

5. Real-time Monitoring

Proprietor can also access Real-time monitoring page (Figure 4.12)

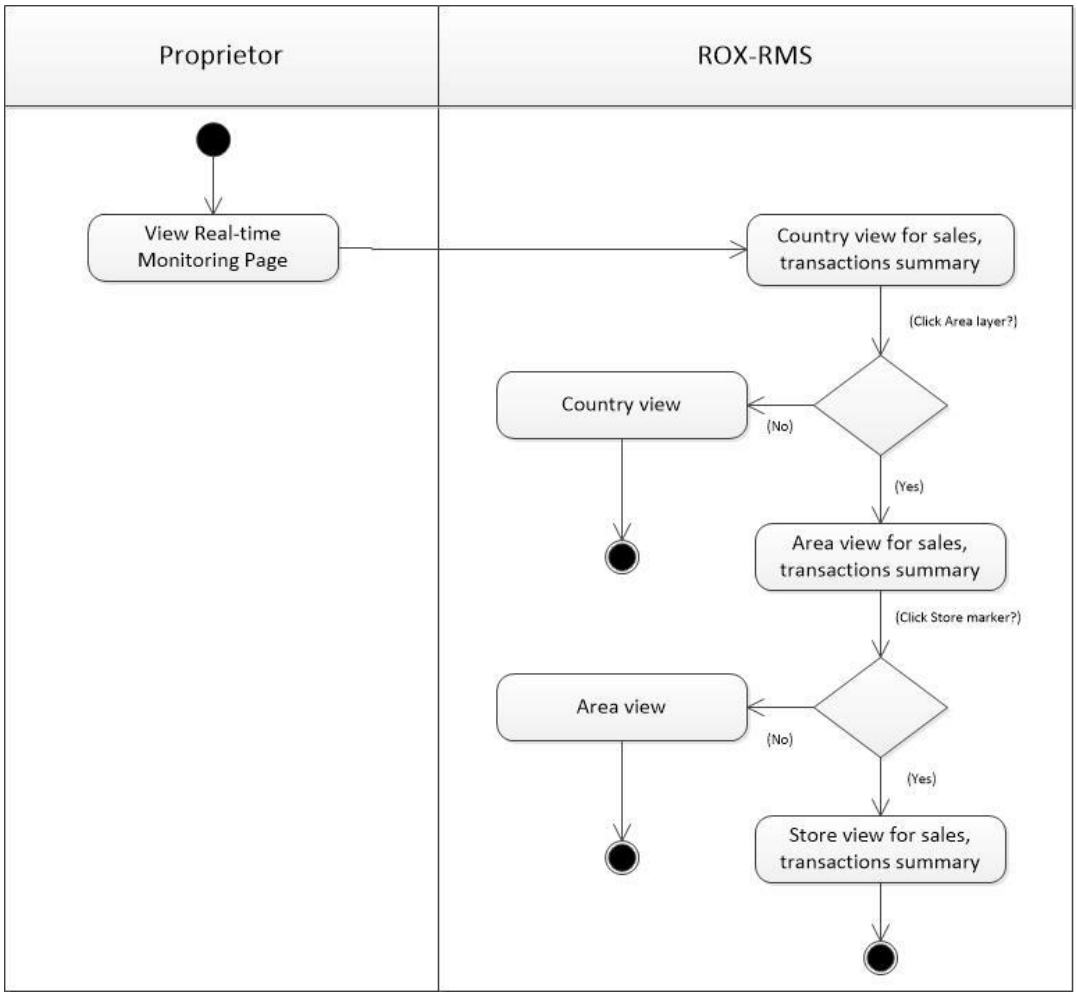


Figure 4.11 Activity Diagram for Real-time Monitoring page

6. Manage Item Inventory

A. Add Store Item Inventory

Only proprietor can add an item inventory. Figure 4.12 illustrates how the user can do this.

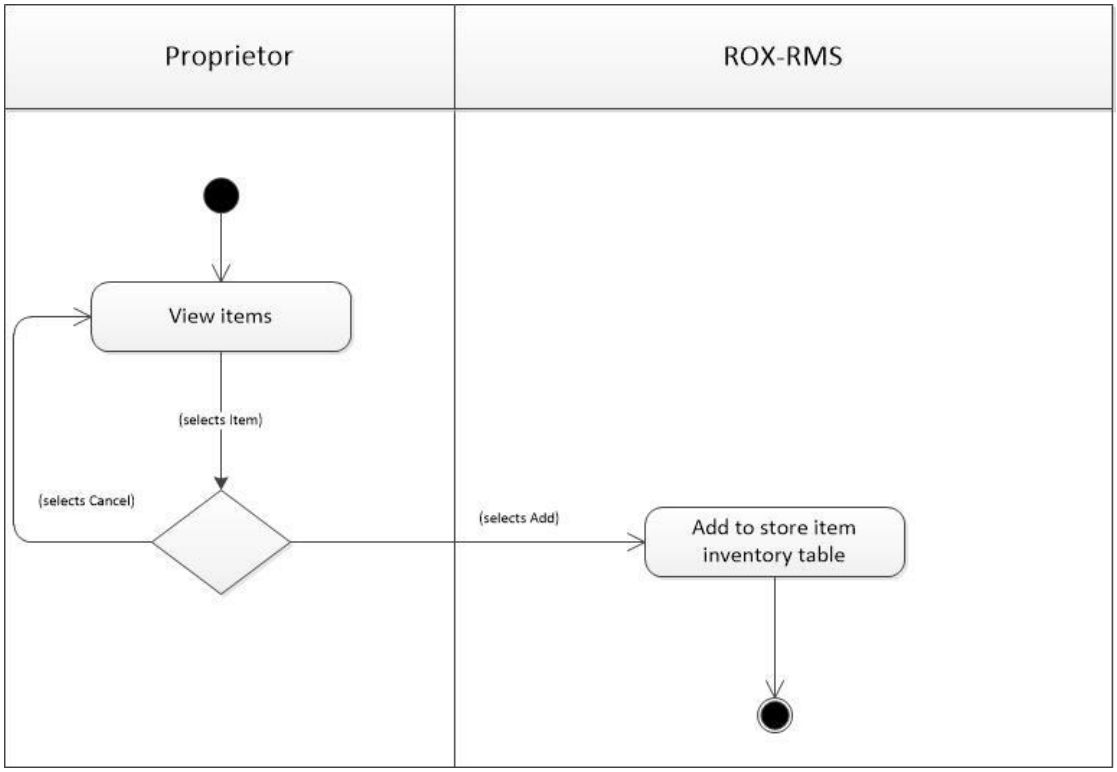


Figure 4.12 Activity Diagram for Adding an item inventory for stores

B. Update Item Inventory quantity

Only proprietors can update a store item inventory quantity. Figure 4.13 below illustrates this.

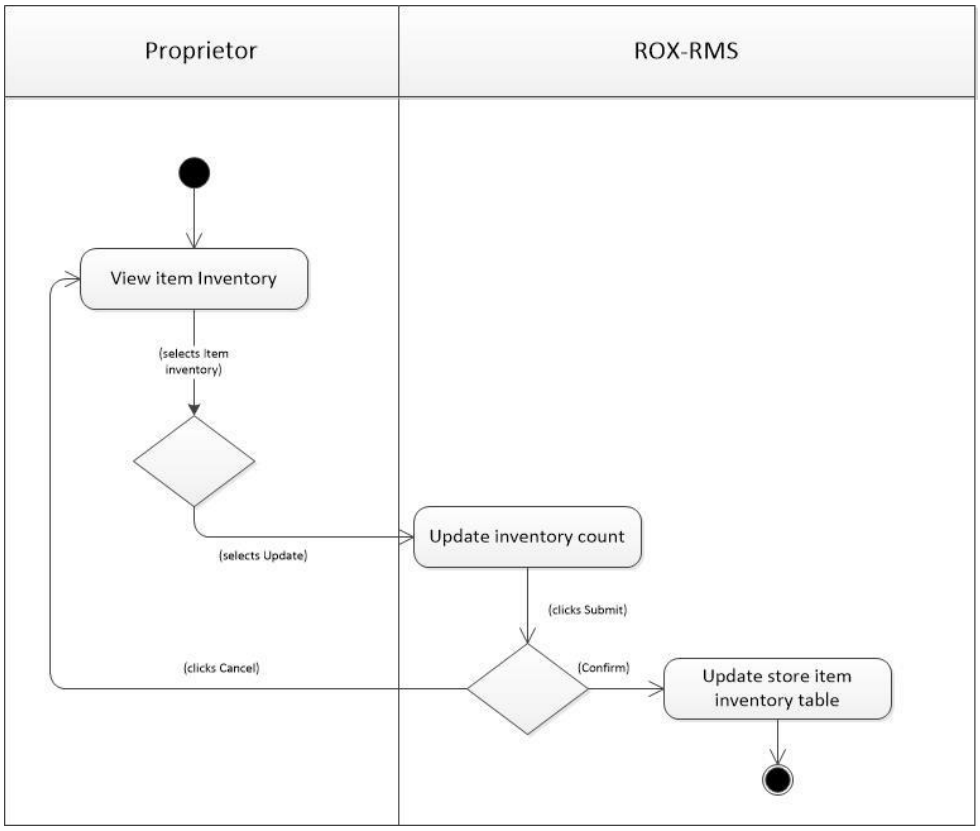


Figure 4.13 Activity Diagram for updating an item inventory for stores

C. View/Search Store inventories

Store managers and cashiers can search for available inventories of other stores (Figure 4.14).

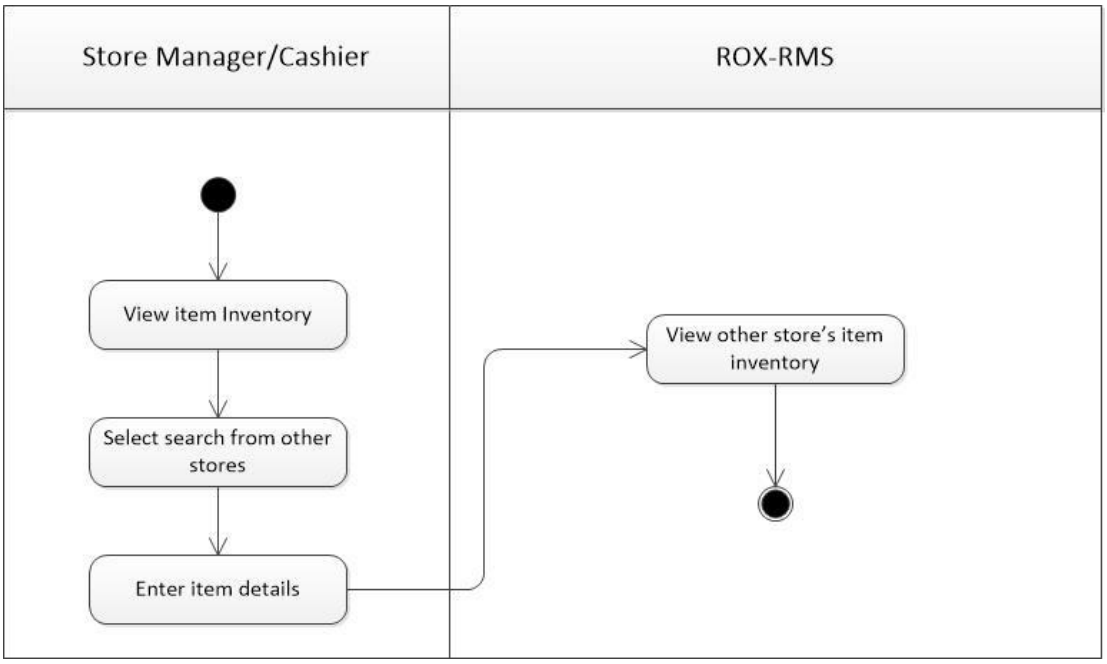


Figure 4.14 Activity Diagram for Adding an item inventory for stores

7. POS Transactions

A. Checkout Items

Only store managers and cashiers can do Checkout Items activity (Figure 4.15)

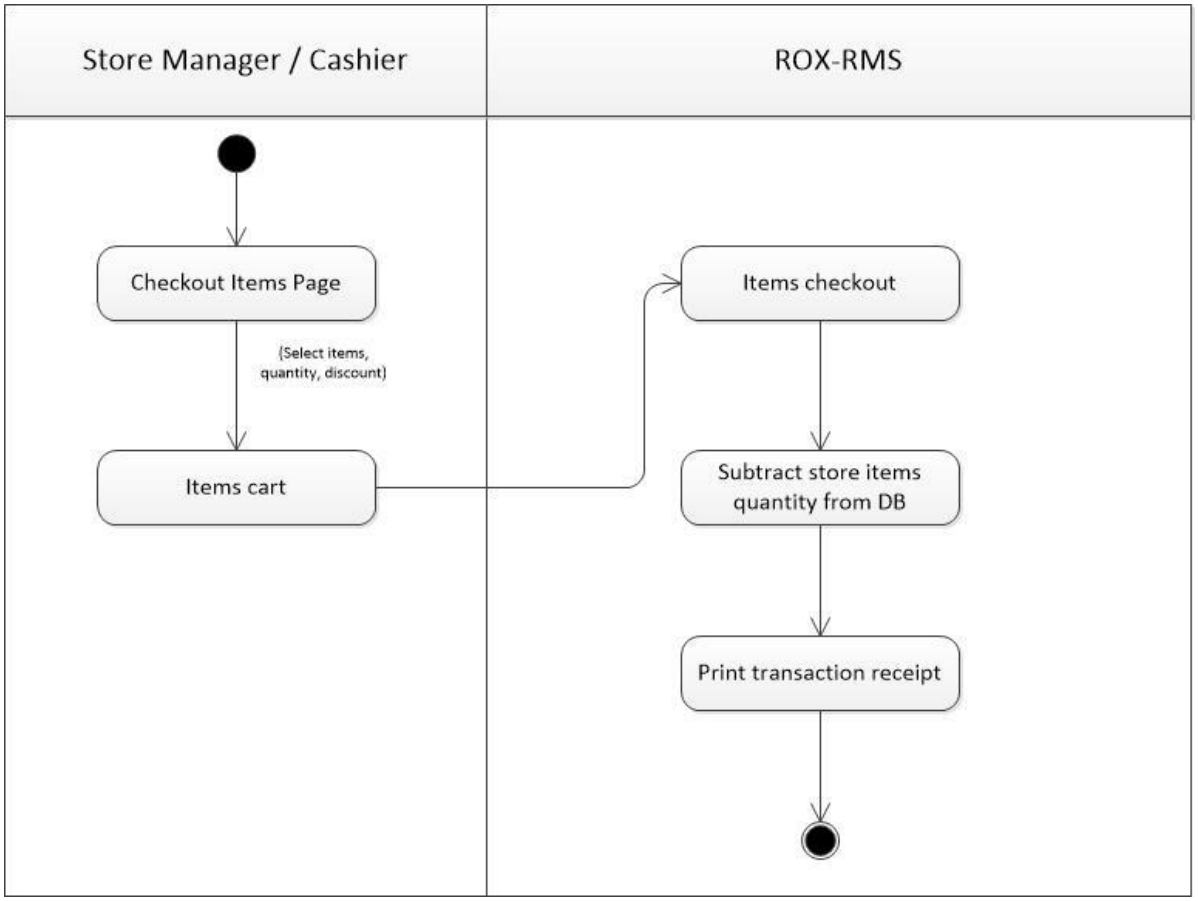


Figure 4.15 Activity Diagram for Checking out items

B. Return items

Store managers and cashiers can also do Return Items activity (Figure 4.16).

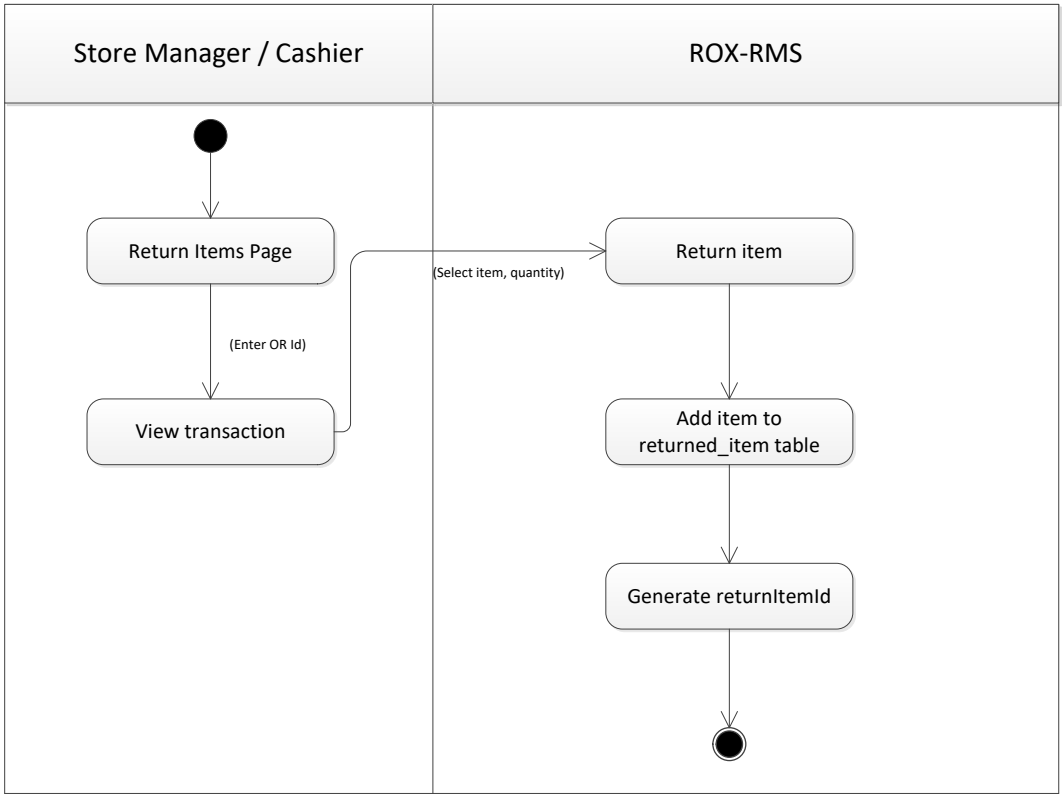


Figure 4.16 Activity Diagram for Return Items

C. Context Diagram

ROX-RMS is a system that supports three types of users—proprietor, store manager, and cashier. Figure 4.1 shows the context diagram for the system.

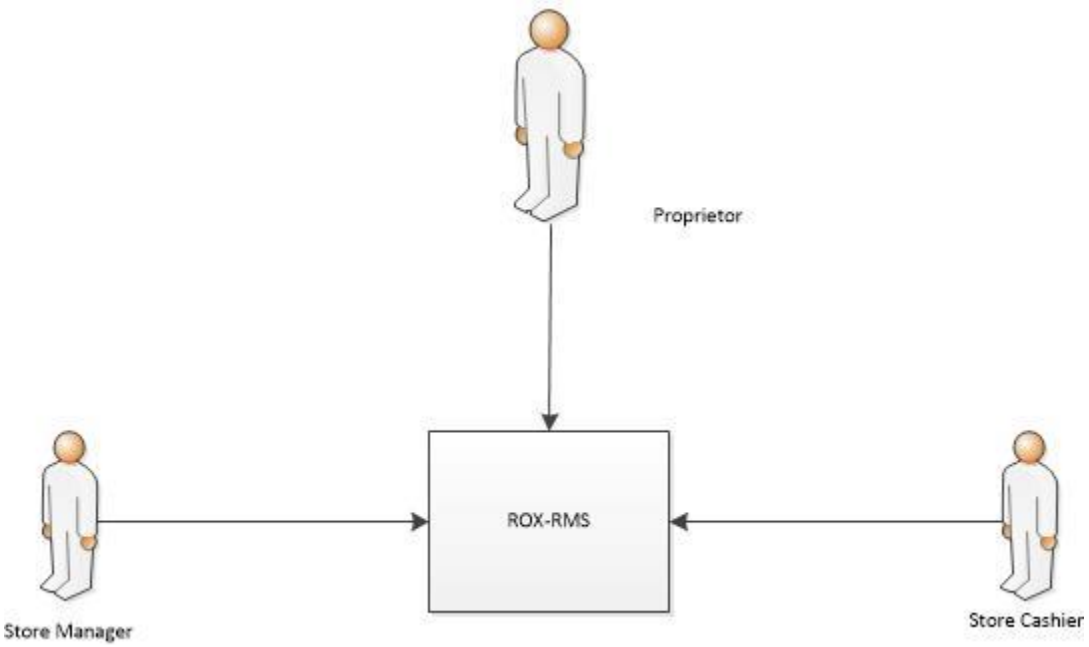


Figure 4. 17 Context Diagram for ROX-RMS

D. Entity Relationship Diagram

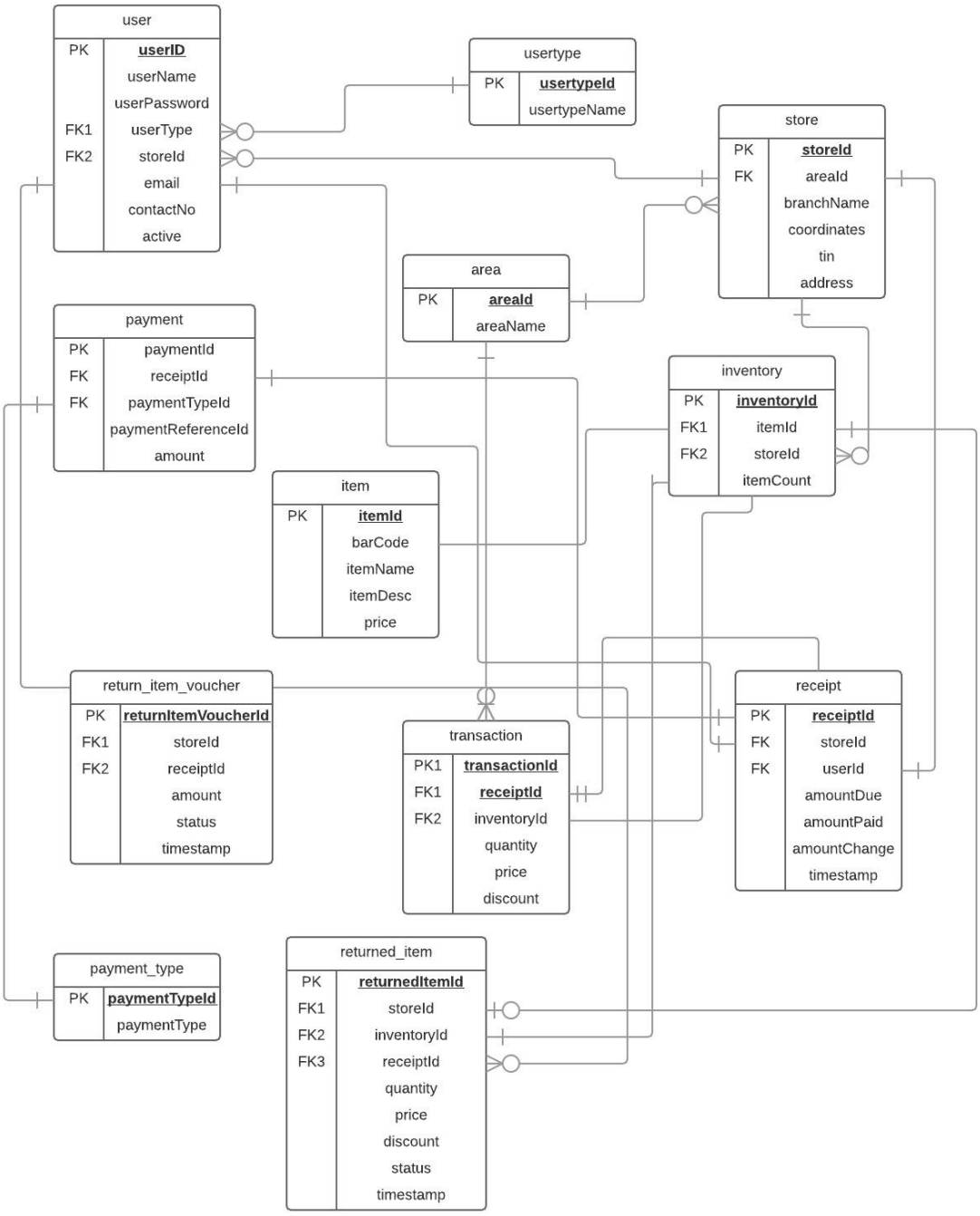


Figure 4. 18 Context Diagram for ROX-RMS

E. Data Dictionary

user – table for all users that can use the system		
Column	**Type**	**Description**
userID	int(11)	Unique identifier for all users
username	varchar(20)	Username used to log in to the system
userPassword	varchar(20)	User password used to log in to system
userType	int(11)	Identifies what type the user is. Links to usertype → usertypeId
storeId	int(11)	Identifies what store the user is under. Links to store → storeId
Email	varchar(90)	Email address of the user
contactNo	varchar(13)	Contact number of the user
Active	int(11)	Indicator if user is active or not

usertype – table that stores the types of users for the system		
Column	**Type**	**Description**
usertypeId	int(11)	Unique identifier for the table
usertypeName	varchar(45)	Value of the usertype

area – table for the area where the stores will be placed under		
Column	**Type**	**Description**
areaId	int(11)	Unique identifier for the table
areaName	varchar(45)	Value of the area

store – table for stores that will be used in the system		
Column	**Type**	**Description**
storeId	int(11)	Unique identifier for the table
areaId	int(11)	Identifies what area the store is under. Links to area → areaId

branchName	varchar(45)	Identifier of store branch
Tin	varchar(45)	Tax Identification Number of the store
Address	varchar(160)	Address of the store
Coordinates	varchar(60)	Stores the x and y coordinates of the stores, to be plotted on map

item – table for items that will be used in stores		
Column	**Type**	**Description**
itemId	int(11)	Unique identifier for the table
barCode	varchar(30)	Composite primary key for the table, stores the bar code of the product
itemName	varchar(30)	Name of the item
itemDesc	varchar(160)	Description of the item
price	Double	Price of the item

inventory– table for items that the store sells, along with the quantity available for each store		
Column	**Type**	**Description**
inventoryId	int(11)	Unique identifier for the table
itemId	int(11)	Identifies the item. Links to item → itemId
storeId	int(11)	Identifies which store the items are for. Links to store → storeId
itemCount	int(11)	Quantity of items available in the store

transaction – table for point of sales (POS) transactions		
Column	**Type**	**Description**
transactionId	int(11)	Unique identifier for the table
receiptId	int(11)	Receipt ID that will be generated after each transaction. Links to receipt → receiptId
inventoryId	int(11)	Inventory ID of the item. Links to inventory → inventoryId
Quantity	int(11)	Quantity of checked out items

Price	Double	Price of item per piece
Discount	int(11)	Discount of the item in percent

receipt – table for receipt and total price per transaction		
Column	**Type**	**Description**
receiptId	int(11)	Unique identifier for the table
storeId	int(11)	Id of the store. Links to store→storeId
userId	int(11)	Id of the cashier. Links to user→userId
amountDue	Double	Total price per transaction
amountPaid	Double	Cash paid to cashier
amountChange	Double	Change after paying
Timestamp	Datetime	Timestamp of the transaction

return_item_voucher – table used to create voucher when returning items; can be used to pay for replacement item(s)		
Column	**Type**	**Description**
returnItemVoucherId	int(11)	Unique identifier for the table
storeId	int(11)	Id of the store. Links to store→storeId
receiptId	int(11)	Receipt ID of the transaction. Links to transaction → receiptId
amount	Double	Amount that can be used by the user to pay for next transaction
status	Varchar(45)	Status to know if amount is already used
timestamp	Datetime	Timestamp of the transaction

returned_item – table for items returned		
Column	**Type**	**Description**
returnedItemId	int(11)	Unique identifier for the table
storeId	int(11)	Id of the store where the item is returned. Links to store → storeId

inventoryId	int(11)	Id of the inventory. Links to inventory → inventoryId
receiptId	int(11)	receiptId of the item returned. Links to transaction → receiptId
Quantity	int(11)	Quantity of the item
Price	Double	Listed price of the item
Discount	int(11)	Discount of the item in percent
Status	Varchar(45)	Status of the item returned
Timestamp	Datetime	Timestamp of the transaction

payment – table for payment		
Column	**Type**	**Description**
paymentId	int(11)	Unique identifier for the table
receiptId	int(11)	Id of the receipt generated. Links to receipt → receiptId
paymentTypeId	int(11)	Id of payment type used. Links to payment_type → paymentTypeId
paymentReferenceId	Varchar(45)	Reference Id of payment. Taken from outside the system.
amount	Double	Amount paid

payment_type – table used to store types of payment		
Column	**Type**	**Description**
paymentTypeId	int(11)	Unique identifier for the table
paymentType	Varchar(45)	Value of payment type (Cash, Debit, Credit, Voucher)

V. Results

In order to use the system, the user has to log in first through a common log-in page as seen in Figure 5.1.

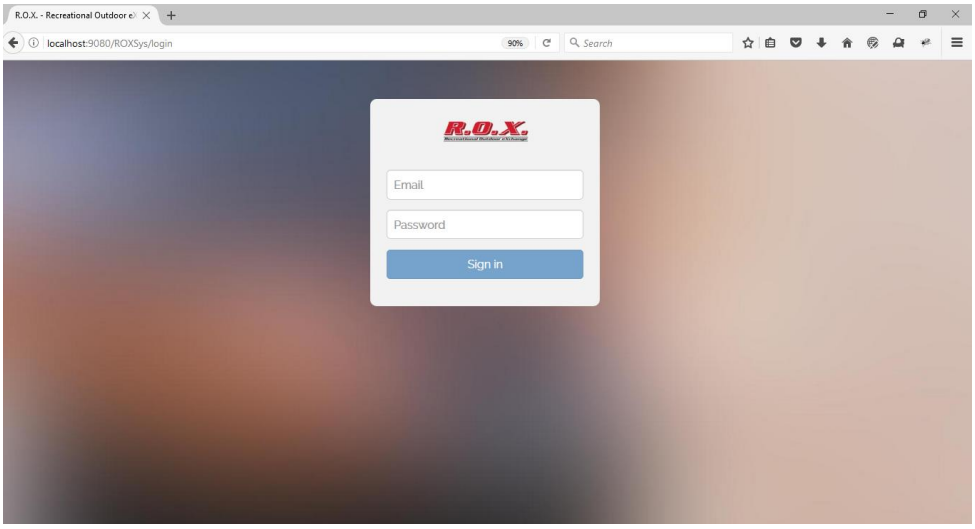


Figure 5.1-Log-in page

Once logged in, user will be directed to the home page. Different menu items are available for each type of user. Let's start with the Proprietor (Figure 5.2).

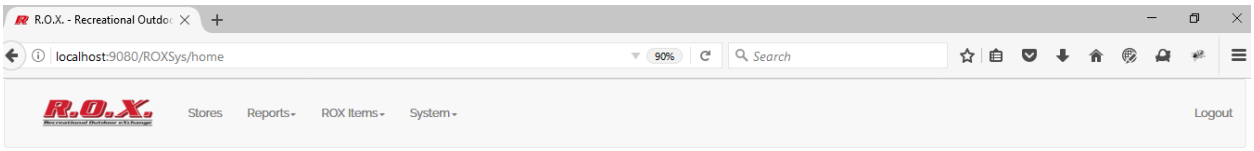


Figure 5.2-Menu items for Proprietor

When user clicks the Stores menu, he/she will be redirected to the Stores page as shown below (Figure 5.3):

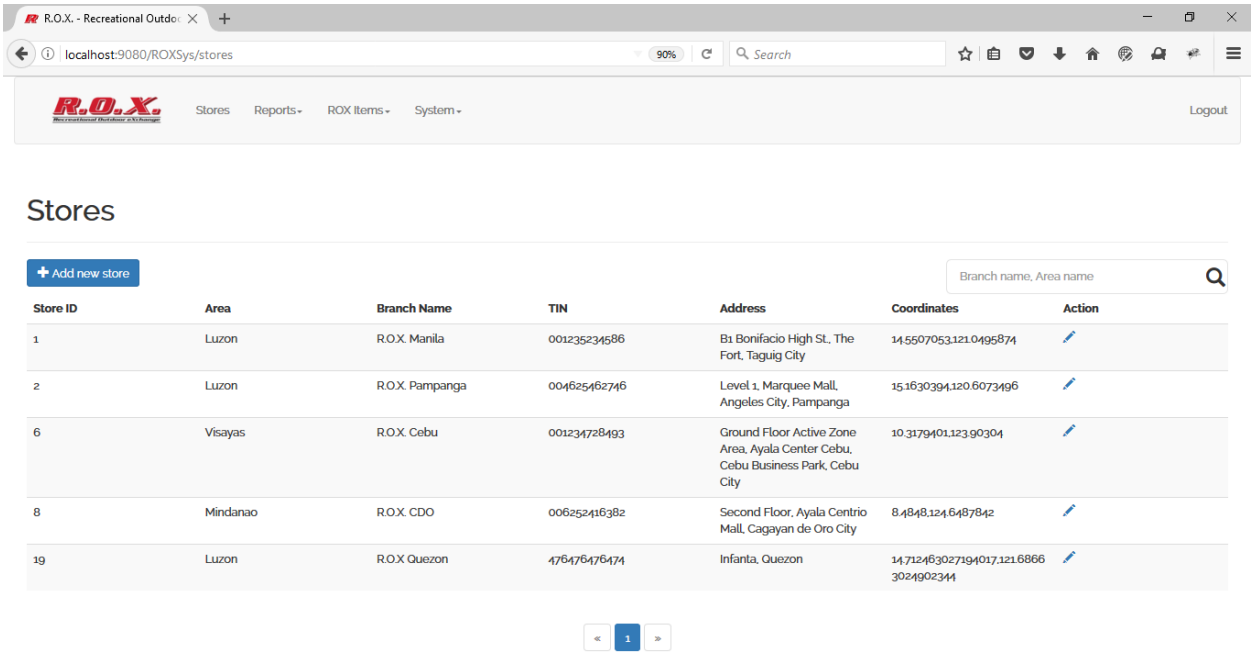


Figure 5.3-List of stores

When user clicks the "Add new store" button, a modal will open and the user has to fill out the required fields (Figure 5.4):

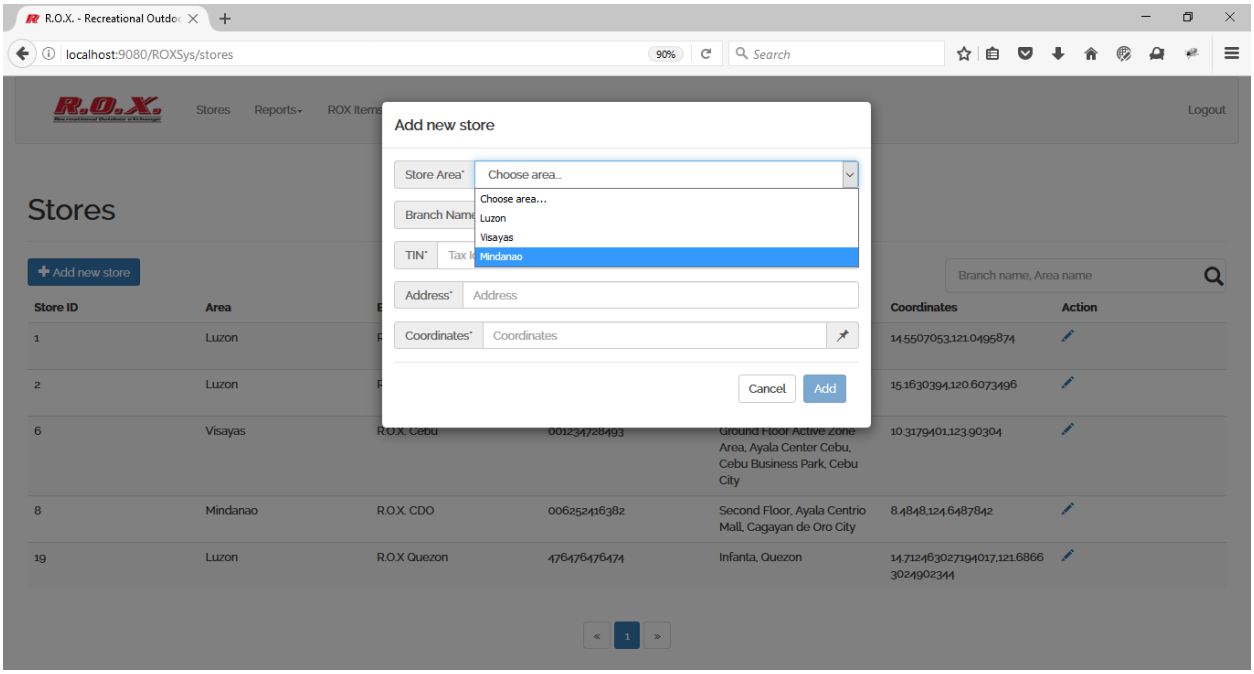


Figure 5.4-Add new store

Adding store coordinates is aided by opening another modal with a map. Clicking it automatically adds the x and y coordinates needed (Figure 5.5).

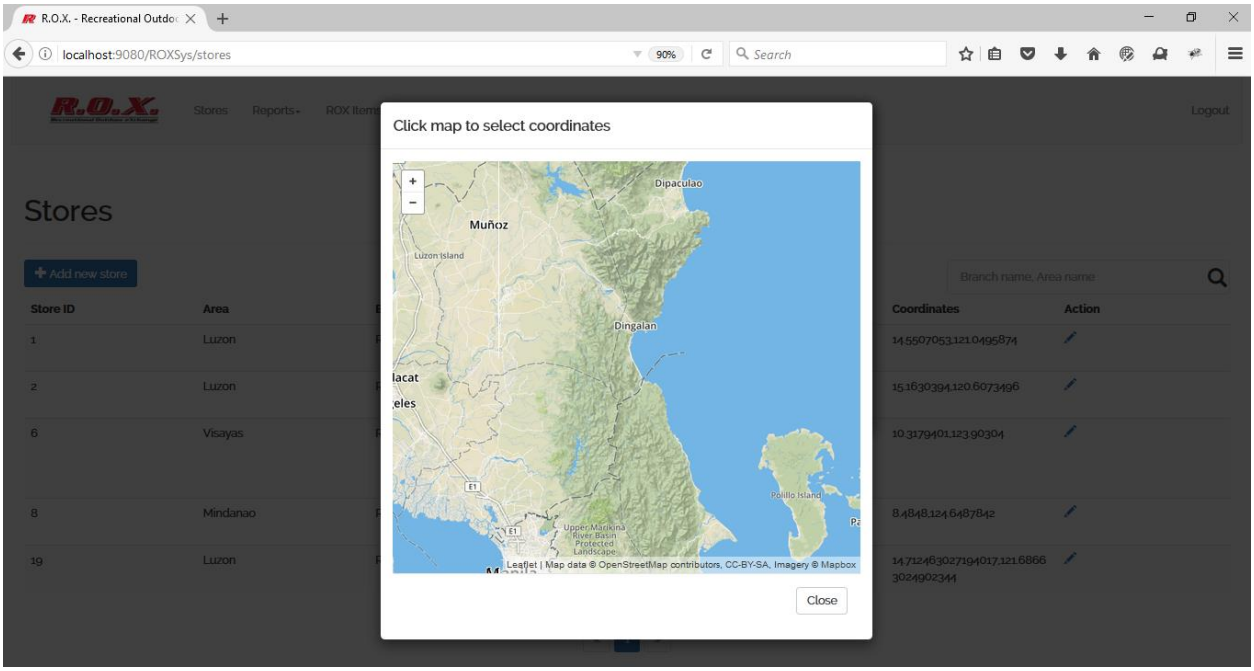


Figure 5.5-Add new store: selecting store coordinates

X and y coordinates are automatically added (Figure 5.6).

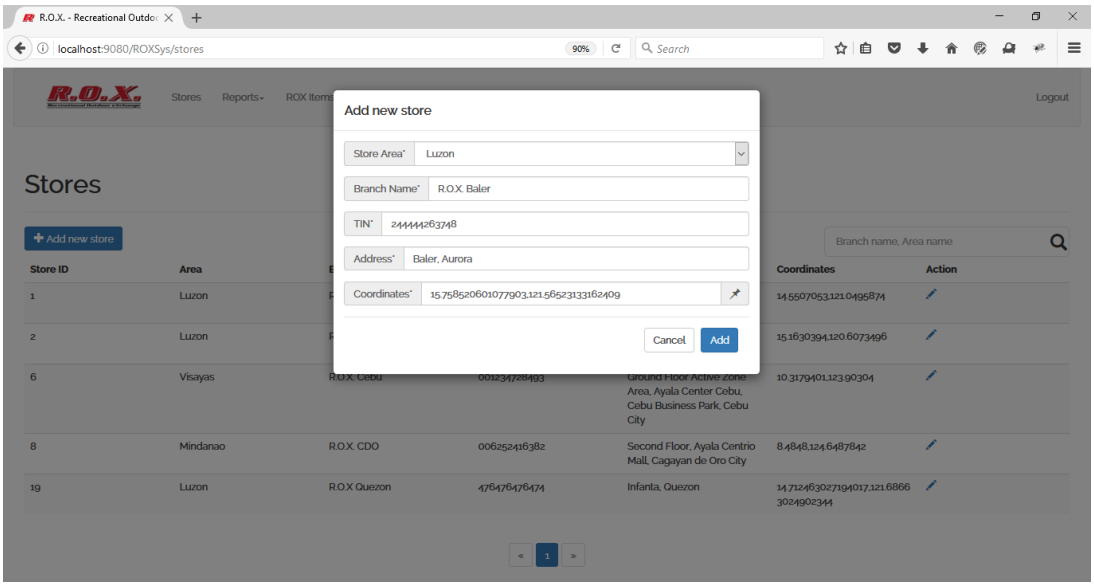


Figure 5.6-Add new store

Once button "Add" is clicked, the system will check if the values are valid. If they are, the new store is added in the database and will display on the table (Figure 5.7).

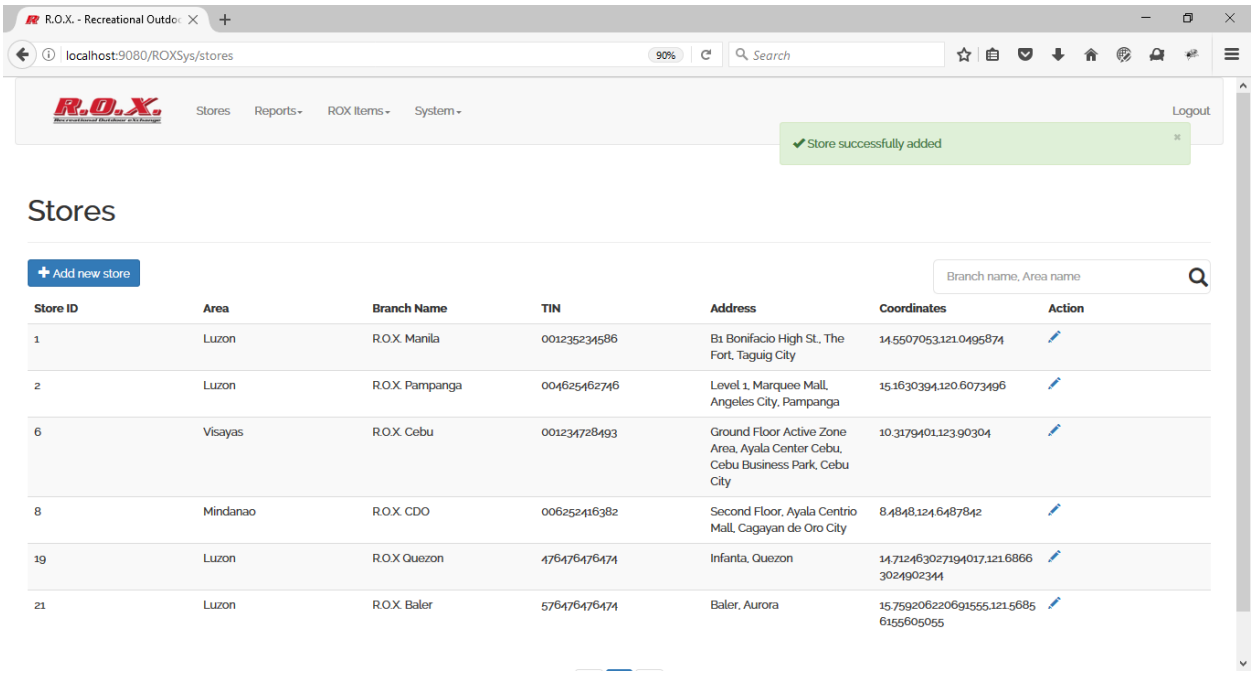


Figure 5.7-User successfully added

Updating the value of the store requires that the user clicks on the "Pen" icon under Action header. Clicking the icon opens a modal with the values of the store displayed (Figure 5.8).

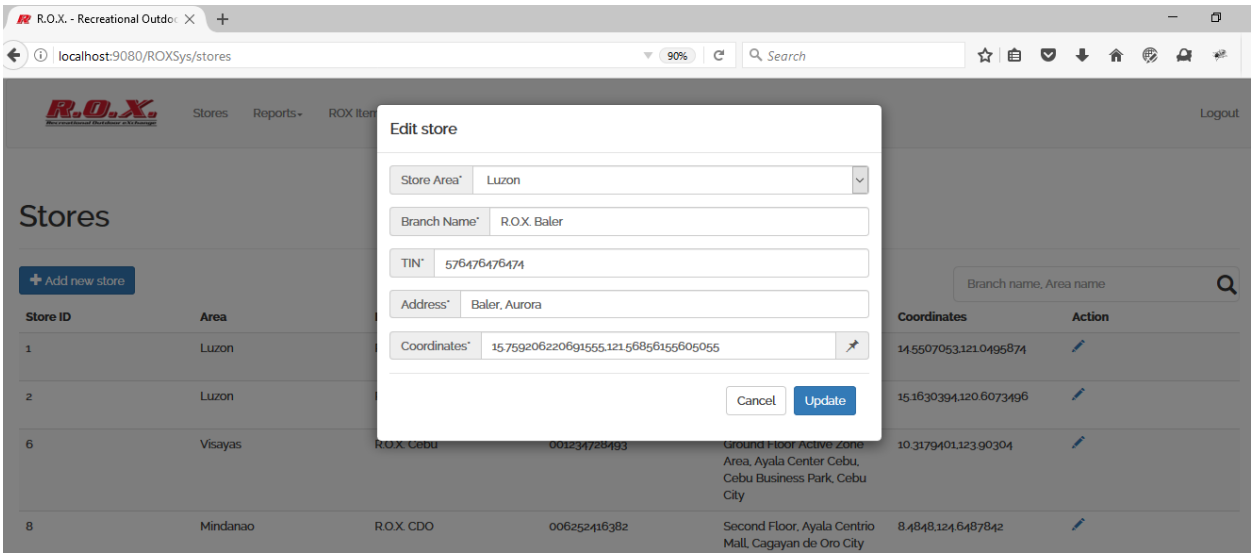


Figure 5.8-Updating store details

Just like in Add function, once user fills in new data and clicks on "Update" button, the values will be updated on the table (Figure 5.9).

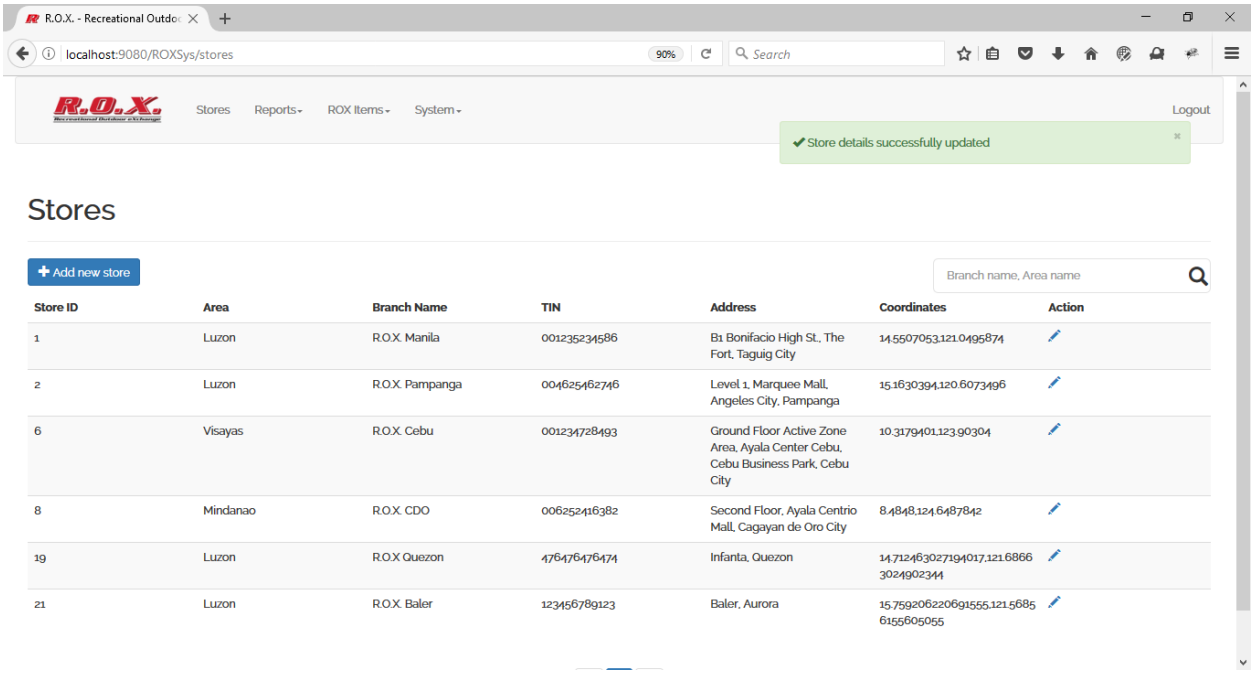


Figure 5.9-Store successfully updated

To navigate to Real-time monitoring page, the user must click on Real-time Monitoring sub-menu. It is under the Reports main menu (Figure 5.10).

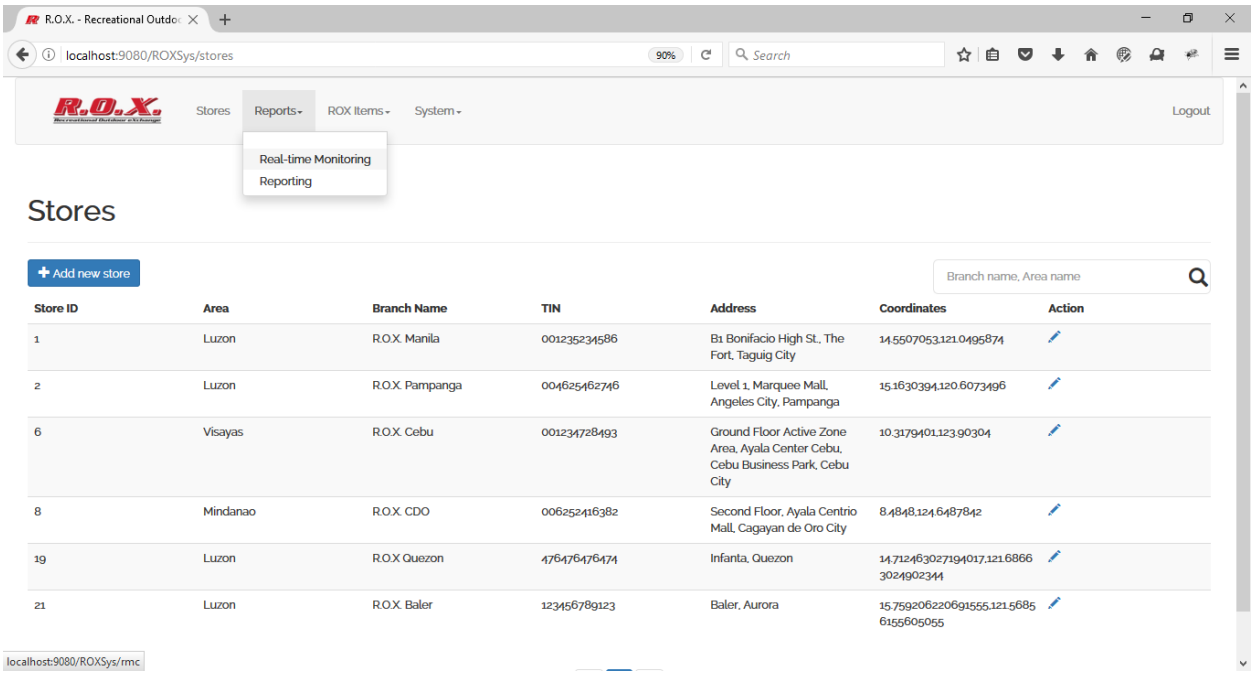


Figure 5.10- Navigating to Real-time Monitoring page

The real-time monitoring page opens with the Philippines view as default. Summaries of sales transactions and other information are displayed on the side of the map (Figure 5.11). The page auto-refreshes every five seconds.

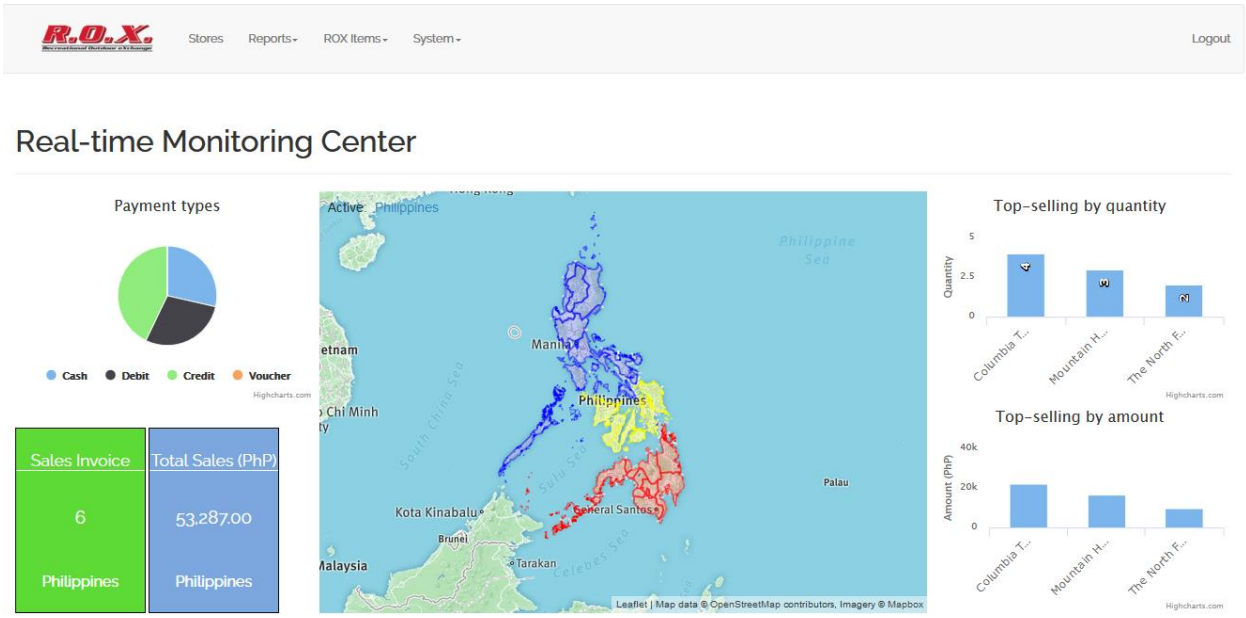


Figure 5.11- Real-time monitoring page—Philippines view

Clicking on any area (Luzon, Visayas, Mindanao) through a map layer zooms in the map to that particular area. The values will be updated based on the data for that area (Figure 5.12).

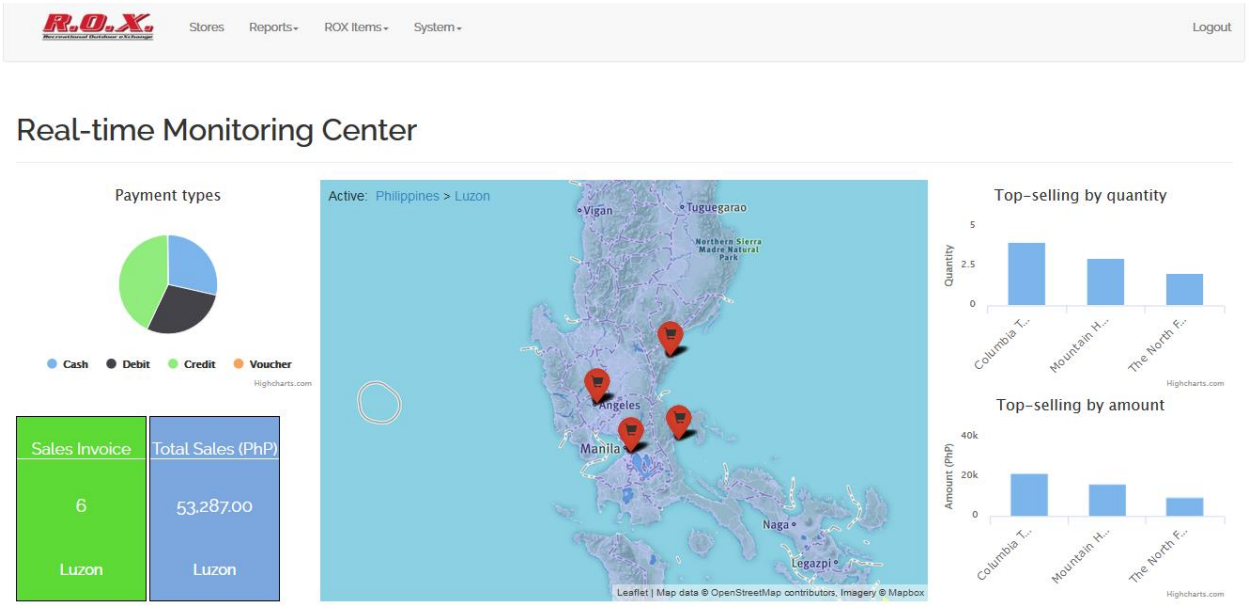


Figure 5.12- Real-time monitoring page—Area view

Clicking on any store under the area through a marker, updates the marker color into green. It mean that the store is active. The values will be updated based on the store clicked (Figure 5.13).

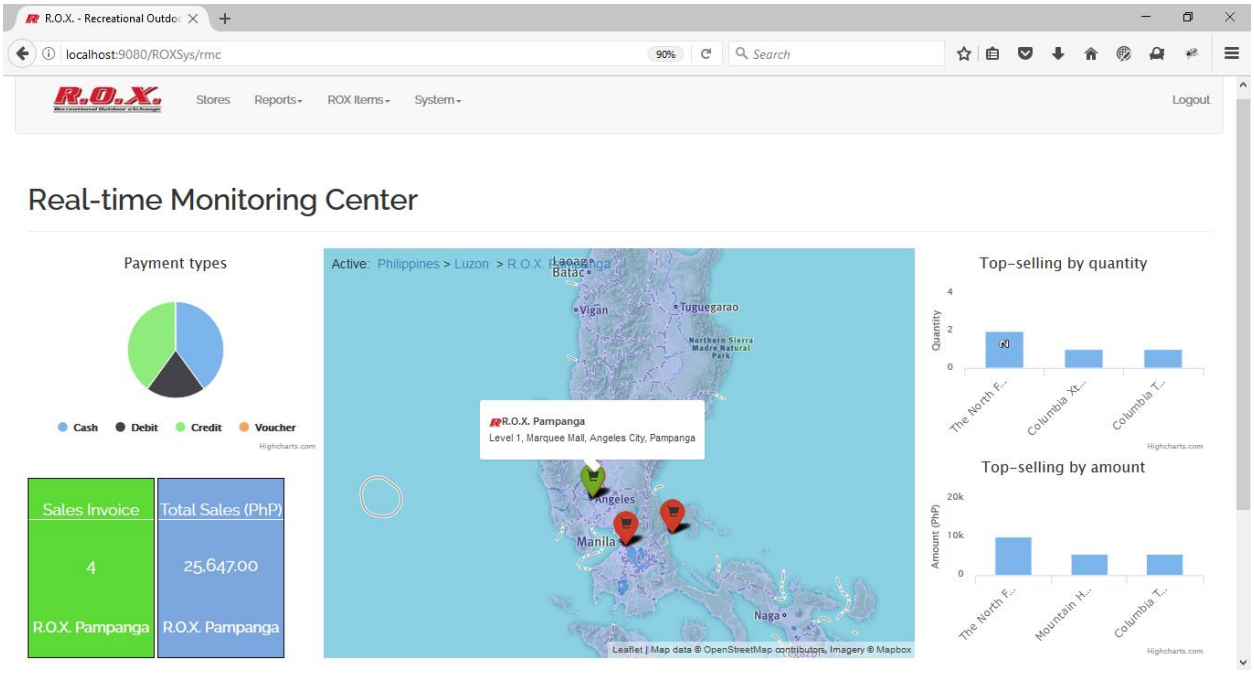


Figure 5.13- Real-time monitoring page—Store view

Navigating to Reporting page requires the user to click Reporting sub-menu from Reports main menu (Figure 5.14).

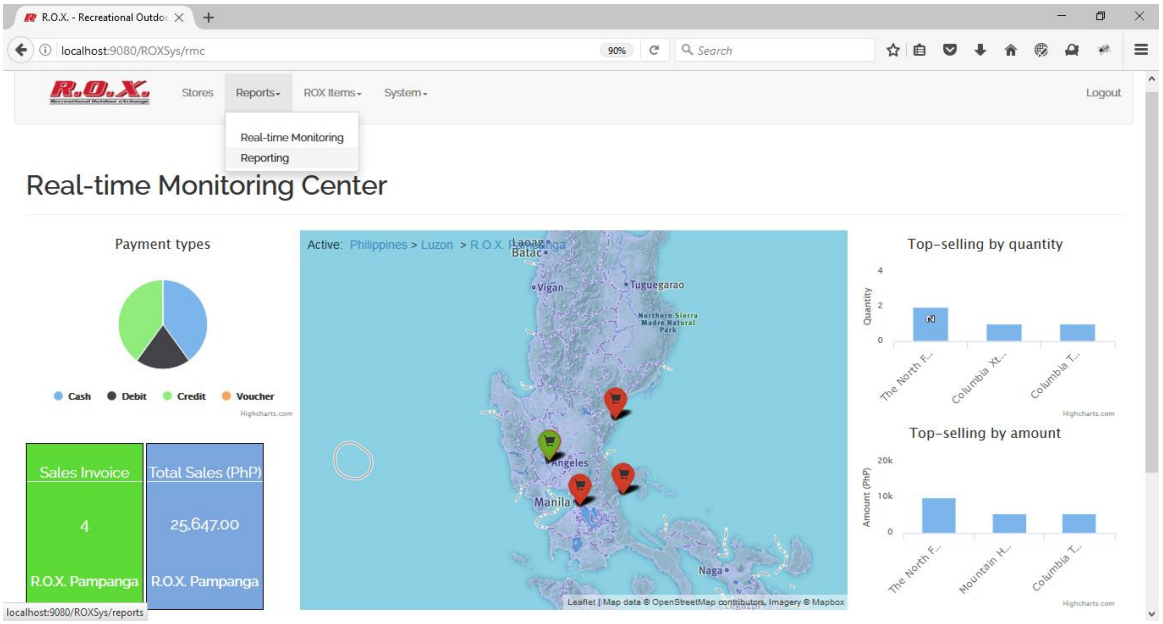


Figure 5.14- Navigating to Reporting page

Inside Reporting page, user selects in what area or store the data will be generated and in between what dates (Figure 5.15). Summary will be generated based on the data selected.

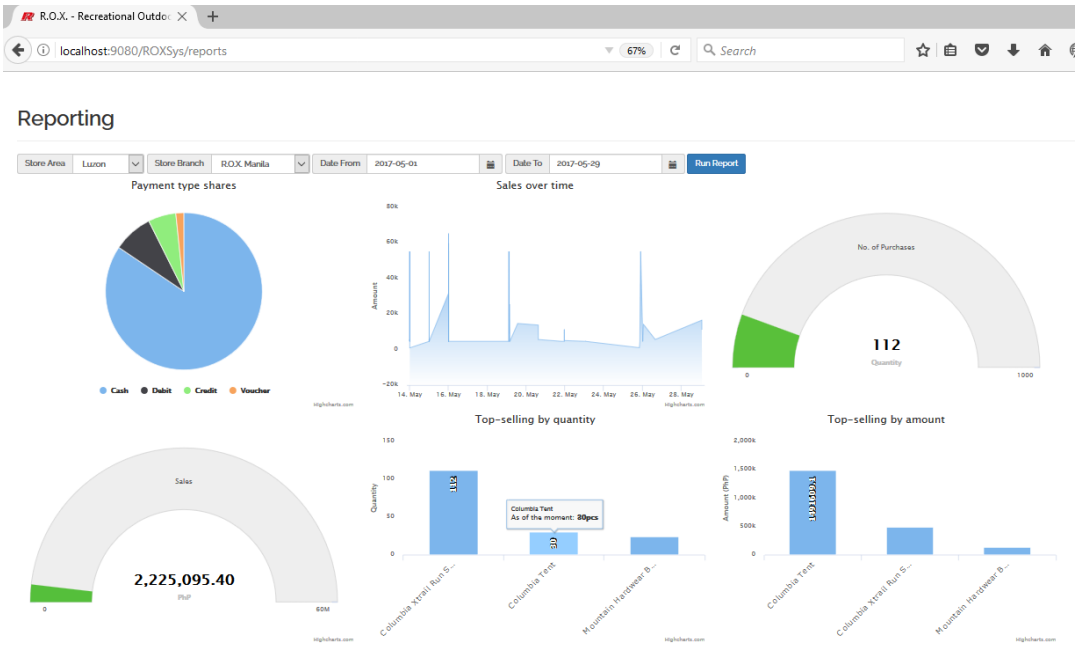


Figure 5.15- Reporting page and its reports

Navigating to Store items page requires the user to click “Store items” sub-menu under ROX Items main menu (Figure 5.16).

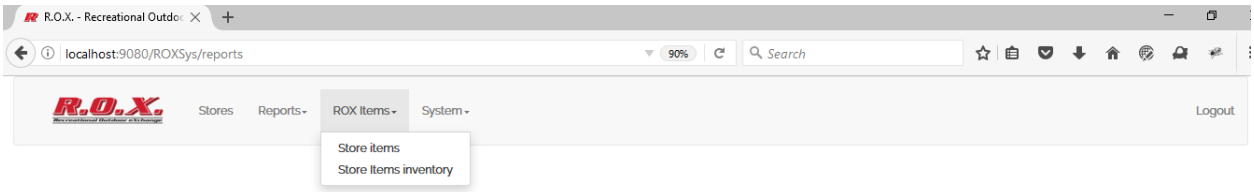


Figure 5.16- Navigating to Store items page

The Store items page stores all the items that are sold in all the stores. No store can sell items that are not listed on this page (Figure 5.17).

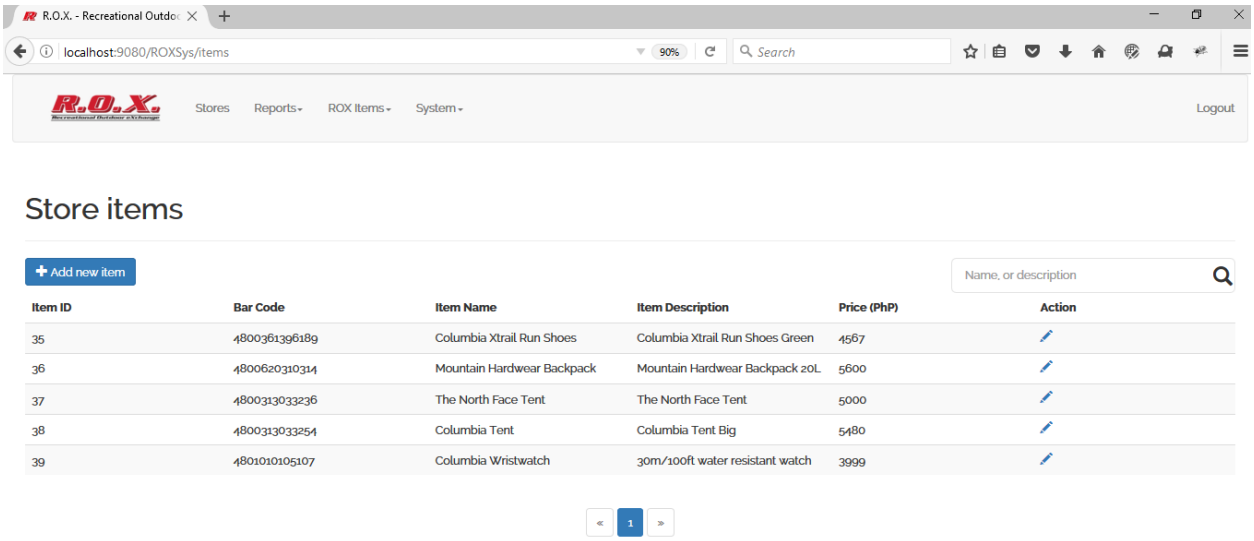


Figure 5.17- List of all store items

Clicking “Add new item” opens a modal where user can fill in the item details (Figure 5.18).

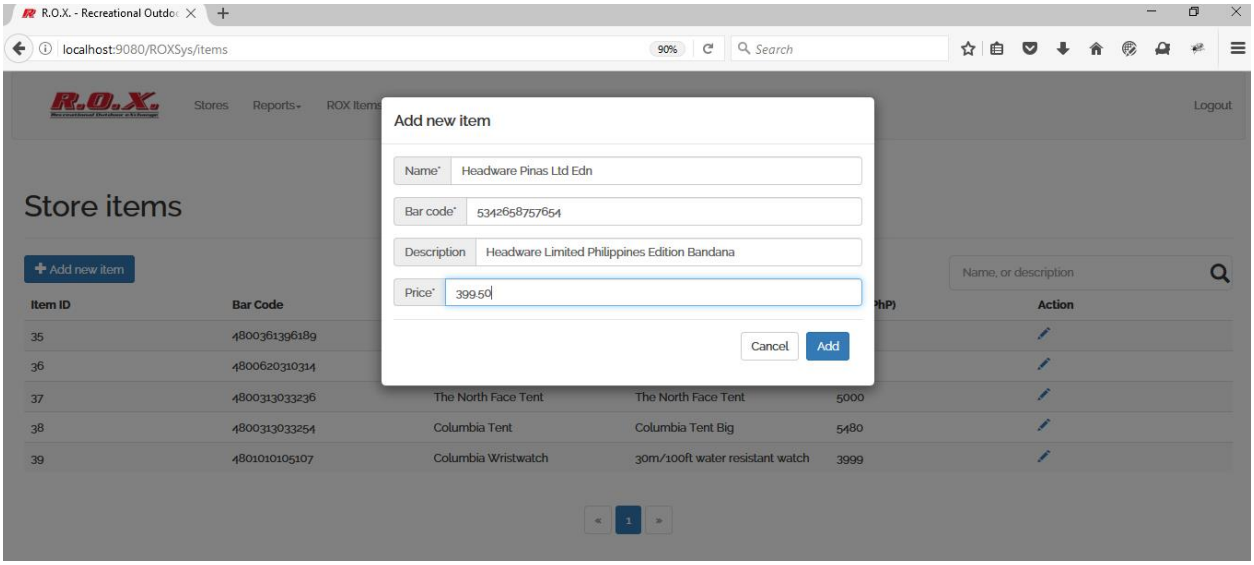


Figure 5.18- Add new item

Clicking Add, the system checks if the values are valid. If they are, the item will be added in the database of items (Figure 5.19).

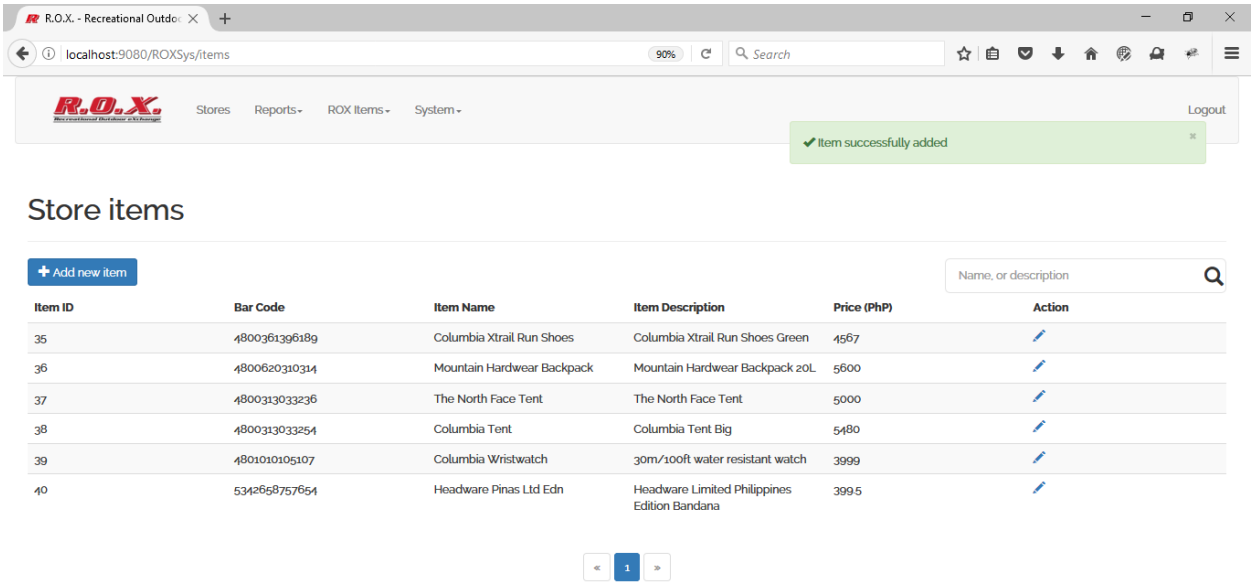


Figure 5.19- Adding new item successful

Editing an item, the user has to click the "Pen" icon under the Action header. A modal opens with the details of the item. The user can then change the details of the item (Figure 5.20).

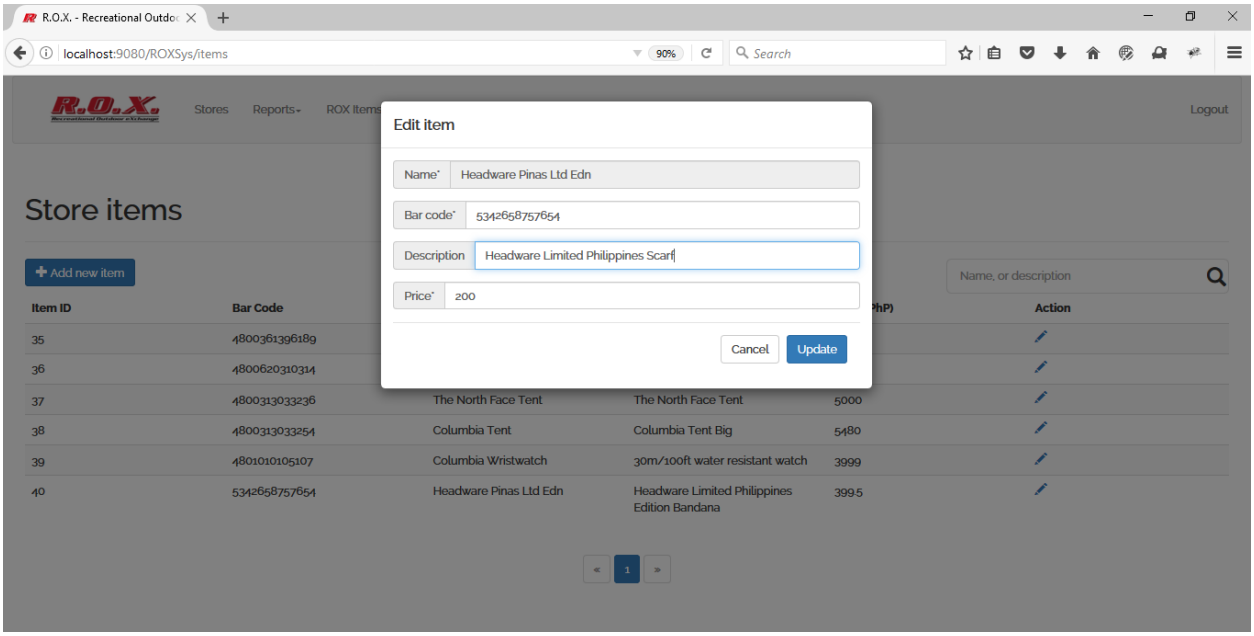


Figure 5.20- Updating item details

Once the user clicks Update, system checks if the data are valid. If they are, the new values will be reflected on the table (Figure 5.21).

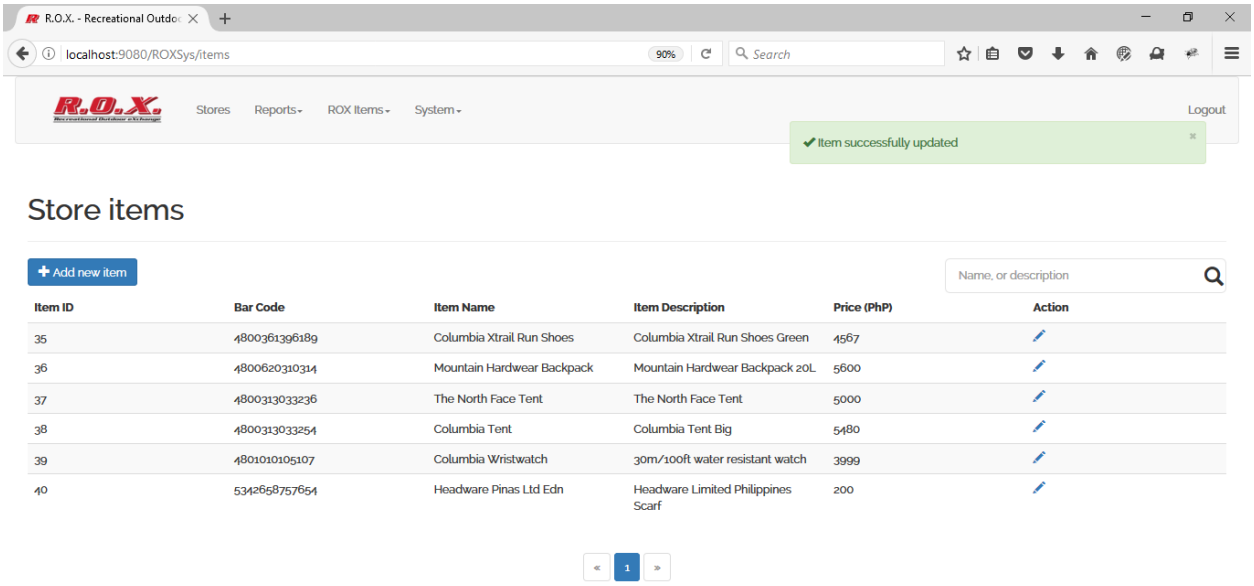


Figure 5.21- Updating item details successful

Navigating to Store items inventory page, the user clicks on the Store items inventory sub-menu under ROX Items main menu. The user will have to select the area and store to see the inventory of that store (Figure 5.22).

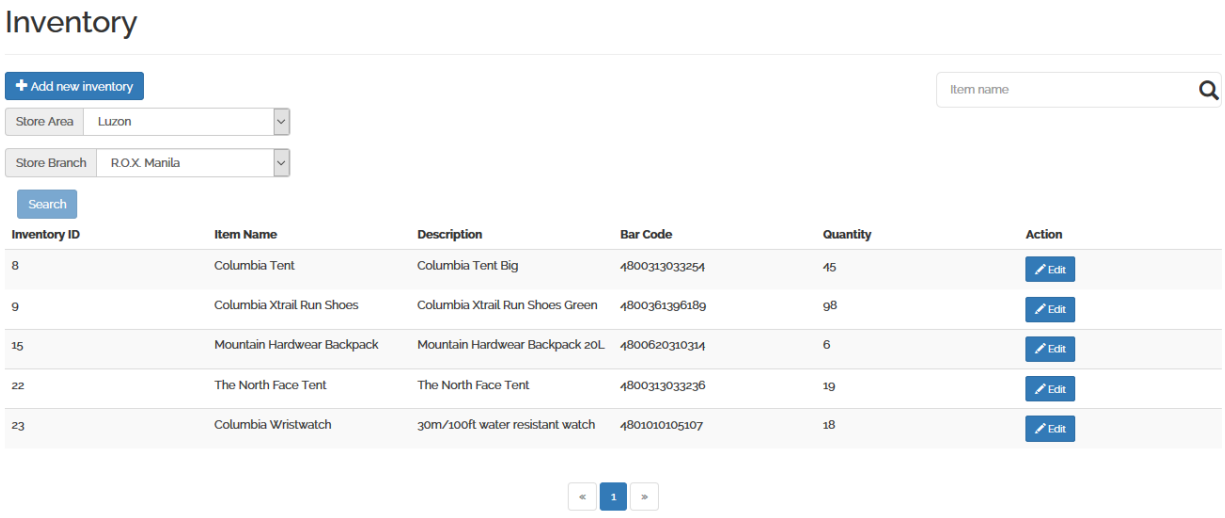


Figure 5.22- Store items inventory

The proprietor can add items to sell to the store by clicking "Add new inventory" button. A modal shows up with the items list that are not yet available to the particular store (Figure 5.23).

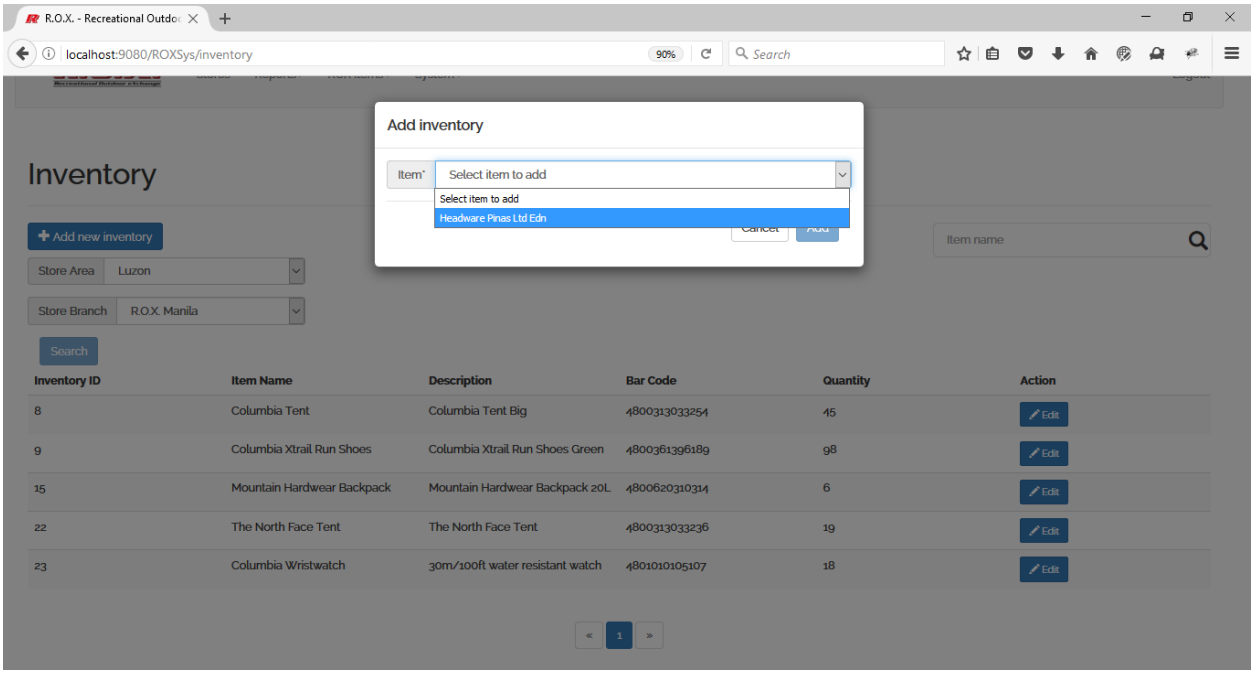


Figure 5.23- Add new store item inventory

Selecting the item and clicking "Add" button adds the item as an inventory of the store (Figure 5.24).

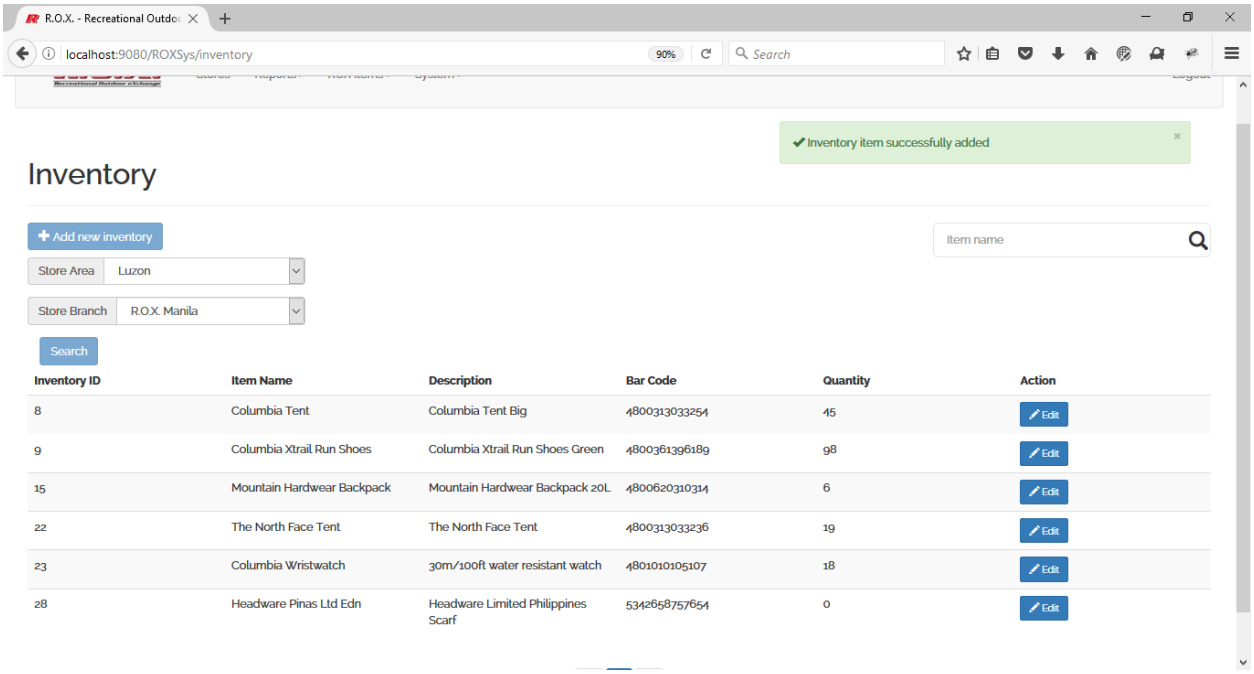


Figure 5.24- Add new store item inventory successful

Updating the quantity of the item, the user clicks "Edit" button. A modal shows up with the current quantity of the item. The user can update it (Figure 5.25).

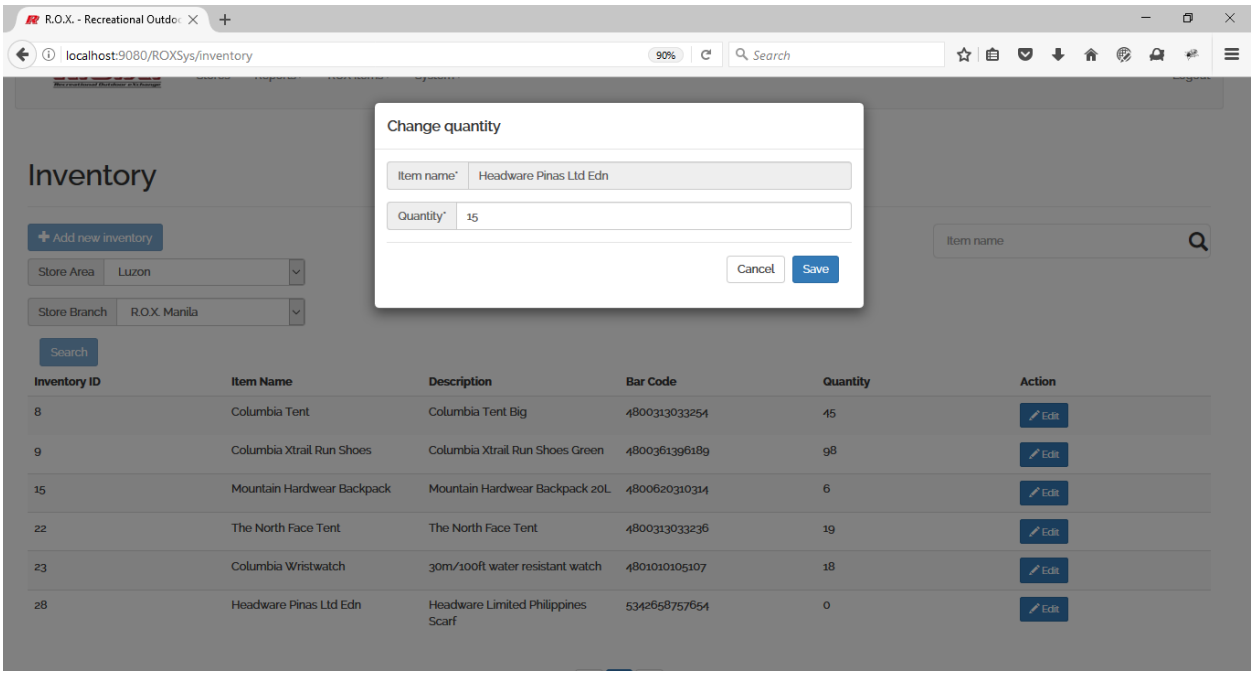


Figure 5.25- Updating inventory item quantity

Once the user clicks "Update" button, quantity will be automatically updated. (Figure 5.26).

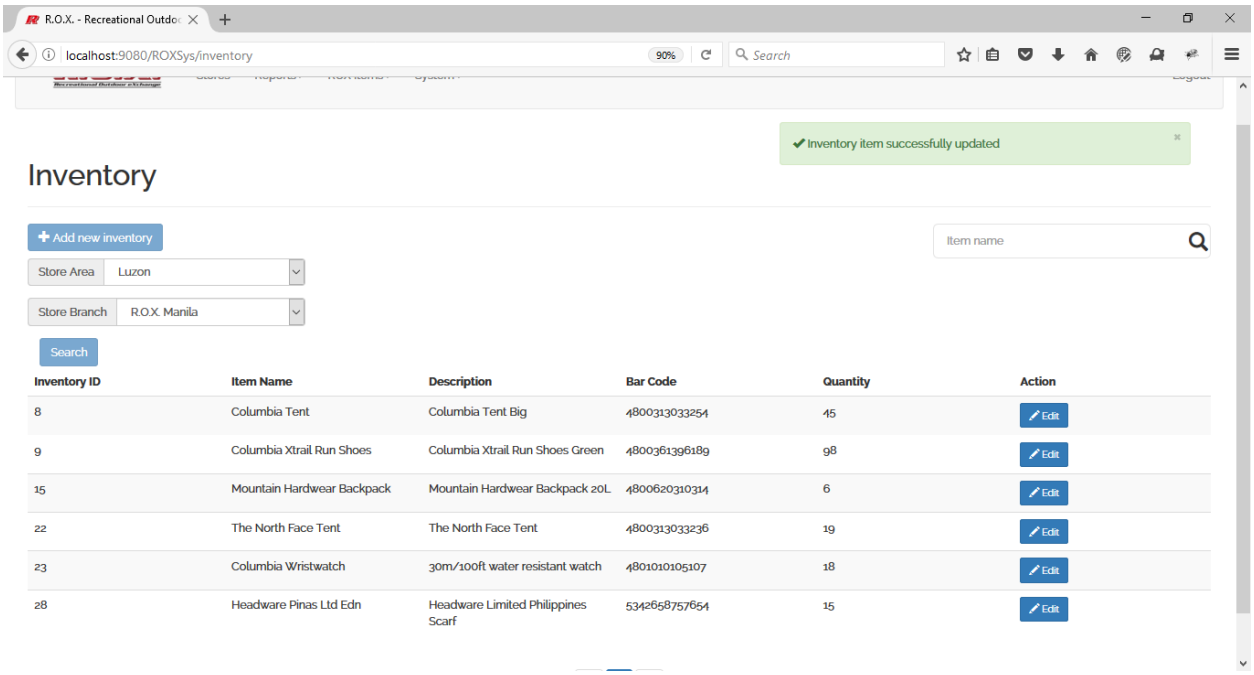


Figure 5.26- Updating inventory item quantity successful

Clicking the System users submenu under System main menu displays the system users (Proprietor sees all the users while Store manager only sees user under the store). (Figure 5.27).

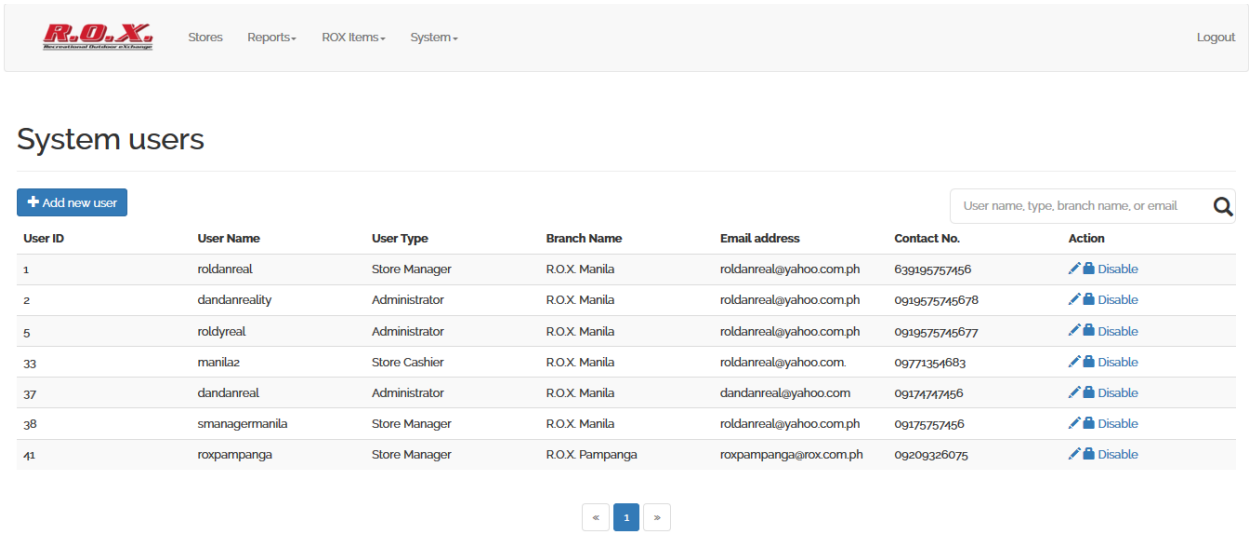


Figure 5.27- System users

Clicking "Add new user" button opens a modal. The user has to fill in the required fields (Figure 5.28).

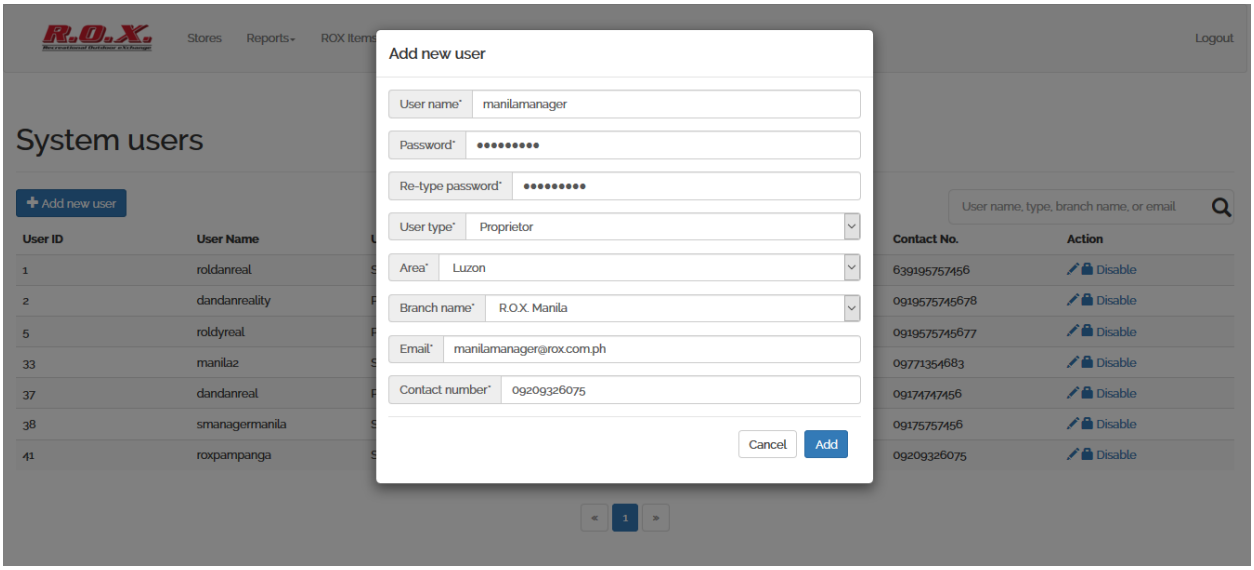


Figure 5.28- Add new user

Once all data are validated, the user will then be added to the database (Figure 5.29).

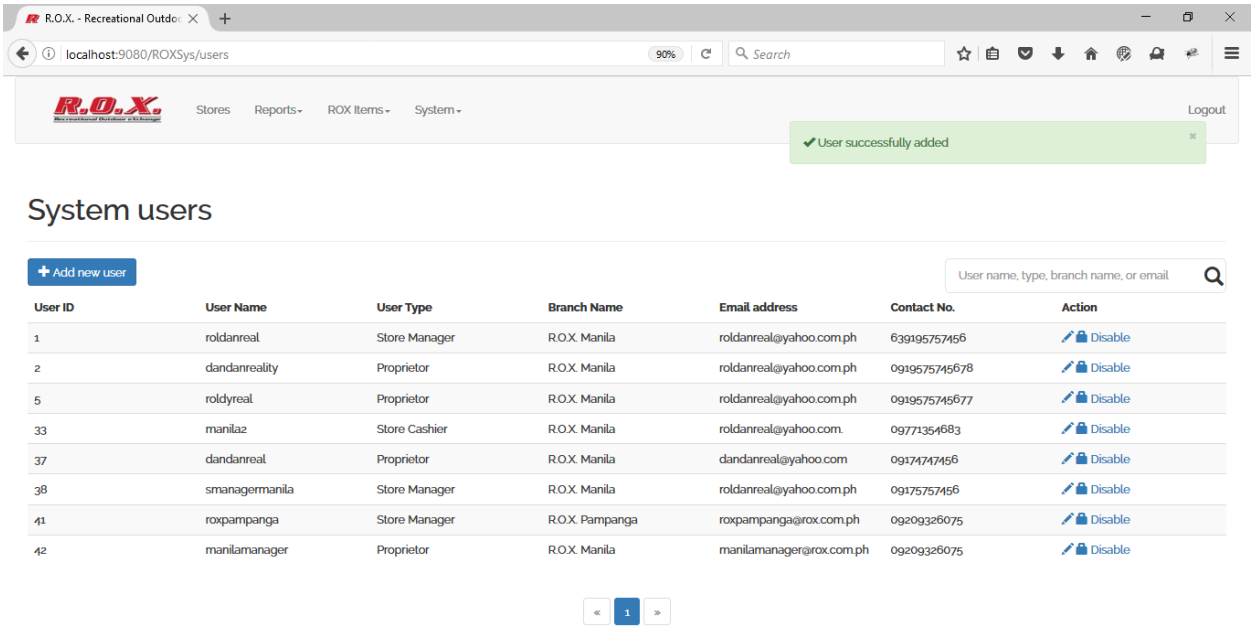


Figure 5.28- Adding new user successful

Updating user details, the user clicks the "Pen" icon. Modal shows with the data of element clicked. User changes the data with new values (Figure 5.29).

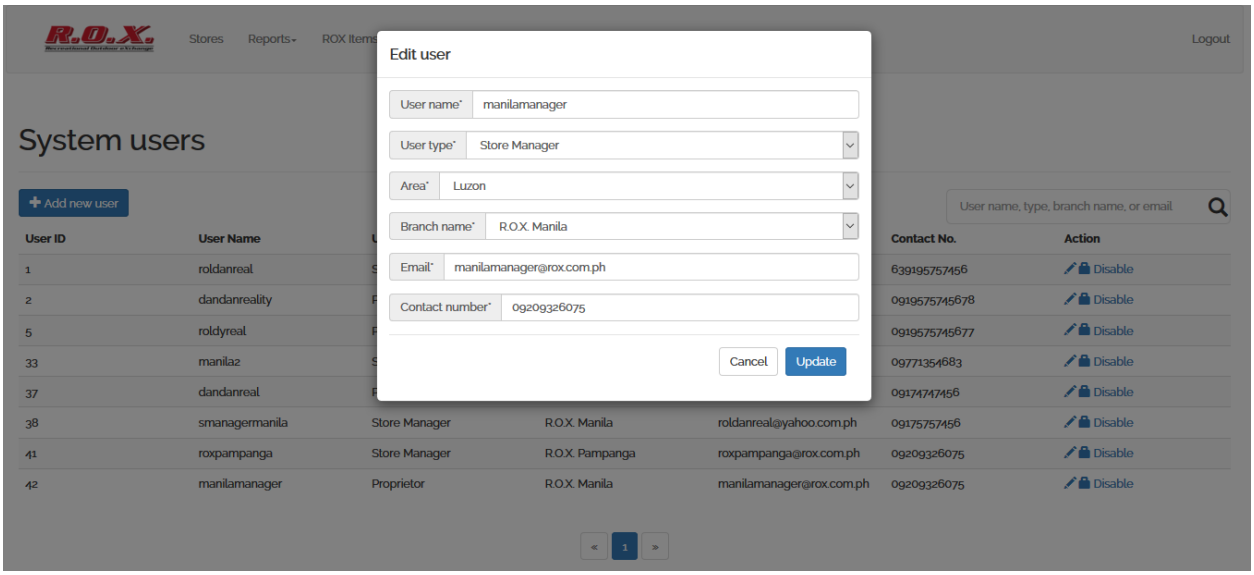


Figure 5.29- Adding new user successful

Once “Update” button is clicked, the system checks if the data are valid. If they are, the new values will reflect on the table (Figure 5.30).

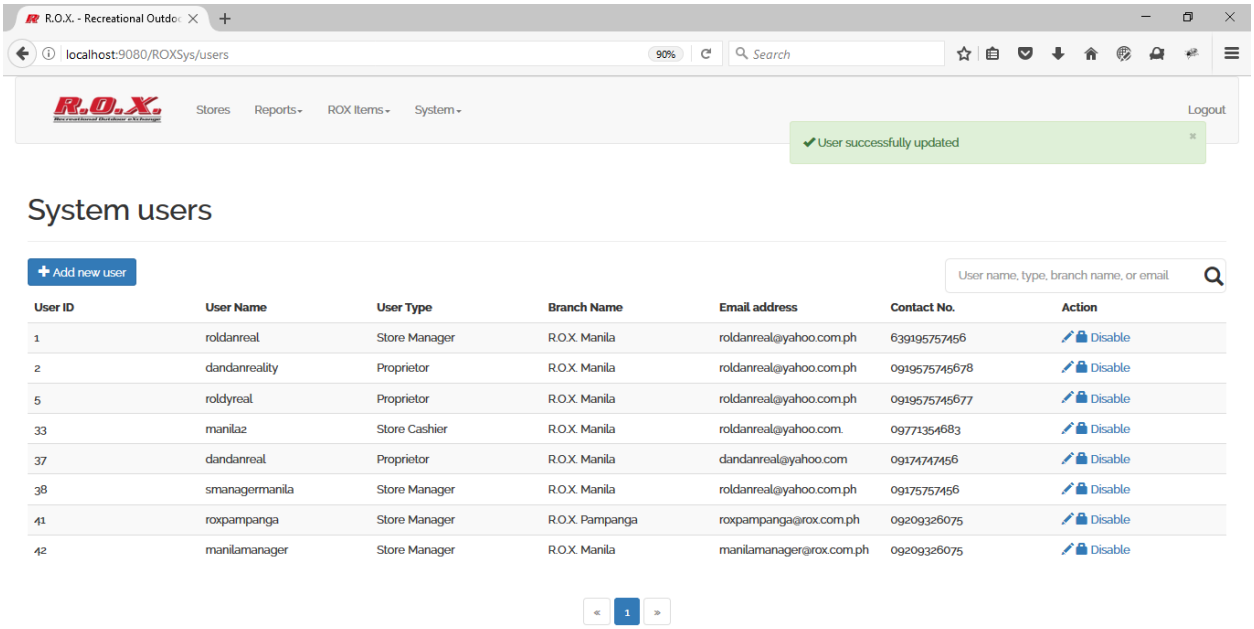


Figure 5.30- Updating user successful

Changing the user password, the user has to click the “Lock” icon under action table header. A modal shows up and the user can type in new password for the system user (Figure 5.31).

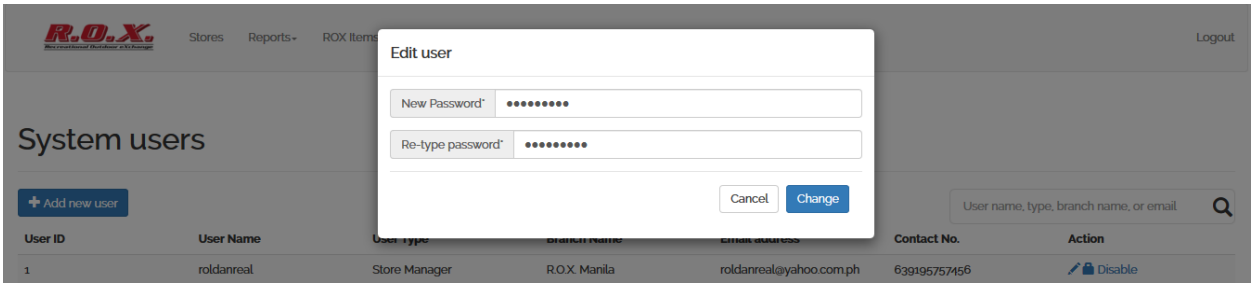


Figure 5.31- Updating user password

Once the two passwords are equal and the user clicks the "Change" button, the password will be changed. Notification is displayed saying that changing of password is successful (Figure 5.32).

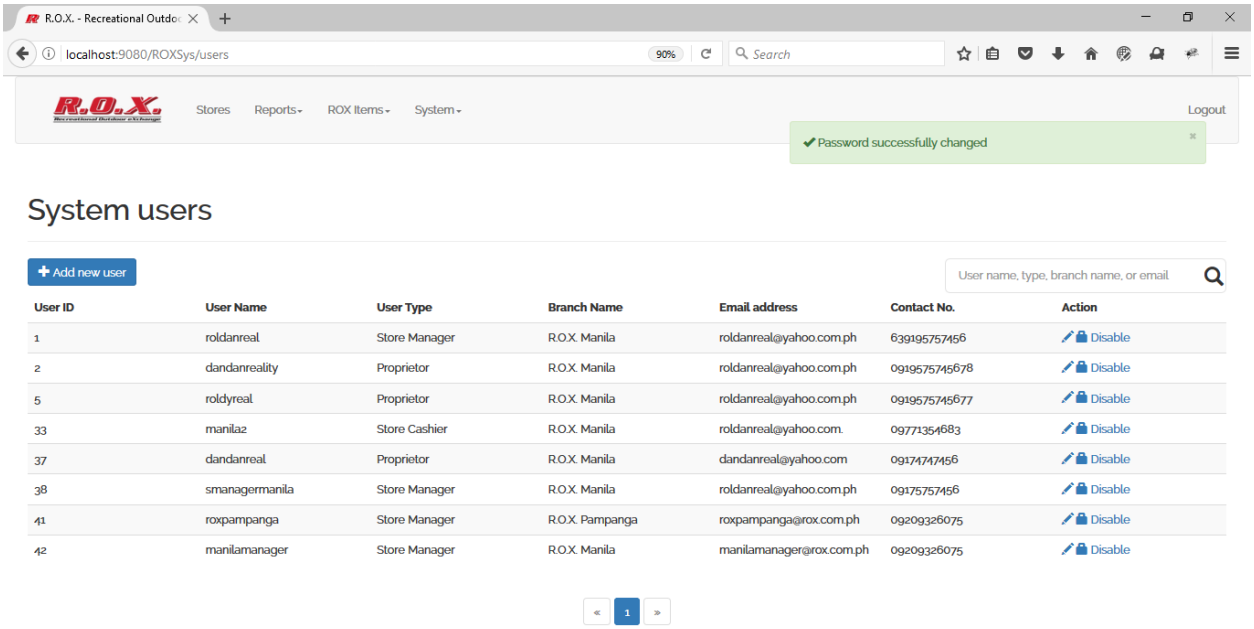


Figure 5.32- Updating user password successful

A user can either be enabled or disabled. An active user can be disabled and inactive user can be enabled. Clicking on "Disable," a confirmation box will ask if the user is wants to disable a system user (Figure 5.33).

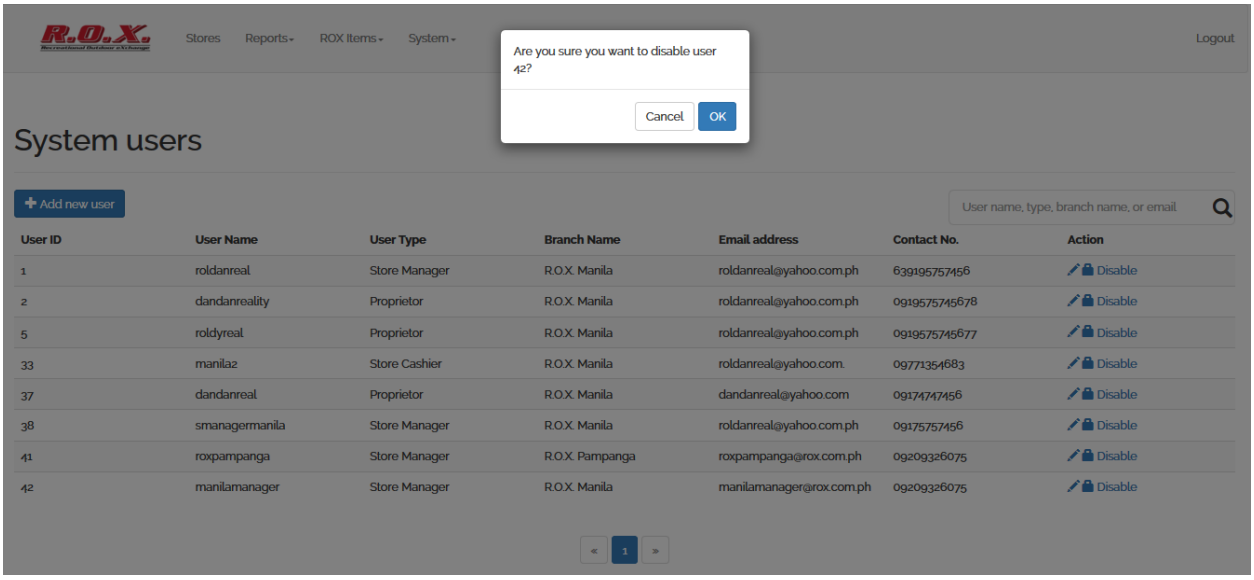


Figure 5.33- Confirmation box for disabling a user

Once the user clicks "OK," the system user will be disabled. Disabled users cannot log in to the system (Figure 5.34).

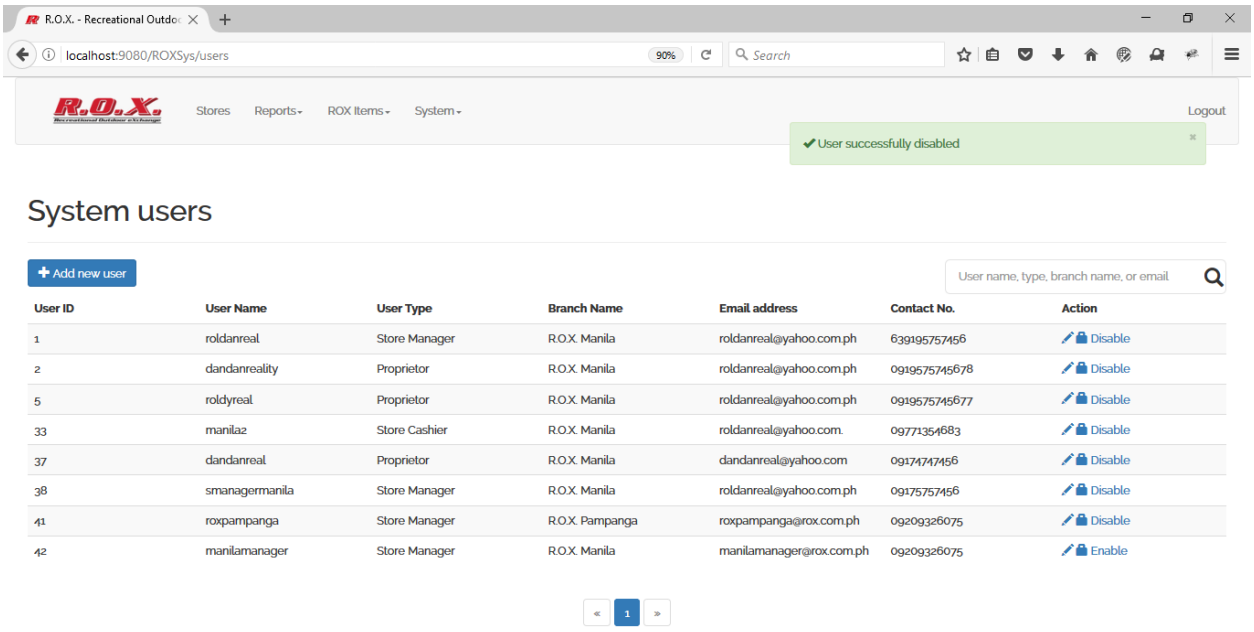


Figure 5.34- Disabling a user successful

To enable a system user, the user has to click on the "Enable" link. A confirmation box shows up as in above. Once the user clicks "OK" button, the system user is now enabled (Figure 5.35).

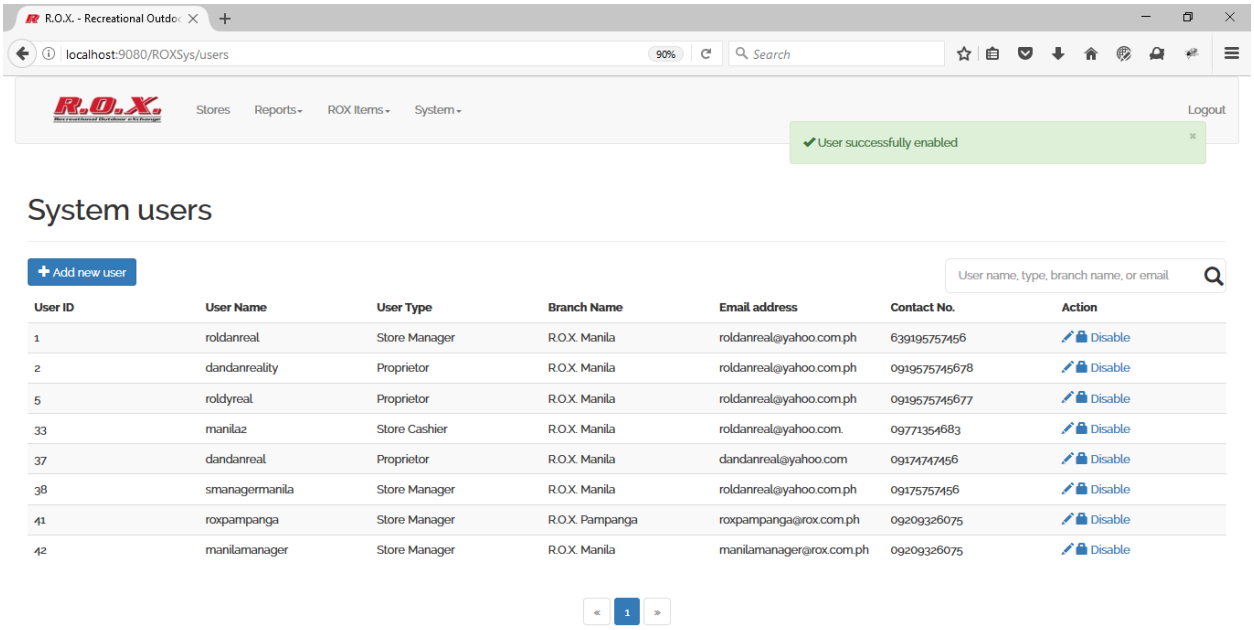


Figure 5.35- Enabling a user successful

Both Store manager and cashier can access the Checkout page. Only items that are added by the proprietor can be sold on each store. Figure 5.36 shows the Checkout page for Manila branch. The right side shows the items sold and the left side shows the sample receipt.

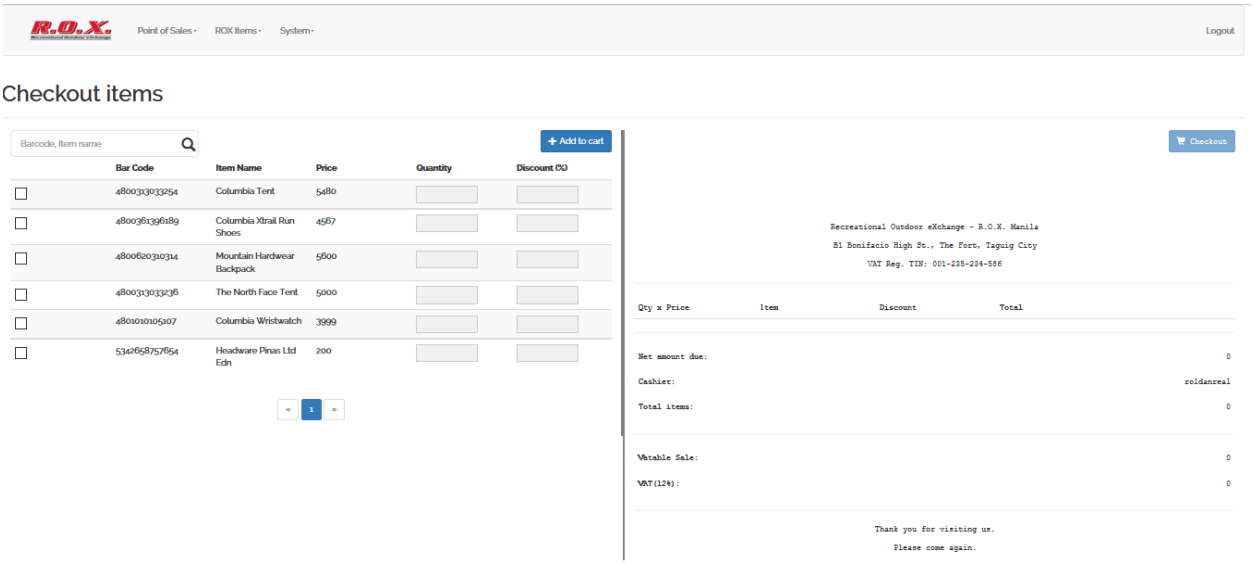


Figure 5.36- Checkout page

There are two type the cashier can select items. One is to search the item on the search box and tick the checkbox, input the quantity of item, discount (if there is any), and click "Add to cart" button. The item will show up on the sample receipt side of the page (Figure 5.37).

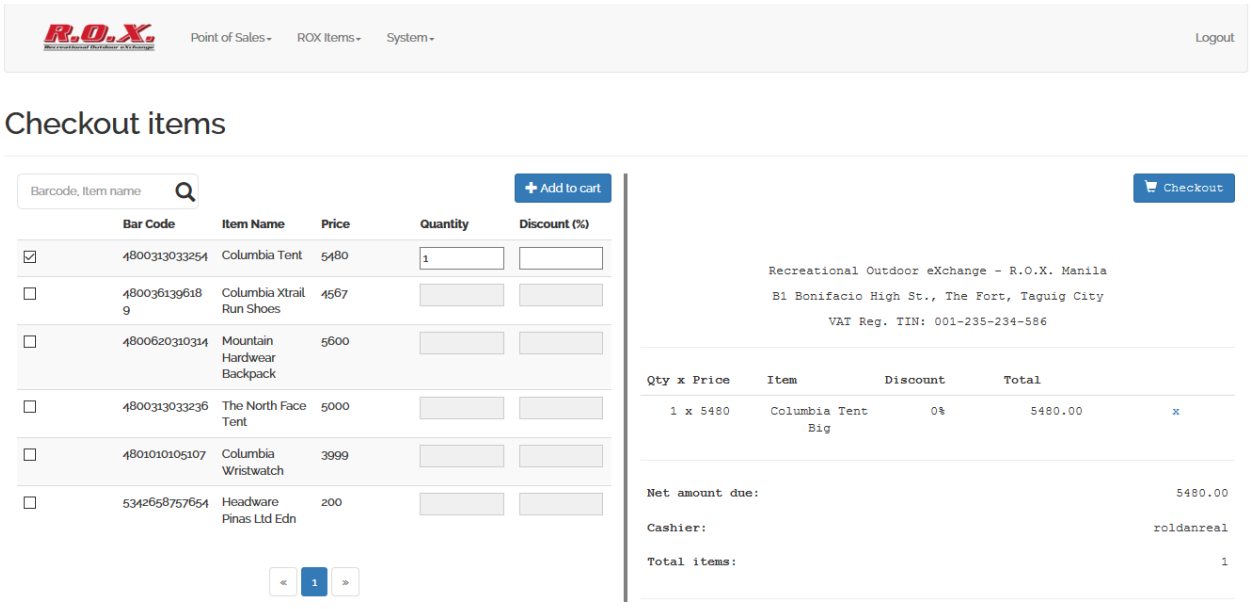


Figure 5.37- Adding item to cart by manual search

Another is to use a barcode scanner. Once the item is searched, a modal shows up and user has to input item quantity and discount, if there is any (Figure 5.38).

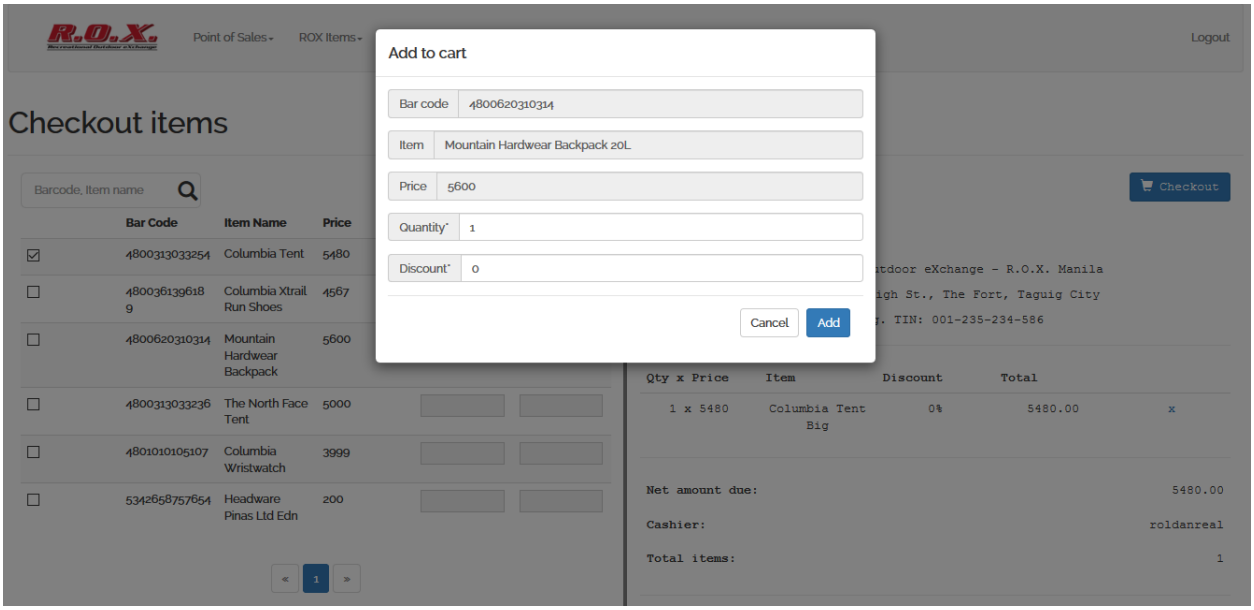


Figure 5.38- Adding item to cart by barcode scanner

Clicking "Add" button adds the item to cart, and will show up at the sample receipt (Figure 5.39).

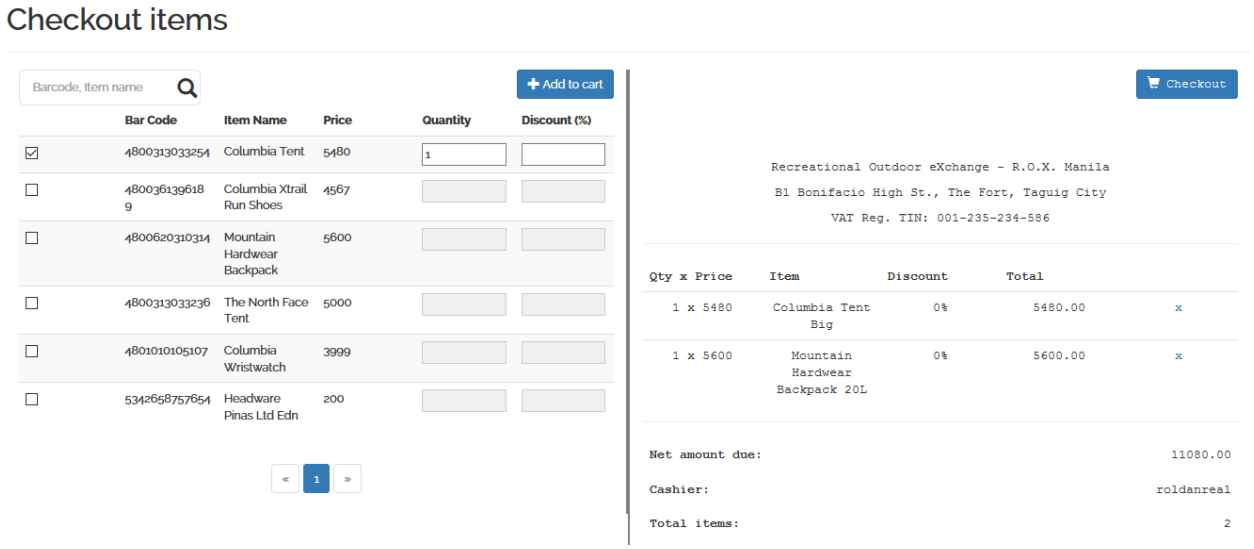


Figure 5.39- Items show up in the sample receipt

Clicking “Checkout” button, a modal for payment details shows up (Figure 5.40).

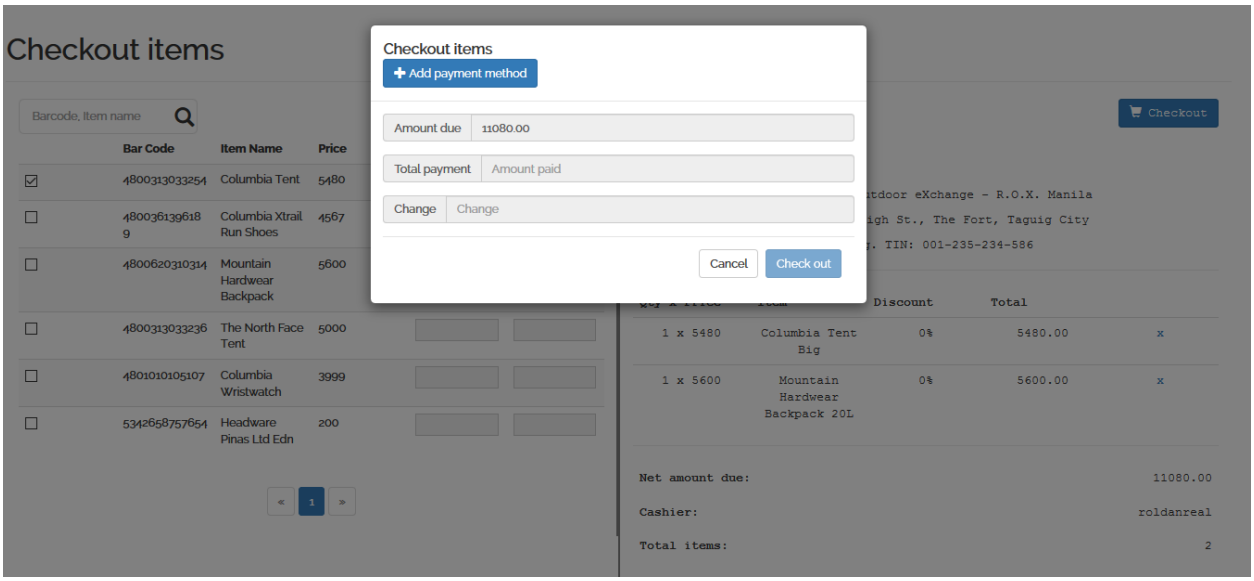


Figure 5.40- Modal for payment

User has to choose which payment type is used. Combination of all payment types is possible (Figure 5.41).

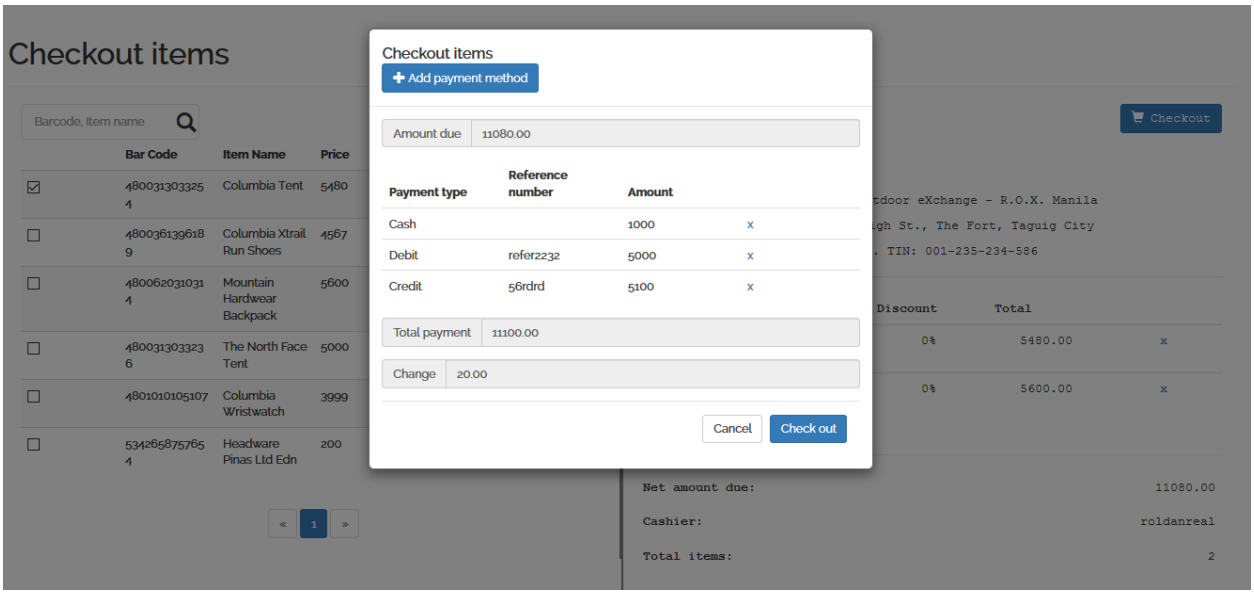
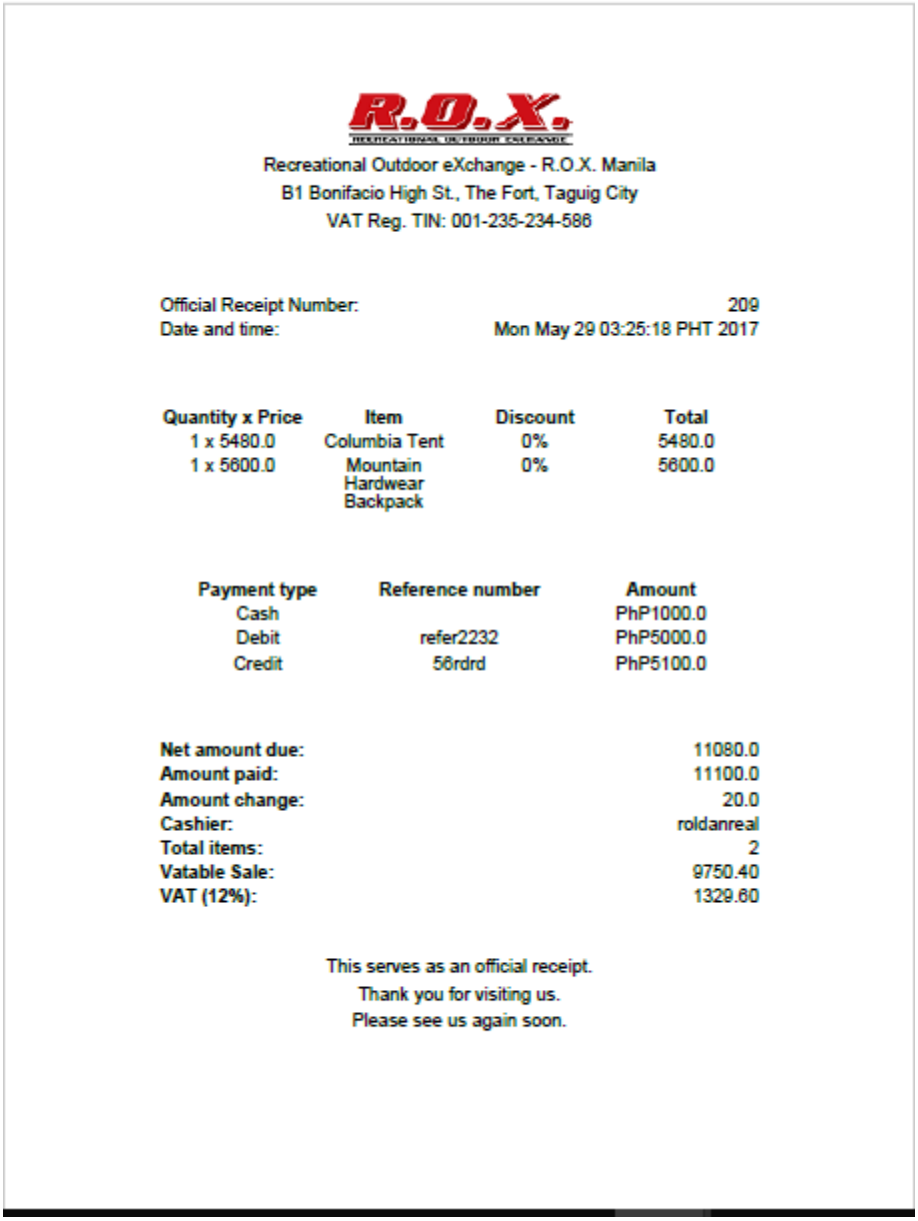


Figure 5.41- Modal for payment with different types of payment

Once “Check out” button is clicked, a receipt will be generated (Figure 5.42).



R.O.X.
RECREATIONAL OUTDOOR EXCHANGE

Recreational Outdoor eXchange - R.O.X. Manila
B1 Bonifacio High St., The Fort, Taguig City
VAT Reg. TIN: 001-235-234-586

Official Receipt Number: 209
Date and time: Mon May 29 03:25:18 PHT 2017

Quantity x Price	Item	Discount	Total
1 x 5480.0	Columbia Tent	0%	5480.0
1 x 5600.0	Mountain Hardwear Backpack	0%	5600.0

Payment type	Reference number	Amount
Cash		PhP1000.0
Debit	refer2232	PhP5000.0
Credit	56rdrd	PhP5100.0

Net amount due: 11080.0
Amount paid: 11100.0
Amount change: 20.0
Cashier: roldanreal
Total items: 2
Vatable Sale: 9750.40
VAT (12%): 1329.60

This serves as an official receipt.
Thank you for visiting us.
Please see us again soon.

Figure 5.42- Receipt generated

Store manager and cashier can do Return item transaction. When a customer comes to the store and returns an item, he/she has to have with him/her the receipt used when buying the item. The store manager/cashier then inputs the receipt number and searches for the items bought (Figure 5.43)

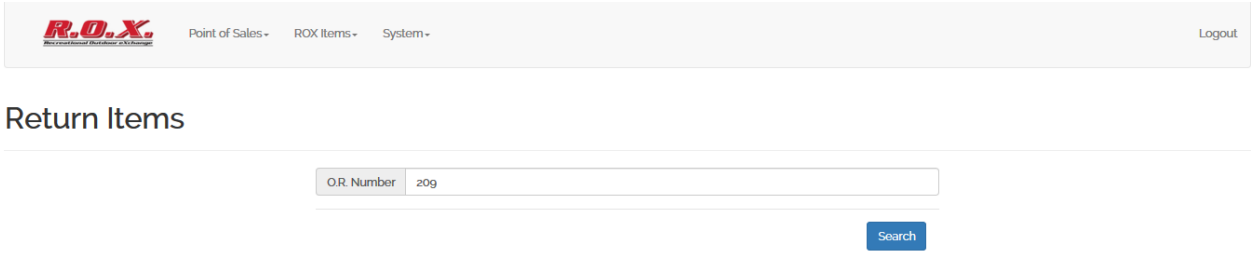


Figure 5.43- Returning item(s) – Search items by receipt number

Once the item(s) is/are searched, store manager/cashier selects the number or items to be returned (Figure 5.44)

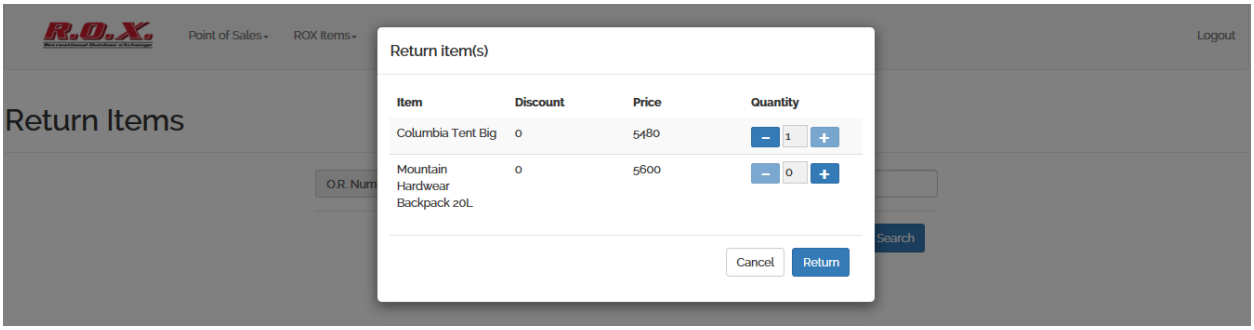


Figure 5.44- Returning item(s) – Select quantity to be returned

When the user clicks "Return" button, a voucher will be generated. This voucher number will be used to pay for the item in exchange of the item returned (Figure 5.45).

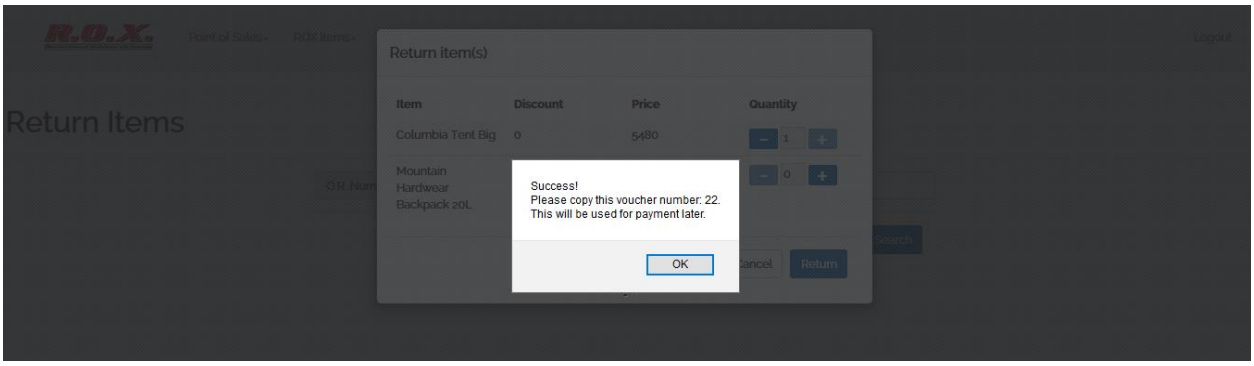


Figure 5.45- Returning item(s) – Voucher number is generated

That voucher number can be used to pay for the new item purchased (Figure 5.46)

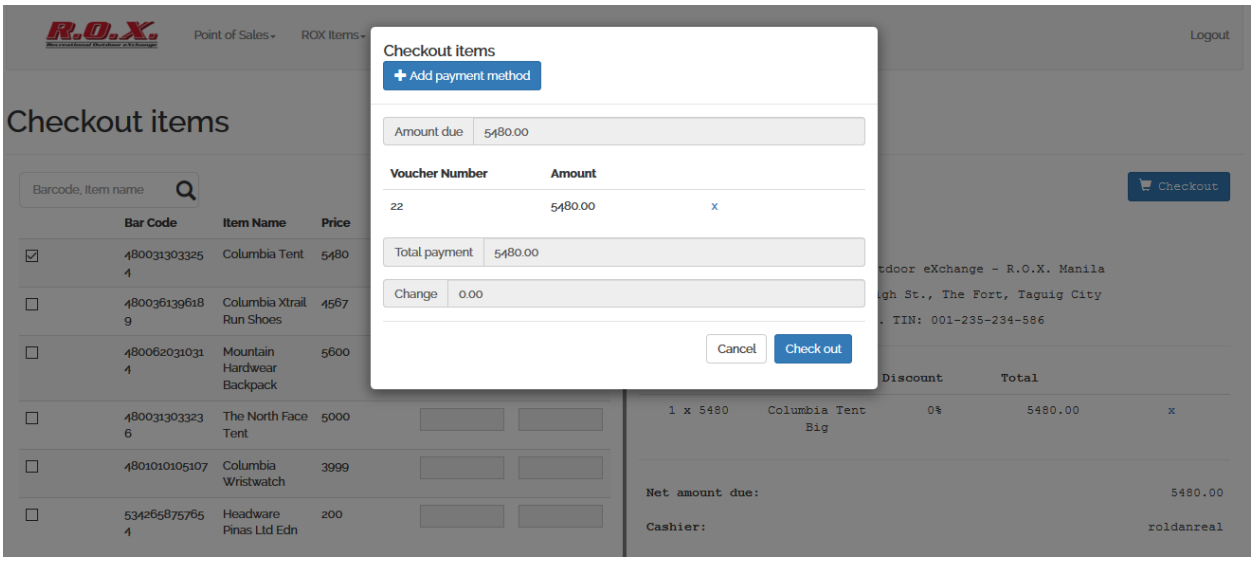


Figure 5.45- Checkout – Paying using returned item voucher

Store managers and cashiers can search for items from other stores where quantity is greater than zero. This is useful if a customer asks for another branch where the item can be bought. The user has to navigate to Store Items inventory sub-menu under ROX Items main menu. Once there, user will see the inventory of the store where they belong. User clicks on "Search from other stores" button and a modal will show up. User has to fill in the required fields (Figure 5.46).

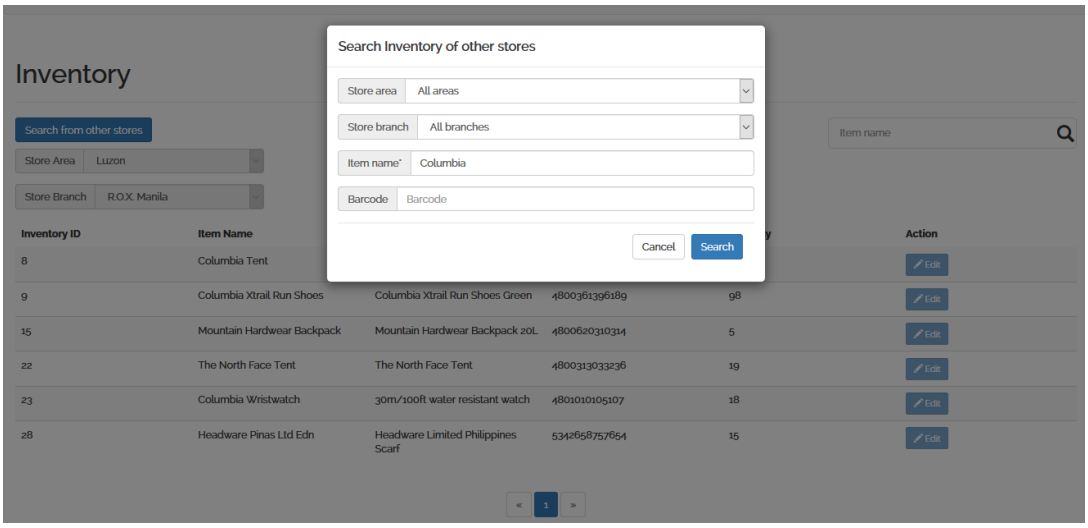


Figure 5.46- Searching for other store's available items

Once “Search” button is clicked, another modal shows up, displaying the items available based on the parameters set by the user (Figure 5.47).

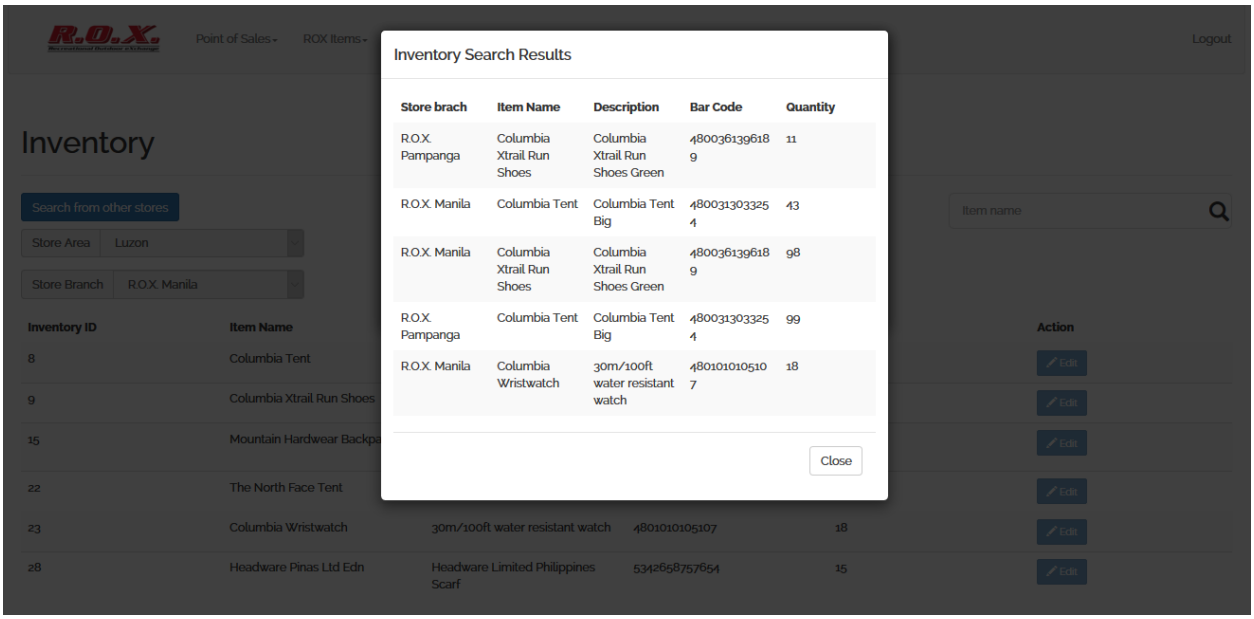


Figure 5.47- Result of searching for other store’s available items

VI. Discussions

ROX-RMS is a web-based point of sale, inventory management, and GIS-based real-time sales and transactions monitoring system for Recreational Outdoor eXchange (R.O.X.). The system has web-based point of sale component that can do items checkout transaction, and return item(s) transaction. Inventory of items is kept track by the system. Monitoring of sales and transactions is done in real-time.

The GIS-based monitoring module helps business owners to see how the business is doing in real-time. The module summarizes the transactions of the day, without waiting for the business day to end before knowing how much money the company makes for the day. It shows the total amount of sales, total invoices made, total payments using cash, credit, debit, and voucher, top-selling items based on quantity, and top-selling items based on amount. It has three views—Philippines, area, and store views. The default view is Philippines, which is the summary of all transactions done by all the stores in the country.

The map has three layers, which is composed of the three areas of the Philippines: Luzon, Visayas, and Mindanao. Clicking any of them zooms to the area and reveals the stores included in the area. The summarized data charts then change to the activities done in that particular area. Clicking the store inside the area then changes the summarized data charts to activities done in that particular store.

The Reporting module helps the owner visualize how the business is doing in particular dates as he or she can generate a report based on the dates selected. The report consists of the total amount of sales, the number of cash, credit, debit, and voucher transactions, time series chart of sales over a period of time, total number of invoices, top-selling item base on amount, and top-selling items based on quantity.

Returning an item is also possible. Doing that, a returned item voucher is generated which can be used to pay for an exchange item. The user will just have to select voucher as payment method and enter the voucher number generated when returning an item.

Searching for available items in another stores is also possible. This is helpful because the store staff can recommend to the customer as to what branch the item is available. So instead of the customer going to a competing store, he/she goes to another branch as the item surely can be found there as of the moment.

There are also disadvantages when using the system. One is that the system does not predict when the store will have big sales. Also, the system does not do suggestions as to what to do when amount of sales is low at a particular time of the day. It is the owner's prerogative on what to do with the data provided to him/her by the system.

Moreover, replenishment of stocks in done outside the system. Central warehouse delivers the stocks to the stores but acquiring the stocks from manufacturers/dealers is done outside the system.

But despite its advantages, the system can do what a regualr point of sale, inventory, and real-time monitoring system can do. The owner has the power to monitor the business in real-time and can do immediate decisions regarding the business.

VII. Conclusions

The system was able to deliver the functions that are expected to it. The point of sale component was able to do checkout and print receipt, and return item and generate item voucher to be used as payment for exchange item. Inventory quantity is deducted after each successful checkout. Real-time monitoring page refreshes every five seconds and reflect the checkout transaction and its corresponding data.

The proprietor is able to see the summary of transactions with three views: Philippines, area, and store views. Each view reflects the summary of data expected to be reflected.

The Reporting page displays the summarized data regarding the parameters selected by the user.

Searching of inventory from another store is helpful as store manager or cashier can redirect the customer to another branch in case the item the customer is looking for is not available in the store.

VIII. Recommendations

ROX-RMS was able to serve its purpose. However, some functions can be considered as future additions to this project. One is that the reports can be exported as excel (CSV) and PDF formats. It will be useful for owners if they wish to have hard copies of the reports generated. Another would be in the process wherein a user returns an item, the store manager should first check if the item is returnable or not. If the item is returnable, only then can the item be returned using the system to be able to generate a voucher.

IX. Bibliography

[1] Primer Group of Companies. (n.d.). *Who We Are.* Retrieved April 11, 2017, from http://www.primergrp.com/primer-group/who-we-are/

[2] Primer Group of Companies (n.d.). *Concept Stores*. Retrieved April 11, 2017, from http://www.primergrp.com/concept-stores/

[3] Recreational Outdoor eXchange (n.d.) *R.O.X. Recreational Outdoor eXchange.* Retrieved April 11, 2017 from http://www.hobiechallenge.ph/12/recreational-outdoor-exchange/

[4] Palma, Ruby (2017, April 11). Personal communication.

[5] Recreational Outdoor eXchange (n.d.) *About Us.* Retrieved April 11, 2017 from http://rox.com.ph/?page_id=2

[6] Shaw, H. (2012). Food access, diet and health in the UK: an empirical study of Birmingham. *British Food Journal*, *114*(4), 598-616.

[7] Aggarwal, S. S. (2009). Retail management. *Journal of Business and Retail Management Research (JBRMR),* 3(2)

[8] Hartoyo, Daryanto, K. H., Arifin, B. (2015). The effects of ICT adoption on marketing capabilities and business performance of Indonesian SMEs in the fashion industry. *Journal of Business and Retail Management Research (JBRMR),* 100-111

[9] Sahay, B.S., & Ranjan, J. (2008). Real time business intelligence in supply chain analytics. *Information Management & Computer Security,* 16(1), 28-48, DOI 10.1108/09685220810862733

[10] Plomp, M.G.A., Rijn, G. v., Batenburg, R.S. (2012). Chain digitisation support by point-of-sale systems: an analysis of the Dutch product software market. *International Journal of Information Technology and Management*, 11(4), 257-272

[11] Shah, H.N., & Raykundaliya, N. (2010). Optimal inventory policies for Weibull deterioration under trade crdit in declining market. *Journal of Business and Retail Management Research (JBRMR),* 4(2)

[12] Reddy, M. & Sawant, V. (2014). Remote monitoring and control system for DC motor using Zigbee protocol. *International Journal of Application or Innovation in Engineering & Management (IJAIEM),* 3(4), 374-379

[13] García, L. R., Elorza, P. B., Rodríguez-Bermejo, J., & Robla, J. I. (2007). Review: Monitoring the intermodal, refrigerated transport of fruit using sensor networks. *Spanish Journal of Agricultural Research*, (2), 142-156.

[14] Singhal, Z., & Gujral, K.R. (2012). Anytime anywhere-remote monitoring of attendance system based on RFID using GSM Network. *International Journal of Computer Applications,* 39(3), 0975-8887

[15] Azaz, L. (2011). The use of Geographic Information System (GIS) in business. *International Conference on Humanities and Economics,* 299-304, Retrieved from http://psrcentre.org/images/extraimages/42.%201211200.pdf, on 22nd August 2016

[16] Smith, K. L., MacGregor, R., & Johnson, W. G. (2005). *U.S. Patent No. 6,868,396*. Washington, DC: U.S. Patent and Trademark Office.

[17] Srinivas, Hari (n. d.). SMEs. *What are SMEs?* Retrieved March 21, 2016 from http://www.gdrc.org/sustbiz/what-are-smes.html

[18] Republic Act No, 8289. *Magna Carta for Small Enterprises.* Retrieved March 21, 2016 from http://www.chanrobles.com/republicactno8289.htm#.Vu9uvJx97IU

[19] Investopedia.com (n. d.) *Point of Sale – POS.* retrieved March 14, 2016 from http://www.investopedia.com/terms/p/point-of-sale.asp

[20] Techtarget: What is (n.d.). *Point-of-sale terminal (POS terminal).* Retrieved March 14, 2016 from http://whatis.techtarget.com/definition/point-of-sale-terminal-POS-terminal

[21] PC Mag Encyclopedia (n. d.) *Definition of: Web application*. Retrieved March 14, 2016 from http://www.pcmag.com/encyclopedia/term/54272/web-application

[22] Acunetix (n. d.). *Web Applications: What are They? What of Them?.* Retireved March 14, 2016 from http://www.acunetix.com/websitesecurity/web-applications/

[23] Taino Systems (n. d.). *What are Retail Management Information Systems?* Retrieved March 14, 2016 from http://tainosystems.com/en/blog/103-what-are-retail-management-information-systems

[24] Springboard Retail (n. d.). *What is a Retail Management System (RMS)?* Retrieved March 14, 2016 from http://blog.springboardretail.com/what-is-a-retail-management-system/

[25] Barcodes, Inc. (n.d.). *What is Inventory Management?* Retrieved March 14, 2016 from https://www.barcodesinc.com/articles/what-is-inventory-management.htm

[26] Wasp Barcode. *What is Inventory Management Software?* Retrieved March 14, 2016 from http://www.waspbarcode.com/buzz/what-is-inventory-management-software/

[27] Techopedia. *Remote Monitoring and Management.* Retrieved March 14, 2016 from https://www.techopedia.com/definition/28529/remote-monitoring-and-management-rmm

[28] National Geographic. *GIS (Geographic Information System).* Retrieved March 17, 2016 from http://education.nationalgeographic.org/encyclopedia/geographic-information-system-gis/

[29] GIS Lounge. *What is GIS?* Retrieved March 17, 2016 from https://www.gislounge.com/what-is-gis/

[30] ESRI. *How GIS Works?* Retrieved March 17, 2016 from http://www.esri.com/what-is-gis/howgisworks

[31] Database Dir. *What is RDBMS?* Retrieved March 17, 2016 from http://www.databasedir.com/what-is-rdbms/

[32] The Free Dictionary by Farlex. *DBMS*. Retrieved March 17, 2016 from http://encyclopedia2.thefreedictionary.com/DBMS

X. Appendix

A. Source Codes

AppConfig.java

```
package edu.up.cas.sp.config;
import
org.springframework.context.MessageSource;
import
org.springframework.context.annotation.Bean;
import
org.springframework.context.annotation.Compone
ntScan;
import
org.springframework.context.annotation.Configu
ration;
import
org.springframework.context.support.ResourceBu
ndleMessageSource;
import
org.springframework.web.servlet.ViewResolver;
import
org.springframework.web.servlet.config.annotat
ion.EnableWebMvc;
import
org.springframework.web.servlet.config.annotat
ion.ResourceHandlerRegistry;
import
org.springframework.web.servlet.config.annotat
ion.WebMvcConfigurerAdapter;
import
org.springframework.web.servlet.view.InternalR
esourceViewResolver;
import
org.springframework.web.servlet.view.JstlView;

@Configuration
@EnableWebMvc
@ComponentScan(basePackages = "edu.up.cas.sp")
public class AppConfig extends
WebMvcConfigurerAdapter{

    @Bean
    public ViewResolver viewResolver() {
        InternalResourceViewResolver
viewResolver = new
InternalResourceViewResolver();

viewResolver.setViewClass(JstlView.class);
        viewResolver.setPrefix("/WEB-
INF/views/");
        viewResolver.setSuffix(".jsp");
        return viewResolver;
    }

    @Bean
    public MessageSource messageSource() {
        ResourceBundleMessageSource
messageSource = new
ResourceBundleMessageSource();
        messageSource.setBasename("messages");
        return messageSource;
    }

    /**
     * This method is used to locate static
resources e.g. js, css, img, etc in jsp file
     *
     */
    @Override
    public void
addResourceHandlers(ResourceHandlerRegistry
registry) {

registry.addResourceHandler("/resources/**")
        .addResourceLocations("/resources/");
    }
}
```

AppInitializer.java

```
package edu.up.cas.sp.config;
import
org.springframework.web.servlet.support.AbstractAnnotati
onConfigDispatcherServletInitializer;
public class AppInitializer extends
AbstractAnnotationConfigDispatcherServletInitializer {

    @Override
    protected Class<?>[] getRootConfigClasses() {
        return new Class[] { AppConfig.class };
    }

    @Override
    protected Class<?>[] getServletConfigClasses() {
        return null;
    }

    @Override
    protected String[] getServletMappings() {
        return new String[] { "/" };
    }

        }
```

HibernateConfiguration.java

```
package edu.up.cas.sp.config;

import java.util.Properties;

import javax.sql.DataSource;



import org.hibernate.SessionFactory;
```

```
import
org.springframework.beans.factory.annotation.Autowired;

import org.springframework.context.annotation.Bean;

import
org.springframework.context.annotation.ComponentScan;

import
org.springframework.context.annotation.Configuration;

import
org.springframework.context.annotation.PropertySource;

import org.springframework.core.env.Environment;

import
org.springframework.jdbc.datasource.DriverManagerDataSou
rce;

import
org.springframework.orm.hibernate4.HibernateTransactionM
anager;

import
org.springframework.orm.hibernate4.LocalSessionFactoryBe
an;

import
org.springframework.transaction.annotation.EnableTransac
tionManagement;


@Configuration

@EnableTransactionManagement

@ComponentScan({ "edu.up.cas.sp.config" })

@PropertySource(value = {
"classpath:application.properties" })

public class HibernateConfiguration {


    @Autowired

    private Environment environment;


    @Bean

    public LocalSessionFactoryBean sessionFactory() {

        LocalSessionFactoryBean sessionFactory = new
LocalSessionFactoryBean();

        sessionFactory.setDataSource(dataSource());

        sessionFactory.setPackagesToScan(new String[] {
"edu.up.cas.sp.model" });


sessionFactory.setHibernateProperties(hibernatePropertie
s());

        return sessionFactory;

     }


    @Bean

    public DataSource dataSource() {

        DriverManagerDataSource dataSource = new
DriverManagerDataSource();


dataSource.setDriverClassName(environment.getRequiredPro
perty("jdbc.driverClassName"));


dataSource.setUrl(environment.getRequiredProperty("jdbc.
url"));


dataSource.setUsername(environment.getRequiredProperty("
jdbc.username"));


dataSource.setPassword(environment.getRequiredProperty("
jdbc.password"));

        return dataSource;

    }


    private Properties hibernateProperties() {

        Properties properties = new Properties();

        properties.put("hibernate.dialect",
environment.getRequiredProperty("hibernate.dialect"));

        properties.put("hibernate.show_sql",
environment.getRequiredProperty("hibernate.show_sql"));

        properties.put("hibernate.format_sql",
environment.getRequiredProperty("hibernate.format_sql"))
;

        return properties;

    }
```

```
    @Bean
    @Autowired
    public HibernateTransactionManager
transactionManager(SessionFactory s) {
       HibernateTransactionManager txManager = new
HibernateTransactionManager();
       txManager.setSessionFactory(s);
       return txManager;
    }
}
```

AppController.java

```
package edu.up.cas.sp.controller;

import
org.springframework.beans.factory.annotation.Autowired;

import org.springframework.context.MessageSource;

import org.springframework.stereotype.Controller;

import
org.springframework.web.bind.annotation.RequestMapping;

import
org.springframework.web.bind.annotation.RequestMethod;

@Controller

public class AppController {

    @Autowired

    MessageSource messageSource;



    /*

     * This method will redirect the page to the home
page.

     */

    @RequestMapping(value = { "/" , "home"}, method =
RequestMethod.GET)

    public String goToHomePage() {

        return "home";
    }
}
```

AreaController.java

```
package edu.up.cas.sp.controller;

import java.util.List;

import javax.servlet.http.HttpServletRequest;

import
org.springframework.beans.factory.annotation.Autowired;

import org.springframework.stereotype.Controller;

import
org.springframework.web.bind.annotation.PathVariable;

import
org.springframework.web.bind.annotation.RequestMapping;

import
org.springframework.web.bind.annotation.RequestMethod;

import
org.springframework.web.bind.annotation.ResponseBody;

import com.google.gson.Gson;

import edu.up.cas.sp.model.Area;

import edu.up.cas.sp.service.AreaService;

@Controller

public class AreaController {

         @Autowired

    AreaService service;



    /*

     * This method will return all areas to the Areas
page.

     */

    @RequestMapping(value = { "/get-areas" }, method =
RequestMethod.GET)

    @ResponseBody

    public String getAreas() {

        List<Area> areas = service.findAllArea();

        Gson gson = new Gson();
```

```
        String json = gson.toJson(areas);
        return json;
    }

    /*
     * This method will add new item to DB.
     */
    @RequestMapping(value = { "/add-area" }, method =
RequestMethod.GET)
    @ResponseBody
    public String saveItem(HttpServletRequest request) {
         String areaName =
request.getParameter("areaName");

         Area area = new Area();
         area.setAreaName(areaName);

         //Save area
         service.saveArea(area);

        Gson gson = new Gson();
        return gson.toJson(area);

    }

    /*
     * This method will update area in the DB.
     */
    @RequestMapping(value = { "/update-area" }, method =
RequestMethod.GET)
    @ResponseBody
    public void UpdateArea(HttpServletRequest request) {
         String areaId =
request.getParameter("areaId");
         String areaName =
request.getParameter("areaName");

         System.out.println("area id: " + areaId +
"\narea name: " + areaName);

         Area area = new Area();
         area.setAreaId(Integer.parseInt(areaId));
         area.setAreaName(areaName);

         service.updateArea(area);
    }

    /*
     * This method will delete an area by its Area Id.
     */
    @RequestMapping(value = { "/delete-{areaId}-area" })
    @ResponseBody
    public void deleteItem(@PathVariable Integer areaId)
{
        service.deleteArea(areaId);
    }
}
```

CheckoutController.java

```
package edu.up.cas.sp.controller;

import java.sql.Timestamp;
import java.util.ArrayList;
import java.util.Calendar;
import java.util.List;
import java.util.TimeZone;
import javax.servlet.http.HttpServletRequest;
import
org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
```

```
import
org.springframework.web.bind.annotation.RequestMapping;
import
org.springframework.web.bind.annotation.RequestMethod;
import
org.springframework.web.bind.annotation.ResponseBody;
import org.json.JSONArray;
import org.json.JSONObject;
import com.google.gson.Gson;
import edu.up.cas.sp.dto.TransactionDto;
import edu.up.cas.sp.model.Inventory;
import edu.up.cas.sp.model.Item;
import edu.up.cas.sp.model.Payment;
import edu.up.cas.sp.model.PaymentType;
import edu.up.cas.sp.model.Receipt;
import edu.up.cas.sp.model.ReturnItemVoucher;
import edu.up.cas.sp.model.Store;
import edu.up.cas.sp.model.Transaction;
import edu.up.cas.sp.model.User;
import edu.up.cas.sp.service.InventoryService;
import edu.up.cas.sp.service.ItemService;
import edu.up.cas.sp.service.PaymentService;
import edu.up.cas.sp.service.PaymentTypeService;
import edu.up.cas.sp.service.ReceiptService;
import edu.up.cas.sp.service.ReturnItemVoucherService;
import edu.up.cas.sp.service.TransactionService;
import edu.up.cas.sp.util.PDFUtil;
import edu.up.cas.sp.util.TransactionUtil;
@Controller
public class CheckoutController {

        @Autowired
    TransactionService transactionService;

        @Autowired
    ReceiptService receiptService;

        @Autowired
    ItemService itemService;

        @Autowired
    InventoryService inventoryService;

        @Autowired
    ReturnItemVoucherService returnItemVoucherService;

        @Autowired
    PaymentTypeService paymentTypeService;

        @Autowired
    PaymentService paymentService;

        /*
     * This method will redirect to the Checkout page.
     */
    @RequestMapping(value = { "/checkout" }, method =
RequestMethod.GET)
    public String goToCheckoutPage() {
        return "checkout";
    }

    /*
     * This method will get transactions based on
receipt Id
     */
```

```
        @RequestMapping(value = {
"/getTransactionsByReceipt" }, method =
RequestMethod.GET)
    @ResponseBody
    public String
getTransactionsByReceipt(HttpServletRequest request) {


                    Integer receiptId =
Integer.parseInt(request.getParameter("receiptId"));
                    List<Transaction> items;
                    List<TransactionDto> transactions =
new ArrayList<TransactionDto>();
                    String json = null;
                    Gson gson = new Gson();
                    try {
                              items =
receiptService.getTransactionsByReceiptIdToday(receiptId
);
                              transactions =
TransactionUtil.getTransactions(items);
                              json =
gson.toJson(transactions);
                    }catch (Exception e) {
                              e.printStackTrace();
                    }
                    return json;
    }


         /*
     * This method will get payment types
     */
         @RequestMapping(value = { "/getPaymentTypes"
}, method = RequestMethod.GET)
    @ResponseBody
    public String getPaymentTypes() {
                    List<PaymentType> paymentTypes;
                    Gson gson = new Gson();
                    String json = null;
                    try {
                              paymentTypes =
paymentTypeService.findAllPAymentTypes();
                              json =
gson.toJson(paymentTypes);
                    }catch(Exception e) {
                              e.printStackTrace();
                    }
        return json;
    }


    /*
     * This method will add purchased items to DB.
     */
    @RequestMapping(value = { "/checkout-items" },
method = RequestMethod.GET)
    @ResponseBody
    public String checkoutItems(HttpServletRequest
request) {
          String userName =
request.getParameter("userName");
          String transactionItems =
request.getParameter("transactionItems");
          String userIdString =
request.getParameter("userId");
          String storeDetails =
request.getParameter("storeDetails");
          String totalItems =
request.getParameter("totalItems");
          String vatableSale =
request.getParameter("vatableSale");
          String vat = request.getParameter("vat");
          String paymentTypes =
request.getParameter("paymentTypes");
```

```
		//vouchers
		String returnedItemVouchers = request.getParameter("returnedItemVouchers");
		//payment methods
		String paymentMethods = request.getParameter("paymentMethods");

		String netAmountDueString = request.getParameter("netAmountDue");
		String amountPaidString = request.getParameter("amountPaid");
		String amountChangeString = request.getParameter("amountChange");

		Double netAmountDue = Double.parseDouble(netAmountDueString);
		Double amountPaid = Double.parseDouble(amountPaidString);
		Double amountChange = Double.parseDouble(amountChangeString);

		Integer userId = Integer.parseInt(userIdString);

		JSONArray voucherData = null;
		JSONArray paymentMethodsData = null;

		//process the string, convert into json format
		transactionItems = "{transaction: " + transactionItems + "}";
		paymentTypes = "{paymentType: " + paymentTypes + "}";

		if(returnedItemVouchers != null) {
			returnedItemVouchers = "{returnedItemVouchers: " + returnedItemVouchers + "}";
			final JSONObject jsonObjVouchers = new JSONObject(returnedItemVouchers);
		voucherData = jsonObjVouchers.getJSONArray("returnedItemVouchers");
		}

		if(paymentMethods != null) {
			paymentMethods = "{paymentMethods: " + paymentMethods + "}";
			final JSONObject jsonObjPaymentMethods = new JSONObject(paymentMethods);
			paymentMethodsData = jsonObjPaymentMethods.getJSONArray("paymentMethods");
		}

		//for transaction items
		final JSONObject jsonObj = new JSONObject(transactionItems);
		final JSONArray transactionData = jsonObj.getJSONArray("transaction");

		//for Payment types
		final JSONObject jsonObjpType = new JSONObject(paymentTypes);
		final JSONArray paymentTypeData = jsonObjpType.getJSONArray("paymentType");

		//Transfer payment Type data to a list
		List<PaymentType> pTypeList = new ArrayList<PaymentType>();
		if(paymentTypes != null) {
			for(int i = 0; i < paymentTypeData.length(); i++) {
				final JSONObject paymentJson = paymentTypeData.getJSONObject(i);
```

```
                        PaymentType paymentType =
new PaymentType();

        paymentType.setPaymentTypeId(paymentJson.getIn
t("paymentTypeId"));

        paymentType.setPaymentType(paymentJson.getStri
ng("paymentType"));

                        pTypeList.add(paymentType);
                }
        }

        Receipt receipt = new Receipt();
        try {
                Store store = new Store();

        store.setStoreId(Integer.parseInt(request.getP
arameter("storeId")));
                receipt.setStore(store);
                User user = new User();
                user.setUserID(userId);
                receipt.setUser(user);
                receipt.setAmountDue(netAmountDue);
                receipt.setAmountPaid(amountPaid);

        receipt.setAmountChange(amountChange);
                receipt.setTimestamp(new
Timestamp(Calendar.getInstance(TimeZone.getTimeZone("Asi
a/Manila")).getTimeInMillis()));

                for (int i = 0; i <
transactionData.length(); ++i) {

                        Transaction transaction =
new Transaction();
                        Inventory inventory = new
Inventory();

                        final JSONObject transactionJson =
transactionData.getJSONObject(i);

                        Item item = new Item();

        item.setItemdesc(transactionJson.getString("it
emName"));
                        inventory.setItem(item);

        inventory.setInventoryId(transactionJson.getIn
t("inventoryId"));

                        //values from JSON

        transaction.setDiscount(transactionJson.getInt
("itemDiscount"));
                        transaction.setInventory(inventory);

        transaction.setPrice(transactionJson.getDouble
("itemPrice"));

        transaction.setQuantity(transactionJson.getInt
("itemQuantity"));

                        receipt.addTransaction(transaction);

                        //update inventory
                        Inventory inventoryToUpdate =
inventoryService.findById(transactionJson.getInt("invent
oryId"));
                        Integer currentQuantity =
inventoryToUpdate.getItemCount();
                        System.out.println("Current count: "
+ currentQuantity);
```

```
                Integer newQuantity = currentQuantity
- transactionJson.getInt("itemQuantity");


                Inventory newInventory = new
Inventory();

        newInventory.setInventoryId(transactionJson.ge
tInt("inventoryId"));

        newInventory.setItemCount(newQuantity);



        inventoryService.updateInventory(newInventory)
;


        }


                //for payment methods
                if (paymentMethods != null) {
                        for (int i = 0; i <
paymentMethodsData.length(); ++i) {
                                Payment payment =
new Payment();
                                PaymentType
paymentType = new PaymentType();
                        final JSONObject
paymentMethodsJson =
paymentMethodsData.getJSONObject(i);



        paymentType.setPaymentTypeId(paymentMethodsJso
n.getInt("pmTypeOptionsId"));
                        payment.setReceipt(receipt);

        payment.setPaymentType(paymentType);

        payment.setPaymentReferenceId(paymentMethodsJs
on.getString("pmPaymentReferenceId"));

        payment.setAmount(paymentMethodsJson.getDouble
("pmAmountPaid"));


                        receipt.addPayment(payment);
                }
                }


                //for vouchers
                if (returnedItemVouchers != null) {
                        for (int i = 0; i <
voucherData.length(); ++i) {
                                Payment payment =
new Payment();
                                PaymentType
paymentType = new PaymentType();


                                ReturnItemVoucher
returnItemVoucher = new ReturnItemVoucher();
                        final JSONObject voucherJson
= voucherData.getJSONObject(i);

        returnItemVoucher.setReturnItemVoucherId(vouch
erJson.getInt("returnedItemVoucherNumber"));



        paymentType.setPaymentTypeId(4); //4 for
voucher
                        payment.setReceipt(receipt);

        payment.setPaymentType(paymentType);

        payment.setPaymentReferenceId(Integer.toString
(voucherJson.getInt("returnedItemVoucherNumber")));

        payment.setAmount(voucherJson.getDouble("retur
nedItemVoucherAmount"));
```

```
                        receipt.addPayment(payment);



        returnItemVoucherService.updateReturnItemVouch
er(returnItemVoucher);



                }

                }



                //save receipt

                receiptService.saveReceipt(receipt);



                //print pdf

                PDFUtil.createPDF(request,
storeDetails, receipt, userName, totalItems,
vatableSale, vat, pTypeList);



        }

        catch(Exception e) {

                e.printStackTrace();

                return "fail";

        }

        return "success";

    }

}
```

InventoryController.java

```
package edu.up.cas.sp.controller;

import java.util.ArrayList;

import java.util.List;

import javax.servlet.http.HttpServletRequest;

import
org.springframework.beans.factory.annotation.Autowired;

import org.springframework.stereotype.Controller;

import
org.springframework.web.bind.annotation.PathVariable;

import
org.springframework.web.bind.annotation.RequestMapping;

import
org.springframework.web.bind.annotation.RequestMethod;

import
org.springframework.web.bind.annotation.ResponseBody;

import com.google.gson.Gson;

import edu.up.cas.sp.dto.InventoryDto;

import edu.up.cas.sp.model.Area;

import edu.up.cas.sp.model.Inventory;

import edu.up.cas.sp.model.Item;

import edu.up.cas.sp.model.Store;

import edu.up.cas.sp.service.InventoryService;

import edu.up.cas.sp.service.ItemService;

import edu.up.cas.sp.util.InventoryUtil;

@Controller

public class InventoryController {



        @Autowired

    InventoryService service;



        @Autowired

    ItemService itemService;



        /*

     * This method will redirect the page to the
Inventory page.

     */

    @RequestMapping(value = { "/inventory" }, method =
RequestMethod.GET)

    public String goToInventoryPage() {

        return "inventory";
```

```
    }

    /*
     * This method will return all inventories to the
Inventory page.
     */
    @RequestMapping(value = { "/search-inventory" },
method = RequestMethod.GET)
    @ResponseBody
    public String searchInventory(HttpServletRequest
request) {
          String searchItem =
request.getParameter("searchItem");
          String searchBarcode =
request.getParameter("searchBarcode");
          String areaIdString =
request.getParameter("areaid2");
          String storeIdString =
request.getParameter("storeid2");

          Integer areaIdInt = null;
          if(areaIdString!=null) {
                    areaIdInt =
Integer.parseInt(areaIdString);
          }

          Integer storeIdInt = null;
          if(storeIdString!=null) {
                    storeIdInt =
Integer.parseInt(storeIdString);
          }

          Item item = new Item();
          item.setItemname(searchItem);
          if(searchBarcode!=null) {
                    item.setBarCode(searchBarcode);
          }

          Store store = new Store();
          if(storeIdInt!=null) {
                    store.setStoreId(storeIdInt);
          }

          Area area = new Area();

          if(areaIdInt!=null) {
                    area.setAreaId(areaIdInt);
                    store.setArea(area);
          }

          Inventory inventory = new Inventory();
          inventory.setItem(item);
          inventory.setStore(store);

          List<Inventory> inventoryList =
service.findInventory(inventory);

        Gson gson = new Gson();
        String json = gson.toJson(inventoryList);
        return json;
    }
    /*
     * This method will return all inventories to the
Inventory page.
     */
```

```
    @RequestMapping(value = { "/get-inventory" }, method
= RequestMethod.GET)
    @ResponseBody
    public String getInventory(HttpServletRequest
request) {
//        String usertype =
request.getParameter("usertype");
          String storeId =
request.getParameter("storeId");


//        Integer usertypeInt =
Integer.parseInt(usertype);
          Integer storeIdInt =
Integer.parseInt(storeId);


          List<Inventory> inventory = new
ArrayList<Inventory>();
          List<InventoryDto> inventoryList = new
ArrayList<InventoryDto>();


          try {
                    inventory =
service.findInventoryByStoreId(storeIdInt);
                    inventoryList =
InventoryUtil.getInventories(inventory);
          }catch(Exception e) {
                    e.printStackTrace();
          }


        Gson gson = new Gson();
        String json = gson.toJson(inventoryList);
        return json;
    }


    /*
     * This method will return all inventories to the
Inventory page.
     */
    @RequestMapping(value = { "/add-inventory" }, method
= RequestMethod.GET)
    @ResponseBody
    public String addInventory(HttpServletRequest
request) {
          String storeId =
request.getParameter("storeId");
          String itemId =
request.getParameter("itemId");


          Integer storeIdInt =
Integer.parseInt(storeId);
          Integer itemIdInt = Integer.parseInt(itemId);


          Item item = itemService.findById(itemIdInt);


          Store store = new Store();
          store.setStoreId(storeIdInt);


          Inventory inventory = new Inventory();
          inventory.setItem(item);
          inventory.setStore(store);
          inventory.setItemCount(0);


          //save inventory
          service.saveInventory(inventory);


          //return newly-added inventory
        Gson gson = new Gson();
        String json = gson.toJson(inventory);
        return json;
```

```
    }

    /*
     * This method will update inventory item in the DB.
     */
    @RequestMapping(value = { "/update-inventory" },
method = RequestMethod.GET)
    @ResponseBody
    public void UpdateItem(HttpServletRequest request) {
          String inventoryId =
request.getParameter("inventoryId");
          String itemCount =
request.getParameter("itemCount");

          Integer inventoryIdInteger =
Integer.parseInt(inventoryId);
          Integer itemCountInteger =
Integer.parseInt(itemCount);

          Inventory inventory = new Inventory();
          inventory.setInventoryId(inventoryIdInteger);
          inventory.setItemCount(itemCountInteger);

          service.updateInventory(inventory);
    }

    /*
     * This method will delete an inventory item by its
Inventory Id.
     */
    @RequestMapping(value = { "/delete-{inventoryId}-
inventory" })
    @ResponseBody
    public void deleteInventory(@PathVariable Integer
inventoryId) {
          service.deleteInventory(inventoryId);
    }
}
```

ItemController.java

```
package edu.up.cas.sp.controller;

import java.util.List;

import javax.servlet.http.HttpServletRequest;

import
org.springframework.beans.factory.annotation.Autowired;

import org.springframework.stereotype.Controller;

import
org.springframework.web.bind.annotation.PathVariable;

import
org.springframework.web.bind.annotation.RequestMapping;

import
org.springframework.web.bind.annotation.RequestMethod;

import
org.springframework.web.bind.annotation.ResponseBody;

import com.google.gson.Gson;

import edu.up.cas.sp.model.Item;

import edu.up.cas.sp.service.ItemService;

@Controller

public class ItemController {

          @Autowired
    ItemService service;

          /*
     * This method will redirect the page to the Items
page.
     */
    @RequestMapping(value = { "/items" }, method =
RequestMethod.GET)
    public String goToItemsPage() {
        return "items";
    }
```

```
    /*
     * This method will return all items to the Items
page.
     */
    @RequestMapping(value = { "/get-items" }, method =
RequestMethod.GET)
    @ResponseBody
    public String getItems() {
        List<Item> items = service.findAllItems();
        Gson gson = new Gson();
        String json = gson.toJson(items);
        return json;
    }

    /*
     * This method will add new item to DB.
     */
    @RequestMapping(value = { "/add-item" }, method =
RequestMethod.GET)
    @ResponseBody
    public String saveItem(HttpServletRequest request) {
          String itemName =
request.getParameter("itemName");
          String barCode =
request.getParameter("barCode");
          String itemDesc =
request.getParameter("itemDesc");
          String itemPrice =
request.getParameter("itemPrice");

          Double iPrice = (itemPrice.equals("")?new
Double(0):Double.parseDouble(itemPrice));

          Item item = new Item();
          item.setItemname(itemName);
          item.setBarCode(barCode);
          item.setItemdesc(itemDesc);
          item.setPrice(iPrice);

          //Save item
          service.saveItem(item);

          //return the item
        Gson gson = new Gson();
        String json = gson.toJson(item);

        return json;
    }

    /*
     * This method will update item in the DB.
     */
    @RequestMapping(value = { "/update-item" }, method =
RequestMethod.GET)
    @ResponseBody
    public void UpdateItem(HttpServletRequest request) {
          String itemId =
request.getParameter("itemId");
          String itemName =
request.getParameter("itemName");
          String barCode =
request.getParameter("barCode");
          String itemDesc =
request.getParameter("itemDesc");
          String itemPrice =
request.getParameter("itemPrice");

          Double iPrice = (itemPrice.equals("")?new
Double(0):Double.parseDouble(itemPrice));
```

```
        Item item = new Item();
        item.setItemId(Integer.parseInt(itemId));
        item.setItemname(itemName);
        item.setBarCode(barCode);
        item.setItemdesc(itemDesc);
        item.setPrice(iPrice);

        service.updateItem(item);
    }

    /*
     * This method will delete an item by its Item Id.
     */
    @RequestMapping(value = { "/delete-{itemId}-item" })
    @ResponseBody
    public void deleteItem(@PathVariable Integer itemId)
{
        service.deleteItem(itemId);
    }
}
```

LoginLogoutController.java

```
package edu.up.cas.sp.controller;

import javax.servlet.http.HttpServletRequest;

import
org.springframework.beans.factory.annotation.Autowired;

import org.springframework.stereotype.Controller;

import
org.springframework.web.bind.annotation.RequestMapping;

import
org.springframework.web.bind.annotation.RequestMethod;

import
org.springframework.web.bind.annotation.ResponseBody;

import com.google.gson.Gson;

import edu.up.cas.sp.model.User;

import edu.up.cas.sp.service.UserService;

@Controller

@RequestMapping("/")

public class LoginLogoutController {


        @Autowired

    UserService service;


        /*

     * This method will redirect browser to login page.

     */

    @RequestMapping(value = { "/login" })

    public String goToLogin() {

        return "login";

    }

    /*

     * This method will log out the user.

     */

    @RequestMapping(value = { "/logout" })

    public String logoutUser() {

        return "login";

    }



    /*

     * This method will be called on form submission,
handling POST request for

     * user login.

     */

    @RequestMapping(value = { "/login-user"}, method =
RequestMethod.GET)

    @ResponseBody
```

```
    public String login(HttpServletRequest request) {


          String userName =
request.getParameter("userName");
          String userPassword =
request.getParameter("userPassword");



          System.out.println("login user; username: " +
userName + "\npassword: " + userPassword);
          User user =
service.findByNameAndPassword(userName, userPassword);
          if (user!=null) {
                    //return the item
             Gson gson = new Gson();
             String json = gson.toJson(user);


             return json;
          } else {
                    return "";
          }


    }









}
```

ReportController.java

```
package edu.up.cas.sp.controller;
import java.io.FileReader;
import java.util.ArrayList;
import java.util.List;
import java.util.Map;
import java.util.TreeMap;
import javax.servlet.ServletContext;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpSession;
import org.json.simple.JSONObject;
import org.json.simple.parser.JSONParser;
import
org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import
org.springframework.web.bind.annotation.RequestMapping;
import
org.springframework.web.bind.annotation.RequestMethod;
import
org.springframework.web.bind.annotation.ResponseBody;
import com.google.gson.Gson;
import edu.up.cas.sp.dto.PaymentDto;
import edu.up.cas.sp.dto.ReceiptDto;
import edu.up.cas.sp.model.Payment;
import edu.up.cas.sp.model.Receipt;
import edu.up.cas.sp.service.PaymentService;
import edu.up.cas.sp.service.RMCService;
import edu.up.cas.sp.service.ReceiptService;
import edu.up.cas.sp.service.TransactionService;
import edu.up.cas.sp.util.PaymentUtil;
import edu.up.cas.sp.util.ReceiptUtil;
import edu.up.cas.sp.util.TopSellingUtil;
@Controller
public class ReportController {


          @Autowired
    RMCService rmcService;


          @Autowired
```

```
    TransactionService transactionService;


         @Autowired
    ReceiptService receiptService;


         @Autowired
    PaymentService paymentService;


         /*
     * This method will redirect the page to the Items
page.
     */
    @RequestMapping(value = { "/rmc" }, method =
RequestMethod.GET)
    public String goToRMCPage() {
        return "rmc";
    }


    /*
     * This method will redirect the page to the Items
page.
     */
    @RequestMapping(value = { "/reports" }, method =
RequestMethod.GET)
    public String goToReportssPage() {
        return "reports";
    }


    /*
     * This method will get coordinates of Luzon map and
return to client.
     */
         @RequestMapping(value = { "/getLuzonJson" },
method = RequestMethod.GET)
    @ResponseBody
    public String getLuzonJson(HttpServletRequest
request) {
        HttpSession session = request.getSession();
        ServletContext sc = session.getServletContext();
        String x = sc.getRealPath("/");
        String areaJson = x +
"resources/js/plugin/json/luzon.geojson";


         JSONParser parser = new JSONParser();
         String json = "";
        try {


            Object obj = parser.parse(new
FileReader(areaJson));


            JSONObject jsonObject = (JSONObject) obj;


            Gson gson = new Gson();
            json = gson.toJson(jsonObject);
            return json;
        } catch (Exception e) {
            e.printStackTrace();
            return null;
        }
    }


         /*
     * This method will get coordinates of Visayas map
and return to client.
     */
         @RequestMapping(value = { "/getVisayasJson" },
method = RequestMethod.GET)
    @ResponseBody
```

```
    public String getVisayasJson(HttpServletRequest
request) {
        HttpSession session = request.getSession();
        ServletContext sc = session.getServletContext();
        String x = sc.getRealPath("/");
        String philJson = x +
"resources/js/plugin/json/visayas.geojson";


         JSONParser parser = new JSONParser();
         String json = "";
        try {


            Object obj = parser.parse(new
FileReader(philJson));


            JSONObject jsonObject = (JSONObject) obj;


            Gson gson = new Gson();
            json = gson.toJson(jsonObject);
            return json;



        } catch (Exception e) {
            e.printStackTrace();
            return null;
        }


    }
         /*
     * This method will get payments based on parameters
set
     */
         @RequestMapping(value = { "/getPayments" },
method = RequestMethod.GET)
    @ResponseBody
    public String getPayments(HttpServletRequest
request) {


                String areaIdString =
request.getParameter("areaId");
                String storeIdString =
request.getParameter("storeId");
                String dateFrom =
request.getParameter("dateFrom");
                String dateTo =
request.getParameter("dateTo");


                Integer areaId = null;
                Integer storeId = null;


                if(areaIdString != null) {
                        areaId =
Integer.parseInt(request.getParameter("areaId"));
                }
                if(storeIdString != null) {
                        storeId =
Integer.parseInt(request.getParameter("storeId"));
                }


                List<Payment> payments = new
ArrayList<Payment>();
                List<PaymentDto> paymentList = new
ArrayList<PaymentDto>();
         try {
                if(storeId != null && storeId > 0) {
                        //get by store Id
                        if(dateFrom==null &&
dateTo==null) {
                                payments =
paymentService.getPaymentsByStoreToday(storeId);
```

```
                        } else {

                                payments =
paymentService.getPaymentsByStoreByDate(storeId,
dateFrom, dateTo);

                        }



                } else if (areaId != null && areaId >
0) {

                        //get by area Id

                        if(dateFrom==null &&
dateTo==null) {

                                payments =
paymentService.getPaymentsByAreaToday(areaId);

                        } else {

                                payments =
paymentService.getPaymentsByAreaByDate(areaId, dateFrom,
dateTo);

                        }

                } else {

                        //get all

                        if(dateFrom==null &&
dateTo==null) {

                                payments =
paymentService.getAllPaymentsToday();

                        } else {

                                payments =
paymentService.getAllPaymentsByDate(dateFrom, dateTo);

                        }

                }

                paymentList =
PaymentUtil.getPayments(payments);

        } catch(Exception e) {

                e.printStackTrace();

        }

       Gson gson = new Gson();

        String json = gson.toJson(paymentList);

        return json;

    }

         /*

     * This method will get receipts based on parameters
set

     */

         @RequestMapping(value = { "/getReceipts" },
method = RequestMethod.GET)

    @ResponseBody

    public String getReceipts(HttpServletRequest
request) {



                String areaIdString =
request.getParameter("areaId");

                String storeIdString =
request.getParameter("storeId");

                String dateFrom =
request.getParameter("dateFrom");

                String dateTo =
request.getParameter("dateTo");



                Integer areaId = null;

                Integer storeId = null;



                if(areaIdString != null) {

                        areaId =
Integer.parseInt(request.getParameter("areaId"));

                }

                if(storeIdString != null) {

                        storeId =
Integer.parseInt(request.getParameter("storeId"));

                }



                List<Receipt> receipts = new
ArrayList<Receipt>();
```

```
                List<ReceiptDto> receiptList = new
ArrayList<ReceiptDto>();


        try {
                if(storeId != null && storeId > 0) {
                        //get by store Id
                        if(dateFrom==null &&
dateTo==null) {
                                receipts =
receiptService.getReceiptsByStoreToday(storeId);
                        } else {
                                receipts =
receiptService.getReceiptsByStoreByDate(storeId,
dateFrom, dateTo);
                        }

                } else if (areaId != null && areaId >
0) {
                        //get by area Id
                        if(dateFrom==null &&
dateTo==null) {
                                receipts =
receiptService.getReceiptsByAreaToday(areaId);
                        } else {
                                receipts =
receiptService.getReceiptsByAreaByDate(areaId, dateFrom,
dateTo);
                        }
                } else {
                        //get all
                        if(dateFrom==null &&
dateTo==null) {
                                receipts =
receiptService.getAllReceiptsToday();
                        } else {
                                receipts =
receiptService.getAllReceiptsByDate(dateFrom, dateTo);
                        }
                }

                receiptList =
ReceiptUtil.getReceipts(receipts);

         } catch(Exception e) {
                e.printStackTrace();
         }
        Gson gson = new Gson();
        String json = gson.toJson(receiptList);
        return json;
    }
         /*
     * This method will get top-selling items based on
amount
     */
         @RequestMapping(value = {
"/getTopSellingByAmount" }, method = RequestMethod.GET)
    @ResponseBody
    public String
getTopSellingByAmount(HttpServletRequest request) {

                String areaIdString =
request.getParameter("areaId");
                String storeIdString =
request.getParameter("storeId");
                String dateFrom =
request.getParameter("dateFrom");
                String dateTo =
request.getParameter("dateTo");

                Integer areaId = null;
                Integer storeId = null;
```

```
                if(areaIdString != null) {
                        areaId =
Integer.parseInt(request.getParameter("areaId"));
                }
                if(storeIdString != null) {
                        storeId =
Integer.parseInt(request.getParameter("storeId"));
                }

                List<Receipt> receipts = new
ArrayList<Receipt>();
                //List<ReceiptDto> receiptList = new
ArrayList<ReceiptDto>();
                Map<String,Double> sortedMap = new
TreeMap<String, Double>();

        try {
                if(storeId != null && storeId > 0) {
                        //get by store Id
                        if(dateFrom==null &&
dateTo==null) {
                                receipts =
receiptService.getReceiptsByStoreToday(storeId);
                        } else {
                                receipts =
receiptService.getReceiptsByStoreByDate(storeId,
dateFrom, dateTo);
                        }

                } else if (areaId != null && areaId >
0) {
                        //get by area Id
                        if(dateFrom==null &&
dateTo==null) {
                                receipts =
receiptService.getReceiptsByAreaToday(areaId);
                        } else {
                                receipts =
receiptService.getReceiptsByAreaByDate(areaId, dateFrom,
dateTo);
                        }
                } else {
                        //get all
                        if(dateFrom==null &&
dateTo==null) {
                                receipts =
receiptService.getAllReceiptsToday();
                        } else {
                                receipts =
receiptService.getAllReceiptsByDate(dateFrom, dateTo);
                        }
                }
                //receiptList =
ReceiptUtil.getReceipts(receipts);
                sortedMap =
TopSellingUtil.getTopSellingByAmount(receipts);

         } catch(Exception e) {
                e.printStackTrace();
         }
        Gson gson = new Gson();
        String json = gson.toJson(sortedMap);
        return json;
    }
        /*
     * This method will get top-selling items based on
quantity
     */
        @RequestMapping(value = {
"/getTopSellingByQuantity" }, method =
RequestMethod.GET)
    @ResponseBody
```

```
    public String getTopSellingByQuantity(HttpServletRequest request) {


                String areaIdString = request.getParameter("areaId");

                String storeIdString = request.getParameter("storeId");

                String dateFrom = request.getParameter("dateFrom");

                String dateTo = request.getParameter("dateTo");



                Integer areaId = null;

                Integer storeId = null;



                if(areaIdString != null) {

                        areaId = Integer.parseInt(request.getParameter("areaId"));

                }

                if(storeIdString != null) {

                        storeId = Integer.parseInt(request.getParameter("storeId"));

                }



                List<Receipt> receipts = new ArrayList<Receipt>();

                //List<ReceiptDto> receiptList = new ArrayList<ReceiptDto>();

                Map<String, Integer> sortedMap = new TreeMap<String, Integer>();



        try {

                if(storeId != null && storeId > 0) {

                        //get by store Id

                        if(dateFrom==null && dateTo==null) {

                                receipts = receiptService.getReceiptsByStoreToday(storeId);

                        } else {

                                receipts = receiptService.getReceiptsByStoreByDate(storeId, dateFrom, dateTo);

                        }



                } else if (areaId != null && areaId > 0) {

                        //get by area Id

                        if(dateFrom==null && dateTo==null) {

                                receipts = receiptService.getReceiptsByAreaToday(areaId);

                        } else {

                                receipts = receiptService.getReceiptsByAreaByDate(areaId, dateFrom, dateTo);

                        }

                } else {

                        //get all

                        if(dateFrom==null && dateTo==null) {

                                receipts = receiptService.getAllReceiptsToday();

                        } else {

                                receipts = receiptService.getAllReceiptsByDate(dateFrom, dateTo);

                        }

                }

                sortedMap = TopSellingUtil.getTopSellingByQuantity(receipts);



        } catch(Exception e) {

                e.printStackTrace();

        }
```

```
        Gson gson = new Gson();
        String json = gson.toJson(sortedMap);
        return json;
    }

         /*
     * This method will get coordinates of Davao map and
return to client.
     */
         @RequestMapping(value = { "/getMindanaoJson"
}, method = RequestMethod.GET)
    @ResponseBody
    public String getMindanaoJson(HttpServletRequest
request) {
        HttpSession session = request.getSession();
        ServletContext sc = session.getServletContext();
        String x = sc.getRealPath("/");
        String philJson = x +
"resources/js/plugin/json/mindanao.geojson";

          JSONParser parser = new JSONParser();
          String json = "";
        try {

            Object obj = parser.parse(new
FileReader(philJson));

            JSONObject jsonObject = (JSONObject) obj;

            Gson gson = new Gson();
            json = gson.toJson(jsonObject);
            return json;
        } catch (Exception e) {
            e.printStackTrace();
            return null;
        }

    }

}
```

ReturnItemVoucherController.java

```
package edu.up.cas.sp.controller;

import java.sql.Timestamp;

import java.util.ArrayList;

import java.util.Date;

import java.util.List;

import javax.servlet.http.HttpServletRequest;

import org.json.JSONArray;

import org.json.JSONObject;

import
org.springframework.beans.factory.annotation.Autowired;

import org.springframework.stereotype.Controller;

import
org.springframework.web.bind.annotation.RequestMapping;

import
org.springframework.web.bind.annotation.RequestMethod;

import
org.springframework.web.bind.annotation.ResponseBody;

import com.google.gson.Gson;

import edu.up.cas.sp.model.Inventory;

import edu.up.cas.sp.model.Receipt;

import edu.up.cas.sp.model.ReturnItemVoucher;

import edu.up.cas.sp.model.ReturnedItem;

import edu.up.cas.sp.model.Store;
```

```
import edu.up.cas.sp.model.Transaction;

import edu.up.cas.sp.service.ReturnItemVoucherService;

import edu.up.cas.sp.service.ReturnedItemService;

@Controller

public class ReturnItemVoucherController {

         @Autowired

    ReturnedItemService returnedItemService;


         @Autowired

    ReturnItemVoucherService returnItemVoucherService;


         /*

     * This method will redirect to the Return items
page.

     */

    @RequestMapping(value = { "/return" }, method =
RequestMethod.GET)

    public String goToReturnItemsPage() {

        return "return";

    }

    /*

     * This method will add voucher as payment

     */

    @RequestMapping(value = { "/add-voucher" }, method =
RequestMethod.GET)

    @ResponseBody

    public String addVoucher(HttpServletRequest request)
{

         String voucherNumberString =
request.getParameter("voucherNumber");

         String storeIdString =
request.getParameter("storeId");


         Integer voucherNumber =
Integer.parseInt(voucherNumberString);

         Integer storeId =
Integer.parseInt(storeIdString);


         Store store = new Store();

         store.setStoreId(storeId);


         ReturnItemVoucher returnItemVoucher = new
ReturnItemVoucher();


         returnItemVoucher.setReturnItemVoucherId(vouch
erNumber);

         returnItemVoucher.setStore(store);


         List<ReturnItemVoucher> vouchers =
returnItemVoucherService.getReturnItemVoucher(returnItem
Voucher);


         Gson gson = new Gson();

        String json = gson.toJson(vouchers);

        return json;

    }

    /*

     * This method will add voucher as payment

     */

    @RequestMapping(value = { "/getVoucherByReceiptId"
}, method = RequestMethod.GET)

    @ResponseBody

    public String
getVoucherByReceiptId(HttpServletRequest request) {

         String receiptIdString =
request.getParameter("receiptId");


         Integer receiptId =
Integer.parseInt(receiptIdString);
```

```
        List<ReturnItemVoucher> vouchers =
returnItemVoucherService.getReturnItemVoucherByReceiptId
(receiptId);


        if(vouchers!=null && vouchers.size() > 0) {
                return "success";
        } else {
                return "failed";
        }
    }
    /*
     * This method will add purchased items to DB.
     */
    @RequestMapping(value = { "/return-items" }, method
= RequestMethod.GET)
    @ResponseBody
    public String saveReturnedItem(HttpServletRequest
request) {
        String returnedItems =
request.getParameter("returnedItems");
        String storeIdString =
request.getParameter("storeId");


        Integer storeId =
Integer.parseInt(storeIdString);


        //process the string, convert into json format
        returnedItems = "{returnedItems: " +
returnedItems + "}";


        System.out.println("returnedItems: " +
returnedItems);


        final JSONObject jsonObj = new
JSONObject(returnedItems);
        final JSONArray returnedItemData =
jsonObj.getJSONArray("returnedItems");


        List<ReturnedItem> itemsToReturn = new
ArrayList<ReturnedItem>();


        //Used to add in returnItem table
        Double amountReturned = 0.0;
        Integer receiptId = 0;
        ReturnItemVoucher returnItem = new
ReturnItemVoucher();


        try {
                for (int i = 0; i <
returnedItemData.length(); ++i) {
                        ReturnedItem returnedItem =
new ReturnedItem();


                        Transaction transaction =
new Transaction();
                Inventory inventory = new
Inventory();


                final JSONObject returnedItemJson =
returnedItemData.getJSONObject(i);
                receiptId =
returnedItemJson.getInt("receiptId");


                Receipt receipt = new Receipt();
                receipt.setReceiptId(receiptId);


                transaction.setReceipt(receipt);


        inventory.setInventoryId(returnedItemJson.getI
nt("inventoryId"));
```

```
//
		transaction.setReceiptId(receiptId);


				//dummy values

				returnedItem.setStoreId(storeId);


				//values from JSON

				returnedItem.setInventory(inventory);


		returnedItem.setTransaction(transaction);


		returnedItem.setQuantity(returnedItemJson.getI
nt("itemQuantityToReturn"));


		returnedItem.setPrice(returnedItemJson.getDoub
le("itemPrice"));


		returnedItem.setDiscount(returnedItemJson.getI
nt("itemDiscount"));

				returnedItem.setStatus("returned");


				amountReturned +=
((returnedItemJson.getInt("itemQuantityToReturn") *
returnedItemJson.getDouble("itemPrice")) * ((double)(1-
(returnedItemJson.getDouble("itemDiscount")/100))));


				System.out.println("amount returned:
" + amountReturned);


				itemsToReturn.add(returnedItem);


				}


				Date today = new Date();

				Timestamp timestamp = new
Timestamp(today.getTime());


				Store store = new Store();

				store.setStoreId(storeId);


				returnItem.setStore(store);

				returnItem.setReceiptId(receiptId);

				returnItem.setStatus("unclaimed");

				returnItem.setTimestamp(timestamp);


				returnItem.setAmount(amountReturned);


				//save amount back to customer


		returnItemVoucherService.saveReturnItemVoucher
(returnItem);


				//save items to return


		returnedItemService.saveReturnedItems(itemsToR
eturn);


		}

		catch(Exception e) {

				e.printStackTrace();

				return "fail";

		}

		return
Integer.toString(returnItem.getReturnItemVoucherId());

	}


}
```

StoreController.java

```
package edu.up.cas.sp.controller;

import java.util.List;
```

```
import javax.servlet.http.HttpServletRequest;

import
org.springframework.beans.factory.annotation.Autowired;

import org.springframework.stereotype.Controller;

import
org.springframework.web.bind.annotation.PathVariable;

import
org.springframework.web.bind.annotation.RequestMapping;

import
org.springframework.web.bind.annotation.RequestMethod;

import
org.springframework.web.bind.annotation.ResponseBody;

import com.google.gson.Gson;

import edu.up.cas.sp.model.Area;

import edu.up.cas.sp.model.Store;

import edu.up.cas.sp.service.StoreService;

@Controller

public class StoreController {

          @Autowired

    StoreService service;



          /*

     * This method will redirect the page to the Stores
page.

     */

    @RequestMapping(value = { "/stores" }, method =
RequestMethod.GET)

    public String goToItemsPage() {

        return "stores";

    }



    /*

     * This method will return all stores to the Stores
page.

     */

    @RequestMapping(value = { "/get-stores" }, method =
RequestMethod.GET)

    @ResponseBody

    public String getStores() {

        List<Store> stores = service.findAllStores();

        Gson gson = new Gson();

        String json = gson.toJson(stores);

        return json;



    }



    /*

     * This method will add new store to DB.

     */

    @RequestMapping(value = { "/add-store" }, method =
RequestMethod.GET)

    @ResponseBody

    public String saveStore(HttpServletRequest request)
{

          String areaId =
request.getParameter("areaId");

          String branchName =
request.getParameter("branchName");

          String tin = request.getParameter("tin");

          String branchaddress =
request.getParameter("branchaddress");

          String coordinates =
request.getParameter("coordinates");



          Area area = new Area();

          area.setAreaId(Integer.parseInt(areaId));



          Store store = new Store();

          store.setArea(area);

          store.setBranchName(branchName);
```

```
        store.setTin(tin);
        store.setAddress(branchaddress);
        store.setCoordinates(coordinates);

        //Save store
        service.saveStore(store);

        //return the newly saved store
        Gson gson = new Gson();
        return gson.toJson(store);
    }
    /*
     * This method will update store in the DB.
     */
    @RequestMapping(value = { "/update-store" }, method
= RequestMethod.GET)
    @ResponseBody
    public void UpdateItem(HttpServletRequest request) {
        String storeId =
request.getParameter("storeId");
        String areaId =
request.getParameter("areaId");
        String branchName =
request.getParameter("branchName");
        String tin = request.getParameter("tin");
        String branchaddress =
request.getParameter("branchaddress");
        String coordinates =
request.getParameter("coordinates");

        Area area = new Area();
        area.setAreaId(Integer.parseInt(areaId));

        Store store = new Store();
        store.setStoreId(Integer.parseInt(storeId));;
        store.setArea(area);
        store.setBranchName(branchName);
        store.setTin(tin);
        store.setAddress(branchaddress);
        store.setCoordinates(coordinates);

        service.updateStore(store);
    }
    /*
     * This method will delete a store by its store Id.
     */
    @RequestMapping(value = { "/delete-{storeId}-store"
})
    @ResponseBody
    public void deleteStore(@PathVariable Integer
storeId) {
        service.deleteStore(storeId);
    }
    /*
     * This method will retrieve a store by its store
Id.
     */
    @RequestMapping(value = { "/get-{storeId}-store" })
    @ResponseBody
    public String getStoreById(@PathVariable Integer
storeId) {
        Store store = service.findById(storeId);

        //return the store
        Gson gson = new Gson();
        return gson.toJson(store);
    }
```

```
}
```

UserController.java

```
package edu.up.cas.sp.controller;

import java.util.List;

import javax.servlet.http.HttpServletRequest;

import
org.springframework.beans.factory.annotation.Autowired;

import
org.springframework.security.crypto.bcrypt.BCrypt;

import org.springframework.stereotype.Controller;

import
org.springframework.web.bind.annotation.PathVariable;

import
org.springframework.web.bind.annotation.RequestMapping;

import
org.springframework.web.bind.annotation.RequestMethod;

import
org.springframework.web.bind.annotation.ResponseBody;

import com.google.gson.Gson;

import edu.up.cas.sp.model.Store;

import edu.up.cas.sp.model.User;

import edu.up.cas.sp.model.Usertype;

import edu.up.cas.sp.service.UserService;

import edu.up.cas.sp.service.UsertypeService;

@Controller

public class UserController {

        @Autowired

    UserService userService;



        @Autowired

    UsertypeService usertypeService;



        /*

     * This method will redirect the page to the Users
page.

     */

    @RequestMapping(value = { "/users" }, method =
RequestMethod.GET)

    public String goToItemsPage() {

        return "users";

    }

    /*

     * This method will return all users depending on
the usertype and storeId

     */

    @RequestMapping(value = { "/get-users" }, method =
RequestMethod.GET)

    @ResponseBody

    public String getUsersByStoreId(HttpServletRequest
request) {

        String storeIdString =
request.getParameter("storeId");

        String usertypeIdString =
request.getParameter("usertypeId");

        Integer storeId =
Integer.parseInt(storeIdString);

        Integer usertypeId =
Integer.parseInt(usertypeIdString);



        List<User> users = null;



        //if usertype is Proprietor

        if(usertypeId==1)

                users = userService.findAllUsers();

        //if usertype is Store Manager

        else if (usertypeId==2)

                users =
userService.findByStoreId(storeId);



        Gson gson = new Gson();

        String json = gson.toJson(users);
```

```
        return json;
    }

    /*
     * This method will return all user types to the
Users page.
     */
    @RequestMapping(value = { "/get-usertypes" }, method
= RequestMethod.GET)
    @ResponseBody
    public String getUsertypes() {
        List<Usertype> usertypes =
usertypeService.findAllUsertypes();
        Gson gson = new Gson();
        String json = gson.toJson(usertypes);
        return json;
    }
    /*
     * This method will add new user to DB.
     */
    @RequestMapping(value = { "/enable-disable-user" },
method = RequestMethod.GET)
    @ResponseBody
    public String enableDisableUser(HttpServletRequest
request) {
         String userId =
request.getParameter("userId");
         String isActive =
request.getParameter("isActive");

         Integer userIdInt =
(Integer.parseInt(userId));
         int isActiveInt =
(Integer.parseInt(isActive));

         User user = new User();
         user.setUserID(userIdInt);
         user.setActive(isActiveInt==0?1:0);

         //Save user
         userService.enableDisableUser(user);

         //return the new user
        Gson gson = new Gson();
        String json = gson.toJson(user);

        return json;
    }
    /*
     * This method will add new user to DB.
     */
    @RequestMapping(value = { "/add-user" }, method =
RequestMethod.GET)
    @ResponseBody
    public String saveItem(HttpServletRequest request) {
         String userName =
request.getParameter("userName");
         String userPassword =
request.getParameter("userPassword");
         String usertype =
request.getParameter("usertype");
         String storeId =
request.getParameter("storeId");
         String email = request.getParameter("email");
         String contactNo =
request.getParameter("contactNo");

         Integer usertypeInt =
(Integer.parseInt(usertype));
```

```
        Integer storeIdInt =
(Integer.parseInt(storeId));


        Usertype userType = new Usertype();
        userType.setUsertypeId(usertypeInt);


        Store store = new Store();
        store.setStoreId(storeIdInt);


        User user = new User();
        user.setUserName(userName);
        //encrypt password

        user.setUserPassword(BCrypt.hashpw(userPasswor
d, BCrypt.gensalt()));
        user.setEmail(email);
        user.setContactNo(contactNo);
        user.setStore(store);
        user.setUsertype(userType);
        user.setActive(1); //Default: Active user


        //Save user
        userService.saveUser(user);


        //return the new user
       Gson gson = new Gson();
       String json = gson.toJson(user);


       return json;
   }
   /*
    * This method will update item in the DB.
    */
   @RequestMapping(value = { "/update-user" }, method =
RequestMethod.GET)
   @ResponseBody
   public void updateUser(HttpServletRequest request) {
        String userId =
request.getParameter("userId");
        String userName =
request.getParameter("userName");
        String usertype =
request.getParameter("usertype");
        String storeId =
request.getParameter("storeId");
        String email = request.getParameter("email");
        String contactNo =
request.getParameter("contactNo");
        String isActive =
request.getParameter("isActive");



        Integer userIdInt = Integer.parseInt(userId);
        Integer usertypeInt =
Integer.parseInt(usertype);
        Integer storeIdInt =
Integer.parseInt(storeId);
        Integer isActiveInt =
Integer.parseInt(isActive);


        Usertype userType = new Usertype();
        userType.setUsertypeId(usertypeInt);


        Store store = new Store();
        store.setStoreId(storeIdInt);


        User user = new User();
        user.setUserID(userIdInt);
        user.setUserName(userName);
```

```
        user.setEmail(email);
        user.setContactNo(contactNo);
        user.setStore(store);
        user.setUsertype(userType);
        user.setActive(isActiveInt);

        //update user
        userService.updateUser(user);
    }

    /*
     * This method will update item in the DB.
     */
    @RequestMapping(value = { "/change-password" },
method = RequestMethod.GET)
    @ResponseBody
    public void changeUserPassword(HttpServletRequest
request) {
        String userId =
request.getParameter("userId");
        String userPassword =
request.getParameter("userPassword");

        Integer userIdInt = Integer.parseInt(userId);

        User user = new User();
        user.setUserID(userIdInt);

        user.setUserPassword(BCrypt.hashpw(userPasswor
d, BCrypt.gensalt()));

        //change user password
        userService.changePassword(user);
    }

    /*
     * This method will get a user by its userId.
     */
    @RequestMapping(value = { "/get-{userId}-user" })
    @ResponseBody
    public String getUser(@PathVariable Integer userId)
{
        User user = userService.findById(userId);

        //return the user
        Gson gson = new Gson();
        String json = gson.toJson(user);

        return json;
    }

    /*
     * This method will delete a user by its userId.
     */
    @RequestMapping(value = { "/delete-{userId}-user" })
    @ResponseBody
    public void deleteUser(@PathVariable Integer userId)
{
        userService.deleteUser(userId);
    }

}
```

AbstractDao.java

```
package edu.up.cas.sp.dao;
import java.io.Serializable;
import java.lang.reflect.ParameterizedType;
```

```
import org.hibernate.Criteria;

import org.hibernate.Session;

import org.hibernate.SessionFactory;

import
org.springframework.beans.factory.annotation.Autowired;


public abstract class AbstractDao<PK extends
Serializable, T> {


    private final Class<T> persistentClass;


    @SuppressWarnings("unchecked")

    public AbstractDao(){

        this.persistentClass =(Class<T>)
((ParameterizedType)
this.getClass().getGenericSuperclass()).getActualTypeArg
uments()[1];

    }


    @Autowired

    private SessionFactory sessionFactory;


    protected Session getSession(){

        return sessionFactory.getCurrentSession();

    }


    @SuppressWarnings("unchecked")

    public T getByKey(PK key) {

        return (T) getSession().get(persistentClass,
key);

    }


    public void persist(T entity) {

        getSession().persist(entity);

    }


    public void delete(T entity) {

        getSession().delete(entity);

    }


    protected Criteria createEntityCriteria(){

        return
getSession().createCriteria(persistentClass);

    }


}
```

InventoryDao.java

```
package edu.up.cas.sp.dao;

import java.util.List;

import edu.up.cas.sp.model.Inventory;

public interface InventoryDao {

         Inventory findById(Integer inventoryId);

         List<Inventory> findAllInventory();

         List<Inventory> findInventoryByStoreId(Integer
storeId);

         List<Inventory> findInventory(Inventory
inventory);

         void saveInventory(Inventory inventory);

         void deleteInventory(Integer inventoryId);

}
```

InventoryDaoImpl.java

```
package edu.up.cas.sp.dao;

import java.util.List;

import org.hibernate.Criteria;

import org.hibernate.Query;

import org.springframework.stereotype.Repository;

import edu.up.cas.sp.model.Inventory;
```

```
@Repository("inventoryDao")

public class InventoryDaoImpl extends
AbstractDao<Integer, Inventory> implements InventoryDao{

	@SuppressWarnings("unchecked")

	public List<Inventory> findAllInventory() {

		Criteria criteria =
createEntityCriteria();

	return (List<Inventory>) criteria.list();

	}

	@SuppressWarnings("unchecked")

	public List<Inventory>
findInventoryByStoreId(Integer storeId) {

		Query query =
getSession().createQuery("from Inventory as inv where
inv.store.storeId="+storeId);

		return (List<Inventory>)
query.list();

	}

	public void saveInventory(Inventory inventory)
{

		persist(inventory);

	}

	public void deleteInventory(Integer
inventoryId) {

		Query query =
getSession().createSQLQuery("delete from inventory where
inventoryId = :inventoryId");

	query.setInteger("inventoryId", inventoryId);

	query.executeUpdate();

	}

	public Inventory findById(Integer inventoryId)
{

		return getByKey(inventoryId);

	}

	@SuppressWarnings("unchecked")

	public List<Inventory> findInventory(Inventory
inventory) {

		System.out.println("inside dao");

		String barcode =
(inventory.getItem().getBarCode()==null ||
inventory.getItem().getBarCode().equals(""))?null:invent
ory.getItem().getBarCode();

		Integer areaId =
(inventory.getStore()==null)?null:(inventory.getStore().
getArea()==null?null:inventory.getStore().getArea().getA
reaId());




	//((Integer)inventory.getStore().getArea().get
AreaId()==null)?null:inventory.getStore().getArea().getA
reaId();

		Integer storeId =
(inventory.getStore()==null)?null:inventory.getStore().g
etStoreId();


	//((Integer)inventory.getStore().getStoreId()=
=null)?null:inventory.getStore().getStoreId();

		String item =
inventory.getItem().getItemname();

		System.out.println("barcode: " +
barcode + "\nareaId: " + areaId + "\nstoreId: " +
storeId + "\nitem: " + item);

		String stringQuery = "from Inventory
as inv where inv.item.itemname LIKE '%" + item + "%' ";

		if(areaId!=null) {

			stringQuery += "AND
inv.store.area.areaId='" + areaId + "' ";

		}

		if(storeId!=null) {

			stringQuery += "AND
inv.store.storeId='" + storeId + "' ";

		}

		if(barcode!=null) {

			stringQuery += "AND
inv.item.barCode='" + barcode + "' ";

		}
```

```
                //add quantity > 0
                stringQuery += "AND inv.itemCount > 0";

                System.out.println(stringQuery);

                Query query = getSession().createQuery(stringQuery);

                System.out.println("Length: " + (List<Inventory>) query.list()==null?null:((List<Inventory>) query.list()).size());
                return (List<Inventory>) query.list();
        }
}
```

ItemDao.java

```
package edu.up.cas.sp.dao;

import java.util.List;

import edu.up.cas.sp.model.Item;

public interface ItemDao {

        Item findById(Integer itemId);

        void saveItem(Item item);

        void deleteItem(Integer itemId);

        List<Item> findAllItems();

}
```

ItemDaoImpl.java

```
package edu.up.cas.sp.dao;

import java.util.List;

import org.hibernate.Criteria;

import org.hibernate.Query;

import org.springframework.stereotype.Repository;

import edu.up.cas.sp.model.Item;

@Repository("itemDao")

public class ItemDaoImpl extends AbstractDao<Integer, Item> implements ItemDao {

        public Item findById(Integer itemId) {
                return getByKey(itemId);
        }

        @SuppressWarnings("unchecked")

        public List<Item> findAllItems() {
                Criteria criteria = createEntityCriteria();
        return (List<Item>) criteria.list();
        }

        public void saveItem(Item item) {
                persist(item);
        }

        public void deleteItem(Integer itemId) {
                Query query = getSession().createSQLQuery("delete from item where itemId = :itemId");
        query.setLong("itemId", itemId);
        query.executeUpdate();
        }

}
```

PaymentDao.java

```
package edu.up.cas.sp.dao;

import java.util.List;

import edu.up.cas.sp.model.Payment;

public interface PaymentDao {

        void savePayments(List<Payment> paymentList);

        List<Payment> getPaymentsByStoreToday(Integer storeId);

        List<Payment> getPaymentsByAreaToday(Integer areaId);
```

```
        List<Payment> getAllPaymentsToday();

        List<Payment> getPaymentsByStoreByDate(Integer
storeId, String dateFrom, String dateTo);

        List<Payment> getPaymentsByAreaByDate(Integer
areaId, String dateFrom, String dateTo);

        List<Payment> getAllPaymentsByDate(String
dateFrom, String dateTo);

}
```

PaymentDaoImpl.java

```
package edu.up.cas.sp.dao;

import java.text.ParseException;

import java.text.SimpleDateFormat;

import java.util.Date;

import java.util.List;

import org.hibernate.Criteria;

import org.hibernate.Query;

import org.hibernate.criterion.Restrictions;

import org.springframework.stereotype.Repository;

import edu.up.cas.sp.model.Payment;

@Repository("paymentDao")

public class PaymentDaoImpl extends AbstractDao<Integer,
Payment> implements PaymentDao{

         public void savePayments(List<Payment>
paymentList) {



                  String savePaymentSql = "insert into
payment(receiptId, paymentTypeId, paymentReferenceId,
amount)"

                                    + "values";

        int paymentListSize = paymentList.size();

        //build the values

        //note this is not good pa

                  for(int i = 0; i < paymentListSize;
i++) {

                           //Transaction transaction =
transactions.get(i);

                           Payment payment =
paymentList.get(i);



                           savePaymentSql +=
"('"+payment.getReceipt().getReceiptId()+"','" +
payment.getPaymentType().getPaymentTypeId()+"','"+paymen
t.getPaymentReferenceId()+"','"

                           +payment.getAmount()+"'";



                           if(i<paymentListSize-1)

                                    savePaymentSql +=
"), ";

                           else

                                    savePaymentSql +=
")";

                  }

                  Query query =
getSession().createSQLQuery(savePaymentSql);



                  query.executeUpdate();

         }

         @SuppressWarnings("unchecked")

         public List<Payment>
getPaymentsByStoreToday(Integer storeId) {

                  //current date and areaId

                  Query query =
getSession().createQuery("FROM Payment payment WHERE
payment.receipt.store.storeId = '" + storeId + "' AND
payment.receipt.timestamp LIKE '"

                           + new
SimpleDateFormat("yyyy-MM-dd").format(new
Date().getTime()) +"%'");

                  return (List<Payment>) query.list();

         }

         @SuppressWarnings("unchecked")

         public List<Payment>
getPaymentsByAreaToday(Integer areaId) {

                  //current date and areaId
```

```
                Query query =
getSession().createQuery("FROM Payment payment WHERE
payment.receipt.store.area.areaId = '" + areaId + "' AND
payment.receipt.timestamp LIKE '"

                        + new
SimpleDateFormat("yyyy-MM-dd").format(new
Date().getTime()) +"%'");

                return (List<Payment>) query.list();

        }

        @SuppressWarnings("unchecked")

        public List<Payment> getAllPaymentsToday() {

                //current date

                Query query =
getSession().createQuery("FROM Payment payment WHERE
payment.receipt.timestamp LIKE '"

                        + new
SimpleDateFormat("yyyy-MM-dd").format(new
Date().getTime()) +"%'");

                return (List<Payment>) query.list();

        }

        @SuppressWarnings("unchecked")

        public List<Payment>
getPaymentsByStoreByDate(Integer storeId,

                        String dateFrom, String
dateTo) {

                SimpleDateFormat df = new
SimpleDateFormat("yyyy-MM-dd HH:mm:ss");

                //add time

                dateFrom += " 00:00:00";

                dateTo += " 23:59:59";



                Date fromDate;

                Date toDate;



                List<Payment> list = null;



                try {

                        fromDate =
df.parse(dateFrom);

                        toDate = df.parse(dateTo);

                        Criteria criteria =
getSession().createCriteria(Payment.class)


        .createCriteria("receipt")


        .add(Restrictions.between("timestamp",
fromDate, toDate))


        .createCriteria("store")


        .add(Restrictions.eq("storeId", storeId));

                        list = criteria.list();

                } catch (ParseException e) {

                        // TODO Auto-generated catch
block

                        e.printStackTrace();

                }

                return list;

        }

        @SuppressWarnings("unchecked")

        public List<Payment>
getPaymentsByAreaByDate(Integer areaId,

                        String dateFrom, String
dateTo) {

                SimpleDateFormat df = new
SimpleDateFormat("yyyy-MM-dd HH:mm:ss");

                //add time

                dateFrom += " 00:00:00";

                dateTo += " 23:59:59";



                Date fromDate;

                Date toDate;
```

```
                List<Payment> list = null;

                try {
                        fromDate =
df.parse(dateFrom);
                        toDate = df.parse(dateTo);
                        Criteria criteria =
getSession().createCriteria(Payment.class)

        .createCriteria("receipt")

        .add(Restrictions.between("timestamp",
fromDate, toDate))

        .createCriteria("store")

        .createCriteria("area")

        .add(Restrictions.eq("areaId", areaId));
                        list = criteria.list();
                } catch (ParseException e) {
                        // TODO Auto-generated catch
block
                        e.printStackTrace();
                }
                return list;
        }
        @SuppressWarnings("unchecked")
        public List<Payment>
getAllPaymentsByDate(String dateFrom, String dateTo) {
                SimpleDateFormat df = new
SimpleDateFormat("yyyy-MM-dd HH:mm:ss");
                //add time
                dateFrom += " 00:00:00";
                dateTo += " 23:59:59";

                Date fromDate;
                Date toDate;

                List<Payment> list = null;

                try {
                        fromDate =
df.parse(dateFrom);
                        toDate = df.parse(dateTo);
                        Criteria criteria =
getSession().createCriteria(Payment.class)

        .createCriteria("receipt")

        .add(Restrictions.between("timestamp",
fromDate, toDate));
                        list = criteria.list();
                } catch (ParseException e) {
                        // TODO Auto-generated catch
block
                        e.printStackTrace();
                }
                return list;
        }
}
```

PaymentTypeDao.java

```
package edu.up.cas.sp.dao;
import java.util.List;
import edu.up.cas.sp.model.PaymentType;
public interface PaymentTypeDao {
        List<PaymentType> findAllPAymentTypes();
}
```

PaymentTypeDaoImpl.java

```
package edu.up.cas.sp.dao;
```

```
import java.util.List;

import org.hibernate.Criteria;

import org.hibernate.criterion.Order;

import org.springframework.stereotype.Repository;

import edu.up.cas.sp.model.PaymentType;

@Repository("paymentTypeDao")

public class PaymentTypeDaoImpl extends
AbstractDao<Integer, PaymentType> implements
PaymentTypeDao {

	@SuppressWarnings("unchecked")

	public List<PaymentType> findAllPAymentTypes()
{

		Criteria criteria =
createEntityCriteria();


	criteria.addOrder(Order.asc("paymentTypeId"));

       return (List<PaymentType>) criteria.list();

	}

}
```

ReceiptDao.java

```
package edu.up.cas.sp.dao;

import java.util.List;

import edu.up.cas.sp.model.Receipt;

import edu.up.cas.sp.model.Transaction;

public interface ReceiptDao {

	void saveReceipt(Receipt receipt);

	Receipt findByKey(Integer receiptId);

	List<Receipt> getAllReceiptsToday();

	List<Receipt> getAllReceiptsByDate(String
dateFrom, String dateTo);

	List<Receipt> getReceiptsByAreaByDate(Integer
areaId, String dateFrom, String dateTo);

	List<Receipt> getReceiptsByStoreByDate(Integer
storeId, String dateFrom, String dateTo);

	List<Receipt> getReceiptsByAreaToday(Integer
areaId);

	List<Receipt> getReceiptsByStoreToday(Integer
storeId);

	List<Transaction>
getTransactionsByReceiptIdToday(Integer receiptId);

}
```

ReceiptDaoImpl.java

```
package edu.up.cas.sp.dao;

import java.text.ParseException;

import java.text.SimpleDateFormat;

import java.util.Date;

import java.util.List;

import org.hibernate.Criteria;

import org.hibernate.Query;

import org.hibernate.criterion.Restrictions;

import org.springframework.stereotype.Repository;

import edu.up.cas.sp.model.Receipt;

import edu.up.cas.sp.model.Transaction;

@Repository("receiptDao")

public class ReceiptDaoImpl extends AbstractDao<Integer,
Receipt> implements ReceiptDao{

	public void saveReceipt(Receipt receipt) {

		persist(receipt);

	}

	public Receipt findByKey(Integer receiptId) {

		return getByKey(receiptId);

	}

	@SuppressWarnings("unchecked")

	public List<Receipt> getAllReceiptsToday() {

		//current date

		Query query =
getSession().createQuery("FROM Receipt receipt WHERE
receipt.timestamp LIKE '"

					+ new
SimpleDateFormat("yyyy-MM-dd").format(new
Date().getTime()) +"%'");
```

```
                return (List<Receipt>) query.list();
        }
        @SuppressWarnings("unchecked")
        public List<Receipt>
getReceiptsByAreaToday(Integer areaId) {
                //current date and areaId
                Query query =
getSession().createQuery("FROM Receipt receipt WHERE
receipt.timestamp LIKE '"
                        + new
SimpleDateFormat("yyyy-MM-dd").format(new
Date().getTime()) +"%'"
                        + "AND
receipt.store.area.areaId='" + areaId + "'");
                return (List<Receipt>) query.list();
        }
        @SuppressWarnings("unchecked")
        public List<Receipt>
getReceiptsByStoreToday(Integer storeId) {
                //current date and storeId
                Query query =
getSession().createQuery("FROM Receipt receipt WHERE
receipt.timestamp LIKE '"
                        + new
SimpleDateFormat("yyyy-MM-dd").format(new
Date().getTime()) +"%'"
                        + "AND
receipt.store.storeId='" + storeId + "'");
                return (List<Receipt>) query.list();
        }
        public List<Transaction>
getTransactionsByReceiptIdToday(Integer receiptId) {
                return
getByKey(receiptId).getTransactions();
        }
        @SuppressWarnings("unchecked")
        public List<Receipt>
getAllReceiptsByDate(String dateFrom, String dateTo) {
                SimpleDateFormat df = new
SimpleDateFormat("yyyy-MM-dd HH:mm:ss");
                //add time
                dateFrom += " 00:00:00";
                dateTo += " 23:59:59";

                Date fromDate;
                Date toDate;

                List<Receipt> list = null;

                try {
                        fromDate =
df.parse(dateFrom);
                        toDate = df.parse(dateTo);
                        Criteria criteria =
getSession().createCriteria(Receipt.class)

.add(Restrictions.between("timestamp", fromDate,
toDate));
                        list = criteria.list();
                } catch (ParseException e) {
                        // TODO Auto-generated catch
block
                        e.printStackTrace();
                }

                return list;
        }
        @SuppressWarnings("unchecked")
        public List<Receipt>
getReceiptsByAreaByDate(Integer areaId,
                        String dateFrom, String
dateTo) {
                SimpleDateFormat df = new
SimpleDateFormat("yyyy-MM-dd HH:mm:ss");
```

```
		//add time
		dateFrom += " 00:00:00";
		dateTo += " 23:59:59";

		Date fromDate;
		Date toDate;

		List<Receipt> list = null;

		try {
			fromDate =
df.parse(dateFrom);
			toDate = df.parse(dateTo);
			Criteria criteria =
getSession().createCriteria(Receipt.class)

	.add(Restrictions.between("timestamp",
fromDate, toDate))

	.createCriteria("store")

	.createCriteria("area")

	.add(Restrictions.eq("areaId", areaId));
			list = criteria.list();
		} catch (ParseException e) {
			// TODO Auto-generated catch
block
			e.printStackTrace();
		}

		return list;
	}
	@SuppressWarnings("unchecked")
	public List<Receipt>
getReceiptsByStoreByDate(Integer storeId,
			String dateFrom, String
dateTo) {
		SimpleDateFormat df = new
SimpleDateFormat("yyyy-MM-dd HH:mm:ss");
		//add time
		dateFrom += " 00:00:00";
		dateTo += " 23:59:59";

		Date fromDate;
		Date toDate;

		List<Receipt> list = null;

		try {
			fromDate =
df.parse(dateFrom);
			toDate = df.parse(dateTo);
			Criteria criteria =
getSession().createCriteria(Receipt.class)

	.add(Restrictions.eq("store.storeId",
storeId))

	.add(Restrictions.between("timestamp",fromDate
,toDate));
			list = criteria.list();
		} catch (ParseException e) {
			// TODO Auto-generated catch
block
			e.printStackTrace();
		}

		return list;
	}
}
```

ReturnedItemDao.java

```
package edu.up.cas.sp.dao;

import java.util.List;

import edu.up.cas.sp.model.ReturnedItem;

public interface ReturnedItemDao {

	void saveReturnedItems(List<ReturnedItem>
returnedItems);

}
```

ReturnedItemDaoImpl.java

```
package edu.up.cas.sp.dao;

import java.sql.Timestamp;

import java.util.Date;

import java.util.List;

import org.hibernate.Query;

import org.springframework.stereotype.Repository;

import edu.up.cas.sp.model.ReturnedItem;

@Repository("returnedItemDao")

public class ReturnedItemDaoImpl extends
AbstractDao<Integer, ReturnedItem> implements
ReturnedItemDao{

	public void
saveReturnedItems(List<ReturnedItem> returnedItems) {

		Date today = new Date();

		Timestamp timestamp = new
Timestamp(today.getTime());



		String savetransactionSql = "insert
into returned_item(storeId, inventoryId, receiptId,
quantity, price, discount, status, timestamp)"

					+ "values";

	int returnedItemSize = returnedItems.size();

	//build the values

	//note this is not good pa

		for(int i = 0; i < returnedItemSize;
i++) {

			ReturnedItem rItem =
returnedItems.get(i);

				savetransactionSql +=
"('"+rItem.getStoreId()+"','" +
rItem.getInventory().getInventoryId()+"','"+rItem.getTra
nsaction().getReceipt().getReceiptId()+"','"


	+rItem.getQuantity()+"','"+rItem.getPrice()+"'
,'"


	+rItem.getDiscount()+"','"+rItem.getStatus()+"
','" + timestamp+"'";



				if(i<returnedItemSize-1)

					savetransactionSql
+= "), ";

				else

					savetransactionSql
+= ")";

		}

		Query query =
getSession().createSQLQuery(savetransactionSql);



		query.executeUpdate();

	}

}
```

ReturnItemVoucherDao.java

```
package edu.up.cas.sp.dao;

import java.util.List;

import edu.up.cas.sp.model.ReturnItemVoucher;

public interface ReturnItemVoucherDao {

	ReturnItemVoucher findByKey(Integer
returnedItemVoucherId);

	void saveReturnItemVoucher(ReturnItemVoucher
returnItemVoucher);

	List<ReturnItemVoucher>
getReturnItemVoucherById(Integer returnItemVoucherId);
```

```
	List<ReturnItemVoucher>
getReturnItemVoucher(ReturnItemVoucher
returnItemVoucher);

	List<ReturnItemVoucher>
getReturnItemVoucherByReceiptId(Integer receiptId);

}
```

ReturnItemVoucherDaoImpl.java

```
package edu.up.cas.sp.dao;

import java.util.List;

import org.hibernate.Query;

import org.springframework.stereotype.Repository;

import edu.up.cas.sp.model.ReturnItemVoucher;

@Repository("returnItemDao")

public class ReturnItemVoucherDaoImpl extends
AbstractDao<Integer, ReturnItemVoucher> implements
ReturnItemVoucherDao{

	public void
saveReturnItemVoucher(ReturnItemVoucher returnItem) {

		persist(returnItem);

	}

	@SuppressWarnings("unchecked")

	public List<ReturnItemVoucher>
getReturnItemVoucherById(Integer returnItemVoucherId) {

		Query query =
getSession().createQuery("FROM ReturnItemVoucher voucher
WHERE voucher.returnItemVoucherId = '"

					+
returnItemVoucherId +"'");

					return
(List<ReturnItemVoucher>) query.list();

	}

	public ReturnItemVoucher findByKey(Integer
returnedItemVoucherId) {

		return
getByKey(returnedItemVoucherId);

	}

	@SuppressWarnings("unchecked")

	public List<ReturnItemVoucher>
getReturnItemVoucher(

				ReturnItemVoucher
returnItemVoucher) {

		Query query =
getSession().createQuery("FROM ReturnItemVoucher voucher
WHERE voucher.returnItemVoucherId = '"

					+
returnItemVoucher.getReturnItemVoucherId() +"' AND
voucher.store.storeId = '"

					+
returnItemVoucher.getStore().getStoreId() + "'");

					return
(List<ReturnItemVoucher>) query.list();

	}

	@SuppressWarnings("unchecked")

	public List<ReturnItemVoucher>
getReturnItemVoucherByReceiptId(

				Integer receiptId) {

		Query query =
getSession().createQuery("FROM ReturnItemVoucher voucher
WHERE voucher.receiptId = '"

					+ receiptId +
"'");

					return
(List<ReturnItemVoucher>) query.list();

	}

}
```

RMCDao.java

```
package edu.up.cas.sp.dao;

import java.util.List;

import edu.up.cas.sp.model.Transaction;

public interface RMCDao {

	List<Transaction> getAllTransactions();

	List<Transaction>
getTransactionsByAreaId(Integer areaId);

}
```

RMCDaoImpl.java

```
package edu.up.cas.sp.dao;

import java.util.List;

import org.hibernate.Query;

import org.springframework.stereotype.Repository;

import edu.up.cas.sp.model.Transaction;

@Repository("rmcDao")

public class RMCDaoImpl extends AbstractDao<Integer,
Transaction> implements RMCDao{

	@SuppressWarnings("unchecked")

	public List<Transaction> getAllTransactions()
{

		Query query =
getSession().createQuery("from Transaction trans group
by trans.receiptId");

		return (List<Transaction>)
query.list();

	}

	public List<Transaction>
getTransactionsByAreaId(Integer areaId) {



		return null;

	}

}
```

StoreDao.java

```
package edu.up.cas.sp.dao;

import java.util.List;

import edu.up.cas.sp.model.Store;

public interface StoreDao {

	Store findById(Integer storeId);

	List<Store> findAllStores();

	void deleteStore(Integer storeId);

	void saveStore(Store store);

}
```

StoreDaoImpl.java

```
package edu.up.cas.sp.dao;

import java.util.List;

import org.hibernate.Criteria;

import org.hibernate.Query;

import org.springframework.stereotype.Repository;

import edu.up.cas.sp.model.Store;

@Repository("storeDao")

public class StoreDaoImpl extends AbstractDao<Integer,
Store> implements StoreDao{

	 @SuppressWarnings("unchecked")

	 public List<Store> findAllStores() {

		Criteria criteria =
createEntityCriteria();

	return (List<Store>) criteria.list();

	 }

	 public void saveStore(Store store) {

		persist(store);

	 }

	 public void deleteStore(Integer storeId) {

		Query query =
getSession().createSQLQuery("delete from store where
storeId = :storeId");

	query.setLong("storeId", storeId);

	query.executeUpdate();

	 }

	 public Store findById(Integer storeId) {

		return getByKey(storeId);

	 }

}
```

TransactionDao.java

```
package edu.up.cas.sp.dao;

import java.util.List;

import edu.up.cas.sp.model.Transaction;

public interface TransactionDao {
```

```
	void saveTransaction(Transaction transaction);

	void saveTransaction(List<Transaction>
transactions);

	List<Transaction> getAllTransactions();

	List<Transaction>
getTransactionsByArea(Integer areaId);

	List<Transaction>
getTransactionsByStore(Integer storeId);

	List<Transaction>
getTransactionsByReceipt(Integer receiptId);

}
```

TransactionDaoImpl.java

```
package edu.up.cas.sp.dao;

import java.text.SimpleDateFormat;

import java.util.Date;

import java.util.List;

import org.hibernate.Query;

import org.springframework.stereotype.Repository;

import edu.up.cas.sp.model.Transaction;

@Repository("transactionDao")

public class TransactionDaoImpl extends
AbstractDao<Integer, Transaction> implements
TransactionDao {

	public void saveTransaction(Transaction
transaction) {

		persist(transaction);

	}

	public void saveTransaction(List<Transaction>
transactions) {

		String savetransactionSql = "insert
into transaction(receiptId, inventoryId, quantity,
price, discount, timestamp)"

					+ "values";

	int transactionSize = transactions.size();

	//build the values

	//note this is not good pa

		for(int i = 0; i < transactionSize;
i++) {

			Transaction transaction =
transactions.get(i);

			savetransactionSql +=
"('"+transaction.getReceipt().getReceiptId()+"','"


	+transaction.getInventory().getInventoryId()+"
','"


	+transaction.getQuantity()+"','"+transaction.g
etPrice()+"','"+transaction.getDiscount()+"'";



			if(i<transactionSize-1)

				savetransactionSql
+= "), ";

			else

				savetransactionSql
+= ")";

		}

		Query query =
getSession().createSQLQuery(savetransactionSql);



		query.executeUpdate();

	}

	@SuppressWarnings("unchecked")

	public List<Transaction> getAllTransactions()
{

		//current date

		Query query =
getSession().createQuery("FROM Transaction trans WHERE
trans.timestamp LIKE '"

		+ new SimpleDateFormat("yyyy-MM-
dd").format(new Date().getTime()) +"%'");

		return (List<Transaction>)
query.list();

	}

	@SuppressWarnings("unchecked")
```

```
	public List<Transaction>
getTransactionsByArea(Integer areaId) {

		//current date and areaId

		Query query =
getSession().createQuery("FROM Transaction trans WHERE
trans.timestamp LIKE '"

			+ new
SimpleDateFormat("yyyy-MM-dd").format(new
Date().getTime()) +"%'"

			+ "AND areaId='" + areaId +
"'");

		return (List<Transaction>)
query.list();

	}

	@SuppressWarnings("unchecked")

	public List<Transaction>
getTransactionsByStore(Integer storeId) {

		//current date and areaId

		Query query =
getSession().createQuery("FROM Transaction trans WHERE
trans.timestamp LIKE '"

			+ new
SimpleDateFormat("yyyy-MM-dd").format(new
Date().getTime()) +"%'"

			+ "AND storeId='" + storeId
+ "'");

		return (List<Transaction>)
query.list();

	}

	@SuppressWarnings("unchecked")

	public List<Transaction>
getTransactionsByReceipt(Integer receiptId) {

		//current date and areaId

				Query query =
getSession().createQuery("FROM Transaction trans WHERE
trans.receipt.receiptId = '" + receiptId + "'");

				return
(List<Transaction>) query.list();

	}
}
```

UserDao.java

```
package edu.up.cas.sp.dao;

import java.util.List;

import edu.up.cas.sp.model.User;

public interface UserDao {

	User findById(Integer userId);

	User findByName(String username);

	User findByNameAndPassword(String username,
String password);

	List<User> findAllUsers();

	List<User> findByStoreId(Integer storeId);

	void SaveUser(User user);

	void deleteUser(Integer userId);

}
```

UserDaoImpl.java

```
package edu.up.cas.sp.dao;

import java.util.List;

import org.hibernate.Query;

import org.springframework.stereotype.Repository;

import edu.up.cas.sp.model.User;

@Repository("userDao")

public class UserDaoImpl extends AbstractDao<Integer,
User> implements UserDao{

	public User findById(Integer userId) {

		return getByKey(userId);

	}

	@SuppressWarnings("unchecked")

	public User findByName(String username) {

		Query query =
getSession().createQuery("Select new User(user.userID,
user.userName, user.userPassword, user.store,
user.email, user.contactNo, user.usertype, user.active)
from User user where user.userName='" +
```

```
                                    username + "'");

                List<User> users = query.list();

                return users.get(0);

        }


        @SuppressWarnings("unchecked")

        public User findByNameAndPassword(String
username, String password) {

                Query query =
getSession().createQuery("Select new User(user.userID,
user.userName, user.store, user.email, user.contactNo,
user.usertype, user.active) from User user where
user.userName='" +

                                    username + "' AND
user.userPassword='" + password + "'");

                List<User> users = query.list();

                return users.get(0);

        }

        @SuppressWarnings("unchecked")

        public List<User> findAllUsers() {

                Query query =
getSession().createQuery("Select new User(user.userID,
user.userName, user.store, user.email, user.contactNo,
user.usertype, user.active) from User user");

                List<User> users = query.list();

                return users;

        }

        public void SaveUser(User user) {

                persist(user);

        }

        public void deleteUser(Integer userId) {

                Query query =
getSession().createSQLQuery("delete from user where
userId = :userId");

        query.setLong("userId", userId);

        query.executeUpdate();

        }

        @SuppressWarnings("unchecked")

        public List<User> findByStoreId(Integer
storeId) {

                Query query =
getSession().createQuery("Select new User(user.userID,
user.userName, user.store, user.email, user.contactNo,
user.usertype, user.active) from User user where
user.usertype.usertypeId != 1 AND user.store.storeId=" +
storeId);

                List<User> users = query.list();

                return users;

        }


}
```

UsertypeDao.java

```
package edu.up.cas.sp.dao;

import java.util.List;

import edu.up.cas.sp.model.Usertype;

public interface UsertypeDao {

        List<Usertype> findAllUsertypes();

}
```

UsertypeDaoImpl.java

```
package edu.up.cas.sp.dao;

import java.util.List;

import org.hibernate.Criteria;

import org.springframework.stereotype.Repository;

import edu.up.cas.sp.model.Usertype;

@Repository("usertypeDao")

public class UsertypeDaoImpl extends
AbstractDao<Integer, Usertype> implements UsertypeDao{

        @SuppressWarnings("unchecked")

        public List<Usertype> findAllUsertypes() {

                Criteria criteria =
createEntityCriteria();
```

```
        return (List<Usertype>) criteria.list();

         }

}
```

InventoryDto.java

```
package edu.up.cas.sp.dto;
public class InventoryDto {

          private int inventoryId;
          private int itemId;
          private int areaId;
          private int storeId;
          private String itemName;
          private String barCode;
          private String itemDescription;
          private double itemPrice;
          private int itemQuantity;

          public int getInventoryId() {
                    return inventoryId;
          }
          public void setInventoryId(int inventoryId) {
                    this.inventoryId = inventoryId;
          }
          public int getItemId() {
                    return itemId;
          }
          public void setItemId(int itemId) {
                    this.itemId = itemId;
          }
          public int getAreaId() {
                    return areaId;
          }
          public void setAreaId(int areaId) {
                    this.areaId = areaId;
          }
          public int getStoreId() {
                    return storeId;
          }
          public void setStoreId(int storeId) {
                    this.storeId = storeId;
          }
          public String getItemName() {
                    return itemName;
          }
          public void setItemName(String itemName) {
                    this.itemName = itemName;
          }
          public String getBarCode() {
                    return barCode;
          }
          public void setBarCode(String barCode) {
                    this.barCode = barCode;
          }
          public String getItemDescription() {
                    return itemDescription;
          }
          public void setItemDescription(String
itemDescription) {
                    this.itemDescription =
itemDescription;
          }
          public double getItemPrice() {
                    return itemPrice;
          }
          public void setItemPrice(double itemPrice) {
                    this.itemPrice = itemPrice;
          }
          public int getItemQuantity() {
                    return itemQuantity;
          }
          public void setItemQuantity(int itemQuantity)
{
                    this.itemQuantity = itemQuantity;
          }
}
```

PaymentDto.java

```
package edu.up.cas.sp.dto;
public class PaymentDto {
          private String paymentType;
          public String getPaymentType() {
                    return paymentType;
          }
          public void setPaymentType(String paymentType)
{
                    this.paymentType = paymentType;
          }
          public Double getAmount() {
                    return amount;
          }
          public void setAmount(Double amount) {
                    this.amount = amount;
          }
          private Double amount;
}
```

ReceiptDto.java

```
package edu.up.cas.sp.dto;
public class ReceiptDto {
          private Double amountDue;
          private long timestamp;

          public Double getAmountDue() {
                    return amountDue;
          }
          public void setAmountDue(Double amountDue) {
                    this.amountDue = amountDue;
          }
          public long getTimestamp() {
                    return timestamp;
          }
          public void setTimestamp(long timestamp) {
                    this.timestamp = timestamp;
          }

}
```

TopSellingDto.java

```
package edu.up.cas.sp.dto;
public class TopSellingDto {
          String itemName;
          int itemCount;
          double amount;

          public String getItemName() {
                    return itemName;
          }
          public void setItemName(String itemName) {
                    this.itemName = itemName;
          }
          public int getItemCount() {
                    return itemCount;
          }
          public void setItemCount(int itemCount) {
                    this.itemCount = itemCount;
          }
          public double getAmount() {
                    return amount;
          }
          public void setAmount(double amount) {
                    this.amount = amount;
          }

}
```

TransactionDto.java

```
package edu.up.cas.sp.dto;
public class TransactionDto {
          private int transactionId;
          private String description;
```

```
	private int receiptId;
	private int inventoryId;
	private int quantity;
	private double price;
	private int discount;
	public int getTransactionId() {
		return transactionId;
	}
	public void setTransactionId(int
transactionId) {
		this.transactionId = transactionId;
	}

	public String getDescription() {
		return description;
	}
	public void setDescription(String description)
{
		this.description = description;
	}
	public int getReceiptId() {
		return receiptId;
	}
	public void setReceiptId(int receiptId) {
		this.receiptId = receiptId;
	}
	public int getInventoryId() {
		return inventoryId;
	}
	public void setInventoryId(int inventoryId) {
		this.inventoryId = inventoryId;
	}
	public int getQuantity() {
		return quantity;
	}
	public void setQuantity(int quantity) {
		this.quantity = quantity;
	}
	public double getPrice() {
		return price;
	}
	public void setPrice(double price) {
		this.price = price;
	}
	public int getDiscount() {
		return discount;
	}
	public void setDiscount(int discount) {
		this.discount = discount;
	}

}
```

Area.java

```
package edu.up.cas.sp.model;

import javax.persistence.Column;

import javax.persistence.Entity;

import javax.persistence.GeneratedValue;

import javax.persistence.GenerationType;

import javax.persistence.Id;

import javax.persistence.Table;

import javax.validation.constraints.NotNull;

@Entity

@Table(name="AREA")

public class Area {

	@Id

	@GeneratedValue(strategy =
GenerationType.IDENTITY)

	private Integer areaId;



	@NotNull

	@Column(name = "areaName", nullable = false)

	private String areaName;



	public Integer getAreaId() {

		return areaId;

	}

	public void setAreaId(Integer areaId) {

		this.areaId = areaId;

	}

	public String getAreaName() {

		return areaName;

	}

	public void setAreaName(String areaName) {

		this.areaName = areaName;

	}

}
```

Inventory.java

```
package edu.up.cas.sp.model;

import javax.persistence.Column;

import javax.persistence.Entity;

import javax.persistence.GeneratedValue;

import javax.persistence.GenerationType;

import javax.persistence.Id;

import javax.persistence.JoinColumn;

import javax.persistence.OneToOne;
```

```
import javax.persistence.Table;

import javax.validation.constraints.NotNull;

@Entity

@Table(name="INVENTORY")

public class Inventory {

	@Id

	@GeneratedValue(strategy =
GenerationType.IDENTITY)

	private int inventoryId;



	@OneToOne

    @JoinColumn(name="itemId")

	private Item item;



	@OneToOne

    @JoinColumn(name="storeId")

	private Store store;



	@NotNull

	@Column(name = "itemCount", nullable = false)

	private Integer itemCount;

	public int getInventoryId() {

		return inventoryId;

	}

	public void setInventoryId(int inventoryId) {

		this.inventoryId = inventoryId;

	}

	public Item getItem() {

		return item;

	}

	public void setItem(Item item) {

		this.item = item;

	}

	public Store getStore() {

		return store;

	}

	public void setStore(Store store) {

		this.store = store;

	}

	public Integer getItemCount() {

		return itemCount;

	}

	public void setItemCount(Integer itemCount) {

		this.itemCount = itemCount;

	}

}
```

Item.java

```
package edu.up.cas.sp.model;

import javax.persistence.Column;

import javax.persistence.Entity;

import javax.persistence.GeneratedValue;

import javax.persistence.GenerationType;

import javax.persistence.Id;

import javax.persistence.Table;

import javax.validation.constraints.NotNull;

@Entity

@Table(name="ITEM")

public class Item {



	@Id

	@GeneratedValue(strategy =
GenerationType.IDENTITY)

	private int itemId;
```

```
	@NotNull
	@Column(name = "barCode", nullable = false)
	private String barCode;

	public String getBarCode() {
		return barCode;
	}
	public void setBarCode(String barCode) {
		this.barCode = barCode;
	}
	@NotNull
	@Column(name = "itemName", nullable = false)
	private String itemname;

	@Column(name = "itemDesc")
	private String itemdesc;

	@Column(name = "price")
private Double price;
	public Double getPrice() {
		return price;
	}
	public void setPrice(Double price) {
		this.price = price;
	}
	public int getItemId() {
		return itemId;
	}
	public void setItemId(int itemId) {
		this.itemId = itemId;
	}
	public String getItemname() {
		return itemname;
	}
	public void setItemname(String itemname) {
		this.itemname = itemname;
	}

	public String getItemdesc() {
		return itemdesc;
	}
	public void setItemdesc(String itemdesc) {
		this.itemdesc = itemdesc;
	}
}
```

Payment.java

```
package edu.up.cas.sp.model;
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.JoinColumn;
import javax.persistence.ManyToOne;
import javax.persistence.OneToOne;
import javax.persistence.Table;
import javax.validation.constraints.NotNull;
@Entity
@Table(name="PAYMENT")
public class Payment {
	private Integer paymentId;
	private Receipt receipt;
	private PaymentType paymentType;
	private String paymentReferenceId;
```

```
	private Double amount;


	@Id
	@GeneratedValue(strategy =
GenerationType.IDENTITY)
	public Integer getPaymentId() {
		return paymentId;
	}
	public void setPaymentId(Integer paymentId) {
		this.paymentId = paymentId;
	}
	@ManyToOne
	@JoinColumn(name = "receiptId", nullable =
false)
	public Receipt getReceipt() {
		return receipt;
	}
	public void setReceipt(Receipt receipt) {
		this.receipt = receipt;
	}
	@Column(name = "paymentReferenceId")
	public String getPaymentReferenceId() {
		return paymentReferenceId;
	}
	public void setPaymentReferenceId(String
paymentReferenceId) {
		this.paymentReferenceId =
paymentReferenceId;
	}
	@NotNull
	@Column(name = "amount")
	public Double getAmount() {
		return amount;
	}
	public void setAmount(Double amount) {
		this.amount = amount;
	}
	@OneToOne
    @JoinColumn(name="paymentTypeId")
	public PaymentType getPaymentType() {
		return paymentType;
	}
	public void setPaymentType(PaymentType
paymentType) {
		this.paymentType = paymentType;
	}



}
PaymentType.java
package edu.up.cas.sp.model;
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.Table;
@Entity
@Table(name="PAYMENT_TYPE")
public class PaymentType {
	@Id
	@GeneratedValue(strategy =
GenerationType.IDENTITY)
	private Integer paymentTypeId;


	@Column(name = "paymentType")
	private String paymentType;
```

```
	public Integer getPaymentTypeId() {
		return paymentTypeId;
	}
	public void setPaymentTypeId(Integer paymentTypeId) {
		this.paymentTypeId = paymentTypeId;
	}
	public String getPaymentType() {
		return paymentType;
	}
	public void setPaymentType(String paymentType) {
		this.paymentType = paymentType;
	}
}
```

Receipt.java

```
package edu.up.cas.sp.model;

import java.sql.Timestamp;

import java.util.ArrayList;

import java.util.Calendar;

import java.util.List;

import javax.persistence.CascadeType;

import javax.persistence.Column;

import javax.persistence.Entity;

import javax.persistence.FetchType;

import javax.persistence.GeneratedValue;

import javax.persistence.GenerationType;

import javax.persistence.Id;

import javax.persistence.JoinColumn;

import javax.persistence.OneToMany;

import javax.persistence.OneToOne;

import javax.persistence.Table;

import javax.validation.constraints.NotNull;

import org.hibernate.annotations.LazyCollection;

import org.hibernate.annotations.LazyCollectionOption;

@Entity

@Table(name="RECEIPT")

public class Receipt {


	private List<Transaction> transactions = new ArrayList<Transaction>();

	private List<Payment> payments = new ArrayList<Payment>();

	private Integer receiptId;

	private Store store;

	private User user;

	private Double amountDue;

	private Double amountPaid;

	private Double amountChange;

	private Timestamp timestamp = new Timestamp(Calendar.getInstance().getTimeInMillis());

	@OneToMany(fetch = FetchType.EAGER, cascade = CascadeType.ALL, mappedBy = "receipt", orphanRemoval = true)

	public List<Transaction> getTransactions() {

		return transactions;

	}

	public void setTransactions(List<Transaction> transactions) {

		this.transactions = transactions;

	}

	@Id

	@GeneratedValue(strategy = GenerationType.IDENTITY)

	public Integer getReceiptId() {

		return receiptId;

	}

	public void setReceiptId(Integer receiptId) {
```

```
		this.receiptId = receiptId;
	}
	@OneToOne
@JoinColumn(name="storeId")
	public Store getStore() {
		return store;
	}
	public void setStore(Store store) {
		this.store = store;
	}
	@OneToOne
@JoinColumn(name="userId")
	public User getUser() {
		return user;
	}
	public void setUser(User user) {
		this.user = user;
	}
	@NotNull
	@Column(name = "amountDue")
	public Double getAmountDue() {
		return amountDue;
	}
	public void setAmountDue(Double amountDue) {
		this.amountDue = amountDue;
	}
	//@NotNull
	@Column(name = "amountPaid")
	public Double getAmountPaid() {
		return amountPaid;
	}
	public void setAmountPaid(Double amountPaid) {
		this.amountPaid = amountPaid;
	}
	//@NotNull
	@Column(name = "amountChange")
	public Double getAmountChange() {
		return amountChange;
	}
	public void setAmountChange(Double
amountChange) {
		this.amountChange = amountChange;
	}



	public void addTransaction(Transaction
transaction) {
		transactions.add(transaction);
		transaction.setReceipt(this);
    }


    public void removeTransaction(Transaction
transaction) {
		transaction.setReceipt(null);
		this.transactions.remove(transaction);
    }
    @LazyCollection(LazyCollectionOption.FALSE)
    @OneToMany(cascade = CascadeType.ALL, mappedBy =
"receipt", orphanRemoval = true)
	public List<Payment> getPayments() {
		return payments;
	}
	public void setPayments(List<Payment>
payments) {
		this.payments = payments;
	}
```

```
	public void addPayment(Payment payment) {
		payments.add(payment);
		payment.setReceipt(this);
    }

    public void removePayment(Payment payment) {
	payment.setReceipt(null);
	this.payments.remove(payment);
    }

    @NotNull
	@Column(name = "timestamp", nullable = false)
	public Timestamp getTimestamp() {
		return timestamp;
	}
	public void setTimestamp(Timestamp timestamp)
{
		this.timestamp = timestamp;
	}

}
```

ReturnedItem.java

```
package edu.up.cas.sp.model;
import java.util.Date;
import edu.up.cas.sp.model.Inventory;
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.JoinColumn;
import javax.persistence.OneToOne;
import javax.persistence.Table;
import javax.validation.constraints.NotNull;
//This is not yet done
@Entity
@Table(name="RETURNED_ITEM")
public class ReturnedItem {
	@Id
	@GeneratedValue(strategy =
GenerationType.IDENTITY)
	private int returnedItemId;

	@NotNull
	@Column(name = "storeId", nullable = false)
	private Integer storeId;

	public Integer getStoreId() {
		return storeId;
	}
	public void setStoreId(Integer storeId) {
		this.storeId = storeId;
	}

	@OneToOne
    @JoinColumn(name = "inventoryId")
	private Inventory inventory;

	@OneToOne
    @JoinColumn(name = "receiptId")
	private Transaction transaction;

	@NotNull
```

```
	@Column(name = "quantity", nullable = false)
	private Integer quantity;

	@NotNull
	@Column(name = "price", nullable = false)
	private Double price;

	@NotNull
	@Column(name = "discount", nullable = false)
	private Integer discount;

	@NotNull
	@Column(name = "status", nullable = false)
	private String status;

	@NotNull
	@Column(name = "timestamp", nullable = false)
	private Date timestamp;
	public int getReturnedItemId() {
		return returnedItemId;
	}
	public void setReturnedItemId(int returnedItemId) {
		this.returnedItemId = returnedItemId;
	}
	public Inventory getInventory() {
		return inventory;
	}
	public void setInventory(Inventory inventory) {
		this.inventory = inventory;
	}
	public Transaction getTransaction() {
		return transaction;
	}
	public void setTransaction(Transaction transaction) {
		this.transaction = transaction;
	}
	public Integer getQuantity() {
		return quantity;
	}
	public void setQuantity(Integer quantity) {
		this.quantity = quantity;
	}
	public Double getPrice() {
		return price;
	}
	public void setPrice(Double price) {
		this.price = price;
	}
	public Integer getDiscount() {
		return discount;
	}
	public void setDiscount(Integer discount) {
		this.discount = discount;
	}
	public String getStatus() {
		return status;
	}
	public void setStatus(String status) {
		this.status = status;
	}
	public Date getTimestamp() {
```

```
		return timestamp;
	}
	public void setTimestamp(Date timestamp) {
		this.timestamp = timestamp;
	}
}
```

ReturnItemVoucher.java

```
package edu.up.cas.sp.model;
import java.util.Date;
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.JoinColumn;
import javax.persistence.OneToOne;
import javax.persistence.Table;
import javax.validation.constraints.NotNull;
@Entity
@Table(name="RETURN_ITEM_VOUCHER")
public class ReturnItemVoucher {

	@Id
	@GeneratedValue(strategy =
GenerationType.IDENTITY)
	private Integer returnItemVoucherId;

	@OneToOne
    @JoinColumn(name="storeId")
	private Store store;

	@NotNull
	@Column(name = "receiptId", nullable = false)
	private Integer receiptId;

	@NotNull
	@Column(name = "amount", nullable = false)
	private Double amount;

	@NotNull
	@Column(name = "status", nullable = false)
	private String status;

	@NotNull
	@Column(name = "timestamp", nullable = false)
	private Date timestamp;
	public int getReturnItemVoucherId() {
		return returnItemVoucherId;
	}
	public void setReturnItemVoucherId(int
returnItemId) {
		this.returnItemVoucherId =
returnItemId;
	}

	public Store getStore() {
		return store;
	}
	public void setStore(Store store) {
		this.store = store;
	}
	public Integer getReceiptId() {
		return receiptId;
	}
	public void setReceiptId(Integer receiptId) {
```

```
                this.receiptId = receiptId;
        }
        public Double getAmount() {
                return amount;
        }
        public void setAmount(Double amount) {
                this.amount = amount;
        }
        public String getStatus() {
                return status;
        }
        public void setStatus(String status) {
                this.status = status;
        }
        public Date getTimestamp() {
                return timestamp;
        }
        public void setTimestamp(Date timestamp) {
                this.timestamp = timestamp;
        }



}
```

Store.java

```
package edu.up.cas.sp.model;
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.JoinColumn;
import javax.persistence.OneToOne;
import javax.persistence.Table;
import javax.validation.constraints.NotNull;
@Entity
@Table(name="STORE")
public class Store {
        @Id
        @GeneratedValue(strategy =
GenerationType.IDENTITY)
        private Integer storeId;


        @NotNull
        @Column(name = "branchName", nullable = false)
        private String branchName;


        @NotNull
        @Column(name = "tin", nullable = false)
        private String tin;


        @NotNull
        @Column(name = "address", nullable = false)
        private String address;


        public String getAddress() {
                return address;
        }
        public void setAddress(String address) {
                this.address = address;
        }
        @NotNull
        @Column(name = "coordinates", nullable =
false)
        private String coordinates;
```

```
	@OneToOne
    @JoinColumn(name="areaId")
	private Area area;
	public Integer getStoreId() {
		return storeId;
	}
	public void setStoreId(Integer storeId) {
		this.storeId = storeId;
	}
	public String getBranchName() {
		return branchName;
	}
	public void setBranchName(String branchName) {
		this.branchName = branchName;
	}
	public String getTin() {
		return tin;
	}
	public void setTin(String tin) {
		this.tin = tin;
	}
	public String getCoordinates() {
		return coordinates;
	}
	public void setCoordinates(String coordinates)
{
		this.coordinates = coordinates;
	}
	public Area getArea() {
		return area;
	}
	public void setArea(Area area) {
		this.area = area;
	}
}
```

Transaction.java

```
package edu.up.cas.sp.model;
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.JoinColumn;
import javax.persistence.ManyToOne;
import javax.persistence.OneToOne;
import javax.persistence.Table;
import javax.validation.constraints.NotNull;
@Entity
@Table(name="TRANSACTION")
public class Transaction {
	private Receipt receipt;

	@ManyToOne
	@JoinColumn(name = "receiptId", nullable =
false)
	public Receipt getReceipt() {
		return receipt;
	}

	public void setReceipt(Receipt receipt) {
		this.receipt = receipt;
	}
```

```
	private Integer transactionId;

	private Inventory inventory;

	private Integer quantity;

	private Double price;

	private Integer discount;



	@Id

	@GeneratedValue(strategy =
GenerationType.IDENTITY)

	public Integer getTransactionId() {

		return transactionId;

	}

	public void setTransactionId(Integer
transactionId) {

		this.transactionId = transactionId;

	}

	/**

	 * @return

	 */



	@OneToOne

    @JoinColumn(name="inventoryId")

	public Inventory getInventory() {

		return inventory;

	}

	public void setInventory(Inventory inventory)
{

		this.inventory = inventory;

	}

	@NotNull

	@Column(name = "price", nullable = false)

	public Double getPrice() {

		return price;

	}

	public void setPrice(Double price) {

		this.price = price;

	}

	@NotNull

	@Column(name = "discount", nullable = false)

	public Integer getDiscount() {

		return discount;

	}

	public void setDiscount(Integer discount) {

		this.discount = discount;

	}

	@NotNull

	@Column(name = "quantity", nullable = false)

	public Integer getQuantity() {

		return quantity;

	}

	public void setQuantity(Integer quantity) {

		this.quantity = quantity;

	}



}
```

User.java

```
package edu.up.cas.sp.model;

import javax.persistence.Column;

import javax.persistence.Entity;

import javax.persistence.GeneratedValue;

import javax.persistence.GenerationType;

import javax.persistence.Id;

import javax.persistence.JoinColumn;

import javax.persistence.OneToOne;
```

```
import javax.persistence.Table;

import javax.validation.constraints.NotNull;

@Entity

@Table(name="USER")

public class User {


	public User(int userId, String userName,
Store store,

		String email, String contactNo,
Usertype usertype, int active) {

		this.userID = userId;

		this.userName = userName;

		this.store = store;

		this.email = email;

		this.contactNo = contactNo;

		this.usertype = usertype;

		this.active =active;

	}


	public User(int userId, String userName,
String userPassword, Store store,

				String email,
String contactNo, Usertype usertype, int active) {

				this.userID =
userId;

				this.userName =
userName;

				this.userPassword
= userPassword;

				this.store =
store;

				this.email =
email;

				this.contactNo =
contactNo;

				this.usertype =
usertype;

				this.active
=active;

			}


	public User() {

		//does nothing

	}

	@Id

	@GeneratedValue(strategy =
GenerationType.IDENTITY)

	private int userID;


	@NotNull

	@Column(name = "userName", nullable = false)

	private String userName;


	@NotNull

	@Column(name = "userPassword", nullable =
false)

	private String userPassword;


	@OneToOne

	@JoinColumn(name="storeId")

	private Store store;

	@NotNull

	@Column(name = "email", nullable = false)

	private String email;


	@NotNull

	@Column(name = "active", nullable = false)

	private int active;
```

```
	 public Store getStore() {
		return store;
	}
	public void setStore(Store store) {
		this.store = store;
	}
	@NotNull
	@Column(name = "contactNo", nullable = false)
	private String contactNo;

	@OneToOne
	@JoinColumn(name="usertype")
	private Usertype usertype;
	public int getUserID() {
		return userID;
	}
	public void setUserID(int userID) {
		this.userID = userID;
	}
	public String getUserName() {
		return userName;
	}
	public void setUserName(String userName) {
		this.userName = userName;
	}
	public String getUserPassword() {
		return userPassword;
	}
	public void setUserPassword(String
userPassword) {
		this.userPassword = userPassword;
	}
	public String getEmail() {
		return email;
	}
	public void setEmail(String email) {
		this.email = email;
	}
	public String getContactNo() {
		return contactNo;
	}
	public void setContactNo(String contactNo) {
		this.contactNo = contactNo;
	}
	public Usertype getUsertype() {
		return usertype;
	}
	public void setUsertype(Usertype usertype) {
		this.usertype = usertype;
	}
	public int getActive() {
		return active;
	}
	public void setActive(int active) {
		this.active = active;
	}
}
```

Usertype.java

```
package edu.up.cas.sp.model;
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
```

```
import javax.persistence.Table;

import javax.validation.constraints.NotNull;

@Entity

@Table(name="USERTYPE")

public class Usertype {

	@Id

	@GeneratedValue(strategy =
GenerationType.IDENTITY)

	private int usertypeId;



	@NotNull

	@Column(name = "usertypeName", nullable =
false)

	private String usertypeName;

	public int getUsertypeId() {

		return usertypeId;

	}

	public void setUsertypeId(int usertypeId) {

		this.usertypeId = usertypeId;

	}

	public String getUsertypeName() {

		return usertypeName;

	}

	public void setUsertypeName(String
usertypeName) {

		this.usertypeName = usertypeName;

	}

}
```

AreaService.java

```
package edu.up.cas.sp.service;

import java.util.List;

import edu.up.cas.sp.model.Area;

public interface AreaService {

	List<Area> findAllArea();

	void saveArea(Area area);

	//List<Area> findByName(String areaName);

	void deleteArea(Integer areaId);

	void updateArea(Area area);

}
```

AreaServiceImpl.java

```
package edu.up.cas.sp.service;

import java.util.List;

import
org.springframework.beans.factory.annotation.Autowired;

import org.springframework.stereotype.Service;

import
org.springframework.transaction.annotation.Transactional
;

import edu.up.cas.sp.dao.AreaDao;

import edu.up.cas.sp.model.Area;

@Service("areaService")

@Transactional

public class AreaServiceImpl implements AreaService {

	@Autowired

    private AreaDao dao;



	public List<Area> findAllArea() {

		return dao.findAllArea();

	}

	public void saveArea(Area area) {

		dao.saveArea(area);

	}

	public void deleteArea(Integer areaId) {

		dao.deleteArea(areaId);

	}

	public void updateArea(Area area) {
```

```
                Area entity =
dao.findById(area.getAreaId());
                if(entity != null) {

        entity.setAreaName(area.getAreaName());
                }
        }
}
```

InventoryService.java

```
package edu.up.cas.sp.service;

import java.util.List;

import edu.up.cas.sp.model.Inventory;

public interface InventoryService {

        Inventory findById(Integer inventoryId);

        List<Inventory> findAllInventory();

        List<Inventory> findInventoryByStoreId(Integer
storeId);

        List<Inventory> findInventory(Inventory
inventory);

        void saveInventory(Inventory inventory);

        void updateInventory(Inventory inventory);

        void deleteInventory(Integer inventoryId);

        void updateInventoryQuantity(Inventory
inventory);

}
```

InventoryServiceImpl.java

```
package edu.up.cas.sp.service;

import java.util.List;

import
org.springframework.beans.factory.annotation.Autowired;

import org.springframework.stereotype.Service;

import
org.springframework.transaction.annotation.Transactional
;

import edu.up.cas.sp.dao.InventoryDao;
import edu.up.cas.sp.model.Inventory;

@Service("inventoryService")

@Transactional

public class InventoryServiceImpl implements
InventoryService{

        @Autowired

    private InventoryDao dao;



        public List<Inventory> findAllInventory() {

                return dao.findAllInventory();

        }

        public List<Inventory>
findInventoryByStoreId(Integer storeId) {

                return
dao.findInventoryByStoreId(storeId);

        }

        public void saveInventory(Inventory inventory)
{

                dao.saveInventory(inventory);

        }

        public void deleteInventory(Integer
inventoryId) {

                dao.deleteInventory(inventoryId);

        }

        public void updateInventory(Inventory
inventory) {

                Inventory entity =
dao.findById(inventory.getInventoryId());

                if(entity != null) {


        entity.setItemCount(inventory.getItemCount());

                }

        }

        public void updateInventoryQuantity(Inventory
inventory) {
```

```
                Inventory entity =
dao.findById(inventory.getInventoryId());
                if(entity != null) {
                        //update quantity

        entity.setItemCount(entity.getItemCount()-
inventory.getItemCount());
                }
        }
        public Inventory findById(Integer inventoryId)
{
                return dao.findById(inventoryId);
        }
        public List<Inventory> findInventory(Inventory
inventory) {
                return dao.findInventory(inventory);
        }
}
```

ItemService.java

```
package edu.up.cas.sp.service;
import java.util.List;
import edu.up.cas.sp.model.Item;
public interface ItemService {
        Item findById(Integer itemId);
        void saveItem(Item item);
        void updateItem(Item item);
        void deleteItem(Integer itemId);
        List<Item> findAllItems();
}
```

ItemServiceImpl.java

```
package edu.up.cas.sp.service;
import java.util.List;
import
org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import
org.springframework.transaction.annotation.Transactional
;
import edu.up.cas.sp.dao.ItemDao;
import edu.up.cas.sp.model.Item;
@Service("itemService")
@Transactional
public class ItemServiceImpl implements ItemService{
        @Autowired
    private ItemDao dao;

        public Item findById(Integer itemId) {
                return dao.findById(itemId);
        }
        public List<Item> findAllItems() {
                return dao.findAllItems();
        }
        public void saveItem(Item item) {
                dao.saveItem(item);
        }
        public void deleteItem(Integer itemId) {
                dao.deleteItem(itemId);
        }

        public void updateItem(Item item) {
                Item entity =
dao.findById(item.getItemId());
                if(entity != null) {

        entity.setItemname(item.getItemname());

        entity.setBarCode(item.getBarCode());
```

```
		entity.setItemdesc(item.getItemdesc());


		entity.setPrice(item.getPrice());

			}

		}

}
```

PaymentService.java

```
package edu.up.cas.sp.service;

import java.util.List;

import edu.up.cas.sp.model.Payment;

public interface PaymentService {

		void savePayments(List<Payment> paymentList);

		List<Payment> getPaymentsByStoreToday(Integer
storeId);

		List<Payment> getPaymentsByAreaToday(Integer
areaId);

		List<Payment> getAllPaymentsToday();

		List<Payment> getPaymentsByStoreByDate(Integer
storeId, String dateFrom, String dateTo);

		List<Payment> getPaymentsByAreaByDate(Integer
areaId, String dateFrom, String dateTo);

		List<Payment> getAllPaymentsByDate(String
dateFrom, String dateTo);

}
```

PaymentServiceImpl.java

```
package edu.up.cas.sp.service;

import java.util.List;

import
org.springframework.beans.factory.annotation.Autowired;

import org.springframework.stereotype.Service;

import
org.springframework.transaction.annotation.Transactional
;

import edu.up.cas.sp.dao.PaymentDao;

import edu.up.cas.sp.model.Payment;

@Service("paymentService")

@Transactional

public class PaymentServiceImpl implements
PaymentService{

		@Autowired

    private PaymentDao dao;



		public void savePayments(List<Payment>
paymentList) {

			dao.savePayments(paymentList);

		}

		public List<Payment>
getPaymentsByStoreToday(Integer storeId) {

			return
dao.getPaymentsByStoreToday(storeId);

		}

		public List<Payment>
getPaymentsByAreaToday(Integer areaId) {

			return
dao.getPaymentsByAreaToday(areaId);

		}

		public List<Payment> getAllPaymentsToday() {

			return dao.getAllPaymentsToday();

		}

		public List<Payment>
getPaymentsByStoreByDate(Integer storeId,

				String dateFrom, String
dateTo) {

			return
dao.getPaymentsByStoreByDate(storeId, dateFrom, dateTo);

		}

		public List<Payment>
getPaymentsByAreaByDate(Integer areaId,

				String dateFrom, String
dateTo) {

			return
dao.getPaymentsByAreaByDate(areaId, dateFrom, dateTo);
```

```
	}

	public List<Payment>
getAllPaymentsByDate(String dateFrom, String dateTo) {

		return
dao.getAllPaymentsByDate(dateFrom, dateTo);

	}

}
```

PaymentTypeService.java

```
package edu.up.cas.sp.service;

import java.util.List;

import edu.up.cas.sp.model.PaymentType;

public interface PaymentTypeService {

	List<PaymentType> findAllPAymentTypes();

}
```

PaymentTypeServiceImpl.java

```
package edu.up.cas.sp.service;

import java.util.List;

import
org.springframework.beans.factory.annotation.Autowired;

import org.springframework.stereotype.Service;

import
org.springframework.transaction.annotation.Transactional
;

import edu.up.cas.sp.dao.PaymentTypeDao;

import edu.up.cas.sp.model.PaymentType;

@Service("paymentTypeService")

@Transactional

public class PaymentTypeServiceImpl implements
PaymentTypeService{

	@Autowired

    private PaymentTypeDao dao;



	public List<PaymentType> findAllPAymentTypes()
{

		return dao.findAllPAymentTypes();

	}

}
```

ReceiptService.java

```
package edu.up.cas.sp.service;

import java.util.List;

import edu.up.cas.sp.model.Receipt;

import edu.up.cas.sp.model.Transaction;

public interface ReceiptService {

	Receipt findByKey(Integer receiptId);

	void saveReceipt(Receipt receipt);

	void updateReceipt(Receipt receipt);

	List<Receipt> getAllReceiptsToday();

	List<Receipt> getAllReceiptsByDate(String
dateFrom, String dateTo);

	List<Receipt> getReceiptsByAreaByDate(Integer
areaId, String dateFrom, String dateTo);

	List<Receipt> getReceiptsByStoreByDate(Integer
storeId, String dateFrom, String dateTo);

	List<Receipt> getReceiptsByAreaToday(Integer
areaId);

	List<Receipt> getReceiptsByStoreToday(Integer
storeId);

	List<Transaction>
getTransactionsByReceiptIdToday(Integer receiptId);

}
```

ReceiptServiceImpl.java

```
package edu.up.cas.sp.service;

import java.util.List;

import
org.springframework.beans.factory.annotation.Autowired;

import org.springframework.stereotype.Service;

import
org.springframework.transaction.annotation.Transactional
;

import edu.up.cas.sp.dao.ReceiptDao;

import edu.up.cas.sp.model.Receipt;
```

```
import edu.up.cas.sp.model.Transaction;

@Service("receiptService")

@Transactional

public class ReceiptServiceImpl implements
ReceiptService{

	@Autowired

    private ReceiptDao dao;



	public void saveReceipt(Receipt receipt) {

		dao.saveReceipt(receipt);

	}

	public void updateReceipt(Receipt receipt) {

		Receipt entity =
dao.findByKey(receipt.getReceiptId());

		if(entity != null) {


	entity.setAmountDue(receipt.getAmountDue());


	entity.setAmountPaid(receipt.getAmountDue());


	entity.setAmountChange(receipt.getAmountChange
());

		}

	}

	public List<Receipt> getAllReceiptsToday() {

		return dao.getAllReceiptsToday();

	}

	public List<Receipt>
getReceiptsByAreaToday(Integer areaId) {

		return
dao.getReceiptsByAreaToday(areaId);

	}

	public List<Receipt>
getReceiptsByStoreToday(Integer storeId) {

		return
dao.getReceiptsByStoreToday(storeId);

	}

	public List<Transaction>
getTransactionsByReceiptIdToday(Integer receiptId) {

		return
dao.getTransactionsByReceiptIdToday(receiptId);

	}

	public Receipt findByKey(Integer receiptId) {

		return dao.findByKey(receiptId);

	}

	public List<Receipt>
getAllReceiptsByDate(String dateFrom, String dateTo) {

		return
dao.getAllReceiptsByDate(dateFrom, dateTo);

	}

	public List<Receipt>
getReceiptsByAreaByDate(Integer areaId,

			String dateFrom, String
dateTo) {

		return
dao.getReceiptsByAreaByDate(areaId, dateFrom, dateTo);

	}

	public List<Receipt>
getReceiptsByStoreByDate(Integer storeId,

			String dateFrom, String
dateTo) {

		return
dao.getReceiptsByStoreByDate(storeId, dateFrom, dateTo);

	}

}
```

ReturnedItemService.java

```
package edu.up.cas.sp.service;

import java.util.List;

import edu.up.cas.sp.model.ReturnedItem;

public interface ReturnedItemService {

	void saveReturnedItems(List<ReturnedItem>
returnedItems);
```

```
}
```

ReturnedItemServiceImpl.java

```
package edu.up.cas.sp.service;

import java.util.List;

import
org.springframework.beans.factory.annotation.Autowired;

import org.springframework.stereotype.Service;

import
org.springframework.transaction.annotation.Transactional
;

import edu.up.cas.sp.dao.ReturnedItemDao;

import edu.up.cas.sp.model.ReturnedItem;

@Service("returnedItemService")

@Transactional

public class ReturnedItemServiceImpl implements
ReturnedItemService{



        @Autowired

    private ReturnedItemDao dao;

        public void
saveReturnedItems(List<ReturnedItem> returnedItems) {

                dao.saveReturnedItems(returnedItems);

        }

}
```

ReturnItemVoucherService.java

```
package edu.up.cas.sp.service;

import java.util.List;

import edu.up.cas.sp.model.ReturnItemVoucher;

public interface ReturnItemVoucherService {

        void saveReturnItemVoucher(ReturnItemVoucher
returnItem);

        void updateReturnItemVoucher(ReturnItemVoucher
returnItem);

        List<ReturnItemVoucher>
getReturnItemVoucherById(Integer returnItemVoucherId);

        List<ReturnItemVoucher>
getReturnItemVoucher(ReturnItemVoucher
returnItemVoucher);

        List<ReturnItemVoucher>
getReturnItemVoucherByReceiptId(Integer receiptId);

}
```

ReturnItemVoucherServiceImpl.java

```
package edu.up.cas.sp.service;

import java.util.List;

import
org.springframework.beans.factory.annotation.Autowired;

import org.springframework.stereotype.Service;

import
org.springframework.transaction.annotation.Transactional
;

import edu.up.cas.sp.dao.ReturnItemVoucherDao;

import edu.up.cas.sp.model.ReturnItemVoucher;

@Service("returnItemService")

@Transactional

public class ReturnItemVoucherServiceImpl  implements
ReturnItemVoucherService{

        @Autowired

    private ReturnItemVoucherDao dao;



        public void
saveReturnItemVoucher(ReturnItemVoucher returnItem) {


        dao.saveReturnItemVoucher(returnItem);

        }

        public List<ReturnItemVoucher>
getReturnItemVoucherById(

                        Integer returnItemVoucherId)
{

                return
dao.getReturnItemVoucherById(returnItemVoucherId);

        }
```

```
	public void
updateReturnItemVoucher(ReturnItemVoucher returnItem) {
		ReturnItemVoucher entity =
dao.findByKey(returnItem.getReturnItemVoucherId());
		if(entity != null) {
			//update status
			entity.setStatus("claimed");
		} else {
			System.out.println("entity
is null");
		}
	}
	public List<ReturnItemVoucher>
getReturnItemVoucher(
			ReturnItemVoucher
returnItemVoucher) {
		return
dao.getReturnItemVoucher(returnItemVoucher);
	}
	public List<ReturnItemVoucher>
getReturnItemVoucherByReceiptId(
			Integer receiptId) {
		return
dao.getReturnItemVoucherByReceiptId(receiptId);
	}
}
```

RMCService.java

```
package edu.up.cas.sp.service;
import java.util.List;
import edu.up.cas.sp.model.Transaction;
public interface RMCService {
	List<Transaction> findAllTransactions();
}
```

RMCServiceImpl.java

```
package edu.up.cas.sp.service;
import java.util.List;
import
org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import
org.springframework.transaction.annotation.Transactional
;
import edu.up.cas.sp.dao.RMCDao;
import edu.up.cas.sp.model.Transaction;
@Service("rmcService")
@Transactional
public class RMCServiceImpl implements RMCService{
	@Autowired
    private RMCDao dao;
	public List<Transaction> findAllTransactions()
{
		return dao.getAllTransactions();
	}
}
```

StoreService.java

```
package edu.up.cas.sp.service;
import java.util.List;
import edu.up.cas.sp.model.Store;
public interface StoreService {
	List<Store> findAllStores();
	Store findById(Integer storeId);
	void updateStore(Store store);
	void deleteStore(Integer storeId);
	void saveStore(Store store);
}
```

StoreServiceImpl.java

```
package edu.up.cas.sp.service;
import java.util.List;
```

```
import
org.springframework.beans.factory.annotation.Autowired;

import org.springframework.stereotype.Service;

import
org.springframework.transaction.annotation.Transactional
;

import edu.up.cas.sp.dao.StoreDao;

import edu.up.cas.sp.model.Store;

@Service("storeService")

@Transactional

public class StoreServiceImpl implements StoreService{

	@Autowired

    private StoreDao dao;



	public List<Store> findAllStores() {

		return dao.findAllStores();

	}

	public void saveStore(Store store) {

		dao.saveStore(store);

	}

	public void deleteStore(Integer storeId) {

		dao.deleteStore(storeId);

	}

	public void updateStore(Store store) {

		Store entity =
dao.findById(store.getStoreId());

		if(entity != null) {


	entity.setArea(store.getArea());


	entity.setBranchName(store.getBranchName());


	entity.setTin(store.getTin());


	entity.setAddress(store.getAddress());


	entity.setCoordinates(store.getCoordinates());

		}

	}

	public Store findById(Integer storeId) {

		return dao.findById(storeId);

	}

}
```

TransactionService.java

```
package edu.up.cas.sp.service;

import java.util.List;

import edu.up.cas.sp.model.Transaction;

public interface TransactionService {

	void saveTransaction(Transaction transaction);

	void saveTransaction(List<Transaction>
transactions);

	List<Transaction> getAllTransactions();

	List<Transaction>
getTransactionsByArea(Integer areaId);

	List<Transaction>
getTransactionsByStore(Integer storeId);

	List<Transaction>
getTransactionsByReceipt(Integer receiptId);

}
```

TransactionServiceImpl.java

```
package edu.up.cas.sp.service;

import java.util.List;

import
org.springframework.beans.factory.annotation.Autowired;

import org.springframework.stereotype.Service;

import
org.springframework.transaction.annotation.Transactional
;

import edu.up.cas.sp.dao.TransactionDao;

import edu.up.cas.sp.model.Transaction;
```

```
@Service("transactionService")

@Transactional

public class TransactionServiceImpl implements
TransactionService{

	@Autowired

    private TransactionDao dao;



	public void saveTransaction(Transaction
transaction) {

		dao.saveTransaction(transaction);

	}

	public void saveTransaction(List<Transaction>
transactions) {

		dao.saveTransaction(transactions);

	}

	public List<Transaction> getAllTransactions()
{

		return dao.getAllTransactions();

	}

	public List<Transaction>
getTransactionsByArea(Integer areaId) {

		return
dao.getTransactionsByArea(areaId);

	}

	public List<Transaction>
getTransactionsByStore(Integer storeId) {

		return
dao.getTransactionsByStore(storeId);

	}

	public List<Transaction>
getTransactionsByReceipt(Integer receiptId) {

		return
dao.getTransactionsByReceipt(receiptId);

	}

}
```

UserService.java

```
package edu.up.cas.sp.service;

import java.util.List;

import edu.up.cas.sp.model.User;

public interface UserService {

	User findById(Integer userId);

	User findByName(String username);

	User findByNameAndPassword(String username,
String password);

	List<User> findAllUsers();

	List<User> findByStoreId(Integer storeId);

	void saveUser(User user);

	void updateUser(User user);

	void changePassword(User user);

	void enableDisableUser(User user);

	void deleteUser(Integer userId);

}
```

UserServiceImpl.java

```
package edu.up.cas.sp.service;

import java.util.List;

import
org.springframework.beans.factory.annotation.Autowired;

import
org.springframework.security.crypto.bcrypt.BCrypt;

import org.springframework.stereotype.Service;

import
org.springframework.transaction.annotation.Transactional
;

import edu.up.cas.sp.dao.UserDao;

import edu.up.cas.sp.model.User;

@Service("userService")

@Transactional

public class UserServiceImpl implements UserService{

	@Autowired

    private UserDao dao;
```

```
	public User findById(Integer userId) {
		return dao.findById(userId);
	}
	public User findByName(String username) {
		return dao.findByName(username);
	}
	public List<User> findAllUsers() {
		return dao.findAllUsers();
	}
	public User findByNameAndPassword(String
username, String password) {
		User user = dao.findByName(username);
		if(user!=null) {
			if(BCrypt.checkpw(password,
user.getUserPassword())) {
				//match
				return user;
			}
			return null;
		} else {
			return null;
		}
	}
	public void saveUser(User user) {
		dao.SaveUser(user);
	}
	public void deleteUser(Integer userId) {
		dao.deleteUser(userId);
	}
	public void updateUser(User user) {
		User entity =
dao.findById(user.getUserID());
		if(entity != null) {
			entity.setUserName(user.getUserName());
			entity.setUsertype(user.getUsertype());
			entity.setStore(user.getStore());
			entity.setEmail(user.getEmail());
			entity.setContactNo(user.getContactNo());
		}
	}
	public void changePassword(User user) {
		User entity =
dao.findById(user.getUserID());
		if(entity != null) {
			entity.setUserPassword(user.getUserPassword())
;
		}
	}
	public List<User> findByStoreId(Integer
storeId) {
		return dao.findByStoreId(storeId);
	}
	public void enableDisableUser(User user) {
		User entity =
dao.findById(user.getUserID());
		if(entity != null) {
			entity.setActive(user.getActive());
		}
	}
}
```

UsertypeService.java

```
package edu.up.cas.sp.service;

import java.util.List;

import edu.up.cas.sp.model.Usertype;

public interface UsertypeService {

	List<Usertype> findAllUsertypes();

}
```

UsertypeServiceImpl.java

```
package edu.up.cas.sp.service;

import java.util.List;

import
org.springframework.beans.factory.annotation.Autowired;

import org.springframework.stereotype.Service;

import
org.springframework.transaction.annotation.Transactional
;

import edu.up.cas.sp.dao.UsertypeDao;

import edu.up.cas.sp.model.Usertype;

@Service("usertypeService")

@Transactional

public class UsertypeServiceImpl implements
UsertypeService{

	@Autowired

    private UsertypeDao dao;



	public List<Usertype> findAllUsertypes() {

		return dao.findAllUsertypes();

	}

}
```

application.properties

```
jdbc.driverClassName = com.mysql.jdbc.Driver
jdbc.url = jdbc:mysql://localhost:3306/mountainshop
jdbc.username = root
jdbc.password = Roldybuabey1
hibernate.dialect = org.hibernate.dialect.MySQLDialect
hibernate.show_sql = true
hibernate.format_sql = true
```

login.css

```
@CHARSET "UTF-8";
/*
over-ride "Weak" message, show font in dark grey
*/
.progress-bar {
    color: #333;
}
/*
Reference:
http://www.bootstrapzen.com/item/135/simple-login-form-
logo/
*/
* {
    -webkit-box-sizing: border-box;
            -moz-box-sizing: border-box;
                 box-sizing: border-box;
          outline: none;
}
    .form-control {
           position: relative;
           font-size: 16px;
           height: auto;
           padding: 10px;
                  @include box-sizing(border-box);
                  &:focus {
                    z-index: 2;
                  }
          }
body {
          background:
url(http://i.imgur.com/GHr12sH.jpg) no-repeat center
center fixed;
    -webkit-background-size: cover;
    -moz-background-size: cover;
    -o-background-size: cover;
    background-size: cover;
}
.login-form {
          margin-top: 60px;
}
form[role=login] {
          color: #5d5d5d;
          background: #f2f2f2;
          padding: 26px;
          border-radius: 10px;
          -moz-border-radius: 10px;
          -webkit-border-radius: 10px;
}
          form[role=login] img {
                   display: block;
                   margin: 0 auto;
                   margin-bottom: 35px;
          }
          form[role=login] input,
          form[role=login] button {
                   font-size: 18px;
                   margin: 16px 0;
          }
          form[role=login] > div {
                   text-align: center;
          }

.form-links {
          text-align: center;
          margin-top: 1em;
          margin-bottom: 50px;
}
          .form-links a {
                   color: #fff;
          }
```

main.css

```
@CHARSET "ISO-8859-1";
/*@import "compass/css3"; */
/*@import
url(http://fonts.googleapis.com/css?family=Raleway:100,3
00,400,700); */
```

```
.error {
        color: red;
}
.red {
    color: red;
}
.one-line {
        display: inline-block;
    overflow: hidden;
    white-space: nowrap;
    float: left;
        max-width: 50%;
}
.highchartContainer {
        width:25%;
}
.table {
        max-width: none;
        table-layout: fixed;
        word-wrap: break-word;
}
.returnInput {
        width: 30px;
}
.inputReceipt {
        width: 100px;
}
.half {
        width: 50%;
}
.posReceipt {
        font-family: "courier";
        text-align: center;
}
#custom-search-input{
    padding: 3px;
    border: solid 1px #E4E4E4;
    border-radius: 6px;
    background-color: #fff;
}
#custom-search-input input{
    border: 0;
    box-shadow: none;
}
#custom-search-input button{
    margin: 2px 0 0 0;
    background: none;
    box-shadow: none;
    border: 0;
    color:        #4169E1;
    padding: 0 8px 0 10px;
    border-left: solid 1px #ccc;
}
#custom-search-input button:hover{
    border: 0;
    box-shadow: none;
    border-left: solid 1px #ccc;
}
#custom-search-input .glyphicon-search{
    font-size: 23px;
}
#rightDiv {
        border-right: 3px solid gray;
}
#leftDiv {
        border-left: 3px solid gray;
}
#purchaseItemButton {
        float: right;
}
#receiptDiv {
        margin-top: 15%;
}
#custom-controls {
        font-size: medium;
}
.right {
        float: right;
        text-align: right;
}
.left {
        float: left;
        text-align: left;
}
.topmargin {
        margin-top:15px;
}
.table-borderless tbody tr td, .table-borderless tbody
tr th, .table-borderless thead tr th {
    border: none;

}
#rmcMap {
    display: inline-block;
    position: relative;
    width: 50%;
    height: 500px;
    right: 25%;
}
#mapContainer {
        width:100%;
        height: 500px;
}
#graphContainer, #gaugeContainer {
        display: inline-block;
        float:left;
    width:25%;
    height: 250px;
}
#barGraphContainer1 {
    float: right;
    width: 25%;
    height: 250px;
    position: relative;
    bottom: 500px;
    display: inline-block;
}
#barGraphContainer2 {
    float: right;
    width: 25%;
    height: 250px;
    position: relative;
    bottom: 250px;
    display: inline-block;
    right: -25%;
}
<!-- Override header in RMC page-->
.rmc-header {
        padding-bottom: 0px;
}
body {
font-family:'Raleway';
}
/*.wrapper {
width:95%;
} */
.header-container {
  color: #fff;
  height: 50px;
  background: black;
  background:rgba(0, 0, 0, .5);
  border-bottom: solid 1px black;
}
.header {
  margin:0 auto;
}
.header h1 {
  font-weight:400;
  margin:0;
  margin-left:.5em;
  padding-top:5px;
}
.blocks {
    margin: 0 auto;
    display: inline-block;
    background: #f16bf3;
    position: relative;
    bottom: 220px;
```

```
}
.block {
  float:left;
  margin-top:0;
  /*position:relative;*/
  /*background: black;*/
  color: #fff;
  width: 12.5%;
 /* margin-right:1%; */
  min-width:155px;
  height: 220px;
  /*box-shadow: 10px 10px 0 rgba(0, 0, 0, .3);*/
  text-align: center;
  overflow:hidden;

  -webkit-backface-visibility: hidden;

  border:solid 1px #000;
}
.heading {
  height:30px;
  position:relative;
  font-size: 1.4em;
  width:100%;
  /*background:#f16bf3;*/
  /*background:#00FFFF;*/
  padding: 25px 0;
  border-bottom: solid 1px;
}
.num, .currentView {
position:relative;
/*background:#f16bf3;*/
font-size: 1.5em;
padding-top: 40px;
}
.green {
/*background: black;*/
background: #5eda37;
}
.orange {
color: #f66538;
}
.purple {
color: #f16bf3;
}
.yellow {
color: #e3ea77;
}
.blue {
background: #7ca6de;
/*color: #7ca6de;*/
}
.black {
         color: black;
}
/*High charts*/
.highcharts-yaxis-grid .highcharts-grid-line {
         display: none;
}
```

area.js

```
/**
 * JQuery file for Area page
 */

//Check if user is logged in

$(function() {

         /**
          * Start of knockout.js
          *
          */


         function Area(area) {

             this.areaId = ko.observable(area.areaId);

             this.areaName =
ko.observable(area.areaName);

             this.zoomLevel =
ko.observable(area.zoomLevel);

             this.delAreaTitle = ko.observable("Delete
area " + area.areaId);

             this.editAreaTitle = ko.observable("Edit
area " + area.areaId);

         }

         // Overall viewmodel for this screen, along
with initial state

         function AreaViewModel() {

                  var self = this;



                  //used in cookies

                  self.loggedInUser =
ko.observable(Cookies.get('username'));

                  self.loggedInUsertype =
ko.observable(Cookies.get('usertype'));

                  self.loggedInUserStoreId =
ko.observable(Cookies.get('userStoreId'));



                  self.areas = ko.observableArray([]);



                  //copy of areas array

                  self.areasCopy =
ko.observableArray([]);



                  //for Add/Edit area form

                  self.areaid = ko.observable(0);

                  self.areaname = ko.observable("");
```

```
                self.zoomlevel = ko.observable(0);

                //These are used in search
                self.searchString =
ko.observable("");
                self.searchResultsArray =
ko.observableArray([]);

                //for paging
            self.pageSize = ko.observable(5);
            self.NumberPages = ko.observable(1);
            self.currPage = ko.observable(1);
            self.pagesArray = ko.observableArray([]);
            self.maxNumberPages = ko.observable(1);

                // get the list of areas on first
load from DB
            $.getJSON("get-areas", function(allData) {
                var mappedItems = $.map(allData,
function(area) { return new Area(area); });

                //make a copy of the items
                self.areasCopy(mappedItems);

                //Do paging on first load
                self.doPaging(self.pageSize());
            });

            self.displayAddArea = function() {
                //change the modal title and button
text
                $(".modal-title").text("Add new
area");

        $("#editAreaSumbitButton").text("Add");
                //Clear the values inside
the form
                self.areaid(0);
                self.areaname("");
                self.zoomlevel("");

                //show the Edit modal
        $('#editAreaModal').modal('show');
            };

            self.displayArea = function() {
                //change the modal title and button
text
                $(".modal-title").text("Edit item");
        $("#editAreaSumbitButton").text("Update");

                //map the values to modal form
                self.areaid(this.areaId());
        self.areaname(this.areaName());
        self.zoomlevel(this.zoomLevel());

                //show the modal
        $('#editAreaModal').modal('show');
            };

            self.cancelEdit = function() {
                //Clear the values of the form
                self.areaid(0);
                self.areaname("");
```

```
        };

        self.editArea = function() {
            var areaId = this.areaid();
            var areaName = this.areaname();

            //Add new area
            if(areaId==0) {
                var url = 'add-area';
                $.ajax({
                    url: url,
                    dataType: 'json',
                    data: {areaName: areaName},
                    success: function(area) {
    //close the modal
    $('[data-dismiss=modal]').click();

    if(self.searchString() != '') {
    self.searchResultsArray.push({
                areaId : ko.observable(area.areaId),
                areaName : ko.observable(area.areaName),
            delAreaTitle : ko.observable("Delete area " + area.areaId),
            editAreaTitle : ko.observable("Edit area " + area.areaId)
    });
                                    }
    else {
    self.areas.push({
                areaId : ko.observable(area.areaId),
                areaName : ko.observable(area.areaName),
            delAreaTitle : ko.observable("Delete area " + area.areaId),
            editAreaTitle : ko.observable("Edit area " + area.areaId)
    });
    }

    //add also in the copy
    self.areasCopy.push({
    areaId : ko.observable(area.areaId),
    areaName : ko.observable(area.areaName),
    delAreaTitle : ko.observable("Delete area " + area.areaId),
    editAreaTitle : ko.observable("Edit area " + area.areaId)
    });
```

```
        //Do paging
self.doPaging(self.pageSize(), self.currPage());

        //notify that adding is successful
        $.notify({
        // options
        icon: 'glyphicon glyphicon-ok',
        message: 'Area successfully added'
        },{
        // settings
        type: 'success',
        delay: 1000,
        offset: 55,
        });
                                        },
                                        error:
function(jqXHR, textStatus, errorThrown) {
        errorMessage = "Cannot add area. Please check
if area " +
        "name already exists. Or try again later.";
        //Display error message
        $("#editAreaError").html(errorMessage);
                                        }
                                });
                        }
                        //Edit area
                        else {
                                var url = 'update-area';
                                        $.ajax({
                                                type:
'GET',
                                                url: url,
                                                data:
{areaId: areaId, areaName: areaName},
                                                success:
function() {
        //loop through the items and update the value
        for(var i = 0; i<self.areas().length; i++) {
                if (self.areas()[i].areaId() ==
areaId) {
        self.areas()[i].areaName(areaName);
break;
}
        }
        //loop through the items copy and update the
value
        for(var i = 0; i<self.areasCopy().length; i++)
{
                if (self.areasCopy()[i].areaId() ==
areaId) {
```

```
                self.areasCopy()[i].areaName(areaName);

break;

}

            }

            //Do paging

self.doPaging(self.pageSize(), self.currPage());



$.notify({

                        // options

                        icon: 'glyphicon glyphicon-ok',

                        message: 'Area successfully updated'

            },{

                        // settings

                        type: 'success',

                        delay: 1000,

                        offset: 55,

            });

            $('[data-dismiss=modal]').click();

                                                    },

                                                    error:
function() {

                errorMessage = "Cannot edit area. Please check
if area " +

                        "name already exists. Or try again
later.";

                //Display error message

                $("#editAreaError").html(errorMessage);

                return false;

                                                    },

                                        });

                                }

                                return true;

                    };

                    self.removeArea = function(area, event) {

                        // get itemId of the row

                                var areaId =
event.currentTarget.id;



                                bootbox.confirm({

                                        message: "You are
about to delete area " + areaId + ".\nDo you want to
proceed?",

                                        closeButton:
false,

                                        size: "small",

                                    callback:
function(result){

                                        if(result) {

                var url = 'delete-' + areaId + '-area';

                $.ajax({
```

```
        url: url,

        success: function() {

                //remove the element from the table

                if(self.searchString() != '') {

                        //remove from
searchResultsArray

                        for(var x in
self.searchResultsArray()) {


        if(self.searchResultsArray()[x].areaId() ==
areaId) {


        self.searchResultsArray.remove(self.searchResu
ltsArray()[x]);

                                        break;

                                }

                        }

        }

        else {

                self.areas.remove(area);

        }

                //remove also from the copy

                for(var x in self.areasCopy()) {

                        if(self.areasCopy()[x].areaId() == areaId) {


                        self.areasCopy.remove(self.areasCopy()[x]);

                                        break;

                        }

                }

                //Do paging

self.doPaging(self.pageSize(), self.currPage());


                $.notify({

                        // options

                        icon: 'glyphicon glyphicon-
ok',

                        message: 'Area successfully
deleted'

                },{

                        // settings

                        type: 'success',

                        delay: 1000,

                        offset: 55,

                });

        },
```

```
        error: function(jqXHR, textStatus,
errorThrown) {

            alert("error:" + textStatus + "
exception:" + errorThrown);

        },

        });
                                            }
                                }
                            });
            };

            self.searchAreas = function() {
                var searchString =
self.searchString();
                self.searchResultsArray.removeAll();

                if(searchString != '') {
                        for(var x in
self.areasCopy()) {
                                if
((self.areasCopy()[x].areaName().toLowerCase().indexOf(s
earchString.toLowerCase()) >= 0)) {

        self.searchResultsArray.push(self.areasCopy()[
x]);
                                }
                        }
                }
                else {

        self.searchResultsArray(self.areasCopy().slice
());
                }

                //Do paging
               self.doPaging(self.pageSize(),
self.currPage());
            };

            self.doPaging = function(pageSize,
nextPage) {

                var areasArray =
ko.observableArray([]);

                //make a copy of the results
                if(self.searchString() != '') {

        areasArray(self.searchResultsArray.slice());
                }
                else {

        areasArray(self.areasCopy().slice());
                }

                //clear areas
                self.areas.removeAll();

                //set current page as next page if
nextPage is defined
                if(nextPage)
                        self.currPage(nextPage);

                //set page size
                self.pageSize(pageSize);

                //calculate number of pages
```

```
self.NumberPages(Math.ceil(areasArray().length/self.page
Size()));


                //clear pages array
                self.pagesArray.removeAll();


                //populate pagesArray
                for(var i = 0; i < self.NumberPages();
i++) {
                 self.pagesArray.push({
                          pageNumber:
ko.observable((i+1))
                 });
                }


                //set max number of pages

self.maxNumberPages(self.NumberPages());


                //if current page is greater than max
number of pages, set currPage = maxNumberPages
                if(self.currPage() >
self.maxNumberPages())
                 self.currPage(self.maxNumberPages());


                //if maxNumberPages is less than 1,
set currPage to 1
                if(self.maxNumberPages() < 1)
                 self.currPage(1);


                var startIndex = (self.currPage()-
1)*self.pageSize();
                 for(var i = startIndex; i <
(self.pageSize() + startIndex); i++) {
                          if(areasArray()[i]) {
                                   self.areas.push(areasArray()[i]);
                          }
                          else {
                                   break;
                          }


                 }
            };
        }
        ko.applyBindings(new AreaViewModel());
});
```

checkout.js

```
/**
 * This is the Jquery file for the pos.jsp page
 */
//before page loads, if user is admin, redirect to home
page
if(Cookies.get('usertype')==1) {
        //redirect to home page
        window.location.replace('home');
}
$(function() {
        /**
         * Start of knockout.js
         *
         */
        function
ReturnedItemVoucher(returnedItemVoucher) {
                this.returnedItemVoucherNumber =
ko.observable(returnedItemVoucher.returnItemVoucherId);
                this.returnedItemVoucherAmount =
ko.observable(returnedItemVoucher.amount);
        }
```

```
        function PaymentType(paymentType) {
                this.paymentTypeId = ko.observable(paymentType.paymentTypeId);
                this.paymentType = ko.observable(paymentType.paymentType);
        }
        function Inventory(inventory) {
                this.itemSelected = ko.observable(false);
            this.inventoryId = ko.observable(inventory.inventoryId);
            this.areaId = ko.observable(inventory.areaId);
            this.storeId = ko.observable(inventory.storeId);
            this.itemName = ko.observable(inventory.itemName);
            this.barCode = ko.observable(inventory.barCode);
            this.itemDescription = ko.observable(inventory.itemDescription);
            this.itemPrice = ko.observable(inventory.itemPrice);
            this.itemQuantity = ko.observable("");
            this.itemDiscount = ko.observable("");
        }


        function POSViewModel() {
                var self = this;


                //list of inventory
                self.inventory = ko.observableArray([]);
                self.inventoryCopy = ko.observableArray([]);
                //used in transaction
                //list of transaction items
                self.transactionItems = ko.observableArray([]);
                self.netAmountDue = ko.observable(0);
                self.vatableSale = ko.observable(0);
                self.totalItems = ko.observable(0);
                self.vat = ko.observable(0);


                //note of the contents: inventory id, description, quantity (kunin na lahat, item name, price)


                //for add-to-cart modal
                self.inventoryid = ko.observable();
                self.barcode = ko.observable("");
                self.itemdescription = ko.observable("");
                self.itemprice = ko.observable("");
                self.itemquantity = ko.observable("1");
                self.itemdiscount = ko.observable("0");
                self.areaid = ko.observable("");
                self.storeid = ko.observable("");
                self.itemname = ko.observable("");


                //for check out modal
                self.amountDue = ko.observable(0);
                self.amountChange = ko.observable(0);
                self.returnedItemVouchers = ko.observableArray([]);


                //total amount paid: summation of voucher amounts, plus amount paid
```

```
                self.totalAmountPaid =
ko.observable(0);



                //used in cookies

                self.loggedInUser =
ko.observable(Cookies.get('username'));

                self.loggedInUsertype =
ko.observable(Cookies.get('usertype'));

                self.loggedInUserStoreId =
ko.observable(Cookies.get('userStoreId'));

                self.loggedInUserId =
ko.observable(Cookies.get('userId'));



                //These are used in search

                self.searchString =
ko.observable("");

                self.searchResultsArray =
ko.observableArray([]);



                //for paging

            self.pageSize = ko.observable(10);

            self.NumberPages = ko.observable(1);

            self.currPage = ko.observable(1);

            self.pagesArray = ko.observableArray([]);

            self.maxNumberPages = ko.observable(1);



            //used in Store details

            self.storeBranchName = ko.observable("");

            self.storeAddress = ko.observable("");

            self.storeTin = ko.observable("");

            self.storeDetailsString =
ko.observable("");



            //returned item voucher

            self.voucherNumber = ko.observable("");



            //for receipt printing

            self.storeDetails =
ko.observableArray([]);



            //for payment types

            self.paymentTypes =
ko.observableArray([]);

            self.paymentTypeOptionsId =
ko.observable();

            self.paymentReferenceId = ko.observable();

            self.amountPaid = ko.observable("");

            self.paymentMethods =
ko.observableArray([]);



            //for use in identification for payment
method array

            self.paymentMethodIdentification =
ko.observable(0);



            //used to read barcodes

            $(document).anysearch({

                reactOnKeycodes: 'all',

                secondsBetweenKeypress: 1,

                searchPattern: {1: '[^~,]*'},

                excludeFocus:
'input,textarea,select,#tfield',

                enterKey: 13,

                backspaceKey: 8,

                checkIsBarcodeMilliseconds: 250,

                searchFunc: function(barCodeString) {

                 //what to do with the barcode

                 //search the inventory

                 for(var x in self.inventoryCopy()) {
```

```
                        if
(self.inventoryCopy()[x].barCode()==barCodeString) {
                                    //set the values

        self.inventoryid(self.inventoryCopy()[x].inven
toryId());

        self.barcode(self.inventoryCopy()[x].barCode()
);

        self.itemdescription(self.inventoryCopy()[x].i
temDescription());

        self.itemprice(self.inventoryCopy()[x].itemPri
ce());

        self.itemquantity(1);

        self.itemdiscount(0);

        self.areaid(self.inventoryCopy()[x].areaId());

        self.storeid(self.inventoryCopy()[x].storeId()
);

        self.itemname(self.inventoryCopy()[x].itemName
());
                                    //open modal

        $('#addToCartModal').modal('show');


                        }
                }
               }
            });


                $.ajax({
                        url: 'getPaymentTypes',
                        dataType: 'json',
                        success: function(allData) {
                                var mappedItems =
$.map(allData, function(paymentType) { return new
PaymentType(paymentType); });
                        //save to array

                self.paymentTypes(mappedItems);
                        }
                });


                var url = 'get-inventory';
                $.ajax({
                        url: url,
                        dataType: 'json',
                        data: {usertype:
self.loggedInUsertype(), storeId:
self.loggedInUserStoreId()},
                        success: function(allData) {
                                var mappedItems =
$.map(allData, function(inventory) { return new
Inventory(inventory); });


                        //make a copy of the
inventory

self.inventoryCopy(mappedItems);
                        //Do paging on first load

self.doPaging(self.pageSize());
                        }
                });


                $.ajax({
                        url: 'get-
'+self.loggedInUserStoreId()+'-store',
                        dataType: 'json',
```

```
                        data: {storeId:
self.loggedInUserStoreId()},
                        success: function(store) {

        self.storeBranchName("Recreational Outdoor
eXchange - " + store.branchName);

        self.storeAddress(store.address);

                                self.storeTin('VAT
Reg. TIN: ' + self.processTin(store.tin));

                                var storeDetails =
self.storeBranchName() + '\n' + self.storeAddress() +
'\n' + self.storeTin();

        self.storeDetailsString(storeDetails);

                                //set store
details

        self.storeDetails.push({

        storeBranchName:
ko.observable(self.storeBranchName()),

        storeAddress:
ko.observable(self.storeAddress()),

                                        cashier:
ko.observable(self.loggedInUser()),

                                        storeId:
self.loggedInUserStoreId()

                                });

                        }

                });

        self.doPayment = function() {

                //set values

self.amountDue(self.netAmountDue());

                //open modal

$('#paymentModal').modal('show');

        };

        self.addPaymentMethod = function() {

                //Clear values

self.paymentTypeOptionsId(null);

            self.paymentReferenceId("");

            self.amountPaid("");

                //open modal

$('#returnedItemVoucherModal').modal('show');

        };

        self.cancelAddPaymentMethod =
function() {

                //Clear values

self.paymentTypeOptionsId(null);

            self.paymentReferenceId("");

            self.amountPaid("");

        };

        self.processTin = function(tin) {

                var processedTin =
tin.charAt(0);

                for (var i=1; i <
tin.length; i++) {
```

```
                                if(i%3==0) {

        processedTin += ('-' + tin.charAt(i));

                                } else {

        processedTin += tin.charAt(i);

                                }

                        }

                        return processedTin;

                };


                self.computeTotalAmountPaidAndChange
= function() {

                        var totalAmountinVouchers =
0.0;

                        var
totalAmountinOtherPayments = 0.0;

                        var totalAmountPaid = 0.0;

                        for (var x in
self.returnedItemVouchers()) {

        totalAmountinVouchers +=
parseFloat(self.returnedItemVouchers()[x].returnedItemVo
ucherAmount());

                        }

                        for (var x in
self.paymentMethods()) {

        totalAmountinOtherPayments +=
parseFloat(self.paymentMethods()[x].pmAmountPaid);

                        }


                        totalAmountPaid =
(totalAmountinVouchers + totalAmountinOtherPayments);

        self.totalAmountPaid(totalAmountPaid.toFixed(2
));

                        //update change

        self.amountChange(parseFloat(totalAmountPaid -
self.amountDue()).toFixed(2));


                };


                self.isVoucherExisting =
function(voucherNumber) {

                        for(var x in
self.returnedItemVouchers()) {

        if(self.returnedItemVouchers()[x].returnedItem
VoucherNumber()==voucherNumber) {

                                        return
true;

                                }

                        }

                        return false;

                };


                self.addPayment = function() {

                        var paymentTypeOptionsId =
self.paymentTypeOptionsId();

                    var paymentReferenceId =
self.paymentReferenceId();

                    var voucherNumber =
self.voucherNumber();

                    var amountPaid =
self.amountPaid();


                    var pmId =
self.paymentMethodIdentification() + 1;

                    //for Cash

                    if(paymentTypeOptionsId==1) {
```

```
                        //push to array the values
                        self.paymentMethods.push({
                                pmTypeOptionsId:
paymentTypeOptionsId,
                                pmId : pmId,
                                pmPaymentType :
"Cash",
          pmPaymentReferenceId : "",
                                pmAmountPaid :
amountPaid
                        });
                        //recompute total amount
self.computeTotalAmountPaidAndChange();

                    } else if
(paymentTypeOptionsId==2 || paymentTypeOptionsId==3) {
// Credit or Debit
                        //push to array the values
                        self.paymentMethods.push({
                                pmTypeOptionsId:
paymentTypeOptionsId,
                                pmId : pmId,
                                pmPaymentType :
(paymentTypeOptionsId==2)?"Debit":"Credit",
          pmPaymentReferenceId : paymentReferenceId,
                                pmAmountPaid :
amountPaid

                        });
                        //recompute total amount
self.computeTotalAmountPaidAndChange();
                    } else if
(paymentTypeOptionsId==4) { // Returned item(s) Voucher

          if(self.isVoucherExisting(voucherNumber)) {
                                        //error
message: Voucher already added
          $("#returnedItemVoucherError").text("Voucher
already added");
                                    } else {
                                        var url =
'add-voucher';
                                        $.ajax({
          url: url,
          dataType: 'json',
          data: {voucherNumber: voucherNumber, storeId:
self.loggedInUserStoreId()},
          success: function(vouchers) {
          if(vouchers.length > 0) {
                    for (var x in vouchers) {
                            //if voucher is not yet
claimed/used

          if(vouchers[x].status=='unclaimed') {

          self.returnedItemVouchers.push({

          returnedItemVoucherNumber:
ko.observable(vouchers[x].returnItemVoucherId),

          returnedItemVoucherAmount :
ko.observable(vouchers[x].amount.toFixed(2))
```

```
                                        });

                                } //error message: Voucher
is already claimed/used

                                else {


                $("#returnedItemVoucherError").text("Voucher
is already claimed/used");

                                }



                        }

                        //recompute total amount

                    self.computeTotalAmountPaidAndChange();

                } else {

                        //error message: Voucher not found


                $("#returnedItemVoucherError").text("Voucher
not found");

                }

                }

                                                        });

                                        }

                            }

                            //update identification

self.paymentMethodIdentification(pmId);

                                //close the modal

                $('[id=paymentMethodModal]').click();

                        };


                        self.addVoucher = function() {

                                var voucherNumber =
self.voucherNumber();


                if(self.isVoucherExisting(voucherNumber)) {

                                                //error message:
Voucher already added


                $("#returnedItemVoucherError").text("Voucher
already added");

                                } else {

                                        var url = 'add-
voucher';

                                        $.ajax({

                                                url: url,

                                                dataType:
'json',

                                                data:
{voucherNumber: voucherNumber, storeId:
self.loggedInUserStoreId()},

                                                success:
function(vouchers) {


                if(vouchers.length > 0) {


                for (var x in vouchers) {


                        //if voucher is not yet claimed/used


                        if(vouchers[x].status=='unclaimed') {


                self.returnedItemVouchers.push({
```

```
        returnedItemVoucherNumber:
ko.observable(vouchers[x].returnItemVoucherId),

        returnedItemVoucherAmount :
ko.observable(vouchers[x].amount.toFixed(2))
                        });

                        //recompute total amount

        self.computeTotalAmountPaidAndChange();

                        //close the modal

        $('[id=voucherModal]').click();

                } //error message: Voucher is already
claimed/used

                else {

        $("#returnedItemVoucherError").text("Voucher
is already claimed/used");

                }

        }

        } else {

        //error message: Voucher not found

        $("#returnedItemVoucherError").text("Voucher
not found");

        }
                                        }
                                });

                        }

                };

                self.checkoutItems = function() {
                        var transactionItems =
self.transactionItems();
                        var returnedItemVouchers =
self.returnedItemVouchers();
                        var userName =
self.loggedInUser();
                        var storeDetails =
self.storeDetailsString();
                        var paymentMethods =
self.paymentMethods();

                        var url = 'checkout-items';

                        $.ajax({
                                url: url,
                                contentType:
'application/json; charset=utf-8',
                                data:
{transactionItems: ko.toJSON(transactionItems),
userName: userName, storeDetails: storeDetails,

        netAmountDue : self.netAmountDue(),
amountPaid: self.totalAmountPaid(), amountChange:
self.amountChange(),

        returnedItemVouchers:
ko.toJSON(returnedItemVouchers), paymentMethods:
ko.toJSON(paymentMethods),

        storeId: self.loggedInUserStoreId(), userId:
self.loggedInUserId(), totalItems: self.totalItems(),
```

```
        vatableSale: self.vatableSale(), vat:
self.vat(), paymentTypes:
ko.toJSON(self.paymentTypes())},
                                    success:
function(returnMessage) {
                                            if
(returnMessage=='success') {

        //unselect inventory, zero the quantity, and
discount

        for (var x in self.inventory()) {

        self.inventory()[x].itemSelected(false);

        self.inventory()[x].itemQuantity("");

        self.inventory()[x].itemDiscount("");

        }



        //reset transaction items

        self.transactionItems.removeAll();

        //reset payment mthods array

        self.paymentMethods.removeAll();



        //set net amount due

        self.netAmountDue(0);

        //set vatableSale

        self.vatableSale(0);

        //set tax

        self.vat(0);

        //set total items bought

        self.totalItems(0);



        // notify

        $.notify({

        // options

        icon: 'glyphicon glyphicon-ok',

        message: 'Items successfully checked out'

        },{

        // settings

        type: 'success',

        delay: 1000,

        offset: 55,

        });



        //close the modal

        $('[data-dismiss=modal]').click();
```

```
        //2.) view receipt

                                                }
                                        }
                                });


                        };

                        self.computeOtherReceiptDetails =
function() {
                                var netAmountDue = 0;
                                var vatableSale = 0;
                                var vat = 0;
                                var totalItems = 0;
                                for(var x in
self.transactionItems()) {
                                        netAmountDue +=
parseFloat(self.transactionItems()[x].itemPriceAfterDisc
ount());
                                        totalItems +=
parseInt(self.transactionItems()[x].itemQuantity());
                                }
                                vatableSale =
(netAmountDue/100)*88;
                                vat = (netAmountDue/100)*12;
                                //set net amount due

        self.netAmountDue(netAmountDue.toFixed(2));
                                //set vatableSale

        self.vatableSale(vatableSale.toFixed(2));
                                //set tax
                                self.vat(vat.toFixed(2));
                                //set total items bought
                                        self.totalItems(totalItems);

                        };

                        self.addToCart = function() {
                                var
isInTransactionItemsArray = null;

                                for(var x in
self.inventory()) {

        isInTransactionItemsArray = false;

        if(self.inventory()[x].itemSelected() &&
self.inventory()[x].itemQuantity() > 0

        && (self.inventory()[x].itemDiscount() > 0 ||
self.inventory()[x].itemDiscount() == "")) {

                                                var
priceTotal = (self.inventory()[x].itemQuantity() *
self.inventory()[x].itemPrice()).toFixed(2);
                                                var
itemDiscountText = "";
                                                var
itemDiscount = 0;
                                                if
(self.inventory()[x].itemDiscount()>0) {

        itemDiscountText =
self.inventory()[x].itemDiscount() + "%(-"+
(priceTotal*(self.inventory()[x].itemDiscount()/100)).to
Fixed(2) + ")";

        itemDiscount =
self.inventory()[x].itemDiscount();
                                                }
                                                else {

        itemDiscountText = "0%";
```

```
                                        }

                                        //check
if item is already on transactionItems array

                                        for(var y
in self.transactionItems()) {

        //if present, add quantity to transactionItems
array

        if((self.transactionItems()[y].barCode() ==
self.inventory()[x].barCode()) &&
(self.transactionItems()[y].itemDiscount() ==
self.inventory()[x].itemDiscount())) {

        var itemQuantity =
parseInt(self.transactionItems()[y].itemQuantity()) +
parseInt(self.inventory()[x].itemQuantity());



        self.transactionItems()[y].itemQuantity(itemQu
antity);

        priceTotal = (itemQuantity *
self.inventory()[x].itemPrice()).toFixed(2);



        if
(self.transactionItems()[y].itemDiscount()>0) {

                itemDiscountText =
self.transactionItems()[y].itemDiscount() + "%(-"+
(priceTotal*(self.transactionItems()[y].itemDiscount()/1
00)).toFixed(2) + ")";

                itemDiscount =
self.transactionItems()[y].itemDiscount();

        }

        else {

                itemDiscountText = "0%";

        }



        self.transactionItems()[y].itemQuantityXPrice(
self.transactionItems()[y].itemQuantity() + " x " +
self.inventory()[x].itemPrice());

        self.transactionItems()[y].itemPriceTotal(pric
eTotal);

        self.transactionItems()[y].itemDiscountText(it
emDiscountText);

        self.transactionItems()[y].itemLessPrice("-"+
priceTotal*(self.transactionItems()[y].itemDiscount()/10
0));

        self.transactionItems()[y].itemPriceAfterDisco
unt((priceTotal * (1-
(self.transactionItems()[y].itemDiscount()/100))).toFixe
d(2));



        isInTransactionItemsArray = true;

        break;

        }

                                        }

                                        //push to
transaction items if it doesn't exist yet
```

```
                                        if
(!isInTransactionItemsArray) {

          self.transactionItems.push({

          itemSelected: ko.observable(false),

          inventoryId:
ko.observable(self.inventory()[x].inventoryId()),

          itemName:
ko.observable(self.inventory()[x].itemName()),

          barCode:
ko.observable(self.inventory()[x].barCode()),

          itemDescription:
ko.observable(self.inventory()[x].itemDescription()),

          itemPrice:
ko.observable(self.inventory()[x].itemPrice()),

          itemQuantity:
ko.observable(self.inventory()[x].itemQuantity()),

          itemQuantityXPrice :
ko.observable(self.inventory()[x].itemQuantity() + " x "
+ self.inventory()[x].itemPrice()),

          itemPriceTotal : ko.observable(priceTotal),

          itemDiscount: ko.observable(itemDiscount),

          itemDiscountText:
ko.observable(itemDiscountText),

          itemLessPrice: ko.observable("-"+
priceTotal*(self.inventory()[x].itemDiscount()/100)),

          itemPriceAfterDiscount:
ko.observable((priceTotal * (1-
(self.inventory()[x].itemDiscount()/100))).toFixed(2))
          });

                                        }

                                        //compute
other details

          self.computeOtherReceiptDetails();

                                        }

                              }

                    };

                    self.removeFromCart =
function(inventory, event) {

                              // get inventoryId of the
row

                              var inventoryId =
event.currentTarget.id;

                              for(var x in
self.transactionItems()) {

          if(self.transactionItems()[x].inventoryId()==i
nventoryId) {

                                        //remove
from transaction

          self.transactionItems.remove(inventory);

                                        //Do
paging

self.doPaging(self.pageSize(), self.currPage());

                                        break;

                                        }
```

```
                        }

                        //compute other details

        self.computeOtherReceiptDetails();

                };

                self.removeVoucher =
function(voucher, event) {
                        // get inventoryId of the
row
                        var voucherId =
event.currentTarget.id;

                        for(var x in
self.returnedItemVouchers()) {

        if(self.returnedItemVouchers()[x].returnedItem
VoucherNumber()==voucherId) {
                                                //remove
from vouchers array

        self.returnedItemVouchers.remove(voucher);
                                break;
                                 }

                        }

                        //recompute totalAmountPaid

        self.computeTotalAmountPaidAndChange();
                };

                self.removePayment =
function(payment, event) {
                        // get inventoryId of the
row
                        var pmId =
event.currentTarget.id;

                        for(var x in
self.paymentMethods()) {

        if(self.paymentMethods()[x].pmId==pmId) {
                                                //remove
from payment methods array

        self.paymentMethods.remove(payment);
                                        break;
                                         }

                        }
                        //recompute totalAmountPaid

        self.computeTotalAmountPaidAndChange();
                };

                self.searchItems = function() {
                var searchString =
self.searchString();
                self.searchResultsArray.removeAll();

                if(searchString != '') {
                        for(var x in
self.inventoryCopy()) {
                                        if
((self.inventoryCopy()[x].itemName().toLowerCase().index
Of(searchString.toLowerCase()) >= 0)

        ||
(self.inventoryCopy()[x].barCode().toLowerCase().indexOf
(searchString.toLowerCase()) >= 0)
                                                ) {
```

```
        self.searchResultsArray.push(self.inventoryCop
y()[x]);
                                    }
                            }
                    }
                    else {

        self.searchResultsArray(self.inventoryCopy().s
lice());
                    }

                    //Do paging
                   self.doPaging(self.pageSize(),
self.currPage());
            };

            self.cancelEdit = function() {
                  //Clear the values of the form
                  self.inventoryid("");
                            self.barcode("");
                            self.itemdescription("");
                            self.itemprice("");
                            self.itemquantity("");
                            self.itemdiscount("");
                            self.areaid("");
                            self.storeid("");
                            self.itemname("");

                            //for check out
                            self.amountDue(0);
                            self.amountPaid("");
                            self.amountChange(0);
        self.returnedItemVouchers.removeAll();

                            //for voucher
                            self.voucherNumber("");
                };

                self.addToCartThruBarcode = function() {
                      var isInTransactionItemsArray = null;
                      //check if item is already on
transactionItems array
                            for(var x in
self.transactionItems()) {

        isInTransactionItemsArray = false;
                                    //if present, add
quantity to transactionItems array

        if((self.transactionItems()[x].barCode() ==
self.barcode()) &&
(self.transactionItems()[x].itemDiscount() ==
self.itemdiscount())) {
                                          var
itemQuantity = parseInt(self.itemquantity()) +
parseInt(self.transactionItems()[x].itemQuantity());

        self.transactionItems()[x].itemQuantity(itemQu
antity);

        priceTotal = (itemQuantity *
self.transactionItems()[x].itemPrice()).toFixed(2);

                                          if
(self.transactionItems()[x].itemDiscount()>0) {

        itemDiscountText =
self.transactionItems()[x].itemDiscount() + "%(-"+
```

```
(priceTotal*(self.transactionItems()[x].itemDiscount()/1
00)).toFixed(2) + ")";

          itemDiscount =
self.transactionItems()[x].itemDiscount();
                                                  }
                                                  else {

          itemDiscountText = "0%";
                                                  }



          self.transactionItems()[x].itemQuantityXPrice(
self.transactionItems()[x].itemQuantity() + " x " +
self.transactionItems()[x].itemPrice());

          self.transactionItems()[x].itemPriceTotal(pric
eTotal);

          self.transactionItems()[x].itemDiscountText(it
emDiscountText);

          self.transactionItems()[x].itemLessPrice("-"+
priceTotal*(self.transactionItems()[x].itemDiscount()/10
0));

          self.transactionItems()[x].itemPriceAfterDisco
unt((priceTotal * (1-
(self.transactionItems()[x].itemDiscount()/100))).toFixe
d(2));



          isInTransactionItemsArray = true;
                                                  break;
                                        }
                              }


                    var priceTotal = (self.itemquantity()
* self.itemprice()).toFixed(2);
                              var itemDiscountText = "";
                              var itemDiscount = 0;
                              if (self.itemdiscount()>0) {
                                        itemDiscountText =
self.itemdiscount() + "%(-"+
(priceTotal*(self.itemdiscount()/100)).toFixed(2) + ")";
                                        itemDiscount =
self.itemdiscount();
                              }
                              else {
                                        itemDiscountText =
"0%";
                              }


                              //push to transaction items
if it doesn't exist yet
                              if
(!isInTransactionItemsArray) {

          self.transactionItems.push({

          itemSelected: ko.observable(false),

          inventoryId:
ko.observable(self.inventoryid()),
                                                  itemName:
ko.observable(self.itemname()),
                                                  barCode:
ko.observable(self.barcode()),

          itemDescription:
ko.observable(self.itemdescription()),

          itemPrice: ko.observable(self.itemprice()),

          itemQuantity:
ko.observable(self.itemquantity()),

          itemQuantityXPrice :
ko.observable(self.itemquantity() + " x " +
self.itemprice()),
```

```
        itemPriceTotal : ko.observable(priceTotal),

        itemDiscount: ko.observable(itemDiscount),

        itemDiscountText:
ko.observable(itemDiscountText),

        itemLessPrice: ko.observable("-"+
priceTotal*(self.itemdiscount()/100)),

        itemPriceAfterDiscount:
ko.observable((priceTotal * (1-
(self.itemdiscount()/100))).toFixed(2))
                                });
                        }

                        //compute other details

        self.computeOtherReceiptDetails();

                        //clear data on form
                        self.cancelEdit();

                        //close the modal
                        $('[data-
dismiss=modal]').click();

            };

                self.doPaging = function(pageSize,
nextPage) {

                var inventoryArray =
ko.observableArray([]);
                //make a copy of the results
                if(self.searchString() != '') {
        inventoryArray(self.searchResultsArray.slice()
);
                }
                else {
        inventoryArray(self.inventoryCopy().slice());
                }

                //clear items
                self.inventory.removeAll();

                //set current page as next page if
nextPage is defined
                if(nextPage)
                        self.currPage(nextPage);

                //set page size
                self.pageSize(pageSize);

                //calculate number of pages
self.NumberPages(Math.ceil(inventoryArray().length/self.
pageSize()));

                //clear pages array
                self.pagesArray.removeAll();

                //populate pagesArray
                for(var i = 0; i < self.NumberPages();
i++) {
                 self.pagesArray.push({
                        pageNumber:
ko.observable((i+1))
                });
```

```
                }


                //set max number of pages

self.maxNumberPages(self.NumberPages());


                //if current page is greater than max
number of pages, set currPage = maxNumberPages
                if(self.currPage() >
self.maxNumberPages())
                  self.currPage(self.maxNumberPages());


                //if maxNumberPages is less than 1,
set currPage to 1
                if(self.maxNumberPages() < 1)
                  self.currPage(1);


                var startIndex = (self.currPage()-
1)*self.pageSize();
                  for(var i = startIndex; i <
(self.pageSize() + startIndex); i++) {
                            if(inventoryArray()[i]) {

         self.inventory.push(inventoryArray()[i]);
                            }
                            else {
                                     break;
                            }


                  }
             };
         }
         ko.applyBindings(new POSViewModel());
});
```

inventory.js

```
/**
 * This is the Jquery file for the inventory.jsp page
 */
$(function() {
          /**
           * Start of knockout.js
           *
           */
          function Inventory(inventory) {
              this.inventoryId =
ko.observable(inventory.inventoryId);
              this.inventoryItemId =
ko.observable(inventory.itemId);
              this.itemname =
ko.observable(inventory.itemName);
              this.itemdescription =
ko.observable(inventory.itemDescription);
              this.barCode =
ko.observable(inventory.barCode);
              this.itemCount =
ko.observable(inventory.itemQuantity);
              this.inventoryStoreId =
ko.observable(inventory.storeId);
              this.inventoryStoreAreaId =
ko.observable(inventory.areaId);
              this.delInventoryTitle =
ko.observable("Remove inventory " +
inventory.inventoryId);
              this.editInventoryTitle =
ko.observable("Edit inventory " +
inventory.inventoryId);
          }


          function SearchInventory(inventory) {
                    this.searchStoreBranch =
ko.observable(inventory.store.branchName);
                    this.searchItemName =
ko.observable(inventory.item.itemname);
```

```
                this.searchItemDescription =
ko.observable(inventory.item.itemdesc);

                this.searchItemBarCode =
ko.observable(inventory.item.barCode);

                this.searchItemQuantity =
ko.observable(inventory.itemCount);

        }



        function Item(item) {

            this.itemId = ko.observable(item.itemId);

            this.itemName =
ko.observable(item.itemname);

        }



        function Area(area) {

            this.areaId = ko.observable(area.areaId);

            this.areaName =
ko.observable(area.areaName);

        }



        function Store(store) {

            this.storeId =
ko.observable(store.storeId);

            this.area_id =
ko.observable(store.area.areaId);

            this.branchName =
ko.observable(store.branchName);

        }



        // Overall viewmodel for this screen, along
with initial state

        function InventoryViewModel() {

                var self = this;



                //list of inventory

                self.inventory =
ko.observableArray([]);

                self.inventoryCopy =
ko.observableArray([]);

                self.searchInventoryArray =
ko.observableArray([]);



                //list of items

                self.items = ko.observableArray([]);

                self.itemsCopy =
ko.observableArray([]);



                //list of areas

                self.areas = ko.observableArray([]);



                //list of store branches

                self.stores = ko.observableArray([]);

                self.stores2 =
ko.observableArray([]); //for search other stores

                self.storesCopy =
ko.observableArray([]);



                //used in select (viewing inventory)

                self.areaid1 = ko.observable();

                self.storeid1 = ko.observable();



                //used in select (search inventory
from other stores)

                self.searchItem = ko.observable("");

                self.searchBarcode = ko.observable();

                self.areaid2 = ko.observable();

                self.storeid2 = ko.observable();



                //used in current store inventory
displayed
```

```
                    self.currentResultStoreId =
ko.observable();



                    //used in modal forms

                    self.formInventoryId =
ko.observable(0);

                    self.formItemName =
ko.observable("");

                    self.formItemQuantity =
ko.observable("");

                    self.inventory_itemId =
ko.observable(0);



                    //used in cookies

                    self.loggedInUser =
ko.observable(Cookies.get('username'));

                    self.loggedInUsertype =
ko.observable(Cookies.get('usertype'));

                    self.loggedInUserStoreId =
ko.observable(Cookies.get('userStoreId'));



                    //for paging

               self.pageSize = ko.observable(10);

               self.NumberPages = ko.observable(1);

               self.currPage = ko.observable(1);

               self.pagesArray = ko.observableArray([]);

               self.maxNumberPages = ko.observable(1);



               //These are used in search

                    self.searchString =
ko.observable("");

                    self.searchResultsArray =
ko.observableArray([]);
```

```
                    //on first load, decide what button
to show

                    self.displayButtons = function() {

          if(self.loggedInUsertype()!=1) {

          $("#addInventoryButton").css('display',
'none');

          $("#searchFromOther").css('display',
'inline');

          $(".searchInventory").css('display', 'none');
                              } else {

          $("#searchFromOther").css('display', 'none');

          $(".searchInventory").css('display',
'inline');

          if(self.inventoryCopy().length>0) {

          $("#addInventoryButton").css('display',
'inline');
                                   }
                              }
                    };



                    //call displayButtons method

                    self.displayButtons();



                    self.getInventory = function() {
                              var storeId = null;

          if(self.loggedInUsertype()!=1) {
                                   storeId =
self.loggedInUserStoreId();
                              } else {
```

```
                                        storeId =
self.storeid1();
                                }
                                var url = 'get-inventory';
                                $.ajax({
                                        url: url,
                                        dataType: 'json',
                                        data: {storeId:
storeId},
                                        success:
function(allData) {
                                                var
mappedItems = $.map(allData, function(inventory) {
return new Inventory(inventory); });


                                //make a copy of the
inventory

self.inventoryCopy(mappedItems);
                                //save in inventory

self.inventory(self.inventoryCopy().slice());


                                //display button


self.displayButtons();



                                //set current result
store Id

self.currentResultStoreId(storeId);
                                //get the items
                                self.getItems();
                                //Do paging

self.doPaging(self.pageSize());


                                        }
                                });
                        };


                        //call getInventory method for non-
admin
                        if(self.loggedInUsertype()!=1) {
                                self.getInventory();


                        }


                        //get the list of areas on first load
from DB
                $.getJSON("get-areas", function(allData) {
                    var mappedItems = $.map(allData,
function(area) { return new Area(area); });
                    //fill areas array
                    self.areas(mappedItems);
                });
                        self.getItems = function() {
                                //clear items array first
                                self.items.removeAll();
                                self.itemsCopy.removeAll();
                                //get the list of items on
first load from DB
                            $.getJSON("get-items",
function(allData) {
                                var mappedItems =
$.map(allData, function(item) { return new Item(item);
});
                                //fill items array
                                self.items(mappedItems);
                                //have a copy
```

```
self.itemsCopy(self.items().slice());

                        //update inventory items to
be added

                        self.updateItems();

                    });

                };

                self.getStores = function() {

                         //get the list of stores
from DB

                    $.getJSON("get-stores",
function(allData) {

                        var mappedItems =
$.map(allData, function(store) { return new
Store(store); });

                        //fill stores array

                        self.stores(mappedItems);

                        //have a copy


self.storesCopy(self.stores.slice());

                        //fill stores array for
searching from other stores


self.stores2(self.stores.slice());



                        //set the initial value of
area, if user type is not Admin

                        if(self.loggedInUsertype() !=
1) {

                         for(var x in
self.storesCopy()) {


         if(self.storesCopy()[x].storeId() ==
self.loggedInUserStoreId()) {


         self.areaid1(self.storesCopy()[x].area_id());


         break;

         }

                                                 }

                        }

                        //set store id in dropdown if
user not admin


if(self.loggedInUsertype()!=1) {


         self.storeid1(self.loggedInUserStoreId());

                        }



                    });

                };


                //get stores

                self.getStores();


                self.updateItems = function() {

                         for(var i in
self.itemsCopy()) {

                for(var j in self.inventory()) {


         if(self.itemsCopy()[i].itemId() ==
self.inventory()[j].inventoryItemId()) {


         self.items.remove(self.itemsCopy()[i]);

                         }

                }

                         }

                };

            //display add inventory modal

            self.displayAddInventory = function() {


                //empty error
```

```
        $("#editInventoryError").html("");
                        self.inventory_itemId(0);
                        //show the Edit modal

        $('#editInventoryModal').modal('show');
            };

            //display search inventory from other
stores modal
            self.displaySearchFromOther = function() {
                //Clear values of search inventory
form
                self.searchItem("");
                        self.searchBarcode(null);
                        self.areaid2(null);
                        self.storeid2(null);

                //empty error

        $("#searchFromOtherError").html("");
                        self.inventory_itemId(0);
                        //show the Edit modal

        $('#searchFromOtherModal').modal('show');
            };

            self.displayInventoryQuantity = function()
{
                //map the values to modal form

        self.formInventoryId(this.inventoryId());
                self.formItemName(this.itemname());

        self.formItemQuantity(""+this.itemCount());
                        //show the modal

        $('#editInventoryQuantityModal').modal('show')
;
            };

            self.cancelEdit = function() {
                //Clear the values of the form
                self.formItemName("");
                self.formItemQuantity("");
            };

            self.cancelSearch = function() {
                //Clear values of search inventory
form
                self.searchItem("");
                        self.searchBarcode(null);
                        self.areaid2(null);
                        self.storeid2(null);
            };

            self.closeViewResult = function() {
                //clear search inventory result array

        self.searchInventoryArray.removeAll();
            };

            self.searchFromOther = function() {

                //Clear error message

                var searchItem = self.searchItem();
                        var searchBarcode =
self.searchBarcode();
```

```
                        var areaid2 =
self.areaid2();
                        var storeid2 =
self.storeid2();


                var url = 'search-inventory';
                        $.ajax({
                                type: 'GET',
                                dataType: 'json',
                                url: url,
                                data: {areaid2:
areaid2, storeid2: storeid2, searchItem: searchItem,
searchBarcode: searchBarcode},
                                success:
function(allData) {
                                        var
mappedItems = $.map(allData, function(inventory) {
return new SearchInventory(inventory); });
                                        //fill
search inventory array

self.searchInventoryArray(mappedItems);



if(self.searchInventoryArray().length==0) {

        errorMessage = "No inventory found.";

        //Display error message

        $("#searchFromOtherError").html(errorMessage);

        return false;
                                } else {
                                        //Clear
error message

        errorMessage = "";

        //Display error message

        $("#searchFromOtherError").html(errorMessage);

        //return false;
                                        //Display
inventory

        $('#viewInventoryModal').modal('show');
                                    }

                                },
                                error: function()
{

        errorMessage = "Something is wrong. Please try
again later.";
                                        //Display
error message

        $("#searchFromOtherError").html(errorMessage);
                                        return
false;
                                },
                        });
            };


            self.editInventory = function() {
                var inventoryId =
this.formInventoryId();
                var itemCount =
this.formItemQuantity();


                var url = 'update-inventory';
                        $.ajax({
                                type: 'GET',
                                url: url,
```

```
                                    data:
{inventoryId: inventoryId, itemCount: itemCount},
                                    success:
function() {

        //loop through the inventory and update the
value

        for(var i = 0; i<self.inventory().length; i++)
{

        if (self.inventory()[i].inventoryId() ==
inventoryId) {


        self.inventory()[i].itemCount(itemCount);


break;
                                                }

        }
                                            $.notify({

        // options

        icon: 'glyphicon glyphicon-ok',

        message: 'Inventory item successfully updated'

        },{

        // settings

        type: 'success',

        delay: 1000,

        offset: 55,

        });
                                                $('[data-
dismiss=modal]').click();


                                    },
                                    error: function()
{

        errorMessage = "Cannot edit item. Please try
again later.";
                                                //Display
error message

        $("#editItemError").html(errorMessage);
                                                return
false;
                                    },
                            });
                };


                self.addInventory = function() {
                        var areaId = this.areaid1();
                        var storeId = this.storeid1();
                        var itemId = this.inventory_itemId();


                        var url = 'add-inventory';
                                $.ajax({
                                        url: url,
                                        dataType: 'json',
                                        data: {storeId:
storeId, itemId: itemId},
                                        success:
function(inventory) {
                                                //close
the modal
                                                $('[data-
dismiss=modal]').click();
```

```
                                        //remove
from items
                                        for(var i
in self.items()) {

        if(self.items()[i].itemId() == itemId) {

        self.items.remove(self.items()[i]);

                                        break;
                                    }

                                        }

                                        var
itemName = "";

                                        for(var i
in self.itemsCopy()) {

        if(self.itemsCopy()[i].itemId() == itemId) {

                                        itemName
= self.itemsCopy()[i].itemName();

                                        break;
                                    }

                                        }


        if(self.searchString() != '') {

        self.searchResultsArray.push({

        inventoryId :
ko.observable(inventory.inventoryId),

        inventoryItemId :
ko.observable(inventory.item.itemId),

        itemname: ko.observable(itemName),

        itemdescription :
ko.observable(inventory.item.itemdesc),

        barCode :
ko.observable(inventory.item.barCode),

        itemCount :
ko.observable(inventory.itemCount),

        inventoryStoreId:
ko.observable(inventory.store.storeId),

        inventoryStoreAreaId: ko.observable(areaId),

        delInventoryTitle : ko.observable("Remove
inventory " + inventory.inventoryId),

        editInventoryTitle : ko.observable("Edit
inventory " + inventory.inventoryId)

        });
                                    }
                                                else {

        self.searchResultsArray.push({

        inventoryId :
ko.observable(inventory.inventoryId),

        inventoryItemId :
ko.observable(inventory.item.itemId),

        itemname: ko.observable(itemName),

        itemdescription :
ko.observable(inventory.item.itemdesc),

        barCode :
ko.observable(inventory.item.barCode),

        itemCount :
ko.observable(inventory.itemCount),

        inventoryStoreId:
ko.observable(inventory.store.storeId),

        inventoryStoreAreaId: ko.observable(areaId),
```

```
          delInventoryTitle : ko.observable("Remove
inventory " + inventory.inventoryId),

          editInventoryTitle : ko.observable("Edit
inventory " + inventory.inventoryId)

          });
                                                  }
                                                  //add
also in the copy

          self.inventoryCopy.push({

          inventoryId :
ko.observable(inventory.inventoryId),

          inventoryItemId :
ko.observable(inventory.item.itemId),

          itemname: ko.observable(itemName),

          itemdescription :
ko.observable(inventory.item.itemdesc),

          barCode :
ko.observable(inventory.item.barCode),

          itemCount :
ko.observable(inventory.itemCount),

          inventoryStoreId:
ko.observable(inventory.store.storeId),

          inventoryStoreAreaId: ko.observable(areaId),

          delInventoryTitle : ko.observable("Remove
inventory " + inventory.inventoryId),

          editInventoryTitle : ko.observable("Edit
inventory " + inventory.inventoryId)
                                                  });
                                                                  //Do
paging

self.doPaging(self.pageSize(), self.currPage());
                                                  //notify
that adding is successful

          $.notify({

          // options

          icon: 'glyphicon glyphicon-ok',

          message: 'Inventory item successfully added'
                                                  },{

          // settings

          type: 'success',

          delay: 1000,

          offset: 55,
                                                  });

                                          }
                                  });
              };
              self.removeInventoryItem =
function(inventory, event) {
                    //get inventoryId of the row
                                  var inventoryId =
event.currentTarget.id;
                                  bootbox.confirm({
                                          message: "You are
about to delete inventory item " + inventoryId + ".\nDo
you want to proceed?",
```

```
                                    closeButton:
false,
                                    size: "small",
                                callback:
function(result){
                                    if(result) {

        var url = 'delete-' + inventoryId + '-
inventory';

        $.ajax({

        url: url,

        success: function() {

                //add to items

                var itemId = 0;

                var itemName = "";

                for(var i in self.inventory()) {

        if(self.inventory()[i].inventoryId() ==
inventoryId) {

                itemId =
self.inventory()[i].inventoryItemId();

                itemName =
self.inventory()[i].itemname();

                break;

        }

                }

                self.items.push({
                        itemId :
ko.observable(itemId),
                        itemName :
ko.observable(itemName)
                });


                //remove the element from the table

                if(self.searchString() != '') {

                        //remove from
searchResultsArray

                        for(var x in
self.searchResultsArray()) {

        if(self.searchResultsArray()[x].inventoryId()
== inventoryId) {

        self.searchResultsArray.remove(self.searchResu
ltsArray()[x]);

                                        break;

                                }

                        }

        }

        else {

                self.inventory.remove(inventory);

        }
```

```
                    //remove also from the copy
                    for(var x in self.itemsCopy()) {

          if(self.inventoryCopy()[x].inventoryId() ==
inventoryId) {

          self.inventoryCopy.remove(self.inventoryCopy()
[x]);
                                        break;
                              }
                    }

                    //Do paging
self.doPaging(self.pageSize(), self.currPage());

                    $.notify({
                              // options
                              icon: 'glyphicon glyphicon-
ok',
                              message: 'Inventory item
successfully deleted'
                    },{
                              // settings
                              type: 'success',
                              delay: 1000,
                              offset: 55,
                    });
          },
          error: function(jqXHR, textStatus,
errorThrown) {
              alert("error:" + textStatus + "
exception:" + errorThrown);
          },
          });
                                                  }
                                        }
                              });
              };

              //listen if new areaid1 is selected in
selection
self.areaid1.subscribe(function(newAreaIdValue) {
                    //clear inventory and stores array
                    self.inventory.removeAll();

                    if(newAreaIdValue) {
                              //update inventory
                              for(var x in
self.inventoryCopy()) {
```

```
                                    if
((self.inventoryCopy()[x].inventoryStoreAreaId() ==
newAreaIdValue)) {

          self.inventory.push(self.inventoryCopy()[x]);
                                    }
                          }
                          //clear stores array
                          self.stores.removeAll();
                          //update the selection in
store branches
                          for(var x in
self.storesCopy()) {
                                    if
((self.storesCopy()[x].area_id() == newAreaIdValue)) {

          self.stores.push(self.storesCopy()[x]);
                                    }
                          }

                }
                else {
                          if(self.storeid1()) {
                          for(var x in
self.inventoryCopy()) {
                                              if
((self.inventoryCopy()[x].inventoryStoreId() ==
self.storeid1())) {

          self.inventory.push(self.inventoryCopy()[x]);
                                              }
                                    }
                          }
                          else {

          self.inventory(self.inventoryCopy().slice());
                          }
                          //return copy of stores

          self.stores(self.storesCopy().slice());

                          }
                }
          });

          //listen if new areaid2 is selected in
selection

self.areaid2.subscribe(function(newAreaIdValue) {

                if(newAreaIdValue) {
                          //clear stores array
                          self.stores2.removeAll();
                          //update the selection in
store branches
                          for(var x in
self.storesCopy()) {
                                    if
((self.storesCopy()[x].area_id() == newAreaIdValue)) {

          self.stores2.push(self.storesCopy()[x]);
                                    }
                          }

                }
                else {
                          //return copy of stores

          self.stores2(self.storesCopy().slice());

                }
          });
```

```
            //listen if new storeid1 is selected in
selection


self.storeid1.subscribe(function(newStoreIdValue) {
                  //alert('subscribe store');
                  self.inventory.removeAll();
                  //update the selection in store
branches
                  if(newStoreIdValue) {
                           for(var x in
self.inventoryCopy()) {
                                    if
((self.inventoryCopy()[x].inventoryStoreId() ==
newStoreIdValue)) {

         self.inventory.push(self.inventoryCopy()[x]);
                                    }
                           }
                  }
                  else {
                           if(self.areaid1()) {
                                    for(var x in
self.inventoryCopy()) {
                                             if
((self.inventoryCopy()[x].inventoryStoreAreaId() ==
self.areaid1())) {

         self.inventory.push(self.inventoryCopy()[x]);
                                             }
                                    }
                           }
                           else {

         self.inventory(self.inventoryCopy().slice());
                           }
                  }
            });

            self.searchInventory = function() {
                  var searchString =
self.searchString();
                  self.searchResultsArray.removeAll();

                  if(searchString != '') {
                           for(var x in
self.inventoryCopy()) {
                                    if
((self.inventoryCopy()[x].itemname().toLowerCase().index
Of(searchString.toLowerCase()) >= 0)) {

         self.searchResultsArray.push(self.inventoryCop
y()[x]);
                                    }
                           }
                  }
                  else {

         self.searchResultsArray(self.inventoryCopy().s
lice());
                  }

                  //Do paging
                 self.doPaging(self.pageSize(),
self.currPage());
            };
            self.doPaging = function(pageSize,
nextPage) {

                  var inventoryArray =
ko.observableArray([]);
                  //make a copy of the results
                  if(self.searchString() != '') {

         inventoryArray(self.searchResultsArray.slice()
);
```

```
                }
                else {
        inventoryArray(self.inventoryCopy().slice());
                }

                //clear inventory
                self.inventory.removeAll();

                //set current page as next page if
nextPage is defined
                if(nextPage)
                        self.currPage(nextPage);

                //set page size
                self.pageSize(pageSize);

                //calculate number of pages
self.NumberPages(Math.ceil(inventoryArray().length/self.
pageSize()));

                //clear pages array
                self.pagesArray.removeAll();

                //populate pagesArray
                for(var i = 0; i < self.NumberPages();
i++) {
                 self.pagesArray.push({
                        pageNumber:
ko.observable((i+1))
                 });
                }

                //set max number of pages
self.maxNumberPages(self.NumberPages());

                //if current page is greater than max
number of pages, set currPage = maxNumberPages
                if(self.currPage() >
self.maxNumberPages())
                  self.currPage(self.maxNumberPages());

                //if maxNumberPages is less than 1,
set currPage to 1
                if(self.maxNumberPages() < 1)
                  self.currPage(1);

                var startIndex = (self.currPage()-
1)*self.pageSize();
                  for(var i = startIndex; i <
(self.pageSize() + startIndex); i++) {
                        if(inventoryArray()[i]) {
        self.inventory.push(inventoryArray()[i]);
                        }
                        else {
                                break;
                        }

                  }
            };

        }

        ko.applyBindings(new InventoryViewModel());

});
```

items.js

```
/**
 * This is the Jquery file for the items.jsp page
 */
if(Cookies.get('usertype')!=1) {
	//redirect to home page
	window.location.replace('home');
}
$(function() {
	/**
	 * Start of knockout.js
	 *
	 */

	function Item(item) {
	    this.itemId = ko.observable(item.itemId);
	    this.barCode = ko.observable(item.barCode);
	    this.itemname = ko.observable(item.itemname);
	    this.itemdesc = ko.observable(item.itemdesc);
	    this.price = ko.observable(item.price);
	    this.delItemTitle = ko.observable("Delete item " + item.itemId);
	    this.editItemTitle = ko.observable("Edit item " + item.itemId);
	}
	// Overall viewmodel for this screen, along with initial state
	function ItemsViewModel() {
		var self = this;

		//used in cookies
		self.loggedInUser = ko.observable(Cookies.get('username'));
		self.loggedInUsertype = ko.observable(Cookies.get('usertype'));
		self.loggedInUserStoreId = ko.observable(Cookies.get('userStoreId'));

		//These are the fields used in the add/edit items form
		self.itemid = ko.observable(0);
		self.itemName = ko.observable("");
		self.barcode = ko.observable("");
		self.itemDesc = ko.observable("");
		self.itemPrice = ko.observable("");

		//These are used in search
		self.searchString = ko.observable("");
		self.searchResultsArray = ko.observableArray([]);

		//used in the list
	    self.items = ko.observableArray([]);

	    //copy of self.items, this one will be used in searching
	    self.itemsCopy = ko.observableArray([]);

	    //for paging
	    self.pageSize = ko.observable(10);
	    self.NumberPages = ko.observable(1);
	    self.currPage = ko.observable(1);
	    self.pagesArray = ko.observableArray([]);
	    self.maxNumberPages = ko.observable(1);
```

```
            // get the list of items on first load
from DB
            $.getJSON("get-items", function(allData) {
                var mappedItems = $.map(allData,
function(item) { return new Item(item); });

                //make a copy of the items
                self.itemsCopy(mappedItems);

                //Do paging on first load
                self.doPaging(self.pageSize());

            });

            self.displayAddItem = function() {
                 //change the modal title and button
text
                 $(".modal-title").text("Add new
item");
         $("#editItemSumbitButton").text("Add");

                            //enable item name text
input
         $("#itemNameInput").prop('disabled', false);

                            //Clear the values inside
the form
                            self.itemid("");
                            self.barcode("");
                            self.itemName("");
                            self.itemDesc("");
                            self.itemPrice("");

                            //show the Edit modal
$('#editItemModal').modal('show');
                };

                self.displayItem = function() {
                     //change the modal title and button
text
                     $(".modal-title").text("Edit item");
$("#editItemSumbitButton").text("Update");

                            //disable item name text
input
$("#itemNameInput").prop('disabled', true);

                     //map the values to modal form
                     self.itemid(this.itemId());
self.itemName(this.itemname());
self.barcode(this.barCode());
self.itemDesc(this.itemdesc());
self.itemPrice(this.price());

                            //show the modal
$('#editItemModal').modal('show');
                };

                self.cancelEdit = function() {
                     //Clear the values of the form
```

```
                self.itemid(0);
                        self.itemName("");
                        self.barcode("");
                        self.itemDesc("");
                        self.itemPrice("");
        };

        self.editItem = function() {
                var itemId = this.itemid();
                var itemName = this.itemName();
                var barCode = this.barcode();
                var itemDesc = this.itemDesc();
                var itemPrice = this.itemPrice();

                //Add new item
                if(itemId==0) {
                        var url = 'add-item';
                                $.ajax({
                                        url: url,
                                        dataType:
'json',
                                        data:
{itemName: itemName, barCode: barCode, itemDesc:
itemDesc, itemPrice: itemPrice},
                                        success:
function(item) {

        //close the modal
        $('[data-dismiss=modal]').click();

        if(self.searchString() != '') {
        self.searchResultsArray.push({
                        itemId : ko.observable(item.itemId),
                        itemname :
ko.observable(item.itemname),
                        barCode :
ko.observable(item.barCode),
                itemdesc : ko.observable(item.itemdesc),
                price: ko.observable(item.price),
                delItemTitle : ko.observable("Delete item
" + item.itemId),
                editItemTitle : ko.observable("Edit item "
+ item.itemId)
        });
                                                }
        else {
        self.searchResultsArray.push({
                        itemId : ko.observable(item.itemId),
                        itemname :
ko.observable(item.itemname),
                        barCode :
ko.observable(item.barCode),
                itemdesc : ko.observable(item.itemdesc),
                price: ko.observable(item.price),
                delItemTitle : ko.observable("Delete item
" + item.itemId),
```

```
            editItemTitle : ko.observable("Edit item "
+ item.itemId)

          });

          }

          //add also in the copy

          self.itemsCopy.push({

          itemId : ko.observable(item.itemId),

          itemname : ko.observable(item.itemname),

          barCode : ko.observable(item.barCode),

itemdesc : ko.observable(item.itemdesc),

price: ko.observable(item.price),

delItemTitle : ko.observable("Delete item " +
item.itemId),

editItemTitle : ko.observable("Edit item " +
item.itemId)

          });



          //Do paging

self.doPaging(self.pageSize(), self.currPage());



          //notify that adding is successful

          $.notify({

          // options

          icon: 'glyphicon glyphicon-ok',

          message: 'Item successfully added'

          },{

          // settings

          type: 'success',

          delay: 1000,

          offset: 55,

          });
                                                    },
                                                    error:
function(jqXHR, textStatus, errorThrown) {

          errorMessage = "Cannot add item. Please check
if bar code " +

          "or item name already exist. Or try again
later.";

          //Display error message

          $("#editItemError").html(errorMessage);
                                                    }
                                        });
                    }
                    //Edit the item
                    else {
                              var url = 'update-item';
                                        $.ajax({
```

```
                                        type:
'GET',
                                        url: url,
                                        data:
{itemId: itemId, itemName: itemName, barCode: barCode,
itemDesc: itemDesc, itemPrice: itemPrice},
                                        success:
function() {

        //loop through the items and update the value

        for(var i = 0; i<self.items().length; i++) {

                if (self.items()[i].itemId() ==
itemId) {


        self.items()[i].itemname(itemName);


        self.items()[i].barCode(barCode);


        self.items()[i].itemdesc(itemDesc);


        self.items()[i].price(itemPrice);

break;

}

        }

        //loop through the items copy and update the
value

        for(var i = 0; i<self.itemsCopy().length; i++)
{

                if (self.itemsCopy()[i].itemId() ==
itemId) {


        self.itemsCopy()[i].itemname(itemName);


        self.items()[i].barCode(barCode);


        self.itemsCopy()[i].itemdesc(itemDesc);


        self.items()[i].price(itemPrice);

break;

}

        }

        //Do paging

self.doPaging(self.pageSize(), self.currPage());


$.notify({

                // options

                icon: 'glyphicon glyphicon-ok',

                message: 'Item successfully updated'

        },{

                // settings

                type: 'success',

                delay: 1000,

                offset: 55,
```

```
        });

        $('[data-dismiss=modal]').click();

                                            },
                                            error:
function() {

        errorMessage = "Cannot add item. Please check
if bar code " +

        "or item name already exist. Or try again
later.";

        //Display error message

        $("#editItemError").html(errorMessage);

        return false;

                                            },
                                    });
                            }
                            return true;
            };

            self.removeItem = function(item, event) {

                // get itemId of the row
                            var itemId =
event.currentTarget.id;

                            bootbox.confirm({
                                    message: "You are
about to delete item " + itemId + ".\nDo you want to
proceed?",
                                    closeButton:
false,
                                    size: "small",
                                    callback:
function(result){
                                    if(result) {

        var url = 'delete-' + itemId + '-item';

        $.ajax({

        url: url,

        success: function() {

                //remove the element from the table

                if(self.searchString() != '') {

                        //remove from
searchResultsArray

                        for(var x in
self.searchResultsArray()) {

        if(self.searchResultsArray()[x].itemId() ==
itemId) {

        self.searchResultsArray.remove(self.searchResu
ltsArray()[x]);

                                    break;
                                    }
                        }

        }

        else {

                self.items.remove(item);
```

```
        }


                //remove also from the copy

                for(var x in self.itemsCopy()) {

        if(self.itemsCopy()[x].itemId() == itemId) {

        self.itemsCopy.remove(self.itemsCopy()[x]);

                                break;

                        }

                }


                //Do paging

self.doPaging(self.pageSize(), self.currPage());


                $.notify({

                        // options

                        icon: 'glyphicon glyphicon-
ok',

                        message: 'Item successfully
deleted'

                },{

                        // settings

                        type: 'success',

                        delay: 1000,

                        offset: 55,

                });

        },

        error: function(jqXHR, textStatus,
errorThrown) {

            alert("error:" + textStatus + "
exception:" + errorThrown);

        },

        });
                                                }
                            }
                        });
            };


            self.searchItems = function() {

                var searchString =
self.searchString();

                self.searchResultsArray.removeAll();


                if(searchString != '') {

                        for(var x in
self.itemsCopy()) {

                                        if
((self.itemsCopy()[x].itemname().toLowerCase().indexOf(s
earchString.toLowerCase()) >= 0)

        ||
```

```
(self.itemsCopy()[x].itemdesc().toLowerCase().indexOf(se
archString.toLowerCase()) >= 0)

        ||
(self.itemsCopy()[x].barCode().toLowerCase().indexOf(sea
rchString.toLowerCase()) >= 0)

                                            ) {

        self.searchResultsArray.push(self.itemsCopy()[
x]);

                                    }

                            }

                }

                else {

        self.searchResultsArray(self.itemsCopy().slice
());

                }


                //Do paging

               self.doPaging(self.pageSize(),
self.currPage());

            };


            self.doPaging = function(pageSize,
nextPage) {


                var itemsArray =
ko.observableArray([]);

                //make a copy of the results

                if(self.searchString() != '') {

        itemsArray(self.searchResultsArray.slice());

                }

                else {

        itemsArray(self.itemsCopy().slice());

                }


                //clear items

                self.items.removeAll();


                //set current page as next page if
nextPage is defined

                if(nextPage)

                        self.currPage(nextPage);


                //set page size

                self.pageSize(pageSize);


                //calculate number of pages

self.NumberPages(Math.ceil(itemsArray().length/self.page
Size()));


                //clear pages array

                self.pagesArray.removeAll();


                //populate pagesArray

                for(var i = 0; i < self.NumberPages();
i++) {

                 self.pagesArray.push({

                        pageNumber:
ko.observable((i+1))

                 });

                }


                //set max number of pages

self.maxNumberPages(self.NumberPages());
```

```
                //if current page is greater than max
number of pages, set currPage = maxNumberPages
                if(self.currPage() >
self.maxNumberPages())
                 self.currPage(self.maxNumberPages());

                //if maxNumberPages is less than 1,
set currPage to 1
                if(self.maxNumberPages() < 1)
                 self.currPage(1);

                var startIndex = (self.currPage()-
1)*self.pageSize();
                 for(var i = startIndex; i <
(self.pageSize() + startIndex); i++) {
                        if(itemsArray()[i]) {
        self.items.push(itemsArray()[i]);
                        }
                        else {
                                break;
                        }
                 }
            };

        }
        ko.applyBindings(new ItemsViewModel());
});
```

login.js

```
/**
 * This is the Jquery file for the login.jsp page
 */
//Check if user is logged in
if(Cookies.get('username') && Cookies.get('username') !=
'') {
                        //redirect to home page if
someone is logged in

        window.location.replace('home');
}
$(function() {
        /**
         * Start of knockout.js
         *
         */

        function LoginViewModel() {

                var self = this;

                //for the form
                self.loginUserName =
ko.observable("");
                self.loginUserPassword =
ko.observable("");

                self.doLogin = function() {
                        var userName =
this.loginUserName();
                        var userPassword =
this.loginUserPassword();

                        var url = 'login-user';
                        $.ajax({
                                type: 'GET',
                                url: url,
                                dataType: 'json',
```

```
                                    data: {userName:
userName, userPassword: userPassword},
                                    success:
function(user) {

          if(user.active != 0) {

          //create cookie session

          Cookies.set('username', user.userName);

          Cookies.set('usertype',
user.usertype.usertypeId);

          Cookies.set('userStoreId',
user.store.storeId);

          Cookies.set('userAreaId',
user.store.area.areaId);

          Cookies.set('userId', user.userID);



          //redirect to home page

          window.location.replace('home');

                                              }
                                              else{

          errorMessage = "User is not allowed to use the
system. Please contact administrator.";

          //Display error message

          $("#loginError").html(errorMessage);

                                              }
                                    },
                                    error: function()
{
          errorMessage = "Username and password do not
match.";
                                                        //Display
error message

          $("#loginError").html(errorMessage);

                                                        return
false;
                                         },
                                    });
                              };


          };

          ko.applyBindings(new LoginViewModel());
});
```

reports.js

```
/**
 * This is the Jquery file for the pos.jsp page
 */
if(Cookies.get('usertype')!=1) {
          //redirect to home page
          window.location.replace('home');
}
$(function() {
          /**
           * Start of knockout.js
           *
           */
          function Area(area) {
              this.areaId = ko.observable(area.areaId);
              this.areaName =
ko.observable(area.areaName);
          }
```

```
        function Store(store) {
            this.storeId =
ko.observable(store.storeId);
            this.area =
ko.observable(store.area.areaName);
            this.area_id =
ko.observable(store.area.areaId);
            this.branchName =
ko.observable(store.branchName);
            this.address =
ko.observable(store.address);
            this.coordinates =
ko.observable(store.coordinates);
        }

        function Payment(payment) {
                this.pPaymentType =
ko.observable(payment.paymentType);
                this.pAmount =
ko.observable(payment.amount);
        }

        function Receipt(receipt) {
                this.amountDue =
ko.observable(receipt.amountDue);
                this.timestamp =
ko.observable(receipt.timestamp);
        }

        function ReportViewModel() {
                var self = this;

                //used in the list
                self.stores = ko.observableArray([]);
                self.storesCopy =
ko.observableArray([]);
                self.areas = ko.observableArray([]);

                //used in selects
                self.areaid = ko.observableArray();
                self.storeid = ko.observableArray();

                //used in charts
                self.transactionCount =
ko.observable(0);
                self.totalSales = ko.observable(0);
                self.payments =
ko.observableArray([]);
                self.paymentSummary =
ko.observableArray([]);
                self.receipts =
ko.observableArray([]);

                //global variables for use in high
charts
                self.currentAreaId =
ko.observable(0);
                self.currentStoreId =
ko.observable(0);

                //used in cookies
                self.loggedInUser =
ko.observable(Cookies.get('username'));
                self.loggedInUsertype =
ko.observable(Cookies.get('usertype'));
                self.loggedInUserStoreId =
ko.observable(Cookies.get('userStoreId'));

                //get the list of stores on first
load from DB
```

```
            $.getJSON("get-stores", function(allData)
{

                var mappedItems = $.map(allData,
function(store) { return new Store(store); });

                //fill stores array

                self.stores(mappedItems);

                //have a copy

                self.storesCopy(self.stores.slice());



            });



            //initialize datetimepickers

            var datepicker1 =
$('#datetimepicker1').datetimepicker({

            format: 'YYYY-MM-DD',

            maxDate: new Date()

        });

             var datepicker2 =
$('#datetimepicker2').datetimepicker({

            format: 'YYYY-MM-DD',

            maxDate: new Date()

        });



             //dates

                  self.dateFrom =
ko.observable($('#datetimepicker1').data('date'));

                  self.dateTo =
ko.observable($('#datetimepicker2').data('date'));



             datepicker1.on('dp.change', function (e) {


self.dateFrom($('#datetimepicker1').data('date'));

             });



             datepicker2.on('dp.change', function (e) {

         self.dateTo($('#datetimepicker2').data('date')
);

             });



             //get the list of areas on first load from
DB

             $.getJSON("get-areas", function(allData) {

                  var mappedItems = $.map(allData,
function(area) { return new Area(area); });

                 //assign to areas

                 self.areas(mappedItems);

             });

                  self.getTotalSales =
function(receipts) {

                            //set to 0

                            self.totalSales(0);

                            var total = 0;

                            for(var i = 0;
i<receipts.length; i++) {

                                     var saleTotal =
receipts[i].amountDue;

                                     total +=
saleTotal;

                            }

                            self.totalSales(total);

                            return;

                  };



                  self.displayMoney =
function(totalSales) {

                            var money =
totalSales.toString().replace(/(\d)(?=(\d\d\d)+(?!\d))/g
, "$1,");

                            var moneyArray =
money.split(".");

                            if(moneyArray.length==1) {
```

```
                                money += ".00";
                        } else {

        if(moneyArray[1].length==1) {

        moneyArray[1] += "0";
                                money =
moneyArray[0] + "." + moneyArray[1];
                                } else {
                                money =
moneyArray[0] + "." + moneyArray[1];
                                }

                        }
                        return
self.totalSales(money);



                };


                //run reports
                self.getReports = function() {
                        self.getAllReports();
                };


                self.getTopSellingByQuantity =
function() {
                        var url =
"getTopSellingByQuantity";
                        $.ajax({
                                url: url,
                                dataType: 'json',
                                data: {areaId:
self.areaid(), storeId: self.storeid(), dateFrom:
self.dateFrom(), dateTo: self.dateTo},
                                success:
function(topSelling) {
                                                var
topSellingArray = [];


                                $.each(topSelling, function(key, value){

console.log(key, value);

                                var element = [];


                                element.push(key);

                                element.push(value);


                                topSellingArray.push(element);
                                                });


                                Highcharts.chart('barGraph1', {

chart: {

type: 'column'
                                                },

title: {

text: 'Top-selling by quantity'
                                                },

xAxis: {

type: 'category',

labels: {
```

```
rotation: -45,
style: {
fontSize: '13px',
fontFamily: 'Verdana, sans-serif'
}
}
},
yAxis: {
min: 0,
title: {
text: 'Quantity'
}
},
legend: {
enabled: false
},
tooltip: {
pointFormat: 'As of the moment: <b>{point.y}pcs</b>'
},
series: [{
name: 'Quantity',
data: topSellingArray,
dataLabels: {
enabled: true,
rotation: -90,
color: '#FFFFFF',
align: 'right',
format: '{point.y}', // one decimal
y: 10, // 10 pixels down from the top
style: {
fontSize: '13px',
fontFamily: 'Verdana, sans-serif'
}
}
}]
});
},
error:
function(jqXHR, textStatus, errorThrown) {
}
});
};
self.getTopSellingByAmount =
function() {
var url =
"getTopSellingByAmount";
$.ajax({
url: url,
```

```
                                    dataType: 'json',
                                    data: {areaId:
self.areaid(), storeId: self.storeid(), dateFrom:
self.dateFrom(), dateTo: self.dateTo},
                                    success:
function(topSelling) {
                                              var
topSellingArray = [];


          $.each(topSelling, function(key, value){
console.log(key, value);
//process receipts
          var element = [];


          element.push(key);
          element.push(value);


          topSellingArray.push(element);
                                              });


          Highcharts.chart('barGraph2', {
chart: {
type: 'column'
                                                   },
title: {
text: 'Top-selling by amount'
                                                                                                        },
xAxis: {
type: 'category',
labels: {
rotation: -45,
style: {
fontSize: '13px',
fontFamily: 'Verdana, sans-serif'
}
                                                                                                             }
                                                                                                        },
yAxis: {
min: 0,
title: {
text: 'Amount (PhP)'
                                                                                                             }
                                                                                                        },
legend: {
enabled: false
                                                                                                        },
tooltip: {
pointFormat: 'As of the moment: <b>{point.y:.2f}Php</b>'
```

```
                                        },

series: [{

name: 'Sales',

data: topSellingArray,

dataLabels: {

enabled: true,

rotation: -90,

color: '#FFFFFF',

align: 'right',

format: '{point.y:.1f}', // one decimal

y: 10, // 10 pixels down from the top

style: {

fontSize: '13px',

fontFamily: 'Verdana, sans-serif'

}

                                    }

                                }]

                            });

                        },

                        error:
function(jqXHR, textStatus, errorThrown) {

                        }

                    });

                };

                self.displayPaymentTypes = function()
{

                        var cashNum = 0;

                        var debitNum = 0;

                        var creditNum = 0;

                        var voucherNum = 0;

                        for(var x in
self.payments()) {

        if(self.payments()[x].pPaymentType()=="Cash")
{

        cashNum++;

                                } else if
(self.payments()[x].pPaymentType()=="Debit") {

        debitNum++;

                                } else if
(self.payments()[x].pPaymentType()=="Credit") {

        creditNum++;

                                } else if
(self.payments()[x].pPaymentType()=="Voucher") {

        voucherNum++;

                                }

                        }

                        //clean the array first

        self.paymentSummary.removeAll();

                        var paymentArray = [{name:
'Cash', y: cashNum},{name: 'Debit', y: debitNum},{name:
'Credit', y: creditNum}, {name: 'Voucher', y:
voucherNum}];

                        //High charts
```

```
                        // Build the chart

Highcharts.chart('chartsContainer', {
                        chart: {
                            plotBackgroundColor:
null,
                            plotBorderWidth: null,
                            plotShadow: false,
                            type: 'pie'
                        },
                        title: {
                            text: 'Payment type
shares'
                        },
                        tooltip: {
                            pointFormat:
'{series.name}:
<b>{point.percentage:.1f}%</b><br/>Quantity:
<b>{point.y}</b>'
                        },
                        plotOptions: {
                            pie: {
                                allowPointSelect:
true,
                                cursor: 'pointer',
                                dataLabels: {
                                    enabled: false
                                },
                                showInLegend: true
                            }
                        },
                        series: [{
                            name: 'Payment types',
                            colorByPoint: true,
                            data: paymentArray
                        }]
                    });
                };
                self.getPayments = function() {
                        var url = 'getPayments';

                        $.ajax({
                                url: url,
                                dataType: 'json',
                                data: {areaId:
self.areaid(), storeId: self.storeid(), dateFrom:
self.dateFrom(), dateTo: self.dateTo},
                                success:
function(payments) {
                                        //map to
payments array
                                        var
mappedItems = $.map(payments, function(payment) { return
new Payment(payment); });
                                    //save to array

                        self.payments(mappedItems);

                                        //clean
data
                        self.displayPaymentTypes();
                                }
                        });
                };

                self.getReceipts = function() {
                        var url = 'getReceipts';
                        $.ajax({
                                url: url,
                                dataType: 'json',
```

```
                                    data: {areaId:
self.areaid(), storeId: self.storeid(), dateFrom:
self.dateFrom(), dateTo: self.dateTo},

                                    success:
function(receipts) {

                                        //set
transactions count

        self.transactionCount(receipts.length);



                                        //get the
total sales

        self.getTotalSales(receipts);

                                        //map to
receipts array

                                        var
mappedItems = $.map(receipts, function(receipt) { return
new Receipt(receipt); });

                                    //save to array

        self.receipts(mappedItems);

                                        var
receiptArray = [];

                                        //process
receipts

                                        for(var x
in self.receipts()) {

        var element = [];



        element.push(self.receipts()[x].timestamp());

        element.push(self.receipts()[x].amountDue());



        receiptArray.push(element);

                                    }

        Highcharts.setOptions({

                                        global: {

useUTC: false

                                        },

                                        lang: {

thousandsSep: ','

                                        }

                                    });

        //generate high chart

        Highcharts.chart('timeseriesContainer', {

                                        chart: {

zoomType: 'x'

                                        },

                                        title: {

                                            text:
'Sales over time'

                                        },

                                        xAxis: {

                                            type:
'datetime'

                                        },

                                        yAxis: {

                                            title:
{

text: 'Amount'

                                            }
```

```
},
legend: {
enabled: false
},
plotOptions: {
    area:
{
fillColor: {
linearGradient: {
x1: 0,
y1: 0,
x2: 0,
y2: 1
},
stops: [
[0, Highcharts.getOptions().colors[0]],
[1,
Highcharts.Color(Highcharts.getOptions().colors[0]).setO
pacity(0).get('rgba')]
]
},
marker: {
radius: 2
},
lineWidth: 1,
states: {
hover: {
lineWidth: 1
}
},
threshold: null
},
series: [{
type:
'area',
name:
'Sales',
data:
receiptArray
}]
});
//high
charts
var
gaugeOptions = {
chart: {
type: 'solidgauge'
},
title: null,
pane:
{
```

```
center: ['50%', '85%'],

size: '140%',

startAngle: -90,

endAngle: 90,

background: {

backgroundColor: (Highcharts.theme &&
Highcharts.theme.background2) || '#EEE',

innerRadius: '60%',

outerRadius: '100%',

shape: 'arc'

                }
            },

tooltip: {

enabled: false

            },

            //
the value axis

yAxis: {

stops: [

[0.1, '#55BF3B'], // green

[0.5, '#DDDF0D'], // yellow

[0.9, '#DF5353'] // red

],

lineWidth: 0,

minorTickInterval: null,

tickAmount: 2,

title: {

y: -70

},

labels: {

y: 16

                }

            },

            plotOptions: {

solidgauge: {

dataLabels: {

y: 5,

borderWidth: 0,

useHTML: true

}

                }

            }

        };

        // The
RPM gauge
```

```
        Highcharts.chart('sales',
Highcharts.merge(gaugeOptions, {

//
        formatter: function () {

//
          return
Highcharts.numberFormat(self.totalSales(), 1, '.', ',');

//
        },


yAxis: {

min: 0,

max: 50000000,

title: {

text: 'Sales'

                                                    }

                                                },


series: [{

name: 'Sales',

data: [self.totalSales()],

dataLabels: {

format: '<div style="text-align:center"><span
style="font-size:25px;color:' +

((Highcharts.theme &&
Highcharts.theme.contrastTextColor) || 'black') + '">{y:
,.2f}</span><br/>' +

'<span style="font-
size:12px;color:silver">PhP</span></div>'

},

tooltip: {

valueSuffix: ' PhP'

                                                        }

                                                    }]

}));


                                                // The
speed gauge

        Highcharts.chart('purchases',
Highcharts.merge(gaugeOptions, {

yAxis: {

min: 0,

max: 1000,

title: {

text: 'No. of Purchases'

                                                        }

                                                },


credits: {

enabled: false

                                                },


series: [{

name: 'Transactions',

data: [self.transactionCount()],
```

```
dataLabels: {

format: '<div style="text-align:center"><span
style="font-size:25px;color:' +

((Highcharts.theme &&
Highcharts.theme.contrastTextColor) || 'black') +
'">{y}</span><br/>' +

'<span style="font-
size:12px;color:silver">Quantity</span></div>'

},

tooltip: {

//valueSuffix: ' km/h'
                                                    }
                                                }]
                                            }));

                                    }
                            });
                    };

                    //get all reports for high charts and
dashboards
                    self.getAllReports = function() {
                            //get reports/charts
                            self.getReceipts();
                            //get payments
                            self.getPayments();
                            //get top-selling

          self.getTopSellingByAmount();

          self.getTopSellingByQuantity();
                    };

                    //listen if new areaid is selected in
selection

self.areaid.subscribe(function(newAreaIdValue) {
                    if(newAreaIdValue) {
                            //clear stores array
                            self.stores.removeAll();
                            //update the selection in
store branches
                            for(var x in
self.storesCopy()) {
                                    if
((self.storesCopy()[x].area_id() == newAreaIdValue)) {
          self.stores.push(self.storesCopy()[x]);
                                    }
                            }

                    }
                    else {
                            //return copy of stores
          self.stores(self.storesCopy().slice());
                            //set value
                            self.storeid(null);

                    }
                });
```

```
	}

	ko.applyBindings(new ReportViewModel());

});

return.js

/**

 * This is the Jquery file for the pos.jsp page

 */

if(Cookies.get('usertype')==1) {

	//redirect to home page

	window.location.replace('home');

}

$(function() {

	/**

	 * Start of knockout.js

	 *

	 */



	function Transaction(transaction) {

		this.transactionId =
ko.observable(transaction.transactionId);

		this.itemDescription =
ko.observable(transaction.description);

		this.receiptId =
ko.observable(transaction.receiptId);

		this.inventoryId =
ko.observable(transaction.inventoryId);

		this.itemQuantity =
ko.observable(transaction.quantity);

		this.itemPrice =
ko.observable(transaction.price);

		this.itemDiscount =
ko.observable(transaction.discount);

		this.itemQuantityToReturn =
ko.observable(0);

	}



	function ReturnItemsViewModel() {

		var self = this;



		//used in cookies

		self.loggedInUser =
ko.observable(Cookies.get('username'));

		self.loggedInUsertype =
ko.observable(Cookies.get('usertype'));

		self.loggedInUserStoreId =
ko.observable(Cookies.get('userStoreId'));



		//These are used to search for
transactions using receipt Id

		self.receiptId = ko.observable("");



		//container of the receipt items

		self.transactionItems =
ko.observableArray([]);



		//container for items to be returned

		self.returnedItems =
ko.observableArray([]);





		//flag for button

		self.isReturnable =
ko.observable(false);



		//addItemQuantity is clicked

		self.addItemQuantity = function() {

	this.itemQuantityToReturn(this.itemQuantityToR
eturn() + 1);
```

```
                        //loop through
self.transactionItems and check if quantity of items to
be returned is > 0

                        for (var x in
self.transactionItems()) {


        if(self.transactionItems()[x].itemQuantityToRe
turn() > 0 ) {


        self.isReturnable(true);

                                        return;

                                }

                        }


                        self.isReturnable(false);


                };


                //subtractItemQuantity is clicked

                self.subtractItemQuantity =
function() {


        this.itemQuantityToReturn(this.itemQuantityToR
eturn() - 1);


                        //loop through
self.transactionItems and check if quantity of items to
be returned is > 0

                        for (var x in
self.transactionItems()) {


        if(self.transactionItems()[x].itemQuantityToRe
turn() > 0 ) {


        self.isReturnable(true);

                                        return;

                                }

                        }


                        self.isReturnable(false);

                };


                //returnItems

                self.returnItems = function() {

                        for (var x in
self.transactionItems()) {

                                if
(self.transactionItems()[x].itemQuantityToReturn() > 0)
{


                self.returnedItems.push(self.transactionItems(
)[x]);

                                }

                        }


                        var url = "return-items";


                        $.ajax({

                                url: url,

                                contentType:
'application/json; charset=utf-8',

                                data:
{returnedItems: ko.toJSON(self.returnedItems()),
storeId: self.loggedInUserStoreId()},

                                success:
function(returnMessage) {

                                        if
(returnMessage!=null) {



        alert("Success!\nPlease copy this voucher
number: " + returnMessage


                + ".\nThis will be used for payment
later.");
```

```
self.returnedItems.removeAll();



// notify

$.notify({

// options

icon: 'glyphicon glyphicon-ok',

message: 'Items successfully returned'

},{

// settings

type: 'success',

delay: 1000,

offset: 55,

});

//2.) view receipt



// close the modal

$('[data-dismiss=modal]').click();



// empty the array of items to return

self.returnedItems.removeAll();

// set isReturnable to false

        self.isReturnable(false);

                                        }

                                }

                        });

                };

                self.getTransactionsByReceipt =
function() {

                        //clear error first

        $("#receiptIdError").text("");

                        var receiptId =
self.receiptId();

                        var url1 =
'getVoucherByReceiptId';

                        $.ajax({

                                url: url1,

                                data: {receiptId:
receiptId},

                                success:
function(result) {

        if(result=='success') {

        //error message: You already have returned
item(s) using the receipt Id before.

        $("#receiptIdError").text("You already have
returned item(s) using the receipt Id before.");

                                        } else {

        var url2 = 'getTransactionsByReceipt';



        $.ajax({

        url: url2,
```

```
		dataType: 'json',

		data: {receiptId: receiptId},

		success: function(transactions) {



			var mappedItems = $.map(transactions,
function(transaction) { return new
Transaction(transaction); });



			//set to transactionItems the results

			self.transactionItems(mappedItems);



			//show returnItem modal

			$('#returnItemModal').modal('show');

		}

		});
                                        }
                                }
                        });

		};

		}

		ko.applyBindings(new ReturnItemsViewModel());
});
```

rmc.js

```
/**
 * This is the Jquery file for the pos.jsp page
 */
if(Cookies.get('usertype')!=1) {
		//redirect to home page
		window.location.replace('home');
}
$(function() {
		/**
		 * Start of knockout.js
		 *
		 */
		function Area(area) {
		    this.areaId = ko.observable(area.areaId);
		    this.areaName =
ko.observable(area.areaName);
		    this.zoomLevel =
ko.observable(area.zoomLevel);
		}

		function Store(store) {
		    this.storeId =
ko.observable(store.storeId);
		    this.area =
ko.observable(store.area.areaName);
		    this.area_id =
ko.observable(store.area.areaId);
		    this.branchName =
ko.observable(store.branchName);
		    this.address =
ko.observable(store.address);
		    this.coordinates =
ko.observable(store.coordinates);
		}

		function Payment(payment) {
```

```
                this.pPaymentType =
ko.observable(payment.paymentType);

                this.pAmount =
ko.observable(payment.amount);
        }


        function RMCViewModel() {
                var self = this;


                //colors pre-assigned to layers:
Blue, Yellow, Red, Gray, Black
                self.layerColors = ['#0000FF',
'#FFFF00', '#FF0000', '#808080', '#000000'];


                //global variables for use in high
charts
                self.currentAreaId =
ko.observable(0);
                self.currentStoreId =
ko.observable(0);
                self.currentViewContainer =
ko.observable("Philippines");
                self.currentView =
ko.observable(self.currentViewContainer());
                self.gTimer = null;
                self.gStatuses = [false, false,
false, false];


                //used in the list
                self.stores = ko.observableArray([]);
                self.areas = ko.observableArray([]);


                //boolean for zooms
                self.LuzonIsZoomed =
ko.observable(false);
                self.VisayasIsZoomed =
ko.observable(false);
                self.MindanaoIsZoomed =
ko.observable(false);


                //used in charts
                self.transactionCount =
ko.observable(0);
                self.totalSales = ko.observable(0);
                self.payments =
ko.observableArray([]);
                self.paymentSummary =
ko.observableArray([]);


                //markers
                self.markers =
ko.observableArray([]);
                // marker properties
                var cartMarker =
L.AwesomeMarkers.icon({
                        markerColor: 'red',
                        prefix: 'glyphicon',
                    icon: 'shopping-cart',
                    iconColor: 'black'
                });
                var clickedMarker =
L.AwesomeMarkers.icon({
                        markerColor: 'green',
                        prefix: 'glyphicon',
                    icon: 'shopping-cart',
                    iconColor: 'black'
                });


                //get the list of stores on first
load from DB
```

```
            $.getJSON("get-stores", function(allData)
{

                var mappedItems = $.map(allData,
function(store) { return new Store(store); });

                //assign to stores

                self.stores(mappedItems);

            });



            var customControl = L.Control.extend({

                                options: {

                                  position: 'topleft'

                                  //control position -
allowed: 'topleft', 'topright', 'bottomleft',
'bottomright'

                                },

                                onAdd: function (map) {

                                                var container
= L.DomUtil.create('div', 'leaflet-control leaflet-
control-custom');

                                                return
container;

                                }

                  });



                  //initialise the map

                  var mymap = L.map('rmcMap', {
zoomControl: false });

                  // Disable drag and zoom handlers.

                  mymap.touchZoom.disable();

                  mymap.scrollWheelZoom.disable();

                  mymap.keyboard.disable();

                  mymap.dragging.disable();

                  mymap.doubleClickZoom.disable();

                  //add custom control

                  mymap.addControl(new
customControl());

                  //popup

                  var layerPopup = null;



                  // load a tile layer


        L.tileLayer('https://api.tiles.mapbox.com/v4/{
id}/{z}/{x}/{y}.png?access_token={accessToken}', {

                      attribution: 'Map data &copy; <a
href="http://openstreetmap.org">OpenStreetMap</a>
contributors, Imagery © <a
href="http://mapbox.com">Mapbox</a>',

                      minZoom: 5,

                      maxZoom: 18,

                      id: 'roldanreal.pmm3gdhh',

                      accessToken:
'pk.eyJ1Ijoicm9sZGFucmVhbCIsImEiOiJjaW15OXZkNGswM3p3djdr
azdmbHVrdHl2In0.Q1oLza6ZqNpyKDiGs24wbg'

                  }).addTo(mymap);



                  //set zoomed to false

                  self.LuzonIsZoomed(false);

                  self.VisayasIsZoomed(false);

                  self.MindanaoIsZoomed(false);



                  //get the list of areas on first load
from DB

            $.getJSON("get-areas", function(allData) {

                  var mappedItems = $.map(allData,
function(area) { return new Area(area); });

                //assign to areas

                self.areas(mappedItems);

            });

                  //GeoJSON Layer Luzon
```

```
                var geojsonLuzonLayer = new
L.GeoJSON.AJAX('getLuzonJson', {color: '#0000FF',
weight: 2}).addTo(mymap);

                //GeoJSON Layer Visayas

                var geojsonVisayasLayer = new
L.GeoJSON.AJAX('getVisayasJson', {color: '#FFFF00',
weight: 2}).addTo(mymap);

                //GeoJSON Layer Mindana

                var geojsonMindanaoLayer = new
L.GeoJSON.AJAX('getMindanaoJson', {color: '#FF0000',
weight: 2}).addTo(mymap);



                self.viewPhils = function() {

                        //initially, display
Philippines only as Control

                        var controlDiv =
$('.leaflet-control-custom');

                        //empty it first

                        controlDiv.empty();

                        var link = "<div id='custom-
controls'>Active:&amp;nbsp;&amp;nbsp;<a id='ph-custom-control'
href='#'>Philippines</a></div>";

                        controlDiv.append(link);

                        //remove the store layers

                        for(var x in self.markers())
{

        mymap.removeLayer(self.markers()[x]);

                        }

                        //empty markers list

                        self.markers.removeAll();



                        //set the strokes


        geojsonLuzonLayer.setStyle({stroke :
'#0000FF'});

        geojsonVisayasLayer.setStyle({stroke
:'#FFFF00'});


        geojsonMindanaoLayer.setStyle({stroke
:'#FF0000'});



                        //bind event

                        $('#ph-custom-
control').on('click', function() {

                                self.viewPhils();

                        });

                        mymap.setView([11.600960,
123.473753], 5); //Visayan Sea, Philippines as center



                        //set global variables

                        self.currentAreaId(0);

                        self.currentStoreId(0);


        self.currentViewContainer("Philippines");



                        //reset flags

                        self.LuzonIsZoomed(false);

                        self.VisayasIsZoomed(false);


        self.MindanaoIsZoomed(false);



                };


                self.getTotalSales =
function(receipts) {

                        //set to 0

                        self.totalSales(0);

                        var total = 0;
```

```
                        for(var i = 0;
i<receipts.length; i++) {
                                var saleTotal =
receipts[i].amountDue;
                                total +=
saleTotal;
                        }
                        self.totalSales(total);
                        return;
                };


                self.displayMoney =
function(totalSales) {
                        var money =
totalSales.toString().replace(/(\d)(?=(\d\d\d)+(?!\d))/g
, "$1,");
                        var moneyArray =
money.split(".");
                        if(moneyArray.length==1) {
                                money += ".00";
                        } else {

        if(moneyArray[1].length==1) {

        moneyArray[1] += "0";
                                        money =
moneyArray[0] + "." + moneyArray[1];
                                } else {
                                        money =
moneyArray[0] + "." + moneyArray[1];
                                }
                        }
                        self.totalSales(money);
                };


                self.displayPaymentTypes = function()
{
                        var cashNum = 0;
                        var debitNum = 0;
                        var creditNum = 0;
                        var voucherNum = 0;
                        for(var x in
self.payments()) {

        if(self.payments()[x].pPaymentType()=="Cash")
{

        cashNum++;
                                } else if
(self.payments()[x].pPaymentType()=="Debit") {

        debitNum++;
                                } else if
(self.payments()[x].pPaymentType()=="Credit") {

        creditNum++;
                                } else if
(self.payments()[x].pPaymentType()=="Voucher") {

        voucherNum++;
                                }
                        }

                        //clean the array first

        self.paymentSummary.removeAll();

                        var paymentArray = [{name:
'Cash', y: cashNum},{name: 'Debit', y: debitNum},{name:
'Credit', y: creditNum}, {name: 'Voucher', y:
voucherNum}];

                        //High charts
                        // Build the chart
```

```
Highcharts.chart('graphContainer', {
                            chart: {
                                plotBackgroundColor:
null,
                                plotBorderWidth: null,
                                plotShadow: false,
                                type: 'pie'
                            },
                            title: {
                                text: 'Payment types'
                            },
                            tooltip: {
                                pointFormat:
'{series.name}:
<b>{point.percentage:.1f}%</b><br/>Quantity:
<b>{point.y}</b>'
                            },
                            plotOptions: {
                                pie: {
                                    allowPointSelect:
true,
                                    cursor: 'pointer',
                                    dataLabels: {
                                        enabled: false
                                    },
                                    showInLegend: true
                                }
                            },
                            series: [{
                                name: 'Payment types',
                                colorByPoint: true,
                                data: paymentArray
                            }]
                                                    });
                                                };
                                                self.getPayments = function() {
                                                        var url = 'getPayments';
                                                        var statusIdx = 2 - 1;
                                                        //if request is not yet
done, wait for request to be done
                                                if (self.gStatuses[statusIdx]) {
                                                console.log("Report 2: waiting for response,
skip request");
                                                        return;
                                                        }
                                                self.gStatuses[statusIdx] = true;
                                                        $.ajax({
                                                                url: url,
                                                                dataType: 'json',
                                                                data: {areaId:
self.currentAreaId(), storeId: self.currentStoreId()},
                                                                success:
function(payments) {
                                                                        //map to
payments array
                                                                        var
mappedItems = $.map(payments, function(payment) { return
new Payment(payment); });
                                                                //save to array
                                                self.payments(mappedItems);
                                                                        //clean
data
                                                self.displayPaymentTypes();
                                                                        //set to
false
```

```
        self.gStatuses[statusIdx] = false;
                                },
                                error:
function(jqXHR, textStatus, errorThrown) {
                                        //set to
false
        self.gStatuses[statusIdx] = false;
                                }
                        });
                };

                self.getCustomers = function() {
                        var url = 'getReceipts';
                        var statusIdx = 1 - 1;
                        //if request is not yet
done, wait for request to be done
                if (self.gStatuses[statusIdx]) {

        console.log("Report 1: waiting for response,
skip request");
                                return;
                        }
                self.gStatuses[statusIdx] = true;
                        $.ajax({
                                url: url,
                                dataType: 'json',
                                data: {areaId:
self.currentAreaId(), storeId: self.currentStoreId()},
                                success:
function(receipts) {
                                        //set
transactions count

        self.transactionCount(receipts.length);
                                        //get the
total sales

        self.getTotalSales(receipts);
                                        //clean
total sales

        self.displayMoney(self.totalSales());
                                        //set
view

        self.currentView(self.currentViewContainer());
                                        //set to
false

        self.gStatuses[statusIdx] = false;
                                },
                                error:
function(jqXHR, textStatus, errorThrown) {
                                        //set to
false

        self.gStatuses[statusIdx] = false;
                                }
                        });
                };
                //
                self.getTopSellingByAmount =
function() {
                        var url =
"getTopSellingByAmount";
                        var statusIdx = 4 - 1;
                        //if request is not yet
done, wait for request to be done
```

```
                if (self.gStatuses[statusIdx]) {
        console.log("Report 4: waiting for response,
skip request");
                                return;
                        }
                self.gStatuses[statusIdx] = true;
                        $.ajax({
                                url: url,
                                dataType: 'json',
                                data: {areaId:
self.currentAreaId(), storeId: self.currentStoreId()},
                                success:
function(topSelling) {
                                        var
topSellingArray = [];



        $.each(topSelling, function(key, value){
console.log(key, value);
//process receipts
        var element = [];



        element.push(key);
        element.push(value);



        topSellingArray.push(element);
                                });


        Highcharts.chart('barGraphContainer2', {
chart: {
type: 'column'
                                },
title: {
text: 'Top-selling by amount'
                                },
xAxis: {
type: 'category',
labels: {
rotation: -45,
style: {
fontSize: '13px',
fontFamily: 'Verdana, sans-serif'
}
                                }
                                },
yAxis: {
min: 0,
title: {
text: 'Amount (PhP)'
                                }
                                },
```

```
legend: {
enabled: false
},
tooltip: {
pointFormat: 'As of the moment: <b>{point.y:.2f}Php</b>'
},
series: [{
name: 'Sales',
data: topSellingArray,
dataLabels: {
enabled: true,
rotation: -90,
color: '#FFFFFF',
align: 'right',
format: '{point.y:.1f}', // one decimal
y: 10, // 10 pixels down from the top
style: {
fontSize: '13px',
fontFamily: 'Verdana, sans-serif'
}
}]
});
//set to false
self.gStatuses[statusIdx] = false;
},
error: function(jqXHR, textStatus, errorThrown) {
//set to false
self.gStatuses[statusIdx] = false;
}
});
};
self.getTopSellingByQuantity = function() {
var url = "getTopSellingByQuantity";
var statusIdx = 3 - 1;
//if request is not yet done, wait for request to be done
if (self.gStatuses[statusIdx]) {
console.log("Report 3: waiting for response, skip request");
return;
}
self.gStatuses[statusIdx] = true;
$.ajax({
url: url,
dataType: 'json',
data: {areaId: self.currentAreaId(), storeId: self.currentStoreId()},
}
```

```
                                    success:
function(topSelling) {
                                                var
topSellingArray = [];


          $.each(topSelling, function(key, value){
console.log(key, value);
          var element = [];


          element.push(key);

          element.push(value);


          topSellingArray.push(element);
                                                });


          Highcharts.chart('barGraphContainer1', {
chart: {
type: 'column'
                                                     },
title: {
text: 'Top-selling by quantity'
                                                     },
xAxis: {
type: 'category',
labels: {
rotation: -45,
style: {
fontSize: '13px',
fontFamily: 'Verdana, sans-serif'
}
                                                                          }
                                                                     },
yAxis: {
min: 0,
title: {
text: 'Quantity'
                                                                          }
                                                                     },
legend: {
enabled: false
                                                                     },
tooltip: {
pointFormat: 'As of the moment: <b>{point.y}pcs</b>'
                                                                     },
series: [{
name: 'Quantity',
```

```
data: topSellingArray,

dataLabels: {

enabled: true,

rotation: -90,

color: '#FFFFFF',

align: 'right',

format: '{point.y}', // one decimal

y: 10, // 10 pixels down from the top

style: {

fontSize: '13px',

fontFamily: 'Verdana, sans-serif'

}
                                                        }
                                                    }]
                                                });


                                                //set to
false
          self.gStatuses[statusIdx] = false;
                                                },
                                                error:
function(jqXHR, textStatus, errorThrown) {
                                                        //set to
false
          self.gStatuses[statusIdx] = false;
                                                        }
                                                });
                                        };

                                        //get all reports for high charts and
dashboards
                                        self.getAllReports = function() {
                                                if (self.gTimer) {
          clearTimeout(self.gTimer);
                                                }
                                                console.log("getting all
reports: current view: " + self.currentViewContainer());
                                                //get reports/charts
                                                self.getCustomers();
                                                //get payments
                                                self.getPayments();
                                                //get topselling by quantity
          self.getTopSellingByQuantity();
                                                //get topselling by amount
          self.getTopSellingByAmount();
                                                gTimer =
setTimeout(function() {self.getAllReports();}, 5000);
                                        };

                                        //first view
                                        self.viewPhils();
                                        //get reports
                                        self.getAllReports();

                                        //a function that sets markers
                                        self.setMarkers = function(feature) {
```

```
                        var areaId =
feature.properties.AreaId;

                        for(var x in self.stores())
{

                                //Add the stores
to map

                        if
((self.stores()[x].area_id() == areaId)) {

                                var lat =
self.stores()[x].coordinates().split(",")[0];

                                var long =
self.stores()[x].coordinates().split(",")[1];

                                //extend the
marker to add the branch name in the options

                                var customMarker =
L.Marker.extend({

                                        options:
{

        branchName: self.stores()[x].branchName(),


        storeId: self.stores()[x].storeId()

                                        }

                                });

                                var marker = new
customMarker([lat,long], {icon:
cartMarker}).addTo(mymap)

                            .bindPopup('<img
src=\'resources/images/favicon.ico\'></img><strong>' +
self.stores()[x].branchName() + '</strong><br />' +


        self.stores()[x].address() + '<br/>'

                                        ,
{autoPan:true})

                            .on('mouseover',
function() { this.openPopup(); })

                                .on('mouseout',
function() { this.closePopup(); })

                                .on('click',
function(e) {

                                        //reset
all icons

                                        for(var y
in self.markers()) {


        var indivMarker = self.markers()[y];


        indivMarker.setIcon(cartMarker);

                                        }

                                        //set
icon color to green


        this.setIcon(clickedMarker);

                                        //remove
first custom control link for branch


        $('#branch-custom-control').remove();

                                        //update
the control

                                        var
controlDiv = $('#custom-controls');

                                        //add the
branch name

                                        var link
= "<span id='branch-custom-control'>  ><a
href='#'> " + e.target.options.branchName; +
"</a></span>";


        controlDiv.append(link);

                                        //set
current view


        self.currentViewContainer(e.target.options.bra
nchName);

                                        //current
storeID


        self.currentStoreId(e.target.options.storeId);

                                    });
```

```
        self.markers.push(marker);
                        }
                }
                };

                self.viewArea = function(area) {
                        //reset all icons
                        for(var x in self.markers())
{
                                var indivMarker =
self.markers()[x];

        indivMarker.setIcon(cartMarker);
                        }
                        //remove the branch in
Control
                        var branchLink = $('#branch-
custom-control');
                        branchLink.remove();
                        //set storeId to 0
                        self.currentStoreId(0);
                        //set the text

        self.currentViewContainer(area);
                };

                //Mouse events for Luzon
                geojsonLuzonLayer.on("mouseover",
function (e) {

        mymap.dragging._draggable._freeze=true;
                        var lat = e.latlng.lat;
                        var lang = e.latlng.lng;
                        layerPopup = L.popup()
                                .setLatLng([lat, lang])
                        .setContent("<b>Luzon</b>")
                        .openOn(mymap);
                         });

                         geojsonLuzonLayer.on("mouseout",
function (e) {

        mymap.dragging._draggable._freeze=true;

        mymap.closePopup(layerPopup);
                                layerPopup = null;
                         });

                         geojsonLuzonLayer.on("click",
function (e) {
                                if(!self.LuzonIsZoomed()) {
                                        //Add the stores

        geojsonLuzonLayer.refilter(function(feature){
                                                //set
markers

        self.setMarkers(feature);
                                        });

        mymap.setView([14.57794,120.9746711],7);
//Rizal Park, Manila as center
                                        //set zoomed to
true

        self.LuzonIsZoomed(true);
                                        //add to control
                                        var controlDiv =
$('#custom-controls');
                                        var link = "<span
id='area-custom-control'> ><a  id='mm-custom-
control' href='#'> Luzon</a></span>";
```

```
        controlDiv.append(link);

                                    //bind event
                                    $('#mm-custom-
control').on('click', function() {

        self.viewArea('Luzon');

                                    });
                                    //set current view
to Luzon, areaId 1 == Luzon

        self.currentAreaId(1);

                                    //set storeId to 0

        self.currentStoreId(0);

                                    //set the text

        self.currentViewContainer("Luzon");

                            }

                            else {

                                    self.viewPhils();

                            }

                    });
                    //Mouse events for Cebu area
                    geojsonVisayasLayer.on("mouseover",
function (e) {
                            var lat = e.latlng.lat;
                            var lang = e.latlng.lng;
                            layerPopup = L.popup()
                            .setLatLng([lat, lang])
                .setContent("<b>Visayas</b>")
                .openOn(mymap);
                    });

                    geojsonVisayasLayer.on("mouseout",
function (e) {

        mymap.closePopup(layerPopup);
                                layerPopup = null;
                    });

                    geojsonVisayasLayer.on("click",
function (e) {
                                if(!self.VisayasIsZoomed())
{
                                        //Add the stores

        geojsonVisayasLayer.refilter(function(feature)
{
                                                //set
markers

        self.setMarkers(feature);

                                        });

        mymap.setView([10.425131, 123.575514], 9);
//set to Tanon Strait, Negros Occidental as center
                                        //set zoomed to
true

        self.VisayasIsZoomed(true);
                                        //add to control
                                        var controlDiv =
$('#custom-controls');
                                        var link = "<span
id='area-custom-control'> ><a  id='cebu-custom-
control' href='#'> Visayas</a></span>";

        controlDiv.append(link);
                                        //bind event
```

```
                                $('#cebu-custom-
control').on('click', function() {

        self.viewArea('Visayas');

                                });

                                //set current view
to Visayas, areaId 2 == Visayas

        self.currentAreaId(2);

                                //set storeId to 0

        self.currentStoreId(0);

                                //set the text

        self.currentViewContainer("Visayas");

                        }


                        else {

                                self.viewPhils();

                        }


                });


                //Mouse events for Davao area

                geojsonMindanaoLayer.on("mouseover",
function (e) {

                        var lat = e.latlng.lat;

                        var lang = e.latlng.lng;

                        layerPopup = L.popup()

                        .setLatLng([lat, lang])

                .setContent("<b>Mindanao</b>")

                .openOn(mymap);

                 });


                geojsonMindanaoLayer.on("mouseout",
function (e) {

        mymap.closePopup(layerPopup);

                        layerPopup = null;

                });


                geojsonMindanaoLayer.on("click",
function (e) {

                        if(!self.MindanaoIsZoomed())
{

                                //Add the stores

        geojsonMindanaoLayer.refilter(function(feature
){

                                        //set
markers

        self.setMarkers(feature);

                                });

        mymap.setView([8.0291503,124.2736951], 8);
//set to Marawi City as center

                                //set zoomed to
true

        self.MindanaoIsZoomed(true);

                                //add to control

                                var controlDiv =
$('#custom-controls');

                                var link = "<span
id='area-custom-control'> ><a  id='davao-custom-
control' href='#'> Mindanao</a></span>";


        controlDiv.append(link);

                                //bind event

                                $('#davao-custom-
control').on('click', function() {
```

```
		self.viewArea('Mindanao');
                                    });
                                    //set current view
to Mindanao, areaId 3 == Mindanao

		self.currentAreaId(3);
                                    //set storeId to 0

		self.currentStoreId(0);
                                    //set the text

		self.currentViewContainer("Mindanao");
                        }

                        else {
                                    self.viewPhils();
                        }
                });

		}
		ko.applyBindings(new RMCViewModel());
});
stores.js
/**
 * This is the Jquery file for the stores.jsp page
 */
if(Cookies.get('usertype') != 1) {
		//redirect to home page
		window.location.replace('home');
}
$(function() {
		/**
		 * Start of knockout.js
		 *
		 */
		function Store(store) {
			this.storeId =
ko.observable(store.storeId);
			this.area =
ko.observable(store.area.areaName);
			this.area_id =
ko.observable(store.area.areaId);
			this.branchName =
ko.observable(store.branchName);
			this.tin = ko.observable(store.tin);
			this.address =
ko.observable(store.address);
			this.coordinates =
ko.observable(store.coordinates);
			this.delItemTitle = ko.observable("Delete
item " + store.storeId);
			this.editItemTitle = ko.observable("Edit
item " + store.storeId);
		}

		function Area(area) {
			this.areaId = ko.observable(area.areaId);
			this.areaName =
ko.observable(area.areaName);
		}

		// Overall viewmodel for this screen, along
with initial state
		function StoresViewModel() {
                        var self = this;

                        //used in cookies
                        self.loggedInUser =
ko.observable(Cookies.get('username'));
```

```
                self.loggedInUsertype =
ko.observable(Cookies.get('usertype'));

                self.loggedInUserStoreId =
ko.observable(Cookies.get('userStoreId'));



                //used in the list

                self.stores = ko.observableArray([]);

                self.areas = ko.observableArray([]);



                //copy of self.stores, this one will
be used in searching
            self.storesCopy = ko.observableArray([]);



            //for paging

            self.pageSize = ko.observable(10);

            self.NumberPages = ko.observable(1);

            self.currPage = ko.observable(1);

            self.pagesArray = ko.observableArray([]);

            self.maxNumberPages = ko.observable(1);



                //for the form

                self.storeid = ko.observable(0);

                self.areaid = ko.observable();

                self.branchname = ko.observable("");

                self.taxIdNumber = ko.observable("");

                self.branchaddress =
ko.observable("");

                self.coordinates = ko.observable("");



                //These are used in search

                self.searchString =
ko.observable("");

                self.searchResultsArray =
ko.observableArray([]);
```

```
                //initialise the map

                self.mymap =
L.map('mapContainer').setView([11.600960, 123.473753],
5); //Visayan Sea, Philippines as center

                //.setView([51.505, -0.09], 13);

                // Disable drag and zoom handlers.

                self.mymap.touchZoom.disable();

                self.mymap.scrollWheelZoom.disable();

                self.mymap.keyboard.disable();

//              mymap.dragging.disable();

                self.mymap.doubleClickZoom.disable();

//              //add custom control

//              mymap.addControl(new
customControl());



                //popup

                //var layerPopup = null;


        L.tileLayer('https://api.tiles.mapbox.com/v4/{
id}/{z}/{x}/{y}.png?access_token={accessToken}', {

                    attribution: 'Map data &copy; <a
href="http://openstreetmap.org">OpenStreetMap</a>
contributors, <a
href="http://creativecommons.org/licenses/by-
sa/2.0/">CC-BY-SA</a>, Imagery © <a
href="http://mapbox.com">Mapbox</a>',

                    minZoom: 5,

                    maxZoom: 18,

                    id: 'mapbox.streets',

                    accessToken:
'pk.eyJ1Ijoicm9sZGFucmVhbCIsImEiOiJjajMyZW5odTkwMDA0Mndu
d3FsbWU4MGEyIn0.4XNhZ5NL07H5tmKZJlTN7A'

                }).addTo(self.mymap);



                // load a tile layer
```

```
//
         L.tileLayer('https://api.tiles.mapbox.com/v4/{
id}/{z}/{x}/{y}.png?access_token={accessToken}', {

//                     attribution: 'Map data &copy; <a
href="http://openstreetmap.org">OpenStreetMap</a>
contributors, Imagery © <a
href="http://mapbox.com">Mapbox</a>',

//                     minZoom: 5,

//                     maxZoom: 18,

//                     id: 'mapbox.streets',

//                     accessToken:
'pk.eyJ1Ijoicm9sZGFucmVhbCIsImEiOiJjajMyZW5odTkwMDA0Mndu
d3FsbWU4MGEyIn0.4XNhZ5NL07H5tmKZJlTN7A'

//                 }).addTo(mymap);


                    //get the list of stores on first
load from DB

             $.getJSON("get-stores", function(allData)
{

                 var mappedItems = $.map(allData,
function(store) { return new Store(store); });


                 //make a copy of stores

                 self.storesCopy(mappedItems);


                 //Do paging on first load

                 self.doPaging(self.pageSize());


             });


             //get the list of areas on first load from
DB

             $.getJSON("get-areas", function(allData) {

                 var mappedItems = $.map(allData,
function(area) { return new Area(area); });

                 //fill areas array

                 self.areas(mappedItems);

             });


             self.displayStore = function() {

                  //change the modal title and button
text

                  $(".modal-title").text("Edit store");


         $("#editStoreSumbitButton").text("Update");


                  //map the values to modal form


         self.storeid(this.storeId());

                             self.areaid(this.area_id());


         self.branchname(this.branchName());


         self.taxIdNumber(this.tin());


         self.branchaddress(this.address());


         self.coordinates(this.coordinates());


                             //show the modal


         $('#editStoreModal').modal('show');

             };


             self.displayCoordinates = function() {


                             if(self.areaid() == 1) {


         self.mymap.setView([14.57794,120.9746711],7);
//Rizal Park, Manila as center

                             } else if(self.areaid() ==
2) {
```

```
        self.mymap.setView([10.425131, 123.575514],
9); //set to Tanon Strait, Negros Occidental as center
                            } else if(self.areaid() ==
3) {

        self.mymap.setView([8.0291503,124.2736951],
8); //set to Marawi City as center
                            } else {

        self.mymap.setView([11.600960, 123.473753],
5); //Visayan Sea, Philippines as center
                            }


                            //show the modal

        $('#mapSelectModal').modal('show');
                            $(".mod-title").text("Click
map to select coordinates");


            };


            $('#mapSelectModal').on('shown.bs.modal',
function(){
                              setTimeout(function() {

self.mymap.invalidateSize();
                              }, 10);
                 });


            self.mymap.on('click', function(e)
                            {
                                    //set latitude and
longitude to coordinates

        self.coordinates(e.latlng.lat + "," +
e.latlng.lng);
                                    //close the modal
        $('[id=mapModal]').click();
                            });


            self.selectCoordinates = function() {


            };


            self.displayAddStore = function() {
                 //change the modal title and button
text
                 $(".modal-title").text("Add new
store");

        $("#editStoreSumbitButton").text("Add");

                            //Clear the values inside
the form
                            self.storeid(0);
                            self.areaid("");
                            self.branchname("");
                            self.taxIdNumber("");
                            self.branchaddress("");
                            self.coordinates("");

                            //show the Edit modal

        $('#editStoreModal').modal('show');
            };


            self.cancelEdit = function() {
                 //Clear the values inside the form
                            self.areaid("");
                            self.branchname("");
```

```
                        self.taxIdNumber("");
                        self.branchaddress("");
                        self.coordinates("");
                };

                self.editStore = function() {
                        var storeId = this.storeid();
                        var areaId = this.areaid();
                        var branchname = this.branchname();
                        var tin = this.taxIdNumber();
                        var branchaddress =
this.branchaddress();
                        var coordinates = this.coordinates();

                        //Add new store
                        if(storeId==0) {
                                var url = 'add-store';
                                        $.ajax({
                                                url: url,
                                                dataType:
'json',
                                                data:
{areaId: areaId, branchName: branchname, branchaddress:
branchaddress, coordinates: coordinates, tin: tin},
                                                success:
function(store) {

        //close the modal

        $('[data-dismiss=modal]').click();



        var areaName = "";

        //loop through the areas array and get the
name
        for(var i = 0; i<self.areas().length; i++) {

        if (self.areas()[i].areaId() == areaId) {

                areaName =
self.areas()[i].areaName();

                break;
                                                }

        }

        if(self.searchString() != '') {

        self.searchResultsArray.push({

                storeId :
ko.observable(store.storeId),

            area : ko.observable(areaName),

            area_id: ko.observable(store.area.areaId),

            branchName :
ko.observable(store.branchName),

            tin: ko.observable(store.tin),

            address : ko.observable(store.address),

            coordinates :
ko.observable(store.coordinates),

            delItemTitle : ko.observable("Delete item
" + store.storeId),

            editItemTitle : ko.observable("Edit item "
+ store.storeId)

        });
```

```
                                        }

        else {

        self.stores.push({

                storeId :
ko.observable(store.storeId),

            area : ko.observable(areaName),

            area_id: ko.observable(store.area.areaId),

            branchName :
ko.observable(store.branchName),

            tin: ko.observable(store.tin),

            address : ko.observable(store.address),

            coordinates :
ko.observable(store.coordinates),

            delItemTitle : ko.observable("Delete item
" + store.storeId),

            editItemTitle : ko.observable("Edit item "
+ store.storeId)

        });

        }



        //add also in the copy

        self.storesCopy.push({

        storeId : ko.observable(store.storeId),

area : ko.observable(areaName),

area_id: ko.observable(store.area.areaId),

branchName : ko.observable(store.branchName),

tin: ko.observable(store.tin),

address : ko.observable(store.address),

coordinates : ko.observable(store.coordinates),

delItemTitle : ko.observable("Delete item " +
store.storeId),

editItemTitle : ko.observable("Edit item " +
store.storeId)

        });



        //Do paging

self.doPaging(self.pageSize(), self.currPage());



        //notify that adding is successful

        $.notify({

        // options

        icon: 'glyphicon glyphicon-ok',

        message: 'Store successfully added'

        },{

        // settings

        type: 'success',
```

```
          delay: 1000,

          offset: 55,

          });
                                                  },
                                                  error:
function(jqXHR, textStatus, errorThrown) {

          errorMessage = "Cannot add store. Please check
if branch name " +

          "already exists. Or try again later.";

          //Display error message

          $("#editStoreError").html(errorMessage);
                                                  }
                                        });
                    }
                    //Edit store details
                    else {
                              var url = 'update-store';
                                        $.ajax({
                                                  type:
'GET',
                                                  url: url,
                                                  data:
{storeId: storeId, areaId: areaId, branchName:
branchname, branchaddress: branchaddress, coordinates:
coordinates, tin: tin},
                                                  success:
function() {

          var areaName = "";

          //loop through the areas array and get the
name

          for(var i = 0; i<self.areas().length; i++) {

                    if (self.areas()[i].areaId() ==
areaId) {

                              areaName =
self.areas()[i].areaName();

                              break;

                    }

          }

          //loop through the items and update the value

          for(var i = 0; i<self.stores().length; i++) {

                    if (self.stores()[i].storeId() ==
storeId) {

                        self.stores()[i].area(areaName);


          self.stores()[i].area_id(areaId);

          self.stores()[i].branchName(branchname);

                              self.stores()[i].tin(tin);

          self.stores()[i].address(branchaddress);

          self.stores()[i].coordinates(coordinates);

break;

}
```

```
        }

        //loop through the items copy and update the
value

        for(var i = 0; i<self.stores().length; i++) {

                if (self.storesCopy()[i].storeId() ==
storeId) {

                    self.stores()[i].area(areaName);


        self.stores()[i].area_id(areaId);


        self.stores()[i].branchName(branchname);

                        self.stores()[i].tin(tin);


        self.stores()[i].address(branchaddress);


        self.stores()[i].coordinates(coordinates);

break;

}

        }

        //Do paging

self.doPaging(self.pageSize(), self.currPage());


$.notify({

                // options

                icon: 'glyphicon glyphicon-ok',

                message: 'Store details successfully
updated'

        },{

                // settings

                type: 'success',

                delay: 1000,

                offset: 55,

        });

        $('[data-dismiss=modal]').click();


                                                },

                                                error:
function() {

        errorMessage = "Cannot edit item. Please check
if item " +

                "code already exists. Or try again
later.";

        //Display error message

        $("#editItemError").html(errorMessage);

        return false;

                                                },

                                        });

                        }

                        return true;
```

```
            };

            self.removeStore = function(store, event)
{

                 // get storeId of the row

                          var storeId =
event.currentTarget.id;



                          bootbox.confirm({

                                   message: "You are
about to delete store " + storeId + ".\nDo you want to
proceed?",

                                   closeButton:
false,

                                   size: "small",

                              callback:
function(result){

                                   if(result) {


         var url = 'delete-' + storeId + '-store';


         $.ajax({


         url: url,


         success: function() {


                  //remove the element from the table


                  if(self.searchString() != '') {


                           //remove from
searchResultsArray


                           for(var x in
self.searchResultsArray()) {



         if(self.searchResultsArray()[x].storeId() ==
storeId) {


         self.searchResultsArray.remove(self.searchResu
ltsArray()[x]);


                                                  break;


                                    }


                           }


         }


         else {


                  self.stores.remove(store);


         }



                  //remove also from the copy


                  for(var x in self.storesCopy()) {


         if(self.storesCopy()[x].storeId() == storeId)
{


         self.storesCopy.remove(self.storesCopy()[x]);


                                    break;


                           }


                  }



                  //Do paging
```

```
self.doPaging(self.pageSize(), self.currPage());

                $.notify({

                        // options

                        icon: 'glyphicon glyphicon-
ok',

                        message: 'Store successfully
deleted'

                },{

                        // settings

                        type: 'success',

                        delay: 1000,

                        offset: 55,

                });

        },

        error: function(jqXHR, textStatus,
errorThrown) {

            alert("error:" + textStatus + "
exception:" + errorThrown);

        },

        });
                                        }
                            }
                        });
            };

            self.searchStores = function() {

                var searchString =
self.searchString();

                self.searchResultsArray.removeAll();


                if(searchString != '') {

                        for(var x in
self.storesCopy()) {

                                if
((self.storesCopy()[x].area().toLowerCase().indexOf(sear
chString.toLowerCase()) >= 0)


        ||
(self.storesCopy()[x].branchName().toLowerCase().indexOf
(searchString.toLowerCase()) >= 0)


        ||
(self.storesCopy()[x].address().toLowerCase().indexOf(se
archString.toLowerCase()) >= 0)
                                                ) {


        self.searchResultsArray.push(self.storesCopy()
[x]);

                                }

                        }

                }

                else {


        self.searchResultsArray(self.storesCopy().slic
e());

                }


                //Do paging

               self.doPaging(self.pageSize(),
self.currPage());

            };
```

```
            self.doPaging = function(pageSize,
nextPage) {

                  var storesArray =
ko.observableArray([]);

                  //make a copy of the results

                  if(self.searchString() != '') {


         storesArray(self.searchResultsArray.slice());

                  }

                  else {


         storesArray(self.storesCopy().slice());

                  }



                  //clear items

                  self.stores.removeAll();



                  //set current page as next page if
nextPage is defined

                  if(nextPage)

                           self.currPage(nextPage);



                  //set page size

                  self.pageSize(pageSize);



                  //calculate number of pages


self.NumberPages(Math.ceil(storesArray().length/self.pag
eSize()));



                 //clear pages array

                 self.pagesArray.removeAll();



                 //populate pagesArray


                 for(var i = 0; i < self.NumberPages();
i++) {

                   self.pagesArray.push({

                            pageNumber:
ko.observable((i+1))

                   });

                 }



                 //set max number of pages


self.maxNumberPages(self.NumberPages());



                 //if current page is greater than max
number of pages, set currPage = maxNumberPages

                 if(self.currPage() >
self.maxNumberPages())

                   self.currPage(self.maxNumberPages());



                 //if maxNumberPages is less than 1,
set currPage to 1

                 if(self.maxNumberPages() < 1)

                   self.currPage(1);



                 var startIndex = (self.currPage()-
1)*self.pageSize();

                   for(var i = startIndex; i <
(self.pageSize() + startIndex); i++) {

                            if(storesArray()[i]) {


         self.stores.push(storesArray()[i]);

                            }

                            else {

                                     break;

                            }


                 }
```

```
            };


        };

        ko.applyBindings(new StoresViewModel());

});
```

users.js

```
/**
 * This is the Jquery file for the users.jsp page
 */
if(Cookies.get('usertype') == 3) {

        //redirect to home page

        window.location.replace('home');

}

$(function() {

        /**
         * Start of knockout.js
         *
         */

        function User(user) {

            this.userId = ko.observable(user.userID);

            this.userName =
ko.observable(user.userName);

            this.branchName =
ko.observable(user.store.branchName);

            this.usertype =
ko.observable(user.usertype.usertypeName);

            this.email = ko.observable(user.email);

            this.contactNo =
ko.observable(user.contactNo);

            this.isActive =
ko.observable(user.active);

            this.enableDisable =
ko.observable(((user.active==0)?"Enable":"Disable"));

            this.enableDisableTitle =
ko.observable(((this.isActive()==0)?"Enable":"Disable")
+ " user " + user.userID);

            this.editUserTitle = ko.observable("Edit
user " + user.userID);

            this.editPassTitle = ko.observable("Change
user " + user.userID + "'s password");

        }


        function Usertype(usertype) {

                this.usertypeId =
ko.observable(usertype.usertypeId);

                this.usertypeName =
ko.observable(usertype.usertypeName);

        }


        function Area(area) {

                this.areaId =
ko.observable(area.areaId);

                this.areaName =
ko.observable(area.areaName);

        }


        function Store(store) {

            this.storeId =
ko.observable(store.storeId);

            this.storeAreaId =
ko.observable(store.area.areaId);

            this.storeBranchName =
ko.observable(store.branchName);

        }



        // Overall viewmodel for this screen, along
with initial state

        function UsersViewModel() {

                var self = this;
```

```
                    //used in cookies

                    self.loggedInUser =
ko.observable(Cookies.get('username'));

                    self.loggedInUsertype =
ko.observable(Cookies.get('usertype'));

                    self.loggedInUserStoreId =
ko.observable(Cookies.get('userStoreId'));

                    self.loggedInUserAreaId =
ko.observable(Cookies.get('userAreaId'));



                    // list of users

                    self.users = ko.observableArray([]);

                    self.usersCopy =
ko.observableArray([]);



                    // list of usertypes

                    self.usertypes =
ko.observableArray([]);

                    self.usertypesForDisplay =
ko.observableArray([]);



                    // list of areas

                    self.areas = ko.observableArray([]);



                    // list of stores

                    self.stores = ko.observableArray([]);

                    self.storesCopy =
ko.observableArray([]);



                    //for paging

               self.pageSize = ko.observable(10);

               self.NumberPages = ko.observable(1);

               self.currPage = ko.observable(1);

               self.pagesArray = ko.observableArray([]);

               self.maxNumberPages = ko.observable(1);



               //These are used in search

                    self.searchString =
ko.observable("");

                    self.searchResultsArray =
ko.observableArray([]);




                    //used in form

                    self.userid = ko.observable(0);

                    self.username = ko.observable();

                    self.userpassword1 = ko.observable();

                    self.userpassword2 = ko.observable();

                    self.usertypeid = ko.observable();

                    self.areaid = ko.observable();

                    self.storeid = ko.observable();

                    self.passwordsEqual =
ko.observable(false);

                    self.emailAd = ko.observable();

                    self.emailValid =
ko.observable(false);

                    self.contactNum = ko.observable();

                    self.isactive = ko.observable();



                    //used in change password form

                    self.user_id = ko.observable(0);

                    self.userpassword_1 =
ko.observable();

                    self.userpassword_2 =
ko.observable();



                    //get users based on user type and
storeId

               self.getUsers = function() {
```

```
                var url = 'get-users';
                        $.ajax({
                                url: url,
                                dataType: 'json',
                                data: {usertypeId:
self.loggedInUsertype(), storeId:
self.loggedInUserStoreId()},
                                success:
function(allData) {
                               var mappedItems =
$.map(allData, function(user) { return new User(user);
});

                                //make a copy of the
users

self.usersCopy(mappedItems);

                                //Do paging on first
load

self.doPaging(self.pageSize());
                                }
                        });
            };

                // get the list of users on first
load from DB
                self.getUsers();

            // get the list of usertypes on first load
from DB
            $.getJSON("get-usertypes",
function(allData) {
                var mappedItems = $.map(allData,
function(usertype) { return new Usertype(usertype); });
                //fill the usertypes array
                self.usertypes(mappedItems);

                //display only applicable usertypes

self.processUsertypes(self.usertypes());
            });

            //get the list of areas on first load from
DB
            $.getJSON("get-areas", function(allData) {
                var mappedItems = $.map(allData,
function(area) { return new Area(area); });
                //fill areas array
                self.areas(mappedItems);
            });

            //get the list of stores on first load
from DB
            $.getJSON("get-stores", function(allData)
{
                var mappedItems = $.map(allData,
function(store) { return new Store(store); });
                //fill stores array
                self.stores(mappedItems);
                //have a copy

self.storesCopy(self.stores().slice());
            });

            //process user types to be displayed
            self.processUsertypes =
function(usertypes) {
                //for Proprietors, display only
Proprietor and Store Manager
```

```
                if(self.loggedInUsertype()==1) {
                        for(var x in usertypes) {
                                if
((usertypes[x].usertypeId() == 1) ||
(usertypes[x].usertypeId() == 2)) {

        self.usertypesForDisplay.push(usertypes[x]);
                                }
                        }
                }
                //for Store Managers, display only
Store cashier and Store Staff
                else if(self.loggedInUsertype()==2) {
                        for(var x in usertypes) {
                                if
((usertypes[x].usertypeId() == 3) ||
(usertypes[x].usertypeId() == 4)) {

        self.usertypesForDisplay.push(usertypes[x]);
                                }
                        }
                }
            };

            self.enableDisableUser = function(user,
event) {
                // get userId of the row
                        var userId =
event.currentTarget.id;
                        var isActive =
event.currentTarget.name;

                        bootbox.confirm({
                                message: "Are you
sure you want to " + (isActive==0?"enable":"disable") +
" user " + userId + "?",
                                closeButton:
false,
                                size: "small",
                            callback:
function(result){
                                if(result) {

        var url = 'enable-disable-user';

        $.ajax({

        url: url,

        data: {userId: userId, isActive: isActive},

        success: function(user) {


                //loop through the items and update
the value
                for(var i = 0; i<self.users().length;
i++) {
                        if (self.users()[i].userId()
== userId) {

        self.users()[i].isActive(isActive==0?1:0);

        self.users()[i].enableDisable(isActive==0?"Dis
able":"Enable");

        self.users()[i].enableDisableTitle(((isActive=
=0)?"Disable":"Enable") + " user " + userId);

break;
```

```
}

                    }


                    //loop through the items copy and
update the value

                    for(var i = 0;
i<self.usersCopy().length; i++) {

                              if
(self.usersCopy()[i].userId() == userId) {

          self.users()[i].isActive(isActive==0?1:0);


          self.users()[i].enableDisable(isActive==0?"Dis
able":"Enable");


          self.users()[i].enableDisableTitle(((isActive=
=0)?"Disable":"Enable") + " user " + userId);

break;

}

                    }


                    //Do paging

self.doPaging(self.pageSize(), self.currPage());


                    $.notify({

                              // options

                              icon: 'glyphicon glyphicon-
ok',

                              message: 'User successfully
' + (isActive==0?"enabled":"disabled")

                    },{

                              // settings

                              type: 'success',

                              delay: 1000,

                              offset: 55,

                    });

          },

          error: function(jqXHR, textStatus,
errorThrown) {

               alert("error:" + textStatus + "
exception:" + errorThrown);

          },

          });

                                                  }

                              }

                    });

               };


               self.cancelEdit = function() {

                    //Clear the values of the form
```

```
                self.userid(0);
                        self.username("");
                        self.userpassword1("");
                        self.userpassword2("");
                        self.usertypeid("");
                        self.areaid("");
                        self.storeid("");
                        self.emailAd("");
                        self.isactive("");
                        self.contactNum("");

                        //change password form
                        self.userpassword_1("");
                        self.userpassword_2("");

                        //empty error
        $("#editUserError").html("");
            };

            self.displayAddUser = function() {
                //change the modal title and button
text
                $(".modal-title").text("Add new
user");
        $("#editUserSubmitButton").text("Add");

                        //Clear the values inside
the form
                        self.userid(0);
                        self.username("");
                        self.userpassword1("");
                        self.userpassword2("");
                        self.usertypeid("");
        self.areaid(self.loggedInUserAreaId());
        self.storeid(self.loggedInUserStoreId());
                        self.emailAd("");
                        self.isactive("");
                        self.contactNum("");

                        //empty error
        $("#editUserError").html("");

                        //show the Edit modal
        $('#editUserModal').modal('show');
            };

            self.displayUser = function() {

                var userId = this.userId();
                $.ajax({
                                url: 'get-
'+userId+'-user',
                                dataType: 'json',
                                success:
function(user) {
                                        //map the
values to modal form
        self.userid(user.userID);
        self.username(user.userName);
        self.usertypeid(user.usertype.usertypeId);
```

```
            self.areaid(user.store.area.areaId);

            self.storeid(user.store.storeId);

            self.isactive(user.active);

            self.emailAd(user.email);

            self.contactNum(user.contactNo);
                                    }

                });

                //change the modal title and button
text

                $(".modal-title").text("Edit user");

$("#editUserSubmitButton").text("Update");


                        //show the modal

$('#editUserModal').modal('show');

        };


        self.displayChangePassword = function() {

                self.user_id(this.userId());


                //empty error

$("#changePasswordError").html("");


                        //show the Edit modal

$('#changePasswordModal').modal('show');


        };


self.areaid.subscribe(function(newAreaIdValue) {

                        //clear stores array

                        self.stores.removeAll();

                        //update the selection in store
branches

                        for(var x in self.storesCopy()) {

                                if
((self.storesCopy()[x].storeAreaId() == newAreaIdValue))
{


                self.stores.push(self.storesCopy()[x]);

                                }

                        }

                });



self.emailAd.subscribe(function(newEmailAdValue) {

                        var pattern = /^([a-z\d!#$%&'*+\-
\/=?^_`{|}~\u00A0-\uD7FF\uF900-\uFDCF\uFDF0-
\uFFEF]+(\.[a-z\d!#$%&'*+\-\/=?^_`{|}~\u00A0-
\uD7FF\uF900-\uFDCF\uFDF0-\uFFEF]+)*|"((([ \t]*\r\n)?[
\t]+)?([\x01-\x08\x0b\x0c\x0e-\x1f\x7f\x21\x23-\x5b\x5d-
\x7e\u00A0-\uD7FF\uF900-\uFDCF\uFDF0-\uFFEF]|\\[\x01-
\x09\x0b\x0c\x0d-\x7f\u00A0-\uD7FF\uF900-\uFDCF\uFDF0-
\uFFEF]))*(([ \t]*\r\n)?[ \t]+)?")@(([a-z\d\u00A0-
\uD7FF\uF900-\uFDCF\uFDF0-\uFFEF]|[a-z\d\u00A0-
\uD7FF\uF900-\uFDCF\uFDF0-\uFFEF][a-z\d\-._~\u00A0-
\uD7FF\uF900-\uFDCF\uFDF0-\uFFEF]*[a-z\d\u00A0-
\uD7FF\uF900-\uFDCF\uFDF0-\uFFEF])\.)+([a-z\u00A0-
\uD7FF\uF900-\uFDCF\uFDF0-\uFFEF]|[a-z\u00A0-
\uD7FF\uF900-\uFDCF\uFDF0-\uFFEF][a-z\d\-._~\u00A0-
\uD7FF\uF900-\uFDCF\uFDF0-\uFFEF]*[a-z\u00A0-
\uD7FF\uF900-\uFDCF\uFDF0-\uFFEF])\.?$/i;


                self.emailValid(pattern.test(newEmailAdValue))
;

                        if(!self.emailValid()) {

                                var errorMessage = "Invalid
email address.";

                                        //Display error
message
```

```
        $("#editUserError").html(errorMessage);
                }
                else {

        $("#editUserError").html("");
                }


            });


            self.checkPasswords = function() {
                if(self.userpassword1() !=
self.userpassword2()) {
                        var errorMessage =
"Passwords do not match.";
                                //Display error
message

        $("#editUserError").html(errorMessage);

        self.passwordsEqual(false);
                }
                else {

        $("#editUserError").html("");
                        self.passwordsEqual(true);
                }
            };


            self.checkChangePasswords = function() {
                if(self.userpassword_1() !=
self.userpassword_2()) {
                        return false;
                }
                return true;
            };

            self.editUser = function() {
                var userId = this.userid();
                var userName = this.username();
                var userPassword =
this.userpassword1();
                var usertype = this.usertypeid();
                var storeId = this.storeid();
                var email = this.emailAd();
                var contactNo = this.contactNum();
                var isActive = this.isactive();


                //Add new item
                if(userId==0) {
                        var url = 'add-user';
                                $.ajax({
                                        url: url,
                                        dataType:
'json',
                                        data:
{userName: userName, userPassword: userPassword,
usertype: usertype,

        storeId: storeId, email: email, contactNo:
contactNo},
                                        success:
function(user) {

        //close the modal

        $('[data-dismiss=modal]').click();



        var branchName = "";

        var usertypeName = "";
```

```
          //get branch name

          for(var x in self.storesCopy()) {

          if(self.storesCopy()[x].storeId() ==
user.store.storeId) {

                    branchName =
self.storesCopy()[x].storeBranchName();

                    break;

          }

          }

          //get usertype

          for(var x in self.usertypes()) {

          if(self.usertypes()[x].usertypeId() ==
user.usertype.usertypeId) {

                    usertypeName =
self.usertypes()[x].usertypeName();

                    break;

          }

          }



          if(self.searchString() != '') {

          self.searchResultsArray.push({

                    userId : ko.observable(user.userID),

                    userName : ko.observable(user.userName),

                    branchName : ko.observable(branchName),

                    usertype : ko.observable(usertypeName),

                    email : ko.observable(user.email),

                    contactNo : ko.observable(user.contactNo),

                    isActive: ko.observable(1),

                    enableDisable : ko.observable("Disable"),

                    enableDisableTitle :
ko.observable("Disable user " + user.userID),

                    editUserTitle : ko.observable("Edit user "
+ user.userID),

                    editPassTitle : ko.observable("Change user
" + user.userID + "'s password")

          });

                                                  }

          else {

          self.searchResultsArray.push({

                    userId : ko.observable(user.userID),

                    userName : ko.observable(user.userName),

                    branchName : ko.observable(branchName),

                    usertype : ko.observable(usertypeName),

                    email : ko.observable(user.email),
```

```
            contactNo : ko.observable(user.contactNo),

            isActive: ko.observable(1),

            enableDisable : ko.observable("Disable"),

            enableDisableTitle :
ko.observable("Disable user " + user.userID),

            editUserTitle : ko.observable("Edit user "
+ user.userID),

            editPassTitle : ko.observable("Change user
" + user.userID + "'s password")

        });

        }

        //add also in the copy

        self.usersCopy.push({

        userId : ko.observable(user.userID),

userName : ko.observable(user.userName),

branchName : ko.observable(branchName),

usertype : ko.observable(usertypeName),

email : ko.observable(user.email),

contactNo : ko.observable(user.contactNo),

isActive: ko.observable(1),

enableDisable : ko.observable("Disable"),

enableDisableTitle : ko.observable("Disable user " +
user.userID),

delUserTitle : ko.observable("Delete user " +
user.userID),

editUserTitle : ko.observable("Edit user " +
user.userID),

editPassTitle : ko.observable("Change user " +
user.userID + "'s password")

        });



        //Do paging

self.doPaging(self.pageSize(), self.currPage());



        //notify that adding is successful

        $.notify({

        // options

        icon: 'glyphicon glyphicon-ok',

        message: 'User successfully added'

        },{

        // settings

        type: 'success',

        delay: 1000,

        offset: 55,
```

```
        });
                                        },
                                        error:
function(jqXHR, textStatus, errorThrown) {
        errorMessage = "Cannot add user. Please check
if username " +
        "already exists. Or try again later.";
        //Display error message
        $("#editUserError").html(errorMessage);
                                        }
                                });
                }
                else {
                        var url = 'update-user';
                        $.ajax({
                                        type:
'GET',
                                        url: url,
                                        data:
{userId: userId, userName: userName, usertype: usertype,
storeId: storeId, email: email, contactNo: contactNo,
isActive: isActive},
                                        success:
function() {
        var branchName = "";
        var usertypeName = "";
        //get branch name
        for(var x in self.storesCopy()) {
        if(self.storesCopy()[x].storeId() == storeId)
{
                        branchName =
self.storesCopy()[x].storeBranchName();
                        break;
        }
        }
        //get usertype
        for(var x in self.usertypes()) {
        if(self.usertypes()[x].usertypeId() ==
usertype) {
                        usertypeName =
self.usertypes()[x].usertypeName();
                        break;
        }
        }
        //loop through the items and update the value
        for(var i = 0; i<self.users().length; i++) {
                        if (self.users()[i].userId() ==
userId) {
```

```
            self.users()[i].userName(userName);

            self.users()[i].branchName(branchName);

            self.users()[i].usertype(usertypeName);

            self.users()[i].email(email);

            self.users()[i].contactNo(contactNo);

            self.users()[i].isActive(isActive);

break;

}

            }


            //loop through the items copy and update the
value

            for(var i = 0; i<self.usersCopy().length; i++)
{

                        if (self.usersCopy()[i].userId() ==
userId) {

            self.usersCopy()[i].userName(userName);

            self.usersCopy()[i].branchName(branchName);

            self.usersCopy()[i].usertype(usertypeName);

            self.usersCopy()[i].email(email);

            self.usersCopy()[i].contactNo(contactNo);

            self.usersCopy()[i].isActive(isActive);

break;

}

            }


            //Do paging

self.doPaging(self.pageSize(), self.currPage());


$.notify({

                        // options

                        icon: 'glyphicon glyphicon-ok',

                        message: 'User successfully updated'

            },{

                        // settings

                        type: 'success',

                        delay: 1000,
```

```
                offset: 55,

        });

        $('[data-dismiss=modal]').click();

                                        },
                                        error:
function() {
        errorMessage = "Cannot edit user. Please try
again later.";
        //Display error message
        $("#editItemError").html(errorMessage);
        return false;
                                        },
                                });
                }
                return true;
            };

            self.changePassword = function() {
                var userId = this.user_id();
                    var userPassword =
this.userpassword_1();
                    var url = 'change-password';
                        $.ajax({
                                type: 'GET',
                                url: url,
                                data: {userId:
userId, userPassword: userPassword},
                                success:
function() {
                                        $.notify({
        // options
        icon: 'glyphicon glyphicon-ok',
        message: 'Password successfully changed'
        },{
        // settings
        type: 'success',
        delay: 1000,
        offset: 55,
        });
                                        $('[data-
dismiss=modal]').click();
                                },
                                error: function()
{
        errorMessage = "Cannot change password. Please
try again later.";
                                        //Display
error message
        $("#changePasswordError").html(errorMessage);
                                        return
false;
                                },
                        });
```

```
                    //alert('user id: ' + userId +
'\nuser password: ' + userPassword);
            };

            self.removeUser = function(user, event) {

                // get userId of the row
                        var userId =
event.currentTarget.id;

                        bootbox.confirm({
                                message: "You are
about to delete user " + userId + ".\nDo you want to
proceed?",
                                closeButton:
false,
                                size: "small",
                            callback:
function(result){
                                if(result) {

        var url = 'delete-' + userId + '-user';

        $.ajax({

        url: url,

        success: function() {

                //remove the element from the table

                if(self.searchString() != '') {

                        //remove from
searchResultsArray

                        for(var x in
self.searchResultsArray()) {

        if(self.searchResultsArray()[x].userId() ==
userId) {

        self.searchResultsArray.remove(self.searchResu
ltsArray()[x]);

                                                break;

                                }

                        }

        }

        else {

                self.users.remove(user);

        }

                //remove also from the copy

                for(var x in self.usersCopy()) {

        if(self.usersCopy()[x].userId() == userId) {

        self.usersCopy.remove(self.usersCopy()[x]);

                                break;

                        }

                }
```

```
                    //Do paging
self.doPaging(self.pageSize(), self.currPage());



                    $.notify({
                            // options
                            icon: 'glyphicon glyphicon-
ok',
                            message: 'User successfully
deleted'
                    },{
                            // settings
                            type: 'success',
                            delay: 1000,
                            offset: 55,
                    });
            },
            error: function(jqXHR, textStatus,
errorThrown) {
                alert("error:" + textStatus + "
exception:" + errorThrown);
            },
        });
                                                }
                                    }
                                });
                    };

                    self.searchUsers = function() {
                            var searchString =
self.searchString();
                            self.searchResultsArray.removeAll();

                            if(searchString != '') {
                                    for(var x in
self.usersCopy()) {
                                            if
((self.usersCopy()[x].userName().toLowerCase().indexOf(s
earchString.toLowerCase()) >= 0)
                                    ||
(self.usersCopy()[x].branchName().toLowerCase().indexOf(
searchString.toLowerCase()) >= 0)
                                    ||
(self.usersCopy()[x].usertype().toLowerCase().indexOf(se
archString.toLowerCase()) >= 0)
                                    ||
(self.usersCopy()[x].email().toLowerCase().indexOf(searc
hString.toLowerCase()) >= 0)
                                                ) {
                                            self.searchResultsArray.push(self.usersCopy()[
x]);
                                            }
                                    }
                            }
                            else {
```

```
        self.searchResultsArray(self.usersCopy().slice
());
                }


                //Do paging
               self.doPaging(self.pageSize(),
self.currPage());
            };
            self.doPaging = function(pageSize,
nextPage) {


                var usersArray =
ko.observableArray([]);
                //make a copy of the results
                if(self.searchString() != '') {

        usersArray(self.searchResultsArray.slice());
                }
                else {

        usersArray(self.usersCopy().slice());
                }


                //clear users
                self.users.removeAll();


                //set current page as next page if
nextPage is defined
                if(nextPage)
                        self.currPage(nextPage);


                //set page size
                self.pageSize(pageSize);


                //calculate number of pages

self.NumberPages(Math.ceil(usersArray().length/self.page
Size()));


                //clear pages array
                self.pagesArray.removeAll();


                //populate pagesArray
                for(var i = 0; i < self.NumberPages();
i++) {
                 self.pagesArray.push({
                         pageNumber:
ko.observable((i+1))
                 });
                }


                //set max number of pages

self.maxNumberPages(self.NumberPages());


                //if current page is greater than max
number of pages, set currPage = maxNumberPages
                if(self.currPage() >
self.maxNumberPages())
                 self.currPage(self.maxNumberPages());


                //if maxNumberPages is less than 1,
set currPage to 1
                if(self.maxNumberPages() < 1)
                 self.currPage(1);


                var startIndex = (self.currPage()-
1)*self.pageSize();
                 for(var i = startIndex; i <
(self.pageSize() + startIndex); i++) {
```

```
                        if(usersArray()[i]) {


        self.users.push(usersArray()[i]);

                        }

                        else {

                                break;

                        }



                }

            };



        }

        ko.applyBindings(new UsersViewModel());

});
```

main.js

```
/**
 * This Javascript file will be used in all pages
 */
if(!(Cookies.get('username') && Cookies.get('username')
!= '')) {
                        //redirect to login page is
someone is not logged in

        window.location.replace('login');
}
$(function() {
        "use strict";

        // initialize tooltip
        $('[data-toggle="tooltip"]').tooltip();

        //bind showing loading gif on ajax start
        $(document).ajaxStart(function() {
                var href = document.location.href;
                var lastPathSegment =
href.substr(href.lastIndexOf('/') + 1);
                //exclude rmc
                if (lastPathSegment!='rmc' &&
lastPathSegment!='rmc#') {
                        $.blockUI({
                                message: "<img
src='resources/img/circular-load.GIF'>",
                                baseZ: 2000,
                                css: {
                               border: 'none',
                               backgroundColor:
'transparent'
                            }
                        });
                }

        });
        $(document).ajaxStop(function() {
                $.unblockUI();
        });

        //reset modal form data on data-dismiss
        $('[data-dismiss=modal]').click(function (e) {
            var $t = $(this),
                target = $t[0].href ||
$t.data("target") || $t.parents('.modal') || [];
              $(target)
                .find("input,textarea,select")
                   .val('')
                   .end()
                .find("input[type=checkbox],
input[type=radio]")
                   .prop("checked", "")
                   .end()
                .find("div")
                     .removeClass("has-error")
                     .end()
                .find(".error")
                     .empty()
                     .end();
        });

        //listen when the link is clicked, clear all
the cookies
        $("#logoutLink").click(function() {
                //alert('im clicked');
                Cookies.set('username', '');
                Cookies.set('usertype', '');
                //Cookies.set('username', '');
                //redirect to login page
                window.location.replace('login');
        });

        var usertype = Cookies.get('usertype');
        if(usertype==1) {
                $(".storesMenu").css('display',
'inline');
                $(".reportsMenu").css('display',
'inline');
        } else {
                $(".storesMenu").css('display',
'none');
                $(".reportsMenu").css('display',
'none');
        }
        if(usertype==2 || usertype==3) {
                $(".posMenu").css('display',
'inline');
        } else {
                $(".posMenu").css('display', 'none');
        }
        if(usertype==1) {
                $(".itemsMenu").css('display',
'inline');
        } else {
                $(".itemsMenu").css('display',
'none');
        }
        if(usertype==1 || usertype==2) {
                $(".usersMenu").css('display',
'inline');
        } else {
                $(".usersMenu").css('display',
'none');
        }
});
```

checkout.jsp

```
<%@ page language="java" contentType="text/html;
charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01
Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
        <jsp:include page="header.jsp" />
```

```
        <!-- Javascript resource for this particular
page -->
        <script
src="resources/js/plugin/anysearch.min.js"></script>
        <script
src="resources/js/pages/checkout.js"></script>


</head>
<body class="container">
  <div class="page-header">
    <h1>Checkout items</h1>
  </div>
  <div id="rightDiv" class="col-xs-6 col-sm-6 col-md-6
col-lg-6 col-xl-6">
        <div class="row">
       <div class="col-md-4 col-sm-4 col-lg-4 col-xs-
6">
           <div id="custom-search-input">
               <div class="input-group col-md-12">
                   <input data-bind="value:
searchString, valueUpdate:'keyup', event: { keyup:
searchItems }" type="text" class="form-control input-md"
placeholder="Barcode, Item name" />
                   <span class="input-group-btn">
                       <i class="glyphicon glyphicon-
search"></i>
                   </span>
               </div>
           </div>
       </div>
       <div class="col-md-8 col-sm-8 col-lg-8 col-xs-
6">
           <button type="button" class="btn btn-primary
btn-md" id="purchaseItemButton" data-bind="click:
addToCart">
                                   <span
class="glyphicon glyphicon-plus"></span> Add to cart
                           </button>
       </div>
        </div>
        <div class"table-responsive">
        <table id="itemsTable" class="table table-
striped">
        <thead>
        <tr>
                 <th></th>
        <th>Bar Code</th>
        <th>Item Name</th>
        <th>Price</th>
        <th>Quantity</th>
        <th>Discount (%)</th>
        </tr>
        </thead>
        <tbody data-bind="foreach: inventory">
                 <td><input type="checkbox" data-
bind="checked: itemSelected"/></td>
                 <td data-bind="text: barCode"></td>
           <td data-bind="text: itemName"></td>
           <td data-bind="text: itemPrice"></td>
           <td><input class="inputReceipt" type="text"
data-bind="value: itemQuantity, enable: itemSelected(),
valueUpdate:'afterkeydown'"/></td>
           <td><input class="inputReceipt" type="text"
data-bind="value: itemDiscount, enable: itemSelected(),
valueUpdate:'afterkeydown'"/></td>
        </tbody>
        <tbody data-bind="visible: inventory().length
== 0">
                 <td></td>
           <td></td>
           <td></td>
           <td>No inventory to show</td>
           <td></td>
           <td></td>
           <td></td>
        </tbody>
        </table>
  </div>
  <div class="text-center col-md-4 col-md-offset-4">
        <ul class="pagination pagination-md">
                 <li data-bind="css: { disabled:
(currPage() === 1) }"><a data-bind="css: { disabled:
(currPage() === 1) }, click: function() { if(currPage()
!= 1) { doPaging(pageSize(), currPage()-1)}}"
href="">&laquo;</a></li>
        </ul>
    <ul data-bind="foreach: pagesArray"
class="pagination pagination-md">
                 <li data-bind="css: { active:
$parent.currPage() === pageNumber()}"><a data-
bind="text: pageNumber, click: function() {
$parent.doPaging($parent.pageSize(), pageNumber())}"
href=""></a></li>
        </ul>
        <ul class="pagination pagination-md">
                 <li data-bind="css: { disabled:
(currPage() === maxNumberPages() || maxNumberPages() ===
0) }"><a data-bind="click: function() { if(currPage() !=
maxNumberPages()) { doPaging(pageSize(),
currPage()+1)}}" href="">&raquo;</a></li>
        </ul>
  </div>
</div>
  <div id="leftDiv" class="col-xs-6 col-sm-6 col-md-6
col-lg-6 col-xl-6 posReceipt">
    <div class="right">
       <button type="button" class="btn btn-primary btn-
md" id="checkoutButton" data-bind="click: doPayment,
enable: transactionItems().length>0">
                           <span class="glyphicon
glyphicon-shopping-cart"></span> Checkout
                 </button>
    </div>
    <div id="receiptDiv">
        <p data-bind="text: storeBranchName"></p>
        <p data-bind="text: storeAddress"></p>
        <p data-bind="text: storeTin"></p>
        <hr />
        <table id="itemsTable" class="table">
          <thead>
                 <tr>
                   <th>Qty x Price</th>
          <th>Item</th>
          <th>Discount</th>
          <th>Total</th>
          <th></th>
                 </tr>
         </thead>
          <tbody data-bind="foreach:
transactionItems">
            <td data-bind="text:
itemQuantityXPrice"></td>
           <td data-bind="text: itemDescription"></td>
           <td data-bind="text: itemDiscountText"></td>
           <td data-bind="text:
itemPriceAfterDiscount"></td>
           <td><a data-bind="tooltip: {title: 'Remove
from cart', placement: 'left' }, attr: {id:
inventoryId}, click: $parent.removeFromCart" href=""
data-toggle="tooltip" data-placement="top">x</a></td>
          </tbody>
        </table>
        <hr />
        <table class="table table-borderless">
                 <tr>
                           <td><label class="left">Net
amount due:</label></td>
                           <td class="right" data-
bind="text: netAmountDue"></td>
                 </tr>
                 <tr>
                           <td><label
class="left">Cashier:</label></td>
                           <td class="right" data-
bind="text: loggedInUser"></td>
                 </tr>
                 <tr>
```

```
                        <td><label
class="left">Total items:</label></td>
                        <td class="right" data-
bind="text: totalItems"></td>
                    </tr>
            </table>
            <hr />
            <table class="table table-borderless">
                    <tr>
                        <td><label
class="left">Vatable Sale:</label></td>
                        <td class="right" data-
bind="text: vatableSale"></td>
                    </tr>
                    <tr>
                        <td><label
class="left">VAT(12%):</label></td>
                        <td class="right" data-
bind="text: vat"></td>
                    </tr>
            </table>
            <hr />
            <p>Thank you for visiting us.</p>
            <p>Please come again.</p>
        </div>
    </div>
<!-- Add-to-cart modal -->
<div id="addToCartModal" class="modal fade"
role="dialog">
  <div class="modal-dialog">
    <!-- Modal content -->
    <div class="modal-content">
      <div class="modal-header">
          <h4 class="modal-title">Add to cart</h4>
      </div>
      <div class="modal-body">
        <form>
          <input data-bind="value: inventoryid"
type="hidden"/>
                    <div class="form-group input-group">
                        <span class="input-group-
addon">Bar code</span>
                        <input data-bind="value:
barcode, valueUpdate:'afterkeydown', enable: false"
type="text" class="form-control" placeholder="Bar code"
required="" maxlength="45"/>
                    </div>
                    <div class="form-group input-group">
                        <span class="input-group-
addon">Item</span>
                        <input data-bind="value:
itemdescription, valueUpdate:'afterkeydown', enable:
false" type="text" class="form-control"
placeholder="Item" required="" maxlength="160"/>
                    </div>
                    <div class="form-group input-group">
                        <span class="input-group-
addon">Price</span>
                        <input data-bind="value:
itemprice, valueUpdate:'afterkeydown', enable: false"
type="text" class="form-control"
placeholder="Description"/>
                    </div>
                    <div class="form-group input-group">
                        <span class="input-group-
addon">Quantity*</span>
                        <input data-bind="value:
itemquantity, valueUpdate:'afterkeydown'" type="text"
class="form-control" placeholder="Price"
maxlength="10"/>
                    </div>
                    <div class="form-group input-group">
                        <span class="input-group-
addon">Discount*</span>
                        <input data-bind="value:
itemdiscount, valueUpdate:'afterkeydown'" type="text"
class="form-control" placeholder="Discount"
maxlength="10"/>
                    </div>
                    <div class="error"
id="addToCartError">
                    </div>
                    <div class="modal-footer">
                    <button type="button" class="btn btn-
default" data-dismiss="modal" data-bind="click:
cancelEdit, enable: true">Cancel</button>
                  <button type="button"
id="editItemSumbitButton" class="btn btn-primary" data-
bind="click: addToCartThruBarcode, enable:
(itemdescription().length > 0 && barcode().length > 9 &&
itemprice() > 0)">Add</button>
                    </div>
                </form>
      </div>
    </div>
  </div>
</div>
<!-- Checkout items modal -->
<div id="paymentModal" class="modal fade" role="dialog">
  <div class="modal-dialog">
    <!-- Modal content -->
    <div class="modal-content">
      <div class="modal-header">
          <span><h4 class="modal-title">Checkout
items</h4>
                        <button data-bind="tooltip:
{title: 'Add payment type', placement: 'right' }, click:
addPaymentMethod" type="button" style="position-right:
0px;" class="btn btn-primary btn-md">
                                                <span
class="glyphicon glyphicon-plus"></span> Add payment
method
                        </button>
                    </span>
      </div>
      <div class="modal-body">
        <form>
                    <div class="form-group input-group">
                        <span class="input-group-
addon">Amount due</span>
                        <input data-bind="value:
amountDue, enable: false" type="text" class="form-
control" placeholder="Amount due" required=""
maxlength="45"/>
                    </div>
                    <table data-bind="visible:
returnedItemVouchers().length > 0"
id="returnItemsVoucherTable" class="table">
                      <thead>
                        <tr>
                          <th>Voucher Number</th>
                      <th>Amount</th>
                      <th></th>
                        </tr>
                      </thead>
                      <tbody data-bind="foreach:
returnedItemVouchers">
                        <td data-bind="text:
returnedItemVoucherNumber"></td>
                        <td data-bind="text:
returnedItemVoucherAmount"></td>
                        <td><a data-bind="tooltip: {title:
'Remove voucher', placement: 'left' }, attr: {id:
returnedItemVoucherNumber}, click:
$parent.removeVoucher" href="" data-toggle="tooltip"
data-placement="top">x</a></td>
                      </tbody>
                    </table>
                    <table data-bind="visible:
paymentMethods().length > 0"
id="returnItemsVoucherTable" class="table">
                      <thead>
                        <tr>
                          <th>Payment type</th>
                      <th>Reference number</th>
                      <th>Amount</th>
                      <th></th>
                        </tr>
```

```
                    </thead>
                     <tbody data-bind="foreach:
paymentMethods">
                       <td data-bind="text:
pmPaymentType"></td>
                       <td data-bind="text:
pmPaymentReferenceId"></td>
                       <td data-bind="text:
pmAmountPaid"></td>
                       <td><a data-bind="tooltip: {title:
'Remove payment', placement: 'left' }, attr: {id: pmId},
click: $parent.removePayment" href="" data-
toggle="tooltip" data-placement="top">x</a></td>
                     </tbody>
                   </table>
                   <div class="form-group input-group">
                            <span class="input-group-
addon">Total payment</span>
                            <input data-bind="value:
totalAmountPaid, event: { keyup:
computeTotalAmountPaidAndChange}, disable: true"
type="text" class="form-control" placeholder="Amount
paid" required="" maxlength="160"/>
                   </div>
                   <div class="form-group input-group">
                            <span class="input-group-
addon">Change</span>
                            <input data-bind="value:
amountChange, enable: false" type="text" class="form-
control" placeholder="Change"/>
                   </div>
                   <div class="error"
id="checkoutError">
                   </div>
                   <div class="modal-footer">
                   <button type="button" class="btn btn-
default" data-dismiss="modal" data-bind="click:
cancelEdit, enable: true">Cancel</button>
                 <button type="button"
id="checkoutSubmitButton" class="btn btn-primary" data-
bind="enable: amountChange() >= 0 && totalAmountPaid() >
0, click: checkoutItems">Check out</button>
                   </div>
               </form>
      </div>
    </div>
  </div>
</div>
<!-- Add returned item(s) voucher modal -->
<div id="returnedItemVoucherModal" class="modal fade"
role="dialog">
  <div class="modal-dialog">
    <!-- Modal content -->
    <div class="modal-content">
      <div class="modal-header">
          <h4 class="modal-title">Add payment
method</h4>
      </div>
      <div class="modal-body">
        <form>
          <div class="form-group input-group">
                            <span class="input-group-
addon">Payment type*</span>
                            <select class="form-control"
data-bind="options: paymentTypes,
                       optionsText: 'paymentType',
                       optionsValue: 'paymentTypeId',
                       value: paymentTypeOptionsId,
                       optionsCaption: 'Payment type'
                       ">
                 </select>
                   </div>
                   <div class="form-group input-group"
data-bind="visible: paymentTypeOptionsId() &&
paymentTypeOptionsId()==4">
                            <span class="input-group-
addon">Voucher number*</span>
                            <input data-bind="value:
voucherNumber, valueUpdate:'afterkeydown'" type="text"
class="form-control" placeholder="Voucher number"
required="" maxlength="45"/>
                   </div>
                   <div class="form-group input-group"
data-bind="visible: paymentTypeOptionsId() &&
(paymentTypeOptionsId() == 2 || paymentTypeOptionsId()
== 3)">
                            <span class="input-group-
addon">Payment Reference Id*</span>
                            <input data-bind="value:
paymentReferenceId, valueUpdate:'afterkeydown'"
type="text" class="form-control" placeholder="Reference
Id" required="" maxlength="160"/>
                   </div>
                   <div class="form-group input-group"
data-bind="visible: paymentTypeOptionsId() &&
(paymentTypeOptionsId() < 4)">
                            <span class="input-group-
addon">Amount*</span>
                            <input data-bind="value:
amountPaid, valueUpdate:'afterkeydown'" type="text"
class="form-control" placeholder="Amount paid"
required="" maxlength="160"/>
                   </div>
                   <div class="error"
id="returnedItemVoucherError">
                   </div>
                   <div class="modal-footer">
                   <button type="button" class="btn btn-
default" id="paymentMethodModal" data-dismiss="modal"
data-bind="click: cancelAddPaymentMethod, enable:
true">Cancel</button>
                 <button type="button"
id="addPaymentSubmitButton" class="btn btn-primary"
data-bind="enable: paymentTypeOptionsId() &&
((paymentTypeOptionsId() == 1 && amountPaid() &&
amountPaid() != '' && amountPaid() > 0) ||
((paymentTypeOptionsId() == 2 || paymentTypeOptionsId()
== 3) && paymentReferenceId() && paymentReferenceId() !=
'' && amountPaid() && amountPaid() > 0) ||
(paymentTypeOptionsId() == 4 && voucherNumber() != '' &&
voucherNumber() >= 0)), click: addPayment">Add
payment</button>
                   </div>
               </form>
      </div>
    </div>
  </div>
</div>
</body>
</html>
```

header.jsp

```
<%@ page language="java" contentType="text/html;
charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<%@ taglib prefix="c"
uri="http://java.sun.com/jsp/jstl/core" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01
Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
        <meta http-equiv="Content-Type"
content="text/html; charset=ISO-8859-1">
    <title>R.O.X. - Recreational Outdoor eXchange
System</title>

    <!-- favicon -->
    <link rel="shortcut icon" type="image/x-icon"
href="resources/images/favicon.ico"/>

    <!-- JQuery -->
    <script src="resources/js/jquery/jquery-
2.1.4.min.js"></script>

     <!-- Moment -->
```

```
    <script
src="resources/js/plugin/moment.js"></script>

     <!-- Timepicker -->
    <script src="resources/js/plugin/bootstrap-
datetimepicker.js"></script>

    <!-- Bootstrap -->
    <link href="resources/css/bootstrap.min.css"
rel="stylesheet">
    <script src="resources/js/plugin/bootstrap-3.3.6-
dist/bootstrap.min.js"></script>

    <!-- Bootbox: Used for Alerts -->
    <script
src="resources/js/plugin/bootbox.min.js"></script>

    <!-- Bootstrap notify: Used for Awesome
notifications -->
    <script src="resources/js/plugin/bootstrap-
notify.min.js"></script>

    <!-- Jquery Block UI -->
    <script
src="resources/js/plugin/jquery.blockUI.js"></script>

    <!-- Knockout.js -->
    <script src="resources/js/plugin/knockout-
3.4.0.js"></script>
    <script
src="resources/js/plugin/knockstrap.min.js"></script>

    <!-- JS cookie -->
    <script
src="resources/js/plugin/js.cookie.js"></script>

    <!-- Data table -->
    <script
src="resources/js/plugin/jQuery.dataTables.min.js"></scr
ipt>

    <!-- Date timepicker CSS -->
    <link href="resources/css/bootstrap-
datetimepicker.min.css" rel="stylesheet">

    <!-- custom resources -->
    <link href="resources/css/main.css"
rel="stylesheet">
    <script src="resources/js/main.js"></script>

</head>
<body class="col-sm-12 col-md-12 col-lg-12">
         <nav class="navbar navbar-default">
         <!-- Collect the nav links, forms, and other
content for toggling -->
         <div class="collapse navbar-collapse" id="bs-
example-navbar-collapse-1">
                  <ul class="nav navbar-nav">
                  <li><a href="<c:url value='/home'
/>"><img src='resources/img/rox.png'></img></a></li>
                  <li class="topmargin storesMenu"><a
href="<c:url value='/stores' />">Stores</a></li>
                  <li class="dropdown topmargin
reportsMenu">
                                       <a href="#"
class="dropdown-toggle" data-toggle="dropdown"
role="button" aria-haspopup="true" aria-
expanded="false">Reports<span class="caret"></span></a>
                                                <ul
class="dropdown-menu">
                                                <li
class="topmargin"><a href="<c:url value='/rmc' />">Real-
time Monitoring</a></li>
                                                <li><a
href="<c:url value='/reports' />">Reporting</a></li>
                                                </ul>
                  </li>
                  <li class="dropdown topmargin
posMenu">
                                       <a href="#"
class="dropdown-toggle" data-toggle="dropdown"
role="button" aria-haspopup="true" aria-
expanded="false">Point of Sales<span
class="caret"></span></a>
                                                <ul
class="dropdown-menu">
                                                <li><a
href="<c:url value='/checkout' />">Checkout</a></li>
                                                <li><a
href="<c:url value='/return' />">Return items</a></li>
                                                </ul>
                  </li>
                  <li class="dropdown topmargin">
                                       <a href="#"
class="dropdown-toggle" data-toggle="dropdown"
role="button" aria-haspopup="true" aria-
expanded="false">ROX Items<span
class="caret"></span></a>
                                                <ul
class="dropdown-menu">
                                                <li
class="itemsMenu"><a href="<c:url value='/items'
/>">Store items</a></li>
                                                <li><a
href="<c:url value='/inventory' />">Store Items
inventory</a></li>
                                                </ul>
                  </li>
                  <!-- <li class="dropdown topmargin">
                                       <a href="#"
class="dropdown-toggle" data-toggle="dropdown"
role="button" aria-haspopup="true" aria-
expanded="false">Staff<span class="caret"></span></a>
                                                <ul
class="dropdown-menu">
                                                <li><a
href="<c:url value='/time' />">Time and
attendance</a></li>
                                                </ul>
                  </li> -->
                  <li class="dropdown topmargin
usersMenu">
                                       <a href="#"
class="dropdown-toggle" data-toggle="dropdown"
role="button" aria-haspopup="true" aria-
expanded="false">System<span class="caret"></span></a>
                                                <ul
class="dropdown-menu">
                                                <li><a
href="<c:url value='/users' />">System users</a></li>
                                                </ul>
                  </li>
                  </ul>
                  <ul class="nav navbar-nav navbar-
right topmargin">
                  <li id="logoutLink"><a href="<c:url
value='/logout' />">Logout</a></li>
                  </ul>
         </div><!-- /.navbar-collapse -->
         </nav>
</body>
</html>
```

home.jsp

```
<%@ page language="java" contentType="text/html;
charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<%@ taglib prefix="c"
uri="http://java.sun.com/jsp/jstl/core" %>
<html>
<head>
    <jsp:include page="header.jsp" />
</head>


<body>
```

```
</body>
</html>
```

inventory.jsp

```
<%@ page language="java" contentType="text/html;
charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<%@ taglib prefix="c"
uri="http://java.sun.com/jsp/jstl/core" %>
<%@ taglib prefix="form"
uri="http://www.springframework.org/tags/form"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01
Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
         <jsp:include page="header.jsp" />

         <!-- Javascript resource for this particular
page -->
         <script
src="resources/js/pages/inventory.js"></script>
</head>
<body class="container">
<div class="container-fluid">
         <div class="page-header">
      <h1>Inventory</h1>
         </div>
         <div class="row">
                  <div class="col-md-8 col-sm-8 col-lg-
9 col-xs-6">
                           <button type="button" data-
bind="click: displayAddInventory, enable: (storeid1() >
0 && items().length > 0 &&
currentResultStoreId()==storeid1())"  class="btn btn-
primary btn-md" id="addInventoryButton">
                                    <span
class="glyphicon glyphicon-plus"></span> Add new
inventory
                           </button>
                           <button type="button" data-
bind="click: displaySearchFromOther"  class="btn btn-
primary btn-md" id="searchFromOther">
                                    Search from other
stores
                           </button>
                  </div>
        <div class="col-md-4 col-sm-4 col-lg-3 col-xs-
6">
            <div id="custom-search-input">
                <div class="input-group col-md-12">
                    <input data-bind="value:
searchString, valueUpdate:'keyup', event: { keyup:
searchInventory }" type="text" class="form-control
input-md" placeholder="Item name" />
                    <span class="input-group-btn">
                        <i class="glyphicon glyphicon-
search"></i>
                    </span>
                </div>
            </div>
        </div>
         </div>
         <div class="row">
                  <div class="col-md-4 col-sm-6 col-lg-
3 col-xs-12">
                           <div class="form-group
input-group">
                           <span class="input-group-
addon">Store Area</span>
                           <select class="form-control"
data-bind="options: areas,
                     optionsText: 'areaName',
                     optionsValue: 'areaId',
                     value: areaid1,
                     optionsCaption: 'All areas',
                     disable: loggedInUsertype()!=1
                     ">
                  </select>
                     </div>
                     <div class="form-group input-group">
                           <span class="input-group-
addon">Store Branch</span>
                           <select class="form-control"
data-bind="options: stores,
                     optionsText: 'branchName',
                     optionsValue: 'storeId',
                     value: storeid1,
                     optionsCaption: 'All branches',
                     disable: loggedInUsertype()!=1
                     ">
                  </select>
                     </div>
                     <div class="col-md-8 col-sm-8 col-lg-
9 col-xs-6 searchInventory">
                                    <button
type="button" data-bind="click: getInventory, enable:
(storeid1() > 0 && currentResultStoreId()!=storeid1())"
class="btn btn-primary btn-md"
id="searchInventoryButton">
                                             Search
                                    </button>
                           </div>
                  </div>
         </div>
         <div data-bind="visible:
inventoryCopy().length > 0 ||
currentResultStoreId()==storeid1()" class="table-
responsive inventory">
         <table id="itemsTable" class="table table-
striped">
         <thead>
         <tr>
         <th>Inventory ID</th>
         <th>Item Name</th>
         <th>Description</th>
         <th>Bar Code</th>
         <th>Quantity</th>
         <th>Action</th>
         </tr>
         </thead>
         <tbody data-bind="foreach: inventory">
                  <td data-bind="text:
inventoryId"></td>
            <td data-bind="text: itemname"></td>
            <td data-bind="text: itemdescription"></td>
            <td data-bind="text: barCode"></td>
            <td data-bind="text: itemCount"></td>
            <td>
            <button data-bind="tooltip: {title:
editInventoryTitle, placement: 'left' }, attr: {id:
inventoryId}, click: $parent.displayInventoryQuantity,
disable: $parent.loggedInUsertype()!=1" type="button"
class="btn btn-primary btn-sm">
                                    <span
class="glyphicon glyphicon-pencil"></span> Edit
                                    </button>
            <!-- <button data-toggle="tooltip" data-
placement="top" type="button" data-bind="tooltip:
{title: delInventoryTitle, placement: 'left' }, click:
$parent.removeInventoryItem, enable: (itemCount() <= 0
&& $parent.loggedInUsertype()==1), attr: {id:
inventoryId}"  class="btn btn-primary btn-sm">
                                    <span
class="glyphicon glyphicon-remove red"></span> Remove
                                    </button>  -->
            </td>
         </tbody>
         <tbody data-bind="visible: inventory().length
== 0">
                  <td></td>
            <td></td>
            <td>No inventory to show</td>
            <td></td>
            <td></td>
```

```
        </tbody>
        </table>
  </div>
  <div data-bind="visible: inventoryCopy().length > 0 ||
currentResultStoreId()==storeid1()" class="text-center
col-md-4 col-md-offset-4 inventory">
        <ul class="pagination pagination-md">
                <li data-bind="css: { disabled:
(currPage() === 1) }"><a data-bind="css: { disabled:
(currPage() === 1) }, click: function() { if(currPage()
!= 1) { doPaging(pageSize(), currPage()-1)}}"
href="">&laquo;</a></li>
        </ul>
    <ul data-bind="foreach: pagesArray"
class="pagination pagination-md">
                <li data-bind="css: { active:
$parent.currPage() === pageNumber()}"><a data-
bind="text: pageNumber, click: function() {
$parent.doPaging($parent.pageSize(), pageNumber())}"
href=""></a></li>
        </ul>
        <ul class="pagination pagination-md">
                <li data-bind="css: { disabled:
(currPage() === maxNumberPages() || maxNumberPages() ===
0) }"><a data-bind="click: function() { if(currPage() !=
maxNumberPages()) { doPaging(pageSize(),
currPage()+1)}}" href="">&raquo;</a></li>
        </ul>
  </div>
</div>
<!-- Edit inventory modal -->
<div id="editInventoryModal" class="modal fade"
role="dialog">
  <div class="modal-dialog">
    <!-- Modal content-->
    <div class="modal-content">
      <div class="modal-header">
          <h4 class="modal-title">Add inventory</h4>
      </div>
      <div class="modal-body">
        <form>
                <div class="form-group input-group">
                        <span class="input-group-
addon">Item*</span>
                        <select class="form-control"
data-bind="options: items,
                    optionsText: 'itemName',
                    optionsValue: 'itemId',
                    value: inventory_itemId,
                    optionsCaption: 'Select item to
add'
                    ">
              </select>
                </div>
                <div class="error"
id="editInventoryError">
                </div>
                <div class="modal-footer">
                <button type="button" class="btn btn-
default" data-dismiss="modal">Cancel</button>
              <button type="button" class="btn btn-
primary" data-bind="click: addInventory, enable:
(inventory_itemId() > 0)"
id="editInventorySubmitButton">Add</button>
                </div>
            </form>
      </div>
    </div>
  </div>
</div>
<!-- Search other stores' inventory modal -->
<div id="searchFromOtherModal" class="modal fade"
role="dialog">
  <div class="modal-dialog">
    <!-- Modal content-->
    <div class="modal-content">
      <div class="modal-header">
          <h4 class="modal-title">Search Inventory of
other stores</h4>
      </div>
      <div class="modal-body">
        <form>
                <div class="form-group input-group">
                        <span class="input-group-
addon">Store area</span>
                        <select class="form-control"
data-bind="options: areas,
                    optionsText: 'areaName',
                    optionsValue: 'areaId',
                    value: areaid2,
                    optionsCaption: 'All areas',
                    ">
              </select>
                </div>
                <div class="form-group input-group">
                        <span class="input-group-
addon">Store branch</span>
                        <select class="form-control"
data-bind="options: stores2,
                    optionsText: 'branchName',
                    optionsValue: 'storeId',
                    value: storeid2,
                    optionsCaption: 'All branches',
                    ">
              </select>
                </div>
                <div class="form-group input-group">
                        <span class="input-group-
addon">Item name*</span>
                        <input data-bind="value:
searchItem, valueUpdate:'afterkeydown'" type="text"
class="form-control" placeholder="Item name"/>
                </div>
                <div class="form-group input-group">
                        <span class="input-group-
addon">Barcode</span>
                        <input data-bind="value:
searchBarcode, valueUpdate:'afterkeydown'" type="text"
class="form-control" placeholder="Barcode"/>
                </div>
                <div class="error"
id="searchFromOtherError">
                </div>
                <div class="modal-footer">
                <button type="button" data-
bind="click: cancelSearch" class="btn btn-default" data-
dismiss="modal">Cancel</button>
              <button type="button" class="btn btn-
primary" data-bind="click: searchFromOther, enable:
(searchItem().length > 0)"
id="searchFromOtherSubmitButton">Search</button>
                </div>
            </form>
      </div>
    </div>
  </div>
</div>
<!-- Edit inventory modal -->
<div id="editInventoryQuantityModal" class="modal fade"
role="dialog">
  <div class="modal-dialog">
    <!-- Modal content-->
    <div class="modal-content">
      <div class="modal-header">
          <h4 class="modal-title">Change quantity</h4>
      </div>
      <div class="modal-body">
        <form>
          <input data-bind="value: formInventoryId"
type="hidden"/ >
                <div class="form-group input-group">
                        <span class="input-group-
addon">Item name*</span>
                        <input data-bind="value:
formItemName, enable: false" type="text" class="form-
control" placeholder="Item name"/>
                </div>
                <div class="form-group input-group">
```

```
                        <span class="input-group-
addon">Quantity*</span>
                        <input data-bind="value:
formItemQuantity, valueUpdate:'afterkeydown'"
type="text" class="form-control"
placeholder="Quantity"/>
                    </div>
                    <div class="error"
id="editInventoryQuantityError">
                    </div>
                    <div class="modal-footer">
                    <button data-bind="click: cancelEdit"
type="button" class="btn btn-default" data-
dismiss="modal">Cancel</button>
                <button data-bind="click: editInventory,
enable: formItemQuantity() && formItemQuantity() >= 0"
type="button" class="btn btn-primary"
id="editInventorySubmitButton">Save</button>
                    </div>
                </form>
      </div>
    </div>
  </div>
</div>
<!-- View inventory from other stores modal -->
<div id="viewInventoryModal" class="modal fade"
role="dialog">
  <div class="modal-dialog">
    <!-- Modal content-->
    <div class="modal-content">
      <div class="modal-header">
          <h4 class="modal-title">Inventory Search
Results</h4>
      </div>
      <div class="modal-body">
        <form>
                    <div class="table-responsive
inventory">
                                        <table
id="resultsTable" class="table table-striped"
cellspacing="0" width="100%">
                                                <thead>
                                            <tr>
                                                <th>Store
brach</th>
                                                <th>Item
Name</th>

          <th>Description</th>
                                                <th>Bar
Code</th>

          <th>Quantity</th>
                                            </tr>
                                        </thead>
                                         <tbody data-
bind="foreach: searchInventoryArray">
                                                <td data-
bind="text: searchStoreBranch"></td>
                                            <td data-
bind="text: searchItemName"></td>
                                            <td data-
bind="text: searchItemDescription"></td>
                                            <td data-
bind="text: searchItemBarCode"></td>
                                            <td data-
bind="text: searchItemQuantity"></td>
                                        </tbody>
                                        </table>
                                        <div class="modal-
footer">
                        <button data-bind="click:
closeViewResult" type="button" class="btn btn-default"
data-dismiss="modal">Close</button>
                        </div>
                        </div>
                </form>
      </div>
    </div>
  </div>
</div>
</body>
</html>
```

items.jsp

```
<%@ page language="java" contentType="text/html;
charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<%@ taglib prefix="c"
uri="http://java.sun.com/jsp/jstl/core" %>
<%@ taglib prefix="form"
uri="http://www.springframework.org/tags/form"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01
Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
        <jsp:include page="header.jsp" />

        <!-- Javascript resource for this particular
page -->
        <script
src="resources/js/pages/items.js"></script>
</head>
<body class="container">
<div class="container-fluid">
        <div class="page-header">
      <h1>Store items</h1>
        </div>
        <div class="row">
                <div class="col-md-8 col-sm-8 col-lg-
9 col-xs-6">
                        <button data-bind="click:
displayAddItem" type="button" class="btn btn-primary
btn-md" id="addItemButton">
                                <span
class="glyphicon glyphicon-plus"></span> Add new item
                        </button>
                </div>
        <div class="col-md-4 col-sm-4 col-lg-3 col-xs-
6">
            <div id="custom-search-input">
                <div class="input-group col-md-12">
                    <input data-bind="value:
searchString, valueUpdate:'keyup', event: { keyup:
searchItems }" type="text" class="form-control input-md"
placeholder="Name, or description" />
                    <span class="input-group-btn">
                        <i class="glyphicon glyphicon-
search"></i>
                    </span>
                </div>
            </div>
        </div>
        </div>
  <div class"table-responsive">
        <table id="itemsTable" class="table table-
striped">
        <thead>
        <tr>
        <th>Item ID</th>
        <th>Bar Code</th>
        <th>Item Name</th>
        <th>Item Description</th>
        <th>Price (PhP)</th>
        <th>Action</th>
        </tr>
        </thead>
        <tbody data-bind="foreach: items">
                <td data-bind="text: itemId"></td>
                <td data-bind="text: barCode"></td>
          <td data-bind="text: itemname"></td>
          <td data-bind="text: itemdesc"></td>
          <td data-bind="text: price"></td>
          <td>
```

```
            <a data-bind="tooltip: {title:
editItemTitle, placement: 'left' }, attr: {id: itemId},
click: $parent.displayItem" href="" data-
toggle="tooltip" data-placement="top"><span
class="glyphicon glyphicon-pencil" aria-
hidden="true"></span></a>
            <!-- <a data-bind="tooltip: {title:
delItemTitle, placement: 'left' }, attr: {id: itemId},
click: $parent.removeItem" href="" data-toggle="tooltip"
data-placement="top"><span class="glyphicon glyphicon-
remove red" aria-hidden="true"></span></a> -->
            </td>
          </tbody>
          <tbody data-bind="visible: items().length ==
0">
                  <td></td>
            <td></td>
            <td>No item to show</td>
            <td></td>
            <td></td>
          </tbody>
          </table>
  </div>
  <div class="text-center col-md-4 col-md-offset-4">
          <ul class="pagination pagination-md">
                  <li data-bind="css: { disabled:
(currPage() === 1) }"><a data-bind="css: { disabled:
(currPage() === 1) }, click: function() { if(currPage()
!= 1) { doPaging(pageSize(), currPage()-1)}}"
href="">&laquo;</a></li>
          </ul>
    <ul data-bind="foreach: pagesArray"
class="pagination pagination-md">
                  <li data-bind="css: { active:
$parent.currPage() === pageNumber()}"><a data-
bind="text: pageNumber, click: function() {
$parent.doPaging($parent.pageSize(), pageNumber())}"
href=""></a></li>
          </ul>
          <ul class="pagination pagination-md">
                  <li data-bind="css: { disabled:
(currPage() === maxNumberPages() || maxNumberPages() ===
0) }"><a data-bind="click: function() { if(currPage() !=
maxNumberPages()) { doPaging(pageSize(),
currPage()+1)}}" href="">&raquo;</a></li>
          </ul>
  </div>
</div>
<!-- Add/Edit item modal -->
<div id="editItemModal" class="modal fade"
role="dialog">
  <div class="modal-dialog">
    <!-- Modal content-->
    <div class="modal-content">
      <div class="modal-header">
          <h4 class="modal-title">Add new item</h4>
      </div>
      <div class="modal-body">
        <form>
          <input data-bind="value: itemid"
type="hidden"/>
                  <div class="form-group input-group">
                          <span class="input-group-
addon">Name*</span>
                          <input id="itemNameInput"
data-bind="value: itemName, valueUpdate:'afterkeydown'"
type="text" class="form-control" placeholder="Item name"
required="" maxlength="30"/>
                  </div>
                  <div class="form-group input-group">
                          <span class="input-group-
addon">Bar code*</span>
                          <input data-bind="value:
barcode, valueUpdate:'afterkeydown'" type="text"
class="form-control" placeholder="Bar code" required=""
maxlength="45"/>
                  </div>
                  <div class="form-group input-group">
                          <span class="input-group-
addon">Description</span>
                          <input data-bind="value:
itemDesc, valueUpdate:'afterkeydown'" type="text"
class="form-control" placeholder="Description"
maxlength="160"/>
                  </div>
                  <div class="form-group input-group">
                          <span class="input-group-
addon">Price*</span>
                          <input data-bind="value:
itemPrice, valueUpdate:'afterkeydown'" type="text"
class="form-control" placeholder="Price"
maxlength="10"/>
                  </div>
                  <div class="error"
id="editItemError">
                  </div>
                  <div class="modal-footer">
                  <button type="button" class="btn btn-
default" data-dismiss="modal" data-bind="click:
cancelEdit, enable:true">Cancel</button>
                <button type="button"
id="editItemSumbitButton" class="btn btn-primary" data-
bind="click: editItem, enable: (itemName().length > 0 &&
barcode().length > 9 && itemPrice() > 0)">Add</button>
                  </div>
              </form>
      </div>
    </div>
  </div>
</div>
</body>
</html>
```

login.jsp

```
<%@ page language="java" contentType="text/html;
charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<%@ taglib prefix="spring"
uri="http://www.springframework.org/tags"%>
<%@ taglib prefix="form"
uri="http://www.springframework.org/tags/form"%>
<%@ taglib prefix="c"
uri="http://java.sun.com/jsp/jstl/core" %>
<html>
<head>
    <meta http-equiv="Content-Type" content="text/html;
charset=ISO-8859-1">
    <title>R.O.X. - Recreational Outdoor eXchange
System</title>

    <!-- JQuery -->
    <script src="resources/js/jquery/jquery-
2.1.4.min.js"></script>

    <!-- Bootstrap -->
    <link href="resources/css/bootstrap.min.css"
rel="stylesheet">
    <script src="resources/js/plugin/bootstrap-3.3.6-
dist/bootstrap.min.js"></script>

    <!-- Jquery Block UI -->
    <script
src="resources/js/plugin/jquery.blockUI.js"></script>

    <!-- JS cookie -->
    <script
src="resources/js/plugin/js.cookie.js"></script>

    <!-- Knockout.js -->
    <script src="resources/js/plugin/knockout-
3.4.0.js"></script>

    <!-- custom resources -->
     <link href="resources/css/main.css"
rel="stylesheet">
```

```
    <!-- <script src="resources/js/main.js"></script> -->

    <!-- Javascript resource for this particular page -->
         <script src="resources/js/pages/login.js"></script>

         <link href="resources/css/login.css" rel="stylesheet">


</head>

<body>
         <div class="container">
           <div class="row" id="pwd-container">
             <div class="col-md-4"></div>

             <div class="col-md-4">
               <section class="login-form">
                 <form method="post" action="#" role="login">
                   <img src="resources/img/rox.png" class="img-responsive" alt="" />
                   <input type="text" data-bind="value: loginUserName, valueUpdate:'keyup'" placeholder="Username" required class="form-control input-lg" />

                   <input data-bind="value: loginUserPassword, valueUpdate:'keyup'" type="password" class="form-control input-lg" id="password" placeholder="Password" required="" />


                   <div class="pwstrength_viewport_progress"></div>


                   <button data-bind="enable: loginUserName() && loginUserName().length>0 && loginUserPassword() && loginUserPassword().length>0, click: doLogin" type="submit" name="go" class="btn btn-lg btn-primary btn-block">Sign in</button>
                   <div class="error" id="loginError">
                        </div>

                 </form>
               </section>
               </div>

               <div class="col-md-4"></div>

  </div>
</body>
</html>
```

reports.jsp

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<%@ taglib prefix="form" uri="http://www.springframework.org/tags/form"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
         <jsp:include page="header.jsp" />

         <!-- Highcharts -->
         <script src="resources/js/plugin/highcharts.js"></script>
         <script src="resources/js/plugin/highcharts-more.js"></script>
         <script src="resources/js/plugin/solid-gauge.js"></script>

         <!-- Javascript resource for this particular page -->
         <script src="resources/js/pages/reports.js"></script>
</head>
<body class="container">
<div class="container-fluid">
         <div class="page-header">
      <h1>Reporting</h1>
         </div>
         <div class="row">
                  <div class="form-inline col-md-4 col-sm-6 col-lg-12 col-xs-12">
                           <div class="form-group input-group">
                           <span class="input-group-addon">Store Area</span>
                           <select class="form-control" data-bind="options: areas,
                           optionsText: 'areaName',
                           optionsValue: 'areaId',
                           value: areaid,
                           optionsCaption: 'All areas'
                      "></select>
            </div>
                           <div class="form-group input-group">
                                    <span class="input-group-addon">Store Branch</span>
                                    <select class="form-control" data-bind="options: stores,
                                optionsText: 'branchName',
                                optionsValue: 'storeId',
                                value: storeid,
                                optionsCaption: 'All branches'
                                ">
                           </select>
                  </div>
                  <div class='input-group date' id='datetimepicker1'>
                                    <span class="input-group-addon">Date From</span>
                     <input type='text' data-bind="datepicker: dateFrom" class="form-control" />
                     <span class="input-group-addon">
                         <span class="glyphicon glyphicon-calendar"></span>
                     </span>
            </div>
            <div class='input-group date' id='datetimepicker2'>
                                    <span class="input-group-addon">Date To</span>
                     <input type='text'data-bind="datepicker: dateTo" class="form-control" />
                     <span class="input-group-addon">
                         <span class="glyphicon glyphicon-calendar"></span>
                     </span>
            </div>
                           <div class="form-group input-group">
                                    <button type="button" data-bind="enable: dateFrom() && dateFrom().length > 0 > 0 && dateTo() && dateTo().length > 0 > 0, click: getReports"  class="btn btn-primary btn-md" id="runReportsButton">

         Run Report
                                    </button>
                           </div>
```

```
                </div>
        </div>
        <div class="row" style="width: 100%;">
                <div class="one-line"
id="chartsContainer"></div>
                <div class="one-line"
id="timeseriesContainer"></div>
                <div class="one-line"
id="purchases"></div>
                <div class="one-line"
id="sales"></div>
                <div class="one-line"
id="barGraph1"></div>
                <div class="one-line"
id="barGraph2"></div>
    </div>
</div>
</body>
</html>
```

return.jsp

```
<%@ page language="java" contentType="text/html;
charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01
Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
        <jsp:include page="header.jsp" />

        <!-- Javascript resource for this particular
page -->
        <script
src="resources/js/pages/return.js"></script>

</head>
<body class="container">
  <div class="page-header">
    <h1>Return Items</h1>
  </div>
  <center>
        <div class="half">
                <form>
                        <div class="form-group
input-group">
                                <span
class="input-group-addon">O.R. Number</span>
                                <input data-
bind="value: receiptId, valueUpdate:'afterkeydown'"
type="text" class="form-control" placeholder="Official
Receipt number" maxlength="11"/>
                        </div>
                        <div class="error"
id="receiptIdError"></div>
                        <div class="modal-footer">
                        <button data-bind="click:
getTransactionsByReceipt, enable: (receiptId().length >
0 && receiptId() > 0)" type="button"
id="editAreaSumbitButton" class="btn btn-
primary">Search</button>
                    </div>
                </form>
        </div>
  </center>
<!-- Return item modal -->
<div id="returnItemModal" class="modal fade"
role="dialog">
  <div class="modal-dialog">
    <!-- Modal content-->
    <div class="modal-content">
      <div class="modal-header">
          <h4 class="modal-title">Return item(s)</h4>
      </div>
      <div class="modal-body">
        <form>
        <input data-bind="value: receiptId"
type="hidden"/>
        <div class"table-responsive">
                                <table
id="itemsTable" class="table table-striped">
                                <thead>
                                <tr>
                                <th>Item</th>
                                <th>Discount</th>
                                <th>Price</th>
                                <th>Quantity</th>
                                </tr>
                                </thead>
                                <tbody data-
bind="foreach: transactionItems">
                                        <td data-
bind="text: itemDescription"></td>
                                        <td data-
bind="text: itemDiscount"></td>
                                    <td data-
bind="text: itemPrice"></td>
                                    <td>
                                        <span>

        <button data-bind="click:
$parent.subtractItemQuantity, enable:
(itemQuantityToReturn() > 0)" type="button" class="btn
btn-primary btn-sm">

                            <span class="glyphicon
glyphicon-minus"></span>

                    </button>

                    <input class="returnInput"
type="text" data-bind="value: itemQuantityToReturn,
disable: true" />

                    <button data-bind="click:
$parent.addItemQuantity, enable: (itemQuantityToReturn()
< itemQuantity())" type="button" class="btn btn-primary
btn-sm">

                            <span class="glyphicon
glyphicon-plus"></span>

                    </button>

          </span>
                                    </td>
                                </tbody>
                                <tbody data-
bind="visible: transactionItems().length == 0">
                                        <td></td>
                                    <td></td>
                                    <td>No item to
show</td>
                                    <td></td>
                                    <td></td>
                                </tbody>
                                </table>
                            </div>
                    <div class="error"
id="editItemError">
                    </div>
                    <div class="modal-footer">
                    <button type="button" class="btn btn-
default" data-dismiss="modal" data-
bind="enable:true">Cancel</button>
                        <button type="button"
id="editItemSumbitButton" class="btn btn-primary" data-
bind="enable: isReturnable(), click:
returnItems">Return</button>
                    </div>
                </form>
      </div>
    </div>
  </div>
</div>
```

```
</body>
</html>
```

rmc.jsp

```
<%@ page language="java" contentType="text/html;
charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<%@ taglib prefix="c"
uri="http://java.sun.com/jsp/jstl/core" %>
<%@ taglib prefix="form"
uri="http://www.springframework.org/tags/form"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01
Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
        <jsp:include page="header.jsp" />

        <!-- Leaflet -->
        <link rel="stylesheet"
href="http://cdn.leafletjs.com/leaflet-
0.7.3/leaflet.css"/>
        <script src="http://cdn.leafletjs.com/leaflet-
0.7.3/leaflet.js"></script>


        <meta name='viewport' content='initial-
scale=1,maximum-scale=1,user-scalable=no' />
        <!-- Mapbox JS and CSS -->
        <script
src='https://api.mapbox.com/mapbox.js/v2.4.0/mapbox.js'>
</script>
        <link
href='https://api.mapbox.com/mapbox.js/v2.4.0/mapbox.css
' rel='stylesheet' />

        <!-- Leaflet AJAX -->
        <script
src="resources/js/plugin/leaflet.ajax.min.js"></script>

        <!-- Ionicons CSS -->
        <link href='resources/css/ionicons.min.css'
rel='stylesheet' />

        <!-- Awesome Markers -->
        <script
src="resources/js/plugin/leaflet.awesome-
markers.min.js"></script>
        <link rel="stylesheet"
href="resources/css/leaflet.awesome-markers.css">

        <!-- Highcharts -->
        <script
src="resources/js/plugin/highcharts.js"></script>

        <!-- Javascript resource for this particular
page -->
        <script
src="resources/js/pages/rmc.js"></script>

</head>
<body class="container">
<div class="container-fluid">
        <div class="page-header rmc-header">
      <h1>Real-time Monitoring Center</h1>
        </div>
        <div>
                <div id="graphContainer"></div>
                <div id="rmcMap"></div>
                <div id="graphContainer"></div>
                        <div class="blocks wrapper">
                          <div class="block green">
                            <div class="heading">
                              Sales Invoice
                            </div>
                            <div class="num" data-
bind="text: transactionCount"></div>
                            <div
class="currentView"><h4 data-bind="text:
currentView"></h4></div>
                          </div>
                        </div>
                        <div class="blocks
wrapper">
                          <div class="block blue">
                            <div class="heading">
                              Total Sales (PhP)
                            </div>
                            <div class="num" data-
bind="text: totalSales"></div>
                            <div
class="currentView"><h4 data-bind="text:
currentView"></h4></div>
                          </div>
                        </div>
            <div id="barGraphContainer1"></div>
            <div id="barGraphContainer2"></div>
        </div>

</div>
</body>
</html>
```

stores.jsp

```
<%@ page language="java" contentType="text/html;
charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01
Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<head>
        <jsp:include page="header.jsp" />

        <link rel="stylesheet"
href="https://unpkg.com/leaflet@1.0.3/dist/leaflet.css"
   integrity="sha512-
07I2e+7D8p6he1SIM+1twR5TIrhUQn9+I6yjqD53JQjFiMf8EtC93ty0
/5vJTZGF8aAocvHYNEDJajGdNx1IsQ=="
   crossorigin=""/>
   <script
src="https://unpkg.com/leaflet@1.0.3/dist/leaflet.js"
   integrity="sha512-
A7vV8IFfih/D732iSSKi20u/ooOfj/AGehOKq0f4vLT1Zr2Y+RX7C+w8
A1gaSasGtRUZpF/NZgzSAu4/Gc41Lg=="
   crossorigin=""></script>


        <!-- Leaflet
        <link rel="stylesheet"
href="http://cdn.leafletjs.com/leaflet-
0.7.3/leaflet.css"/>
        <script src="http://cdn.leafletjs.com/leaflet-
0.7.3/leaflet.js"></script>
        -->

        <meta name='viewport' content='initial-
scale=1,maximum-scale=1,user-scalable=no' />
        <!-- Mapbox JS and CSS -->
        <script
src='https://api.mapbox.com/mapbox.js/v2.4.0/mapbox.js'>
</script>
        <link
href='https://api.mapbox.com/mapbox.js/v2.4.0/mapbox.css
' rel='stylesheet' />

        <!-- Leaflet AJAX -->
        <script
src="resources/js/plugin/leaflet.ajax.min.js"></script>

        <!-- Ionicons CSS -->
        <link href='resources/css/ionicons.min.css'
rel='stylesheet' />
```

```
        <!-- Awesome Markers -->
        <script
src="resources/js/plugin/leaflet.awesome-
markers.min.js"></script>
        <link rel="stylesheet"
href="resources/css/leaflet.awesome-markers.css">

        <!-- Highcharts -->
        <script
src="resources/js/plugin/highcharts.js"></script>

        <!-- Javascript resource for this particular
page -->
        <script
src="resources/js/pages/stores.js"></script>

</head>
<body>
<div class="container-fluid">
        <div class="page-header">
      <h1>Stores</h1>
        </div>
        <div class="row">
                <div class="col-md-8 col-sm-8 col-lg-
9 col-xs-6">
                        <button data-bind="click:
displayAddStore" type="button" class="btn btn-primary
btn-md" id="addStoreButton">
                                <span
class="glyphicon glyphicon-plus"></span> Add new store
                        </button>
                </div>
        <div class="col-md-4 col-sm-4 col-lg-3 col-xs-
6">
            <div id="custom-search-input">
                <div class="input-group col-md-12">
                    <input data-bind="value:
searchString, valueUpdate:'keyup', event: { keyup:
searchStores }" type="text" class="form-control input-
md" placeholder="Branch name, Area name" />
                    <span class="input-group-btn">
                        <i class="glyphicon glyphicon-
search"></i>
                    </span>
                </div>
            </div>
        </div>
        </div>
  <div class"table-responsive">
        <table id="storesTable" class="table table-
striped">
        <thead>
        <tr>
        <th>Store ID</th>
        <th>Area</th>
        <th>Branch Name</th>
        <th>TIN</th>
        <th>Address</th>
        <th>Coordinates</th>
        <th>Action</th>
        </tr>
        </thead>
        <tbody data-bind="foreach: stores">
                <td data-bind="text: storeId"></td>
          <td data-bind="text: area, value:
area_id"></td>
            <td data-bind="text: branchName"></td>
            <td data-bind="text: tin"></td>
            <td data-bind="text: address"></td>
            <td data-bind="text: coordinates"></td>
            <td>
            <a data-bind="tooltip: {title:
editItemTitle, placement: 'left' }, attr: {id: storeId},
click: $parent.displayStore" href="" data-
toggle="tooltip" data-placement="top"><span
class="glyphicon glyphicon-pencil" aria-
hidden="true"></span></a>
            <!-- <a data-bind="tooltip: {title:
delItemTitle, placement: 'left' }, attr: {id: storeId},
click: $parent.removeStore" href="" data-
toggle="tooltip" data-placement="top"><span
class="glyphicon glyphicon-remove red" aria-
hidden="true"></span></a> -->
            </td>
        </tbody>
        <tbody data-bind="visible: stores().length ==
0">
                <td></td>
          <td></td>
          <td>No store to show</td>
          <td></td>
          <td></td>
        </tbody>
        </table>
  </div>
  <div class="text-center col-md-4 col-md-offset-4">
        <ul class="pagination pagination-md">
                <li data-bind="css: { disabled:
(currPage() === 1) }"><a data-bind="css: { disabled:
(currPage() === 1) }, click: function() { if(currPage()
!= 1) { doPaging(pageSize(), currPage()-1)}}"
href="">&laquo;</a></li>
        </ul>
    <ul data-bind="foreach: pagesArray"
class="pagination pagination-md">
                <li data-bind="css: { active:
$parent.currPage() === pageNumber()}"><a data-
bind="text: pageNumber, click: function() {
$parent.doPaging($parent.pageSize(), pageNumber())}"
href=""></a></li>
        </ul>
        <ul class="pagination pagination-md">
                <li data-bind="css: { disabled:
(currPage() === maxNumberPages() || maxNumberPages() ===
0) }"><a data-bind="click: function() { if(currPage() !=
maxNumberPages()) { doPaging(pageSize(),
currPage()+1)}}" href="">&raquo;</a></li>
        </ul>
  </div>
</div>
<!-- Add/Edit store modal -->
<div id="editStoreModal" class="modal fade"
role="dialog">
  <div class="modal-dialog">
    <!-- Modal content-->
    <div class="modal-content">
      <div class="modal-header">
          <h4 class="modal-title">Add new store</h4>
      </div>
      <div class="modal-body">
        <form>
          <input type="hidden" data-bind="value:
storeid"/>
                <div class="form-group input-group">
                        <span class="input-group-
addon">Store Area*</span>
                        <select class="form-control"
data-bind="options: areas,
                    optionsText: 'areaName',
                    optionsValue: 'areaId',
                    value: areaid,
                    optionsCaption: 'Choose area...',
                    ">
                </select>
                </div>
                <div class="form-group input-group">
                        <span class="input-group-
addon">Branch Name*</span>
                        <input data-bind="value:
branchname, valueUpdate:'afterkeydown'" type="text"
class="form-control" placeholder="Branch name"
required="" maxlength="45"/>
                </div>
                <div class="form-group input-group">
                        <span class="input-group-
addon">TIN*</span>
```

```
                        <input data-bind="value:
taxIdNumber, valueUpdate:'afterkeydown'" type="text"
class="form-control" placeholder="Tax Identification
Number" required="" maxlength="12"/>
                    </div>
                    <div class="form-group input-group">
                        <span class="input-group-
addon">Address*</span>
                        <input data-bind="value:
branchaddress, valueUpdate:'afterkeydown'" type="text"
class="form-control" placeholder="Address" required=""
maxlength="160"/>
                    </div>
                    <div class="form-group input-group">
                        <span class="input-group-
addon">Coordinates*</span>
                        <input data-bind="value:
coordinates, valueUpdate:'afterkeydown', numeric"
type="text" class="form-control"
placeholder="Coordinates" required="" maxlength="60"/>
                        <span data-bind="click:
displayCoordinates" class="input-group-addon">
                            <span class="glyphicon
glyphicon-pushpin"></span>
                    </span>
                    </div>
                    <!-- <div class="form-group input-
group">
                        <span class="input-group-
addon">Longitude*</span>
                        <input data-bind="value:
longitude, valueUpdate:'afterkeydown', numeric"
type="text" class="form-control" placeholder="Longitude"
required="" maxlength="30"/>
                    </div> -->
                    <div class="error"
id="editStoreError">
                    </div>
                    <div class="modal-footer">
                    <button type="button" class="btn btn-
default" data-dismiss="modal" data-bind="click:
cancelEdit">Cancel</button>
                    <button type="button"
id="editStoreSumbitButton" class="btn btn-primary" data-
bind="click: editStore, enable: (areaid &&
branchname().length > 0 && taxIdNumber().length == 12 &&
branchaddress().length > 0 && coordinates().length >
0)">Add</button>
                    </div>
                </form>
        </div>

    </div>
  </div>
</div>
<!-- Modal that contains map -->
<div id="mapSelectModal" class="modal fade"
role="dialog">
  <div class="modal-dialog">
    <!-- Modal content-->
    <div class="modal-content">
      <div class="modal-header">
          <h4 class="modal-title mod-title">Click map to
select coordinates</h4>
      </div>
      <div class="modal-body">
        <div class="table-responsive">
                        <div
id="mapContainer"></div>
                        <div class="modal-footer">
                    <button id="mapModal" type="button"
class="btn btn-default" data-
dismiss="modal">Close</button>
                    </div>
                    </div>
      </div>
    </div>
  </div>
</div>
```

```
</body>
</html>
```

users.jsp

```
<%@ page language="java" contentType="text/html;
charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01
Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
        <jsp:include page="header.jsp" />

        <!-- Javascript resource for this particular
page -->
        <script
src="resources/js/pages/users.js"></script>
</head>
<body class="container">
<div class="container-fluid">
        <div class="page-header">
      <h1>System users</h1>
        </div>
        <div class="row">
                    <div class="col-md-8 col-sm-8 col-lg-
9 col-xs-6">
                        <button data-bind="click:
displayAddUser" type="button" class="btn btn-primary
btn-md" id="addItemButton">
                                    <span
class="glyphicon glyphicon-plus"></span> Add new user
                        </button>
                    </div>
       <div class="col-md-4 col-sm-4 col-lg-3 col-xs-
6">
            <div id="custom-search-input">
                <div class="input-group col-md-12">
                    <input data-bind="value:
searchString, valueUpdate:'keyup', event: { keyup:
searchUsers }" type="text" class="form-control input-md"
placeholder="User name, type, branch name, or email" />
                    <span class="input-group-btn">
                        <i class="glyphicon glyphicon-
search"></i>
                    </span>
                </div>
            </div>
        </div>
        </div>
        <div class"table-responsive">
        <table id="itemsTable" class="table table-
striped">
        <thead>
        <tr>
        <th>User ID</th>
        <th>User Name</th>
        <th>User Type</th>
        <th>Branch Name</th>
        <th>Email address</th>
        <th>Contact No.</th>
        <th>Action</th>
        </tr>
        </thead>
        <tbody data-bind="foreach: users">
                <td data-bind="text: userId"></td>
            <td data-bind="text: userName"></td>
            <td data-bind="text: usertype"></td>
            <td data-bind="text: branchName"></td>
            <td data-bind="text: email"></td>
            <td data-bind="text: contactNo"></td>
            <td>
            <a data-bind="tooltip: {title:
editUserTitle, placement: 'left' }, attr: {id: userId},
click: $parent.displayUser" href="" data-
toggle="tooltip" data-placement="top"><span
```

```
class="glyphicon glyphicon-pencil" aria-
hidden="true"></span></a>
            <a data-bind="tooltip: {title:
editPassTitle, placement: 'left' }, attr: {id: userId},
click: $parent.displayChangePassword" href="" data-
toggle="tooltip" data-placement="top"><span
class="glyphicon glyphicon-lock" aria-
hidden="true"></span></a>
            <a data-bind="text: enableDisable, tooltip:
{title:  enableDisableTitle, placement: 'left' }, attr:
{id: userId, name: isActive}, click:
$parent.enableDisableUser" href="" data-toggle="tooltip"
data-placement="top"></a>
          </td>
        </tbody>
        </table>
  </div>
  <div class="text-center col-md-4 col-md-offset-4">
        <ul class="pagination pagination-md">
                <li data-bind="css: { disabled:
(currPage() === 1) }"><a data-bind="css: { disabled:
(currPage() === 1) }, click: function() { if(currPage()
!= 1) { doPaging(pageSize(), currPage()-1)}}"
href="">&laquo;</a></li>
        </ul>
    <ul data-bind="foreach: pagesArray"
class="pagination pagination-md">
                <li data-bind="css: { active:
$parent.currPage() === pageNumber()}"><a data-
bind="text: pageNumber, click: function() {
$parent.doPaging($parent.pageSize(), pageNumber())}"
href=""></a></li>
        </ul>
        <ul class="pagination pagination-md">
                <li data-bind="css: { disabled:
(currPage() === maxNumberPages() || maxNumberPages() ===
0) }"><a data-bind="click: function() { if(currPage() !=
maxNumberPages()) { doPaging(pageSize(),
currPage()+1)}}" href="">&raquo;</a></li>
        </ul>
  </div>
</div>
<!-- Add/Edit item modal -->
<div id="editUserModal" class="modal fade"
role="dialog">
  <div class="modal-dialog">
    <!-- Modal content-->
    <div class="modal-content">
      <div class="modal-header">
        <h4 class="modal-title">Add new user</h4>
      </div>
      <div class="modal-body">
        <form>
          <input data-bind="value: userid"
type="hidden"/>
                <div class="form-group input-group">
                        <span class="input-group-
addon">User name*</span>
                        <input data-bind="value:
username, valueUpdate:'afterkeydown'" type="text"
class="form-control" placeholder="User name" required=""
maxlength="20"/>
                </div>
                <div class="form-group input-group"
data-bind="visible: !(userid() && userid() > 0)">
                        <span class="input-group-
addon">Password*</span>
                        <input data-bind="value:
userpassword1, valueUpdate:'afterkeydown'"
type="password" class="form-control"
placeholder="Password" maxlength="20"/>
                </div>
                <div class="form-group input-group"
data-bind="visible: !(userid() && userid() > 0)">
                        <span class="input-group-
addon">Re-type password*</span>
                        <input data-bind="value:
userpassword2, valueUpdate:'afterkeydown', event: {
blur: checkPasswords}" type="password" class="form-
control" placeholder="Password" maxlength="20"/>
                </div>
                <div class="form-group input-group">
                        <span class="input-group-
addon">User type*</span>
                        <select class="form-control"
data-bind="options: usertypesForDisplay,
                optionsText: 'usertypeName',
                optionsValue: 'usertypeId',
                value: usertypeid,
                optionsCaption: 'User type...',
                ">
              </select>
                </div>
                <div class="form-group input-group">
                        <span class="input-group-
addon">Area*</span>
                        <select class="form-control"
data-bind="options: areas,
                optionsText: 'areaName',
                optionsValue: 'areaId',
                value: areaid,
                optionsCaption: 'Area...',
                disable: (loggedInUsertype()==2)
|| !(usertypeid() > 0)
                ">
              </select>
                </div>
                <div class="form-group input-group">
                        <span class="input-group-
addon">Branch name*</span>
                        <select class="form-control"
data-bind="options: stores,
                optionsText: 'storeBranchName',
                optionsValue: 'storeId',
                value: storeid,
                optionsCaption: 'Branch...',
                disable: (loggedInUsertype()==2)
|| !(areaid() > 0) || !(usertypeid() > 0)
                ">
              </select>
                </div>
                <div class="form-group input-group">
                        <span class="input-group-
addon">Email*</span>
                        <input data-bind="value:
emailAd, valueUpdate:'afterkeydown'" type="text"
class="form-control" placeholder="Email address"
required="" maxlength="90"/>
                </div>
                <div class="form-group input-group">
                        <span class="input-group-
addon">Contact number*</span>
                        <input data-bind="value:
contactNum, valueUpdate:'afterkeydown'" type="text"
class="form-control" placeholder="Contact number"
required="" maxlength="13"/>
                </div>
                <input data-bind="value: isactive"
type="hidden"/>
                <div class="error"
id="editUserError">
                </div>
                <div class="modal-footer">
                <button type="button" class="btn btn-
default" data-dismiss="modal" data-bind="click:
cancelEdit">Cancel</button>
                <button type="button"
id="editUserSubmitButton" data-bind="enable: (username()
&& username().length > 0 && usertypeid() > 0 &&
storeid() > 0 && emailValid() && contactNum() > 0 &&
(passwordsEqual() || (userid() && userid() > 0))),
click: editUser" class="btn btn-primary">Add</button>
                </div>
            </form>
      </div>

    </div>
```

```
  </div>
</div>
<!-- Change password modal -->
<div id="changePasswordModal" class="modal fade"
role="dialog">
  <div class="modal-dialog">
    <!-- Modal content-->
    <div class="modal-content">
      <div class="modal-header">
          <h4 class="modal-title">Change password</h4>
      </div>
      <div class="modal-body">
        <form>
          <input data-bind="value: user_id"
type="hidden"/>
                    <div class="form-group input-group"
data-bind="visible: !(userid() && userid() > 0)">
                        <span class="input-group-
addon">New Password*</span>
                        <input data-bind="value:
userpassword_1, valueUpdate:'afterkeydown', event: {
keyup: checkChangePasswords}" type="password"
class="form-control" placeholder="Password"
maxlength="20"/>
                    </div>
                    <div class="form-group input-group"
data-bind="visible: !(userid() && userid() > 0)">
                        <span class="input-group-
addon">Re-type password*</span>
                        <input data-bind="value:
userpassword_2, valueUpdate:'afterkeydown', event: {
keyup: checkChangePasswords}" type="password"
class="form-control" placeholder="Password"
maxlength="20"/>
                    </div>
                    <div class="error"
id="changePasswordError">
                    </div>
                    <div class="modal-footer">
                    <button type="button" class="btn btn-
default" data-dismiss="modal" data-bind="click:
cancelEdit">Cancel</button>
                <button type="button"
id="changePasswordSubmitButton" data-bind="enable:
(userpassword_1() && userpassword_2() &&
checkChangePasswords()), click: changePassword"
class="btn btn-primary">Change</button>
                    </div>
            </form>
      </div>

    </div>
  </div>
</div>
</body>
</html>
```

pom.xml

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/maven-v4_0_0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <groupId>edu.up.cas.sp</groupId>
  <artifactId>ROXSys</artifactId>
  <packaging>war</packaging>
  <version>0.0.1-SNAPSHOT</version>
  <name>Mountainshop Maven Webapp</name>
  <url>http://maven.apache.org</url>

  <properties>

<springframework.version>4.0.6.RELEASE</springframework.
version>

<hibernate.version>4.3.6.Final</hibernate.version>

<mysql.connector.version>5.1.31</mysql.connector.version
>
        <joda-time.version>2.3</joda-time.version>
        <testng.version>6.9.4</testng.version>
        <mockito.version>1.10.19</mockito.version>
        <h2.version>1.4.187</h2.version>
        <dbunit.version>2.2</dbunit.version>
  </properties>
  <dependencies>
    <dependency>
      <groupId>junit</groupId>
      <artifactId>junit</artifactId>
      <version>3.8.1</version>
      <scope>test</scope>
    </dependency>

    <!-- Spring -->
    <dependency>
      <groupId>org.springframework</groupId>
      <artifactId>spring-core</artifactId>
      <version>${springframework.version}</version>
    </dependency>
    <dependency>
      <groupId>org.springframework</groupId>
      <artifactId>spring-web</artifactId>
      <version>${springframework.version}</version>
    </dependency>
    <dependency>
      <groupId>org.springframework</groupId>
      <artifactId>spring-webmvc</artifactId>
      <version>${springframework.version}</version>
    </dependency>
    <dependency>
      <groupId>org.springframework</groupId>
      <artifactId>spring-tx</artifactId>
      <version>${springframework.version}</version>
    </dependency>
    <dependency>
      <groupId>org.springframework</groupId>
      <artifactId>spring-orm</artifactId>
      <version>${springframework.version}</version>
    </dependency>
    <dependency>
      <groupId>org.springframework.security</groupId>
      <artifactId>spring-security-config</artifactId>
      <version>4.0.3.RELEASE</version>
    </dependency>
    <!-- Hibernate -->
    <dependency>
       <groupId>org.hibernate</groupId>
       <artifactId>hibernate-core</artifactId>
       <version>${hibernate.version}</version>
    </dependency>

        <!-- jsr303 validation -->
        <dependency>
          <groupId>javax.validation</groupId>
          <artifactId>validation-api</artifactId>
          <version>1.1.0.Final</version>
        </dependency>
        <dependency>
          <groupId>org.hibernate</groupId>
          <artifactId>hibernate-validator</artifactId>
          <version>5.1.3.Final</version>
        </dependency>

        <!-- MySQL -->
        <dependency>
          <groupId>mysql</groupId>
          <artifactId>mysql-connector-
java</artifactId>

<version>${mysql.connector.version}</version>
        </dependency>

        <!-- Gson -->
        <dependency>
          <groupId>com.google.code.gson</groupId>
```

```
            <artifactId>gson</artifactId>
            <version>2.5</version>
          </dependency>

          <!-- Joda-Time -->
          <dependency>
            <groupId>joda-time</groupId>
            <artifactId>joda-time</artifactId>
            <version>${joda-time.version}</version>
          </dependency>

          <!-- To map JodaTime with database type -->
          <dependency>
            <groupId>org.jadira.usertype</groupId>
            <artifactId>usertype.core</artifactId>
            <version>3.0.0.CR1</version>
          </dependency>

          <!-- Servlet+JSP+JSTL -->
          <dependency>
            <groupId>javax.servlet</groupId>
            <artifactId>javax.servlet-api</artifactId>
            <version>3.1.0</version>
            <scope>provided</scope>
          </dependency>
          <dependency>
            <groupId>javax.servlet.jsp</groupId>
            <artifactId>javax.servlet.jsp-
api</artifactId>
            <version>2.3.1</version>
            <scope>provided</scope>
          </dependency>
          <dependency>
            <groupId>javax.servlet</groupId>
            <artifactId>jstl</artifactId>
            <version>1.2</version>
          </dependency>

          <!-- Json dependencies -->
          <dependency>
            <groupId>org.json</groupId>
            <artifactId>json</artifactId>
            <version>20160212</version>
          </dependency>

    <!-- Itext PDF Writer -->
    <!--
https://mvnrepository.com/artifact/com.itextpdf/itextpdf
-->
          <dependency>
          <groupId>com.itextpdf</groupId>
          <artifactId>itextpdf</artifactId>
          <version>5.0.6</version>
          </dependency>

          <!-- Testing dependencies -->
          <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-test</artifactId>

<version>${springframework.version}</version>
            <scope>test</scope>
          </dependency>
          <dependency>
            <groupId>org.testng</groupId>
            <artifactId>testng</artifactId>
            <version>${testng.version}</version>
            <scope>test</scope>
          </dependency>
          <dependency>
            <groupId>org.mockito</groupId>
            <artifactId>mockito-all</artifactId>
            <version>${mockito.version}</version>
            <scope>test</scope>
          </dependency>
          <dependency>
            <groupId>com.h2database</groupId>
            <artifactId>h2</artifactId>
            <version>${h2.version}</version>
            <scope>test</scope>
          </dependency>
          <dependency>
            <groupId>dbunit</groupId>
            <artifactId>dbunit</artifactId>
            <version>${dbunit.version}</version>
            <scope>test</scope>
          </dependency>
          <!-- JSON Parser -->
          <dependency>
        <groupId>com.googlecode.json-simple</groupId>
        <artifactId>json-simple</artifactId>
        <version>1.1.1</version>
          </dependency>
    </dependencies>
          <build>
         <pluginManagement>
             <plugins>
                 <plugin>

<groupId>org.apache.maven.plugins</groupId>
                     <artifactId>maven-war-
plugin</artifactId>
                     <version>2.4</version>
                     <configuration>

<warSourceDirectory>src/main/webapp</warSourceDirectory>

<warName>ROXSys</warName>

<failOnMissingWebXml>false</failOnMissingWebXml>
                     </configuration>
                 </plugin>
             </plugins>
         </pluginManagement>
         <finalName>ROXSys</finalName>
     </build>
</project>
```

XI. Acknowlegement

Gusto ko lang magpasalamat sa lahat ng taong tumulong sa akin. Sa mga taong nakikinig sa mga kwento ko kahit pauli-ulit na. Haha. Sa aking mga kaibigan sa kabundukan, sa aking mga blockmates na nakakausap ko pa rin hanggang ngayon, maraming salamat sa inyong lahat.

Hindi ko alam kung saan magsisimula kasi andami kong gustong sabihin. Pero nais ko lamang sabihin sa lahat na tumulong sa akin na hindi ko kayo makakalimutan. Iisa-isahin ko na kayo.

Una, maraming-maraming salamat sa'yo sir Marvin Ignacio dahil pumayag ka na maging adviser ko. Akala ko hindi na talaga ako gagraduate. Maraming salamat kay sir Bryann Chua dahil pinasa niya ako sa'yo.

Pangalawa, kay ma'am Ruby Palma ng R.O.X., maraming salamat po sa inyo. Hindi ko kayo makakalimutan.

Pangatlo, sa mga kaibigan ko sa trabaho noon at sa kasalukuyan: sina Bogs, Von, Francis, Jayson, Belen, Jenny, Kristian, Melody. Kahit hindi niyo man alam pinagdadaanan ko, napapasaya niyo ako kahit papaano. Haha.

Sa Witty Birds, hindi ko alam ano'ng nangyari sa atin pero nagpapasalamat ako at nakilala ko kayo. Napapasaya niyo ako kapag umaalis tayo at pumupunta sa beach, falls o kaya ay namumundok. Salamat sa inyong lahat lalo na kina Vevey, Eloisa, Jayson, Francis, Wenjon, at Sheila.

Maraming salamat sa nanay ko na hanggang sa huli ay umaasa pa rin na makita akong magmartsa. Eto na ma, haha. Sampung taon din ito. Alam kong pinagdarasal mo ako palagi in silence, nararamdaman ko iyon kahit di mo man sabihin. Sana matupad na mga pangarap natin sa buhay at sisikapin kong mapabuti buhay natin lalo. Sana humaba pa buhay mo para maibigay ko sayo yung buhay na gusto mo.

Higit sa lahat, nagpapasalamat ako sa Panginoon kasi hindi Niya ako pinapabayaan. May trabaho pa rin ako na may kinalaman sa kurso ko, nakakatulong sa pamilya at nagagawa ang mga bagay na gustong gawin. Hindi ko alam plano niya sa akin pero alam ko lang na gusto niya akong mapabuti. Hay. Dahil din sa karanasan na ito natuto akong maging mapagkumbaba at maging maunawain.

Sa mga ka-block ko pang hindi pa natatapos, sana matapos na tayong lahat. Alam ko may kanya-kanya tayong rason at darating ang panahon na matutupad mga pangarap natin. Di ko na alam pinagsasabi ko. Haha.

Salamat sa inyong lahat! Pag magpapalibre kayo, sabihan niyo lang ako. Haha.