SOFTWARE ENGINEERING I

REPORT 1

Beat.My.Run

October 16, 2015

Nivetha Balasamy

Careena Braganza

Valia Kalokyri

Thara Philipson

Nirali Shah

Rahul Shome

Rutgers University

Contents

1 CUSTOMER STATEMENT OF REQUIREMENTS (CSR)

1.1 Problem Statement

Physical activity in any form improves physical health and mental well-being. Research[1] indicates that regular exercise can add up to five years to your life. Even a short burst of 10 minute aerobic exercise like brisk walking or running can improve mental alertness, energy and positive mood. Sticking to an exercise regime is difficult and motivation levels are low. Even exercise fanatics find it hard to get motivated from time to time. Thus, when physical activity is kept interesting by having a constant source of motivation it helps to keep the exercise regime firm, because if it's not fun it's not sustainable.

In the present era, things are easily made accessible. The internet of things and proliferation of sensors in handheld mobile devices allow access to various features in minimal clicks and minimal time. Thus, it would be a good option to gain motivation an application running on a mobile device.

Studies[2][3][4] have found that music reduces the perception of how hard you are running by about 10 percent. An external stimulus as exciting as music can actually block some of the internal stimuli trying to reach the brain-such as fatigue-related responses from muscles and organs. When these messages are blocked, reduced physical effort is experienced which encourages to run faster and further. Music also elevates positive aspects of mood[5][6] such as elation and happiness, and reduces negative aspects such as tension, fatigue, and confusion. It is common to find people exercising with their headphones on, listening to the music. Music makes exercise interesting, limits the feeling of compulsion and also makes it more productive. However, studies have shown that not just listening but controlling and creating music in time to one's pace had an even more profound effect on perceived effort during a workout. Synchrony can help the body use energy more efficiently [7] [8]. When moving rhythmically to a beat, the body would have to make as many adjustments to coordinated movements as it would without regular external cues. Music can function as a metronome, helping to maintain a steady pace, reducing false steps and decreasing energy expenditure.

Wanting to be fit but not feeling motivated enough is a common concern. It requires a lot of dedication and time allotment. Making the exercising experience better is essential. Standard music apps do not reason about the state of physical activity.Research shows that the correct music can enhance the exercising experience. User-defined playlists will be adapted to instantaneous rates of activity, ie. music ideal for jogging might not prove pleasurable while sprinting.

Towards this end, an application trying to enhance user's running experience by matching the music playback with the intensity of the workout would maintain the positive effect of music on the running experience. In order to achieve this, the application uses the embedded accelerometer of the phone to monitor the speed or velocity of the running or jogging activity of the user and based on the intensity can playback songs that match user's activity with the tempo of the song. For example, while running fast s/he will be able to listen to a high tempo song. In this solution, the cadence of the run has to be matched with an attribute of the music file selected such as the duration, energy or beats per minute.

In addition, the application can be made more personalized by playing songs that match user's preferences. In order to achieve this, the application can have access to the phone's music library and you-tube's playlists/songs that a user has previously liked or listened to. This way it can playback songs that the user enjoys and not something which is irrelevant to the user's taste.

Also, exercising might seem monotonous, which is mainly due to the lack of knowledge about workout.It leads to unnecessary stressing of the body and mind, thus reducing the time of exercise leading to faster exhaustion and mental stresses. Many a times acquiring a proper training and meeting the trainer's schedules is difficult which again keeps us from exercising due to the fear of not doing it correctly.

A feature that can add to the enhanced user experience would be allowing the user to choose from a predetermined workout template which could be designed to sequentially increase and decrease the intensity of running. For example, the exercise session can be broken down to a five minute jog followed by a short sprint for two minutes and another segment of a jog for three minutes[9]. This can be useful in cases when you don't know how to plan an effective workout where the application would be able to suggest some predefined template workout plans and allow to select one based on one's fitness level.

Again, running alone proves to be a boring and strenuous task and lacks motivation. By creating competitive situations, motivation levels increase and can enhance performance in exercise events[10]. Competitive conditions also augment the cortisol response to exercise, suggesting that enhanced sympatho-adrenal system activation occur in such situations which may be one of the key "driving forces" to performance improvement.

Exercise is generally targeted to maintain fitness but if it is modified to be a game which could challenge friends it would create additional drive and encouragement. In order to achieve that, the application is targeted towards maintaing personal running achievements (the distance travelled and the time taken), allow searching friends that use the same application,send out requests for challenges and also accept a posed challenge. The challenge could

be based on running time, meaning that the person challenegd should run the same distance in less time in order to win the challenge.

Furthermore, it is an undeniable fact that outdoor fitness training comes with many advantages. It provides opportunities to explore new places, breathe in fresh air and eliminates the need of a mundane indoor exercise routine like running or jogging on a treadmill.Hence, a feature which could provide real time suggestions on location of nearby parks and tracks would be an added advantage. It can be in the form of a map which can show the names and locations of the parks and gyms nearby to our current location. This gives enough information to plan a workout depending on the location in which the user is currently in.

However, when exercising outside, weather becomes an important consideration - extreme heat/cold and rain could be a hindrance to a productive outdoor workout. Weather data integration can become an added feature which could help make an informed decision. Temperature, heat advisory and climate changes could be checked and notified while going outdoors for a workout or even exactly at the the time the user starts running.

After all this, analysing a workout is the most effective way of providing encouraging information.It is always necessary to understand the previous performances and keep track of them in order to perform better. Comparisons of the workouts through a period of time can achieve exactly this. Hence, data visualizations of the workout pattern for a duration of time with regard to the distance covered during the run and the time taken can be a great way of keeping track. Last but not least, in a number of studies[11], researchers reported that one out of every 10 premature deaths around the planet are caused by not exercising. In other words, not exercising kills almost the same number of people as smoking does, and Time magazine went so far as to label the situation as a global pandemic. Specifically in the United States, the American Heart Association's "Circulation" research journal reports that approximately 250,000 deaths every year are caused by not exercising. To counteract these startling numbers, the journal highlights that many studies linking a lack of exercise to premature death recommend that people exercise three days per week for 30 to 60 minutes each day. This app would contribute towards helping the general population stick to their exercise regime and stay healthy.

The project proposes a personalized and user-friendly exercise application, supporting music playback that adapts to the intensity of the workout while running. Users can also get contextual information about weather and location, keep track of their workout attributes, compete with their friends and track high scores through leaderboards. The aim of the project is to promote physical well-being by enhancing the running experience and encouraging users to exercise.

1.2 Glossary of Terms

- **Beats per minute**: Beats per minute (BPM) is a unit typically used as a measure of tempo in music and heart rate. The BPM tempo of a piece of music is conventionally shown in its score as a metronome mark. This indicates that every one minute there should be 120 quarter notes.
- **Metronome**: A metronome is any device that produces regular, metrical ticks (beats, clicks) - settable in beats per minute. These ticks represent a fixed, regular aural pulse; some metronomes also include synchronized visual motion (e.g. pendulum-swing).
- **Pulse**: In music and music theory, the pulse consists of beats in a (repeating) series of identical yet distinct periodic short-duration stimuli perceived as points in time occurring at the mensural level.
- **Sprinting**: Sprinting is the act of running over a short distance at (or near) top speed. It is used in many sports that incorporate running, typically as a way of quickly reaching a target or goal, or avoiding or catching an opponent.
- **UI**: Its the user interface that in the industrial design field of human-machine interaction, is the space where interactions between humans and machines occur. The goal of this interaction is to allow effective operation and control of the machine from the human end, whilst the machine simultaneously feeds back information that aids the operators' decision making process.
- **Accelerometer**: An accelerometer is an electromechanical device used to measure acceleration forces.Acceleration is the measurement of the change in velocity, or speed divided by time.
- **Data Visualization**: Data visualization is the presentation of data in a pictorial or graphical format. For centuries, people have depended on visual representations such as charts and maps to understand information more easily and quickly.
- **Leaderboard**: A scoreboard showing the names and current scores of the leading competitors. Leaderboards can contain text, images, or even animations.
- **API**: In computer programming, an application programming interface (API) is a set of routines, protocols, and tools for building software applications. An API expresses a software component in terms of its operations, inputs, outputs, and underlying types.

2 USER STORIES

The functional and non-functions requirements have been enumerated from the point of view of the user as user stories

- **Logging in:** The user can log into the application using his *Google* account credentials. Once the login credentials are verified, the user can access all the features of the app. The *Google* account gives access to the *Youtube* account history and the music preferences of the user on *Youtube*.
- **Planning a run:** The user wants to schedule a run.
    1. *Running conditions:* The user can see an UI with the current weather conditions based on current location and a forecast over the next 3 hours.
    2. *Running locations:* The user can look for running locations around the vicinity like jogging tracks, parks and gyms. The UI shows a map along with the names and addresses of the locations.
- **Starting a run:** The user wants to initiate a new run. A new run can also initiated by accepting a challenge. The UI provides the current weather conditions based on location and a forecast for the next 3 hours. In addition, the UI provides a button to start the run.
    1. *Music Playback:* The app chooses a sound track based on the intensity of the user's running. The faster the user runs, the higher the tempo of the music becomes. The running rate is obtained over a window of time from accelerometer data. A change of tempo is reflected in the next song selection. The tracks are selected based on the user's *Youtube* history and playlists. The UI shows the track information and music controls to pause, play, skip and replay the last song.
    2. *Real-time Running Information:* The UI shows the current distance covered during the run and also the current duration of the run.
    3. *Stopping a Run:* There would be a button in the UI to stop the run. Once the run ends, the UI shows a summary of the completed run, including distance and the duration of the run. The UI also shows a map with the route covered during the run.
    4. *Challenging a Friend:* After the user stops a run the user can challenge a friend to run the same distance, but in a shorter time. The UI will let the user select friend/s from a list or search for friend/s by username and challenge them.

- **Training:** The app gives the option to exercise according to a predetermined workout template which progressively increases and decreases intensity of running. The workout template includes sprinting and jogging on and off during a run. The UI lets the user select between three inbuilt templates - Beginner, Intermediate and Advanced. For example , the exercise session can be broken down to a five minute jog followed by a short sprint for two minutes and another segment of jog for three minutes

    1. *Starting Training:* The user interface would display weather information based on the current location.
    2. *During Training:* The user interface would be identical to running except for the nature of music playback. The tracks would be selected based on the workout template and can provide audio cues for switching between training modes(jogging and sprinting).

- **Past Exercise Data:** The app aggregates past exercise data and provides an UI to display data visualizations of the data stored for the user. It can show the changes in the running distances and durations over time.

- **Challenges:** Challenges can be sent or accepted between friends.

    1. *Adding a Friend:* An UI lets the user search for the friend using the username and send a friend request. The user can also accept friend requests sent by other users.
    2. *Accepting a Challenge:* The user sees an UI with a list of pending challenges from friends. Once the user accepts a challenge, the user proceeds to the UI to start a run.
    3. *Challenge Result:* At the end of a challenge run, the user either wins or loses the challenge based on how much time the user took to cover the challenged distance. The UI shows the current run statistics and route covered. The UI will also notify the user of this result and also allow the user to use the current run to challenge other friends.
    4. *Leaderboards:* The results of challenges update a leaderboard UI which shows the win-lose statistics of the friends of the user, sorted by the number of wins in descending order.

Table 1: Priority weights of the requirements

ID	Priority Weight	Requirement
REQ-1	1	Logging in the System
REQ-2	2	Authentication from Google
REQ-3	5	Accessing YouTube Playlist
REQ-4	5	Start Phone Accelerometer
REQ-5	5	Extracting BPM from Music file
REQ-6	4	Sync Run with the Music BPM
REQ-7	4	Save the Statistics
REQ-8	3	Challenge Friends
REQ-9	2	Respond to Challenges

3 On-Screen Appearance Requirements

The following drawings, Figure 1 and 2 show paper prototypes of the way the screens appear to the user while s/he uses the various features of the App.

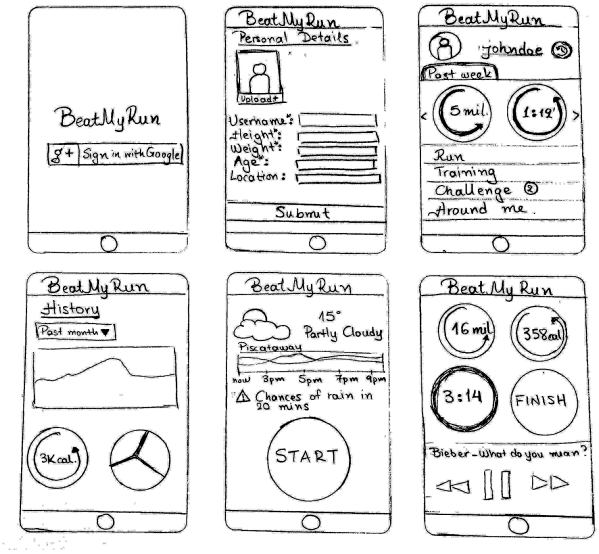

Figure 1: Paper Prototyping for the user stories

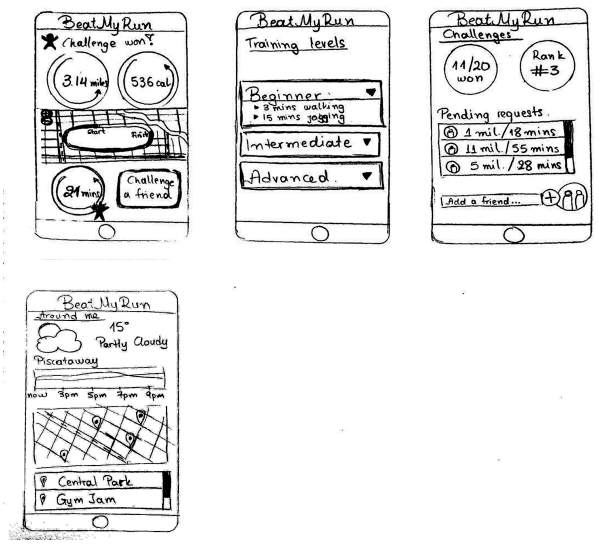

Figure 2: Continued....Paper Prototyping for the user stories

4 FUNCTIONAL REQUIREMENTS SPECIFICATIONS

4.1 Stakeholders

This is a personalized and user-friendly exercise application for a better physical health. This application aims in making exercise more interesting with its features. Every project has stakeholders and they are people who have an interest in the successful completion of the project. There are many different types of stakeholders, and they vary according to the project.The ideal stakeholders of this project are:

1. *End Users*: This application targets people who finds exercise boring and need motivation for the same.Achieving a goal or target with competitive spirit gives many users encouragement and motivation.Many people like to keep track of their workout pattern and compare it with friends.This application is ideal for such kind of users.

2. *Clients:* This application is competitive and can be an investment for many clients to invest or take up the project after its successful completion.

4.2 Actors and Goals

An actor can be human, physical object, or another system external to the application that interacts or participate with the system to achieve a goal. Each actor has a goal or responsibility that contribute to the total working of project.

Actor	Actor Goal	Use Case Name
Users	A logged in user accessing the application and its features	UC1
Friends	User can add friends and challenge them with a task that user has already accomplished	UC6
Echonest	This is used to find the bpm of a song that needs to be selected and played during workout	UC3
Android SDK	This is for collecting phone's accelerometer data in order to calculate user's running rate	UC4
Google and YouTube	Google account is for logging into application and youtube account is for getting user's prefered songs	UC1
Music Player	The song selected is played using music player with artist details and with stop,pause and next options.	UC4
Cloud Infrastructure Component	The details of the user while registering is stored and is used for various calculations.	UC1

4.3 Use cases

In a real world application, the various users communicate with the system to perform various functions. These components which play a role in the application to perform various functions as an initiator or a participant are called Usecases. A description of the Usecases,Diagramatic Representation and the Tracebility Matrix mapping each use case to the Requirements is shown below:

4.3.1 Casual Description

The summary use cases are as follows:

UC-1: Login - Allows the user to login into the system
UC-2: Personal Details - Allows the user to store his/her name, age, weight and height into the Database. («include» UC-1)
UC-3: Song Information Details - It obtains the beats per minutes of songs in the user's playlist.
UC-4: Run - Once the user has completed login and entered his personal details, s/he can begin his Run. In this use case, the app finds the speed of the user and plays a song with an equivalent beats per minute. («include» UC-3, «extend» UC-5)
UC-5: Train - Allows the user to select one of the three training levels of the system.That is beginner, intermediate and expert.
UC-6: Challenge - Allows the user to challenge one of his friends or one of the users of the application.
UC-7: Respond to Challenge - Allows he user to view pending challenge/s and accept/decline it/them.
UC-8 View Statistics - Displays all the information about the user like total miles run,calories burned,time and the challenge results. The history of data can also be viewed in statistics.

4.3.2 Use Case Diagram

The Use Case Diagram is a graphical representation of the broad ways in which the User can interact with the application. Use case diagrams are meant to capture the dynamic aspects of the system from a high level point of view. It is used to represent

- Requirements of a system.

- Present a holistic view of a system.
- Identify factors affecting the system, both external and internal.
- The interactions among actors and requirements

The diagram shown below represents the view of our system and how the user can interact with the Android app that speaks to external agents like the backend database, Google or Echonest.

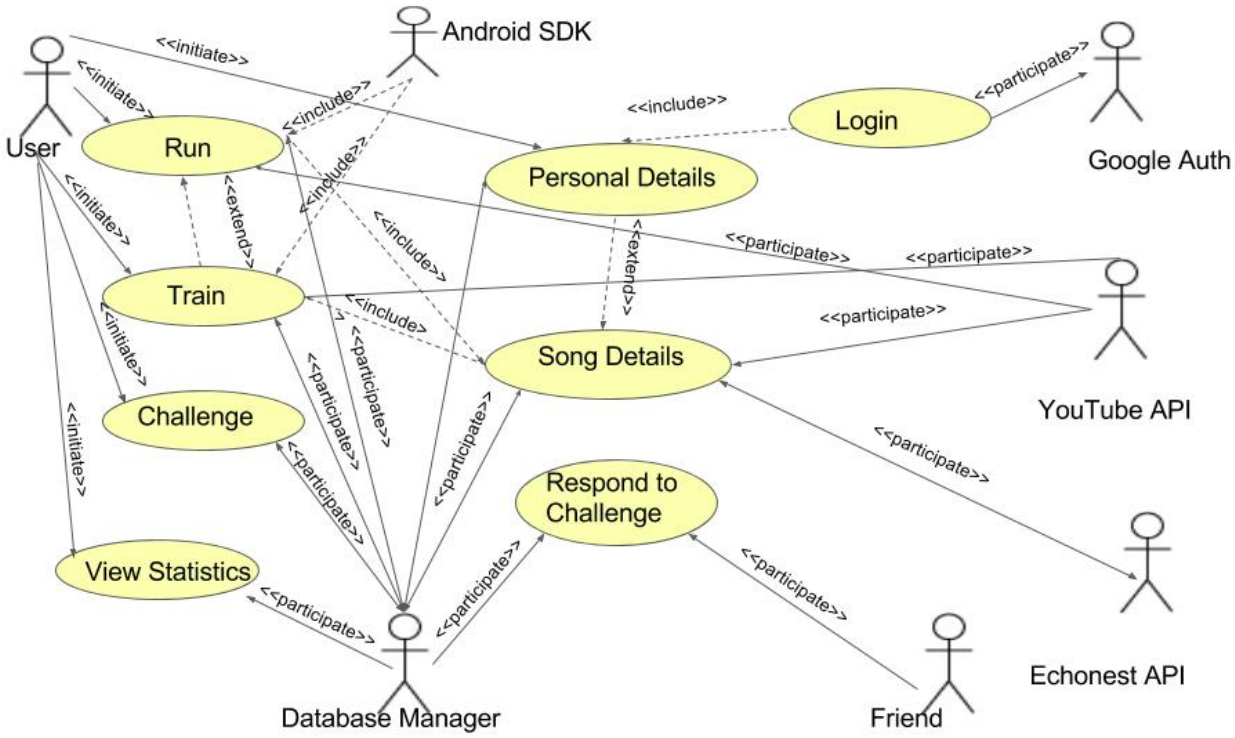

Figure 3: Use Case Diagram

4.3.3 Traceability Matrix

Traceability Matrix maps which Use Case is used to fulfill what Requirement. This way we know the functions of each Use Case and their responsibilities.Following Table shows the Traceability Matrix of the above described Use Cases.

Table 2: Traceability Matrix of Use Cases versus Requirements

REQ-N	PW	UC1	UC2	UC3	UC4	UC5	UC6	UC7	UC8
R1	1	X	-	-	-	-	-	-	-
R2	2	X	-	-	X	-	-	-	-
R3	5	-	-	X	X	X	-	-	-
R4	5	-	X	X	X	X	-	-	-
R5	5	X	-	X	X	X	-	-	-
R6	4	-	-	-	X	-	-	-	-
R7	4	X	-	-	-	X	-	-	X
R8	3	X	-	-	-	-	X	-	X
R9	2	-	-	-	-	-	X	X	-
MAX PW		4	5	4	5	4	3	2	3
TOTAL PW		19	10	19	26	23	10	4	10

4.3.4 Fully-Dressed Description

Below are the Tables giving an elaborate description of each use case. The use cases are elaborated to give a more detailed description on the purpose of the use case, the requirement which it maps to, the actor,the actor's goal, participating actor and the pre and post conditions needed for the particular use case.

Table 3: Use Case 1 : Login

Use Case	**Functionality**
Use Case 1	Login
Related Req	REQ-1, REQ-2, REQ-5, REQ-7, REQ-8.
Initiating Actor	User
Actors Goal	To login and access app features
Participating Actors	Google, Database
Pre conditions	1. The user has a Google account. 2. The UI displays a 'Sign in with Google account' choice.
Post conditions	1. The user's Google credentials validated. 2. A new Beat my Run account is created in the database for the user. 3. The user's Youtube history is accessible. 4. (a) If it is first time login, the user is prompted to enter personal details. (b) All the available features of the app are displayed on the screen.
Flow of events for Main Success Scenario	1. The user opens the app and is prompted to type in his Google account credentials. 2. The system verifies the Google credentials. 3. (a) For first time login, the personal details UI is displayed and the user is able to enter his details. (b) The UI displays a menu of features available from the app.
Flow of events for Extensions (Alternate Scenarios)	1. User enters invalid Google credentials. 2. Google authentication fails. 3. User is prompted to re-enter credentials.

Table 4: Use Case 2 : Personal Details

Use Case	**Functionality**
Use Case 2	Personal Details
Related Req	REQ-4.
Initiating Actor	User
Actors Goal	To store his personal details into the Database so that it can be used for 1. accessing songs from his/her Youtube playlist 2. calculating his speed based on no.of steps.
Participating Actors	Database
Pre conditions	1. The user has completed login. 2. The UI displays a form with the required fields of personal details.
Post conditions	The user's personal information is stored in the Database
Flow of events for Main Success Scenario	1. The user enters his/her required personal details into each field. 2. User submits the details. 3. The data is stored into the Database.

Table 5: Use Case 3 : Song Information Details

Use Case	**Functionality**
Use Case 3	Song Information Details
Related Req	REQ-3, REQ-4, REQ-5.
Initiating Actor	System
Actors Goal	To obtain the beats per minute of songs in the user's playlist.
Participating Actors	Echonest, Database
Pre conditions	1. The Google authentication is completed. 2. The song names from the user's Youtube playlist are available in the Database.
Post conditions	The beats per minute of all songs from the user's playlist are available in the Database.
Flow of events for Main Success Scenario	1. Echonest Initial API Connection is established. 2. The beats per minute of all songs from the user's playlist are obtained from Echonest. 3. The beats per minute of each song is stored in the Database.

Table 6: Use Case 4 : Run

Use Case	**Functionality**
Use Case 4	Run
Related Req	REQ-2, REQ-3, REQ-4, REQ-5, REQ-6.
Initiating Actor	User
Actors Goal	To go for a run
Participating Actors	Android SDK, Database, Youtube
Pre conditions	1. The user is logged into the system. 2. The user's personal details are stored in the Database. 3. The beats per minute of songs (obtained from Echonest) from the user's Youtube playlist are available in the Database.
Post conditions	1. The user can start his/her run. 2. The app chooses a sound track based on the intensity of the user's running. 3. The run information is recorded in the database.
Flow of events for Main Success Scenario	1. The user selects menu item 'Run'. 2. The app retrieves the steps per minute of the user through the Android SDK and calculates his/her speed. 3. The app matches the User's speed with a song having an equivalent beats per minute. 4. The UI displays the music controls i.e. controls for play/pause, next song and previous song. 5. The app records the user activity and stores it in the database. 6. The UI also displays a 'STOP' button to stop the run.

Table 7: Use Case 5 : Train

Use Case 5	**Functionality**
Use Case	Train
Related Req	REQ-3, REQ-4, REQ-5, REQ-7.
Initiating Actor	User
Actors Goal	To select one of the three training routines for the run.
Participating Actors	Youtube, Database
Pre conditions	1. The user is logged into the application using his/her Google account. 2. The beats per minute of songs (obtained from Echonest) from the user's Youtube playlist are available in the Database. 3. The UI displays a menu of all available functions.
Post conditions	The user can begin his Run in one of the three training modes.
Flow of events for Main Success Scenario	1. The UI displays a menu with option Train. 2. On selection of train, the UI displays three training options i.e. Beginner, Intermediate and Advanced. 3. When the user selects one of the three options, the pre-decided beats per minute for the routine is retrieved from the Database. 4. Songs are selected from the user's playlist which match the predecided beats per minute. 5. The UI displays the 'STOP' button to end the run and music controls i.e. controls for play/ pause, next song, previous song.

Table 8: Use Case 6 : Challenge

Use Case	**Functionality**
Use Case 6	Challenge
Related Req	REQ-8, REQ-9
Initiating Actor	User
Actors Goal	To compete with his/her friend who is registered on the app.
Participating Actors	Friend, Database, Android SDK
Pre conditions	1. The user completed his Run. 2. The app stored the duration and distance of the run in the database and displays it on the screen. 3. On completion of the run, the app displays an option 'Challenge Friend'.
Post conditions	On clicking 'Challenge Friend' the user is able to type in the username of the person he wishes to challenge.
Flow of events for Main Success Scenario	1. The user goes for a run. 2. The user's run duration and distance is stored in the database. 3. On completion of the run, the app displays an option to challenge friend/s. 4. The user can select friend/s by typing their username into the UI. 5. The selected friend/s is notified about the challenge.

Table 9: Use Case 7 : Respond to Challenge

Use Case	**Functionality**
Use Case 7	Respond to a Challenge
Related Req	REQ-9
Initiating Actor	User
Actors Goal	To accept or decline a challenge from a friend..
Participating Actors	Friend, Database
Pre conditions	The user has pending challenge/s.
Post conditions	1. The user has accepted/denied the challenge. 2. The friend (who initiated the challenge) is notified.
Flow of events for Main Success Scenario	1. The user sees an UI with a list of pending challenges from friends. 2. If the user accepts a challenge a)the display proceeds to the UI to start a run. b) Once the run is completed, app will also notify the user of this result and also allow the user to use the current run to challenge other friends. c) The friend (who initiated the challenge) is notified about the result. 3. If the user declines the challenge, the friend is notified.

Table 10: Use Case 8 : View Statistics

Use Case	**Functionality**
Use Case 8	View Statistics
Related Req	REQ-7, REQ-8
Initiating Actor	User
Actors Goal	To view user's past exercise activity.
Participating Actors	Database
Pre conditions	The user has completed atleast one run.
Post conditions	The UI displays visual information of the User's past running activity.
Flow of events for Main Success Scenario	1. The user selects 'View Statistics' from the menu. 2. The app aggregates past exercise data and provides an UI to display data visualizations of the data stored for the user. 3. It can show the changes in the running distances and durations over time.

4.4 System Sequence Diagrams

The Use Case Description is diagrammatically depicted in the System Sequence Diagrams. They show the step-wise implementation of the functions of the use cases.The sequence diagrams highlight the interaction between the various actors. Following are the System Sequence Diagrams which represent the Use Case Descriptions.

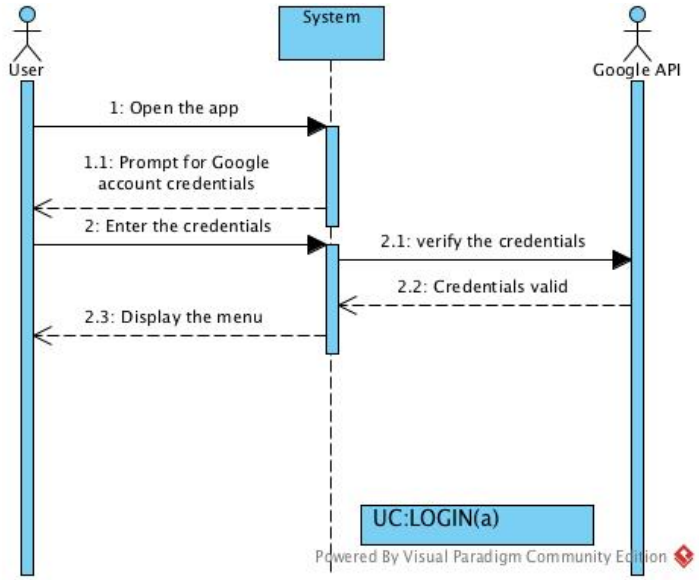

Figure 4: SD:Login(Main Scenario)

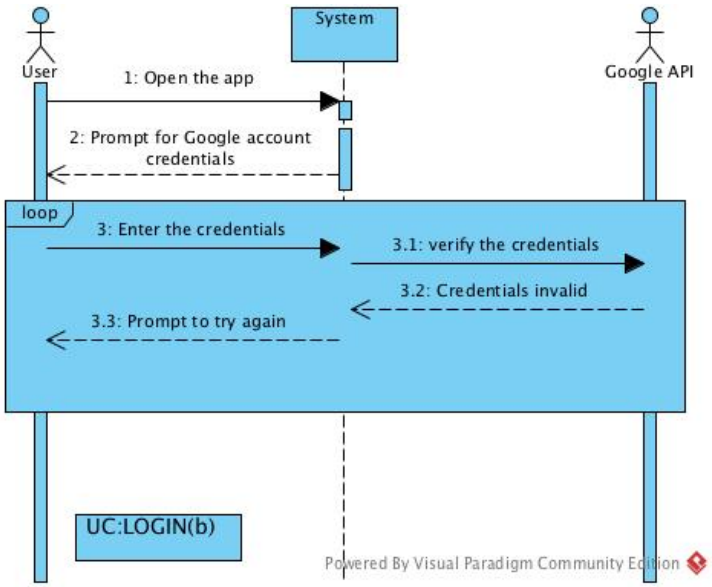

Figure 5: SD:Login(Alternate Scenario)

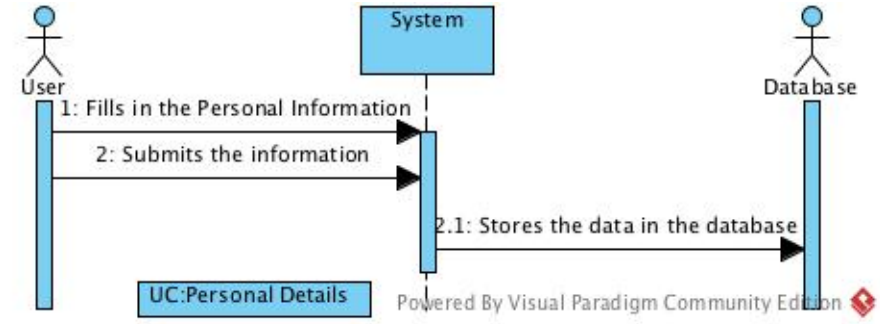

Figure 6: SD:Personal Details

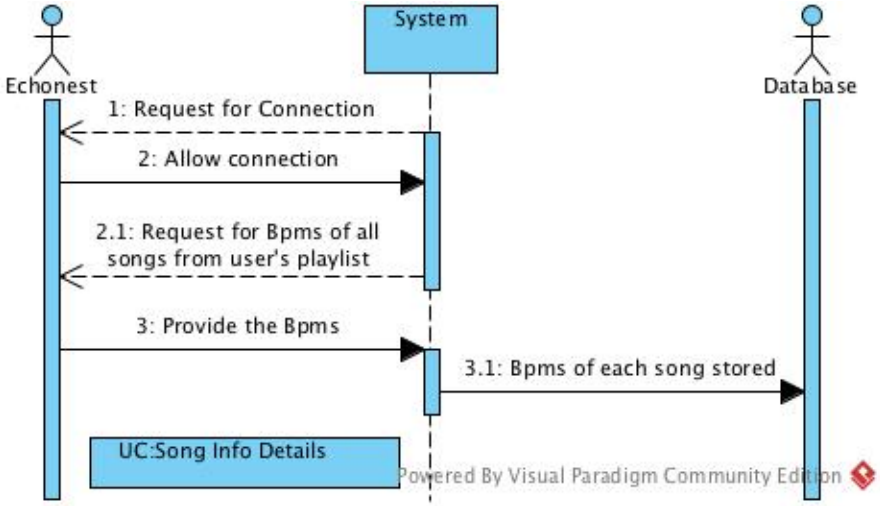

Figure 7: SD:Song Information Details

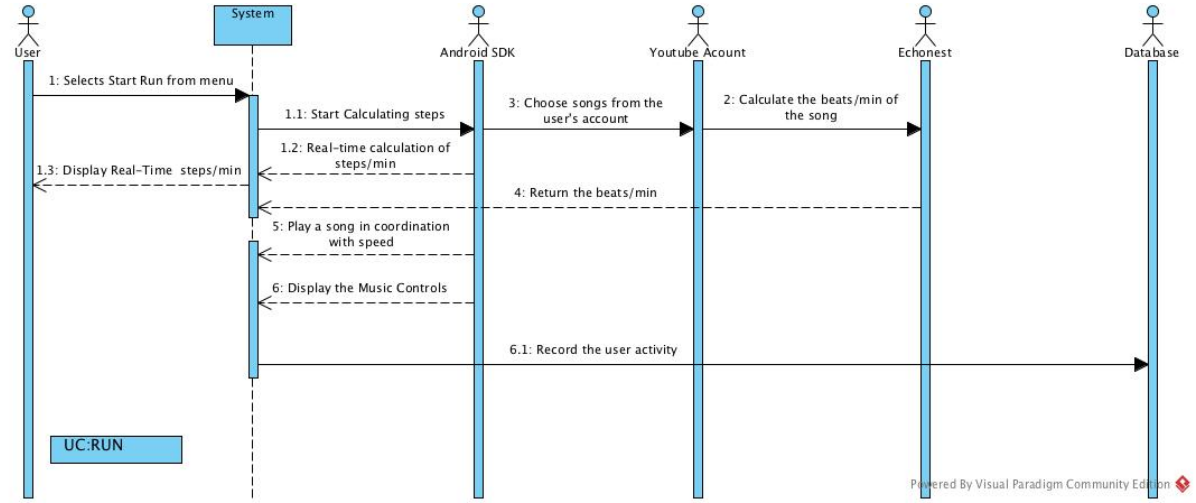

Figure 8: SD:Run

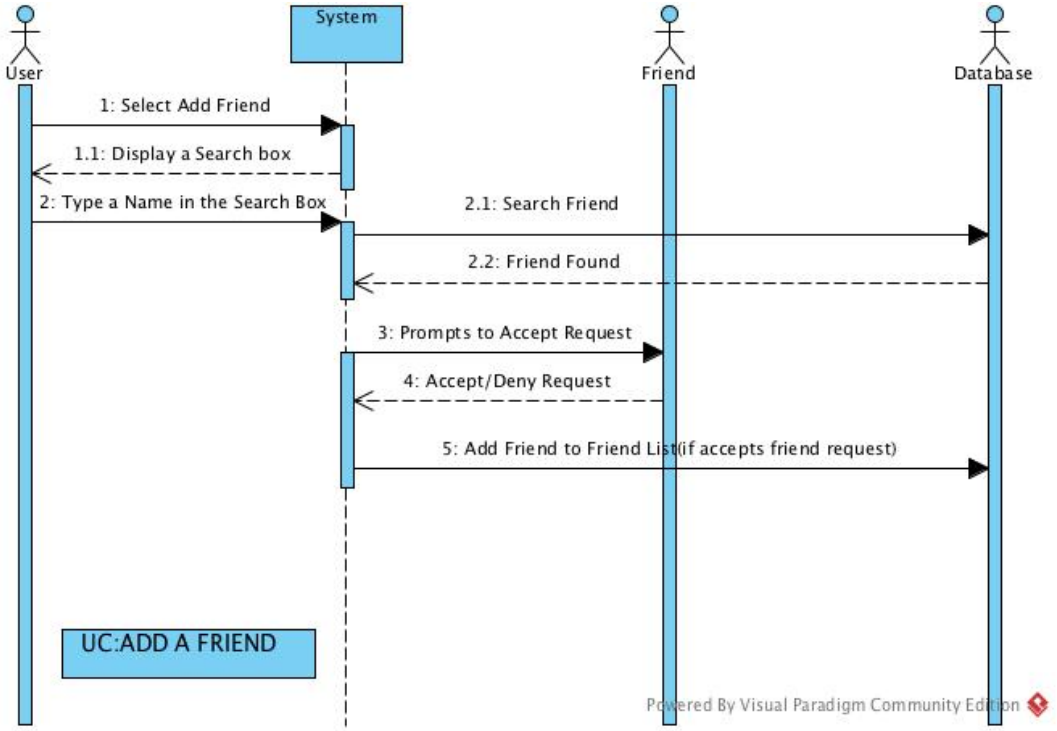

Figure 9: SD:Add a Friend(Main Scenario)

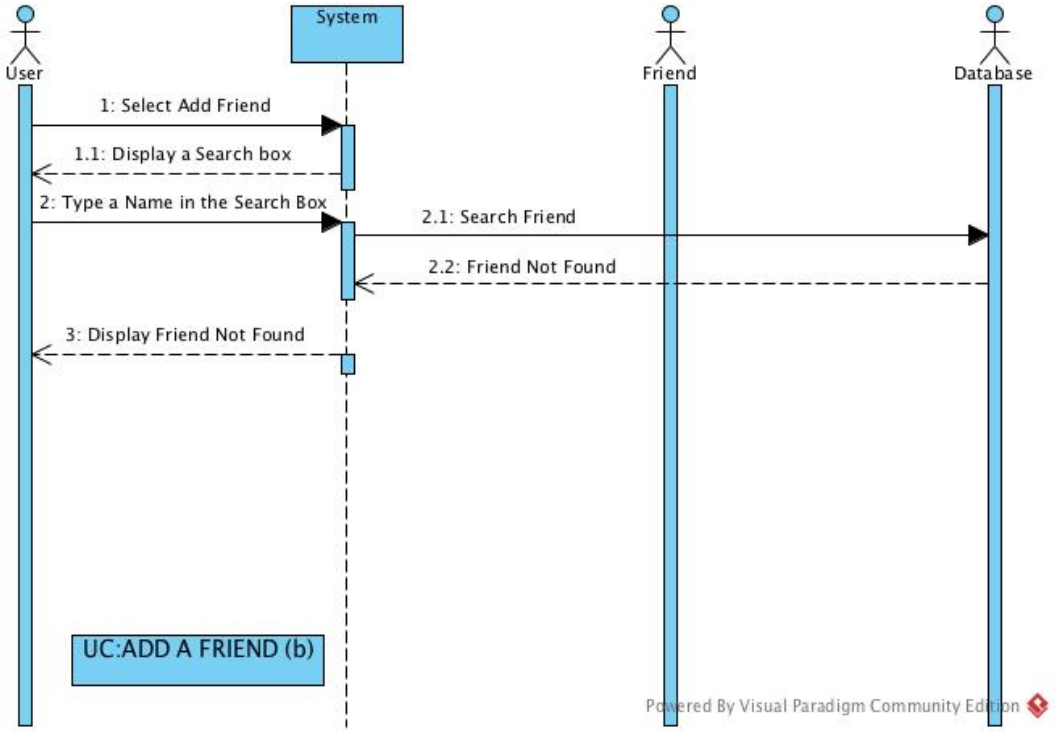

Figure 10: SD:Add a friend(Alternate Scenario)

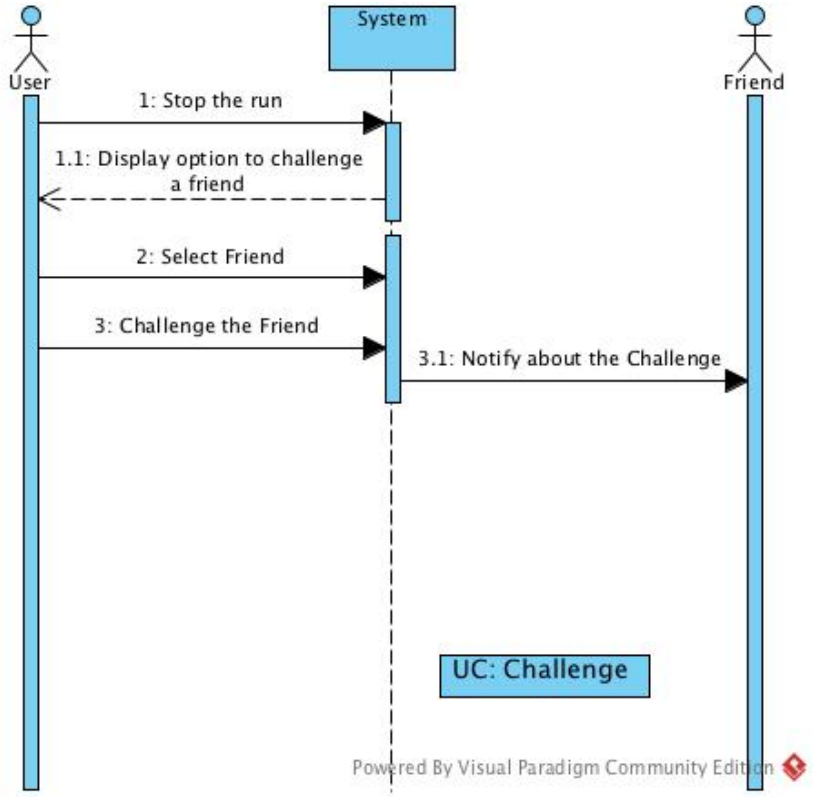

Figure 11: SD:Challenge

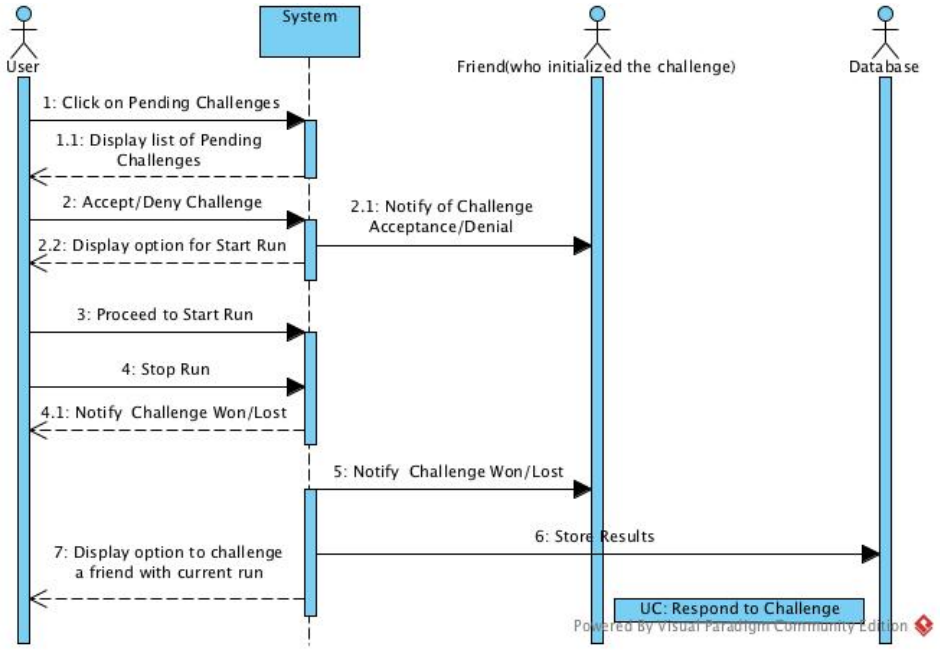

Figure 12: SD:Respond to Challenge

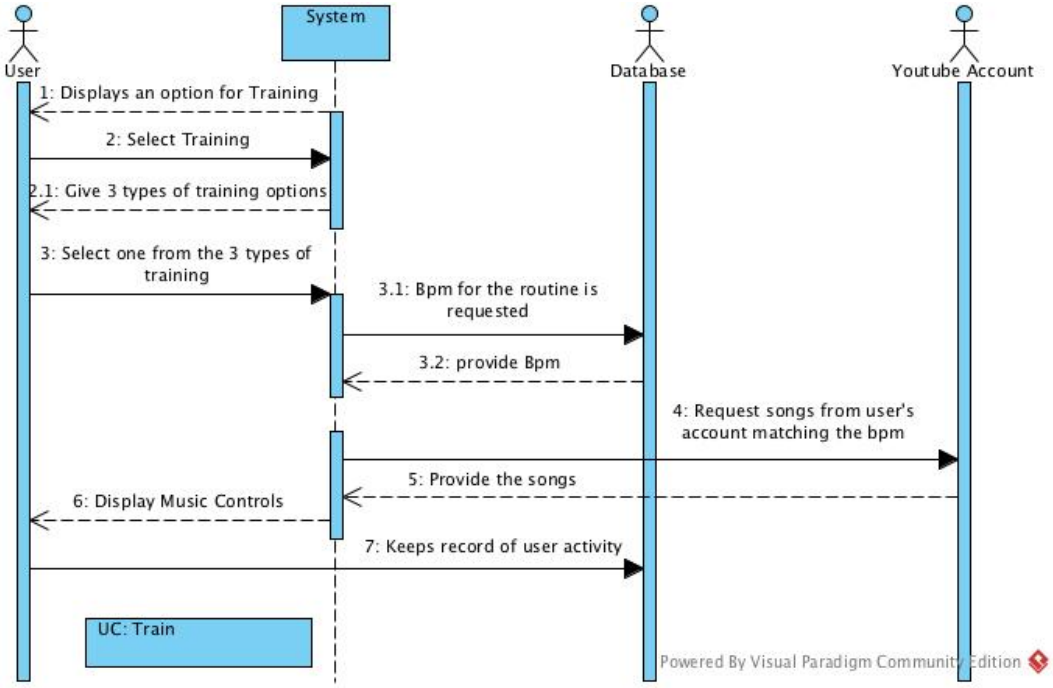

Figure 13: SD:Train

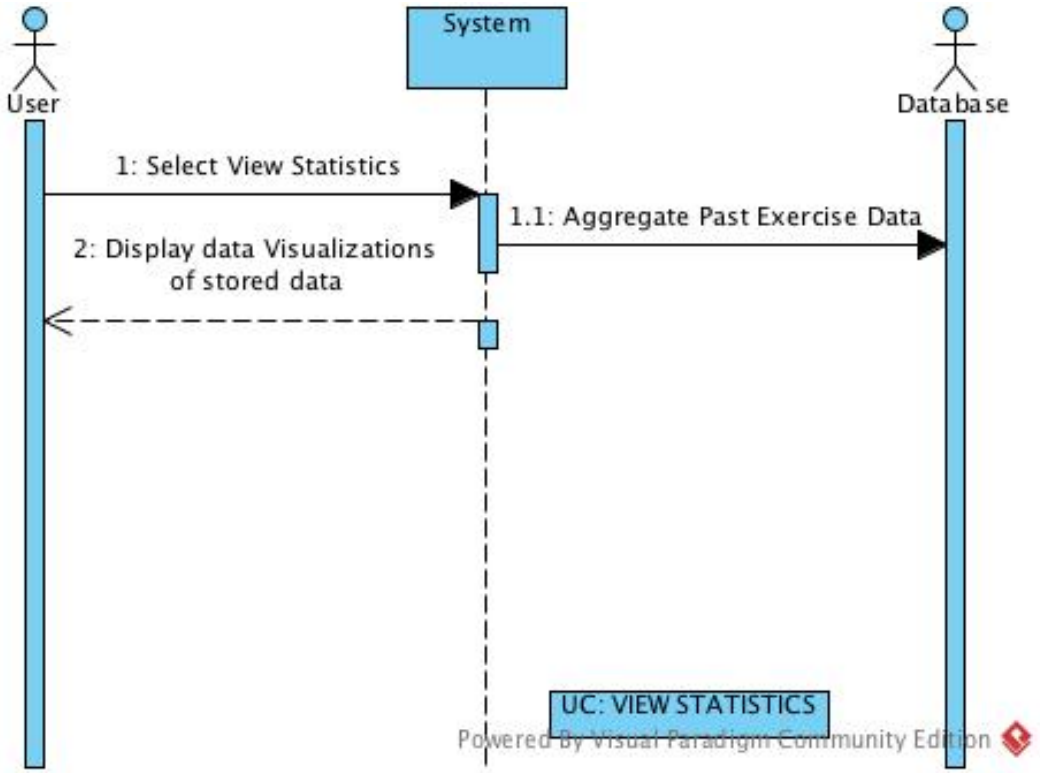

Figure 14: SD:View Statistics

5 USER INTERFACE SPECIFICATION

The User Interface is the part of the software that interacts with the user. In the instance of a mobile application, this is in the form of a Graphical User Interface based on the Android SDK. Every component of the GUI has to be designed to be intuitive for the user as well as provide ease of use.

The interaction events with the user can be the following for a mobile application:

- **Touch/Click** A screen touch can be used to register the intent of the user to click or activate the GUI component that registers the touch event. If the touch happens on a textbox, the focus of the application switches to the textbox and the contents of the textbox become editable. If the touch happens on a button, the button action is executed.
- **Gesture** Elongated touches can be interpreted as special gestures. Swiping left or right allows scrolling of horizontal content. Switing up or down allows scrolling vertically.
- **Text Input** Focusing on a textbox allows text input. Text input occurs by typing on the on-screen/physical keyboard of the phone.

Keeping the standard user interactions in mind, the UI attempts to provide maximum, intuitive functionality while minimizing user effort.

5.1 Preliminary Design

The initial design of the GUI has been demonstrated through the use of mockups, Figure 15 and 16.

- **Login:** The top-left in Figure 15 shows the interface to log into the application. The logo is shown in a central and prominent way to draw the user's experience into the app - *Beat.My.Run*. The user can log in using his *Google* credentials. The user presses the *Sign in with Google* button to be either redirected to *Google's* login interface or to be logged in automatically.
- **Personal Details:** The top-right of the Figure 16 shows the interface for the user to input account details into the application to create the user profile. This screen is shown the very first time the use logs into the app. The picture on the top left can be clicked to upload/change the current profile picture. The *Username* textbox can accept the text input of the user's name. Similarly, the *height*, *weight* and *age* can be entered

into appropriately labeled textboxes. The *location* textbox supports auto-complete to fill in possible location names in the US. By default, this field is populated by the current GPS location. The *Submit* button is pressed to confirm the details and create the account.

- **Main Menu:** The bottom-left of Figure 15 shows the main menu screen. The top of the interface has the picture of the user and the username. Either of these can be clicked to open the *Personal Details UI* Below this is shown the weekly, monthly and all-time statistics of the user. These summaries can be swiped/scrolled left or right, as indicated by the arrows on either side. The clickable *Main Menu* items can be seen below this as *Run, Training* and *Challenge. Challenge* additionally shows the number of pending challenge requests in a small bubble to the right of the text.

- **Starting a Run:** The bottom-right of the Figure 15 shows the interface to start running. The interface shows a summary of the weather information for the current location, obtained from GPS data. The graph shows a hourly forecast and a text warning below it shows precipitation warnings. The *Start* button at the button can be clicked to start the run.

- **Running and Music Playback:** The top-left of the Figure 16 shows the interface the user sees when the user is running. The top two graphics show the distance and calorie count of the current run. These along with the time statistic updates continuously. The time icon can be clicked to pause the run and the clock. The *Stop Run* button can be pressed to stop the current run. The music playback occupies the bottom of the UI. The playback progression is shown on a line. The name of the song, the artist and the album name are shown. At the bottom, the music playback buttons consist of *Previous, Play/Pause,* and *Next.* The buttons perform the standard music playback functions.

- **Run Summary:** The top-right of the Figure 16 is what the user sees once the user presses the *Stop Run* button. The summary of the distance, calories burnt, duration for the current run is shown. The *Challenge A Friend* button can be clicked to challenge a friend to run the same distance in lesser time. Once the user presses the button, the user can enter the name of the friend in a popup UI and challenge the friend. The *Run Summary* UI also shows whether the challenge was successfully completed, if the run was initiated as a part of an incoming challenge.

- **History:** The bottom-left of the Figure 16 shows the historical summary of the user's running statistics. The time period of the statistics can be selected by clicking on the

Past Week button, to change it to *Past Month* or *All Time*. Graphs can show information about the user's steps, distance, time and calories burnt.

- **Challenges:** The bottom-right of the Figure 16 shows the interface for accepting challenges and friend requests. The top graphics shows the statistics of the user's challenges as the success rate and rank among friends. The *Pending Requests* pane shows a list of challenges from friends. Every row shows the picture of the friend, challenge distance and challenge time. Incoming friend requests also come up here. This pane is scrollable by swiping up and down. Every row is clickable to go to the *Starting a Run* interface. The bottom of the UI has a textbox to add a friend by using their username.

- **Training:** The Figure **??** shows the interface for *Training*. The user can select from a set of three predefined training regimens according to intended intensity of the workout. The three categories are *Beginner, Intermediate* and *Advanced*. Every training routine consists of a preset sequence of jogging and sprinting of different durations. The UI shows the level details. Every level row is clickable and takes the user to the Starting a Run interface.

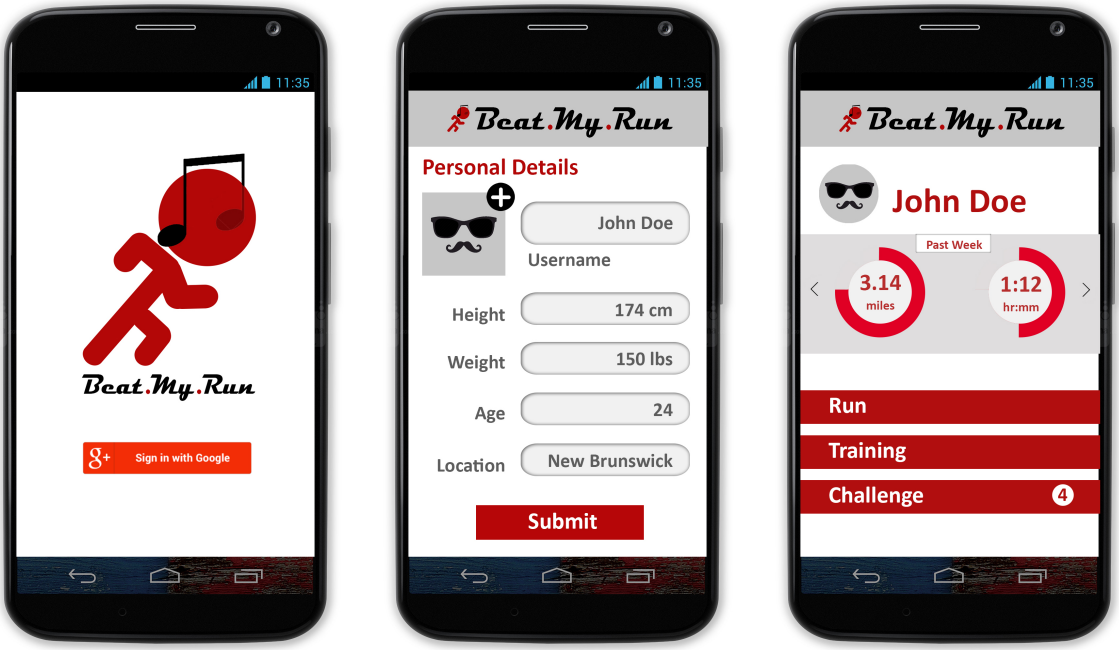

Figure 15: Mock-ups: Login, Personal Details and Main Menu

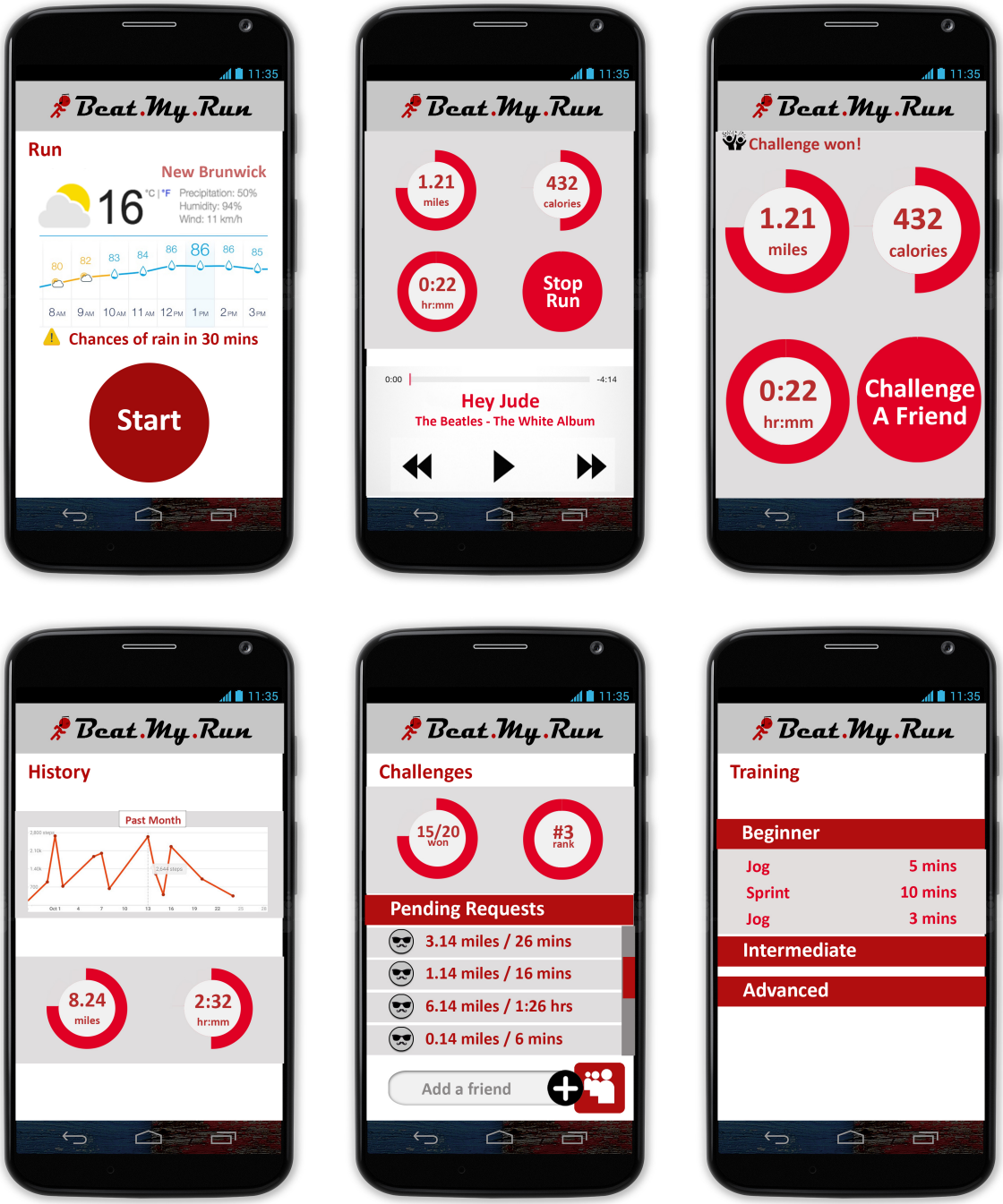

Figure 16: Mock-ups: Starting a Run, Running and Music Playback, Run Summary, History, Challenges and Training

5.2 User Effort Estimation

User-interface estimation refers to the mouse clicks or keystrokes needed to navigate through different windows of the user interface until you reach the appropriate context where you can enter the data. ("Context" roughly corresponds to the window in which the data entry will take place.)

1. **Login**
   (a) *NAVIGATION*: total 2 mouse clicks, as follows
      - Click button "Sign in with google"
      - Click button "Allow access to Beat.My.Run"
2. **Change my personal photo**
   (a) *NAVIGATION*: total 4 mouse clicks, as follows
      - Click on the name link in the main page
      - Click on the "plus" icon next on the image
      - Select a photo from your phone's library
      - Press the "select" button
3. **Fill or change my personal details**
   (a) *NAVIGATION*: total 1 mouse clicks, as follows
      - Click on the name link in the main page
        — after completing data entry as shown below —
      - Press the "submit" button to submit the changes
   (b) *DATA ENTRY*: total 1 mouse click and 11 keys (minimum)
      - Click cursor to Username field
      - Write your username (keys>1)
      - Press the "Tab" key to move to the next text field (âĂIJHeightâĂİ)
      - Write your height (4 keys)
      - Press the "Tab" key to move to the next text field (âĂIJWeightâĂİ)
      - Write your weight (3 keys)
      - Press the "Tab" key to move to the next text field (âĂIJAgeâĂİ)
      - Write your age (2 keys)

- Press the "Tab" key to move to the next text field (âĂIJLocationâĂİ)
- Write your location (keys>1)
- Press the âĂIJselectâĂİ button

4. **Run**

   (a) *NAVIGATION*: total 2 mouse clicks, as follows

   - Click "Run" in the main screen
   - Click button "Start" to start the run

5. **Stop/Resume run**

   (a) *NAVIGATION*: total 1 mouse clicks, as follows

   - Click "Stop Run"/"Resume Run" while in the running screen

6. **Start/resume song**

   (a) *NAVIGATION*: total 1 mouse clicks, as follows

   - Click the play/pause button to play/pause the song while in the running screen

7. **Go to the previous/next song**

   (a) *NAVIGATION*: total 1 mouse clicks, as follows

   - Click the previous/next button to go to the previous/next song while in the running screen

8. **Challenge a friend after finishing the run**

   (a) *NAVIGATION*: total 3 mouse clicks and 3 keystrokes as follows

   - Click the "challenge a friend" button
     — after completing data entry as shown below —
   - Select the name of your friend from the list

   (b) *DATA ENTRY*:

   - Click cursor to search field
   - Write the first three letters of the name of your friends

9. **Challenge a friend**

(a) *NAVIGATION*: total 3 mouse clicks and 3 keystrokes as follows

- Click the âĂIJchallengesâĂİ button
  — after completing data entry as shown below —
- Select the name of your friend from the list retrieved

(b) *DATA ENTRY*:

- Click cursor to search field
- Write the first three letters of the name of your friends

10. **Accept a challenge invitation**

(a) *NAVIGATION*: total 2 mouse clicks

- Click the "Challenges" button
- Select the challenge you want to accept from the list

11. **Training**

(a) *NAVIGATION*: total 3 mouse clicks

- Click the "Training" button
- Select the level of your training
- Select the "Start" button to start your run

12. **Show my history**

(a) *NAVIGATION*: total 1 mouse click

- Click the "history" link in the main screen

6 DOMAIN ANALYSIS

6.1 Domain Model

A Domain Model explain meaningful conceptual classes in a problem domain.It represents the real world concepts and not the software components. Domain model relates objects in problem domain to each other. The objects in the domain model are the candidates for programming. The domain analysis of the problem has been done with the help of

- **Concepts:** Objects in the domain
- **Attributes:** Properties of the concept objects
- **Associations:** Relationships between concept objects

The following figure illustrates the complete process of domain analysis[12] in the context of Software Engineering. The information about the domain is captured in the model and can be used and reused effectively.

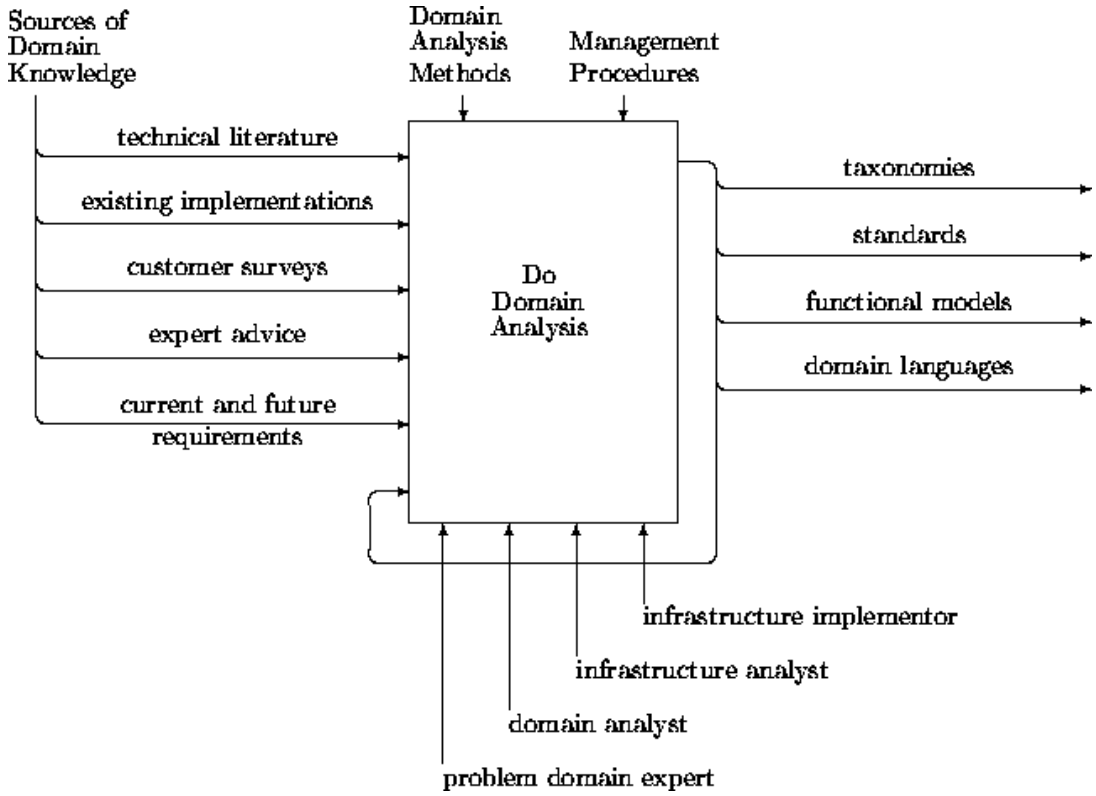

Figure 17: A model of the domain analysis process

6.1.1 Concept Definitions

As we list the Requirements, we figure out the components that work towards fulfilling the requirement. These, components are called concepts.[13]Concepts can be ideas, thing or object in the domain.A class describes a set of objects with the same semantics, properties and behavior.When used for domain modeling, it is a visualization of a real world concept.Concepts can be physical objects,roles and responsibilities of people.We derive domain model concepts and corresponding responsibilities from the formerly defined system use cases.In use case login GUIGenerator is a concept name that is responsible for generating the initial login GUI. They are in the form of noun phrases. The Tables below show the various Concepts Mapped to their Responsibilities. The Type D and K refers to 'Does and 'Knows' which means the D type Concepts are the one which perform tasks and K-type Concepts are the ones which just knows or are notified of the various tasks that happen.

Table 11: Concept Definition of Use Case:Login

Responsibility Description	**Type**	**Concept Name**
Responsible for generating the initial-login GUI	D	GUIGenerator
Detects the button selection	D	SelectionDetector
It connects to the Google API and authenticates the user	D	GoogleAuthenticator
Downloads all useful google account information (user profile details and youtube song preferences)	D	GoogleDataDownloader
It keeps all the google related data of the user (user profile details and youtube song preferences)	K	GoogleAccountKeeper
Responsible for signing out from the app	D	Disconnector

Table 12: Concept Definition of Use Case: Personal Details

Responsibility Description	**Type**	**Concept Name**
Responsible for generating the personal information GUI	D	GUIGenerator
Keeps all the personal details of the user	K	PersonalDetailsKeeper
It detects the submit button selection	D	SelectionDetector
Upload new profile picture	D	ProfilePictureUploader
It passes the user entered information to the PersonalDetailsKeeper and retrieves all the account information from the database	D	AccountController
Stores data from the PersonalDetailsKeeper to the Database	D	DatabaseManager

Table 13: Concept Definition of Use Case: Get Song Information

Responsibility Description	**Type**	**Concept Name**
Establishes the initial API connection	D	EchonestAPIConnector
It will query the EchonestAPI by using the GoogleAccountKeeper to get the song details and write the response in the SongInfoKeeper	D	Controller
Stores the Echonest responses containing the details of all the songs in the user's Youtube preferences	K	SongInfoKeeper
Stores the song information in the SongInfoKeeper in to the database.	D	DatabaseManager

Table 14: Concept Definition of Use Case: Respond to Challenge

Responsibility Description	**Type**	**Concept Name**
Displays the GUI with pending challenges	D	GUIGenerator
Detects accept or reject of the selected challenge	D	SelectionDetector
Container for the collection of open challenges for the user	K	PendingChallengeInfoKeeper
Populates the PendingChallengeInfoKeeper from the database	D	DatabaseManager
Notifies the opponent about a rejected challenge	D	ChallengeNotifier
To scroll through the list of challenges in the UI	D	ScrollDetector

Table 15: Concept Definition of Use Case: View Statistics

Responsibility Description	**Type**	**Concept Name**
Displays the GUI with statistics visualization	D	GUIGenerator
Scrolls through past week, past month and all time statistics visualizations	D	ScrollDetector
Contains the statistics of the user	K	StatisticsKeeper
Retrieve the statistics from the database and populates the StatisticsKeeper	D	DatabaseManager
Depending upon view of past week, past month and all time, it calculates the statistics to display and forwards it to the GUIGenerator to form the visualization.	D	StatisticsCalculator

Table 16: Concept Definition of Use Case: Run

Responsibility Description	**Type**	**Concept Name**
Responsible for generating GUI	D	GUIGenerator
Detects the selection of:The option Run from the menu StopPlay/pause, previous, next in the media player	D	SelectionDetector
Responds to the Selection of Start/Pause/Resume RunStop	D	RunController
Accesses sensors through the AndroidSDK to calculate speed of the user during the run and release access to the sensors on completion of the run.	D	AccelerometerController
Stores the distance run, time and calories burnt for the current run.	K	RunInfoKeeper
Calculates the values in the RunInfoKeeper	D	RunInfoCalculator
Store data gathered after the run by the RunInfoKeeper in the Database and retrieve beats per minutes of songs from Database	D	DatabaseManager
Stores the information about songs in the user preferences from the database. It contains the Song name, artist and other details, as well as BPM of the song.	K	SongInfoKeeper
It selects the proper song by comparing the bpm data of all the songs kept by the SongInfoKeeper and the bpm data of the user held by the RunInfoKeeper. It forwards the selected song to the MusicController.	D	SongSelector
Plays/pauses or goes to next/ previous song during music playback	D	MusicController

Table 17: Concept Definition of Use Case: Train

Responsibility Description	**Type**	ConceptName
Responsible for generating menu GUI	D	GUIGenerator
Detect the selection of the option Train from the menu	D	SelectionDetector
SelectionDetector invokes the TrainingManager of the training level and initializes SongSelector from the run use case and starts the run use case.	D	TrainingManager

Table 18: Concept Definition of Use Case: Send a Challenge

Responsibility Description	**Type**	**Concept Name**
Detects the selection of Challenge option	D	SelectionDetector
Displays a GUI with option to enter friend username	D	GUIGenerator
Searches for the opponent with the username within the Database	D	FriendFinder
It keeps the list of all the user information data to store results from the FriendFinder	K	UserInfoKeeper
Aggregates the RunInfoKeeper, the current user's username and the opponent's username.	K	SendChallengeInfoKeeper
Sends a request to the DatabaseManager to store the SendChallengeInfoKeeper	D	SendChallengeController
Stores the ChallengeInfoKeeper into the Database	D	DatabaseManager
If the run was initiated as a challenge, check outcome from the RunInfoKeeper and invoke ChallengeNotifier	D	ChallengeOutcomeDecider
Notify friend about challenge thrown or the challenge result	D	ChallengeNotifier

6.1.2 Attribute Definitions

An attribute can be defined as description of a named slot of a specified type in a domain class.Each instance of the class separately holds a value.Once the concept definitions are made, each concept performs its Responsibility using various attributes.Its through these various attributes each concept is implemented.Attributes are the required quantities for the concepts to perform their functions.Attributes can be a button or an entry field depending on the concept. Following tables show the Attributes of all the Concepts created previously along with the Attribute Definitions for each use case.

Table 19: Attribute Definition for Use Case : Login

Concept	**Attributes**	**Attribute Description**
GUIGenerator	button	To go to the login credentials
GoogleAuthenticator	username	Google account username
	password	Google account password
GoogleAccountKeeper	email	Email id associated with google account
	list of songs	List of songs from Youtube history

Table 20: Attribute Definition for Use Case : Personal Details

Concept	**Attributes**	**Attribute Description**
GUIGenerator	entry fields	To obtain user information
	profile picture	To obtain profile picture
PersonalDetailsKeeper	user's identity	The user's name, age, height, weight, location and picture
ProfilePictureUploader	profile picture	Needed to upload profile picture

Table 21: Attribute Definition for Use Case : Song Information

Concept	**Attributes**	**Attribute Description**
SongInfoKeeper	listOfSongs	User's Youtube song names
	songAttributes	bpm, duration, tempo, genre

Table 22: Attribute Definition for Use Case: Run

Concept	**Attribute Name**	**Attribute Description**
GUIGenerator	start run button	Starts the run
	pause run button	Pause the run
	music player	Displays music information
	statistics	Running metrics
RunInfoKeeper	run-related data	The user's distance run, time and calories burnt for the current run.
SongInfoKeeper	listOfSongs	User's Youtube song names
SongSelector	selectedSong	The selected song based on the user's tempo
	training level	If in training mode, select bpm sequence

Table 23: Attribute Definition for Use Case : Train

Concept	**Attribute Name**	**Attribute Description**
GUIGenerator	training level buttons	UI for three training level selection
TrainingManager	selected training level	user selected training level

Table 24: Attribute Definition for Use Case : Send a Challenge

Concept	**Attribute Name**	**Attribute Description**
GUIGenerator	Challenge button	Displays challenge button
FriendFinder	search text	Search text used for database query
UserInfoKeeper	list of usernames	List of usernames matching search text
SendChallengeInfoKeeper	current user name	Username of the challenge sender
	opponent username	Username of selected opponent
	challenge details	Distance and time of current run

Table 25: Attribute Definition for Use Case : Respond to Challenges

Concept	**Attribute Name**	**Attribute Description**
GUIGenerator	List of open challenges UI	Scrolling view UI of open challenges
	Accept/Reject icons	Accept and reject options for every open challenge
PendingChallengeInfoKeeper	List of SendChallenge-InfoKeepers	List of open challenges

Table 26: Attribute Definition for Use Case : View Statistics

Concept	**Attribute Name**	**Attribute Description**
GUIGenerator	graphs	Scrollable visualizations for the statistics data
StatisticsKeeper	exercise history information	distance, time and calories burnt per day
StatisticsCalculator	period of time	The period for calculating the statistics, ie. past week, past month or all time.

6.1.3 Association Definitions

Associations describe the relationship between classes.It specify the meaningful and interesting connection between them.There can be more than one association between concepts and it can be unidirectional and bidirectional.It is a link between concept classes which are the fundamental building block for describing relationships in domain model.Associations are created when a concept need to know about other.It can be a input from other concept or even an invoke from a concept.So first all the concept pairs are identified,then the relation between them are noted down.In usecase login SelectionDectector should send request to GoogleAuthenticator and it should authenticate. So we group them together and the Association here is send request.Associations are shown in domain model as a line that connects two concepts with an arrow to specify the direction and the association name.The association definitions ,name and the corresponding concept pairs for each use case is listed below.

Table 27: Association definitions of Use Case: Respond to Challenge

Concept pair	**Association Description**	**Association Name**
GUIGenerator ↔ PendingChallengeInfoKeeper	The GUIGenerator displays all the open challenges taken from the ChallengeInfoKeeper	displays
DatabaseManager ↔ PendingChallengeInfoKeeper	The DatabaseManager populates the PendingChallengeInfoKeeper from the database	populates
SelectionDetector ↔ ChallengeNotifier	The SelectionDetector informs the ChallengeNotifier about what the user has selected (accept/reject)	informs
ScrollDetector ↔ GUIGenerator	The ScrollDetector informs the GUIGenerator to display the rest of the data	informs

Table 28: Association definitions of Use Case: Login

Concept pair	**Association Description**	**Association Name**
SelectionDetector↔GoogleAuthenticator	Send Authentication request and Authenticate	Send request
GoogleDataDownloader ↔ GoogleAccountKeeper	The GoogleDataDownloader populates the data in the GoogleAccountKeeper	Populate the data
GUIGenerator↔Disconnector	The Disconnector has to inform the GUIGenerator that the user signed out so it changes the UI	Inform

Table 29: Association definitions of Use Case: Get Song Information

Concept pair	**Association Description**	**Association Name**
Controller ↔ EchonestAPIConnector	Controller passes the song details from the GoogleAccountKeeper to the EchonestAPI and stores the responses in the SongInfoKeeper	query song info
Controller ↔ DatabaseManager	Store the SongInfoKeeper in the Database	store data

Table 30: Association definitions of Use Case: Train

Concept pair	**Association Description**	**Association Name**
SelectionDetector ↔ TrainingManager	Communicates the training level	invokes
TrainingManager ↔ SongSelector	Calculates bpm sequences for the training and affects song selection logic	provides bpm sequence
TrainingManager ↔ GUIGenerator	Changes to Run GUI	updates

Table 31: Association definitions of Use Case: Run

Concept pair	**Association Description**	**Association Name**
SelectionDetector ↔ MusicController	Starts or stops the Music on selection	plays/stops
SelectionDetector ↔ RunController	Starts or stops run according to selection	starts/stops
RunController ↔ RunInfoCalculator	Calculates the Bpm once run starts	sends run info
RunInfoCalculator ↔ AccelerometerController	Gets the number of steps	gets user steps
RunInfoCalculator ↔ RunInfoKeeper	Calculates the statistics and stores it	calculates
SongInfoKeeper ↔ SongSelector	SongInfoKeeper informs SongSelector about the current user song characteristics	informs
SongSelector ↔ RunInfoKeeper	SongSelector reads the running information	reads
RunController ↔ DatabaseManager	Asks the DatabaseManager to store RunInfoKeeper	ask
SongSelector ↔ MusicController	Selects the next song to play	selects song
RunController ↔ GUIController	Changes the GUI at end of run	changes UI
MusicController ↔ GUIController	Changes the GUI to update music information and control buttons	changes UI
RunInfoKeeper ↔ GUIController	Changes the GUI to update instantaneous statistics	changes UI

Table 32: Association definitions of Use Case: Send Challenge

Concept pair	**Association Description**	**Association Name**
SelectionDetector ↔ GUIGenerator	Presents the challenge option and textbox to search for friends	accept challenge and search friend
SelectionDetector ↔ FriendFinder	Senses the search text for the name of a friend to challenge	find friend
SelectionDetector ↔ SendChallengeController	Sends request to sendChallengeInfoKeeper to store the username of user and opponent along with RunInfoKeeper information	initiate new challenge
FriendFinder ↔ DatabaseManager	FriendFinder requests DataManager to find the username entered into the textbox, within the Database	requests
FriendFinder ↔ UserInfoKeeper	Updates the results of the database query of list of usernames in the UserInfoKeeper	updates
FriendFinder ↔ GUIGenerator	Show the list of usernames matching search text	changes
SendChallengeController ↔ DatabaseManager	Stores the new challenge entry with SendChallengeInfoKeeper	store
ChallengeOutcomeDecider ↔ DatabaseManager	Stores the result of the challenge in database	stores result
SendChallengeController ↔ ChallengeNotifier	Sends notification about new challenge	notify
ChallengeOutcomeDecider ↔ ChallengeNotifier	Sends notification about challenge outcome	notify

Table 33: Association definitions of Use Case: Show/Edit Personal Information

Concept pair	**Association Description**	**Association Name**
SelectionDetector ↔ AccountController	SelectionDetector invokes accountController	invokes
AccountController ↔ DatabaseManager	Passes the PersonalDetailsKeeper to the database	passes data
ProfilePictureUploader ↔ PersonalDetailsKeeper	Passes the profile picture to the PersonalDetailsKeeper	passes picture
PersonalDetailsKeeper ↔ GUIGenerator	Passes the profile picture from PersonalDetailsKeeper to the GUIGenerator	passes picture
SelectionDetector ↔ GUIGenerator	changes the GUI	change UI

Table 34: Association definitions of Use Case: View Statistics

Concept pair	**Association Description**	**Association Name**
ScrollDetector ↔ GUIGenerator	The ScrollDetector informs the GUIGenerator to display the rest of the statistics	scroll up/down
StatisticsCalculator ↔ StatisticKeeper	Calculates the statistics for the given period	calculates
DatabaseManager ↔ StatisticsKeeper	The DatabaseManager populates the StatisticsKeeper	populates
StatisticsCalculator ↔ GUIGenerator	Forwards the data for the generation of visualizations	forwards data

6.1.4 Traceability Matrix

Traceability Matrix is a table that shows the relationship between two elements.Here the usecases are mapped to domain concepts.Priority weight is given to each use case and we identify the concepts that are related to that usecase.The matrix helps to identify wheather all the concepts are covered for a usecase.The priority weight assigned can be referenced during implementation for how work should be planned. The below table show the traceability matrix for concepts and usecase.

Use cases	PW	GUIGenerator	SelectionDetector	GoogleAuthenticator	GoogleDataDownloader	GoogleAccountKeeper	Disconnector	PersonalDetailsKeeper	ProfilePictureUploader	AccountController	DatabaseManager	EchonestAPIConnector	Controller	SongInfoKeeper	RunController	AccelerometerController	RunInfoKeeper
Login	5	✗	✗	✗	✗	✗	✗										
Personal Details	4	✗	✗					✗	✗	✗	✗						
Song detail info	5										✗	✗	✗	✗			
Run	5	✗	✗								✗				✗	✗	✗
Train	3	✗	✗														
Send Challenge	2	✗	✗								✗						
Challenge Respond	2	✗	✗								✗						
View Statistics	1	✗									✗						

Use cases	PW	RunInfoCalculator	SongInfoKeeper	SongSelector	MusicController	TrainingManager	FriendFinder	UserInfoKeeper	SendChallengeInfoKeeper	SendChallengeController	ChallengeOutcomeDecider	ChallengeNotifier	PendingChallengeInfoKeeper	ScrollDetector	StatisticsKeeper	StatisticsCalculator
Login	5															
Personal Details	4															
Song detail info	5															
Run	5	✗	✗	✗	✗											
Train	3					✗										
Send Challenge	2						✗	✗	✗	✗	✗	✗				
Challenge Respond	2											✗	✗	✗		
View Statistics	1													✗	✗	✗

Table 35: Traceability Matrix for Concepts versus Usecases

6.1.5 Domain Model Diagrams

A Domain model is constructed from a set of abstractions pertaining to the general context and knowledge base of the application. It is a representation of the solution in terms of the units defined in the the application domain. Domain model is created from the defined Concepts, Associations and Attributes.

First we identify the concepts in the domain. Concepts names are usually nouns and are written in the top compartment of the class box. Then associations to define relationship between concepts class is identified. Associations are shown as lines with arrows specifying the direction of flow between the box. Attributes necessary for particular concepts is preserved.Attributes are shown in second compartment of the class box. The domain model will not include any software.

The graphical representation of a domain model allows the visualization of the relationships between the different concepts and actors in the domain. The domain model for each usecase is shown below.

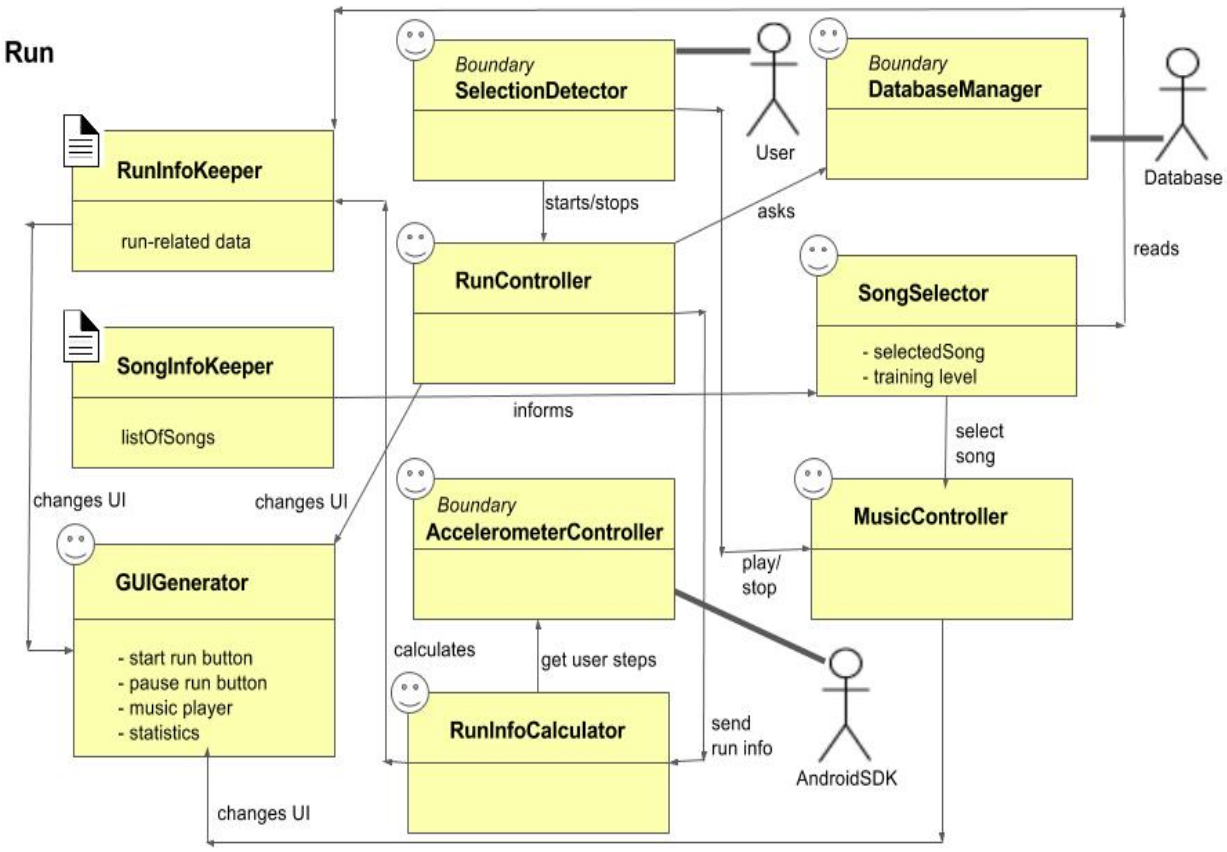

Figure 18: Domain Model:Run

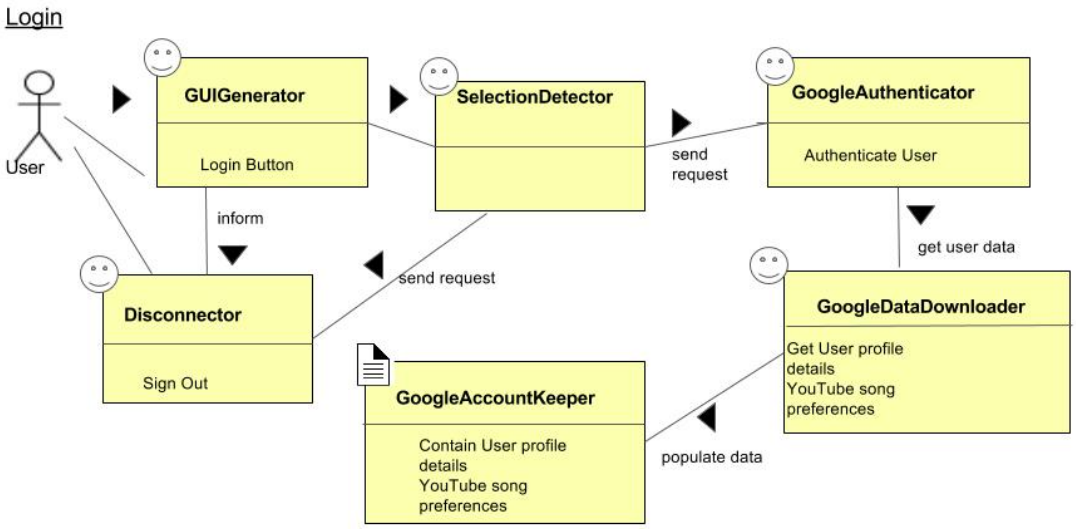

Figure 19: Domain Model:Login

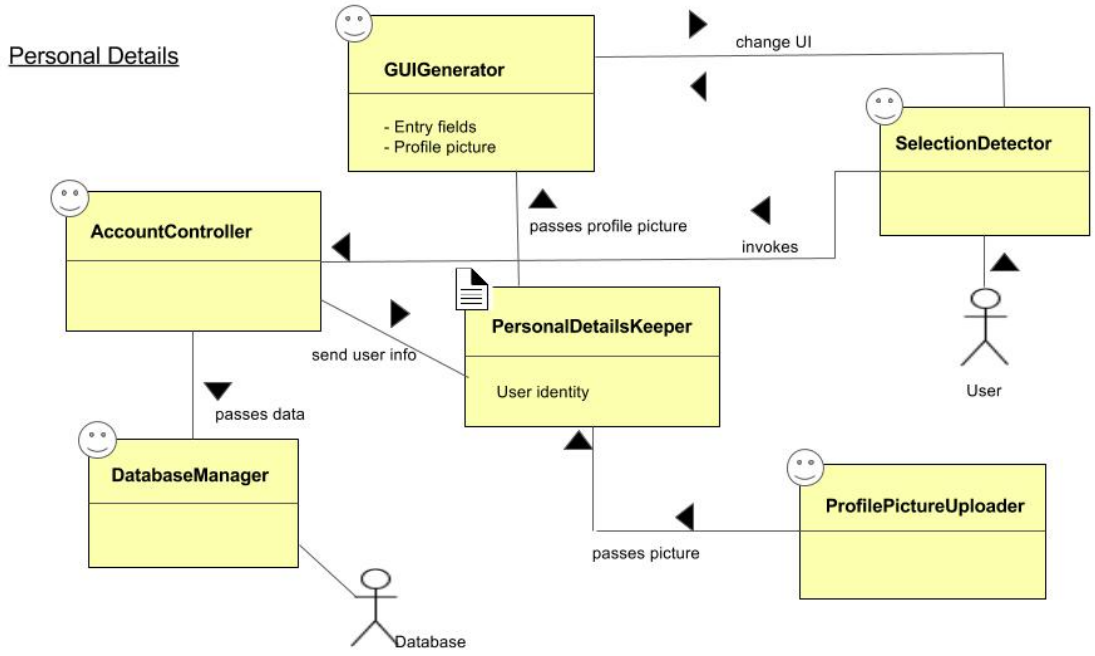

Figure 20: Domain Model:Personal Details

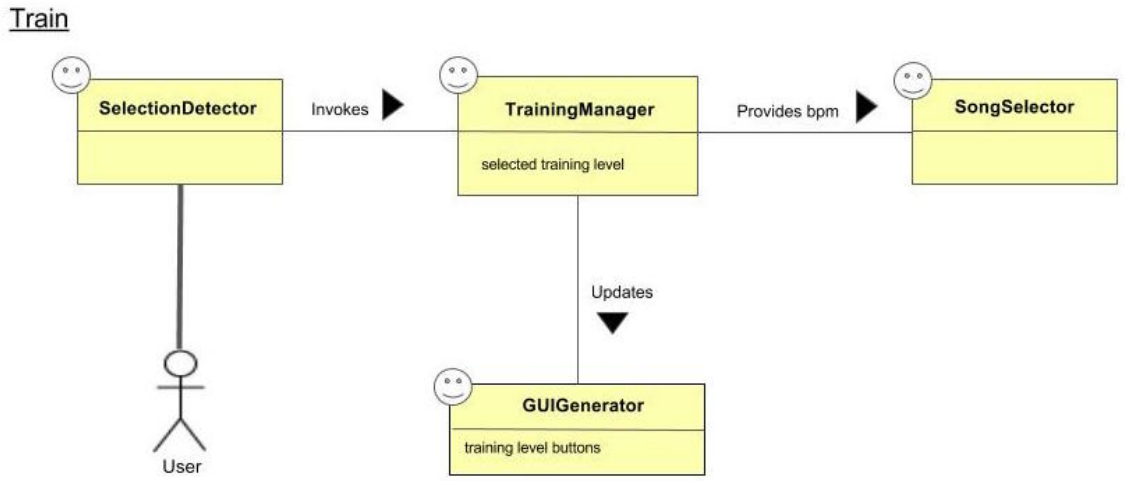

Figure 21: Domain Model:Train

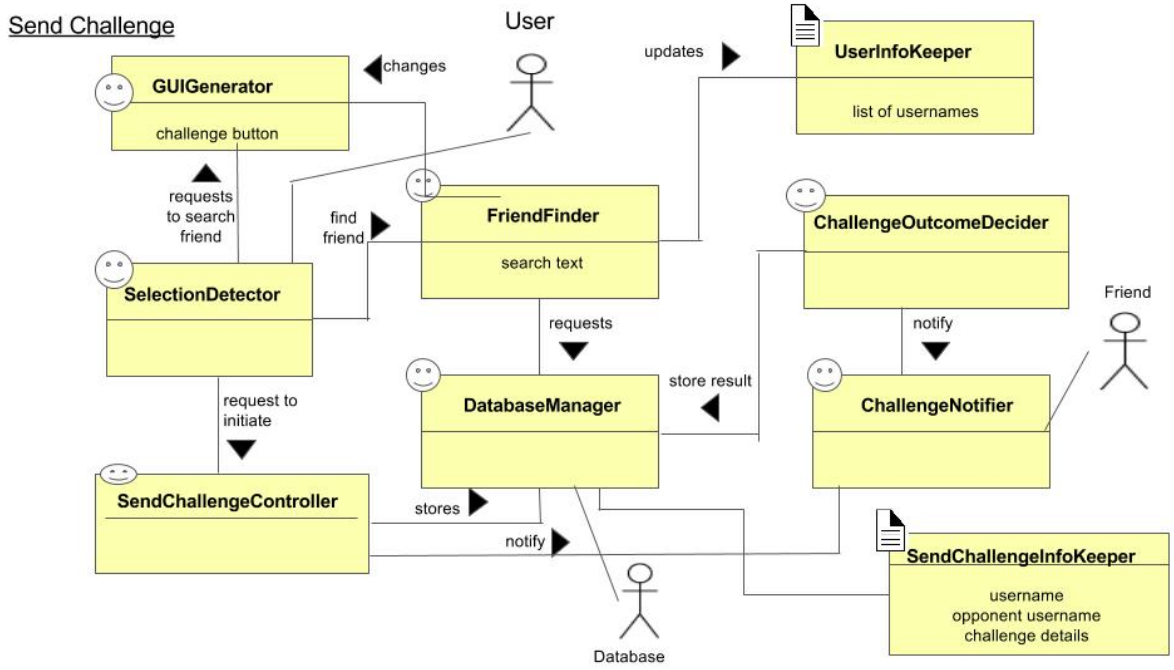

Figure 22: Domain Model:Send Challenge

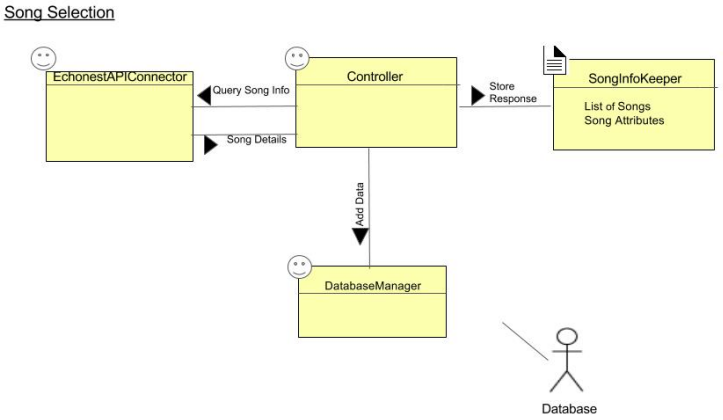

Figure 23: Domain Models:Song Selection

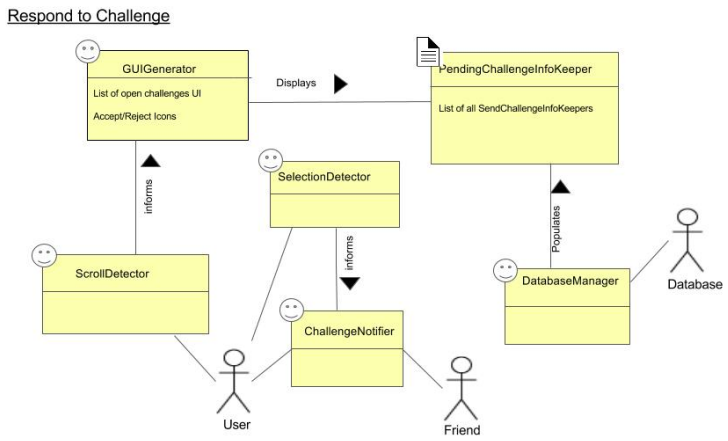

Figure 24: Domain Model:Respond to Challenge

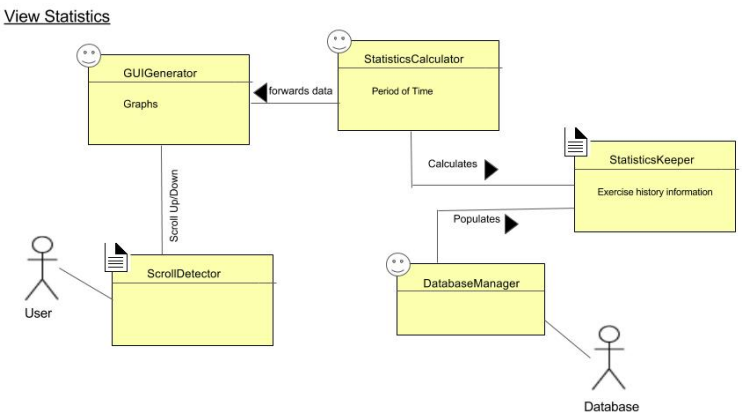

Figure 25: Domain Model:View Statistics

6.2 System Operation Contracts

Operations contracts specify any important conditions about the attributes in the domain model.It specify preconditions and post-conditions for each attributes.While making a operation contract we need to think about the state before the action and after the action.Preconditions can be enter values to a field or an association is formed or closed.Post-condition can be change in state or database updated.Thus post-condition specifies what the system will do for that attribute when all the preconditions are performed. Thus contracts helps to model the behavior of a system.The tables below specify the system operation contract of attributes for each use case.

Table 36: System Operation Contracts for Use Case : Login

Operation	**Login**
Preconditions	button='pressed', username not empty, password not empty
Postconditions	email = 'email of the user' and listOfSongs not empty

Table 37: System Operation Contracts for Use Case : Personal Details

Operation	**Show/Edit personal information**
Preconditions	User's name, age, height, weight and location not empty
Postconditions	Database updated with user details

Table 38: System Operation Contracts for Use Case : Song Information Details

Operation	**Get all the song detail information**
Preconditions	listOfSongs is not empty
Postconditions	songAttributes is not empty

Table 39: System Operation Contracts for Use Case : Run

Operation	**Run**
Preconditions	run button='pressed' and listOfSongs is not empty
Postconditions	run-related data is not empty, selectedSong is not empty and it is playing

Table 40: System Operation Contracts for Use Case : Train

Operation	**Train**
Preconditions	one of the training buttons='pressed'
Postconditions	selected training level is not empty and SongSelector training level = selected training level

Table 41: System Operation Contracts for Use Case : Send Challenge

Operation	**Send Challenge**
Preconditions	SendChallengeInfoKeeper is not empty
Postconditions	Database is updated with new SendChallengeInfoKeeper entry and opponent is notified.

Table 42: System Operation Contracts for Use Case : Respond to a Challenge

Operation	**Respond to a Challenge**
Preconditions	PendingChallengeInfoKeeper is not empty
Postconditions	Run Operation executed or the selected challenge entry is deleted from open challenges database

Table 43: System Operation Contracts for Use Case : View Statistics

Operation	**View Statistics**
Preconditions	exercise history is not empty and period of time is specified
Postconditions	visualization updated with calculated subset of exercise history

6.3 Mathematical Model

The project has many different components that require devoted mathematical models to be implemented. From the characteristics that can be used to analyze running or walking, we choose acceleration as the relevant parameter.The first step in order to find the acceleration of the user is to compute the number of steps he is doing while walking or running. We don't intend to acquire accelerometer data from the user's phone in crude form but rather use an already implemented android software that calculates the number of steps of the user. Thus, after computing the steps parameter, we will use the following algorithm [14] in order to get to get the distance parameter, the speed parameter and the calories parameter.

Distance parameter
The distance parameter which is actually the distance travelled by the user is calculated by the following formula.

$$\text{Distance} = \text{number of steps} \times \text{distance per step} \tag{1}$$

The Distance per step depends on the speed and the height of user. The step length would be longer if the user is taller or running at higher speed. Our system updates the distance, speed, and calories parameter every two seconds. We use the steps counted in every two seconds to judge the current stride length. The following table shows the experimental data used to judge the current stride.

Table 44: Stride as a Function of Speed (steps per 2 s) and Height

Steps per 2 s	**Stride (m/s)**
0 2	Height/5
2 3	Height/4
3 4	Height/3
4 5	Height/2
5 6	Height/1.2
6 8	Height
>=8	1.2xHeight

Speed parameter

As we know the speed can be calculated by:

$$\text{Speed} = \text{distance}/\text{time} \tag{2}$$

so in order to get the speed parameter, as steps per 2 s and stride we will use the following:

$$\text{Speed} = \text{steps per 2 s} \times \text{stride}/2\ \text{s} \tag{3}$$

Calories parameter

There is no accurate means for calculating the rate of expending calories. Some factors that determine it include body weight, intensity of workout, conditioning level, and metabolism. We can estimate it using a conventional approximation, however. Table 3 shows a typical relationship between calorie expenditure and running speed.

Table 45: Calories Expended vs. Running Speed

Running Speed(km/h)	**Calories Expended (C/kg/h)**
8	10
12	15
16	20
20	25

From this table, we get:

$$\text{Calories (C/kg/h)} = 1.25 \times \text{running speed (km/h)} \tag{4}$$

However because we want m\s the equation becomes:

$$\text{Calories (C/kg/h)} = 1.25 \times \text{speed (m/s)} \times 3600/1000 = 4.5 \times \text{speed (m/s)} \tag{5}$$

The calories parameter would be updated every 2 s with the distance and speed parameters. So, to account for a given athlete's weight, we can convert our last equation as following: Weight (kg) is a user input, and one hour is equal to 1800 2-second intervals.

$$\text{Calories (C/2 s) } 4.5 \times \text{speed} \times \text{weight}/1800 = \text{speed} \times \text{weight}/400 \tag{6}$$

Now if the user takes a break in place after walking or running, there would be no change in steps and distance, speed should be zero, then the calories expended can use Equation the following equation since the caloric expenditure is around 1 C/kg/hour while resting.

$$\text{Calories (C/2 s)} = 1 \times \text{weight}/1800 \tag{7}$$

Finally, we get the total calories by adding the calories for all 2-second intervals.

The selection of which track to play requires a mathematical model as well. This consists of selecting the track with the closest BPM, that is to say minimizing the difference in BPM:

$$\min\left(|target_{BPM} - track_{BPM}|\right) \tag{8}$$

If time permits, this simple model can be replaced with a more complex model incorporating Machine Learning to learn which tracks are more effective than others at changing pulse.

7 PROJECT MANAGEMENT

This project has three major subsystems. Some of them would be developed in parallel by dividing the whole team for this project to three development teams. For each subsystem (team) the plan is to use an agile development method. The current section introduces the most important and obvious parts of those subsystems. The group will start to develop it in parallel with the fundamental tasks such as finalizing specification phase or introducing system architecture phase.

7.1 Product Ownership

- Nivetha and Thara will be responsible for the View Statistics,Personal Details and Login Use Cases
- Valia and Rahul will work on Run and Train Use Cases
- Careena and Nirali will work on Send Challenge,Song Details and Respond to Challenge Use cases
- These three teams will work on their assigned use cases in parallel. We use agile developing method in each team's plan of work separately. There are 3-4 iterations per subsystems in the Table 1 that need to be done for each team. The first ones are the

Table 46: Task Breakdown based on subsystems

Subsystems	**Code**	**Task**
Mobile Sensing	T01	User input specification in the application
	T02	Mobile sensors readings (accelerometer, GPS)
	T03	Server communication to send logs
User Interface	T04	Graphical user interface design
	T05	User account profiles and information management
	T06	Data visualization from user information.
Infrastructure	T07	Application/server communications protocol specification
	T09	Specifying the information the system needs to maintain
	T10	Database creation to maintain all the logs
	T11	External API integration(*google, accuweather, echonest*)

ones that all teams need to start to develop from beginning. All the specification tasks will be confirmed with all the members to finalize.

The following subsection introduces all the deadlines we will meet for this project. These deadlines are for project deliveries such as reports, presentations, etc. But, as mentioned before, the development phase will be held in parallel with all these due to the lack of sufficient time.

Table 47: Project's deadlines

Milestones	Description	Planned Date
M0	Project Proposal	Sep. 25, 2015
	Finalize the project scope after getting feedback	Oct. 3, 2015
M1	First Report	Oct. 16, 2015
	Statement of work and requirements	Oct. 06, 2015
	Functional requirements Spec and user interface	Oct. 11, 2015
	Full report submission	Oct. 16, 2015
M2	Second Report	Nov. 9, 2015
	Interaction Diagrams	Oct. 23, 2015
	Class Diagram and System Architecture	Nov. 5, 2015
	Full report submission	Nov. 9, 2015
M3	Third report	Dec. 10, 2015
M4	First Demo	Oct. 31, 2015
M5	Second Demo	Dec. 7, 2015
M6	Electronic Project Archive	Dec. 12, 2015

7.2 Gantt Chart

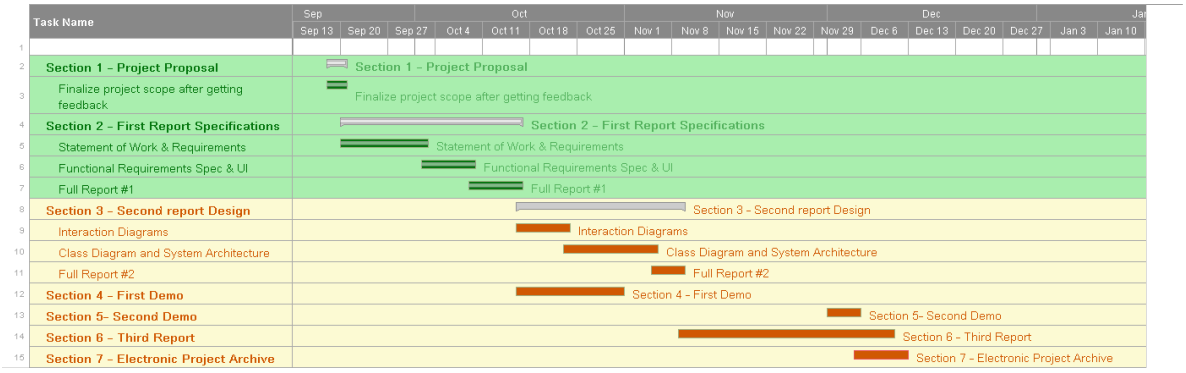

Figure 26: Gantt Chart

REFERENCES

[1] I. Holme and S. Anderssen, "Increases in physical activity is as important as smoking cessation for reduction in total mortality in elderly men: 12 years of follow-up of the oslo ii study," *British journal of sports medicine*, vol. 49, no. 11, pp. 743–748, 2015.

[2] S. H. Boutcher and M. Trenske, "The effects of sensory deprivation and music on perceived exertion and affect during exercise," *Journal of sport and exercise psychology*, vol. 12, no. 2, pp. 167–176, 1990.

[3] J. A. Potteiger, J. M. Schroeder, and K. L. Goff, "Influence of music on ratings of perceived exertion during 20 minutes of moderate intensity exercise," *Perceptual and Motor Skills*, vol. 91, no. 3, pp. 848–854, 2000.

[4] K. A. Brownley, R. G. McMurray, and A. C. Hackney, "Effects of music on physiological and affective responses to graded treadmill exercise in trained and untrained runners," *International Journal of Psychophysiology*, vol. 19, no. 3, pp. 193–201, 1995.

[5] C. I. Karageorghis and P. C. Terry, "The psychophysical effects of music in sport and exercise: A review," *Journal of Sport Behavior*, vol. 20, no. 1, p. 54, 1997.

[6] J. Edworthy and H. Waring, "The effects of music tempo and loudness level on treadmill exercise," *Ergonomics*, vol. 49, no. 15, pp. 1597–1610, 2006.

[7] C. I. Karageorghis, P. C. Terry, and A. M. Lane, "Development and initial validation of an instrument to assess the motivational qualities of music in exercise and sport: The brunel music rating inventory," *Journal of sports sciences*, vol. 17, no. 9, pp. 713–724, 1999.

[8] C. Bacon, T. Myers, and C. Karageorghis, "Effect of music-movement synchrony on exercise oxygen consumption.," *The Journal of sports medicine and physical fitness*, vol. 52, no. 4, pp. 359–365, 2012.

[9] AboutHealth, "http://running.about.com/od/getstartedwithrunning/ht/runwalk.html,"

[10] M. Viru, A. Hackney, K. Karelson, T. Janson, M. Kuus, and A. Viru, "Competition effects on physiological responses to exercise: performance, cardiorespiratory and hormonal factors," *Acta Physiologica Hungarica*, vol. 97, no. 1, pp. 22–30, 2010.

[11] I.-M. Lee, E. J. Shiroma, F. Lobelo, P. Puska, S. N. Blair, P. T. Katzmarzyk, L. P. A. S. W. Group, *et al.*, "Effect of physical inactivity on major non-communicable diseases worldwide: an analysis of burden of disease and life expectancy," *The lancet*, vol. 380, no. 9838, pp. 219–229, 2012.

[12] R. Prieto-Diaz and G. Arango, "Domain analysis: Acquisition of reusable information for software construction," 1989.

[13] D. M. Eichberg, "Domain model and domain modelling,"

[14] N. Zhao, "Full-featured pedometer design realized with 3-axis digital accelerometer," *Analog Dialogue*, vol. 44, no. 06, 2010.