University of the Philippines Manila
College of Arts and Sciences
Department of Physical Sciences and Mathematics

GRAPT: A Gesture Recognition Application for Physical Therapy

A special problem in partial fulfillment
of the requirements for the degree of
Bachelor of Science in Computer Science

Submitted by:

Andrei Mikail B. Macatangay

June 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 "GRAPT: A Gesture Recognition Application for Physical Therapy" prepared and submitted by Andrei Mikail B. Macatangay in partial fulfillment of the requirements for the degree of Bachelor of Science in Computer Science has been examined and is recommended for acceptance.

Ma. Sheila A. Magboo, M.Sc.
Adviser

EXAMINERS:

	Approved	**Disapproved**
1. Gregorio B. Baes, Ph.D. (*cand.*)	_____	_____
2. Avegail D. Carpio, M.Sc.	_____	_____
3. Richard Bryann L. Chua, Ph.D. (*cand.*)	_____	_____
4. Perlita E. Gasmen, M.Sc. (*cand.*)	_____	_____
5. Marvin John C. Ignacio, M.Sc. (*cand.*)	_____	_____
6. Vincent Peter C. Magboo, M.D., M.Sc.	_____	_____

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

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

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

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

Abstract

The physical therapist relies on intuition to check how close or far the patient is from performing a movement they have to perform. A personal perspective is a biased one which will not properly reflect the patient's progress.

Movements can be stored in the form of accelerometer signals and because of the time series nature of these signals, the researcher is able to use Dynamic Time Warping to compare them. The scores generated by this approach will be used in determining if the patient is making progress in his rehabilitation.

Keywords: gesture recognition, accelerometer, dynamic time warping

Contents

List of Figures

I. Introduction

A. Background of the Study

Smartphones are becoming increasingly indispensable in everyday life while offering a substantial variety of mobile applications which are used in disseminating information, connecting people, and providing leisure [1]. Needless to say, the smartphone has become the ultimate tool for the everyday traveler.

Smartphones were introduced to new sensors such as the accelerometer, gyroscope, and GPS [2]. These sensors are used in collecting data in hope of not only creating life logs but in predicting outcomes as well. An accelerometer is an electromechanical device used in measuring acceleration forces [3]. It has been used in applications such as pedometers [4], speedometers, and other acceleration related metrics. A functionality which is yet to be introduced to smartphones is storing motion gestures which can be done with the help of the accelerometer.

Accelerometer input can be transformed into time series data. Time series data can be compared with another time series data by using a technique called Dynamic Time Warping. Movements can be described as accelerometer input and can therefore be compared with other movements.

A gesture is any movement of any part of the body that represents meaning. These gestures have a significant amount of uses which range from unlocking your phone, communication such as sign languages [5], and controlling movements in virtual reality [6] [7].

People who are experiencing neurological movement disorders such as stroke [8] [9], Cerebral Palsy [10] [11], Parkinsons Disease [12], and spinal cord injuries [13] are asked to perform repeated movements to gain lost control or improve muscle function. Repeated movements are also performed by people who want to practice a specific movement to develop a sport science term called muscle memory [14]. Muscle memory

is the process in which the body grows accustomed to certain types of movements. This in turn helps athletes perform these movements at will.

B. Statement of the Problem

A major problem encountered in the field of Physical Therapy is that the attending professional cannot be with the patient at all times [7]. In his absence, there isn't anyone who is able to check if the movements being performed by the patient are on point.

People undergoing therapy tend to rely on intuition to check how close or far they are from performing the gesture they have to perform. With this being said, the treatment process becomes ineffective because the proper movements are not performed by the patient. The improper performance of these said movements could aggravate the patient's condition or lead to new injuries.

Multiple models for gesture recognition use accelerometer input data. The sign sequence approach has generated 95% accuracy in finding non-specific gestures [15]. The dynamic time warping model is able to compare gestures with an accuracy of 99.4% on specific gestures [16]. Even with the achievement of these near perfect accuracies, there is still no gesture recognition application available for smartphones even if it comes equipped with the accelerometer sensor.

The Wii remote is the only publicly available device that implements gesture recognition using the accelerometer and requires a Nintendo Wii to pair with [17]. Smartphones nowadays have the capability to perform gesture recognition and more people own them compared to the Wii-mote.

C. Objectives of the Study

The study aims to do the following:

1. Enable the physical therapist to:

   (a) Record three-dimensional gestures which will be performed by the patient.

   (b) Perform gesture recognition on the recorded three-dimensional gestures.

   (c) Simulate a session or a sequence of gestures which will be performed by the patient.

2. Enable the patient to:

   (a) Select which motivational feedback to play upon performing a gesture during each session.

   (b) Record a video describing how the gesture should be performed.

   (c) Perform the session recorded by the Physical Therapist.

   (d) Perform the gestures required in each session.

   (e) Be alerted when the date and time comes for him to undergo the session has come.

   (f) Store the results of the session performed by the patient regardless if schedule is met or not.

   (g) Generate a report of the performance of activity, in pdf form, containing the scores per gesture performed, the date on which the session was held, and name of the session.

D. Significance of the Project

The application will be able to help patients suffering from neurological movement disorders perform repeated movements even in the absence of his trained professional. Therapy sessions may not take up a fortune but they are not cheap either. The local scene describes the cost to range between one and two thousand pesos per session

while lasting only for an hour. This tool will be able to reduce the number of meetings with the therapist which will help bring down therapy expenses without sacrificing improvement.

The application will serve as the starting point for the field of physical therapy in the exploration to better understand the human body. It may soon be able to answer one of the biggest questions in their field which is when do you say a gesture is performed correctly. The application will allow them, through more research, to create thresholds or scores which will aid in the production of answers to that question.

This application will show your progress in rehabilitation through the reports it generates. The patient will be able to see how much he has improved throughout the days allowing him to estimate how long he still has to be in therapy.

Motivation is an important factor in rehabilitation and is frequently used as a determinant of rehabilitation outcome [18]. The application gives motivational feedback which is one of the many principles needed in rehabilitation [7].

The heart of the application lies with the accelerometer sensor. Accelerometers in smartphones are currently being used for measuring tilt with respect to the earth, measuring the number of steps you take and calculating how fast you are going. The sensor is capable of gesture recognition and not using this feature leaves it underutilized.

E. Scope and Limitations

1. The tool will only be used to detect hand gestures while holding the phone.
2. The application will require the user to tell that he is starting a gesture by holding the record button while releasing it will stop recording the gesture.
3. Patient must perform the gesture with the same orientation.

F. Assumptions

1. The user will be trained to properly hold and move the phone while performing the gesture.

2. The OS running in the mobile device is Android .

3. The mobile device must have an accelerometer.

II. Review of Related Literature

There are two methods in recognizing gestures. The first method is vision based which involves the use of a camera. The second method is motion based which requires the use of motion sensing devices such as the accelerometer.

A. Gesture Recognition Through Image Processing

Yunda Liu et al. [19] used the Kinect sensor to recognize gestures. The Kinect sensor is a somatosensory camera which solved the problem of locating the position of the hand which traditional gesture recognition techniques find hard to do. The system combined the key points of skeletal trails matching with static key frames matching to carry on dynamic gesture recognition. Dynamic gestures compared to static gestures consist of a series of motions while static gestures are defined as a single move of the hand. The study was only able to detect three kinds of dynamic gestures which are the left hand waving through right gesture, right hand waving through left gesture and right hand lifting up gesture with at least 95% accuracy.

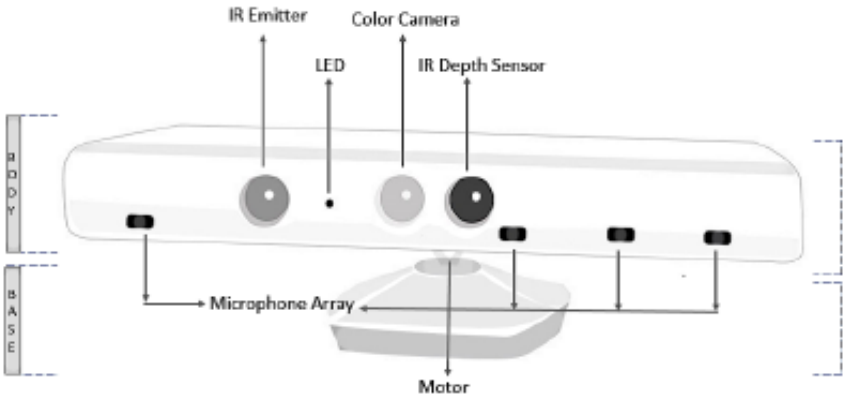


Figure 1: The Kinect Sensor

Montejo [20] used the Kinect sensor for moving and rotating images in the operating room. Inspiration for his work came from the fact that controller devices may possibly contaminate therefore it is best to find a way in which surgeons do not come

in contact with them. The gestures allowed the user to zoom in and out of a picture, browse left and right, adjust brightness and pan up, down, left and right. The paper used a template matching algorithm and had a limited amount of gestures you could perform.

B. Gesture Recognition Through Motion Sensing Devices

Rengqiang Xie et al. [21] created a gesture recognition model based on accelerometer input through the use of the Feedforward Neural Networks on a specific set of gestures which are the right, left, forward, backward, and stop. FNN is an ANN that has encoded learned knowledge in weights which made prediction really fast. Every gesture at least had 100 samples which were used to train the FNN.

Ruize Xu et al. [15] presented three models for gesture recognition using accelerometer input. The models are Sign Sequences and Hopfield Networks, Velocity Increment, and Sign Sequences and Template Matching. The models were used in recognizing the up, down, left, right, tick, circle and cross gestures. The first model looked for sign sequences which are obtained by subtracting the mean from elements of the data set and then finding the alteration in signs. A Hopfield Network was used as associative memory to make it fault tolerant. The second model looked for the sign sequences then calculates for the area bounded by the curve which is represented by each sign and the x axis. The third model is similar to the first model only that it did not use Hopfield Networks. It showed a higher accuracy than their previous works which used Hidden Markov Models. HMM has required training and the three models presented by Ruize Xu et al. do not. Similarly Jayaraman D. et al.[22] also published a paper using the same three models with the only difference being that the third model achieved a higher accuracy of 96% compared to the 95.6% of Ruize Xu et al.

Zhe Ji et al. [16] presented a model using the Dynamic Time Warping algorithm

for gesture recognition. DTW has computed for the optimal alignment between two time series which made it a good way of evaluating their similarity. They introduced the Fast Dynamic Time Warping algorithm which is eight times faster than the normal DTW because it can find the optimal alignment in linear time.

Ahmad Akl et al. [23] presented a model which used a clustering algorithm to determine gestures. It involved the use of both DTW and Affinity Propagation. Affinity Propagation is an algorithm that continuously creates representatives for each gesture. The recognition system can recognize 18 gestures. Their model had garnered a higher accuracy compared to models using the Continuous-HMM system and system of discrete HMMs.

Tea Marasovic et al. [24] presented and evaluated a gesture recognition system which uses the accelerometer which was used to capture sensor data generated by hand movements. The K nearest neighbour algorithm and principal component analysis were used in the system. PCA is a simple, nonparametric method for extracting relevant information from complex data sets. K nearest neighbour algorithm was used in the classification of an object through the majority vote of its neighbours, with the object being assigned to the class having become the most common among its k nearest neighbours. The experimental results were based on acceleration data collected for seven different gesture types and show that the system reaches recognition accuracy between 85 and 87%, with the lowest calculated precision rate of 71%.

Hari Gupta et al. [25] present a CHG technique that is capable of continuous recognition of hand gestures using a smart device. A key attribute of this CHG technique is that it did not require any active user intervention in terms of the start or end points of the continuous gestures.

C. Applications of Gesture Recognition

Luis Sucar et al. [6] created a virtual reality application that helped in motor rehabilitation. The principles encouraging effective rehabilitation which were discussed are repetition, feedback, motivation, and task oriented training. The application was be described as a series of games that included flipping steak, cleaning windows and killing a mosquito. The system used a webcam to track the movements of the pressure sensor. The tracking algorithm was based on a particle filter which recognizes the tracking target based on color and texture features. Particle filters are a probabilistic technique based on Monte Carlo methods. They maintained a sampled representation of the target objects distribution, where each sample is a particle that is a point in a state space with a certain mass depending on its significance.

III. Theoretical Framework

A. The Accelerometer

An accelerometer is an electromechanical device that measures acceleration [3]. The accelerometer was originally introduced to the smartphone in order to determine the angle at which the device is tilted with respect to earth. The things considered by the researcher in the development of applications which use the accelerometer are the following.

1. Number of axes

   Accelerometer come in either two or three axes.

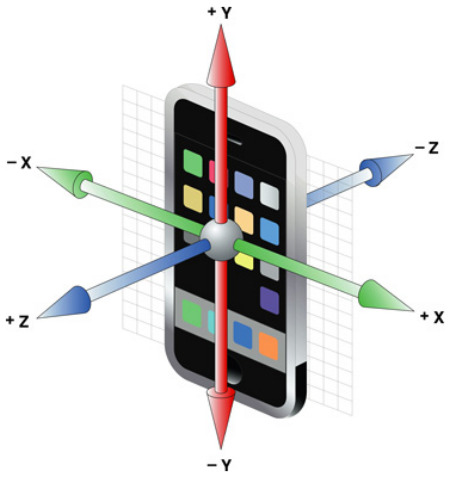


Figure 2: Orientation of a 3-axis Accelerometer in a Smartphone

2. Maximum Swing

   A 1.5g accelerometer is good for measuring the tilt with respect to earth. A 2g should be used if you are going to use the accelerometer to measure the motion of a car, plane or robot. A 5g is used for projects that experience very sudden starts and stops.

3. Sensitivity

   Generally speaking, the more sensitivity the better. This means that for a given

change in acceleration, there will be a larger change in signal. Since larger signal changes are easier to measure, you will get more accurate readings.

4. Bandwidth

   Bandwidth is the amount of times per second you can take a reliable acceleration reading. For slow moving tilt sensing applications, a bandwidth of 50Hz will probably suffice. Vibration measurement, or control a fast moving machine, will need a bandwidth of several hundred Hz.

B. Gesture Segmentation

Gesture segmentation is the process of finding the start and end of each gesture [15]. The gesture segmentation process implemented by the researcher is manual which means that there will be user participation to find these terminal points. In this study, the user will be prompted to hold a button to start recording the gesture and releasing it will mark the end of the recording.

C. Moving Average Filter

Accelerometer data can have large deviations because of shaky hands [15] and because of this, most gesture recognition methods filter out the noise before working on the data. The moving average filter operates by averaging a number of points from the input signal to produce each point in the output signal. The moving average filter is optimal for reducing random noise while keeping a sharp step response. The equation is written in this form:

$$y[i] = 1/M \sum_{j=0}^{M-1} x[i+j] \tag{1}$$

Y[i] is the output signal whose index is at i. M is the number of points to be averaged from the input signal. The moving average filter is named based on how

many points are taken to filter out noise. If 51 points were averaged then it is called the 51 point moving average.

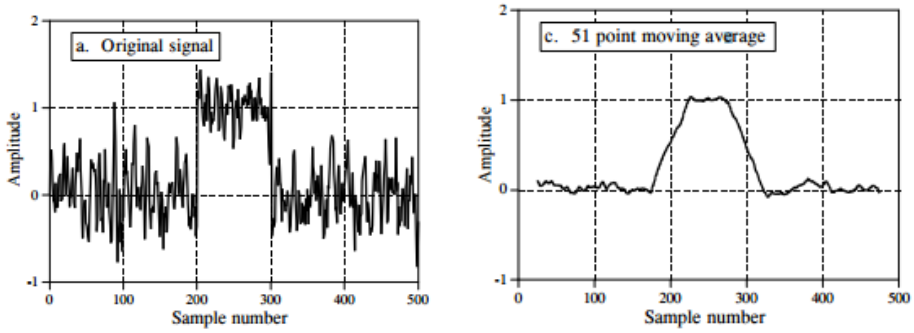


Figure 3: Signal Before and After Using the 51 point Moving Average Filter

D. Series Stretching and Withdrawal

Dynamic Time Warping is able to match similar shapes even when the series are of different lengths. Gesture recognition is performed by comparing the gesture to every element available in the library. The one with the least value or the one closest to zero is then hailed as the most likely gesture.

The said score is obtained by creating a matrix of size m by n where m and n are the sizes of the two being compared. The DTW score is dependent on this said matrix and comparing scores which are generated by matrices with different sizes will provide bias. The DTW matrix with smaller size will most likely have the smallest score since the path it will traverse will be smaller than that of a bigger sized matrix.

There are two cases the researcher must handle. The first is if the series is longer than the one it is being compared to and the latter when the series is shorter. The steps in handling the first case are as follows.

1. Take the ratio of the lengths of the two series.

2. Take the whole number part of the ratio. For example if the ratio is 2.5, then the whole number part will be 2.

3. In the smaller series, take a data point and a data point next to it.

4. Add them then divide them by the whole number.

5. Add this quotient to the data point, the whole number less one times.

6. Take the decimal value. For example if the data point is 2.5, then the decimal value is 0.5.

7. In every iteration, add the decimal value to itself then check if it is greater than one. If so the number of times you will add the quotient will be done one more time.

The second case is handled by performing the moving average filter n times until it matches the length. This is also called the Multiple-Pass Moving Average Filter.

E. Dynamic Time Warping

Dynamic Time Warping is a non-linear alignment algorithm that produces an intuitive similarity measure which allows similar shapes to match even if they are out of phase in the time axis. The warping function finds the best alignment in which the distance is minimized between time series A and time series B.

The first step in DTW in creating a matrix. This matrix contains the difference between each element in time series A and time series B. The second step is squaring these differences to keep symmetry intact with the goal being to find the shortest distance from $(0, 0)$ to (m, n).

There are restrictions the researchers can add to the warping function in order to reduce the search space which are the following:

1. Monotonicity

   The alignment path does not go back in the time index. This restriction guarantees that features are not repeated in the alignment.

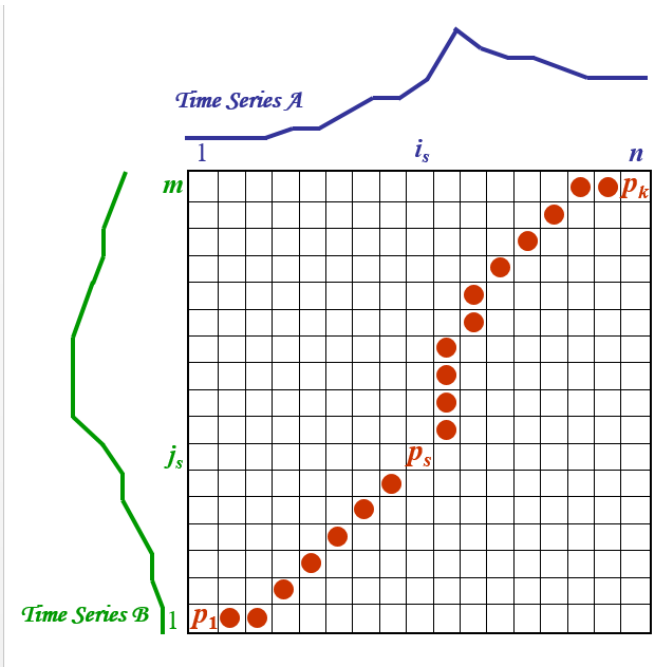


Figure 4: The Warping Function P

2. Continuity

   The alignment path does not jump in the time index. This restriction guarantees that the alignment does not omit important features.

3. Boundary Conditions

   The alignment path starts at the bottom left and ends at the top right. This restriction guarantees that the alignment does not consider partially one of the sequences.

4. Warping Window

   Restrict the alignment path to be close to the diagonal because a good alignment path does not wander too far from the diagonal. This restriction guarantees that the alignment does not try to skip different features and get stuck at similar ones.

5. Slope Constraint

   This suggests that the alignment path should not be too steep or shallow. This

restriction prevents the very short paths to be matched with very long ones.

IV. Design and Implementation

A. Use Case Diagram

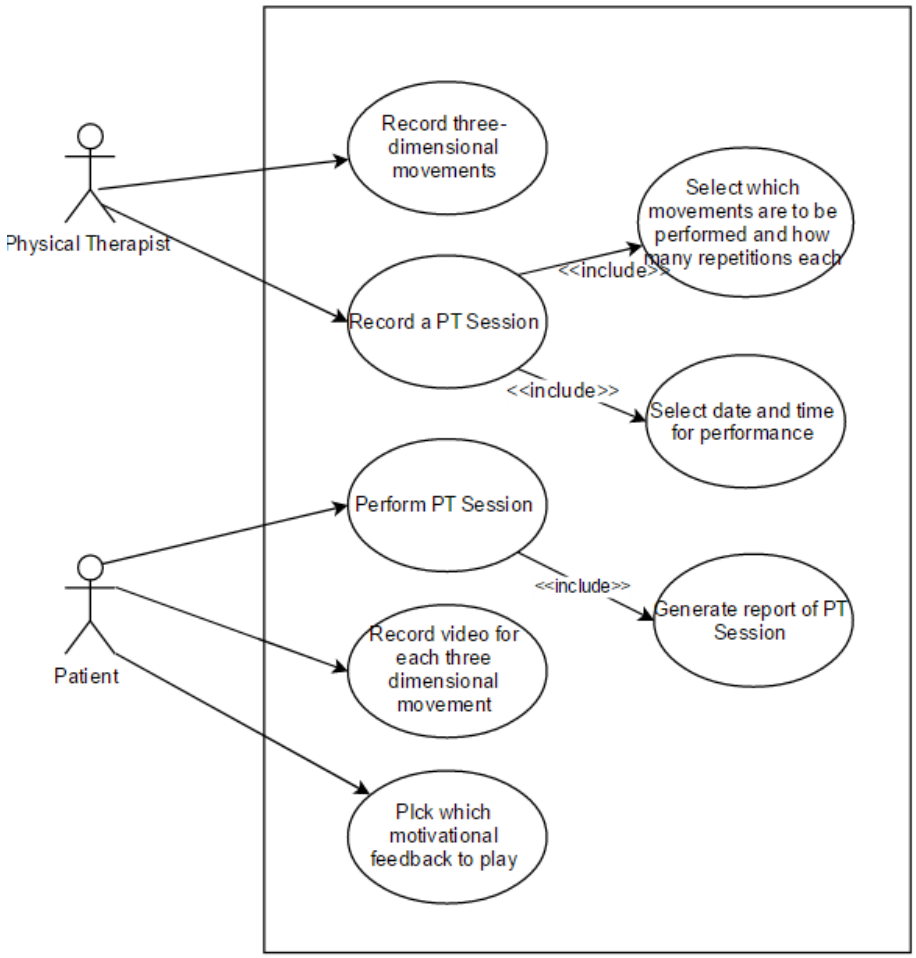


Figure 5: Use Case Diagram

There are two distinct actors which take part in the system. The physical therapist stores the gestures that the patient must perform. The physical therapist also creates the sequence of gestures and schedules when the patient must perform it. The patient performs the session created by the physical therapist, this results into the generation of a report which is interpreted by the physical therapist as well. The user stores videos for the gestures that he will be performing in case he forgets how they are done. The patient picks which motivational feedback will be played upon the performance

of each gesture.

B. Flowchart

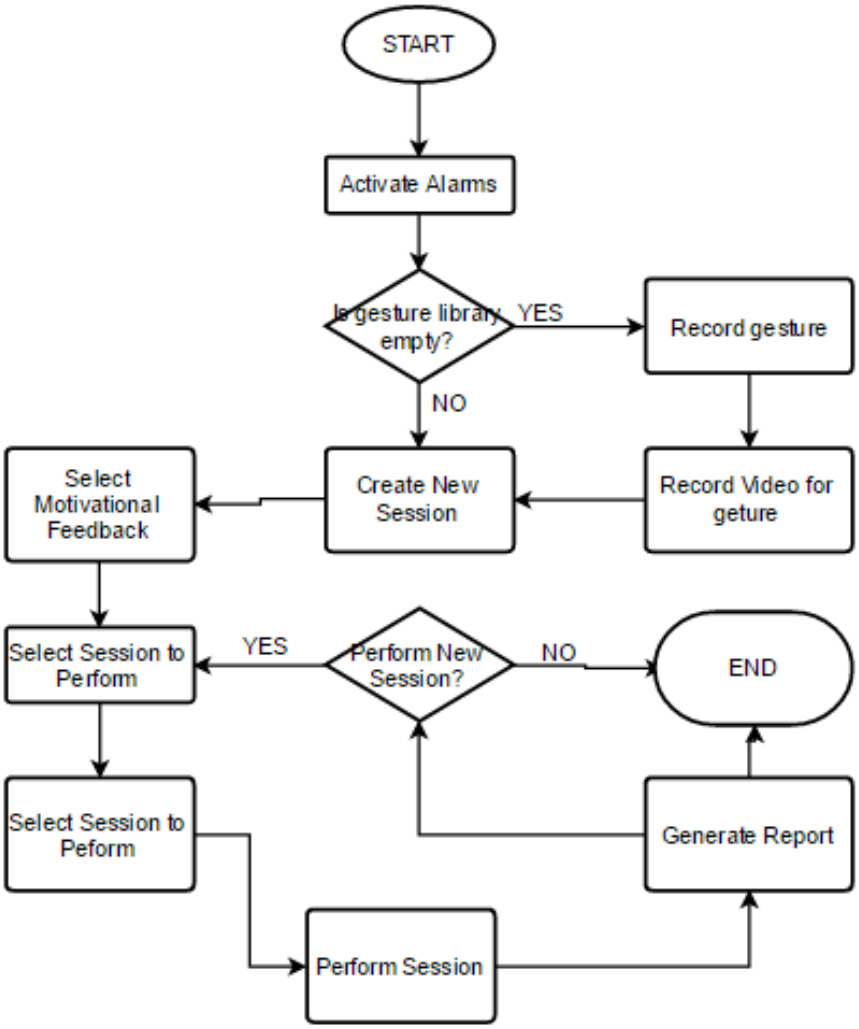


Figure 6: Flowchart

The application performs the first process which is to activate all pending alarms. The only alarms active are those that will play in the future. The application then checks if the gesture library is empty. When it is empty, it must be populated. The patient records videos for each gesture in the library. The physical therapist creates

sessions using the gestures stored in the library and afterwards the patients perform them. The tool generates a report every time a session is performed.

C. Block Diagram for Gesture Recognition

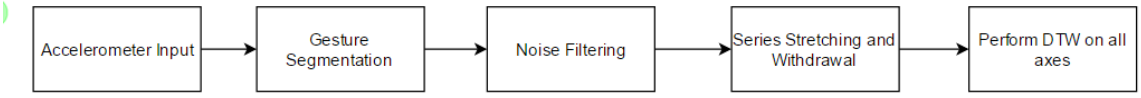


Figure 7: Block Diagram

The process of gesture recognition begins with the accelerometer input which is a set of data points. The accelerometer has three axes making three separate data points. The system segments the accelerometer inputs manually to get the data points for the gesture. It then filter this data through a noise filtering algorithm which is the moving average filter for each axes. Series stretching and withdrawal is done to make those series being compared equal in length. Dynamic Time Warping is performed to provide the minimum distance which will be the basis in determining how close the two series are.

V. Results

1. Main Menu

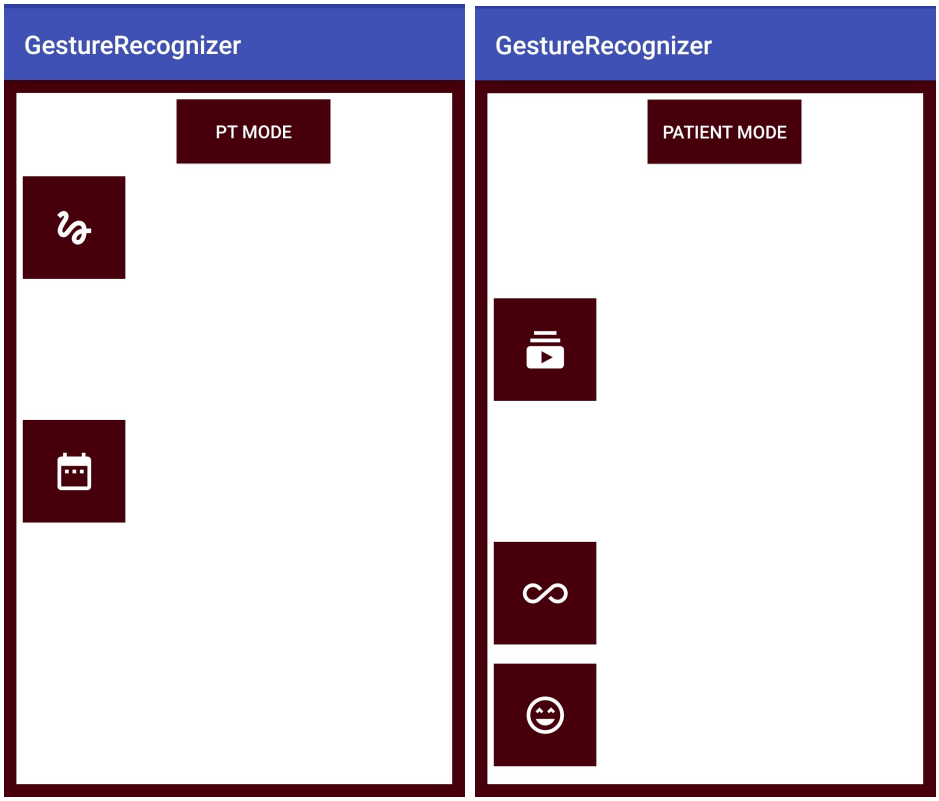


Figure 8: Main Menu

Upon starting the application, the main menu will be displayed. There user can switch between modes which will result in the division of functionalities.

2. Recording a Gesture

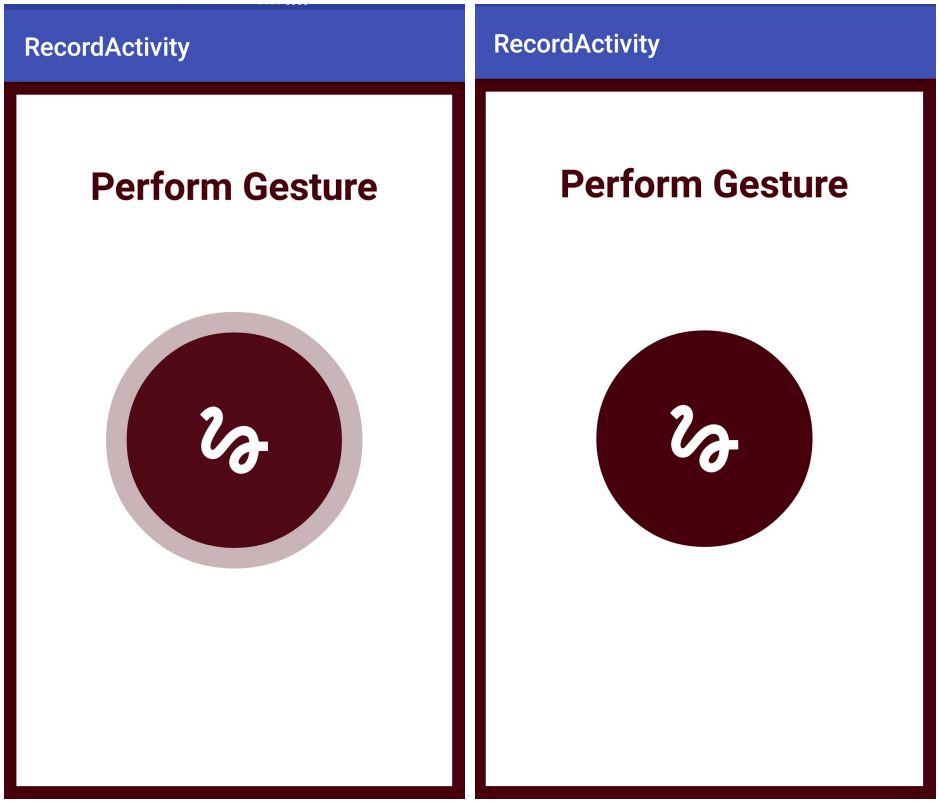


Figure 9: Recording a Gesture

While in PT mode, the user can record gestures by clicking the button with the gesture logo. A new activity will begin where the user will be prompted to press the button with the gesture logo to mark the beginning of the recording and releasing it will mark its end.

3. Moving Average Filter

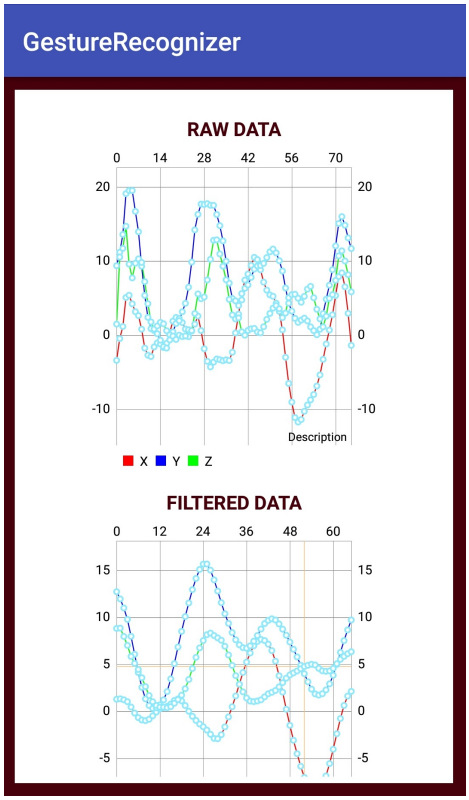


Figure 10: Data Graphs

After recording the gesture, the user can view the raw and filtered data collected through the accelerometer in graph form.

4. Saving the Gesture

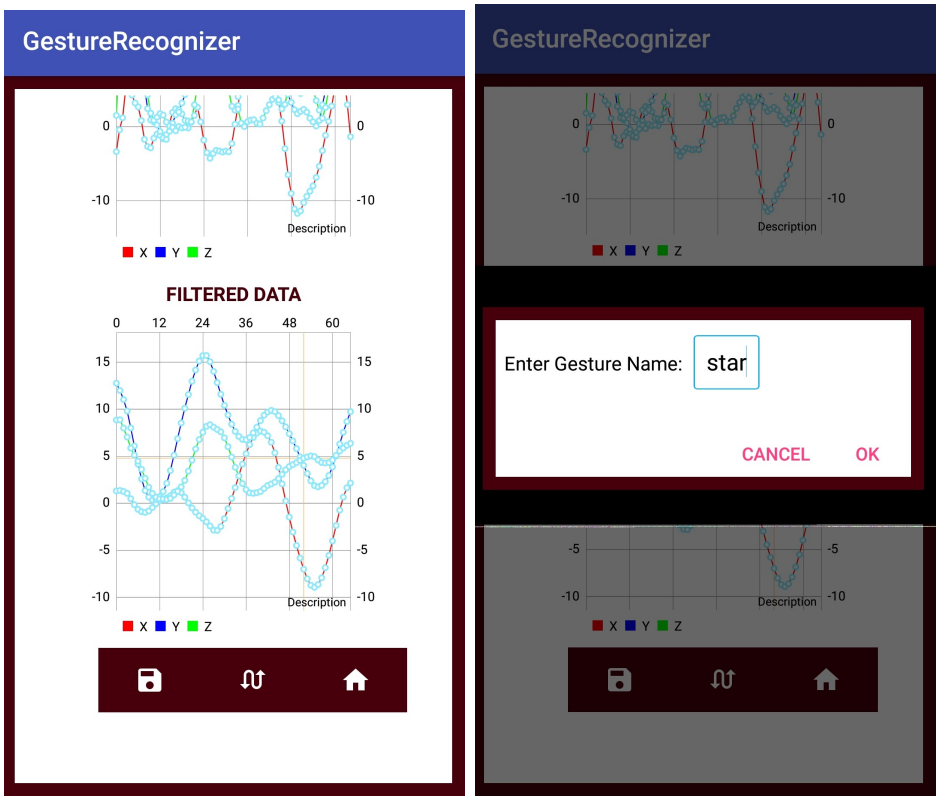


Figure 11: Saving the Gesture

By clicking the button that has a floppy disk logo, you will be prompted to enter the name of the gesture you want to store. Pressing the ok button will cause the said button to be rendered invisible.

5. Gesture Recognition

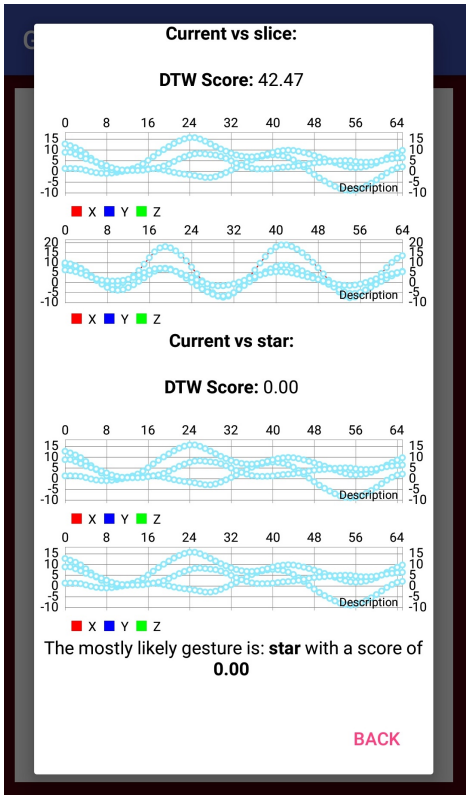


Figure 12: Gesture Recognition

By clicking the two-way button, a dialog will pop-up showing DTW scores by the gesture you currently performed against those of which you have stored. It will also show you the filtered graphs for both of them. The last line will tell you which one has the least DTW score hailing it as the most likely gesture. The user can return to the main menu by pressing the button with the home logo.

6. Viewing the Recorded Gestures

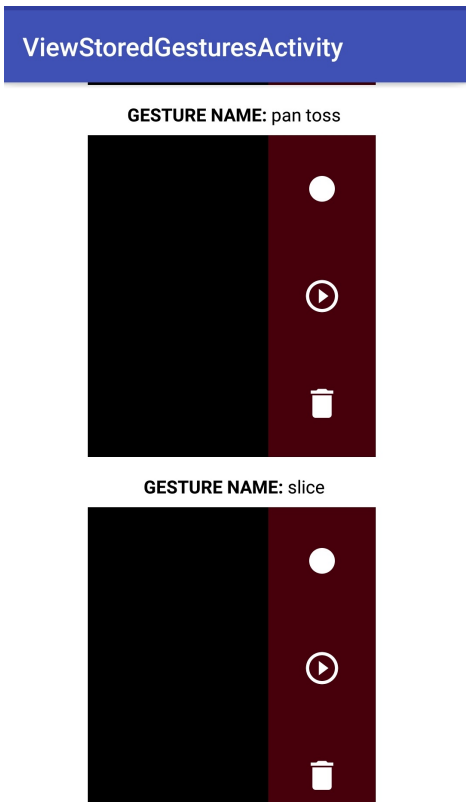


Figure 13: Viewing the Recorded Gestures

While in Patient Mode, the user can record videos by clicking the button with the record logo in it. Playing the video will require the user to press the button with play logo on it. The delete button will only be available if the gesture is not needed in any of the sessions.

7. Creating a Session

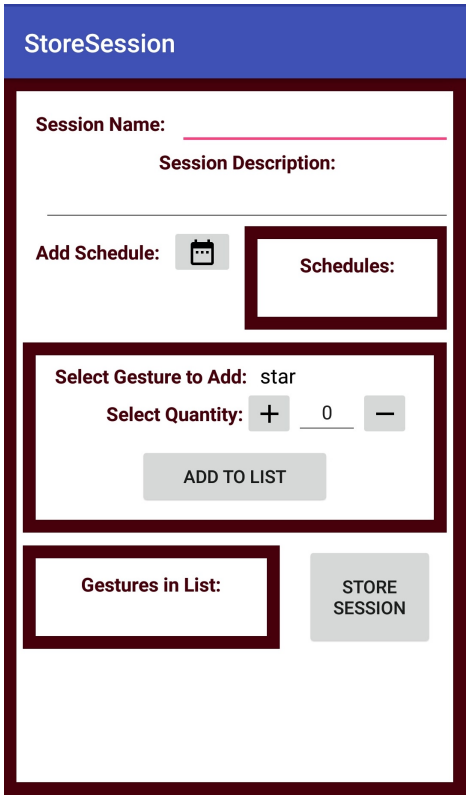


Figure 14: Creating a Session

While in PT Mode, the user can create a session by filling up the fields which include the session name, session description, the schedule, and the gestures to be performed in the session with their corresponding repetitions.

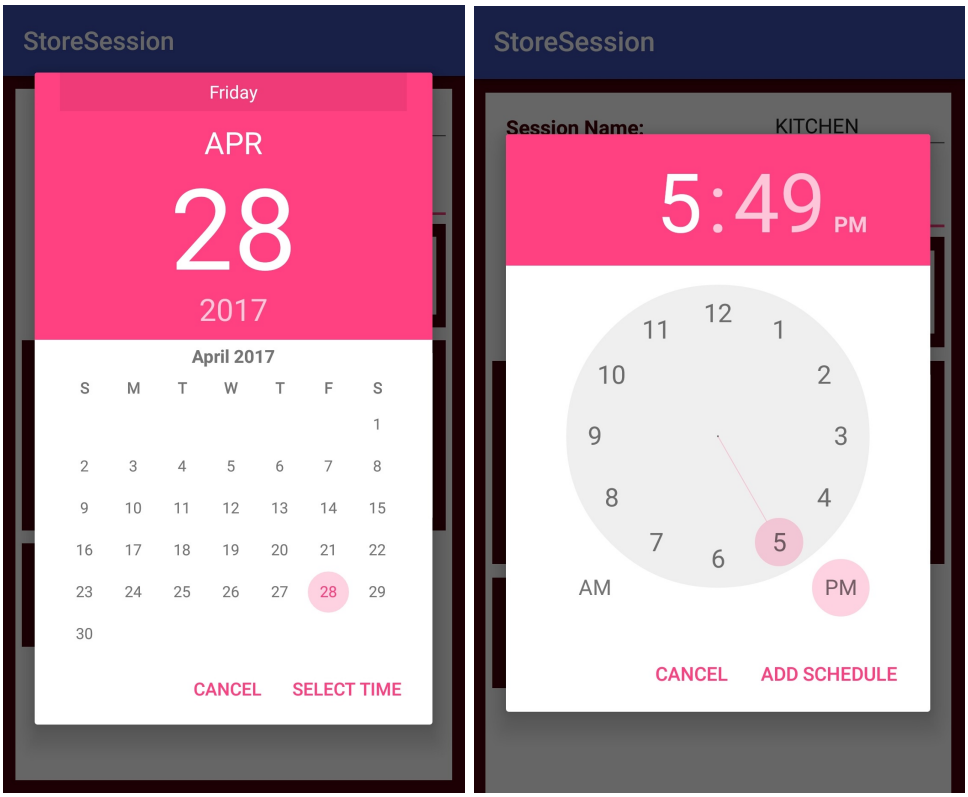


Figure 15: Recording the Schedule

The user can create multiple schedules on which an alarm will set off to remind the user to perform the session whenever the application is running.

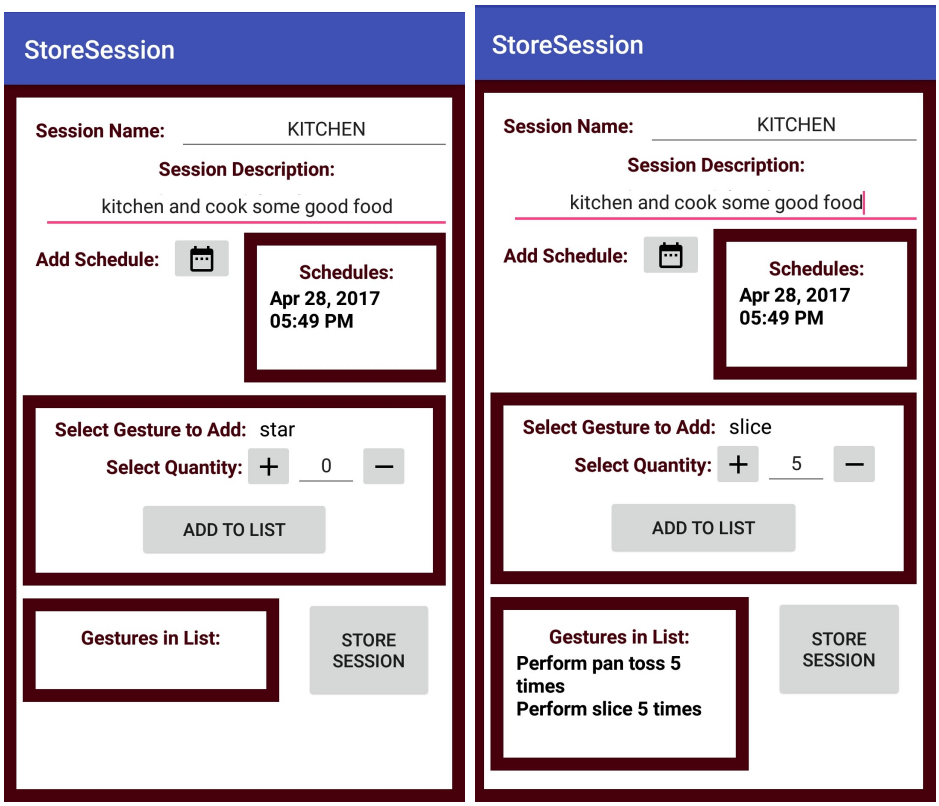


Figure 16: Recording the Exercises

The user can add multiple gestures with their corresponding repetitions. The order in which they will be performed is the same as how they have been added. Clicking the store session button will create the session provided that all the fields are filled up.

8. Performing Sessions

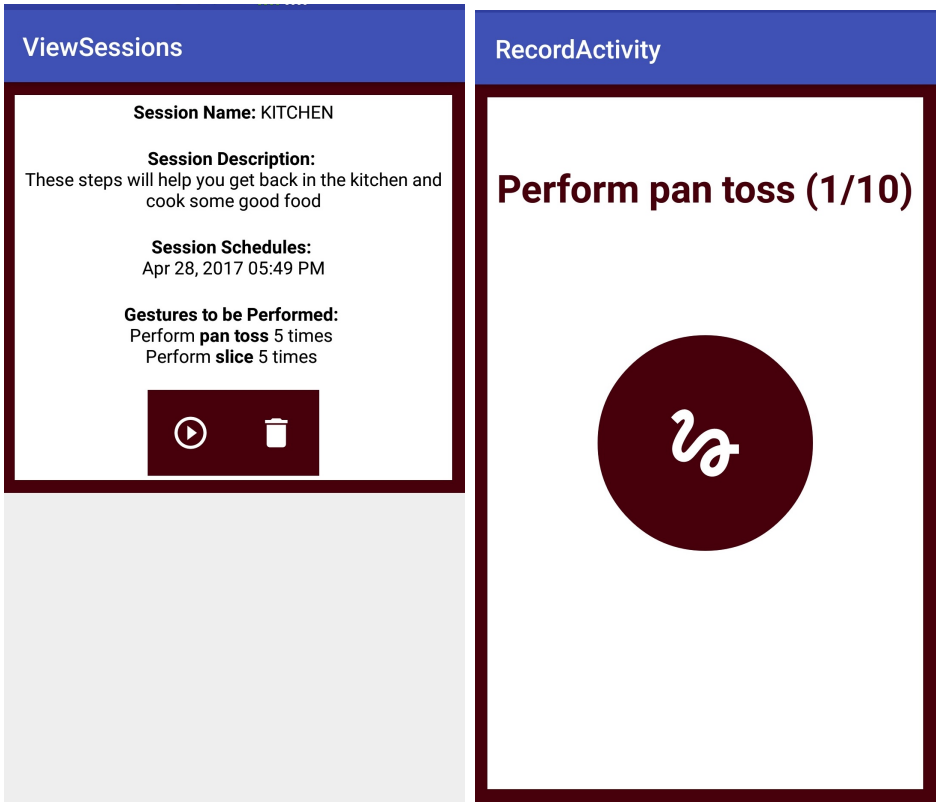
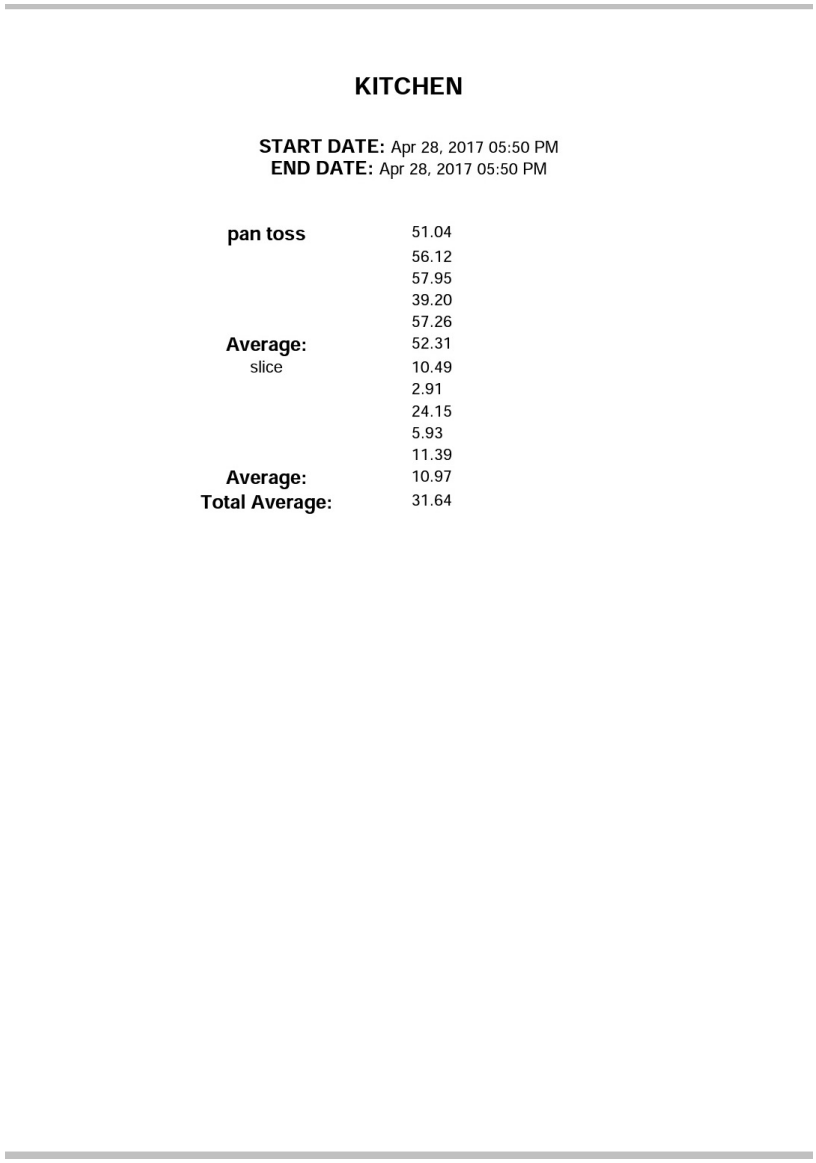


Figure 17: Performing Sessions

While in Patient Mode, the user can perform the sessions created by clicking the button with the play logo on it. Sessions can be deleted by clicking their corresponding delete button. Upon clicking the play button, a new activity will start where the user will be told of which gesture he should perform. The denominator in the fraction represents the total number of gestures you will have to perform.

9. Viewing Results



KITCHEN

START DATE: Apr 28, 2017 05:50 PM
END DATE: Apr 28, 2017 05:50 PM

pan toss	51.04
	56.12
	57.95
	39.20
	57.26
Average:	52.31
slice	10.49
	2.91
	24.15
	5.93
	11.39
Average:	10.97
Total Average:	31.64

Figure 18: Viewing Results

After performing the gestures in the session. The report about the session which is in pdf form will be generated and can be viewed in the documents folder in the internal storage of the phone.

VI. Discussion

GRAPT is an application that aims to help physical therapists monitor their patients progress in rehabilitation because they can't always be with their patients. The sequence of exercises involve multiple repetitions and the report this application produces guides them in checking if they are nearing the end of their rehabilitation. It involves major functions such as (1) recording a gesture, (2) viewing the stored gestures, (3) creating a session, (4) viewing all available sessions, and (5) performing a session to achieve this goal which the application allows you to do.

Movements are stored in the form of accelerometer signals and because of the time series nature of these signals, we are able to use Dynamic Time Warping to compare them. Physical therapists rely on intuition to determine if a movement was properly performed by the patient. They do not have any metric or standard to determine how different two movements are. The DTW score serves as the starting point of research in creating this missing standard.

The whole gesture recognition process is composed of multiple steps. The system first segments the data from the accelerometer. It then performs the matching of series lengths to be able to compare them without bias. It then filters the data using the moving average filter. Lastly, it uses DTW to compare the filtered scores.

These steps are repeated for each gesture in each session. Each session is composed of the date, time, movements performed and how many times each movement is repeated. The alarm module alerts the user when the date and time is upon it. The performance of each session generates a report. Android studio provides you with all the tools in creating these sequences.

The whole application was coded using Android Studio since it was able to access the accelerometer sensor of the smartphone. Multiple serializable classes were made so as to store the recorded gestures and sessions. Smartphones are imperfect and subject to noise which is why the moving average filter was applied to reduce noise.

Android studio allowed the user to pick how sensitive the system would be, or how much data the system will read per a given amount of time. The one picked by the researcher is the mode which produces the most amount of data to properly represent the gesture. The application reads with this configuration in all phones in which it is installed to provide uniformity. The gestures recorded in one phone may not be properly compared with those stored in another since one phone may have a higher potential in sensitivity than the other. The gestures stored and read in the same phone is compared properly. The analogy can be viewed as how two trees produce the same fruit, and the fruit produced by both those trees can be compared perfectly with fruits produced within themselves but not with the fruit produced by another tree. This again, is due to the fact that as the accelerometers in phones improve over time and produce better sensitivity.

VII. Conclusion

GRAPT has been developed to help physical therapists and their patients in their goal of rehabilitation. The smartphone, being an affordable and accessible tool, was utilized in this goal. It's ability to access the accelerometer hardware made it the top candidate for this research.

Due to the nature to the accelerometer input data being a times series while being able to describe performed movements, it is the tool used to collect these data. Serialization to files were used in storing the data read. Having them stored, the tool is able to compare it with other accelerometer input through a technique called Dynamic Time Warping. This comparison is called gesture recognition.

DTW provides you with scores which can be used to determine how close or far, how proper or improper, the movement the patient performs is. The DTW technique is not subject to bias of the human eye which will make it a good candidate for the establishment of standards in Physical Therapy.

The problem of not being able to be with the patient with all times is solved through the smartphone application. It allows the therapist to monitor the patient's progress through time by generating reports which the therapist will be able to interpret.

As the therapist interprets this, he will be able to see trends or establish patterns which would result in standards to rate or check how correctly performed a movement is. This will forever change Physical Therapy, as it is known.

VIII. Recommendations

The work has not yet tapped into commercialization via the web. In order to do this, the researcher must have an online portal to which this gestures will be stored. This will allow other works such as those of machine learning since data sets can be obtained. Furthermore, it will be more fault-tolerant since your records will have a backup copy in the cloud.

Another good technique will be to also use the gyroscope function of the smartphone with that of the accelerometer while adding weights to their outputs. Better checks and balances will be done with this addition.

Developments in hardware can result to the smartphone being inferior. If data gloves which can read accelerometer will be commercially acccessible then it will be a better substitute because the user will no longer have to hold the phone, he will be able to wear it. It makes it more convenient.

Lastly, since this study is restricted to hand movements, the researcher may also focus other kinds of movements such as movements in the legs, waist, or a combination of all of them. This studies will further help in the automation of Physical Therapy.

IX. Bibliography

[1] S. Haug, R. P. Castro, M. Kwon, A. Filler, T. Kowatsch, and M. P. Schaub, "Smartphone use and smartphone addiction among young people in switzerland," *Journal of behavioral addictions*, vol. 4, no. 4, pp. 299–307, 2015.

[2] A. Campbell and T. Choudhury, "From smart to cognitive phones," *IEEE Pervasive Computing*, vol. 3, no. 11, pp. 7–11, 2012.

[3] "A beginners guide to the accelerometer." http://www.dimensionengineering.com/info/accelerometers. Accessed: 2010-10-11.

[4] L.-M. Wu, J.-S. Sheu, W.-C. Jheng, and Y.-T. Hsiao, "Pedometer development utilizing an accelerometer sensor," in *Proceedings of World Academy of Science, Engineering and Technology*, no. 79, p. 35, World Academy of Science, Engineering and Technology (WASET), 2013.

[5] S. K. Liddell and M. Metzger, "Gesture in sign language discourse," *Journal of Pragmatics*, vol. 30, no. 6, pp. 657–697, 1998.

[6] L. E. Sucar, R. Luis, R. Leder, J. Hernández, and I. Sánchez, "Gesture therapy: A vision-based system for upper extremity stroke rehabilitation," in *2010 Annual International Conference of the IEEE Engineering in Medicine and Biology*, pp. 3690–3693, IEEE, 2010.

[7] L. E. Sucar, F. Orihuela-Espina, R. L. Velazquez, D. J. Reinkensmeyer, R. Leder, and J. Hernández-Franco, "Gesture therapy: An upper limb virtual reality-based motor rehabilitation platform," *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 22, no. 3, pp. 634–643, 2014.

[8] R. P. Van Peppen, G. Kwakkel, S. Wood-Dauphinee, H. J. Hendriks, P. J. Van der Wees, and J. Dekker, "The impact of physical therapy on functional outcomes after stroke: what's the evidence?," *Clinical rehabilitation*, vol. 18, no. 8, pp. 833–862, 2004.

[9] A. Sunderland, D. Tinson, E. Bradley, D. Fletcher, R. L. Hewer, and D. Wade, "Enhanced physical therapy improves recovery of arm function after stroke. a randomised controlled trial.," *Journal of Neurology, Neurosurgery & Psychiatry*, vol. 55, no. 7, pp. 530–535, 1992.

[10] M. Ketelaar, A. Vermeer, H. Hart, E. van Petegem-van Beek, and P. J. Helders, "Effects of a functional therapy program on motor abilities of children with cerebral palsy," *Physical Therapy*, vol. 81, no. 9, pp. 1534–1545, 2001.

[11] F. B. Palmer, B. K. Shapiro, R. C. Wachtel, M. C. Allen, J. E. Hiller, S. E. Harryman, B. S. Mosher, C. L. Meinert, and A. J. Capute, "The effects of physical therapy on cerebral palsy," *New England Journal of Medicine*, vol. 318, no. 13, pp. 803–808, 1988.

[12] M. E. Morris, "Movement disorders in people with parkinson disease: a model for physical therapy," *Physical therapy*, vol. 80, no. 6, pp. 578–597, 2000.

[13] L. A. Simpson, J. J. Eng, J. T. Hsieh, Wolfe, and D. L. the Spinal Cord Injury Rehabilitation Evidence (SCIRE) Research Team, "The health and life priorities of individuals with spinal cord injury: a systematic review," *Journal of neurotrauma*, vol. 29, no. 8, pp. 1548–1555, 2012.

[14] "Muscle memory from a coach's perspective." http://www.dna-sports-performance.com/muscle-memory-a-coaches-perspective/. Accessed: 2010-10-11.

[15] R. Xu, S. Zhou, and W. J. Li, "Mems accelerometer based nonspecific-user hand gesture recognition," *IEEE sensors journal*, vol. 12, no. 5, pp. 1166–1173, 2012.

[16] Z. Ji, Z.-Y. Li, P. Li, and M. An, "A new effective wearable hand gesture recognition algorithm with 3-axis accelerometer," in *Fuzzy Systems and Knowledge Discovery (FSKD), 2015 12th International Conference on*, pp. 1243–1247, IEEE, 2015.

[17] R. S. Leder, G. Azcarate, R. Savage, S. Savage, L. E. Sucar, D. Reinkensmeyer, C. Toxtli, E. Roth, and A. Molina, "Nintendo wii remote for computer simulated arm and wrist therapy in stroke survivors with upper extremity hemipariesis," in *2008 Virtual Rehabilitation*, pp. 74–74, IEEE, 2008.

[18] R. Colombo, F. Pisano, A. Mazzone, C. Delconte, S. Micera, M. C. Carrozza, P. Dario, and G. Minuco, "Design strategies to improve patient motivation during robot-aided rehabilitation," *Journal of neuroengineering and rehabilitation*, vol. 4, no. 1, p. 1, 2007.

[19] Y. Chen, B. Luo, Y.-L. Chen, G. Liang, and X. Wu, "A real-time dynamic hand gesture recognition system using kinect sensor," in *2015 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pp. 2026–2030, IEEE, 2015.

[20] R. Montejo, "Image navigation and manipulation in the operating room using hand gesture recognition," *UP Manila Journal*, 2014.

[21] R. Xie and J. Cao, "Accelerometer-based hand gesture recognition by neural network and similarity matching," *IEEE Sensors Journal*, vol. 16, no. 11, pp. 4537–4545, 2016.

[22] D. Jayaraman and K. Vanitha, "Nonspecific-user hand gesture recognition by using mems accelerometer," in *Information Communication and Embedded Systems (ICICES), 2014 International Conference on*, pp. 1–6, IEEE, 2014.

[23] A. Akl, C. Feng, and S. Valaee, "A novel accelerometer-based gesture recognition system," *IEEE Transactions on Signal Processing*, vol. 59, no. 12, pp. 6197–6205, 2011.

[24] T. Marasović and V. Papić, "Accelerometer-based gesture classification using principal component analysis," in *Software, Telecommunications and Computer Networks (SoftCOM), 2011 19th International Conference on*, pp. 1–5, IEEE, 2011.

[25] H. P. Gupta, H. S. Chudgar, S. Mukherjee, T. Dutta, and K. Sharma, "A continuous hand gestures recognition technique for human-machine interaction using accelerometer and gyroscope sensors," *IEEE Sensors Journal*, vol. 16, no. 16, pp. 6425–6432, 2016.

X. Appendix

A. Source Code

```
package com.example.sirwheistein.gesturerecognizer;

import android.content.Context;
import android.widget.Toast;

import java.math.BigDecimal;
import java.util.ArrayList;

/**
 * Created by Sir.Wheistein on 3/11/2017.
 */

public class FilterClass {

    /*public float[] MovingAverageFilter(float[] series, float alpha) {
        float[] newSeries = new float[series.length];
        newSeries[0] = series[0];
        for(int i = 1; i < series.length; i++) {
            newSeries[i] = newSeries[i-1] + alpha * (series[i-1] - newSeries[i-1]);
        }
        return newSeries;
    }*/

    // array1 yung pagcocomparean or yung orig
    public float[] matchArrayLengthsViaDuplication(float[] array1, float[] array2, Context context) {
        double ratio = (double) array1.length / array2.length;
        ArrayList<Float> newarr = new ArrayList<>();
        //Toast.makeText(context, "FLOAT: " + ratio, Toast.LENGTH_LONG).show();

        if(ratio >= 1.0f) {
            int front = array1.length / array2.length;
            double back = ratio - Math.floor(ratio);

            double moving_back = 0f;
            for(int i = 0; i < array2.length -1; i++) {
                newarr.add(array2[i]);
                moving_back = moving_back + back;
                if(front >= 1) {
                    float divisions = front;
                    if(moving_back > 1f) {
                        divisions = divisions +1;
                        moving_back = moving_back - 1f;
                    }
                    for (int j = 1; j < divisions; j++) {
                        newarr.add(array2[i] + (j * (array2[i+1] - array2[i]) / divisions));
                    }
                }
            }
            newarr.add(array2[array2.length -1]);
            return convertFloat(newarr);
        } else {
            float[] dummy = array2;
            while(dummy.length >= array1.length) {
                dummy = MovingAverageFilter(dummy, 2);
            }
            return dummy;
        }
    }

    public float[] convertFloat(ArrayList<Float> list) {
        float[] newarr = new float[list.size()];
        for(int i = 0; i < list.size(); i++) {
            newarr[i] = list.get(i);
        }
        return newarr;
    }

    public float[] MovingAverageFilter(float[] series, int window) {
        try {
            /*Toast.makeText(context,
                    "SERIES LENGTH: " + series.length + " NEW SERIES LENGTH: " +
                            (series.length - Math.round(series.length * 0.1f) +1) + "", Toast.LENGTH_LONG).show();*
            //int newlength = series.length - Math.round(series.length * 0.1f) +1;
            //Toast.makeText(context, filteredseries.length + "", Toast.LENGTH_LONG).show();
            float[] newseries = new float[series.length - window +1];
            for (int i = 0; i < newseries.length; i++) {
                newseries[i] = 1f;
                float temp = 0f;
                for (int j = 0; j < window; j++) {
                    temp = temp + series[i + j];
                }
```

```
                newseries[i] = temp / window;
            }
            return newseries;
        } catch (Exception e) {
            //Toast.makeText(context, e + "", Toast.LENGTH_LONG).show();
        }
        return series;
    }


    public float[] MovingAverageFilter(float[] series, Context context) {
        try {
            int window = Math.round(series.length * 0.15f);
            /*Toast.makeText(context,
                    "SERIES LENGTH: " + series.length + " NEW SERIES LENGTH: " +
                            (series.length - Math.round(series.length * 0.1f) +1) + "", Toast.LENGTH_LONG).show();*
            //int newlength = series.length - Math.round(series.length * 0.1f) +1;
            //Toast.makeText(context, filteredseries.length + "", Toast.LENGTH_LONG).show();
            float[] newseries = new float[series.length - window +1];
            for (int i = 0; i < newseries.length; i++) {
                newseries[i] = 1f;
                float temp = 0f;
                for (int j = 0; j < window; j++) {
                    temp = temp + series[i + j];
                }
                newseries[i] = temp / window;
            }
            return newseries;
        } catch (Exception e) {
            Toast.makeText(context, e + "", Toast.LENGTH_LONG).show();
        }
        return series;
    }
}
```

```
package com.example.sirwheistein.gesturerecognizer;

import android.content.Context;
import android.os.Environment;
import android.widget.Toast;

import java.io.File;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;
import java.util.UUID;

import static android.content.Context.MODE_PRIVATE;

/**
 * Created by Sir.Wheistein on 3/15/2017.
 */

public class GestureFileHandler {

    private String fileName;

    public GestureFileHandler(String filename) {
        this.fileName = filename;
    }

    public void storeNewGesture(GestureRecord record, Context context) {
        if(record != null) {
            GestureRecord[] prevRecords = getStoredGestures(context);
            GestureRecord[] new_rec = new GestureRecord[prevRecords.length + 1];
            for (int i = 0; i < prevRecords.length; i++) {
                new_rec[i] = prevRecords[i];
            }
            new_rec[new_rec.length -1] = record;
            save(new_rec, context);
        }
    }

    public void save(GestureRecord[] rec, Context context) {
        try {
            FileOutputStream fs = context.openFileOutput(fileName, MODE_PRIVATE);
            ObjectOutputStream os = new ObjectOutputStream(fs);
            os.writeObject(rec);
            os.close();
        }
        catch(Exception e) {
            Toast.makeText(context, "SAVE GFH" + e.toString(), Toast.LENGTH_SHORT).show();
        }
    }

    public GestureRecord[] getStoredGestures(Context context) {
        GestureRecord[] records = new GestureRecord[0];
        try {
            FileInputStream fi = context.openFileInput(fileName);
```

```
            ObjectInputStream is = new ObjectInputStream(fi);
            records = (GestureRecord[]) is.readObject();
            is.close();
        }
        catch(Exception e) {
            Toast.makeText(context, "LOAD GFH" + e.toString(), Toast.LENGTH_SHORT).show();
        }
        return records;
    }

    public String generateRandomID() {
        String uniqueID = UUID.randomUUID().toString();
        return uniqueID;
    }

    public GestureRecord getRecordByID(String id, Context context) {
        GestureRecord[] list = getStoredGestures(context);
        for(int i = 0; i < list.length; i++) {
            if(id.equals(list[i].getGestureID())) {
                return list[i];
            }
        }
        return null;
    }

    public void DeleteAtIndex(int index, Context context) {
        GestureRecord[] list = getStoredGestures(context);
        GestureRecord[] newList = new GestureRecord[list.length -1];
        for(int i = 0; i < index; i++) {
            newList[i] = list[i];
        }
        for(int i = index +1; i < list.length; i++) {
            newList[i-1] = list[i];
        }
        save(newList, context);
    }

    public void DeleteFile(Context context) {
        context.deleteFile(fileName);
    }

}


package com.example.sirwheistein.gesturerecognizer;

import java.io.Serializable;

/**
 * Created by Sir.Wheistein on 3/15/2017.
 */

public class GestureRecord implements Serializable{

    private String gestureName;
    private String gestureID;
    private float[] x_accel;
    private float[] y_accel;
    private float[] z_accel;

    //private String videoUri;

    public GestureRecord(String gestureName, String gestureID,
                         float[] x, float[] y, float[] z) {
        this.gestureName = gestureName;
        this.gestureID = gestureID;
        x_accel = x;
        y_accel = y;
        z_accel = z;
    }

    public String getGestureName() {
        return gestureName;
    }

    public String getGestureID() {
        return gestureID;
    }

    public float[] getX_accel() {
        return x_accel;
    }

    public float[] getY_accel() {
        return y_accel;
    }

    public float[] getZ_accel() {
        return z_accel;
    }

    /*
```

```
    public void setUri(String vidUri) {
        videoUri = vidUri;
    }

    public String getVideoUri() {
        return videoUri;
    }*/
}


package com.example.sirwheistein.gesturerecognizer;

import android.app.Dialog;
import android.content.Context;
import android.content.DialogInterface;
import android.content.Intent;
import android.graphics.Color;
import android.support.v7.app.AlertDialog;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.text.Html;
import android.view.Gravity;
import android.view.View;
import android.widget.EditText;
import android.widget.ImageButton;
import android.widget.LinearLayout;
import android.widget.ScrollView;
import android.widget.TextView;
import android.widget.Toast;

import com.github.mikephil.charting.charts.LineChart;
import com.github.mikephil.charting.data.Entry;
import com.github.mikephil.charting.data.LineData;
import com.github.mikephil.charting.data.LineDataSet;
import com.github.mikephil.charting.interfaces.datasets.ILineDataSet;

import java.util.ArrayList;

public class GraphActivity extends AppCompatActivity {

    private Context myContext;
    private GestureFileHandler fh;

    private float[] x_movement;
    private float[] y_movement;
    private float[] z_movement;

    private ImageButton storeButton;
    private ImageButton compareButton;
    private ImageButton mainMenuButton;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_graph);

        myContext = this.getApplicationContext();
        fh = new GestureFileHandler ("ges.ser");

        Bundle bundle = getIntent().getExtras();
        x_movement = bundle.getFloatArray("x_values");
        y_movement = bundle.getFloatArray("y_values");
        z_movement = bundle.getFloatArray("z_values");

        rawGraph();
        LPFGraph();

        storeButton = (ImageButton) findViewById(R.id.Store);
        storeButton.setOnClickListener(new View.OnClickListener() {

            public void onClick(View v) {
            try {

                LinearLayout ll = new LinearLayout(myContext);
                ll.setOrientation(LinearLayout.HORIZONTAL);
                ll.setPadding(50, 50, 50, 50);
                final EditText ges_name = new EditText(myContext);
                ges_name.setTextColor(Color.BLACK);
                TextView prompt = new TextView(myContext);
                prompt.setTextColor(Color.BLACK);
                prompt.setText("Enter Gesture Name: ");
                ll.addView(prompt);
                ll.addView(ges_name);

                AlertDialog.Builder builder = new AlertDialog.Builder(GraphActivity.this);
                builder.setView(ll)
                .setCancelable(false)
                .setPositiveButton("OK",
                        new DialogInterface.OnClickListener() {
                            public void onClick(DialogInterface dialog, int id) {
                                // raw data ang isave
                                FilterClass fc = new FilterClass();
```

```
                                GestureRecord[] prevRecord = fh.getStoredGestures(myContext);
                                GestureRecord rec = new GestureRecord(ges_name.getText()+"", fh.generateRandomID(),
                                        x_movement, y_movement, z_movement);
                                fh.storeNewGesture(rec, GraphActivity.this);
                                Toast.makeText(myContext, ges_name.getText()+"", Toast.LENGTH_SHORT).show();
                                storeButton.setVisibility(View.INVISIBLE);
                            }
                        })
                .setNegativeButton("CANCEL",
                        new DialogInterface.OnClickListener() {
                            public void onClick(DialogInterface dialog, int id) {
                            }
                        });

                Dialog dialog = builder.create();
                dialog.getWindow().setBackgroundDrawableResource(R.drawable.border_background);
                dialog.show();
            }
            catch(Exception e) {
                Toast.makeText(GraphActivity.this, e.toString(), Toast.LENGTH_SHORT).show();
            }
        }
    });

        compareButton = (ImageButton) findViewById(R.id.Compare);
        compareButton.setOnClickListener(new View.OnClickListener() {

            public void onClick(View v) {
                try {
                    /*GestureRecord[] stored = fh.getStoredGestures(myContext);

                    String temp = "";

                    for(int i = 0; i < stored.length; i++) {
                        temp = temp + stored[i].getGestureName();
                        temp = temp + "" + stored[i].getX_accel().length;
                    }

                    double low = 1000;
                    int index = 0;
                    for(int i = 0; i < stored.length; i++) {
                        if(getWarpDistance(stored[i]) < low) {
                            low = getWarpDistance(stored[i]);
                            index = i;
                        }
                    }

                    LinearLayout ll = new LinearLayout(myContext);
                    ll.setOrientation(LinearLayout.HORIZONTAL);
                    ll.setPadding(50, 50, 50, 50);
                    TextView message = new TextView(myContext);
                    message.setTextColor(Color.BLACK);
                    message.setText(Html.fromHtml("Most Likely Gesture is <b>" + stored[index].getGestureName() +
                            "</b> with DTW score of " + Math.round(getWarpDistance(stored[index]) * 100.0) / 100.0)
                    message.setGravity(Gravity.CENTER);
                    ll.addView(message);

                    AlertDialog.Builder builder = new AlertDialog.Builder(GraphActivity.this);
                    builder.setView(ll)
                    .setCancelable(true);

                    Dialog dialog = builder.create();
                    dialog.show();

                    //Toast.makeText(GraphActivity.this, getWarpDistance(stored[0]) + "", Toast.LENGTH_SHORT).show(
                    compareAndShow();
                }
                catch(Exception e) {
                    Toast.makeText(GraphActivity.this, e.toString(), Toast.LENGTH_SHORT).show();
                }
           }
        });

        mainMenuButton = (ImageButton) findViewById(R.id.main_menu_button);
        mainMenuButton.setOnClickListener(new View.OnClickListener() {

            public void onClick(View v) {
                Intent newActivity = new Intent(myContext, MainActivity.class);
                newActivity.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP);
                startActivity(newActivity);
            }
        });

    }

    private void compareAndShow() {
        FilterClass fc = new FilterClass();
        GestureRecord[] gr = fh.getStoredGestures(GraphActivity.this);

        LinearLayout main = new LinearLayout(GraphActivity.this);
        main.setLayoutParams(new LinearLayout.LayoutParams(LinearLayout.LayoutParams.MATCH_PARENT,
                LinearLayout.LayoutParams.WRAP_CONTENT));
```

```
        main.setOrientation(LinearLayout.VERTICAL);

        int best_index = 0;
        double best_score = Double.POSITIVE_INFINITY;

        for(int i = 0; i < gr.length; i++) {
            float[] new_x = fc.matchArrayLengthsViaDuplication(x_movement, gr[i].getX_accel(), GraphActivity.this);
            float[] new_y = fc.matchArrayLengthsViaDuplication(y_movement, gr[i].getY_accel(), GraphActivity.this);
            float[] new_z = fc.matchArrayLengthsViaDuplication(z_movement, gr[i].getZ_accel(), GraphActivity.this);

            MyDTW xscore = new MyDTW(new_x, x_movement);
            MyDTW yscore = new MyDTW(new_y, y_movement);
            MyDTW zscore = new MyDTW(new_z, z_movement);
            double dtw_score = xscore.getDistance() + yscore.getDistance() + zscore.getDistance();

            MyDTW filtered_xscore = new MyDTW(fc.MovingAverageFilter(new_x, GraphActivity.this), fc.MovingAverageF
            MyDTW filtered_yscore = new MyDTW(fc.MovingAverageFilter(new_y, GraphActivity.this), fc.MovingAverageF
            MyDTW filtered_zscore = new MyDTW(fc.MovingAverageFilter(new_z, GraphActivity.this), fc.MovingAverageF
            double filtered_dtw_score = filtered_xscore.getDistance() + filtered_yscore.getDistance() + filtered_zs

            if(filtered_dtw_score < best_score) {
                best_score = filtered_dtw_score;
                best_index = i;
            }

            TextView title = new TextView(GraphActivity.this);
            title.setText(Html.fromHtml("<b> Current vs " + gr[i].getGestureName() + ": </b> <br>"));
            title.setTextColor(Color.BLACK);
            title.setGravity(Gravity.CENTER);

            TextView scores = new TextView(GraphActivity.this);
            scores.setText(Html.fromHtml("<b> DTW Score: </b>" + String.format("%.2f", filtered_dtw_score)
+ " <br>"));
            scores.setTextColor(Color.BLACK);
            scores.setGravity(Gravity.CENTER);

            // orig data ng current
            LineChart lc1 = new LineChart(GraphActivity.this);
            lc1.setLayoutParams(new LinearLayout.LayoutParams(LinearLayout.LayoutParams.MATCH_PARENT, 250));
            lc1 = createGraph(lc1, x_movement, y_movement, z_movement);

            // filtered orig data
            LineChart lc2 = new LineChart(GraphActivity.this);
            lc2.setLayoutParams(new LinearLayout.LayoutParams(LinearLayout.LayoutParams.MATCH_PARENT, 250));
            lc2 = createGraph(lc2, fc.MovingAverageFilter(x_movement, GraphActivity.this),
                    fc.MovingAverageFilter(y_movement, GraphActivity.this), fc.MovingAverageFilter(z_movement, Grap

            // orig stored data
            LineChart lc3 = new LineChart(GraphActivity.this);
            lc3.setLayoutParams(new LinearLayout.LayoutParams(LinearLayout.LayoutParams.MATCH_PARENT, 250));
            lc3 = createGraph(lc3, gr[i].getX_accel(), gr[i].getY_accel(), gr[i].getZ_accel());

            // relengthed data
            LineChart lc4 = new LineChart(GraphActivity.this);
            lc4.setLayoutParams(new LinearLayout.LayoutParams(LinearLayout.LayoutParams.MATCH_PARENT, 250));
            lc4 = createGraph(lc4, new_x, new_y, new_z);

            // filtered orig data
            LineChart lc5 = new LineChart(GraphActivity.this);
            lc5.setLayoutParams(new LinearLayout.LayoutParams(LinearLayout.LayoutParams.MATCH_PARENT, 250));
            lc5 = createGraph(lc5, fc.MovingAverageFilter(new_x, GraphActivity.this),
                    fc.MovingAverageFilter(new_y, GraphActivity.this), fc.MovingAverageFilter(new_z, GraphActivity.

            main.addView(title);
            main.addView(scores);
            //main.addView(lc1);
            main.addView(lc2);
            //main.addView(lc3);
            //main.addView(lc4);
            main.addView(lc5);
        }

        TextView best_gesture = new TextView(GraphActivity.this);
        best_gesture.setText(Html.fromHtml("The mostly likely gesture is: <b>" + gr[best_index].getGestureName() +
                                           "</b> with a score of <b>" + String.format("%.2f", best_score) + "</b> <
        best_gesture.setTextColor(Color.BLACK);
        best_gesture.setGravity(Gravity.CENTER);
        main.addView(best_gesture);

        ScrollView sv = new ScrollView(GraphActivity.this);
        sv.addView(main);
        AlertDialog.Builder builder = new AlertDialog.Builder(GraphActivity.this);
        builder.setView(sv)
                .setCancelable(true)
                .setNegativeButton("BACK",
                new DialogInterface.OnClickListener() {
                    public void onClick(DialogInterface dialog, int id) {
                    }
                });
        Dialog dialog = builder.create();
        dialog.show();
    }
```

```
    public double getWarpDistance(GestureRecord rec) {

        MyDTW for_x = new MyDTW(rec.getX_accel(), x_movement);
        MyDTW for_y = new MyDTW(rec.getY_accel(), y_movement);
        MyDTW for_z = new MyDTW(rec.getZ_accel(), z_movement);

        return for_x.getDistance() + for_y.getDistance() + for_z.getDistance();
    }

    private LineChart createGraph(LineChart lc, float[] x, float[] y, float[] z) {
        ArrayList<Entry> x_entry = new ArrayList<>();
        ArrayList<Entry> y_entry = new ArrayList<>();
        ArrayList<Entry> z_entry = new ArrayList<>();
        String[] ground = new String[x.length];

        for(int i = 0; i < x.length; i++) {
            x_entry.add(new Entry(x[i], i));
            y_entry.add(new Entry(y[i], i));
            z_entry.add(new Entry(z[i], i));
            ground[i] = i + "";
        }

        LineDataSet x_lds = new LineDataSet(x_entry, "X");
        x_lds.setColor(Color.RED);
        LineDataSet y_lds = new LineDataSet(y_entry, "Y");
        y_lds.setColor(Color.BLUE);
        LineDataSet z_lds = new LineDataSet(z_entry, "Z");
        z_lds.setColor(Color.GREEN);

        ArrayList<ILineDataSet> lds = new ArrayList<>();
        lds.add(x_lds);
        lds.add(y_lds);
        lds.add(z_lds);
        lc.setData(new LineData(ground, lds));

        return lc;
    }


    private void rawGraph() {
        LineChart lc = (LineChart) findViewById(R.id.raw_graph);
        lc.clear();
        createGraph(lc, x_movement, y_movement, z_movement);
    }

    private void LPFGraph() {
        FilterClass fc = new FilterClass();
        LineChart lc = (LineChart) findViewById(R.id.lpf_graph);
        lc.clear();

        try {
            float[] new_x = fc.MovingAverageFilter(x_movement, GraphActivity.this);
            float[] new_y = fc.MovingAverageFilter(y_movement, GraphActivity.this);
            float[] new_z = fc.MovingAverageFilter(z_movement, GraphActivity.this);
            createGraph(lc, new_x, new_y, new_z);
        } catch(Exception e) {
            Toast.makeText(GraphActivity.this, e + "PUTANGINA ASAN YUNG LOOP", Toast.LENGTH_LONG).show();
        }

    }
}
```

```
package com.example.sirwheistein.gesturerecognizer;

import android.app.AlarmManager;
import android.app.Dialog;
import android.app.PendingIntent;
import android.content.DialogInterface;
import android.content.Intent;
import android.provider.MediaStore;
import android.support.v7.app.AlertDialog;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.CompoundButton;
import android.widget.ImageButton;
import android.widget.LinearLayout;
import android.widget.RadioButton;
import android.widget.RadioGroup;
import android.widget.Toast;
import android.widget.ToggleButton;

import java.util.Calendar;

public class MainActivity extends AppCompatActivity {

    private ImageButton record_gesture_button;
    private ImageButton view_stored_gestures_button;
    private ImageButton view_session_button;
```

```
private ImageButton set_session_button;
private ImageButton motiv_select;
private ToggleButton toggle;

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main_menu);

    record_gesture_button = (ImageButton) findViewById(R.id.record_gesture_button);
    view_stored_gestures_button = (ImageButton) findViewById(R.id.view_stored_gestures_button);
    set_session_button = (ImageButton) findViewById(R.id.set_session_button);
    view_session_button = (ImageButton) findViewById(R.id.view_session_button);
    motiv_select = (ImageButton) findViewById(R.id.pick_motivation);

    ToggleButton toggle = (ToggleButton) findViewById(R.id.toggle_it);
    toggle.setOnCheckedChangeListener(new CompoundButton.OnCheckedChangeListener() {
        public void onCheckedChanged(CompoundButton buttonView, boolean isChecked) {
            if (isChecked) {
                record_gesture_button.setVisibility(View.INVISIBLE);
                set_session_button.setVisibility(View.INVISIBLE);

                view_stored_gestures_button.setVisibility(View.VISIBLE);
                view_session_button.setVisibility(View.VISIBLE);
                motiv_select.setVisibility(View.VISIBLE);
            } else {
                record_gesture_button.setVisibility(View.VISIBLE);
                set_session_button.setVisibility(View.VISIBLE);

                view_stored_gestures_button.setVisibility(View.INVISIBLE);
                view_session_button.setVisibility(View.INVISIBLE);
                motiv_select.setVisibility(View.INVISIBLE);
            }
        }
    });


    record_gesture_button.setOnClickListener(new View.OnClickListener() {
        public void onClick(View v) {
            Intent newActivity = new Intent(MainActivity.this, RecordActivity.class);
            newActivity.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP);
            newActivity.putExtra("used_for", "single");
            startActivity(newActivity);
        }
    });

    view_stored_gestures_button.setOnClickListener(new View.OnClickListener() {
        public void onClick(View v) {
            Intent newActivity = new Intent(MainActivity.this, ViewStoredGesturesActivity.class);
            newActivity.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP);
            startActivity(newActivity);
        }
    });

    set_session_button.setOnClickListener(new View.OnClickListener() {
        public void onClick(View v) {
            Intent newActivity = new Intent(MainActivity.this, StoreSessionActivity.class);
            newActivity.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP);
            startActivity(newActivity);
        }
    });

    view_session_button.setOnClickListener(new View.OnClickListener() {
        public void onClick(View v) {
            /*Intent newActivity = new Intent(MainActivity.this, RecordActivity.class);
            int[] session_seq = {1, 1, 1, 2}; // eto important
            double[] scores = new double[session_seq.length];

            newActivity.putExtra("session_seq", session_seq);
            newActivity.putExtra("scores", scores);
            newActivity.putExtra("session_index", 0);
            newActivity.putExtra("used_for", "multiple");
            startActivity(newActivity);*/

            Intent newActivity = new Intent(MainActivity.this, ViewSessions.class);
            newActivity.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP);
            startActivity(newActivity);
        }
    });

    motiv_select.setOnClickListener(new View.OnClickListener() {
        public void onClick(View v) {

            final MotivationFileHandler m = new MotivationFileHandler("motive.ser");

            final RadioGroup rg = new RadioGroup(MainActivity.this);
            MotivationalFeedback mf = m.read(MainActivity.this);
            String[] list = mf.getNames();

            RadioButton[] rb = new RadioButton[list.length];

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

```
                    rb[i] = new RadioButton(MainActivity.this);
                    rb[i].setText(list[i]);
                    //rb[i].setId(i);
                    rg.addView(rb[i]);
                }

                try {
                    rb[mf.getIndex()].setChecked(true);
                } catch(Exception e) {
                    Toast.makeText(MainActivity.this, e +"", Toast.LENGTH_LONG);
                }

                Toast.makeText(MainActivity.this, mf.getIndex() + " INDEX", Toast.LENGTH_LONG);

                AlertDialog.Builder builder = new AlertDialog.Builder(MainActivity.this);
                builder.setView(rg).setCancelable(false)
                .setPositiveButton("SUBMIT",
                        new DialogInterface.OnClickListener() {
                            public void onClick(DialogInterface dialog, int id) {
                                int index = 0;
                                for(int i = 0; i < rg.getChildCount(); i++) {
                                    if(((RadioButton) rg.getChildAt(i)).isChecked()) {
                                        index = i;
                                    }
                                }
                                MotivationalFeedback mf = new MotivationalFeedback(index);
                                m.save(mf, MainActivity.this);
                            }
                        });
                Dialog dialog = builder.create();
                dialog.show();
            }
        });

        soundAlarms();
    }

    private void soundAlarms() {
        SessionFileHandler sh = new SessionFileHandler("ses.ser");
        SessionRecord[] list = sh.getStoredSessions(MainActivity.this);
        for(int i = 0; i < list.length; i++) {
            Calendar[] cal = list[i].getSessionCalendar();
            for(int j = 0; j < cal.length; j++) {
                Intent intent = new Intent(MainActivity.this, MyAlarm.class);
                intent.putExtra("session_name", list[i].getSession_name());
                PendingIntent pi = PendingIntent.getBroadcast(MainActivity.this, i, intent, 0);

                if(System.currentTimeMillis() < cal[j].getTimeInMillis()) {
                    AlarmManager as = (AlarmManager) getSystemService(ALARM_SERVICE);
                    as.set(AlarmManager.RTC_WAKEUP, cal[j].getTimeInMillis(), pi);
                }
            }
        }

    }
}

package com.example.sirwheistein.gesturerecognizer;


import java.io.Serializable;

/**
 * Created by Sir.Wheistein on 4/14/2017.
 */

public class MotivationalFeedback implements Serializable{

    private String[] list = {"Clap", "Tada", "Pat On Back"};
    private int[] ids = {R.raw.clap, R.raw.tada, R.raw.pat};
    private int index;

    MotivationalFeedback(int index) {
        this.index = index;
    }

    public String[] getNames() {
        return list;
    }

    public int[] getIds() {
        return ids;
    }

    public int getIndex() {
        return index;
    }

    public void setIndex(int i) {
        index = i;
    }
```

```
}
```

```
package com.example.sirwheistein.gesturerecognizer;

import android.content.Context;
import android.widget.Toast;

import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;

import static android.content.Context.MODE_PRIVATE;

/**
 * Created by Sir.Wheistein on 4/14/2017.
 */

public class MotivationFileHandler {

    private String fileName;

    public MotivationFileHandler(String filename) {
        this.fileName = filename;
    }

    public void save(MotivationalFeedback m, Context context) {

        try {
            FileOutputStream fs = context.openFileOutput(fileName, MODE_PRIVATE);
            ObjectOutputStream os = new ObjectOutputStream(fs);
            os.writeObject(m);
            os.close();
        }
        catch(Exception e) {
            Toast.makeText(context, "HINDI NASTORE" + e.toString(), Toast.LENGTH_SHORT).show();
        }
    }

    public MotivationalFeedback read(Context context) {
        MotivationalFeedback m = new MotivationalFeedback(0);
        try {
            FileInputStream fi = context.openFileInput(fileName);
            ObjectInputStream is = new ObjectInputStream(fi);
            m = (MotivationalFeedback) is.readObject();
            is.close();
        }
        catch(Exception e) {
            Toast.makeText(context, "HINDI NAKAPAGBASA " + e.toString(), Toast.LENGTH_SHORT).show();
        }
        save(m, context);
        return m;
    }

    public void DeleteFile(Context context) {
        context.deleteFile(fileName);
    }
}
```

```
package com.example.sirwheistein.gesturerecognizer;

import android.app.Dialog;
import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.DialogInterface;
import android.content.Intent;
import android.os.Bundle;
import android.os.Vibrator;
import android.support.v7.app.AlertDialog;
import android.widget.LinearLayout;
import android.widget.RadioButton;
import android.widget.TextView;
import android.widget.Toast;

/**
 * Created by Sir.Wheistein on 4/13/2017.
 */

public class MyAlarm extends BroadcastReceiver{

    @Override
    public void onReceive(Context context, Intent intent) {
        Bundle bundle = intent.getExtras();
        String session_name = bundle.getString("session_name");
        Toast.makeText(context, "It's time to perform: " + session_name, Toast.LENGTH_LONG).show();
        Vibrator vibrator = (Vibrator) context.getSystemService(Context.VIBRATOR_SERVICE);
        vibrator.vibrate(10000);
    }
}
```

```
package com.example.sirwheistein.gesturerecognizer;
/**
 * This class implements the Dynamic Time Warping algorithm
 * given two sequences
 * <pre>
 *    X = x1, x2,..., xi,..., xn
 *    Y = y1, y2,..., yj,..., ym
 *  </pre>
 *
 * @author              Cheol-Woo Jung (cjung@gatech.edu)
 * @version     1.0
 */
public class MyDTW {

    protected float[] seq1;
    protected float[] seq2;
    protected int[][] warpingPath;

    protected int n;
    protected int m;
    protected int K;

    protected double warpingDistance;


    public MyDTW(float[] sample, float[] templete) {
        seq1 = sample;
        seq2 = templete;

        n = seq1.length;
        m = seq2.length;
        K = 1;

        warpingPath = new int[n + m][2];         // max(n, m) <= K < n + m
        warpingDistance = 0.0;

        this.compute();
    }

    public void compute() {
        double accumulatedDistance = 0.0;

        double[][] d = new double[n][m];         // local distances
        double[][] D = new double[n][m];         // global distances

        for (int i = 0; i < n; i++) {
            for (int j = 0; j < m; j++) {
                d[i][j] = distanceBetween(seq1[i], seq2[j]);
            }
        }

        D[0][0] = d[0][0];

        for (int i = 1; i < n; i++) {
            D[i][0] = d[i][0] + D[i - 1][0];
        }

        for (int j = 1; j < m; j++) {
            D[0][j] = d[0][j] + D[0][j - 1];
        }

        for (int i = 1; i < n; i++) {
            for (int j = 1; j < m; j++) {
                accumulatedDistance = Math.min(Math.min(D[i-1][j], D[i-1][j-1]), D[i][j-1]);
                accumulatedDistance += d[i][j];
                D[i][j] = accumulatedDistance;
            }
        }
        accumulatedDistance = D[n - 1][m - 1];

        int i = n - 1;
        int j = m - 1;
        int minIndex = 1;

        warpingPath[K - 1][0] = i;
        warpingPath[K - 1][1] = j;

        while ((i + j) != 0) {
            if (i == 0) {
                j -= 1;
            } else if (j == 0) {
                i -= 1;
            } else {    // i != 0 && j != 0
                double[] array = { D[i - 1][j], D[i][j - 1], D[i - 1][j - 1] };
                minIndex = this.getIndexOfMinimum(array);

                if (minIndex == 0) {
                    i -= 1;
                } else if (minIndex == 1) {
                    j -= 1;
                } else if (minIndex == 2) {
                    i -= 1;
```

```
                    j -= 1;
                }
            } // end else
            K++;
            warpingPath[K - 1][0] = i;
            warpingPath[K - 1][1] = j;
        } // end while
        warpingDistance = accumulatedDistance / K;

        this.reversePath(warpingPath);
    }

    /**
     * Changes the order of the warping path (increasing order)
     *
     * @param path      the warping path in reverse order
     */
    protected void reversePath(int[][] path) {
        int[][] newPath = new int[K][2];
        for (int i = 0; i < K; i++) {
            for (int j = 0; j < 2; j++) {
                newPath[i][j] = path[K - i - 1][j];
            }
        }
        warpingPath = newPath;
    }
    /**
     * Returns the warping distance
     *
     * @return
     */
    public double getDistance() {
        return warpingDistance;
    }

    /**
     * Computes a distance between two points
     *
     * @param p1        the point 1
     * @param p2        the point 2
     * @return          the distance between two points
     */
    protected double distanceBetween(double p1, double p2) {
        return (p1 - p2) * (p1 - p2);
    }

    /**
     * Finds the index of the minimum element from the given array
     *
     * @param array             the array containing numeric values
     * @return                          the min value among elements
     */
    protected int getIndexOfMinimum(double[] array) {
        int index = 0;
        double val = array[0];

        for (int i = 1; i < array.length; i++) {
            if (array[i] < val) {
                val = array[i];
                index = i;
            }
        }
        return index;
    }

    /**
     *  Returns a string that displays the warping distance and path
     */
    public String toString() {
        String retVal = "Warping Distance: " + warpingDistance + "\n";
        retVal += "Warping Path: {";
        for (int i = 0; i < K; i++) {
            retVal += "(" + warpingPath[i][0] + ", " +warpingPath[i][1] + ")";
            retVal += (i == K - 1) ? "}" : ", ";

        }
        return retVal;
    }

    /**
     * Tests this class
     *
     * @param args      ignored
     */
    public static void main(String[] args) {
        float[] n2 = {1.5f, 3.9f, 4.1f, 3.3f};
        float[] n1 = {2.1f, 2.45f, 3.673f, 4.32f, 2.05f, 1.93f, 5.67f, 6.01f};
        MyDTW dtw = new MyDTW(n1, n2);
        System.out.println(dtw);
    }
}
```

```
package com.example.sirwheistein.gesturerecognizer;

import android.content.Context;
import android.content.Intent;
import android.hardware.Sensor;
import android.hardware.SensorEvent;
import android.hardware.SensorEventListener;
import android.hardware.SensorManager;
import android.media.MediaPlayer;
import android.os.Bundle;
import android.support.v7.app.AppCompatActivity;
import android.view.MotionEvent;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;
import android.widget.Toast;

import java.util.logging.FileHandler;
import java.util.logging.Filter;

import at.markushi.ui.CircleButton;

public class RecordActivity extends AppCompatActivity
                implements SensorEventListener{

    private float[] x_movement;
    private float[] y_movement;
    private float[] z_movement;

    boolean isDown = false;

    private CircleButton recordBttn;

    private GestureFileHandler fh;
    private Context myContext;
    private String used_for;

    private String[] session_seq;
    private double[] scores;
    private int session_index;

    private int session_name;
    private long start_time;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_record);
        try {

            SensorManager sm = (SensorManager) getSystemService(SENSOR_SERVICE);
            Sensor mySensor = sm.getDefaultSensor(Sensor.TYPE_ACCELEROMETER);
            sm.registerListener(this, mySensor, SensorManager.SENSOR_STATUS_ACCURACY_LOW);
            myContext = this.getApplicationContext();

            receiveMultiple();

            x_movement = new float[0];
            y_movement = new float[0];
            z_movement = new float[0];

            recordBttn = (CircleButton) findViewById(R.id.Record);
            recordBttn.setOnTouchListener(new View.OnTouchListener() {
                @Override
                public boolean onTouch(View v, MotionEvent event) {
                    if (event.getAction() == MotionEvent.ACTION_DOWN) {
                        isDown = true;
                    } else if (event.getAction() == MotionEvent.ACTION_UP) {
                        isDown = false;
                        if (used_for.equals("multiple")) {

                            MotivationFileHandler m = new MotivationFileHandler("motive.ser");
                            MotivationalFeedback mf = m.read(RecordActivity.this);

                            MediaPlayer motiv_sound = MediaPlayer.create(myContext, mf.getIds()[mf.getIndex()]);
                            motiv_sound.start();
                            updateScore();
                            sendMultiple();
                        } else {
                            showGraph();
                        }
                    }
                    return false;
                }
            });
        } catch(Exception e) {
            Toast.makeText(myContext, "" + e, Toast.LENGTH_LONG);
        }
    }

    private void receiveMultiple() {
```

```
        Bundle bundle = getIntent().getExtras();
        used_for = bundle.getString("used_for");
        if(used_for.equals("multiple")) {
            session_seq = bundle.getStringArray("session_seq");
            scores = bundle.getDoubleArray("scores");
            session_index = bundle.getInt("session_index");
            session_name = bundle.getInt("session_name");
            fh = new GestureFileHandler("ges.ser");
            TextView gesture_needed = (TextView) findViewById(R.id.gesture_instruction);
            gesture_needed.setText("Perform " + fh.getRecordByID(session_seq[session_index], RecordActivity.this).g
                    " (" + (session_index + 1) + "/" + scores.length + ")");
            start_time = bundle.getLong("start_time");
        }
    }

    private void sendMultiple() {
        Intent newActivity = null;

        if(session_index == session_seq.length -1) {
            newActivity = new Intent(this, Session_Results.class);
        } else {
            newActivity = new Intent(this, RecordActivity.class);
            newActivity.putExtra("used_for", "multiple");
            newActivity.putExtra("session_index", session_index +1);
        }

        newActivity.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP);
        newActivity.putExtra("session_seq", session_seq);
        newActivity.putExtra("scores", scores);
        newActivity.putExtra("session_name", session_name);
        newActivity.putExtra("start_time", start_time);
        startActivity(newActivity);
    }

    private void updateScore() {
        scores[session_index] = getWarpDistance(fh.getRecordByID(session_seq[session_index], RecordActivity.this));
    }

    private void showGraph() {
        FilterClass fc = new FilterClass();
        Intent newActivity = new Intent(this, GraphActivity.class);
        newActivity.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP);
        newActivity.putExtra("x_values", x_movement);
        newActivity.putExtra("y_values", y_movement);
        newActivity.putExtra("z_values", z_movement);
        startActivity(newActivity);
    }

    private float[] extendArray(float[] arr, float val) {
        float[] returnArray = new float[arr.length +1];
        for(int i = 0; i < arr.length; i++) {
            returnArray[i] = arr[i];
        }
        returnArray[arr.length ] = val;
        return returnArray;
    }

    @Override
    public void onSensorChanged(SensorEvent event) {
        if(isDown == true) {
            x_movement = extendArray(x_movement, event.values[0]);
            y_movement = extendArray(y_movement, event.values[1]);
            z_movement = extendArray(z_movement, event.values[2]);
        }
    }

    public double getWarpDistance(GestureRecord rec) {

        FilterClass fc = new FilterClass();

        float[] scaled_x = fc.matchArrayLengthsViaDuplication(rec.getX_accel(), x_movement, RecordActivity.this);
        float[] scaled_y = fc.matchArrayLengthsViaDuplication(rec.getY_accel(), y_movement, RecordActivity.this);
        float[] scaled_z = fc.matchArrayLengthsViaDuplication(rec.getZ_accel(), z_movement, RecordActivity.this);


       MyDTW for_x = new MyDTW(fc.MovingAverageFilter(rec.getX_accel(), RecordActivity.this), fc.MovingAverageFilt
       MyDTW for_y = new MyDTW(fc.MovingAverageFilter(rec.getY_accel(), RecordActivity.this), fc.MovingAverageFilt
       MyDTW for_z = new MyDTW(fc.MovingAverageFilter(rec.getZ_accel(), RecordActivity.this), fc.MovingAverageFilt

        return for_x.getDistance() + for_y.getDistance() + for_z.getDistance();
    }

    @Override
    public void onAccuracyChanged(Sensor sensor, int accuracy) {

    }

}


package com.example.sirwheistein.gesturerecognizer;
```

```
import android.content.Context;
import android.graphics.Color;
import android.os.Bundle;
import android.os.Environment;
import android.support.v7.app.AppCompatActivity;
import android.text.Html;
import android.view.Gravity;
import android.widget.LinearLayout;
import android.widget.ScrollView;
import android.widget.TextView;
import android.widget.Toast;

import com.itextpdf.text.BaseColor;
import com.itextpdf.text.Chunk;
import com.itextpdf.text.Document;
import com.itextpdf.text.Element;
import com.itextpdf.text.Font;
import com.itextpdf.text.FontFactory;
import com.itextpdf.text.Paragraph;
import com.itextpdf.text.Phrase;
import com.itextpdf.text.Rectangle;
import com.itextpdf.text.pdf.PdfPCell;
import com.itextpdf.text.pdf.PdfPTable;
import com.itextpdf.text.pdf.PdfWriter;

import java.io.FileOutputStream;
import java.text.SimpleDateFormat;
import java.util.Date;

public class Session_Results extends AppCompatActivity {

    private String[] session_seq;
    private double[] scores;
    private int session_name;
    private long start;

    GestureFileHandler fh;
    SessionFileHandler sh;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        //setContentView(R.layout.activity_session__results);

        Bundle bundle = getIntent().getExtras();

        session_seq = bundle.getStringArray("session_seq");
        scores = bundle.getDoubleArray("scores");
        session_name = bundle.getInt("session_name");
        fh = new GestureFileHandler("ges.ser");
        sh = new SessionFileHandler("ses.ser");
        start = bundle.getLong("start_time");

        showResults();
        createDocument();
    }

    private void showResults() {
        try {
            LinearLayout main = new LinearLayout(Session_Results.this);
            main.setLayoutParams(new LinearLayout.LayoutParams(LinearLayout.LayoutParams.MATCH_PARENT,
                    LinearLayout.LayoutParams.WRAP_CONTENT));
            main.setOrientation(LinearLayout.VERTICAL);

            SessionRecord gr = sh.getStoredSessions(Session_Results.this)[session_name];

            TextView session_name = new TextView(Session_Results.this);
            session_name.setText(Html.fromHtml("<b> Session Name: </b>" + gr.getSession_name() + "<br>"));
            session_name.setTextColor(Color.BLACK);
            session_name.setGravity(Gravity.CENTER);

            TextView start_time = new TextView(Session_Results.this);
            Date date = new Date(start);
            SimpleDateFormat format1 = new SimpleDateFormat("MMM dd, yyyy hh:mm aaa");
            start_time.setText(Html.fromHtml("<b> Start Date: </b>" + format1.format(date) + "<br>"));
            start_time.setTextColor(Color.BLACK);
            start_time.setGravity(Gravity.CENTER);

            TextView end_time = new TextView(Session_Results.this);
            date = new Date(System.currentTimeMillis());
            format1 = new SimpleDateFormat("MMM dd, yyyy hh:mm aaa");
            end_time.setText(Html.fromHtml("<b> End Date: </b>" + format1.format(date) + "<br>"));
            end_time.setTextColor(Color.BLACK);
            end_time.setGravity(Gravity.CENTER);

            TextView session_results = new TextView(Session_Results.this);
            String breakdown = "Results: <br> <b>" + fh.getRecordByID(session_seq[0], Session_Results.this).getGestu
            int counter = 0;
            double sumperline = 0;
            double sum = 0;
            for(int j = 0 ; j < scores.length -1; j++) {
                breakdown = breakdown + String.format("%.2f", scores[j]) + "<br>";
```

```
                counter = counter +1;
                sumperline = sumperline + scores[j];
                sum = sum + scores[j];
                if(!session_seq[j].equals(session_seq[j +1])) {
                    breakdown = breakdown + "<b> Average: </b>" + String.format("%.2f", (sumperline * 1.0 / counter
                    breakdown = breakdown + "<b>" + fh.getRecordByID(session_seq[j+1], Session_Results.this).getGes
                    counter = 0;
                    sumperline = 0;
                }
            }
            breakdown = breakdown + String.format("%.2f", scores[scores.length-1]) + "<br>";
            sumperline = sumperline + scores[scores.length -1];
            counter = counter +1;
            breakdown = breakdown + "<b> Average: </b>" + (sumperline * 1.0 / counter) + "<br>";
            sum = sum + scores[scores.length -1];
            breakdown = breakdown + "<b> Total Average: </b>" + String.format("%.2f", (sum * 1.0 / scores.length))
            session_results.setText(Html.fromHtml(breakdown));
            session_results.setTextColor(Color.BLACK);
            //session_results.setGravity(Gravity.CENTER);

            main.addView(session_name);
            main.addView(start_time);
            main.addView(end_time);
            main.addView(session_results);

            ScrollView sv = new ScrollView(Session_Results.this);
            sv.addView(main);

            setContentView(sv);
        } catch (Exception e) {
            Toast.makeText(Session_Results.this, e +"", Toast.LENGTH_SHORT).show();
        }
    }

    /* String place = "";
        for(int i = 0; i < session_seq.length; i++) {
            place = place + "" + fh.getRecordByID(session_seq[i], Session_Results.this).getGestureName() +
                    " has score of " + scores[i] + "\n";
        }
        EditText session_results = (EditText) findViewById(R.id.session_results);
        session_results.setText(place); */

    public void createDocument() {

        SessionFileHandler sh = new SessionFileHandler("ses.ser");
        SessionRecord sr = sh.getStoredSessions(Session_Results.this)[session_name];

        Document document = new Document();

        Date end_date = new Date(System.currentTimeMillis());
        SimpleDateFormat format2 = new SimpleDateFormat("MMddyyyyhhmmssaaa");
        String document_name = sr.getSession_name() + "_" + format2.format(end_date);

        String outpath = Environment.getExternalStoragePublicDirectory(Environment.DIRECTORY_DOCUMENTS) + "/" + doc
        try {
            PdfWriter writer = PdfWriter.getInstance(document, new FileOutputStream(outpath));
            document.open();
            Paragraph title = new Paragraph(sr.getSession_name(), FontFactory.getFont(FontFactory.HELVETICA, 18, Fo
            title.setAlignment(Element.ALIGN_CENTER);
            document.add(title);

            document.add(Chunk.NEWLINE);

            Date date = new Date(start);
            SimpleDateFormat format1 = new SimpleDateFormat("MMM dd, yyyy hh:mm aaa");
            Paragraph start_date = new Paragraph();
            start_date.add(new Phrase("START DATE: ", FontFactory.getFont(FontFactory.HELVETICA, 14, Font.BOLD)));
            start_date.add(new Phrase(format1.format(date)));
            start_date.setAlignment(Element.ALIGN_CENTER);
            document.add(start_date);

            date = new Date(System.currentTimeMillis());
            format1 = new SimpleDateFormat("MMM dd, yyyy hh:mm aaa");
            Paragraph end_date2 = new Paragraph();
            end_date2.add(new Phrase("END DATE: ", FontFactory.getFont(FontFactory.HELVETICA, 14, Font.BOLD)));
            end_date2.add(new Phrase(format1.format(date)));
            end_date2.setAlignment(Element.ALIGN_CENTER);
            document.add(end_date2);

            document.add(Chunk.NEWLINE);

            PdfPTable table = new PdfPTable(2);

            //String breakdown = "Results: <br> <b>" + fh.getRecordByID(session_seq[0], Session_Results.this).getGe
            PdfPCell editable = new PdfPCell(new Paragraph(fh.getRecordByID(session_seq[0], Session_Results.this).g
            editable.setHorizontalAlignment(Element.ALIGN_CENTER);
            editable.setBorder(Rectangle.NO_BORDER);
            table.addCell(editable);

            int counter = 0;
            double sumperline = 0;
            double sum = 0;
```

```
            for(int j = 0 ; j < scores.length -1; j++) {
                PdfPCell editable2 = new PdfPCell(new Paragraph(String.format("%.2f", scores[j])));
                editable2.setBorder(Rectangle.NO_BORDER);
                table.addCell(editable2);

                counter = counter +1;
                sumperline = sumperline + scores[j];
                sum = sum + scores[j];
                if(!session_seq[j].equals(session_seq[j +1])) {
                    //breakdown = breakdown + "<b> Average: </b>" + String.format("%.2f", (sumperline * 1.0 / count
                    //breakdown = breakdown + "<b>" + fh.getRecordByID(session_seq[j+1], Session_Results.this).getG

                    PdfPCell editable3 = new PdfPCell(new Paragraph("Average: ", FontFactory.getFont(FontFactory.HE
                    editable3.setBorder(Rectangle.NO_BORDER);
                    editable3.setHorizontalAlignment(Element.ALIGN_CENTER);
                    table.addCell(editable3);

                    PdfPCell editable4 = new PdfPCell(new Paragraph(String.format("%.2f", (sumperline * 1.0 / count
                    editable4.setBorder(Rectangle.NO_BORDER);
                    table.addCell(editable4);

                    PdfPCell editable5 = new PdfPCell(new Paragraph(fh.getRecordByID(session_seq[j+1], Session_Resu
                    editable5.setHorizontalAlignment(Element.ALIGN_CENTER);
                    editable5.setBorder(Rectangle.NO_BORDER);
                    table.addCell(editable5);

                    counter = 0;
                    sumperline = 0;
                } else {
                    PdfPCell editableoptional = new PdfPCell(new Paragraph(""));
                    editableoptional.setHorizontalAlignment(Element.ALIGN_CENTER);
                    editableoptional.setBorder(Rectangle.NO_BORDER);
                    table.addCell(editableoptional);
                }
            }

            PdfPCell editable6 = new PdfPCell(new Paragraph(String.format("%.2f", scores[scores.length -1])));
            editable6.setBorder(Rectangle.NO_BORDER);
            table.addCell(editable6);

            sumperline = sumperline + scores[scores.length -1];
            counter = counter +1;
            //breakdown = breakdown + "<b> Average: </b>" + (sumperline * 1.0 / counter) + "<br>";

            PdfPCell editable7 = new PdfPCell(new Paragraph("Average: ", FontFactory.getFont(FontFactory.HELVETICA,
            editable7.setHorizontalAlignment(Element.ALIGN_CENTER);
            editable7.setBorder(Rectangle.NO_BORDER);
            table.addCell(editable7);

            PdfPCell editable8 = new PdfPCell(new Paragraph(String.format("%.2f", (sumperline * 1.0 / counter))));
            editable8.setBorder(Rectangle.NO_BORDER);
            table.addCell(editable8);

            sum = sum + scores[scores.length -1];
           // breakdown = breakdown + "<b> Total Average: </b>" + String.format("%.2f", (sum * 1.0 / scores.length)

            PdfPCell editable9 = new PdfPCell(new Paragraph("Total Average: ", FontFactory.getFont(FontFactory.HELV
            editable9.setHorizontalAlignment(Element.ALIGN_CENTER);
            editable9.setBorder(Rectangle.NO_BORDER);
            table.addCell(editable9);

            PdfPCell editable10 = new PdfPCell(new Paragraph(String.format("%.2f", (sum * 1.0 / scores.length))));
            editable10.setBorder(Rectangle.NO_BORDER);
            table.addCell(editable10);

            document.add(table);
            document.close();
        } catch (Exception e) {

        }
    }

}


package com.example.sirwheistein.gesturerecognizer;

import android.content.Context;
import android.widget.Toast;

import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;

import static android.content.Context.MODE_PRIVATE;

/**
 * Created by Sir.Wheistein on 4/4/2017.
 */

public class SessionFileHandler {
```

```
    private String fileName;

    public SessionFileHandler(String filename) {
        this.fileName = filename;
    }

    public void storeNewSession(SessionRecord record, Context context) {
        if(record != null) {
            SessionRecord[] prevRecords = getStoredSessions(context);
            SessionRecord[] new_rec = new SessionRecord[prevRecords.length + 1];
            for (int i = 0; i < prevRecords.length; i++) {
                new_rec[i] = prevRecords[i];
            }
            new_rec[new_rec.length - 1] = record;
            save(new_rec, context);
        }
    }

    public void save(SessionRecord[] rec, Context context) {
        try {
            FileOutputStream fs = context.openFileOutput(fileName, MODE_PRIVATE);
            ObjectOutputStream os = new ObjectOutputStream(fs);
            os.writeObject(rec);
            os.close();
        } catch(Exception e) {
            Toast.makeText(context, e+"", Toast.LENGTH_LONG);
        }
    }

    public SessionRecord[] getStoredSessions(Context context) {
        SessionRecord[] records = new SessionRecord[0];
        try {
            FileInputStream fi = context.openFileInput(fileName);
            ObjectInputStream is = new ObjectInputStream(fi);
            records = (SessionRecord[]) is.readObject();
            is.close();
        }
        catch(Exception e) {
            Toast.makeText(context, "HINDI NAKAPAGBASA " + e.toString(), Toast.LENGTH_SHORT).show();
        }
        return records;
    }

    public void DeleteAtIndex(int index, Context context) {
        SessionRecord[] sr = getStoredSessions(context);
        SessionRecord[] newarr = new SessionRecord[sr.length -1];
        for(int i = 0; i < index; i++) {
            newarr[i] = sr[i];
        }
        for(int i = index +1; i < sr.length; i++) {
            newarr[i-1] = sr[i];
        }
        save(newarr, context);
    }

    public void DeleteFile(Context context) {
        context.deleteFile(fileName);
    }
}


package com.example.sirwheistein.gesturerecognizer;

import java.io.Serializable;
import java.util.Calendar;

/**
 * Created by Sir.Wheistein on 4/4/2017.
 */

public class SessionRecord implements Serializable{

    private String[] session_sequence;
    private String session_name;
    private String session_description;
    private Calendar cal[];

    public SessionRecord(String[] session_sequence, String session_name) {
        this.session_sequence = session_sequence;
        this.session_name = session_name;
    }

    public String[] getSession_sequence() {
        return session_sequence;
    }

    public void setSession_sequence(String[] session_sequence) {
        this.session_sequence = session_sequence;
    }

    public String getSession_description() {
```

```
        return session_description;
    }

    public void setSession_description(String session_description) {
        this.session_description = session_description;
    }

    public void setSessionCalendar(Calendar cal[]) {
        this.cal = cal;
    }

    public Calendar[] getSessionCalendar() {
        return cal;
    }

    public String getSession_name() {
        return session_name;
    }

    public void setSession_name(String session_name) {
        this.session_name = session_name;
    }
}
```

```
package com.example.sirwheistein.gesturerecognizer;

import android.app.AlarmManager;
import android.app.Dialog;
import android.app.PendingIntent;
import android.content.Context;
import android.content.DialogInterface;
import android.content.Intent;
import android.graphics.Color;
import android.os.Bundle;
import android.support.v7.app.AlertDialog;
import android.support.v7.app.AppCompatActivity;
import android.view.View;
import android.widget.ArrayAdapter;
import android.widget.Button;
import android.widget.DatePicker;
import android.widget.EditText;
import android.widget.ImageButton;
import android.widget.Spinner;
import android.widget.TextView;
import android.widget.TimePicker;
import android.widget.Toast;

import java.text.SimpleDateFormat;
import java.util.Calendar;
import java.util.Date;
import java.util.GregorianCalendar;

public class StoreSessionActivity extends AppCompatActivity {

    private Context myContext;
    private GestureFileHandler fh;
    private SessionFileHandler sh;

    private int[] performances; // ilan beses gagawin
    private int[] gesture_ids; // ano yung gagawin

    private Spinner gestureChoices;
    private Button add_to_list;
    private Button store_session;
    private EditText number_of_performances;
    private TextView list_elements;
    private EditText session_name;
    private EditText session_description;

    private ImageButton set_date;

    private DatePicker dp;
    private TimePicker tp;
    private Calendar[] callist;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_store_session);

        myContext = this.getApplicationContext();
        fh = new GestureFileHandler("ges.ser");
        sh = new SessionFileHandler("ses.ser");

        callist = new Calendar[0];

        performances = new int[0];
        gesture_ids = new int[0];

        gestureChoices = (Spinner) findViewById(R.id.gesture_choices);
        ArrayAdapter<String> adapter = new ArrayAdapter<>(myContext, R.layout.spinner_item, getGestureNames());
```

```
gestureChoices.setAdapter(adapter);

for( int i = 0; i < gestureChoices.getChildCount(); i++) {
    TextView spinnerText = (TextView) gestureChoices.getChildAt(i);
    spinnerText.setTextColor(Color.BLACK);
}

number_of_performances = (EditText) findViewById(R.id.number_of_performances);
list_elements = (TextView) findViewById(R.id.list_elements);
//list_elements.setText(sh.getStoredSessions(myContext).length);
session_name = (EditText) findViewById(R.id.session_name);
add_to_list = (Button) findViewById(R.id.add_to_list);
add_to_list.setOnClickListener(new View.OnClickListener() {
    public void onClick(View v) {
        add_to_list();
        updateList();
    }
});

store_session = (Button) findViewById(R.id.store_session);
store_session.setOnClickListener(new View.OnClickListener() {
    public void onClick(View v) {
        storeToFile();

    }
});

set_date = (ImageButton) findViewById(R.id.set_date_button);
set_date.setOnClickListener(new View.OnClickListener() {
    public void onClick(View v) {
try {
    AlertDialog.Builder builder = new AlertDialog.Builder(StoreSessionActivity.this);
    dp = new DatePicker(StoreSessionActivity.this);
    builder.setView(dp);

    builder.setCancelable(false)
    .setPositiveButton("SELECT TIME",
    new DialogInterface.OnClickListener() {
        public void onClick(DialogInterface dialog, int id) {
            AlertDialog.Builder builder2 = new AlertDialog.Builder(StoreSessionActivity.this);
            tp = new TimePicker(StoreSessionActivity.this);
            builder2.setView(tp);

            builder2.setCancelable(false)
            .setPositiveButton("ADD SCHEDULE",
                    new DialogInterface.OnClickListener() {
                        public void onClick(DialogInterface dialog, int id) {
                            addToCalList();
                            updateScheduleView();
                        }
                    })
            .setNegativeButton("CANCEL",
                    new DialogInterface.OnClickListener() {
                        public void onClick(DialogInterface dialog, int id) {

                        }
                    });
            Dialog dialog2 = builder2.create();
            dialog2.show();
        }
    })
    .setNegativeButton("CANCEL",
    new DialogInterface.OnClickListener() {
        public void onClick(DialogInterface dialog, int id) {

        }
    });

    Dialog dialog = builder.create();
    dialog.show();
    //Toast.makeText(myContext, "BYE", Toast.LENGTH_SHORT).show();
    } catch(Exception e) {
        Toast.makeText(myContext, e + "", Toast.LENGTH_SHORT).show();
    }
    }
});

ImageButton add = (ImageButton) findViewById(R.id.add_button);
add.setOnClickListener(new View.OnClickListener() {
    public void onClick(View v) {
       EditText amount = (EditText) findViewById(R.id.number_of_performances);
        int amt = 0;
        if(amount.getText() != null && !amount.getText().equals("")) {
            amt = Integer.parseInt(amount.getText()+"");
        }
        amount.setText((amt+1) + "");
    }
});

ImageButton subt = (ImageButton) findViewById(R.id.subt_button);
subt.setOnClickListener(new View.OnClickListener() {
```

```
        public void onClick(View v) {
            EditText amount = (EditText) findViewById(R.id.number_of_performances);
            int amt = 0;
            if(amount.getText() != null && !amount.getText().equals("")) {
                amt = Integer.parseInt(amount.getText()+"");
                if(amt <= 0) {
                    amt = 1;
                }
            }
            amount.setText((amt-1) + "");
        }
    });
}

private void addToCalList() {
    Calendar cal = new GregorianCalendar();
    cal.set(Calendar.MONTH, dp.getMonth());
    cal.set(Calendar.DAY_OF_MONTH, dp.getDayOfMonth());
    cal.set(Calendar.YEAR, dp.getYear());
    cal.set(Calendar.HOUR_OF_DAY, tp.getCurrentHour());
    cal.set(Calendar.MINUTE, tp.getCurrentMinute());
    cal.set(Calendar.SECOND, 0);

    Calendar[] newcallist = new Calendar[callist.length +1];
    for(int i = 0; i < callist.length; i++) {
        newcallist[i] = callist[i];
    }
    newcallist[newcallist.length -1] = cal;
    callist = newcallist;
}

private void updateScheduleView() {
    TextView schedulelist = (TextView) findViewById(R.id.list_schedules);
    String scheds = "";
    for(int i = 0; i < callist.length; i++) {
        try {
            Date date = callist[i].getTime();
            SimpleDateFormat format1 = new SimpleDateFormat("MMM dd, yyyy hh:mm aaa");
            scheds = scheds + format1.format(date) + "\n";
        } catch (Exception e) {
            Toast.makeText(StoreSessionActivity.this, e +"", Toast.LENGTH_SHORT).show();
        }
    }
    schedulelist.setText(scheds);
}

private String[] getGestureNames() {
    String[] names =  new String[fh.getStoredGestures(myContext).length];
    for(int i = 0; i < names.length; i++) {
        names[i] = fh.getStoredGestures(myContext)[i].getGestureName();
    }
    return names;
}

private void add_to_list() {
    performances = extendArray(performances, Integer.parseInt(number_of_performances.getText() + ""));
    gesture_ids = extendArray(gesture_ids, gestureChoices.getSelectedItemPosition());
    //list_elements.setText(performances.length);
}

private void updateList() {
    String temp = "";
    for(int i = 0; i < performances.length; i++) {
        temp = temp + "Perform " + fh.getStoredGestures(myContext)[gesture_ids[i]].getGestureName() + " " +
                performances[i] + " times\n";
    }
    list_elements.setText(temp);
}

private void storeToFile() {
    try {
        session_description = (EditText) findViewById(R.id.session_description);
        if(getSessionSequence().length != 0 &&
                !session_name.getText().equals("") &&
                callist.length != 0 &&
                !session_description.getText().equals("")) {
            SessionRecord rec = new SessionRecord(getSessionSequence(), session_name.getText() + "");
            Intent intent = new Intent(myContext, MyAlarm.class);
            PendingIntent pi = PendingIntent.getBroadcast(myContext, 1, intent, 0);
            rec.setSessionCalendar(callist);
            rec.setSession_description(session_description.getText() + "");
            sh.storeNewSession(rec, myContext);
            soundCurrentAlarms();
            toMain();
        } else {
            Toast.makeText(myContext, "THERE ARE MISSING FIELDS!", Toast.LENGTH_LONG).show();
        }

    } catch(Exception e) {
        Toast.makeText(myContext, e + "", Toast.LENGTH_LONG).show();
    }
}
```

```
    private void soundCurrentAlarms() {
        for(int j = 0; j < callist.length; j++) {
            Intent intent = new Intent(StoreSessionActivity.this, MyAlarm.class);
            intent.putExtra("session_name", session_name.getText()+"");
            PendingIntent pi = PendingIntent.getBroadcast(StoreSessionActivity.this, j, intent, 0);

            if(System.currentTimeMillis() < callist[j].getTimeInMillis()) {
                AlarmManager as = (AlarmManager) getSystemService(ALARM_SERVICE);
                as.set(AlarmManager.RTC_WAKEUP, callist[j].getTimeInMillis(), pi);
            }
        }
    }

    private void toMain() {
        Intent newActivity = new Intent(myContext, MainActivity.class);
        newActivity.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP);
        startActivity(newActivity);
    }

    private String[] getSessionSequence() {
        String[] session_sequence = new String[0];
        GestureRecord[] list = fh.getStoredGestures(myContext);
        for(int i = 0 ; i < performances.length; i++) {
            for(int j = 0; j < performances[i]; j++) {
                session_sequence = extendArray(session_sequence, list[gesture_ids[i]].getGestureID());
            }
        }
        return session_sequence;
    }

    private int[] extendArray(int[] arr, int val) {
        int[] returnArray = new int[arr.length +1];
        for(int i = 0; i < arr.length; i++) {
            returnArray[i] = arr[i];
        }
        returnArray[arr.length ] = val;
        return returnArray;
    }

    private String[] extendArray(String[] arr, String val) {
        String[] returnArray = new String[arr.length +1];
        for(int i = 0; i < arr.length; i++) {
            returnArray[i] = arr[i];
        }
        returnArray[arr.length ] = val;
        return returnArray;
    }

}


package com.example.sirwheistein.gesturerecognizer;

import android.content.Context;
import android.content.Intent;
import android.graphics.Color;
import android.os.Bundle;
import android.support.v7.app.AppCompatActivity;
import android.text.Html;
import android.view.Gravity;
import android.view.View;
import android.widget.Button;
import android.widget.ImageButton;
import android.widget.LinearLayout;
import android.widget.ScrollView;
import android.widget.TextView;
import android.widget.Toast;

import com.itextpdf.text.Image;

import org.w3c.dom.Text;

import java.text.SimpleDateFormat;
import java.util.Calendar;
import java.util.Date;

public class ViewSessions extends AppCompatActivity {

    private Context myContext;
    private SessionFileHandler sh;
    private GestureFileHandler fh;
    private LinearLayout ll;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_view_sessions);

        myContext = this.getApplicationContext();
        sh = new SessionFileHandler("ses.ser");
        fh = new GestureFileHandler("ges.ser");
```

```
        SessionRecord[] list = sh.getStoredSessions(this.getApplicationContext());
        String temp = "";
        for (int i = 0; i < list.length; i++) {
            temp = temp + ", " + list[i].getSession_name();
        }
        Toast.makeText(this.getApplicationContext(), temp, Toast.LENGTH_SHORT);

        ll = (LinearLayout) findViewById(R.id.view_session_layout);
        try {
            //createOptions();
            showSessions();
        } catch(Exception e) {
            Toast.makeText(myContext, e+"", Toast.LENGTH_LONG).show();
        }
    }

    private void createOptions() {

        SessionRecord[] list = sh.getStoredSessions(ViewSessions.this);

        /*Button deleteSessions = new Button(myContext);
        deleteSessions.setText("DELETE SESSIONS");
        deleteSessions.setLayoutParams(new LinearLayout.LayoutParams(LinearLayout.LayoutParams.MATCH_PARENT,
                LinearLayout.LayoutParams.WRAP_CONTENT));
        deleteSessions.setOnClickListener(new View.OnClickListener() {
            public void onClick(View v) {
               DeleteAllSessions();
            }
        });
        ll.addView(deleteSessions);*/

        LinearLayout[] options = new LinearLayout[list.length];
        for(int i = 0; i < options.length; i++) {

            LinearLayout mainl = new LinearLayout(myContext);
            mainl.setLayoutParams(new LinearLayout.LayoutParams(LinearLayout.LayoutParams.MATCH_PARENT,
                    LinearLayout.LayoutParams.WRAP_CONTENT));
            mainl.setOrientation(LinearLayout.HORIZONTAL);

            // sub 1
            LinearLayout subl1 = new LinearLayout(myContext);
            subl1.setLayoutParams(new LinearLayout.LayoutParams(LinearLayout.LayoutParams.MATCH_PARENT,
                    LinearLayout.LayoutParams.MATCH_PARENT, 0.5f));
            subl1.setOrientation(LinearLayout.VERTICAL);

            TextView session_name = new TextView(myContext);
            session_name.setText(Html.fromHtml("<b>NAME:</b> " + list[i].getSession_name()));
            session_name.setTextColor(Color.BLACK);
            subl1.addView(session_name);

            /*TextView session_date = new TextView(myContext);
            session_date.setText(Html.fromHtml("<b>DATE:</b>"));
            session_date.setTextColor(Color.BLACK);
            subl1.addView(session_date);

            TextView session_time = new TextView(myContext);
            session_time.setText(Html.fromHtml("<b>TIME:</b>"));
            session_time.setTextColor(Color.BLACK);
            subl1.addView(session_time);*/

            TextView session_scheds = new TextView(myContext);
            session_scheds.setText(Html.fromHtml("<b>SCHEDULES:</b>"));
            session_scheds.setTextColor(Color.BLACK);
            subl1.addView(session_scheds);

            TextView actual_scheds = new TextView(myContext);
            String scheds = "";
            Calendar[] callist = list[i].getSessionCalendar();
            for(int j = 0; j < callist.length; j++) {
                try {
                    Date date = callist[j].getTime();
                    SimpleDateFormat format1 = new SimpleDateFormat("MMM dd, yyyy hh:mm aaa");
                    scheds = scheds + format1.format(date) + "\n";
                } catch (Exception e) {
                    Toast.makeText(ViewSessions.this, e +"", Toast.LENGTH_SHORT).show();
                }
            }
            actual_scheds.setText(Html.fromHtml(scheds));
            actual_scheds.setTextColor(Color.BLACK);
            subl1.addView(actual_scheds);

            // sub 2
            LinearLayout subl2 = new LinearLayout(myContext);
            subl2.setLayoutParams(new LinearLayout.LayoutParams(LinearLayout.LayoutParams.MATCH_PARENT,
                    LinearLayout.LayoutParams.MATCH_PARENT));
            subl2.setOrientation(LinearLayout.VERTICAL);

            TextView gestures_to_perform = new TextView(myContext);
            gestures_to_perform.setText(Html.fromHtml("<b>GESTURES PERFORMED</b>"));
            gestures_to_perform.setTextColor(Color.BLACK);
            subl2.addView(gestures_to_perform);
```

```
            String[] session_sequence = list[i].getSession_sequence();
            GestureRecord[] gr = fh.getStoredGestures(myContext);

            int accum = 0;
            for(int j = 0; j < session_sequence.length -1; j++) {
                accum += 1;
                if(session_sequence[j] != session_sequence[j +1]) {
                    TextView gesture = new TextView(myContext);
                    gesture.setText(Html.fromHtml("Perform <b>" + fh.getRecordByID(session_sequence[j], ViewSession
                    gesture.setTextColor(Color.BLACK);
                    subl2.addView(gesture);
                    accum = 0;
                }
            }
            if(accum > 0) {
                accum +=1;
                TextView gesture = new TextView(myContext);
                gesture.setText(Html.fromHtml("Perform <b>" + fh.getRecordByID(session_sequence[session_sequence.le
                gesture.setTextColor(Color.BLACK);
                subl2.addView(gesture);
            }

            ScrollView sc = new ScrollView(myContext);
            LinearLayout.LayoutParams params = new LinearLayout.LayoutParams(
                    LinearLayout.LayoutParams.MATCH_PARENT, 400);
            params.weight = 0.5f;
            sc.setLayoutParams(params);
            sc.addView(subl2);

            Button deleteSession = new Button(ViewSessions.this);
            deleteSession.setText("DELETE");
            deleteSession.setLayoutParams(new LinearLayout.LayoutParams(250, 250));
            deleteSession.setOnClickListener(new ForDeleting(i));

            Button startSession = new Button(ViewSessions.this);
            startSession.setText("START");
            startSession.setLayoutParams(new LinearLayout.LayoutParams(250, 250));
            startSession.setOnClickListener(new ForPerforming(i));

            mainl.addView(subl1);
            mainl.addView(sc);
            mainl.addView(deleteSession);
            mainl.addView(startSession);
            mainl.setBackground(getResources().getDrawable(R.drawable.border_background));
            mainl.setPadding(15, 15, 15, 15);

            ll.addView(mainl);
            //mainl.setClickable(true);
            //mainl.setOnClickListener(new ForPerforming(i));

            mainl.setPadding(40, 20, 40, 20);
        }

    }

    private void showSessions() {
        LinearLayout main = new LinearLayout(ViewSessions.this);
        main.setLayoutParams(new LinearLayout.LayoutParams(LinearLayout.LayoutParams.MATCH_PARENT,
                LinearLayout.LayoutParams.WRAP_CONTENT));
        main.setOrientation(LinearLayout.VERTICAL);

        SessionRecord[] sr = sh.getStoredSessions(ViewSessions.this);

        for(int i = 0; i < sr.length; i++) {
            LinearLayout subelem = new LinearLayout(ViewSessions.this);
            subelem.setLayoutParams(new LinearLayout.LayoutParams(LinearLayout.LayoutParams.MATCH_PARENT,
                    LinearLayout.LayoutParams.WRAP_CONTENT));
            subelem.setOrientation(LinearLayout.VERTICAL);

            TextView session_name = new TextView(ViewSessions.this);
            session_name.setText(Html.fromHtml("<b> Session Name: </b>" + sr[i].getSession_name() + "<br>"));
            session_name.setTextColor(Color.BLACK);
            session_name.setGravity(Gravity.CENTER);

            TextView session_description = new TextView(ViewSessions.this);
            session_description.setText(Html.fromHtml("<b> Session Description: </b> <br>" + sr[i].getSession_descr
            session_description.setTextColor(Color.BLACK);
            session_description.setGravity(Gravity.CENTER);

            TextView session_schedules = new TextView(ViewSessions.this);
            String schedperline = "<b> Session Schedules: </b> <br>";
            for(int j = 0; j < sr[i].getSessionCalendar().length; j++) {
                Date date = sr[i].getSessionCalendar()[j].getTime();
                SimpleDateFormat format1 = new SimpleDateFormat("MMM dd, yyyy hh:mm aaa");
                schedperline = schedperline + format1.format(date) + "<br>";
            }
            session_schedules.setText(Html.fromHtml(schedperline));
            session_schedules.setTextColor(Color.BLACK);
            session_schedules.setGravity(Gravity.CENTER);

            TextView session_gestures = new TextView(ViewSessions.this);
            String[] gesture_sequence = sr[i].getSession_sequence();
```

```
            String sequenceperline = "<b> Gestures to be Performed: </b> <br>";
            int counter = 0;
            for(int j = 0 ; j < gesture_sequence.length -1; j++) {
                counter = counter +1;
                if(!gesture_sequence[j].equals(gesture_sequence[j +1])) {
                    sequenceperline = sequenceperline + "Perform <b>" + fh.getRecordByID(gesture_sequence[j], ViewS
                                      counter + " times <br>";
                    counter = 0;
                }
            }
            counter = counter +1;
            sequenceperline = sequenceperline + "Perform <b>" + fh.getRecordByID(gesture_sequence[gesture_sequence.
                              counter + " " + " times <br>";

            session_gestures.setText(Html.fromHtml(sequenceperline));
            session_gestures.setTextColor(Color.BLACK);
            session_gestures.setGravity(Gravity.CENTER);

            LinearLayout sub1 = new LinearLayout(ViewSessions.this);
            sub1.setLayoutParams(new LinearLayout.LayoutParams(LinearLayout.LayoutParams.WRAP_CONTENT,
                    LinearLayout.LayoutParams.WRAP_CONTENT));
            sub1.setOrientation(LinearLayout.HORIZONTAL);
            sub1.setPadding(300, 0, 0 ,0);

            ImageButton startSession = new ImageButton(ViewSessions.this);
            startSession.setImageResource(R.drawable.ic_play_circle_outline_black_24dp);
            startSession.setBackgroundColor(Color.parseColor("#47000b"));
            startSession.setLayoutParams(new LinearLayout.LayoutParams(200, 200));
            startSession.setOnClickListener(new ForPerforming(i));

            ImageButton deleteSession = new ImageButton(ViewSessions.this);
            deleteSession.setImageResource(R.drawable.ic_delete_black_24dp);
            deleteSession.setBackgroundColor(Color.parseColor("#47000b"));
            deleteSession.setLayoutParams(new LinearLayout.LayoutParams(200, 200));
            deleteSession.setOnClickListener(new ForDeleting(i));

            sub1.addView(startSession);
            sub1.addView(deleteSession);

            subelem.addView(session_name);
            subelem.addView(session_description);
            subelem.addView(session_schedules);
            subelem.addView(session_gestures);
            subelem.addView(sub1);
            subelem.setBackground(getResources().getDrawable(R.drawable.border_background));
            subelem.setPadding(40, 40, 40, 40);

            main.addView(subelem);
        }

        ScrollView sv = new ScrollView(ViewSessions.this);
        sv.addView(main);

        setContentView(sv);
    }

    /*private void DeleteAllSessions() {
        sh.DeleteFile(myContext);
        ll.removeAllViews();
        createOptions();
    }*/

    private void performSession(int index) {
        Intent newActivity = new Intent(ViewSessions.this, RecordActivity.class);

        String[] session_sequence = sh.getStoredSessions(ViewSessions.this)[index].getSession_sequence();
        double[] scores = new double[session_sequence.length];

        newActivity.putExtra("session_seq", session_sequence);
        newActivity.putExtra("scores", scores);
        newActivity.putExtra("session_index", 0);
        newActivity.putExtra("used_for", "multiple");
        newActivity.putExtra("session_name", index);

        Long start = System.currentTimeMillis();

        newActivity.putExtra("start_time", start);
        startActivity(newActivity);
    }

    private void deleteSession(int index) {
        sh.DeleteAtIndex(index, ViewSessions.this);
        //ll.removeAllViews();
        showSessions();
    }

    private class ForPerforming implements View.OnClickListener{
        private int index;
        public ForPerforming(int index){
            this.index = index;
        }
```

```
        @Override
        public void onClick(View v){
            performSession(index);
        }
    }

    private class ForDeleting implements View.OnClickListener{
        private int index;
        public ForDeleting(int index){
            this.index = index;
        }

        @Override
        public void onClick(View v){
            deleteSession(index);
        }
    }

}


package com.example.sirwheistein.gesturerecognizer;

import android.content.Context;
import android.content.Intent;
import android.graphics.Color;
import android.net.Uri;
import android.os.Bundle;
import android.os.Environment;
import android.provider.MediaStore;
import android.support.v7.app.AppCompatActivity;
import android.text.Html;
import android.view.Gravity;
import android.view.View;
import android.widget.ImageButton;
import android.widget.LinearLayout;
import android.widget.ScrollView;
import android.widget.TextView;
import android.widget.Toast;
import android.widget.VideoView;

import java.io.File;

public class ViewStoredGesturesActivity extends AppCompatActivity {

    private GestureFileHandler fh;
    private ScrollView parent;
    private Context myContext;

    private VideoView[] vid_list;
    private File dir;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        myContext = this.getApplicationContext();
        try {
            fh = new GestureFileHandler("ges.ser");
            dir = Environment.getExternalStoragePublicDirectory(Environment.DIRECTORY_DCIM);
            update();
        } catch(Exception e) {
            Toast.makeText(this.getApplicationContext(), e + "", Toast.LENGTH_SHORT).show();
        }
    }

    private void update() {
        parent = new ScrollView(this.getApplicationContext());
        GestureRecord[] storedGestures = fh.getStoredGestures(this.getApplicationContext());
        //Toast.makeText(myContext, storedGestures.length + "", Toast.LENGTH_SHORT).show();

        LinearLayout subparent = new LinearLayout(ViewStoredGesturesActivity.this);
        subparent.setOrientation(LinearLayout.VERTICAL);
        subparent.setGravity(Gravity.CENTER);
        vid_list = new VideoView[storedGestures.length];

        for(int i = 0; i < storedGestures.length; i++) {
            //Toast.makeText(myContext, storedGestures[i].getGestureName(), Toast.LENGTH_SHORT).show();

            LinearLayout subl = new LinearLayout(myContext);
            LinearLayout.LayoutParams lpp = new LinearLayout.LayoutParams(LinearLayout.LayoutParams.MATCH_PARENT,
                    LinearLayout.LayoutParams.WRAP_CONTENT);
            lpp.setMargins(0, 10, 0, 10);
            subl.setLayoutParams(lpp);
            subl.setOrientation(LinearLayout.VERTICAL);

            LinearLayout subl1 = new LinearLayout(myContext);
            subl1.setLayoutParams(new LinearLayout.LayoutParams(LinearLayout.LayoutParams.MATCH_PARENT,
                    LinearLayout.LayoutParams.MATCH_PARENT, 0.5f));
            subl1.setOrientation(LinearLayout.HORIZONTAL);

            vid_list[i] = new VideoView(this.getApplicationContext());
            LinearLayout.LayoutParams lp = new LinearLayout.LayoutParams(new LinearLayout.LayoutParams(420, 750));
```

```
        lp.setMargins(200, 20, 0, 20);
        vid_list[i].setLayoutParams(lp);

        LinearLayout rightsub1 = new LinearLayout(myContext);
        LinearLayout.LayoutParams lp2 = new LinearLayout.LayoutParams(new LinearLayout.LayoutParams(250, 750));
        lp2.setMargins(0, 20, 0, 0);
        rightsub1.setLayoutParams(lp2); //w h
        rightsub1.setOrientation(LinearLayout.VERTICAL);

        File output = new File(dir, storedGestures[i].getGestureID() + ".mp4");
        Uri videoUri = Uri.fromFile(output);
        if(output.exists()) {
            vid_list[i].setVideoURI(videoUri);
        }

        ImageButton recordButton = new ImageButton(myContext);
        recordButton.setLayoutParams(new LinearLayout.LayoutParams(250, 250));
        recordButton.setImageResource(R.drawable.ic_fiber_manual_record_black_24dp);
        recordButton.setBackgroundColor(Color.parseColor("#47000b"));
        recordButton.setOnClickListener(new ForStorage(i));

        ImageButton playButton = new ImageButton(myContext);
        playButton.setLayoutParams(new LinearLayout.LayoutParams(250, 250));
        playButton.setImageResource(R.drawable.ic_play_circle_outline_black_24dp);
        playButton.setBackgroundColor(Color.parseColor("#47000b"));
        playButton.setOnClickListener(new ForReading(i));

        rightsub1.addView(recordButton);
        rightsub1.addView(playButton);
        if(!isInSession(storedGestures[i])) {
            ImageButton deleteButton = new ImageButton(myContext);
            deleteButton.setLayoutParams(new LinearLayout.LayoutParams(250, 250));
            deleteButton.setImageResource(R.drawable.ic_delete_black_24dp);
            deleteButton.setBackgroundColor(Color.parseColor("#47000b"));
            deleteButton.setOnClickListener(new ForDelete(i));
            rightsub1.addView(deleteButton);
        }
        subl1.addView(vid_list[i]);
        subl1.addView(rightsub1);

        TextView ges_name = new TextView(ViewStoredGesturesActivity.this);
        ges_name.setText(Html.fromHtml("<b>GESTURE NAME: </b>" + storedGestures[i].getGestureName()));
        ges_name.setGravity(Gravity.CENTER);
        ges_name.setTextColor(Color.parseColor("#000000"));

        TextView ges_id = new TextView(ViewStoredGesturesActivity.this);
        ges_id.setText(Html.fromHtml("<b>GESTURE ID: </b>" + storedGestures[i].getGestureID()));
        ges_id.setGravity(Gravity.CENTER);

        subl.addView(ges_name);
        //subl.addView(ges_id);
        subl.addView(subl1);

        subparent.addView(subl);
    }

    if(storedGestures.length <= 0) {
        TextView warning = new TextView(ViewStoredGesturesActivity.this);
        warning.setText(Html.fromHtml("NO GESTURES STORED"));
        warning.setGravity(Gravity.CENTER);
        warning.setTextColor(Color.parseColor("#000000"));

        subparent.addView(warning);
    }

    /*Button deleteListButton = new Button(myContext);
    deleteListButton.setText("DELETE ENTIRE LIST");
    deleteListButton.setOnClickListener(new View.OnClickListener() {
        public void onClick(View v) {
            fh.DeleteFile(myContext);
            update();
        }
    });*/

    ImageButton mainMenuButton = new ImageButton(myContext);
    mainMenuButton.setLayoutParams(new LinearLayout.LayoutParams(250, 250));
    mainMenuButton.setImageResource(R.drawable.maroon_home);
    mainMenuButton.setBackgroundColor(Color.parseColor("#FFFFFF"));
    mainMenuButton.setOnClickListener(new View.OnClickListener() {
        public void onClick(View v) {
            Intent newActivity = new Intent(myContext, MainActivity.class);
            newActivity.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP);
            startActivity(newActivity);
        }
    });

    //parent.addView(deleteListButton);
    subparent.addView(mainMenuButton);
    parent.addView(subparent);
    parent.setBackgroundColor(Color.parseColor("#FFFFFF"));

    setContentView(parent);
```

```
    }

    private void StoreVideo(int index) {
        GestureRecord[] list = fh.getStoredGestures(myContext);
        File output = new File(dir, list[index].getGestureID() + ".mp4");
        //Toast.makeText(myContext, list[index].getGestureName(), Toast.LENGTH_SHORT).show();
        Intent intent = new Intent(MediaStore.ACTION_VIDEO_CAPTURE);
        intent.putExtra(MediaStore.EXTRA_OUTPUT, Uri.fromFile(output));

        startActivity(intent);

        if(output.exists()) {
            Uri videoUri = Uri.fromFile(output);
            vid_list[index].setVideoURI(videoUri);
        }
    }

    public boolean isInSession(GestureRecord gr) {
        SessionFileHandler sh = new SessionFileHandler("ses.ser");
        SessionRecord[] sr = sh.getStoredSessions(ViewStoredGesturesActivity.this);
        for(int i = 0; i < sr.length; i++) {
            String[] ss = sr[i].getSession_sequence();
            for(int j = 0; j < ss.length; j++) {
                if(gr.getGestureID().equals(ss[j])) {
                    return true;
                }
            }
        }
        return false;
    }

    public void SetUri() {
        GestureRecord[] gr = fh.getStoredGestures(ViewStoredGesturesActivity.this);
        for(int i = 0; i < gr.length; i++) {
            File output = new File(dir, gr[i].getGestureID() + ".mp4");
            if(output.exists()) {
                Uri videoUri = Uri.fromFile(output);
                vid_list[i].setVideoURI(videoUri);
            }
        }
    }

    private void ReadVideo(int index) {
        SetUri();
        vid_list[index].start();
        //Toast.makeText(myContext, vid_index, Toast.LENGTH_SHORT).show();
    }

    private class ForDelete implements View.OnClickListener{
        private int index;
        public ForDelete(int index){
            this.index = index;
        }

        @Override
        public void onClick(View v){
            fh.DeleteAtIndex(index, ViewStoredGesturesActivity.this);
            update();
        }
    }

    private class ForStorage implements View.OnClickListener{
        private int index;
        public ForStorage(int index){
           this.index = index;
        }

        @Override
        public void onClick(View v){
            StoreVideo(index);
        }
    }

    private class ForReading implements View.OnClickListener{
        private int index;
        public ForReading(int index){
            this.index = index;
        }

        @Override
        public void onClick(View v){
            ReadVideo(index);
        }
    }
}


<?xml version="1.0" encoding="utf-8"?>
<ScrollView
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
```

```
    android:background="@drawable/border_background"
    android:padding="15dp"
    android:scrollbars="none">

    <LinearLayout android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:orientation="vertical">

        <TextView
            android:text="RAW DATA"
            android:textSize="15dp"
            android:textStyle="bold"
            android:textColor="#47000b"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:gravity="center"
            android:layout_marginTop="15dp" />

        <com.github.mikephil.charting.charts.LineChart
            android:id="@+id/raw_graph"
            android:layout_width="match_parent"
            android:layout_height="250dp"
            android:layout_marginRight="50dp"
            android:layout_marginLeft="50dp"
            android:layout_marginTop="5dp" />

        <TextView
            android:text="FILTERED DATA"
            android:textSize="15dp"
            android:textStyle="bold"
            android:textColor="#47000b"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:gravity="center"
            android:layout_marginTop="15dp" />

        <com.github.mikephil.charting.charts.LineChart
            android:id="@+id/lpf_graph"
            android:layout_width="match_parent"
            android:layout_height="250dp"
            android:layout_marginRight="50dp"
            android:layout_marginTop="5dp"
            android:layout_marginLeft="50dp" />
        <LinearLayout
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:orientation="horizontal"
            android:paddingBottom="50dp"
            android:paddingTop="10dp"
            android:paddingLeft="15dp"
            android:paddingRight="15dp">
            <ImageButton
                android:id="@+id/Store"
                android:layout_width="80dp"
                android:layout_height="50dp"
                android:layout_marginLeft="45dp"
                android:text="STORE"
                android:src="@drawable/ic_save_black_24dp"
                android:background="#47000b"
                />
            <ImageButton
                android:id="@+id/Compare"
                android:layout_width="80dp"
                android:layout_height="50dp"
                android:text="COMPARE"
                android:src="@drawable/compare"
                android:background="#47000b"
                />
            <ImageButton
                android:id="@+id/main_menu_button"
                android:layout_width="80dp"
                android:layout_height="50dp"
                android:src="@drawable/ic_home_black_24dp"
                android:text="MAIN MENU"
                android:background="#47000b"
                />

        </LinearLayout>


    </LinearLayout>

</ScrollView>


<?xml version="1.0" encoding="utf-8"?>
<LinearLayout android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:background="@drawable/border_background"
    android:padding="15dp">
```

```
    <ToggleButton
        android:id="@+id/toggle_it"
        android:layout_width="120dp"
        android:layout_height="50dp"
        android:textOff="PT MODE"
        android:textOn="PATIENT MODE"
        android:layout_marginLeft="120dp"
        android:background="#47000b"
        android:textColor="#FFFFFF"/>

    <LinearLayout
        android:layout_width="80dp"
        android:layout_height="wrap_content"
        android:orientation="vertical">

        <ImageButton
            android:id="@+id/record_gesture_button"
            android:layout_width="80dp"
            android:layout_height="80dp"
            android:text="Record Gestures"
            android:layout_marginTop="10dp"
            android:background="#47000b"
            android:src="@drawable/ic_gesture_black_big"
            />

        <ImageButton
            android:id="@+id/view_stored_gestures_button"
            android:layout_width="80dp"
            android:layout_height="80dp"
            android:layout_marginTop="15dp"
            android:text="View Stored Gestures"
            android:background="#47000b"
            android:src="@drawable/ic_subscriptions_black_24dp"
            android:visibility="invisible"
            />

        <ImageButton
            android:id="@+id/set_session_button"
            android:layout_width="80dp"
            android:layout_height="80dp"
            android:layout_marginTop="15dp"
            android:text="Set Session"
            android:background="#47000b"
            android:src="@drawable/date_main"
            />

        <ImageButton
            android:id="@+id/view_session_button"
            android:layout_width="80dp"
            android:layout_height="80dp"
            android:text="View Sessions"
            android:layout_marginTop="15dp"
            android:background="#47000b"
            android:src="@drawable/session"
            android:visibility="invisible"
            />
        <ImageButton
            android:id="@+id/pick_motivation"
            android:layout_marginTop="15dp"
            android:layout_width="80dp"
            android:layout_height="80dp"
            android:text="View Sessions"
            android:background="#47000b"
            android:src="@drawable/motiv"
            android:visibility="invisible"
            />


    </LinearLayout>

</LinearLayout>


<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:background="@drawable/border_background"
    android:padding="15dp">

    <TextView
        android:layout_centerHorizontal="true"
        android:id="@+id/gesture_instruction"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Perform Gesture"
        android:textColor="#47000b"
        android:textSize="30dp"
        android:textStyle="bold"
```

```
        android:layout_marginTop="45dp"/>

    <at.markushi.ui.CircleButton
        android:id="@+id/Record"
        android:layout_centerInParent="true"
        android:layout_width="200dp"
        android:layout_height="200dp"
        app:cb_color="#47000b"
        app:cb_pressedRingWidth="16dp"
        android:src="@drawable/ic_gesture_black_24dp"
        />

</RelativeLayout>


<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical" android:layout_width="match_parent"
    android:layout_height="match_parent">

    <EditText
        android:id="@+id/session_results"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="hehe"/>

</LinearLayout>


<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical" android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@drawable/border_background"
    android:padding="15dp">

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="horizontal">

        <TextView
            android:text="Session Name:"
            android:textSize="15dp"
            android:textStyle="bold"
            android:textColor="#47000b"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginLeft="10dp"/>
        <EditText
            android:id="@+id/session_name"
            android:layout_width="232dp"
            android:layout_height="40dp"
            android:textSize="15dp"
            android:layout_marginLeft="10dp"
            android:textAlignment="center"/>
    </LinearLayout>

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="vertical">

        <TextView
            android:text="Session Description:"
            android:textSize="15dp"
            android:textStyle="bold"
            android:textColor="#47000b"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginLeft="10dp"
            android:layout_gravity="center"/>
        <EditText
            android:id="@+id/session_description"
            android:layout_width="match_parent"
            android:layout_height="40dp"
            android:textSize="15dp"
            android:layout_marginLeft="10dp"
            android:textAlignment="center"
            android:layout_gravity="center"/>
    </LinearLayout>

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="horizontal">

        <TextView
            android:text="Add Schedule:"
            android:textSize="15dp"
            android:textStyle="bold"
            android:textColor="#47000b"
            android:layout_width="wrap_content"
```

```
            android:layout_height="wrap_content"
            android:layout_marginTop="10dp"
            android:layout_marginLeft="10dp"/>

        <ImageButton
            android:id="@+id/set_date_button"
            android:layout_width="50dp"
            android:layout_height="40dp"
            android:text="SET DATE"
            android:src="@drawable/ic_date_range_black_24dp"
            android:layout_marginLeft="8dp"/>

        <LinearLayout
            android:layout_width="160dp"
            android:layout_height="wrap_content"
            android:background="@drawable/border_background"
            android:padding="20dp"
            android:layout_marginLeft="8dp"
            android:orientation="vertical">

            <TextView
                android:text="Schedules:"
                android:textSize="15dp"
                android:textStyle="bold"
                android:textColor="#47000b"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_gravity="center"
                />
            <TextView
                android:id="@+id/list_schedules"
                android:textSize="15dp"
                android:textStyle="bold"
                android:textColor="#000000"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_gravity="center"
                />

        </LinearLayout>

    </LinearLayout>

    <!--<LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="horizontal">

        <TextView
            android:text="Session Time:"
            android:textSize="15dp"
            android:textStyle="bold"
            android:textColor="#47000b"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginTop="10dp"
            android:layout_marginLeft="10dp"/>

        <ImageButton
            android:id="@+id/set_time_button"
            android:layout_width="50dp"
            android:layout_height="40dp"
            android:text="SET DATE"
            android:src="@drawable/ic_access_alarm_black_24dp"
            android:layout_marginLeft="5dp"/>

        <TextView
            android:id="@+id/time_picked"
            android:text="12:00 pm"
            android:textSize="15dp"
            android:textStyle="bold"
            android:textColor="#47000b"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginTop="10dp"
            android:layout_marginLeft="10dp"/>
    </LinearLayout>-->

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:background="@drawable/border_background"
        android:padding="15dp"
        android:layout_marginTop="10dp"
        android:orientation="vertical">

        <LinearLayout
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:orientation="horizontal">

            <TextView
```

```
            android:text="Select Gesture to Add:"
            android:textSize="15dp"
            android:textStyle="bold"
            android:textColor="#47000b"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginLeft="10dp" />

        <Spinner
            android:id="@+id/gesture_choices"
            android:layout_width="150dp"
            android:layout_height="20dp"
            android:text="Session Name"
            android:textSize="15dp"
            android:layout_marginLeft="10dp"
            android:textAlignment="center">
        </Spinner>

    </LinearLayout>

    <LinearLayout
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:orientation="horizontal">

        <TextView
            android:text="Select Quantity:"
            android:textSize="15dp"
            android:textStyle="bold"
            android:textColor="#47000b"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginLeft="50dp"
            />

        <ImageButton
            android:id="@+id/add_button"
            android:layout_width="40dp"
            android:layout_height="40dp"
            android:src="@drawable/ic_add_black_24dp"/>

        <EditText
            android:id="@+id/number_of_performances"
            android:layout_width="50dp"
            android:layout_height="40dp"
            android:inputType="number"
            android:textSize="15dp"
            android:textAlignment="center"
            android:text="0"
            android:editable="false"/>

        <ImageButton
            android:id="@+id/subt_button"
            android:layout_width="40dp"
            android:layout_height="40dp"
            android:src="@drawable/ic_remove_black_24dp"/>
    </LinearLayout>

    <Button
        android:id="@+id/add_to_list"
        android:layout_width="150dp"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:text="ADD TO LIST"
        android:layout_marginTop="5dp"
        android:layout_marginBottom="5dp"/>

</LinearLayout>

<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="horizontal"
    android:layout_marginTop="10dp">

    <LinearLayout
        android:layout_width="200dp"
        android:layout_height="wrap_content"
        android:background="@drawable/border_background"
        android:padding="20dp"
        android:orientation="vertical">

        <TextView
            android:text="Gestures in List:"
            android:textSize="15dp"
            android:textStyle="bold"
            android:textColor="#47000b"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_gravity="center"
            />
        <TextView
```

```
                android:id="@+id/list_elements"
                android:textSize="15dp"
                android:textStyle="bold"
                android:textColor="#000000"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_gravity="center"
                />

        </LinearLayout>

        <Button
            android:id="@+id/store_session"
            android:layout_width="100dp"
            android:layout_height="80dp"
            android:text="STORE SESSION"
            android:layout_marginLeft="20dp"/>

    </LinearLayout>


</LinearLayout>


<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:id="@+id/view_session_layout"
    android:background="@drawable/border_background"
    android:padding="15dp">



    <!--
    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="horizontal">

        <LinearLayout
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:layout_weight="0.5"
            android:orientation="vertical">

            <TextView
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="NAME: SESSION NAME"
                android:textStyle="bold"/>
            <TextView
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="DATE: SESSION DATE"
                android:textStyle="bold"/>
            <TextView
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="TIME: SESSION TIME"
                android:textStyle="bold"/>
        </LinearLayout>

        <ScrollView
            android:layout_width="match_parent"
            android:layout_height="70dp"
            android:layout_weight="0.5">

            <LinearLayout
                android:layout_width="match_parent"
                android:layout_height="match_parent"
                android:orientation="vertical">
                <TextView
                    android:layout_width="wrap_content"
                    android:layout_height="wrap_content"
                    android:text="GESTURE SEQUENCE:"
                    android:textStyle="bold"/>

                <TextView
                    android:layout_width="wrap_content"
                    android:layout_height="wrap_content"
                    android:text="5 CIRCLES"/>
                <TextView
                    android:layout_width="wrap_content"
                    android:layout_height="wrap_content"
                    android:text="5 CIRCLES"/>
                <TextView
                    android:layout_width="wrap_content"
                    android:layout_height="wrap_content"
                    android:text="5 CIRCLES"/>
            </LinearLayout>
```

```
        </ScrollView>

    </LinearLayout>

    -->
</LinearLayout>


<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical" android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:id="@+id/view_gestures_layout">

</LinearLayout>


<TextView
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:textSize="16sp"
    android:textColor="#000"
    android:background="#FFF"
    />


<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical" android:layout_width="match_parent"
    android:layout_height="match_parent">

    <RadioGroup
        android:layout_width="match_parent"
        android:layout_height="match_parent">

        <RadioButton
            android:text="fak"
            android:layout_width="match_parent"
            android:layout_height="wrap_content" />
        <RadioButton
            android:text="fak"
            android:layout_width="match_parent"
            android:layout_height="wrap_content" />
        <RadioButton
            android:text="fak"
            android:layout_width="match_parent"
            android:layout_height="wrap_content" />

    </RadioGroup>

</LinearLayout>
```

XI. Acknowledgement

So heres to my final speech in College:

A Series of Thank Yous

Hey **God**! Thank you for the amazing twist of fate you have placed me in. Ive always told you how surprised I am to be here, nonetheless be even saying goodbye, to this wonderful institution you have directed me to. Thank you for the lessons because without them I wouldn't know how to win. I'll keep it up. Thank you again.

Papa, Mama. Salamat sa patuloy na pagsuporta at pagtitiwala. Alam kong marami akong maling ginawa sa talambuhay pero kahit doon ay sinuportahan niyo pa rin ako. Salamat. Salamat. Salamat. Pangako kong babawi ako. **Aldrin**, bro. Salamat sa pag intindi sa mga katamaran ko. **Yhamie**, salamat sa pagiging sweet kahit sobrang nakakaasar ka haha. Haha.

Ma'am **Sheila Magboo**, Salamat po! Salamat at pinili niyo akong advisee, pinaramdam niyo na sa dami dami ng pwedeng piliin ay ako. Hindi rin naman ako nagkamali sa pagpili sa inyo, dahil hinayaan niyo akong makadiskarte sa aking tinahak na pagsasaliksik at yun rin naman ang aking forte. Salamat rin sa mga papuri niyo, kahit kaunti lang ang ating mga meeting, sulit naman po ang mga iyon.

Adolf Gonzales, bro. Salamat talaga kasi kahit ang tagal na nating di nagkakausap, tinulungan mo ako dito sa aking pagsasaliksik. Talagang hinahype mo tong work ko kaya naman nagkaconfidence akong ituloy to. Bawi ako later.

Sir **Ralph Genoguin**, Thank you! Thanks po for entertaining my request na mainterview kayo. Dahil sa inyo nagkabackground at thorough understanding ako sa PT kaya naman nagawa ko yung kailangan gawin. Looking forward to collaborations wherever, whenever, if not busy.

KUMPANERO LIGHT, my squad, my team, my boys. (Alphabetical Order):

1. **Alfred Tacorda** - demi-adviser, busmate madalas, fxmate mga once
2. **Berwin Yu** - Master madalas, kain-mate minsan
3. **Elias Chico** - God-tier coder madalas, groupmate minsan
4. **Ironnel Costales** - co-alumni, pinakasolid, child-prodigy talaga
5. **Jeffrey Delos Reyes** - pinakamaasahan kong groupmate, pinakamatangkad
6. **Jethro Matubis** - tagacheer and tagacheer-up madalas, happy-go-lucky as me
7. **Kier Genoguin** - gym talk buddy, kain-mate minsan

And to top it all off, lahat kayo ka-"joketime minsan pero truepa forever." You always tell me that I created this group and yet you forget how you guys, in turn, helped create me. Salamat. Solid pa rin in years ah.

Ivan, salamat sa years bro. Salamat sa advices na wala naman kinalaman sa SP haha, kept me sane man. **Fadri**, solid kawentuhan to at ka SP-talks kaya salamat pare. **Neil**, forever blockhead and probably the person in ComSci, I know the longest. Salamat sa tiyaga samin tsong, tuloy mo yan haha. Sa mga cool friends ko **Zolets, Pabi, Janssen**, and etc. Salamat sa camaraderie, pansinin niyo ako pag famous na kayo.

Yo **Cap**! Thank you mostly for that "Good Luck" of yours which, honestly, worked a little too well. I don't know, it was just magic. I know you just listen most of the time, so listen to this. Thank you for helping me write a story, more specifically, our story. I've always held myself as a writer, a performer and a doer. I didn't really like sitting around, watching people do things when I can do them myself. I like being heard, more than being the one who listens. Chapter 1 has been written, this paper completes Chapter 2. So let me take this quote from a song.

"Here's where she meets Prince Charming

But she won't discover that it's him

'Til chapter three! "

Looking forward to happier Chapters, Cap. Hihi.

Last but the most (kase not the least). **Block 12**. Salamat sa tawanan, sa tampuhan, sa pikunan. Thank you for the friendships we've made through the years. Some stayed, some went away, and new people came in. Thank you for showing me a glimpse of what life is all about. Life's hard but it's manageable when we work together.