SOFTWARE ENGINEERING 1
REPORT 3

---

napIT

Dec 15, 2017

Madhura Daptardar
John Grun
Preetraj Gujral
Manasi Mehta
Siddhi Patil
Aishwarya Srikanth

URL: https://github.com/napIT-Rutgers-SE-Fall-2017

Contents

SUMMARY OF CHANGES

- The major change is that the food part, where the user could enter the number of calories consumed has been dropped. Because of this, the number of use cases has been reduced to 8.
- Another feature that has been dropped is the smart alarm feature in the sleep section of the application.
- System requirements have been enumerated in a more understandable way.
- A clear distinction between the functional and non-functional requirements has been added
- Priority table has been modified on the basis of more requirements, both functional and non-functional.
- In UC-3, the kind of database being used has been made clear and explicitly mentioned
- UC-10, which is now UC-8 now clearly explains what kind of activity and how it will be read by the system
- Use Case Diagram has been modified to look cleaner and easier to understand
- The traceability matrix has been done again and matches the Requirements with the appropriate use cases.
- Alternate flow of events have been added to some use cases to cover all major possible cases.
- Major problem seemed to be present in the Smart Alarm use case, which no more exists
- Traceability matrix has been redone
- Diagrams have been edited on the basis of new requirements
- Manage Accounts domain model has been reoriented as per the actors' position.
- Changed priority weights of the functional requirements.
- Changes in the database schema.
- Added more explanation on the data collection and working of SVM.
- Added another data structure
- Added few sections in Class Diagrams: Design Patterns and OCL.
- Added History of Work

1. CUSTOMER STATEMENT OF REQUIREMENTS (CSR)

1.1 Problem Statement

"Healthy brains depend on healthy sleep; Healthy bodies depend on healthy sleep"

These are two of the Golden Sleep Principles from the website of World Sleep Day Org.[14] Sleep is not just a passive process, but rather a highly dynamic process that is terminated by waking up. Throughout the night a specific number of sleep stages that are repeatedly changing in various periods of time take place. These specific time intervals and specific sleep stages are very important for determining sleep anomalies and in turn the health problems. There exists an observable relationship between sleep and ability to function throughout the day for each individual. After all, everyone experiences fatigue, bad mood, or lack of focus, that so often follow a night of inadequate sleep. Lack of sleep can be attributed to external factors like stress, work pressure etc. What many do not realize is that an individual's sleeping habits are associated with long-term health consequences, including chronic medical conditions like diabetes, high blood pressure, and heart disease, and that these conditions may lead to a shortened life expectancy.

Research into sleep and its associated health abnormalities has had a relatively recent surge, and sleep quality has been shown in many investigations to be an important element of good health. In the United States, more than 70 million people suffer from a sleep disorder, and modern lifestyles have led Americans sleeping approximately 2 hours less per night as compared to a 100 years ago. Determining the risks posed by anomalous sleeping habits is complicated. Medical conditions are slow to develop and have multiple risk factors connected to them. Abnormalities in the sleep cycle are linked with neurocognitive consequences ranging from performance decrements, slower response times and decreased cognitive ability.

The main idea of our project is to monitor and record the sleep pattern using sensors commonly available in Smartphone. These readings will be recorded over a period of nights. With the help of these readings, a sleep profile will be generated for a specific user. The user profile will undergo further examination to detect whether sleep is proper and improper sleep. This can happen due to a large amount of movement during sleep, shortened duration of sleep, periodic wake ups or variation in sleep timings. Based on the analysis of the sleep pattern and daily activity of a user, remedies will be suggested.

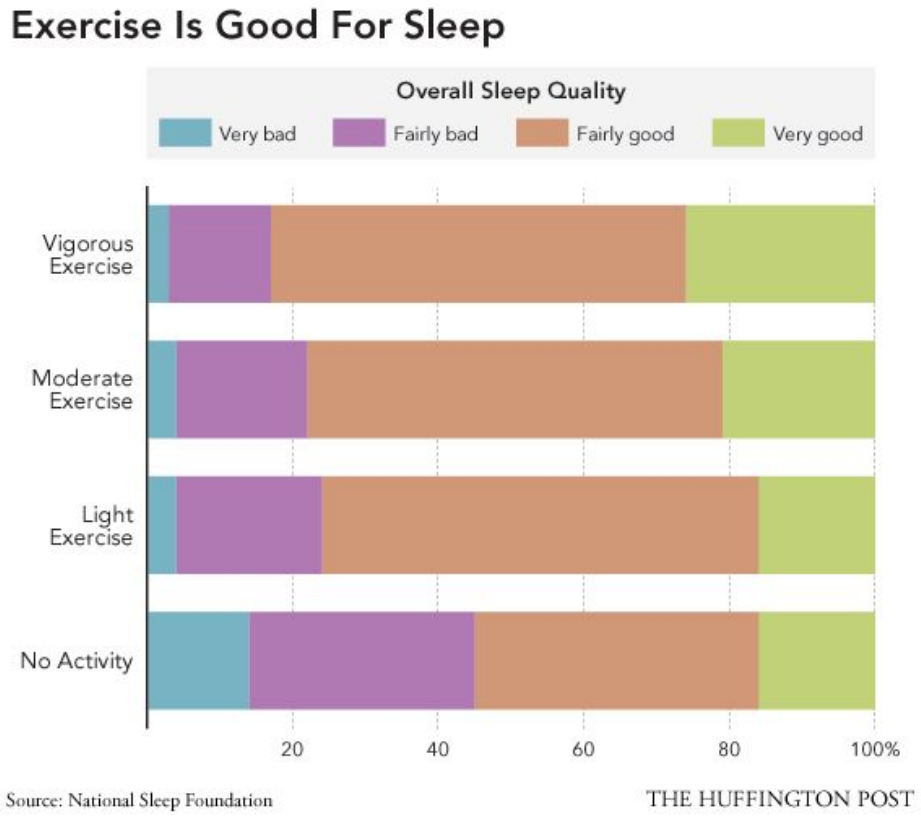


Fig1: Bi-directional relationship between exercise and sleep[17]

The above figure shows that there exists a bi-directional relationship between sleep and exercise. If a person does not get sufficient sleep at night, then he will be less active during the day and if a person is less active during the day, he will not get proper sleep at night. Most of us do not find time to exercise because of our hectic routine. Exercising

regularly and healthy eating habits, help to boost our energy, control our weight, keep us in a good mood and most importantly, reduces our risk to medical conditions, especially those related to heart and brain. Most of us know this but we are still reluctant to work out because we do not have adequate motivation to do so.

In order to implement the above idea, we are planning to design a mobile application to analyze a person's sleep pattern. This application helps detect whether the user's sleep is proper or improper and provide a suggestion accordingly. In addition to an individual, doctors, researchers, hospitals may find this application useful. The step by step method of the functioning of this application from the user's perspective is described below.

After noticing for more and more days where it was difficult to stay awake during the daily grind, it became painfully apparent that we were simply not getting proper sleep. Upon searching around the internet we started finding articles and websites linking poor sleep to many health issues such as heart disease and diabetes. We decided that my sleep issues may be more severe than we had previously realized. We spoke to a few friends about the issue and one recommended an application to monitor my sleep patterns called napIT. We installed napIT on my phone.

This application came as a boon at this stage. After registering, there were quite a things which caught our attention as sections for Exercise and Sleep. We realised that these were the things that we needed to look after as we was neglecting them a lot. This app allows me to track my everyday activity, for example when we are running, it works as a treadmill in a way, where it gives me the amount of time we worked out and the meters covered along with the calories we burnt for the session. Further it provides the most important function where the sleep pattern is analyzed and based on it sleep is classified as proper or improper. The application analyses as to why the sleep is improper and based on that along with the daily activities provides a suggestion as to how we can overcome it to take a restful nap at nights.

1.2 Glossary of Terms

- **Anomaly**: Anomaly is something that differs from the norm, also known as outliers, which may be indicators of a medical condition.

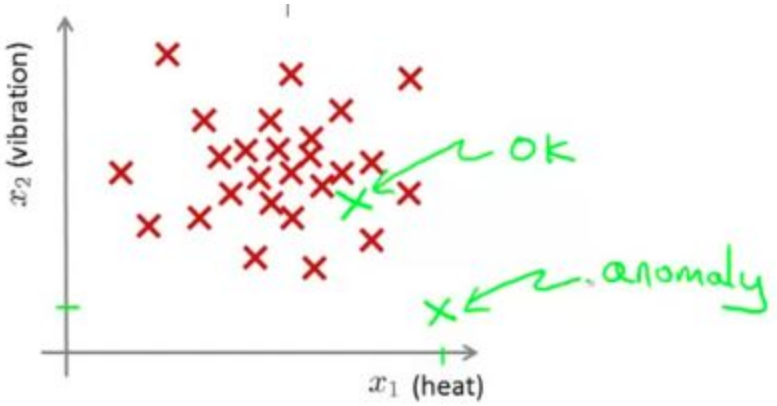


Fig 2: Anomaly

- **Sleep cycle**: The sleep cycle is an oscillation between the slow-wave and REM phases of sleep, sometimes called the ultradian sleep cycle, sleep–dream cycle, or REM-NREM cycle, to distinguish it from the circadian alternation between sleep and wakefulness. In humans, this cycle takes 1–2 hours[9].

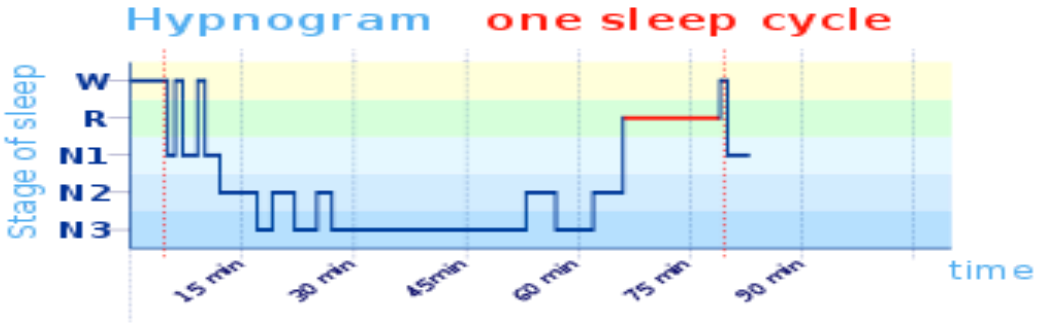


Fig 3: Sleep cycle

- **Sleep disorder**: Also known as somnipathy, it is a medical disorder of the sleep patterns of a person or animal. Some sleep disorders are serious enough to interfere with normal physical, mental, social and emotional functioning.
- **Accelerometer:** A device that measures proper acceleration.Proper acceleration, being the acceleration (or rate of change of velocity) of a body in its own instantaneous rest frame is not the same as coordinate acceleration, being the acceleration in a fixed coordinate system.[8]
- **App Store**: It is a digital distribution platform for mobile apps. The store allows users to browse and download apps.
- **User** : A person operating the system.
- **Unique username**: A username that has not yet been used in the application.
- **User Interface (UI)** : In the industrial design field of human–computer interaction, it 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 operator's decision-making process.

2. USER STORIES

The functional and nonfunctional requirements have been enumerated from the point of view of the user as user stories.

- **Logging in:** The user can create a new account or log into an existing account. While creating a new account the user needs to insert a unique username and password. If the user is logging into an existing account, his login credentials will be verified and the user can access all the features of the application.
- **Exercise:**
    1. *Start Activity:* Once the user starts an activity, for example, running, the sensor, i.e. the accelerometer will sense the activity and start recording the data.
    2. *Stop Activity:* Once the user clicks on stop activity, the data will be saved and the activity will terminate.
    3. *View Statistics:* This section will display results as to how much time the activity was performed, how many meters the user ran and the amount of calories burnt.
- **Sleep:**
    1. *Start Activity:* The user selects this option when he is about to sleep. The sensor will detect the user's sleeping pattern till he is awake.
    2. *Stop Activity:* If the user does not get up because of the alarm, but later, then he can select the stop option to terminate the activity.
    3. *View Statistics:* The user can view his sleeping pattern. This section will display two important results: the anomaly detected and suggestion from the application.
        a. *Anomaly:* If the sleeping pattern is not healthy, and some kind of anomaly is detected, then the detected anomaly will be shown to the user.

b. *Suggestion:* If the detected sleeping pattern is not healthy and some kind of anomaly is detected, then the user will be shown a set of suggestions to improve his sleeping pattern and ultimately help improve his health.

Priority Weight for Functional Requirements from 1 - 5. 5 is highest, 1 is lowest.

Requirements	Priority Weight	Requirement Description
REQ-1	1	Login and Authentication
REQ-2	2	Start accelerometer for running
REQ-3	3	Start sensor to capture the sleeping patterns
REQ-4	5	Sensor reads the data accurately
REQ-5	1	Save the statistics for sleep and exercises
REQ-6	3	Retrieve the sleeping pattern data
REQ-7	4	Process the data for anomaly detection
REQ-8	3	Get suggestions

Table 1: Priority weights for functional requirements

Priority Weight for Non-Functional Requirements from 1 - 5. 5 is highest, 1 is lowest.

Requirements	Priority Weight	Requirement Description
REQ-9	5	Sensor information stored in database
REQ-10	4	User friendly interface
REQ-11	3	Fast and prompt system
REQ-12	2	System is intuitive
REQ-13	4	Store least data on local storage
REQ-14	2	User can access stats whenever he/she wants
REQ-15	4	System communicates with the database automatically
REQ-16	5	Sensor should run in the background

Table 2: Priority weights for non-functional requirements

3. ON SCREEN APPEARANCE REQUIREMENTS

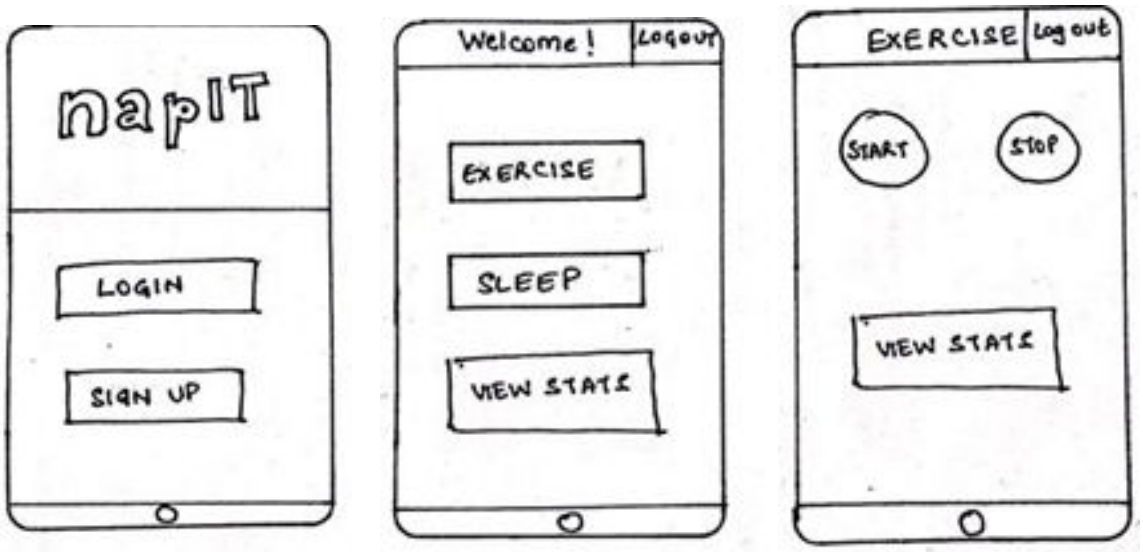


Fig 4: Paper prototyping for user stories

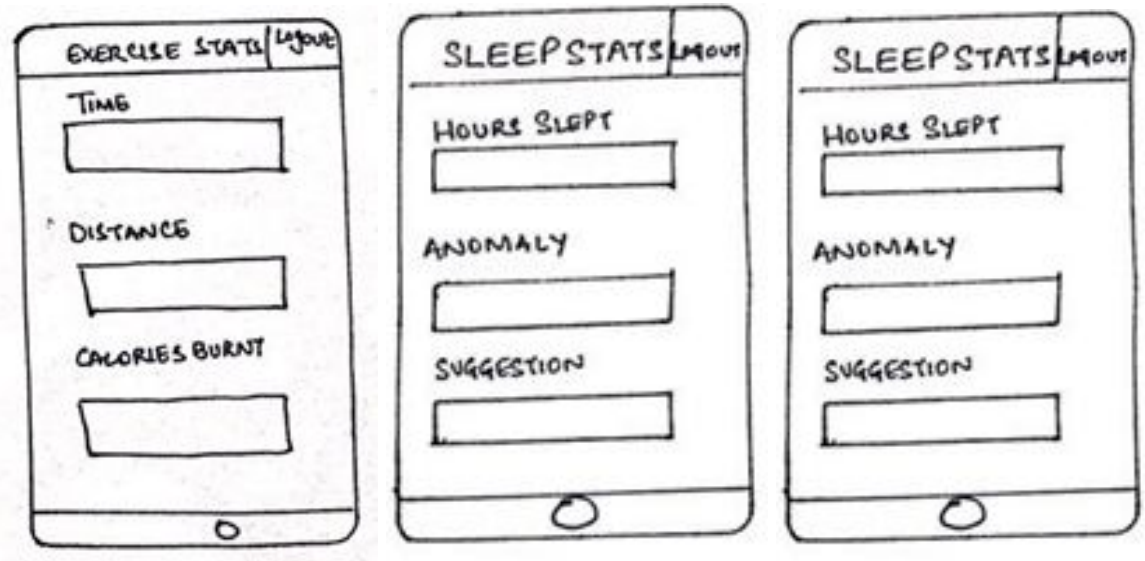


Fig 5: Paper prototyping for user stories

4. FUNCTIONAL REQUIREMENTS SPECIFICATIONS

4.1 Stakeholders

This is a personalized and user-friendly application to detect anomalies caused due to unhealthy sleep patterns. The application also gives suggestions to improve the sleeping cycle, thus improving the overall health. Every project has stakeholders. Stakeholders are people who have an interest in the application, are actively involved in the application, are affected by its outcome or can influence its outcome. They can be human or human organizations, differing from project to project. The ideal stakeholders of our project are:

Stakeholders	Description
Users	This application targets people who are interested in their long term health. This application allows users to monitor their sleeping habits and informs them of any anomalies that may arise.
Project managers	Interested in the successful completion of the project
Developers	Interested in the successful completion of the project
Testers	Interested in the successful completion of the project
Rutgers	Every successfully completed project increases the reputation of Rutgers as a university that produces successful graduates
Interested healthcare	There is an interest in health care monitoring by many in the healthcare field.

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

Table 2: Stakeholders

4.2 Actors and Goals

An actor can be human, a physical object or another system, that initiates the action or participates with the system to achieve the goal. Each actor, actively or passively contributes towards the working of the system.

Actor	Initiating/Participating	Goals
User	Initiating, Participating	A logged in user accessing the application and its features.
Database	Participating	The database stores and manages data along with the user information. It provides query support on data. Database also stores and manages the sensor information.
SDK	Participating, Initiating	SDK provides an interface to APIs in order to access the sensors, displays, system calls, operating system, etc. It also provides framework to construct application.
Sensor	Participating	Sensors collect data regarding the sleeping patterns of the user

Table 3: Actors and Goals

4.3 Use Cases

In software engineering, use cases are a list of actions or event steps typically defining the interactions between the actors and the system or the application, in order to achieve the goal. A textual description of the use cases, a diagrammatic representation and a Traceability Matrix which maps the use cases to the user requirements is given below.

4.3.1 Casual Description

The use cases are as follows:

UC-1: Login - Allows the user to log into the application as an existing user.
UC-2: Register- If the user does not have an existing account, the user can create a new account in the system.
UC-3: Manage Account - Allows the user to store their personal details, such as their name, age, weight, height, gender etc in the application's database. (<<include>> UC 1)
UC-4: Monitor Sleep - After a successful login, the user can use the application to monitor sleep activity, whenever they are about to sleep. The built-in sensor inside the application then starts to detect the user's sleeping pattern data.
UC-5: Anomaly Detection - The user's sleeping pattern data is used for detecting an unhealthy sleeping cycle and any corresponding anomaly is detected. Then the user is informed about the detected anomaly.
UC-6: View statistics - Allows the user to view his statistics, related to the exercises done and sleeping pattern.
UC-7: Recommender - Depending on whether, an anomaly is detected or not from the user's sleeping pattern, the user can be given suggestions to improve his sleeping cycle and thus improve his health.

UC-8: Exercise - Whenever during the day the user's conducting some form of activity, like walking, running or exercising, the user can begin the activity. The application detects the distance covered and the approximate calories burnt.

4.3.2 Use Case Diagram

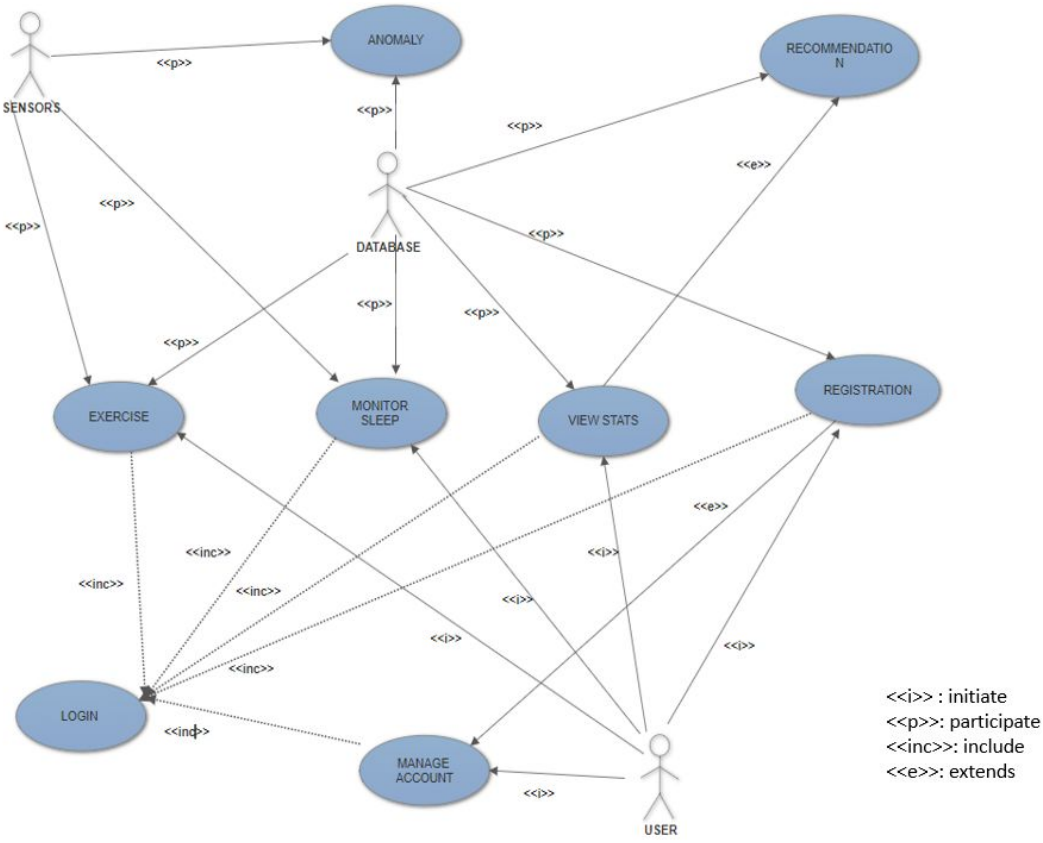


Fig 6: Use Case Diagram

4.3.3 Traceability Matrix

Traceability Matrix maps which Use Case is used to fulfill what Requirement.

	PW	UC-1	UC-2	UC-3	UC-4	UC-5	UC-6	UC-7	UC-8
REQ-1	1	x	x	x					
REQ-2	2								x
REQ-3	3								
REQ-4	5				x				
REQ-5	1				x	x	x		
REQ-6	3					x		x	
REQ-7	4					x	x		
REQ-8	3							x	
MAX PW		1	1	1	5	4	4	3	2
TOTAL PW		1	1	1	6	8	5	6	2

Table 4: Traceability Matrix

4.3.4 Fully-Dressed Description

Here, we have given an elaborated 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.

Use Case 1: Login

Use Case	**Functionality**
Use Case 1	Login
Related Req	REQ-1
Initiating Actor	User
Actors Goal	To login and access the apps features
Participating Actors	Database, Android SDK
Pre conditions	The user has an account
Post conditions	The user is logged in and can access the application Or The user failed to log into the account.
Flow of events for Main Success Scenario	1. The user opens the app and is prompted to type in account credentials. 2. The system verifies the credentials. 3. The user finally logs into the app and can access the app functions.
Flow of events for Extensions (Alternate Scenarios)	1. User enters invalid credentials. 2. Authentication fails. 3. User is prompted to re-enter credentials. a. User is finally able to login b. User fails more than 5 times in under 2 min

Table 5: Use Case 1: Login

Use Case 2: Register

Use Case	**Functionality**
Use Case 2	Registration
Related Req	REQ-1
Initiating Actor	User
Actors Goal	Create a new user account
Participating Actors	Database, SDK
Pre conditions	The user does not have a user account
Post conditions	User account was created for User. Or User gives up and does not finish creating an account and no further action is taken.
Flow of events for Main Success Scenario	1. The User clicks create account 2. The user is asked to create a new username and password. a. The username is checked against existing usernames in the database. Password is checked against regex to ensure password meets minimal requirements. i.) If username exists the user is informed and must pick a new unique username. ii.) If the username does not exists the user is allowed to continue. iii.) The password does not match the minimal requirements. E.g. Does not match regex condition. iv.) The password does match regex and user is allowed to continue. 3. User proceeds to Enter Personal details.
Flow of events for Extensions (Alternate Scenarios)	1. User clicks the create user account. 2. User gives up and does not finish completing creating a new account.

Table 6: Use Case 2: Register

Use Case 3: Manage Account

Use Case	**Functionality**
Use Case 3	Manage Account
Related Req	REQ-1
Initiating Actor	User
Actors Goal	To enter information about themselves and manage app settings
Participating Actors	Database(SQLite)
Pre conditions	The user should be logged into the system.
Post conditions	The user's details have been updated in the system.
Flow of events for Main Success Scenario	1. User clicks settings 2. User enters settings and personal details 3. User entries are validated a. User entries are valid b. User entries are not valid 4. User entries are valid 5. User saves changes
Flow of events for Extensions (Alternate Scenarios)	1. User clicks settings 2. User enters settings and personal details 3. User entries are validated a. User entries are valid b. User entries are not valid 4. User does not saves changes

Table 7: Use Case 3: Manage Account

Use Case 4: Monitor Sleep

Use Case	**Functionality**
Use Case 4	Monitor Sleep
Related Req	REQ-1, REQ-4.
Initiating Actor	User
Actors Goal	Cause the app to monitor user's sleep patterns
Participating Actors	Sensor, Database, operating system, SDK
Pre conditions	The user should be logged into the system.
Post conditions	User sleeping pattern sensor data is logged to the database
Flow of events for Main Success Scenario	1. User clicks begin sleep monitoring 2. Application starts recording sensor data. 3. Application filters data. 4. App attaches a timestamp 5. App stores data into database. 6. User stops sleep monitoring when they wake up.
Flow of events for Extensions (Alternate Scenarios)	1. User forgets to click begin sleep monitoring a.) Data is not collected for that period of time. b.) Data is not used for anomaly detection. 1B.) User does not place phone in required position. 1B.) Data is filtered out since it is far too stable. 2. User forgets to press Stop a.) Data keeps getting collected b.) Useless data recorded and results affected

Table 8: Use Case 4: Monitor Sleep

Use Case 5: Anomaly Detection

Use Case	**Functionality**
Use Case 5	Anomaly detection
Related Req	REQ-1, REQ-4, REQ-5.
Initiating Actor	User -- probably a triggered event
Actors Goal	To analyze the sleep pattern and look for anomalies
Participating Actors	Database, Android SDK
Pre conditions	The user is logged into the system and a statistically minimal amount of data exists for analysis.
Post conditions	The user data is analyzed. If improper sleep is detected, the user is informed about the same.
Flow of events for Main Success Scenario	1. Database is queried for data on users in the same age group. 2. Data from users in same age group are used to established training set and cross validation set to train anomaly detector 3. Database is queried for sensor data on this user 4. User data is passed into anomaly detector. 5. Anomaly is detected. 6. Anomaly is classified. 7. Anomaly is stored in database. 8. User is informed.
Flow of events for Extensions (Alternate Scenarios)	1. Database is queried for data on users in the same age group. 2. Data from users in same age group are used to established training set and cross validation set to train anomaly detector 3. Database is queried for sensor data on this user 4. User data is passed into anomaly detector. 5. No anomaly is detected 6. Nothing new occurs

Table 9: Use Case 5: Anomaly Detection

Use Case 6: View Statistics

Use Case	**Functionality**
Use Case 6	View Statistics
Related Req	REQ-1, REQ-5
Initiating Actor	User
Actors Goal	View statistics
Participating Actors	Database, Android SDK
Pre conditions	The user is logged into the system and the data specific to the user has been collected.
Post conditions	User sleeping pattern sensor data along with exercise data is logged to the database
Flow of events for Main Success Scenario	1. User elects to view exercise or sleeping stats. 2. Application queries database for mentioned user selection. 3. Database gives application the corresponding data. 4. User is shown the appropriate data.
Flow of events for Extensions (Alternate Scenarios)	1. User elects to view the stats for a particular function. 2. Application queries database for user data. 3. Database query returns a data retrieval failure. 4. User is informed that not enough data has been collected.

Table 10: Use Case 6: View Statistics

Use Case 7: Recommender

Use Case	**Functionality**
Use Case 7	Recommender
Related Req	REQ-1, REQ-3, REQ-8
Initiating Actor	User
Actors Goal	Recommend corrective actions to the user
Participating Actors	Database, Android SDK
Pre conditions	User is logged in, and the data specific to the user has been collected. The user data has been passed to the anomaly detector.
Post conditions	The user is told how to attempt to correct the issue.
Flow of events for Main Success Scenario	1. The application informs the user on how to correct their issues. 2. User has acknowledged the recommendation
Flow of events for Extensions (Alternate Scenarios)	1. The application informs the user on how to correct their issues. 2. User closes the application. 3. User is informed again on next login until they acknowledged the recommendation.

Table 11: Use Case 7: Recommender

Use Case 8: Exercise

Use Case	**Functionality**
Use Case 8	Exercise
Related Requirements	REQ-1, REQ-3, REQ-5.
Initiating Actors	User
Actor Goals	To burn calories by keeping track of daily activity.
Participating Actors	Database
Pre Conditions	1. The user should be logged into the system. 2. The UI displays option to start activity.
Post Conditions	The user can begin their activity.
Flow of events for Main Success Scenario	1. a.) The UI displays an option to start the activity. b.) The user chooses to start the activity. c.) The accelerometer senses the distance covered and the calories burnt. 2. User forgets to press Start a.) Data is not collected for that period of time. b.) Data is not used for anomaly detection. 3. User forgets to press Stop a.) Data keeps getting collected b.) Useless data recorded and results affected

Table 14: Use Case 8: Exercise

4.4 System Sequence Diagram

The Use Case Description is diagrammatically depicted in the System Sequence Diagrams. They show the stepwise 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.

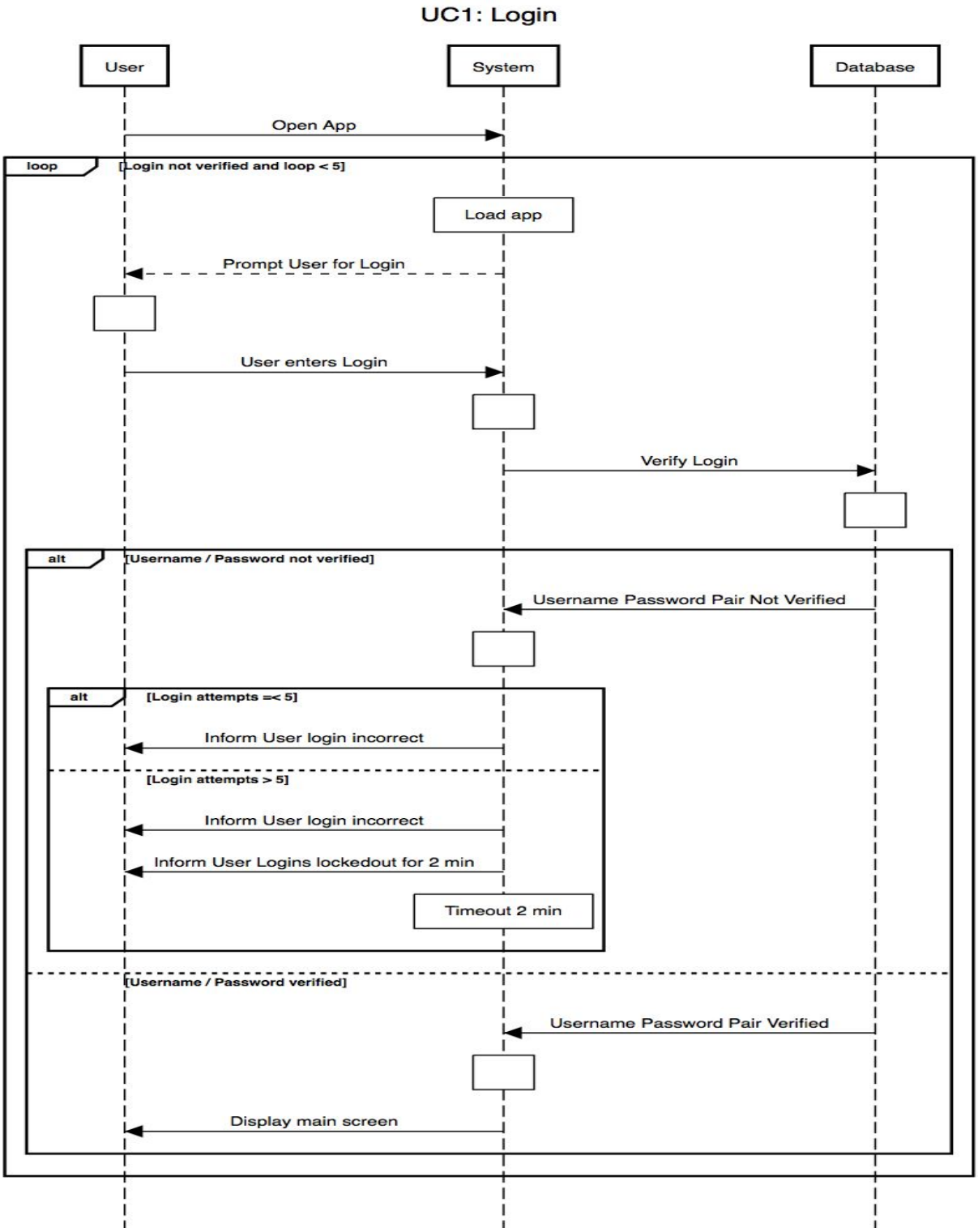


Fig 7: UC 1: Login

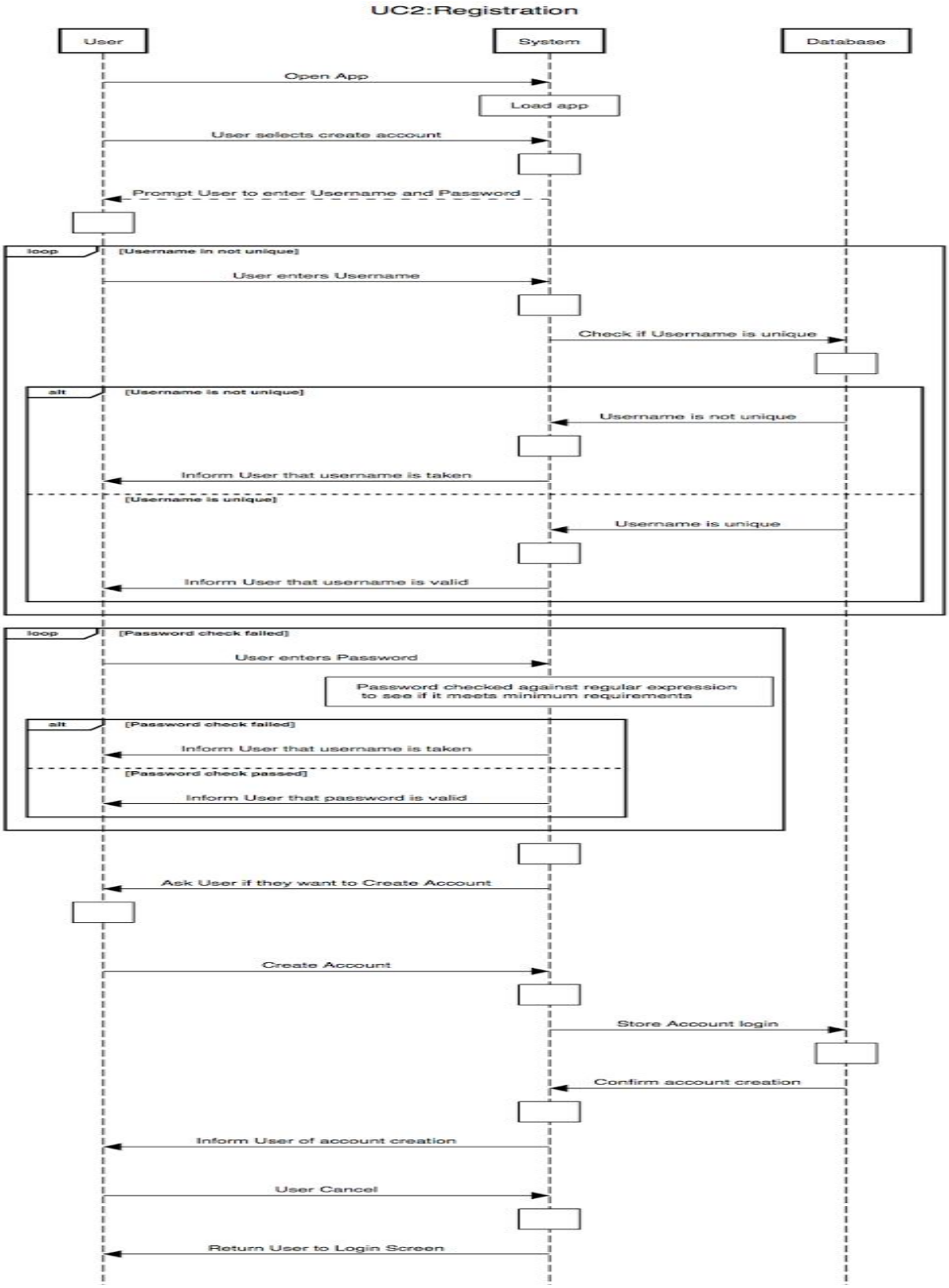


Fig 8: UC2: Registration

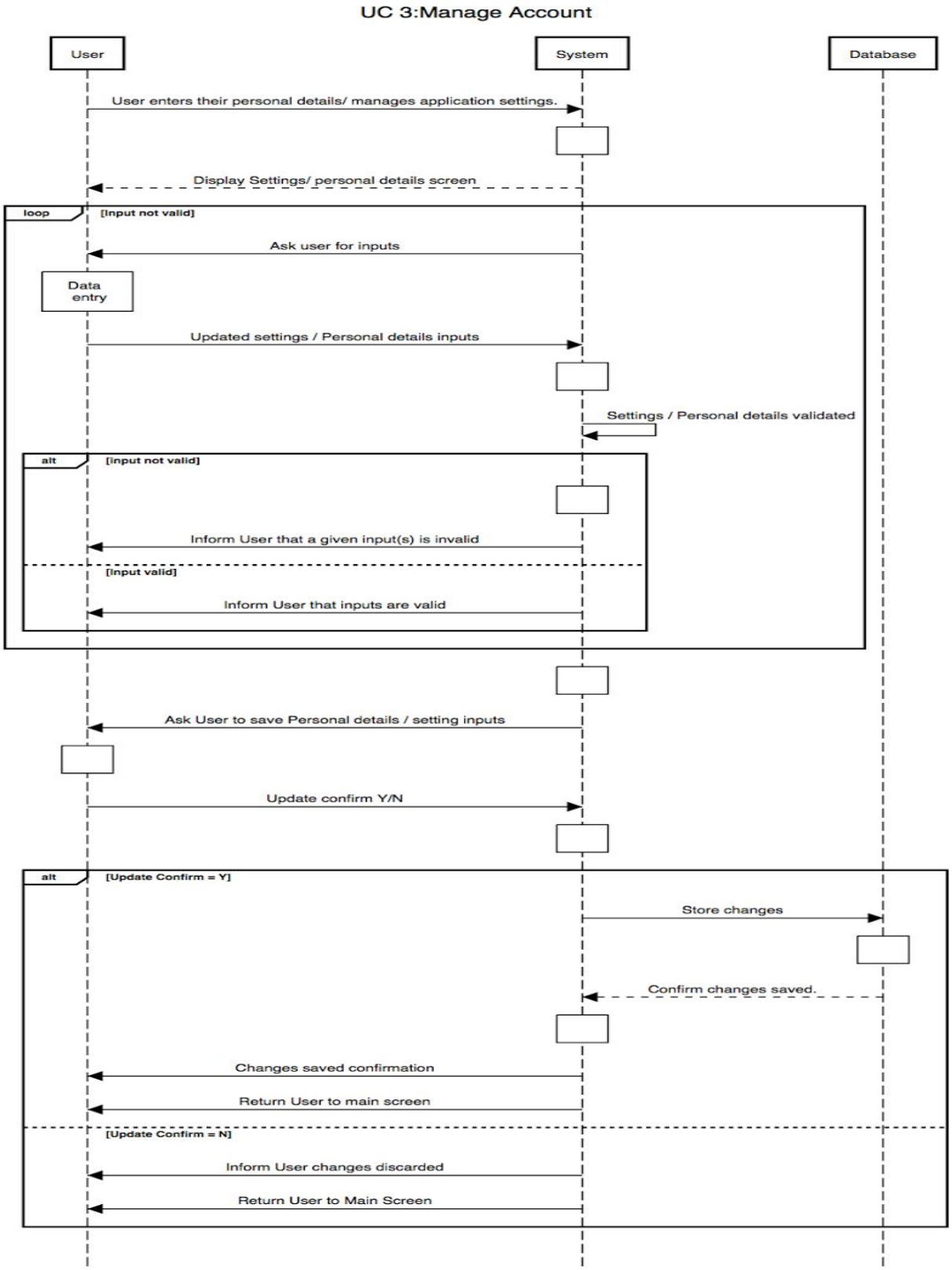


Fig 9: UC3: Manage Account

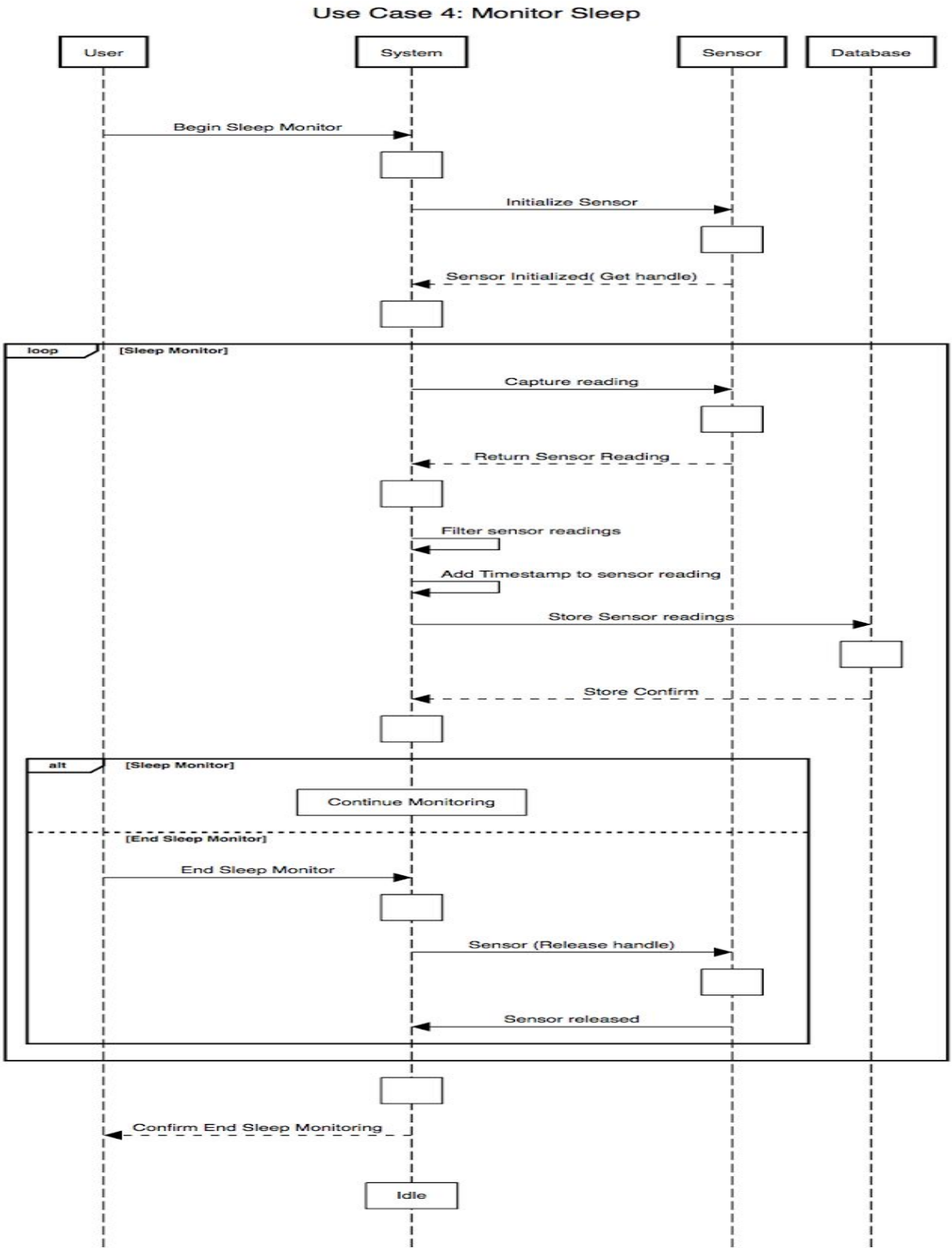


Fig 10: UC4: Monitor sleep

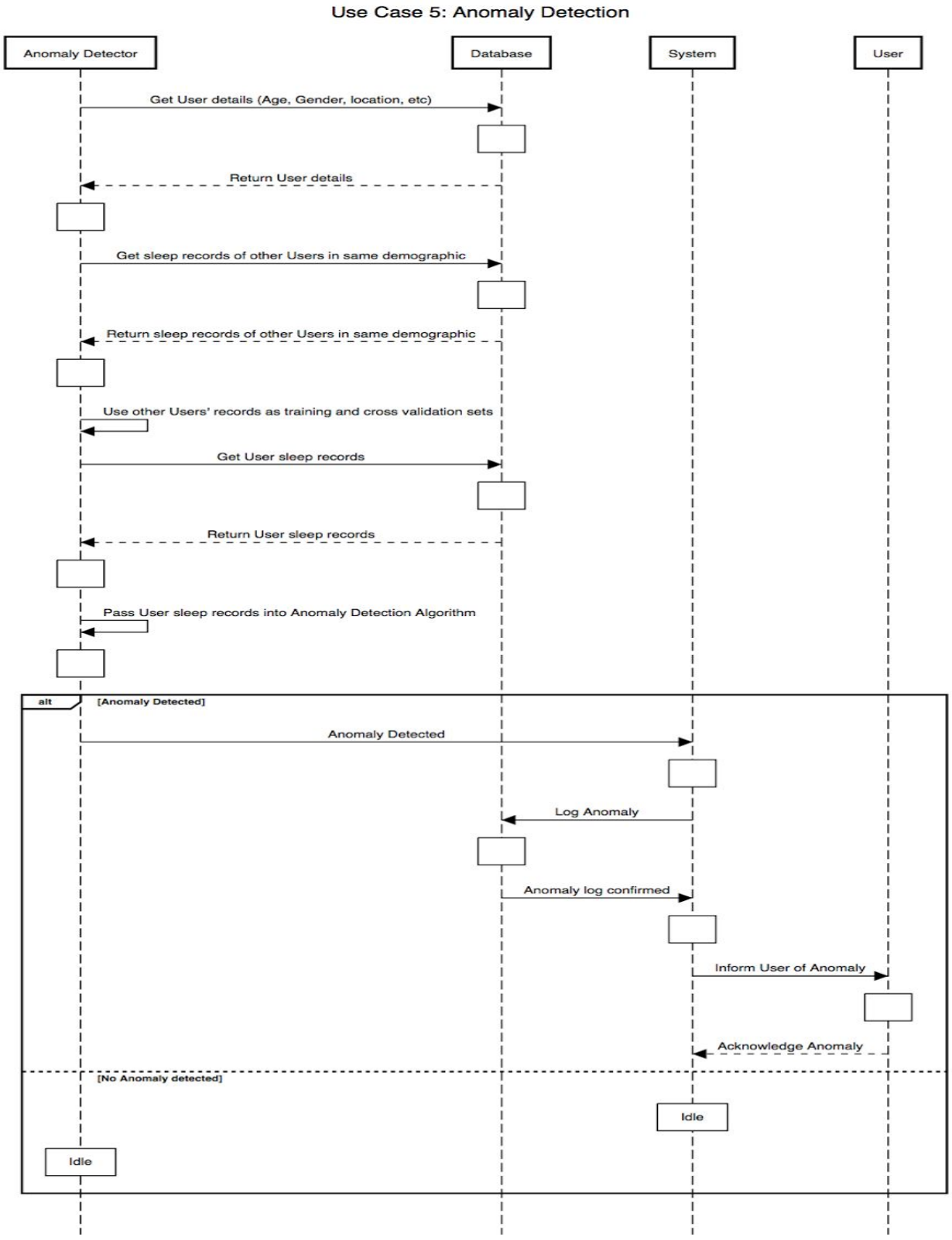


Fig 11: UC 5: Anomaly Detection

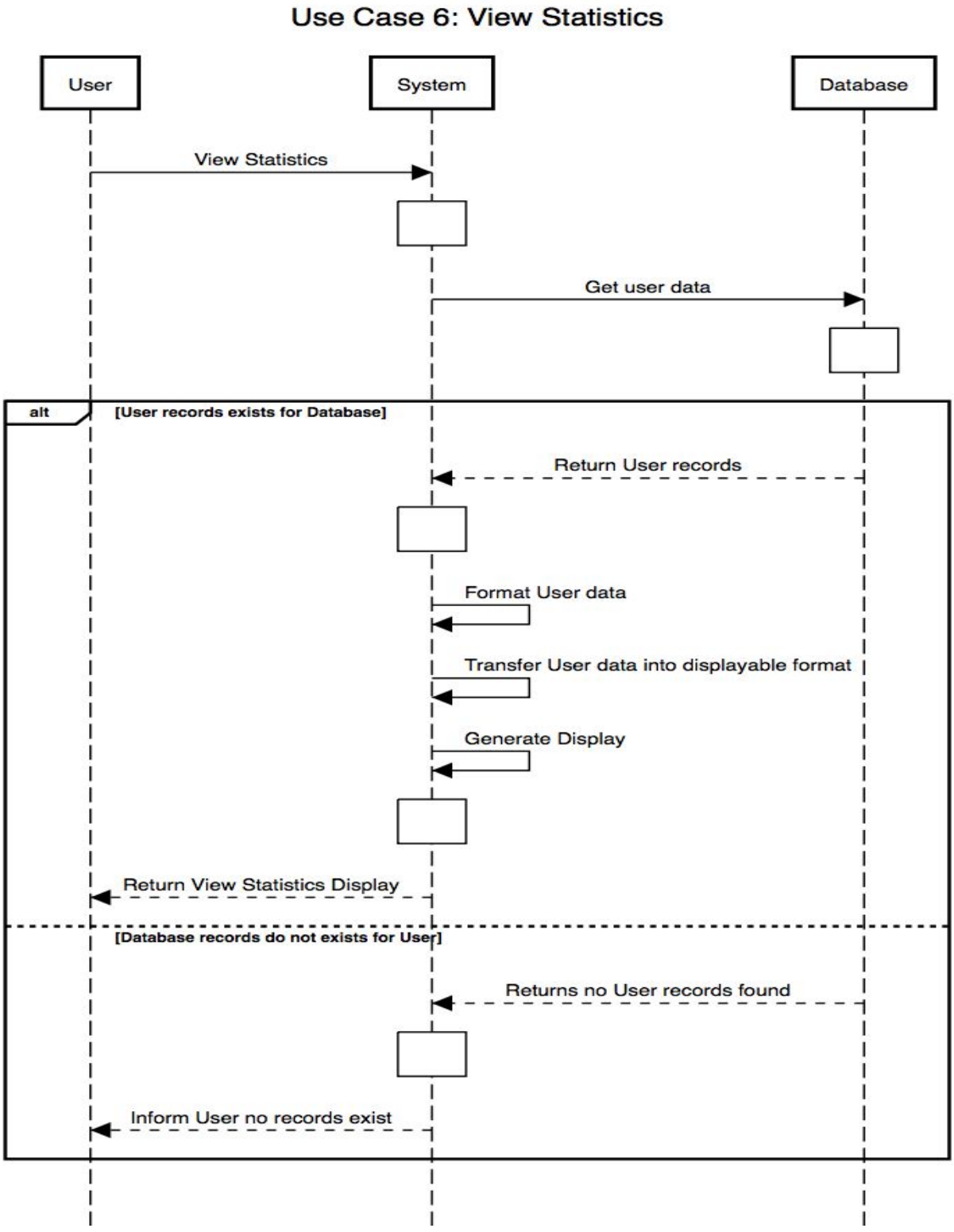


Fig 12: UC 6: View statistics

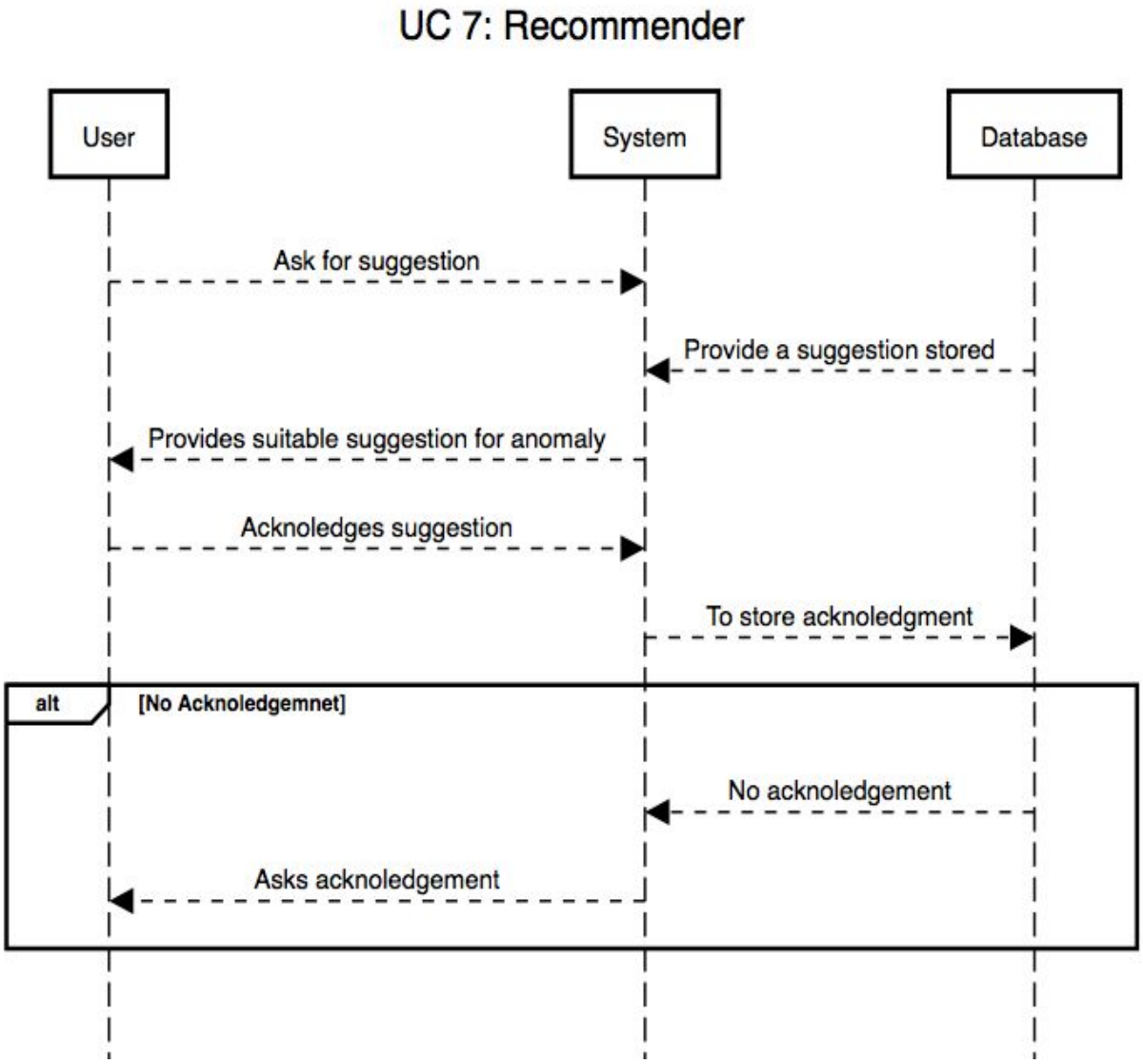


Fig 13: UC 7: Recommender

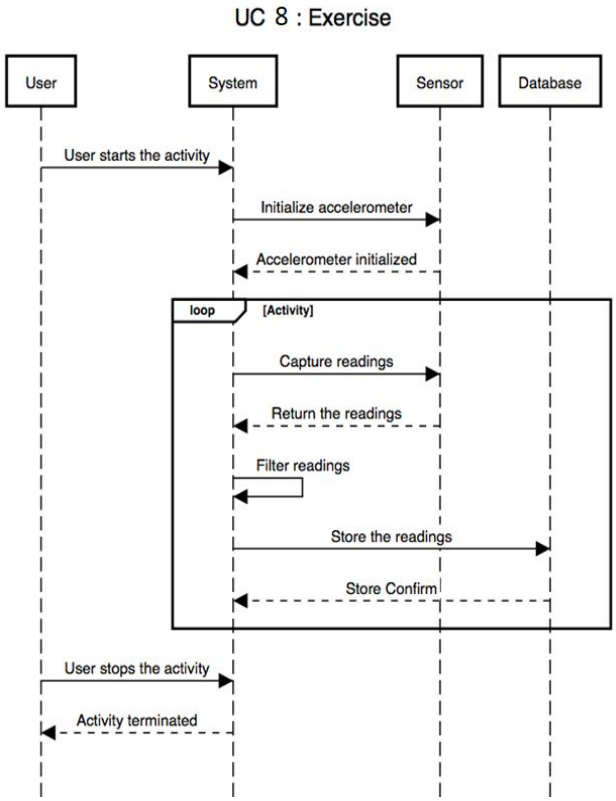


Fig 14: UC8: Exercise

5. USER INTERFACE SPECIFICATION

The user interface (UI), in the industrial design field of human–computer 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 operator's' decision-making process. In the instance of mobile application, it is in the form of a Graphical User Interface based on Android SDK.

The following are some user interactions with the mobile app:

1. Touch/Click : The user can touch the GUI component to express his intent if using it. For example, if the user clicks or touches the textbox, the text box will immediately become editable. If the user touches a drop down list, the options in the list will appear and the user can click on the desired one. Similarly, if the user touches a button, the functionality of the button will be executed.

2. Text Input: If the user touches the text box, an on-screen/physical keyboard will appear which will enable the user to enter text into it.

5.1 Preliminary Design

The initial design is designated as follows:

- **Login and Authentication:** When the user opens the application, he/she will see two buttons - login and register. If the user already has an account, he/she can login with the existing username and password which they already have and then hit the “Login” button. These details will be compared with the data stored in the database. If there is a match, the user will be authenticated and directed to the home page. If the user is new, he/she can create a new account by pressing the “Register” button. Once it is pressed, the user

will be asked to enter his/her personal details. The details entered in this page will be stored in a database once the user hits the "Submit" button.

- **Main Menu(Dashboard):** The main menu has different buttons like sleep, exercise and view statistics for the user to select. Once the user, selects an option he/she will be directed to the respective page. On the top-left corner, the image button will display a drop down list when it is clicked. The drop down list will have options which will allow the user to manage his/her profile and an option to logout. The user can log out anytime by clicking on the "logout" button.

- **Manage Account:** The user can manage his/her account by clicking on the image button on the top-left corner of the application and then select "Manage Profile". This includes updating or adding a profile picture, height, weight, age and current location of the user.

- **Sleep:** On clicking on the sleep button in the main menu, the user will be directed to a page concerning the sleeping habits of the user. This page has two buttons - Start and Stop. When the user is ready to go to sleep, he/she can activate the timer by click on the "Start" button. Once the user gets up, he/she should press the stop button to stop the timer. The user can view his sleep statistics by clicking on "View Statistics" from the main menu.

- **Starting an exercise:** As soon as the user begins to work out, the user can hit the "Start" button. The application will simultaneously be monitoring, recording and displaying the duration, miles covered and calories burnt while the user exercises. The exercise statistics can also be viewed by clicking on the "View Statistics" button in the main menu.

- **View Statistics:** On clicking this button, the user will be directed to a screen which will provide suggestions and display recorded results. This page will show the number of hours the user has slept, the anomaly detected and the suggestions based on anomaly

detected. This will also display the number of calories burnt during workout for that particular day. The user can log out anytime by clicking on the “logout” button.

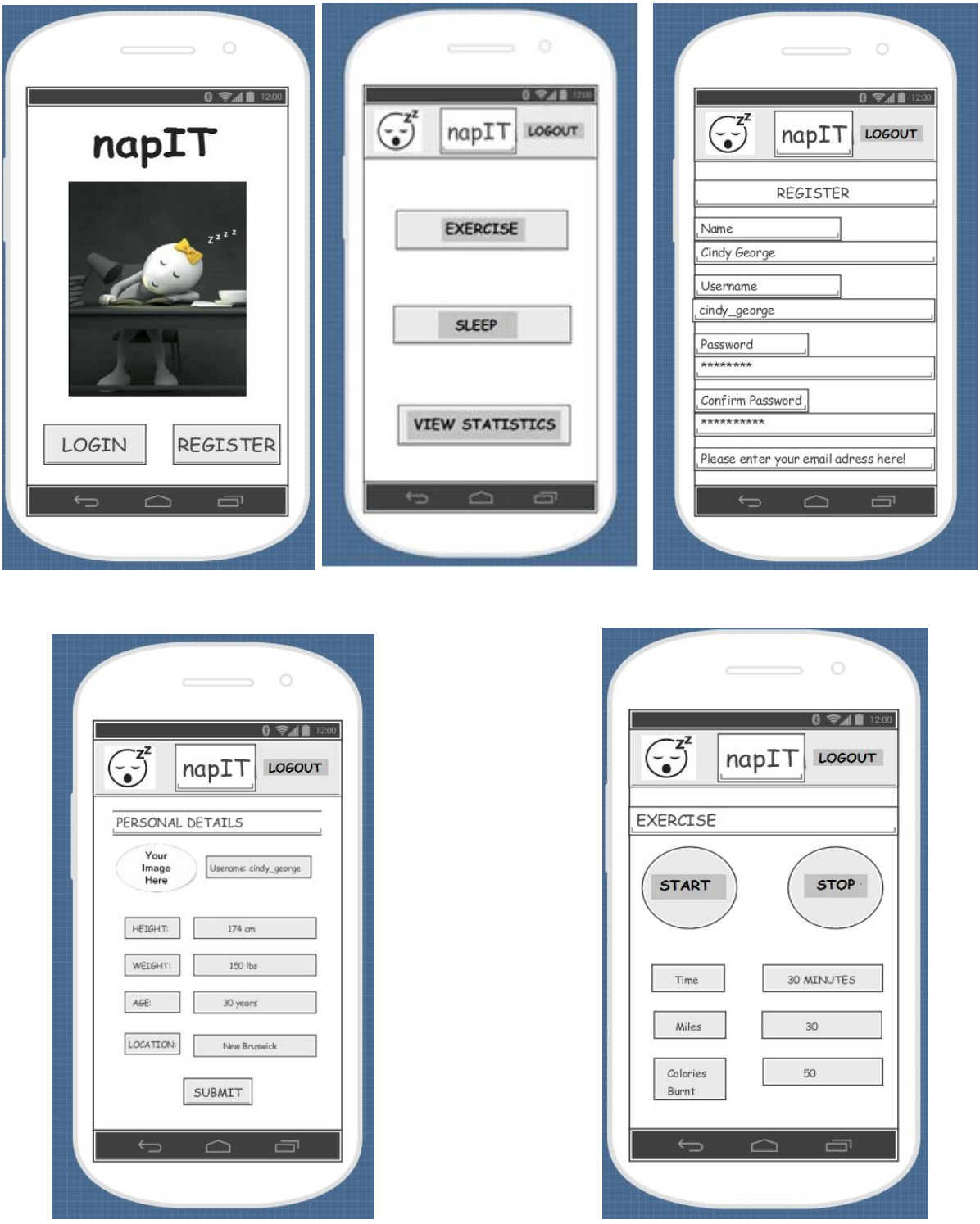


Fig 15: Mock-ups: Login, Main Menu, Register, Personal Details, Exercise

Fig 16: Mock-ups: Sleep, View-Statistics

5.2 User Effort Estimation

This section describes the number of mouse clicks or keystrokes the user has to make to navigate through the different windows in the mobile application and find the way to the context where the user can actually enter data.

1. **Start Page**

    - Navigation - 1 mouse click
        - ➔ Click button “LOGIN” or button “REGISTER”.

2. **Registration**

    - Navigation - 2 mouse clicks
        - ➔ Click on the “Register” button in start page.
        - ➔ Finally click on the “Submit” button.
    - Data Entry - 1 mouse click and 22 key presses minimum.
        - ➔ Click on the ‘Name’ field text box.
        - ➔ Enter the name(keys>1)
        - ➔ Press tab to move to the next field(key=1)
        - ➔ Enter user name(keys>1)
        - ➔ Press tab to move to the next field(key=1)
        - ➔ Enter password(keys>3)
        - ➔ Press tab to move to the next field(key=1)
        - ➔ Enter the password again(keys>3)
        - ➔ Press tab to move to the next field(key=1)
        - ➔ Enter email address(keys>10)

3. **Login**

    - Navigation - 1 mouse click
        - ➔ Click on the “LOGIN” button in start page.
    - Data Entry - 1 mouse click and 6 key presses minimum.

- ➔ Click on the ‘Username field text box.
- ➔ Enter the name(keys>1)
- ➔ Press tab to move to the next field(key=1)
- ➔ Enter password(keys>3)
- ➔ Press tab to move to the next field(key=1)

4. Main Menu

- Navigation - 1 mouse clicks
  - ➔ Click on any of the 4 buttons - “Sleep”, “Exercise”, “Sleep”, “View Statistics”.

5. Sleep

- Navigation - 3 mouse clicks
  - ➔ Click on the “Sleep” button in main menu.
  - ➔ Click on the “Start” button in the sleep page.
  - ➔ After waking up, click on the “Stop” button in the sleep page.

6. Exercise

- Navigation - 3 mouse clicks
  - ➔ Click on the “Exercise” button in main menu.
  - ➔ Click on the “Start” button in the sleep page.
  - ➔ After completing the work out, click on the “Stop” button in the sleep page.

7. View Statistics

- Navigation - 1 mouse click
  - ➔ Click on the “View Statistics” button in main menu.

8. Manage Profile

- Navigation - 3 mouse clicks
  - ➔ Click on the ‘Settings” (top left corner) button in the main menu.
  - ➔ Then click on the “Manage Profile” option.
  - ➔ Finally click on the “Submit” button after entering data.

- Data Entry - 3 mouse clicks and 12 key presses minimum.
  - ➔ Click on the “Height” field.
  - ➔ Enter the height(keys>4)
  - ➔ Press tab to move to the next field(key=1)
  - ➔ Enter weight(keys>3)
  - ➔ Press tab to move to the next field(key=1)
  - ➔ Enter age(keys=2)
  - ➔ Press tab to move to the next field(key=1)
  - ➔ Enter current location.

5.3 Effort Estimation using Use Case Points

Use Case Points (UCP) is a software estimation technique used to forecast the software size for software development projects [17].The Use Case Points (UCP) is calculated based on the following formula:

$$UCP = (UUCW + UAW) \times TCF \times ECF$$

where UUCW is the unadjusted use case weight, UAW the unadjusted actor weight, TCF the technical factor and ECF the environmental factor.

Unadjusted Use Case Weight (UUCW)

The UUCW is calculated based on the number and complexity of the use cases for the system. To find the UUCW for our system, each of our use cases must be identified and classified as Simple, Average or Complex based on the number of transactions the use case contains. Each classification has a predefined weight assigned. Once all use cases have been classified as simple, average or complex, the total weight (UUCW) is determined by summing the corresponding weights for each use case. The following chart shows the different classifications of use cases based on the number of transactions and the weight value assigned for each use case within the classification.

Use Case Classification	**No. of Transactions**	**Weight**
Simple	1 to 3 transactions	5
Average	4 to 7 transactions	10
Complex	8 or more transactions	15

According to the uses cases presented before we have the following use cases represented as Complex: Monitor Sleep, Anomaly Detection, Recommender, Exercise; the following as

Average: Login, Register and finally the following as Simple: Manage Accounts and View Statistics. Therefore:

$UUCW$ =(NumofSimpleUCs×5)+(NumofAverageUCs×10)+(NumofComplexUCs×15)

=2×5+2×10+4×15=90

Unadjusted Actor Weight (UAW)

The calculation for UAW is as follows:

UAW = (Num of Simple Actors×1)+(Num of Average Actors×2)+(Num of Complex Actors×3)

The following chart shows the different classifications of actors and the weight value assigned. For our application we have 1 simple actors, 2 average actors and 1 complex actor. So the UAW calculation becomes:

$$UAW = 1+2\times2+1\times3=8$$

Technical Complexity Factor (TCF)

Actor Classification	**Type of Actor**	**Weight**
Simple	External system that must interact with the system using a well-defined API	1
Average	External system that must interact with the system using standard communication protocols(e.g. TCP/IP, FTP, HTTP, database)	2

Complex	Human actor using a GUI application interface	3

The TCF is calculated by:

$$TCF = 0.6 + TF/100$$

To calculate the ECF, each of the environmental factors is assigned a value based on the team experience level. The diagram below shows the assigned values for our application. The values are multiplied by the weighted values and the total EF is determined.

So for our application:

$$TCF = 0.6 + (40/100) = 1.00$$

Environmental Complexity Factor (ECF)

To calculate the ECF, each of the environmental factors is assigned a value based on the team experience level. The diagram below shows the assigned values for our application.

So the ECF for our application is:

$$ECF = 1.4 + (-0.03 xEF) = 0.92$$

Thus, if we put everything together the use case points are:

$$UCP = (UUCW + UAW) \times TCF \times ECF = (90+8) \times 1 \times 0.92 = 90.16$$

Factor	**Description**	**Weight**	**Assigned Value**	**Weight x Assigned Value**
T1	Distributed system	2	5	10
T2	Response time/performance objectives	1	4	4
T3	End-user efficiency	1	5	5
T4	Internal processing complexity	1	4	4
T5	Code reusability	1	5	5
T6	Easy to install	0.5	5	2.5
T7	Easy to use	0.5	5	2.5
T8	Portability to other platforms	2	0	0
T9	System maintenance	1	2	2
T10	Concurrent/parallel processing	1	4	4
T11	Access for third parties	1	0	0
T12	End user training	1	1	1
			Total	40

Factor	**Description**	**Weight**	**Assigned Value**	**Weight x Assigned Value**
E1	Familiarity with development process	1.5	2	3
E2	Application experience	0.5	3	1.5
E3	Object-oriented experience of team	1.0	4	4
E4	Lead analyst capability	0.5	1	0.5
E5	Motivation of the team	1.0	5	5
E6	Stability of requirements	2.0	5	5
E7	Part-time staff	-1.0	0	0
E8	Difficult programming language	-1.0	3	-3
			Total	16

6. DOMAIN ANALYSIS

6. 1 Domain Model

Domain Analysis is the process of uniquely identifying the objects in an application domain. The main objective of Domain analysis is software reuse. The domain analysis is represented via a Domain Model, where different objects in a domain are related to each other. It is used to represent real world concepts.

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

Although it may take a significant amount of work, a software engineer is expected to work on the domain analysis because of the several advantages during the development phase. A well laid out domain model can lead to faster and an organized development where communication between different stakeholders can be significantly improved. Domain Analysis results in a well laid structure and improves the effectiveness of the domain. Any software or piece of work, once made should be in such a way that it is easy to access and understand later-on. It should be flexible enough to accommodate additional add-ons. It is always good to have a summary and a basic structural layout of what a developer is going to work on.

6.1.1 Concept Definitions

The components that work towards fulfilling the requirement are called concepts.Concepts can be ideas, thing or object in the domain. concept.Concepts can be physical objects,roles and responsibilities of people. For instance, we have DatabaseManager as a use case. DatabaseManager is a concept name for updating the database every time new information has been acquired/updated. The Tables below show the various Concepts mapped to their Responsibilities. The Type D and K refers to ’Does and ’Knows’ which means the 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.

Concept Definition of Use Case 1: Login

Responsibility Description	**Type**	**Concept Name**
Responsible for generating the initial login GUI	D	GUIGenerator
Allows the user to login into the application. Check the validity of the input. Check database for records. Allows access.	D	Login Interface
Responsible for signing out from the application.	D	Disconnector

Table 15: Concept Definition of Use Case 1: Login

Concept Definition of Use Case 2: Registration

Responsibility Description	**Type**	**Concept Name**
Acts as a link by controlling the interaction between different parts of the system.	D	MainController
Allows a new user to register into the system.	K	RegistrationInterface
Checks if the user entries are valid	D	Validator
Stores the registered users details in the database	D	DatabaseManager

Table 16: Concept Definition of Use Case 2: Registration

Concept Definition of Use Case 3: Manage Accounts

Responsibility Description	Type	Concept Name
Responsible for account information GUI	D	GUIGenerator
Keeps user details	K	UserDetailKeeper
Detects submit button selection	D	SelectionDetector
Passes user account information to UserDetailKeeper and retrieves all information from database	D	AccountManager
Stores Data from UserDetailKeeper to database	D	DatabaseManager

Table 17: Concept Definition of Use Case 3: Manage Account

Concept Definition of Use Case 4: Monitor Sleep

Responsibility Description	Type	Concept Name
Responsible for GUI generation	D	GUIGenerator
Detect if Start has been selected	D	SelectionDetector
Act depending on SelectionDetector	D	ResponseControl
Sensors accessed through SDK to monitor sleep and released when sleep time completed	D	SleepControl
Stores observed sleep data	K	ObserveControl
Analyze observed data	D	AnalysisManager
Store analysis	D	StatsControl

Table 18: Concept Definition of Use Case 4: Monitor Sleep

Concept Definition of Use Case 5: Anomaly Detection

Responsibility Description	Type	Concept Name
Can invoke the Anomaly detector with the user's id.	D	AnomalyController
Anomaly detector will inform the Main controller if an anomaly has been detected	D	AnomalyConveyor
Anomaly detector will ask the database interface for the user's information and compare it against the other user's data in the same demographic to train the dataset	D	AnomalyConnector
Anomaly is detected and classified	D	AnomalyDetector
Anomaly is stored in the database	K	DatabaseManager
Anomaly detected will be displayed to the user	D	GUIGenerator

Table 19: Concept Definition of Use Case 5: Anomaly Detection

Concept Definition of Use Case 6: View Statistics

Responsibility Description	Type	Concept Name
Displays the GUI with statistics visualization	D	GUIGenerator
Scrolls through all time statistics visualization	D	ScrollDetector
Contains statistics of the user	K	StatisticsKeeper
Retrieve the statistics from the database and populates the StatisticsKeeper	D	DatabaseManager
Based on the statistics of the user, it detects an anomaly and recommends a corrective action accordingly.	D	StatisticsInformation

Table 20: Concept Definition of Use Case 6: View Statistics

Concept Definition of Use Case 7: Recommender

Responsibility Description	Type	Concept Name
Responsible for recommending the corrective actions to the user.	D	RecommendationGenerator
Detects the button selection for suggestions.	D	SelectionDetector
It connects to the database and retrieves the corrective action.	D	DataRetriever
It keeps all the anomalies and corresponding corrective actions.	K	DataStorer
After acknowledgement, deletes the recommendation.	D	Deleter
Responsible for informing the recommendation again if not acknowledged.	D	Reinformer

Table 21: Concept Definition of Use Case 7: Recommender

Concept Definition of Use Case 8: Exercise

Responsibility Description	Type	Concept Name
Responsible for generating GUI	D	GUIGenerator
Detects the selection of Start/Stop button in the exercise page.	D	SelectionDetector
Responds to the selection of Start/Stop button	D	RunController
Accesses sensors through the Android SDK to calculate the speed of the walk or run once the "Start" button is pressed and release access to the sensors on pressing the "Stop" Button.	D	AccelerometerController
Stores the distance covered, step count and calories burnt in the database.	K	ExerciseInfoKeeper

Table 22: Concept Definition of Use Case 8: Exercise

6.1.2 Attribute Definitions

An attribute can be defined as description of a named slot of a specified type in a domain class. It is through these various attributes each concept is implemented. Attributes are the required quantities for the concepts to perform their functions. They can be a button or an entry field depending on the concept it is used with. The following tables show the Attributes of all the Concepts created previously long with the Attribute Definitions for each use case.

Attribute Definition for Use Case 1: Login

Concept	Attributes	Attribute Description
GUIGenerator	Button	To go to the login credentials.
Login Interface	User Details	To validate user entries

Table 25: Attribute Definition of Use Case 1: Login

Attribute Definition for Use Case 2: Registration

Concept	Attributes	Attribute Description
RegistrationInterface	EditText	To allow the user to register into the application
Validator	User Details	To validate user entries
DatabaseManager	User Information	To store the user details in the database

Table 26: Attribute Definition of Use Case 2: Registration

Attribute Definition for Use Case 3: Manage Accounts

Concept	Attributes	Attribute Description
GUIGenerator	Entry Fields	To get user information entered
UserDetailsKeeper	User Details	To store user's personal information
DatabaseManager	User Information	Stores User details

Table 27: Attribute Definition of Use Case 3: Manage Account

Attribute Definition for Use Case 4: Monitor Sleep

Concept	Attributes	Attribute Description
GUIGenerator	Button	Responsible for GUI generation.
SelectionDetector	Start/Stop Button	Detects if Start /Stop button has been selected.
ResponseControl	Start/Stop Button	Acts based on SelectionControl.
SleepControl	Sensor access	Sensors accessed through SDK to

		monitor sleep and released when sleep time completed
ObserveControl	Sleep data	Stores observed sleep data
AnalysisManager	Sleep data	Analyze observed data
StatsControl	Analyzed Sleep Data	Store analysis

Table 28: Attribute Definition of Use Case 4:Monitor Sleep

Attribute Definition for Use Case 5: Anomaly Detection

Concept	**Attributes**	**Attribute Description**
AnomalyControl	User records	Invokes anomaly detection with user's unique ID
AnomalyDetector	List	Anomaly is detected and classified based on the information
AnomalyConnector	User records	Anomaly detector will ask the database interface for the user's information
DatabaseManager	User records	The database is updated based on anomaly detected.
GUIGenerator	Display	Anomaly is displayed to the user

Table 29: Attribute Definition of Use Case 5: Anomaly Detection

Attribute Definition for Use Case 6: View Statistics

Concept	**Attributes**	**Attribute Description**
GUIGenerator	Graphs	Scrollable visualizations for the statistics data
StatisticsKeeper	History information of activities	distance(miles), time and calories
StatisticsInformation	Anomaly and corrective action.	Anomaly detected and corresponding corrective suggestion to be recommended.

Table 30: Attribute Definition of Use Case 6: View Statistics

Attribute Definition for Use Case 7: Recommender

Concept	Attributes	Attribute Description
GUIGenerator	Suggestion button	Displays Suggestion button
RecommendationKeeper	Recommendation details	The corresponding recommendation for the anomaly
PendingRecommendationInfo Keeper	Unacknowledged Recommendation	The unacknowledged recommendation from the user

Table 31: Attribute Definition of Use Case 7: Recommender

Attribute Definition for Use Case 8: Exercise

Concept	Attribute Name	Attribute Description
GUIGenerator	Start Button Stop Button	Starts the exercise(run/walk). Stops the exercise(run/walk).
SelectionDetector	Start Button Stop Button	Detects the selection of Start/Stop button in the exercise page.
RunController	Start Button Stop Button	Responds to the selection of Start/Stop button
AccelerometerController	Start Button Stop Button	Accesses sensors through the Android SDK to calculate the speed of the walk or run once the “Start” button is pressed and release access to the sensors on pressing the “Stop” Button.
ExerciseInfoKeeper	Exercise related data	Stores the distance covered, step count and calories burnt in the database.

Table 34: Attribute Definition of Use Case 8: Exercise

6.1.3 Association Definitions

Association definition of Use Case 1: Login

Concept Pair	Association Description	Association Name
GUIGenerator ↔ Login Interface	Sends Authentication request and authenticates	Sends request
GUIGenerator ↔Disconnector	The Disconnector has to inform the GUIGenerator the user has signed out so that it changes the user interface.	Inform

Table 35: Association Definition of Use Case 1: Login

Association definition of Use Case 2: Registration

Concept Pair	Association Description	Association Name
MainController ↔ RegistrationInterface	The MainController displays the RegistrationInterface	displays
RegistrationInterface ↔ Validator	RegistrationInterface sends the data entered by the user to the Validator	sends data
Validator ↔ RegistrationInterface	Validators validates and acknowledges the entries to the RegistrationInterface	acknowledges
Validator ↔ DatabaseManager	The Validators gives the data to the DatabaseManager when the user is registered	sends data

Table 36: Association Definition of Use Case 2: Registration

Association definition of Use Case 3 : Manage Accounts

Concept Pair	Association Description	Association Name
SelectionDetector ↔ AccountManager	SelectionDetector invokes AccountManager	invokes
AccountManager ↔ DatabaseManager	User details passed to DatabaseManager	passes information

SelectionDetector ↔ GUIGenerator	Updates GUI	changes GUI
UserDetailKeeper ↔ AccountManager	Keeps user details from AccountManager	Keep user details

Table 37: Association Definition of Use Case 3: Manage Accounts

Association definition of Use Case 4 : Monitor Sleep

Concept Pair	**Association Description**	**Association Name**
SelectionDetector↔GUIGenerator	Presents the button to monitor sleep.	start monitoring
SelectionDetector↔ResponseControl	Acts as the monitoring starts.	gets user sleep pattern
ResponseControl↔SleepControl	User sleep pattern sensed by the sensor	senses
SleepControl↔ObserveControl	Stores the sleep pattern	stores
ObserveControl↔AnalysisManager	Analysis the sleep pattern	analysis
AnalysisManager↔StatsControl	Stores the analysis	stores

Table 38: Association Definition of Use Case 4: Monitor Sleep

Association definition of Use Case 5 : Anomaly Detection

Concept Pair	**Association Description**	**Association Name**
MainController ↔ AnomalyDetector	Can invoke the Anomaly detector with the user's id. Anomaly detector will inform the Main controller if an anomaly has been detected	Anomaly Control
AnomalyDetector ↔ DatabaseManager	Anomaly detector will ask the database interface for the user's information and	Anomaly Connection

	compare it against the other user's in the same demographic	
AnomalyConveyor ↔ GUIGenerator	Anomaly detected will be displayed to the user.	Anomaly Displayer

Table 39: Association Definition of Use Case 5: Anomaly Detection

Association definition of Use Case 6 : 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
StatisticsInformation↔ StatisticsKeeper	Provides the anomaly detected	statsInfo
DatabaseManager↔ StatisticsKeeper	The DatabaseManager populates the StatisticsKeeper	populates
StatisticsInformation↔ GUIGenerator	Forwards the data for the generation of visualizations	forwards data

Table 40: Association Definition of Use Case 6: View Statistics

Association definition of Use Case 7 : Recommender

Concept Pair	Association Description	Association Name
SelectionDetector↔ RecommendationNotifier	The Selection Detector informs the Recommendation Notifier that the user has asked for recommendations.	informs
Database Manager↔ PendingRecommendationInfoKeeper	The Database Manager populates the PendingRecommendationInfo Keeper from the database.	populates
GUIGenerator	The GUIGenerator displays the recommendation to the user for the corresponding anomaly.	displays

Table 41: Association Definition of Use Case 7: Recommender

Association definition of Use Case 8: Exercise

Concept Pair	**Association Description**	**Association Name**
SelectionDetector ↔ RunController	Starts or stops recording exercise progress based on the button selected.	plays/stops
RunController ↔ AccelerometerController	Accesses the accelerometer once the start button is pressed to record the speed of walk/run.	Senses speed.
AccelerometerController ↔ ExerciseInfoKeeper	Stores the distance covered and step count from the accelerometer and the calories burnt calculated from our algorithm in a database.	Stores exercise related information.

Table 44: Association Definition of Use Case 8: Exercise

6.1.4 Traceability Matrix

Traceability Matrix is a table that shows the relationship between two elements.Here the use cases are mapped to domain concepts. The matrix helps to identify whether all the concepts are covered for a use case. The below table show the traceability matrix for concepts and use case.

	GUI Gener a-tor	Login Interfa ce	Discon -nector	Main Control ler	Registr ation Interfa ce	Valida-tor	Databa seMan-ager	UserDe tail Keeper	Selecti onDete -ctor
Login	x	x	x						
Registr ation				x	x	x	x		
Manag e Accoun ts	x						x	x	x
Monito r Sleep	x								x
Anoma ly Detecti on	x						x		
View Statisti cs	x								
Recom mender									
Exercis e	x								x

	AccountManager	Respo--nse Contro-l	Sleep Contro-l	AnalysisManager	ObserveControl	StatsControl	Anomaly Control	Anomaly Conneection	Anomaly Displayer
Login									
Registration									
Manage Accounts	x								
Monitor Sleep		x	x	x	x	x			
Anomaly Detection							x	x	x
View Statistics									
Recommender									
Exercise									

	StatisticsKeeper	StatisticsInformation	RecommendationKeeper	PendingRecommendationInfoKeeper	SleepDecider	AlarmRinger	Timer	DecisionMaker	Snoozer	Dismisser
Login										
Registration										
Manage Accounts										
Monitor Sleep										
Anomaly Detection										
View Statistics	x	x								
Recommender			x	x						
Exercise										

6.1.5 Domain Model Diagrams

In software engineering, a domain model is a conceptual model of the domain that incorporates both behavior and data.[15] In ontology engineering, a domain model is a formal representation of a knowledge domain with concepts, roles, datatypes, individuals, and rules.

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 does 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 use case is shown below:

Login:

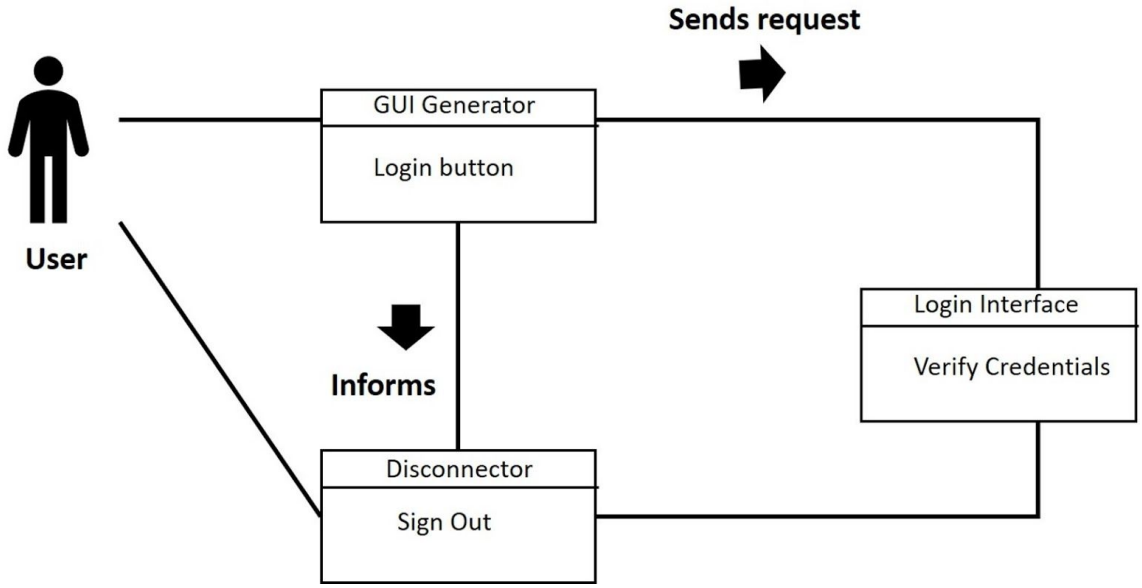


Fig 17: Domain Model: Login

Registration:

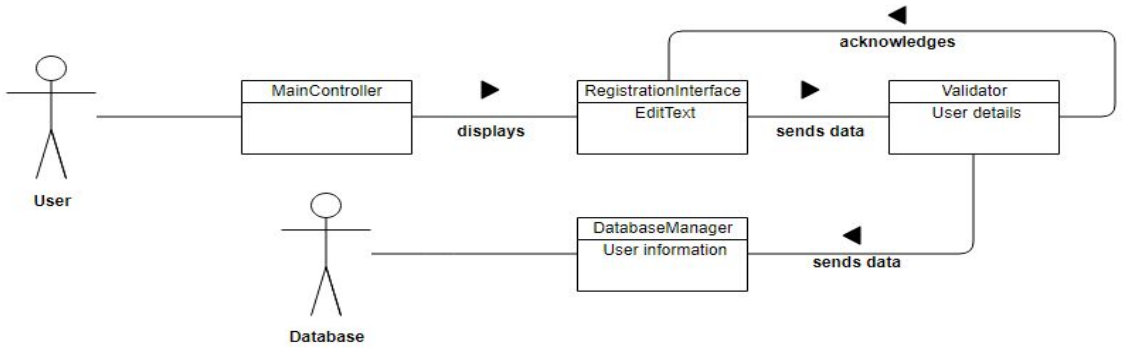


Fig 18: Domain Model: Registration

Manage Accounts:

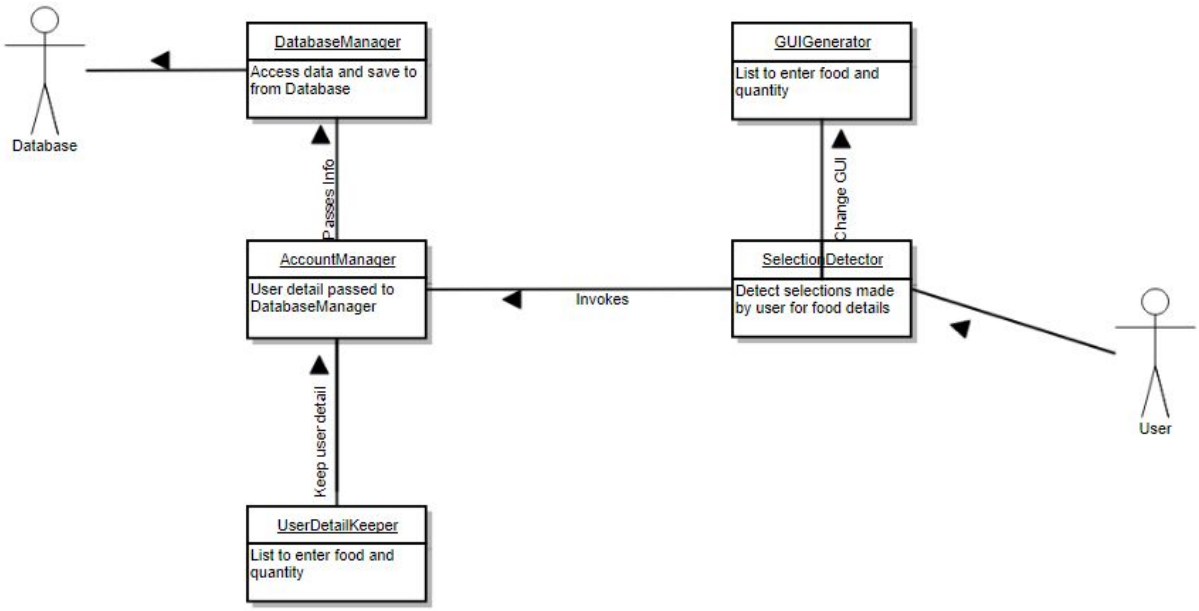


Fig 19: Domain Model: Manage Accounts

Sleep Monitor:

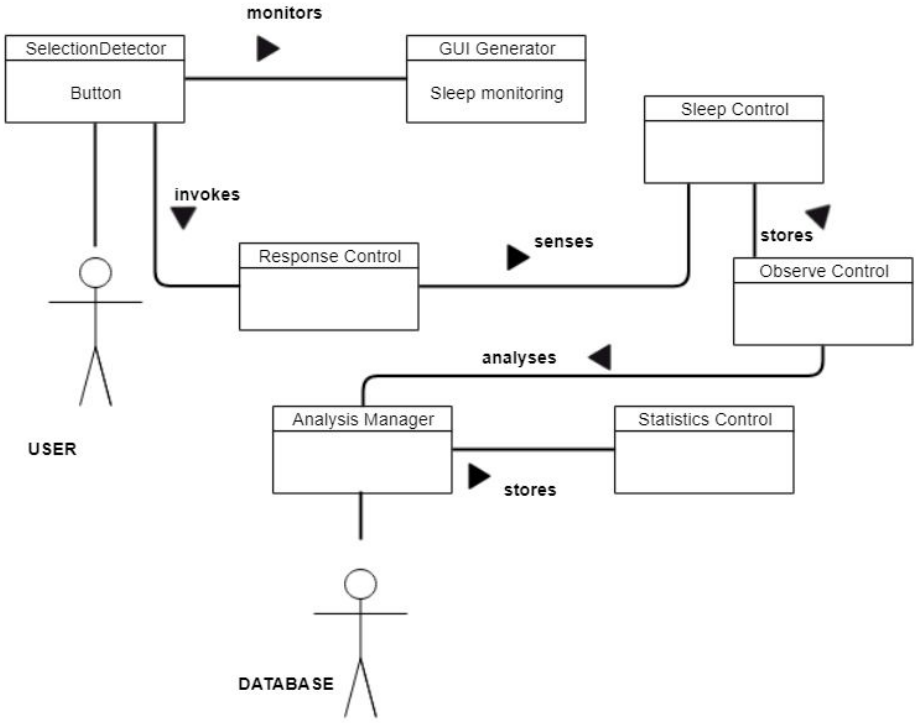


Fig 20: Domain Model: Sleep Monitor

Anomaly Detection:

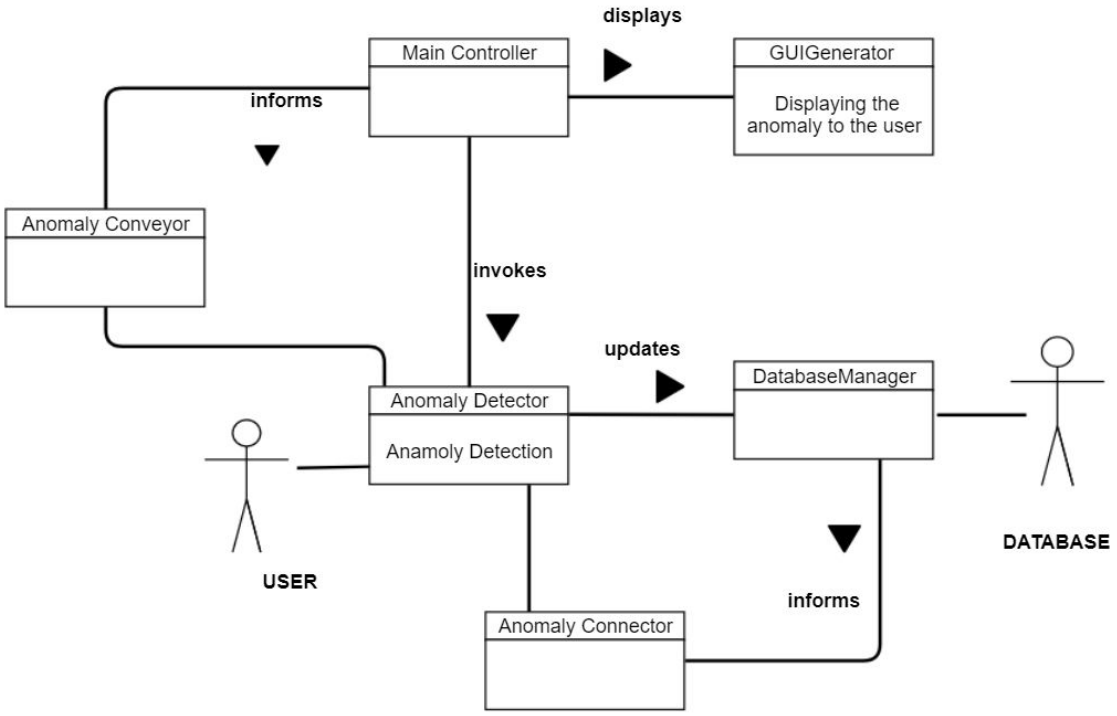


Fig 21: Domain Model: Anomaly Detection

View Statistics:

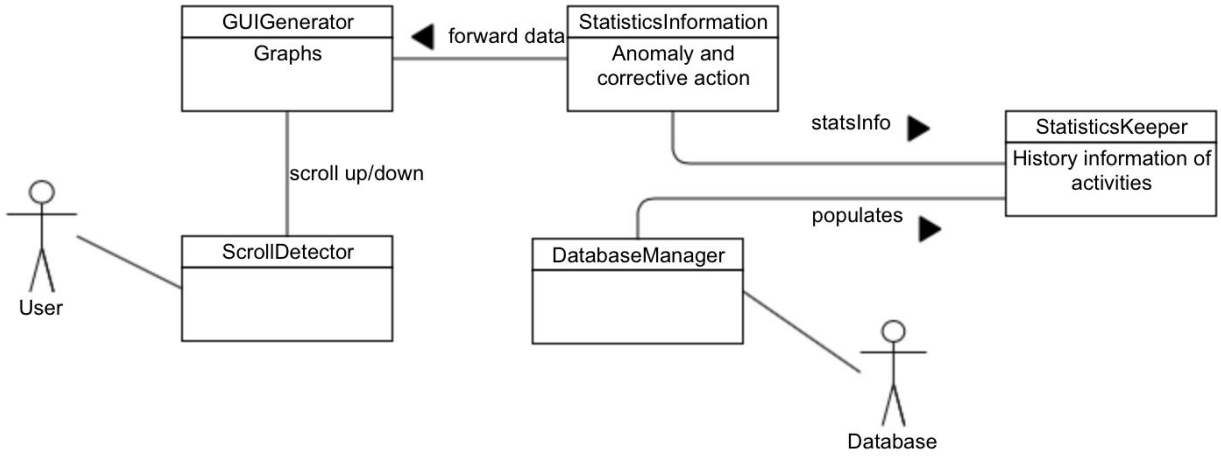


Fig 22: Domain Model: View Statistics

Recommender:

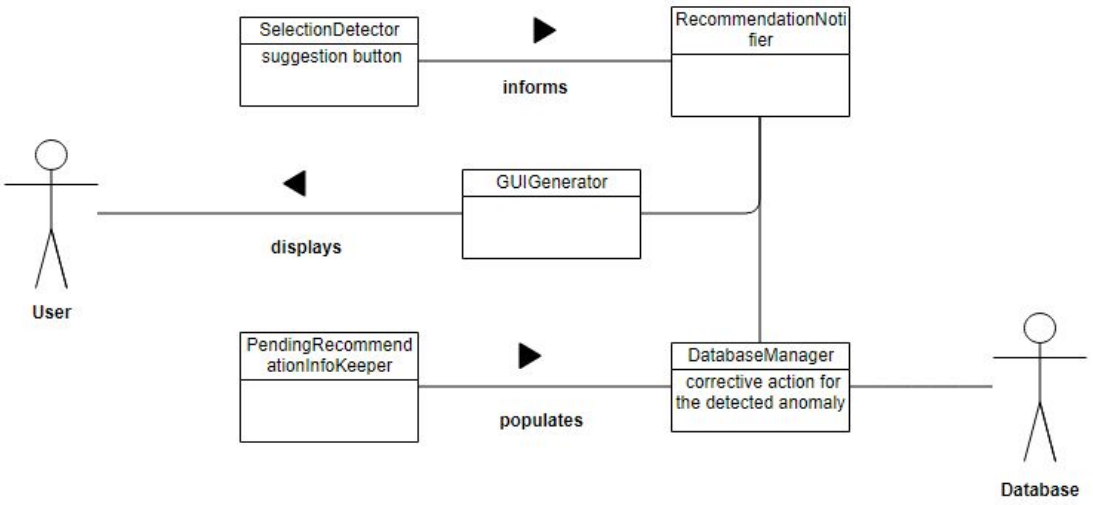


Fig 23: Domain Model: Recommender

Exercise:

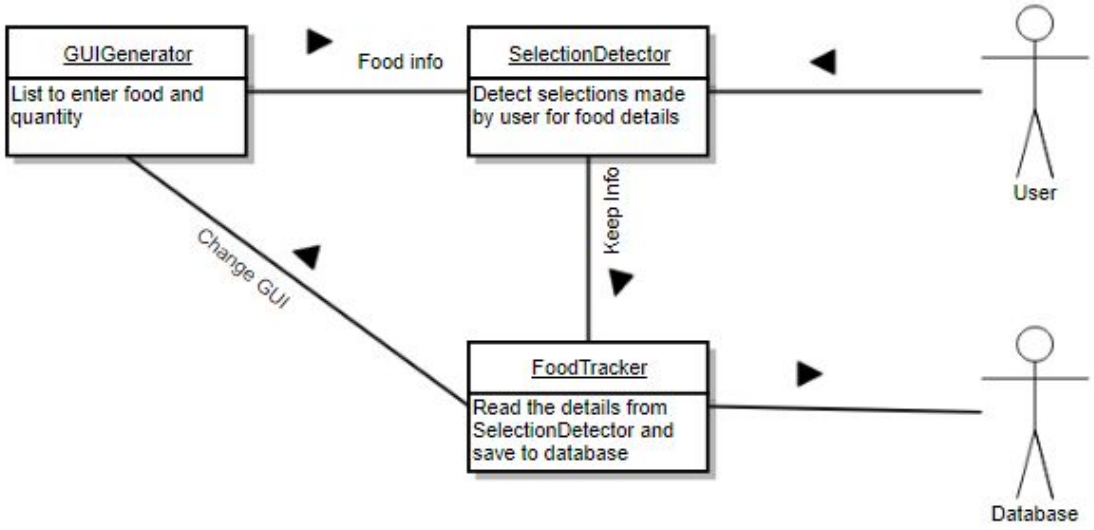


Figure 24: Domain Model: Exercise

6.2 System Operation Contracts

Operations contracts specify any important conditions about the attributes in the domain model.It specify preconditions and postconditions for each attributes.While making an 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. Postcondition can be change in state or database updated.Thus postcondition 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.

Operation	**UC1: Login**
Pre conditions	Numofattempts =0
	Numofattempts > MaxNumOfAttempts
	User_Allowed = false
Post conditions	Numofattempts =0
Alternative	User_Allowed = true

Table 46: System Operation Contracts for Use Case 1: Login

Operation	**UC2: Registration**
Pre conditions	Username = null Password = null
Post conditions	Username = hash(username) Password = bcrypt hash(password)

Table 47: System Operation Contracts for Use Case 2: Registration

Operation	**UC3: Manage Account**
Pre conditions	Currentsettings = {pulled from database} AccountModified = false
Post conditions	Currentsettings = updated from user AccountModified = true
Alternative	Currentsettings = {pulled from database} AccountModified = false

Table 48: System Operation Contracts for Use Case 3: Manage Account

Operation	**UC4: Monitor sleep**
Pre conditions	CurrentSensorReadings = null
Post conditions	CurrentSensorReadings = {sensor readings

Table 49: System Operation Contracts for Use Case 4: Monitor Sleep

Operation	**UC5: Anomaly Detection**
Pre conditions	UserRecords > minNumofRecords AnomalyDetected = false
Post conditions	AnomalyDetected = false
Alternative	AnomalyDetected = true

Table 50: System Operation Contracts for Use Case 5: Anomaly Detection

Operation	**UC6: View Statistics**
Pre conditions	History of all the activities is not empty.
Post conditions	Visualization updated for display.

Table 51: System Operation Contracts for Use Case 6: View Statistics

Operation	**UC7: Recommender**
Pre conditions	suggestion button=”pressed” and recommendations for corresponding anomalies in database.
Post conditions	Recommendation deleted when user acknowledges or reinform user if not acknowledged.

Table 52: System Operation Contracts for Use Case 7: Recommender

Operation	**UC8: Exercise**
Pre conditions	Start Button = ‘pressed’
Post conditions	Stop Button = ‘pressed’ and recorded data sent to database.

Table 55: System Operation Contracts for Use Case 8: Exercise

6.3 Mathematical Model

The project has various 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 calculate the acceleration is to compute the number of steps the user is taking while walking or running. We intend to acquire accelerometer data from the user's phone in crude form. Thus, after computing the steps parameter, we will use the algorithms discussed below in order to get the distance parameter, the speed parameter and the calories parameter.

Step Count parameter:

The number of steps taken by the user can be calculated using an Accelerometer. The basic algorithm to calculate the step count has been discussed below.

```
length = sqrt(x * x + y * y + z * z);

if(length >= 2){

stepCount+ = 1;

}
```

First, we have to make sure we sample the accelerometer frequently enough. Then we're going to have to make sure that we are calculating correctly about what your threshold should be. This is going to require a lot of trial and error. We will graph out what the length is over time and using those values we will get a good threshold value.

Distance parameter:

The distance parameter is basically the distance travelled by the user and it is calculated by the formula:

Distance = number of steps x distance covered per step.

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 five seconds.

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	1.2 × Height

Fig 25: Step calculation

Speed parameter:

The speed can be calculated by:

Speed = Distance / Time

Calorie 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. However, we can estimate it using a conventional approximation. The table below shows a typical relationship between calorie expenditure and running speed.

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

Fig 26: Calories Expended vs. Running Speed

From the table, we get:

Calories (C/kg/h) = 1.25 × running speed (km/h)

However because we want m\s the equation becomes:

Calories (C/kg/h) = 1.25 × speed (m/s) × 3600 /1000 = 4.5 × speed (m/s)

The calories parameter would be updated every 5 second with the distance and speed parameters. So, to account for a given user's weight, we can convert the last equation as follows:

Weight (in kgs) is a user input, and one hour is equal to 720 5-second intervals.

Calories (C/5 s) 4.5 × speed × weight / 5 = 0.9 x (speed × weight)

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 the following equation since the caloric expenditure is around 1 C/kg/hour while resting.

Calories (C/5 s) = 1×weight / 720

Finally, we get the total calories by adding the calories for all 5-second intervals.

Anomaly Detection parameter:

For the anomaly detection, the user's sleep pattern data obtained from the sensor, can be compared with an existing data set. This can be done using a Classification algorithm, where the

user's data is the testing data set. The classifier can be trained using an existing dataset consisting of a set of user data.

Classification is a type of Supervised Learning algorithm. A model is prepared using the training data, which is required for prediction and error correction. The process of prediction and correction is continued till a certain level of accuracy is reached.

One such classification algorithm is Support Vector Machine.

Support Vector Machine (SVM) is a supervised machine learning algorithm which can be used for both classification applications. In this algorithm, we plot sensors' data readings as a point in n-dimensional space (where n is number of features (sensor readings of sleep pattern data) you have) with the value of each feature being the value obtained by the sensor. Then, we perform classification by finding the hyper-plane that differentiate the two classes very well as shown in the figure below.[16] But, the first step in Support Vector Machine classification is to find the best fitting hyper-plane that would give the best accuracy, which can be done by treating the hyper-plane as a convex optimization problem and finding the optimum set.

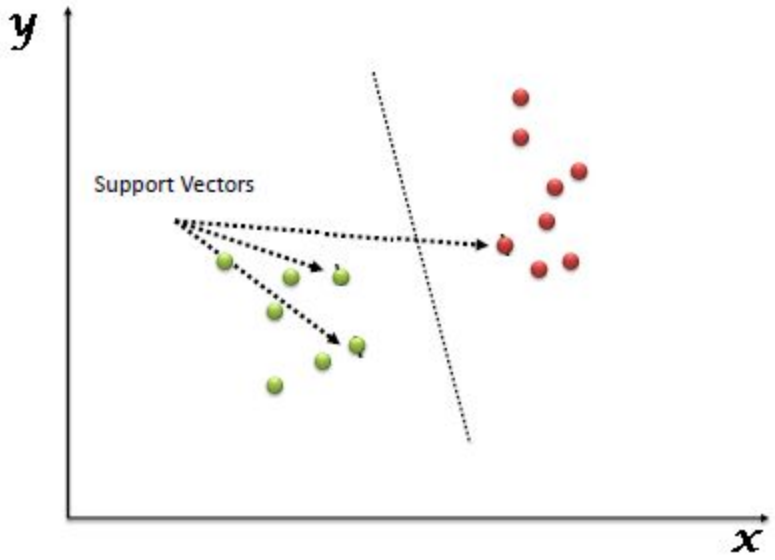


Fig 27: Support Vector Machine Classifier

7. INTERACTION DIAGRAMS

UC 1: Login

The interaction diagram for Use Case 1: Login is shown in the figure above. This use case allows a user with an existing account to log into the system and access functions of the app and the account data. The user initiates an action with the login interface by pressing the login button. The user's data is passed to the controller with a call to verify login. The controller then passes a query request to the database manager. The database manager looks in the database for a for a record of the login. The database may return a record from the database or it may return no results. The database manager will forward the database query results to the controller. The controller will compare the login from the user to the record retrieved from the database. If the records do not match in username and password. The controller will return verify login with a fail to the login interface. The login interface will inform the user that their login attempt was not valid. If the user has already exceeded the max number of login attempts then the user must wait for a 2 min timeout to expire before attempting to log in again.If the user has not exceeded the max allowed number of logins they will be allowed to attempt to log in. If on the other hand the user has entered matching login credentials then the login interface informs the user that their login was successful. The controller will then request GUI Generator to display the next screen to the user.

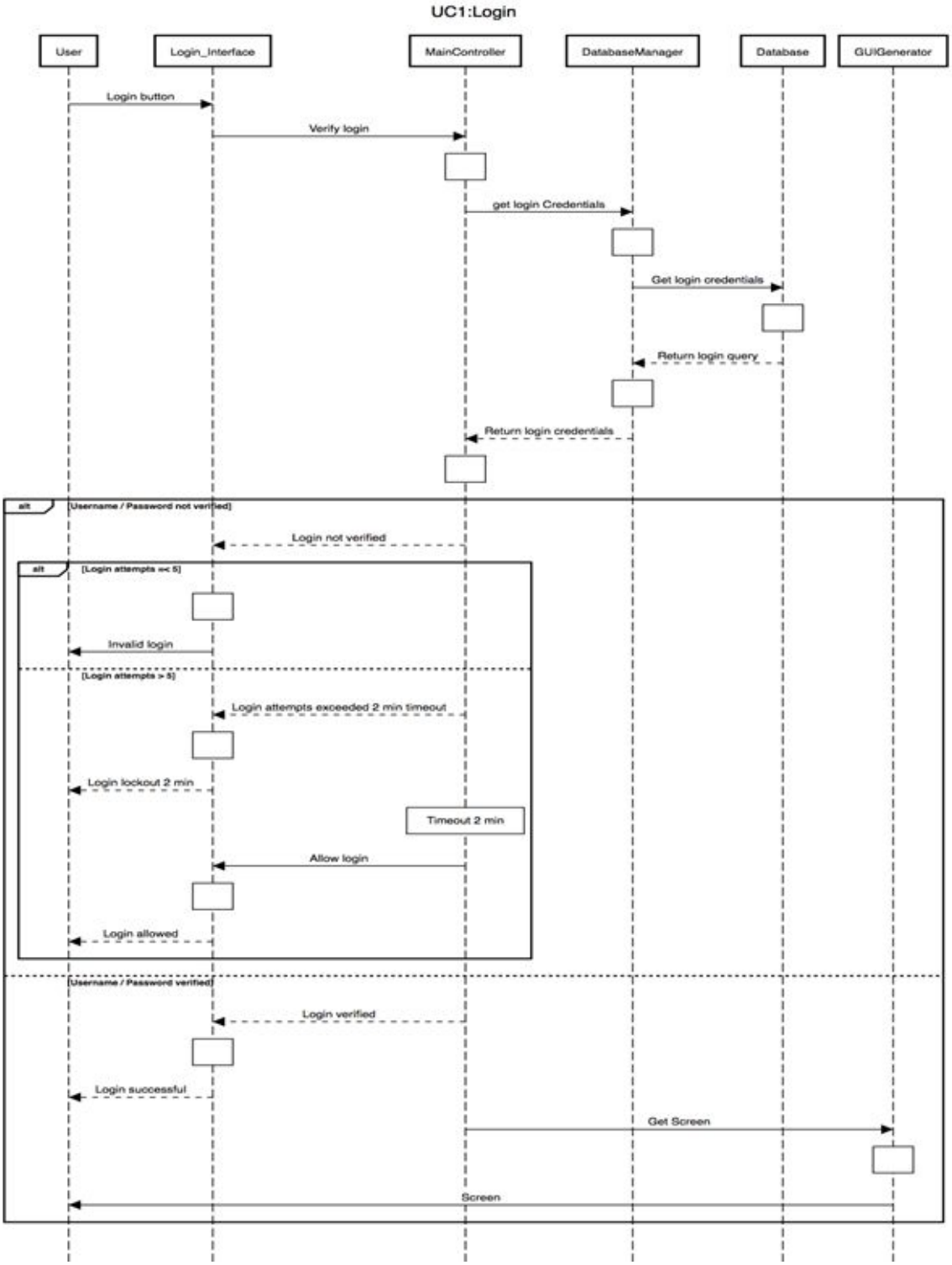
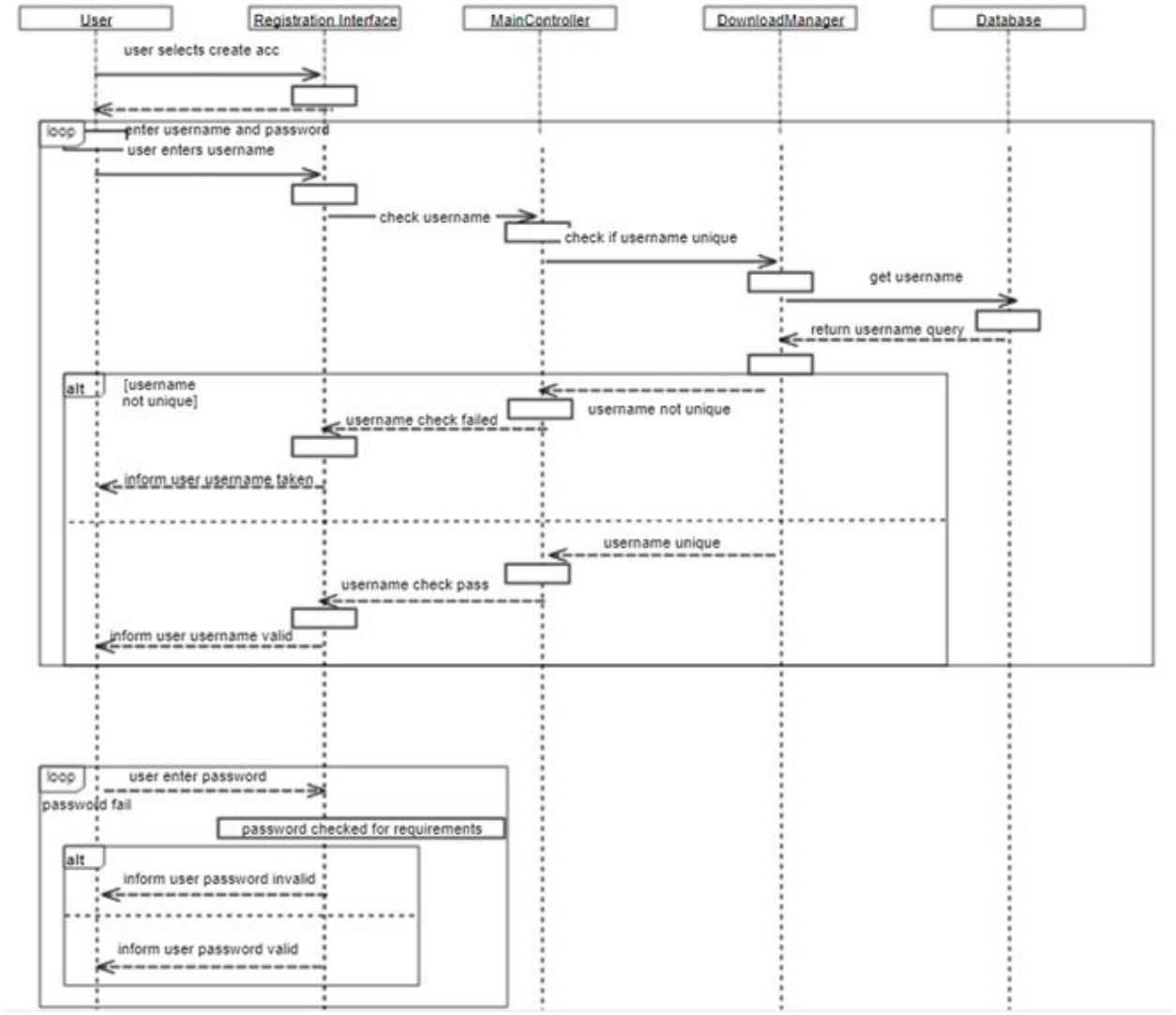


Fig 28: Interaction Diagram for Login

UC 2: Registration



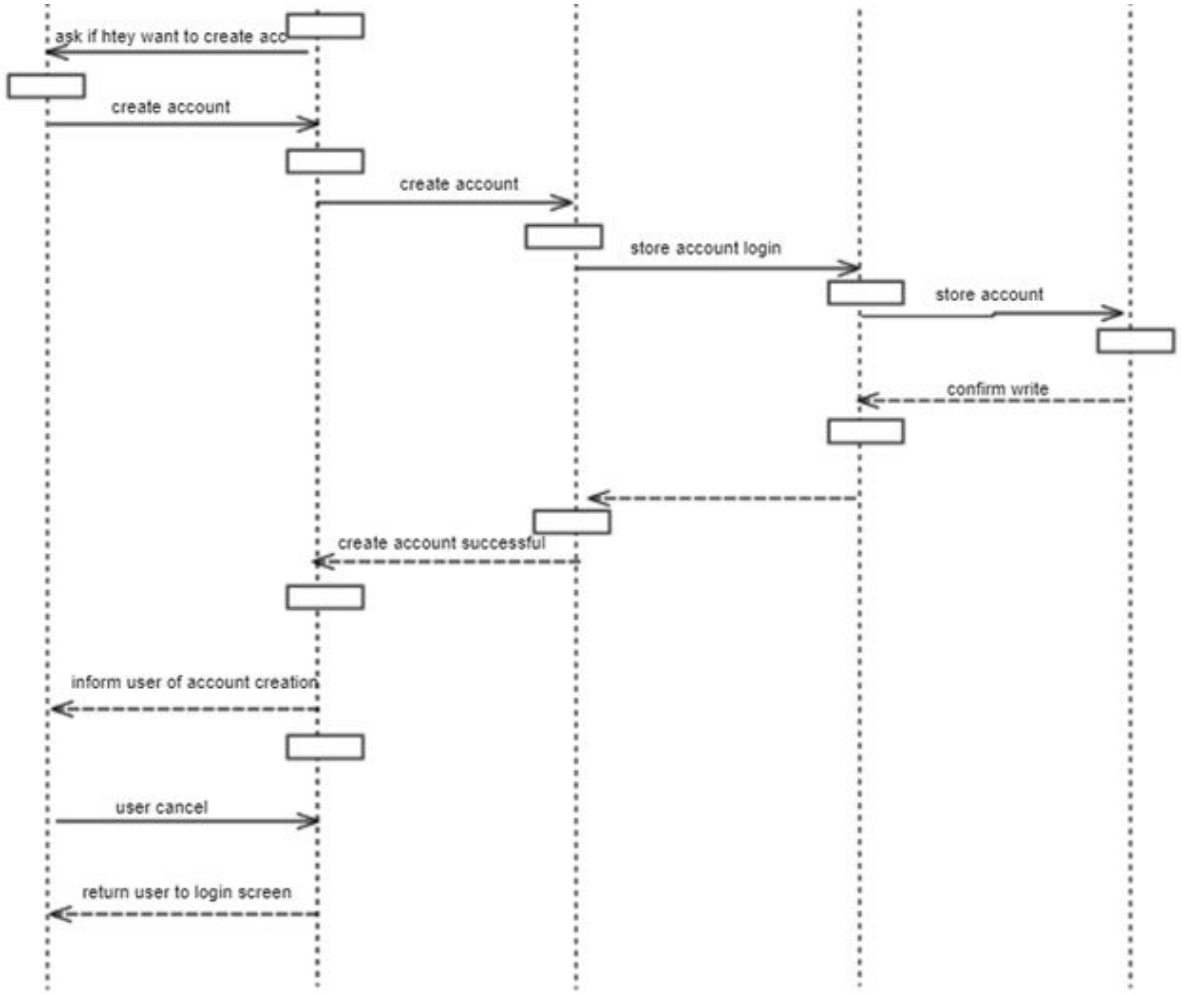


Fig 29: Interaction Diagram for Registration

The interaction diagram for Use Case 2: Registration is shown in the figure above. The user initiates the use case by selecting the create account button. The Registration interface then prompts the user for a new username and password. Once the user enters a username, The registration interface will ask the controller to check for existing usernames. The controller will then ask the database manager to query the database for the username. The database returns the record to the database manager. The database manager returns the query results to the controller. If an existing record was found then the controller tells the registration interface that the username exists and the user cant take it. The registration interface will inform the user. This will

repeat until the user selects a unique username. Otherwise, if the username was unique the user is informed that the username is unique and will be allowed to continue or the user can exit if he wants to. The user will then enter a password. The password will be passed to the controller. The controller will test the password against a regular expression to ensure the password meets the minimum security requirements. If the password does not test then the user will be asked to enter a new password. If the password does pass the test then user is informed. The Registration interface will then prompt the user to create account. If the user elects to create an account then the account details are passed to the controller in a call to create account. The controller will ask database manager to store a new account in the database. The database manager writes the account into the database. The database returns a confirmation to the database manager which then informs the controller of the success. The controller informs registration interface. The registration then informs the user of account creation success. If however the user decides not to create an account then no further action will occur.

UC 3: Manage Accounts

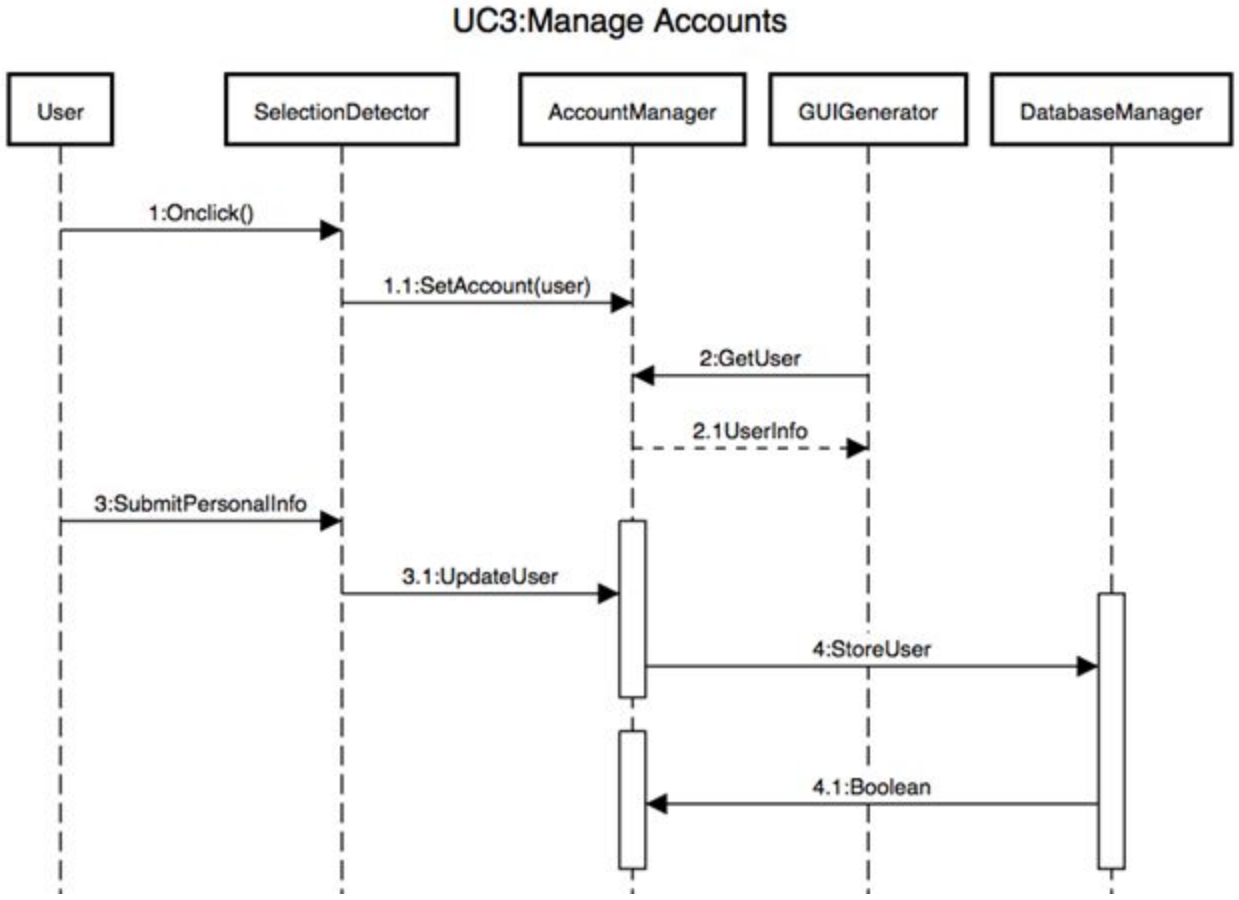


Fig 30: Interaction Diagram for Manage Accounts

The interaction diagram for Use Case 3: Manage Accounts is shown in the figure.The user is the one who initiates the action, by choosing an option out of the ones given on the GUI using the function SelectionDetector, the SelectionDetector then moves to the AccountManager function, where the AccountManager first gets the user's detials needed to sign in and then shows the user's detials on the GUI, which is done by the function GUIGenerator. Now it comes to the user again to chose to submit his/her personal information. The user does by using again, the SelectionDetector on the GUI. This information is then submitted directly to the AccountManager, where the function sends the details to the DatabasManager, where the details that have been entered by the user for their account, gets stored finally.

UC 4: Monitor Sleep

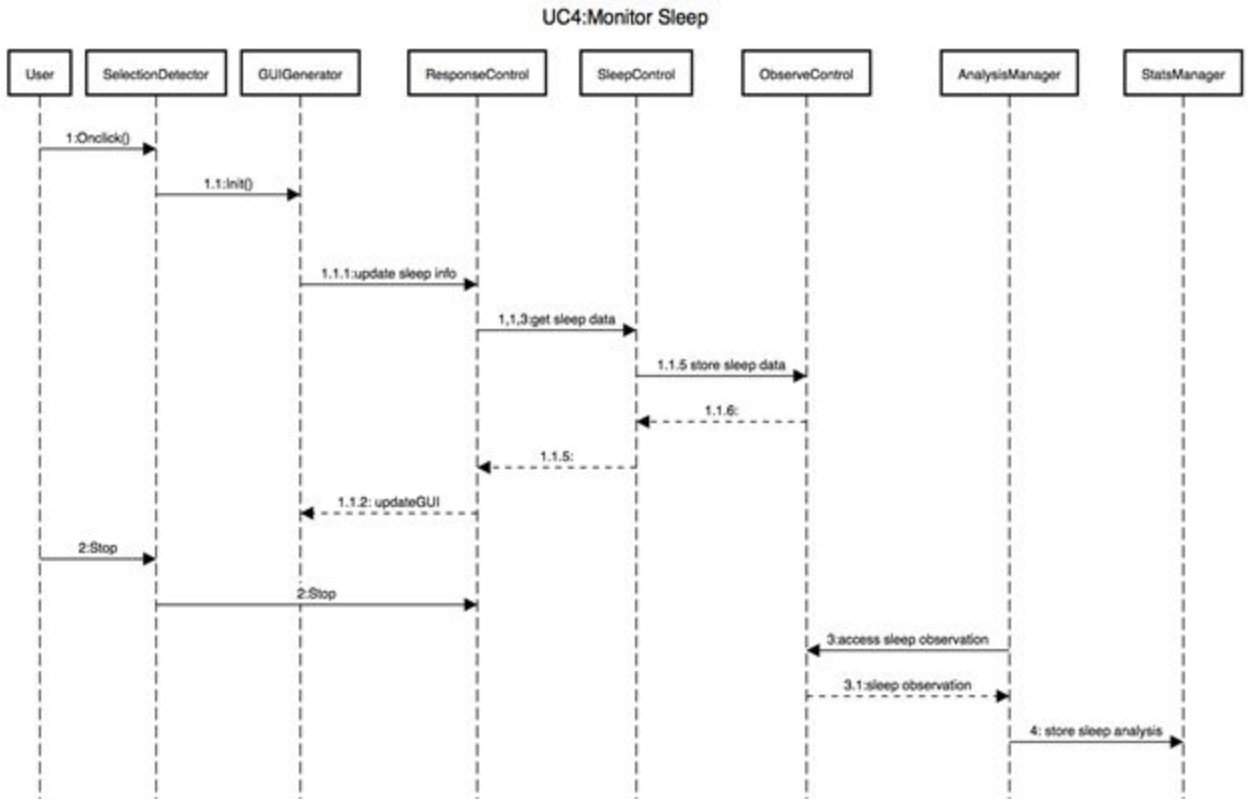


Figure 31: Interaction Diagram for Monitor Sleep

The use case diagram for Use Case 4:Monitor Sleep is shown in the figure.In this use case, the user initiates the function by choosing the option given by the SelectionDetector. If the user selects to monitor sleep, the sleep monitoring starts, and the GUI is updated accordingly via the GUIGenerator. The ResponseControl acts to the input and starts the recording part. It informs the SleepCoontrol function to record the user's sleep, which is done by activating sensors for the operation. The SleepControl records the sleep and updates the ObserveControl function, after which, the GUI is updated, as the process of monitoring keeps going on. This is done until the point the user chooses to stop by choosing the choice given in the GUI by the SelectionDetector. Accordingly, an instruction is sent to the ResponseControl, that stops the recording process. Once it is done, the AnalysisManager accesses the sleep data from the ObserveControl and performs the analysis on the sleep data. This analysis done, is then stored in the StatsManager.

UC 5: Anomaly Detection

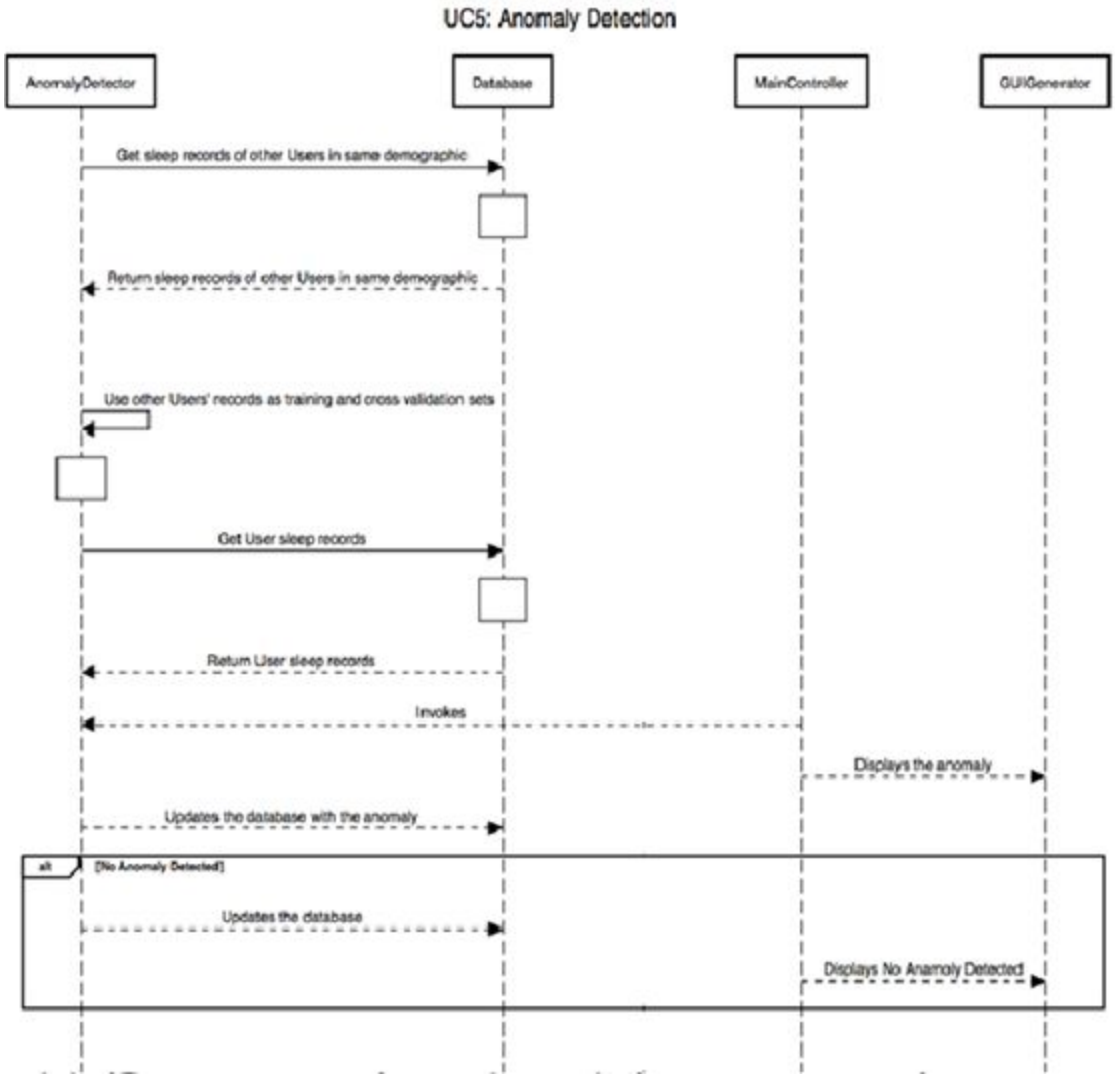


Fig 32: Interaction Diagram for Anomaly Detection

The interaction diagram for Use Case 5: Anomaly Detection is shown in the figure. This use case detects an anomaly based on user's sleeping habits. Initially, the system gets the user's sleep records as well as other user's sleep records in the same demographic. Based on these records, the AnamolyDetector trains and cross validates the data to detect an anomaly. Once an anomaly is detected, it is displayed to the user using GUIGenerator and updated in the database using

DatabaseManager. In an alternate scenario, if AnomalyDetetor detects no anomaly, in a similar manner, the GUIGenerator displays no anomaly detection to the user along with database update using DatabaseManager. The anomaly will be detected once the data is stored in the database.

UC 6: View Statistics

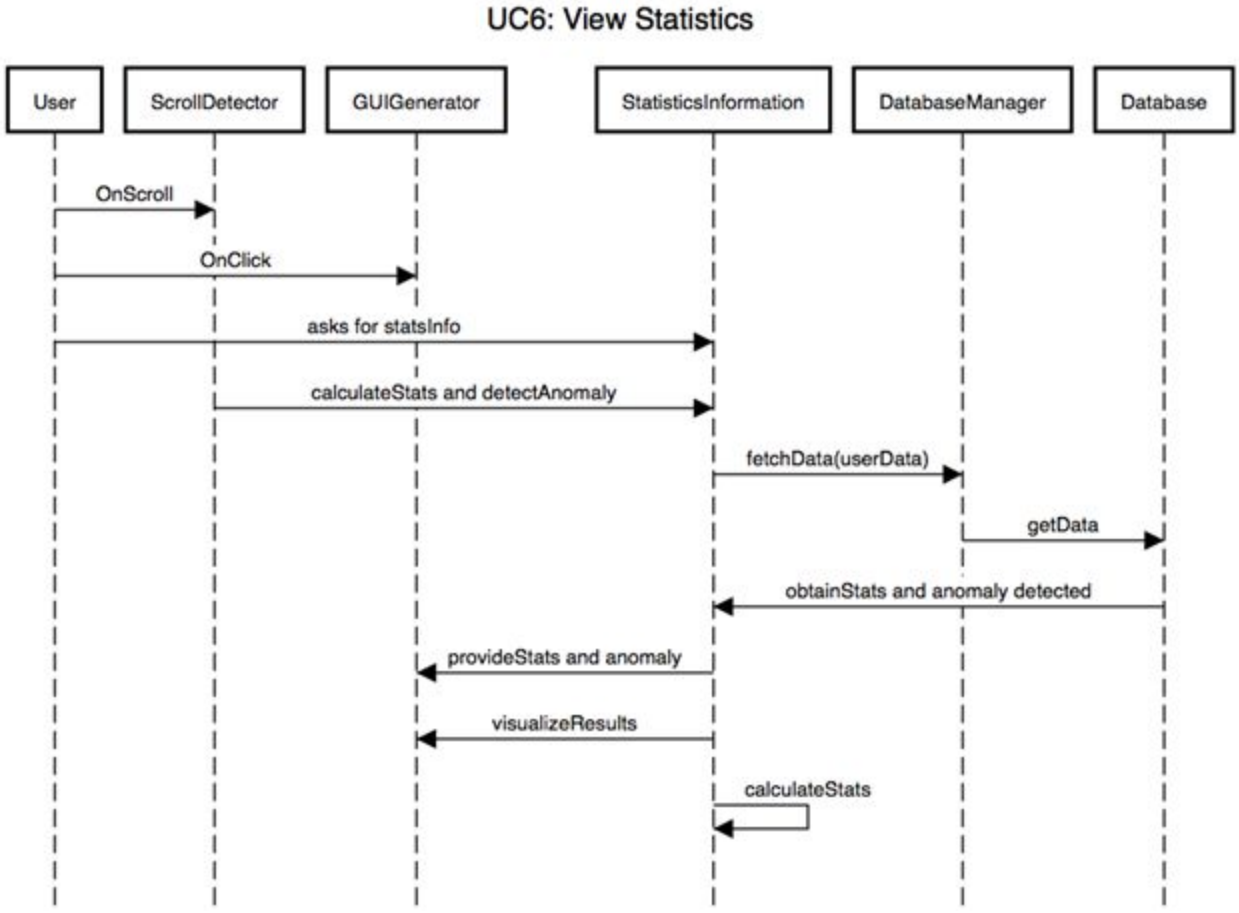


Fig 33: Interaction Diagram for View Statistics

View Statistics is for displaying all the information about the user. After the user clicks on the "View Statistics" button, a page will be displayed showing information from the Sleep, and Exercise data. The "StatisticsInformation" will calculate the calories burnt after Exercise and display it. The data which is used for calculation is fetched from the "Database". Sleep data:

Hours Slept, Anomaly and Suggestions is also displayed. The anomaly detected is obtained from the Database and displayed to the user along with the suggestion. The figure shows the Interaction Diagram for Response to the user when he/she clicks the “View Statistics” button displayed on the GUI.

UC 7: Recommender

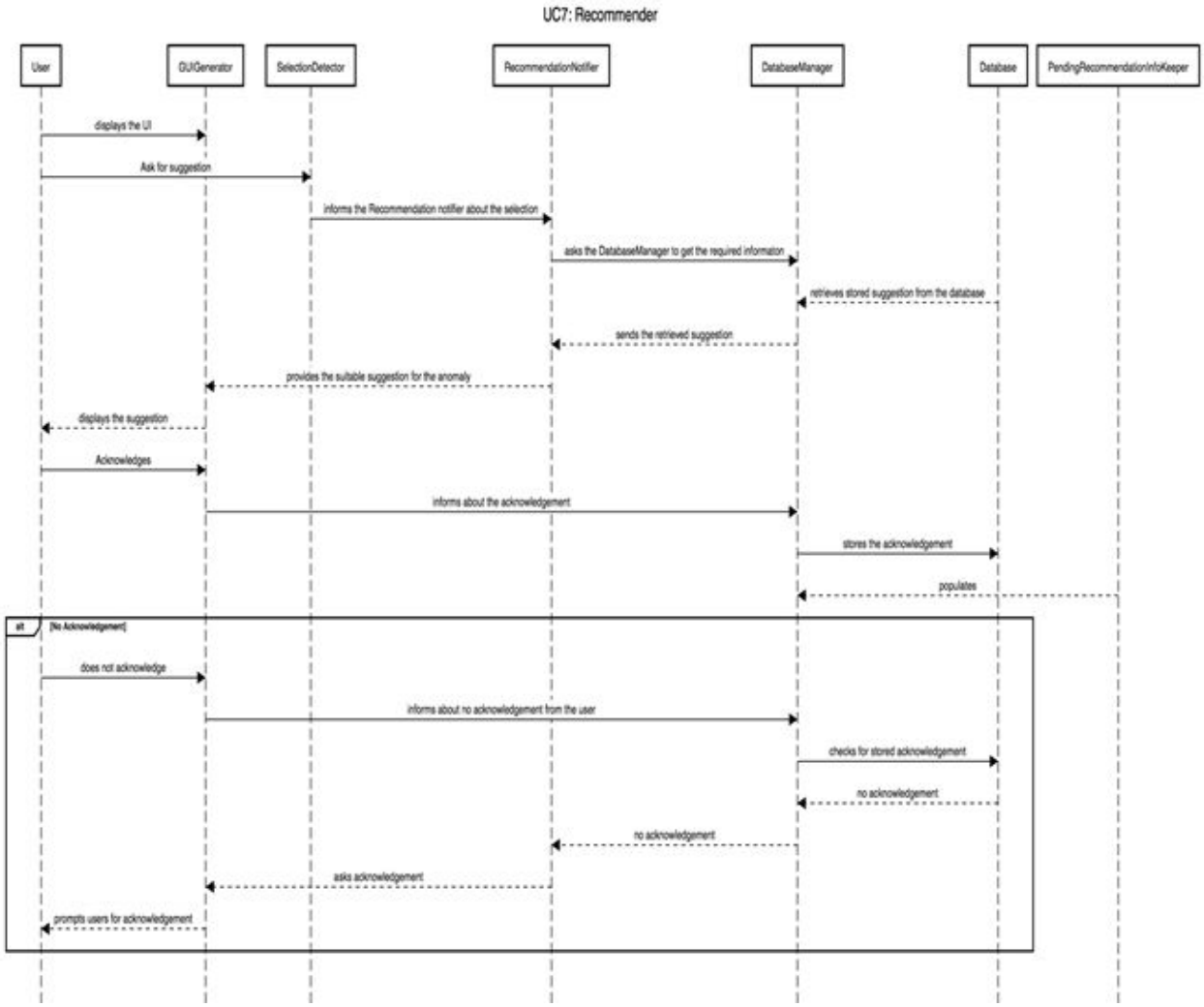


Fig 34: Interaction Diagram for Recommender

The interaction diagram for Use Case 7: Recommender is shown in the figure above. This use case provides the user with certain suggestions pertaining to the detected anomaly. The User asks for the suggestion. The SelectionDetector informs the RecommendationNotifier about the selection. The suggestion for respective anomalies are previously stored in the database. The DatabaseManager retrieves the suggestion from the Database, and provides it to the GUIGenerator, which then displays it on the screen to the User. The User needs to acknowledge

that the has read the suggestion, so that the suggestion does not go unnoticed. If the user acknowledges the suggestion, the GUIGenerator informs the DatabaseManager that the user has acknowledged and read the suggestion, and stores it in the database. In the case that the user does not acknowledge the suggestion, the RecommendationNotifier continuously prompts the user to acknowledge.

UC 8: Exercise

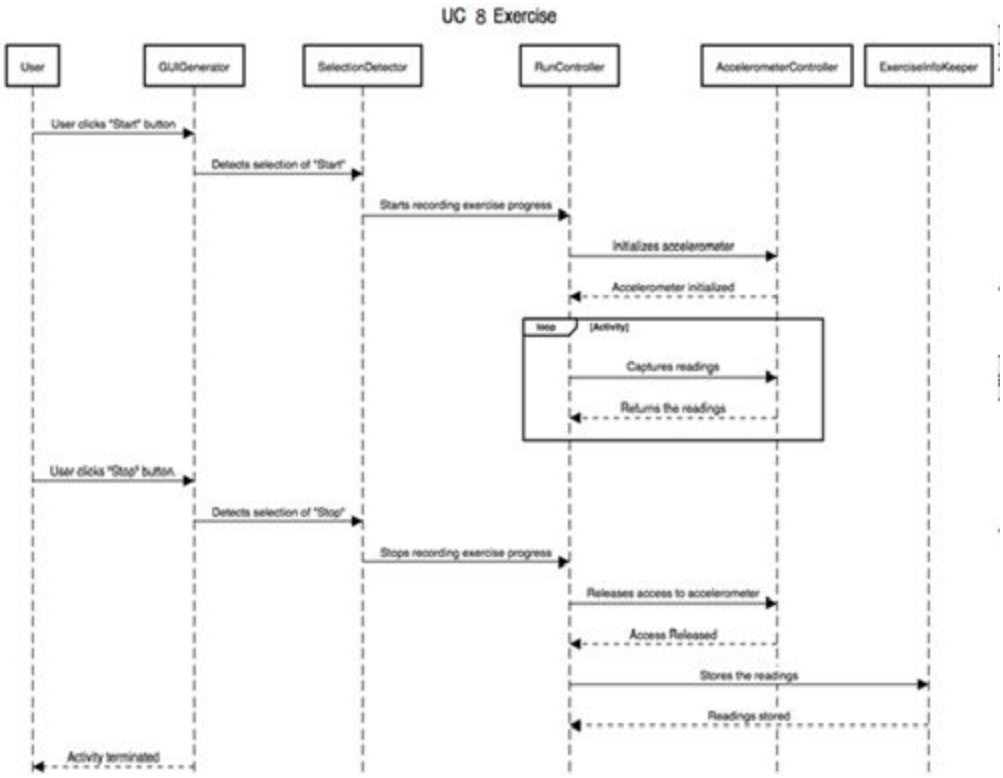


Fig 35: Interaction Diagram for Exercise

According to the interaction diagram above, the user has to first click on the “Start” button in the exercise page (this page will appear if the user clicks on “Exercise” button on the home page). The SelectionDetector automatically senses the touch of the “Start” button and triggers the RunController to automatically record the exercise progress. RunController immediately accesses the accelerometer in the smartphone through the AccelerometerController to measure the miles covered and a timer will be running in the background which will note the time of workout and the approximate calories burnt will be calculated by our algorithm at the end of the workout and will be entered manually by the user. After the user has finished exercising, he/she has to press the “Stop” button in the user interface. Then the SelectionDetector immediately senses the touch of the “Stop” button which in turn makes the RunController to stop recording exercise progress. The RunController will also release access from the accelerometer through the

AccelerometerController. The data collected during the workout will then be stored in the database through ExerciseInfoKeeper.

8. CLASS DIAGRAMS AND INTERFACE SPECIFICATION

8.1 Class Diagrams

In Software Engineering, a class diagram in the Unified Modeling Language(UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among objects.[5] The class diagram in figure 11 below, shows the relationship between the major classes in the application. The diagram contains information about the attributes in the class and the functions. It also shows the flow of events between each class. The classes mentioned in the diagram are coded in Java. Diagram list all the functions, attributes in each class and the details like return type.

1. The part of the application that resides on the mobile application uses components of the Android SDK.

- The Android SDK is used to interface the sensor *accelerometer*.The phone accelerometer data is accessed in application through this android SDK.
- Android also provides access to the GUI components on the phone.

2. The SQLite database has been implemented using the Android API to create the table, update them and retrieve the data from them by querying the database appropriately.

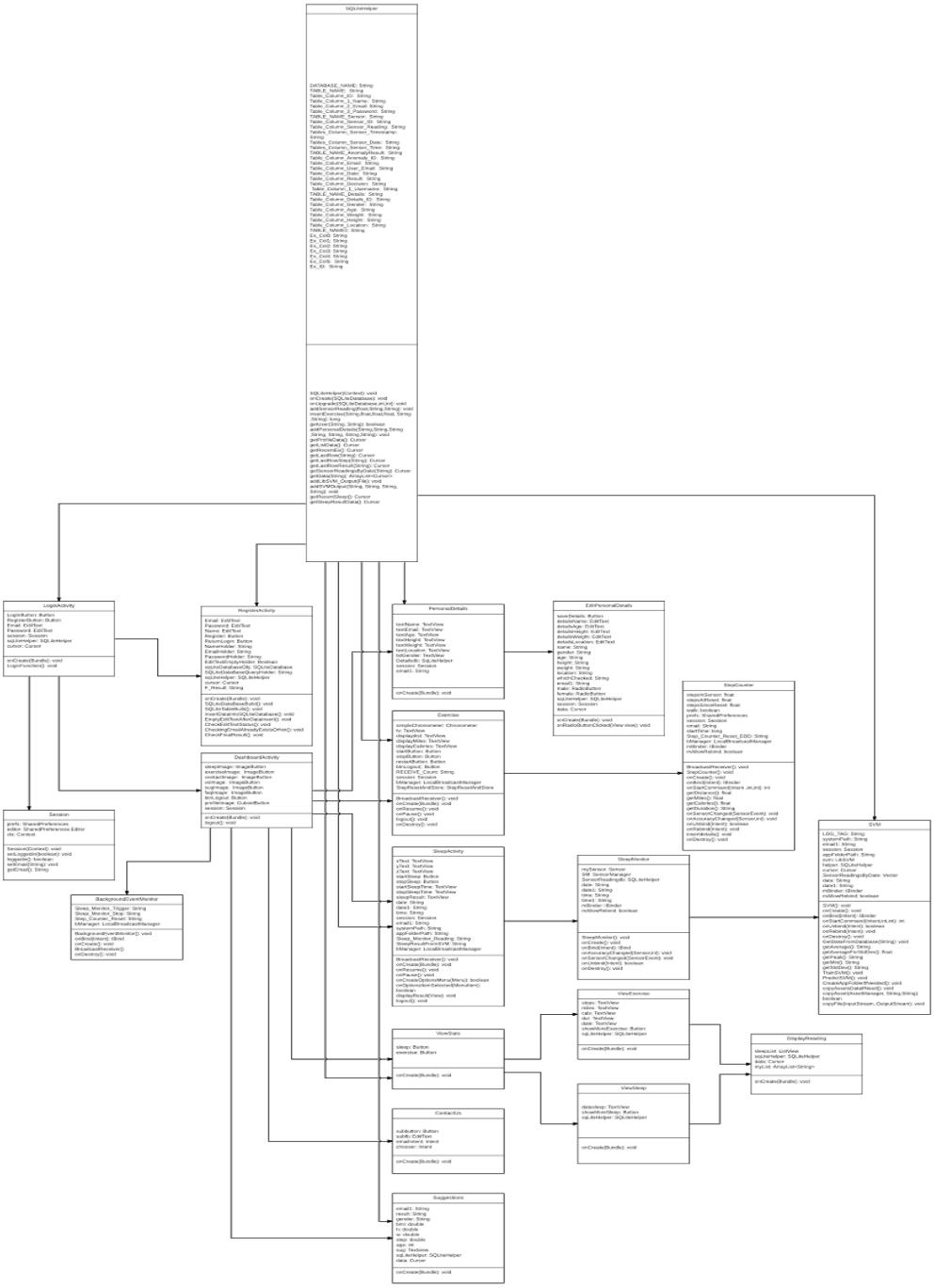


Fig 36 : Class Diagram

8.2 Data types and Operation signatures

The datatypes, operation signatures with the description has been described below for all the classes.

Variable/Method	Description
LoginButton: Button	When clicked, allow the existing user to login to the app
RegisterButton: Button	When clicked, allows a new user to register into the app.
Email: String	Holds the email address of the user
Password: String	Holds the password of the user
Session: Session	Sets the session variable.
sqLiteHelper: SQLiteHelper	Creates an interface to the database
onCreate(Bundle) : void	This function is used to start the app and set up the XML file
LoginFunction():void	This function verifies the credentials of he user

Table 56: LoginActivity

Variable/Method	Description
Email : EditText	Get the email address of the user
Password : EditText	Get the password of the user
Name : EditText	Get the name of the user
Register : Button	When clicked, user is registered in the app.
ReturnLogin: Button	Returns to login page.
sqLiteDatabaseObj : SQLiteDatabase	To access the database
SQLiteDataBaseQueryHolder : String	Stores the SQL queries written by the user
sqLiteHelper : SQLiteHelper	Creates an interface to the database
Cursor: Cursor	Points to each row in table.
onCreate(Bundle) : void	This function is used to start the app and set up the XML file
InsertDataIntoSQLiteDatabase() : void	Inserts details about the new user in the database
EmptyEditTextAfterDataInsert(): void	Clears the edit text boxes after data is submitted.
CheckingEmailAlreadyExistsOrNot(): void	Checks if email id already exists

Table 57: RegisterActivity

Variable/Method	Description
textName : String	Name of the user
textGender: String	Gender of the user
textAge : int	Age of the user
textEmail : String	Email of the user
textHeight : int	Height of the user
textWeight : int	Weight of the user
textLocation: String	Location of the user
Detailsdb: SqLiteHelper	Creates an interface to the database
Session: Session	Enables the user to access session variable.

Table 58 :PersonalDetails

Variable/Method	Description
saveDetails: Button	This button saves all the details entered by user.
detailsName: EditText	Takes in the user's name.
detailsAge: EditText	Takes in the user's age.
detailsHeight: EditText	Takes in the user's height.
detailsWeight: EditText	Takes in the user's weight.
detailsLocation: EditText	Takes in the user's location.
whichChecked: String	If radio button "male" is checked,

	whichChecked is male. Otherwise, whichChecked is female.
male: RadioButton	This is the radio button for gender.
female: RadioButton	This is the radio button for gender.
sqLiteHelper: SQLiteHelper	Creates an interface to the database.
session: Session	Enables the user to access session variable.
data: Cursor	Points to each row in the database.

Table 59: EditPersonalDetails

Variable/Method	Description
Exercise: Button	Enables us to view exercise statistics.
Sleep: Button	Enables us to view sleep statistics.

Table 60: ViewStats

Variable/Method	Description
simpleChronometer: Chronometer	When clicked, it starts taking accelerometer readings
displaydist: TextView	When clicked, it stops taking accelerometer readings
displayMiles: TextView	Displays the duration of exercise to the user
displayCalories: TextView	Displays miles covered by the user
caloriesBurnt: TextView(append)	Displays calories burnt by the user

startButton: Button	Button to start exercise
stopButton: Button	Button to stop recording exercise
restartButton: Button	Reset progress
btnLogout;: Button	Logout of the application
RECEIVE_Count: String	Get the step count
session: Session	Enables the user to access session variable.
StepResetAndStore: StepResetAndStore	Reset and Store step count

Table 61: ExerciseActivity

Variable/Method	**Description**
xText: TextView	Display the movement in x axis
yText: TextView	Display the movement in y axis
zText: TextView	Display the movement in z axis
startSleep: Button	Button to start recording sleep data
stopSleep: Button	Stop recording sleep data
startSleepTme: TextView	View start of sleep time
stopSleepTme: TextView	View end of sleep time

sleepResult:TextView	View results of observed sleep
date: String	String variable for date
date1:String	String variable for date
time:String	String variable for time
session:Session	Enables the user to access session variable.
email1:String	Variable for Email
systemPath: String	Path of the system as string
appFolderPath:String	Path of the application folder
Sleep_Monitor_Reading: String	Get readings from the sleep monitor
SleepResultFromSVM: String	Generate the result string from SVM
onCreate(Bundle): Void	This function is used to start the app and set up the XML file
onResume(): Void	Resume process
onPause(): Void	Pause the process
displayResult(view): Void	Show the results of sleep observed
logout(): void	Logout of the application

Table 62: SleepActivity

Variable/Method	Description
mySensor: Sensor	When clicked, it starts calculating miles covered by the user while exercising
SM:SensorManager	When clicked, it ends calculating miles covered by the user
onCreate(): void	This function is used to start the app and set up the XML file

Table 63:SleepMonitor

Variable/Method	Description
DATABASE_NAME: String	The name of the database.
TABLE_NAME : String	The name of the database table
Table_Column_ID: String	Primary key for user table.
Table_Column_1_Name : String	Column for user names in the user table.
Table_Column_2_Email : String	Column for user's email address in the user table.
Table_Column_3_Password : String	Column for user's password in the user table.
TABLE_NAME_Sensor	Creates a table to store sensor data.
Table_Column_Sensor_ID: String	Primary key for sensor table.
Table_Column_Sensor_Reading: String	Stores the sensor reading in sensor table
Table_Column_Sensor_TimeStamp: String	Stores the timestamp in sensor table.

Table_Column_Sensor_Date: String	Stores the current date in the sensor table.
Table_Column_Sensor_Time: String	Stores the current time in the sensor table.
TABLE_NAME_AnomalyResult: String	Creates a table to store anomaly details.
Table_Column_Anomaly_ID: String	Primary key for anomaly table.
Table_Column_User_Email: String	Stores the user's email in the anomaly table.
Table_Column_Date: String	Stores the current date in the anomaly table.
Table_Column_Result: String	Stores the result in the anomaly table.
Table_Column_Decision: String	Stores the decision in the anomaly table.
TABLE_NAME_details	Creates a table to store personal details of the user.
Table_Column_1_UserName: String	Stores the user's name in the details table.
Table_Column_Gender: String	Stores the gender in the details table.
Table_Column_Age: String	Stores the age in the details table.
Table_Column_Height: String	Stores the height in the details table.
Table_Column_Weight: String	Stores the weight in the details table.
Table_Column_Location: String	Stores the location in the details table.
Table_Name2: String	Creates a table to store exercise data.
Ex_Col0: String	Stores the user's email in the exercise table.
Ex_Col1: String	Stores the user's step count in the exercise table.

Ex_Col2: String	Stores the miles covered in the exercise table.
Ex_Col3: String	Stores the calories burnt in the exercise table.
Ex_Col4: String	Stores the duration of workout in the exercise table.
Ex_Col5: String	Stores the current date in the exercise table.
Ex_ID: String	Primary key for exercise table.
onCreate(SQLiteDatabase) : void	To create database tables
onUpgrade(SQLiteDatabase,int,int) : void	To change the database information
addSensorReading(float,String,String):void	This function is used to insert sensor readings to sensor table.
insertExercise(String,float,float,float,String,String): long	This function is used to insert values into the exercise table.
getUser(String, String): boolean	Used to get the user details from database.
addPersonalDetails(String,String,String ,String, String, String,String): void	Used to add user details to details table.
getProfileData(): Cursor	Used to get user data from details table.
getListData(): Cursor	Used to get exercise data from exercise table.
getRecentEx(): Cursor	Used to get the last row of exercise table.
getLastRow(String): Cursor	Used to get last row of details table.
getLastRowStep(String): Cursor	Used to get last step count in details table.
getLastRowResult(String): Cursor	Used to get last row of anomaly table.

getSensorReadingsByDate(String): Cursor	Used to the sensor readings by date.
getData(String): ArrayList<Cursor>	Used to get data from sensor table.
addLibSVM_Output(File): void	Used to add the output of LibSVM to database.
addSVMOutput(String, String, String, String): void	Used to add output of SVM to anomaly table.
getRecentSleep(): Cursor	Used to get the last row of sensor table.
getSleepResultData(): Cursor	Used to get data from anomaly table.

Table 65: SQLiteHelper

Variable/Method	Description
systemPath: String	Gets the path of the directory.
Session: Session	Enables the user to access session variable.
appFolderPath: String	Returns the path for the folder.
svm:LibSVM	Creates a variable to access the SVM library.
session: Session	Enables the user to access session variable.
data: Cursor	Points to each row in the database.
SensorReadingsByDate:Vector	Creates a vector to store all the sensor readings by date.
SVM(): void	Constructor for SVM class
GetDataFromDatabase(String): void	Gets the data stored in SQLite database.
getAverage(): String	Returns the average value of the sensor readings for that particular date.
getPeak(): String	Returns the maximum value of the sensor readings for that particular date.

getMin(): String	Returns the minimum value of the sensor readings for that particular date.
getStdDev(): String	Returns the standard deviation value of the sensor readings for that particular date.
getAverageForStdDev(): String	Returns the average value for standard deviation of the sensor readings for that particular date.
TrainSVM(): void	Trains the SVM with values.
PredictSVM(): void	Predicts the value for the data-set using SVM algorithm.

Table 66: SVM

Variable/Method	**Description**
email1: String	Variable to store email
Result: String	Variable to store Result
gender: String	Variable for gender
bmi: double	Variable for Body Mass Index
h: double	Variable for height
w: double	Variable for weight
Step : double	Variable for step size
Age: int	Variable for age
suggestion: TextView	Display the generated suggestion
sqliteHelper: SQLiteHelper	Creates an interface to the database
onCreate(Bundle): void	This function is used to start the app and set up the XML file

Table 67: Suggestions

Variable/Method	Description
steps:Textview	Display the steps taken
miles:Textview	Display miles walked
cals:Textview	Display calories burnt
dur:Textview	Display exercise duration
date:Textview	Display date
showMoreExercise: button	Display more exercise details
sqLiteHelper: SQLiteHelper	Creates an interface to the database
onCreate(Bundle): void	This function is used to start the app and set up the XML file

Table 68: View Exercise

Variable/Method	Description
sleepImage: ImageButton	Sleep icon
exerciseImage: ImageButton	Exercise Icon
contactImage: ImageButton	Contact Us Icon
vsImage: ImageButton	View Stats Icon
sugImage: ImageButton	Suggestions Icon
faqImage: ImageButton	FAQ Icon
btnLogout: Button	Logout Button
profileImage: CuboidButton	For profile image
session:Session	Enables the user to access session variable.

Table 69: DashboardActivity

Variable/Method	Description
stepsInSensor: float	Steps stored in sensor
stepsAtReset: float	Steps at reset
stepsSinceReset: float	Steps since reset
Walk: Boolean	Variable for walking
session:Session	Enables the user to access session variable.
Email: String	Variable for email
startTime: Long	Variable for start time
Step_Counter_Reset_EOD: String	Variable for reset step counter
stepCounter(): void	Construtor to the class
onCreate(): void	This function is used to start the app and set up the XML file
getDistance (): float	Get the distance traveled
getMiles (): float	Get the distance in miles
getCalories (): float	Get the calories burnt
getDuration (): float	Get the duration of exercise
Insertdetails(): void	Function to insert details
onDestroy(): void	Destructor to the class

Table 70 :StepCounter

Variable/Method	Description
Sleep_Monitor_Trigger: String	Triggers the sleep monitor.
Sleep_Monitor_Stop: String	Stops the Sleep Monitor.
Step_Counter_Reset: String	Resets the step counter.
Step_Counter_Reset: String	Constructor

Table 71: BackgroundEventMonitor

Variable/Method	Description
prefs: SharedPreferences	Sets shared preferences.
editor: SharedPreferences.Editor	Set shared preferences editor.
ctx: Context	Stores conext.
Session(Context): void	Constructor called.
setLoggedin(boolean): void	Finds out whether the user has logged in.
loggedin(): boolean	Notifies whether the user has lo

Table 72: Session

Variable/Method	Description
sleepList: ListView	View the sleep data
sqLiteHelper: SQLiteHelper	Creates an interface to the database

Table 73: DisplayReading

Variable/Method	Description
subButton: button	Submit user feedback
Subfb:EditText	Text field for user to enter feedback
emailIntent: Intent	Opens email services
chooser : Intent	Chooser to choose one of the email services

Table 74: ContactUs

8.3 Traceability Matrix / Mapping of classes to concepts

	Login Activity	RegisterActivity	Exercise	Sleep Activity	View Stats	Suggestions	SQLiteHelper	StepCounter	Sleep Monitor
GUIGenerator	x	x	x	x	x	x			
Login Interface	x								
RegistrationInterface		x							
Validator	x	x							
DatabaseManager	x	x	x	x	x	x	x		
UserDetailsKeeper									

SelectionDetector	x	x	x	x					
ResponseControl	x	x	x	x					
SleepControl				x					x
ObserveControl				x					x
AnalysisManager				x					x
StatsControl				x					x
AnomalyControl				x					x
AnomalyDetector				x					x
AnomalyConnector				x					x
StatisticsKeeper					x				
StatisticsInformation					x				

RecommendationKeeper						x			
PendingRecommendationInfoKeeper						x			
RunController			x					x	
AccelerometerController			x	x				x	
ExerciseInfoKeeper			x					x	

	ContactUs	ViewExercise	ViewSleep	Dashboard	PersonalDetails	EditPersonalDetails	SVM	DisplayReading
GUIGenerator	x	x	x	x	x	x	x	x
LoginInterface								
RegistrationInt								

erface								
Validator								
DatabaseManager		x	x		x		x	x
UserDetailsKeeper								
SelectionDetector				x	x	x		
ResponseControl				x	x	x		
SleepControl								
ObserveControl								
AnalysisManager								
StatsControl								
AnomalyControl							x	
AnomalyDetector							x	
Anoma							x	

lyConnector								
StatisticsKeeper								
StatisticsInformation								
RecommendationKeeper								
PendingRecommendationInfoKeeper								

8.4 Design Patterns

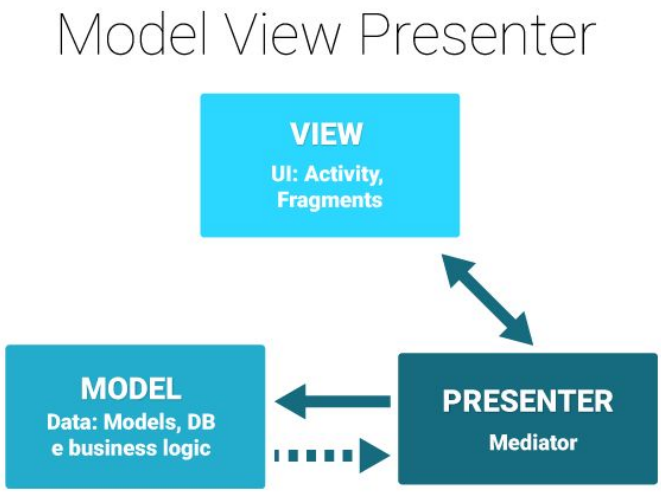


Fig 36: Model View Presenter[27]

MVP design pattern separates the application into three layers: Model, View and Presenter. Each one has its responsibilities and communication between the different layers is controlled by the Presenter. The user can interact with the system through the View, which is basically a user interface such as an Activity or Fragment. The Presenter retrieves data from the Model and shows it in the View. It also processes user actions forwarded to it by the View.

In MVP, the View is not directly triggered by the Controller. Instead the user provides input gestures to the view. The View translates the input into an event that is sent to the Presenter. The Presenter coordinates the calls to change the model. The Model changes and returns its values to the Presenter. The Presenter returns the values to the View and the View decides how to display data. The Interactor resolves how different events map to changes in data. These events consist of the user-initiated actions like mouse movements and clicks, keyboard keystrokes or some touch gestures.

In our application, since there are a lot of touch interactions like menu selections, button and checkbox clicks, different events needs to be mapped to different data changes. Thus, the Presenter decides how the data is changed and how it should be specified. The Presenter interprets the events and gestures initiated by the user and provide the business logic that maps them onto the appropriate commands.

8.5 Object Constraint Language (OCL) Contracts

context PersonalDetails

inv : self.weight >=0 and self.height >=0 and self.age >= 0

context Exercise

inv : self.steps>=0 and self.miles >=0 and self.cals >= 0 and self.dur>=0

context StepCounter
inv: self.stepCount >= 0

context LoginActivity

inv: LoginActivity.SQLiteHelper.isConnected()==true

context RegisterActivity

inv: if self.CurrentPerson<> null then

self.CurrentPerson.getDisplayName()<> null and

self.CurrentPerson.getPassword()<> null and

self.CurrentPerson.getEmail()<> null endif

context SVM

inv: SVM.SQLiteHelper.isConnected()==true

Inv: fromDate<=toDate

context Suggestions

inv: self.bmi >= 0 and self.stepCount >=0

Inv: .SQLiteHelper.isConnected()==true

context ViewStatistics

inv: fromDate <= toDate

9. SYSTEM ARCHITECTURE AND SYSTEM DESIGN

9.1 Architectural Styles

The application uses a client-server model. In client-server architecture, one or more clients can communicate with the server, in a request - response pattern. The client sends a request and the server responds back with the requested information. Initially the user logs into the application, by entering his login details (email id and password). These entered details are verified with the details stored in database, when the user registered into the application. Once the user is logged into the system, they can request to view statistics, to get the amount of calories consumed or burnt, to start monitoring of their sleep or start the running/ walking activity. Here, the user using the application is the client. The server is the application on the back end which responds to the request by displaying the complete statistics of the user, calculating and displaying the consumed/ burnt calories, activating the sensor and starting the monitoring of sleep or tracking the walk/ run respectively. The user's sleep and activity records are also stored in the database. Also, the sleep anomaly is detected by retrieving the sleep data from the database and the system provides corresponding suggestions to the user. Thus, there is a need of constant connection between the user and the backend application, which can be the system, the database or any external API. The client/ server architectural style provides this kind of relationship between the client and the server. In this way, the workflow is divided between various subsystems of the application, which are Accelerometer, Mobile Interface, Server and Database. The major advantage of this type of architecture would be centralization of the data storage, ensuring security (other users cannot view the user's data), better management, scalability and accessibility.

9.2 Identifying Subsystems

The various subsystems of the application are Accelerometer, Mobile Interface, Server and Database.

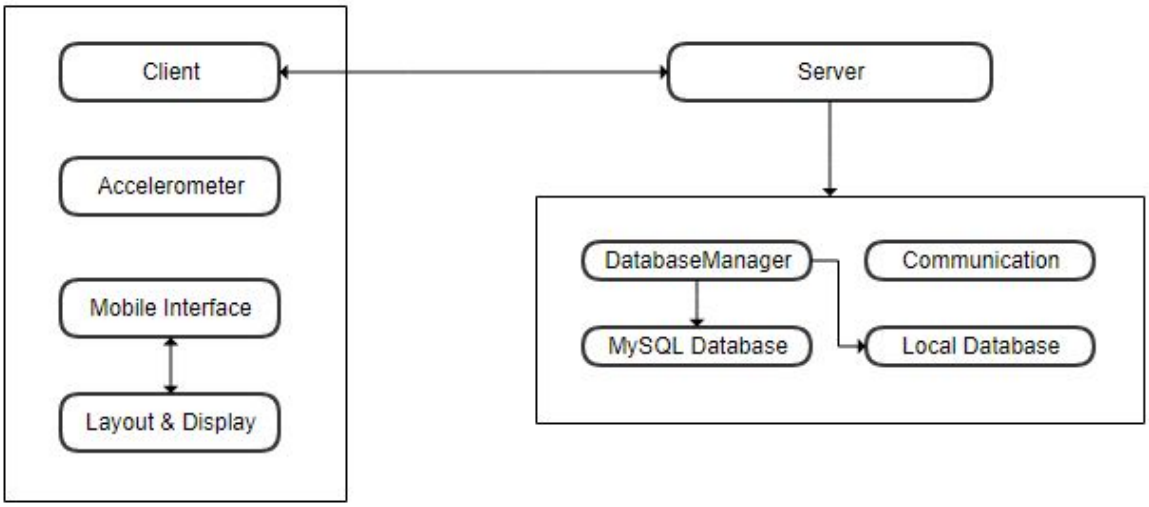


Figure 37: Subsystems of napIT

9.3 Mapping subsystems to hardware

Client: The client is the user's account in the Android application. Thus, the hardware used is the Android Smartphone, on which the application runs.

Server: The server resides on the desktop computer or a laptop of the administrators.

Database: The database used is the SQLite, for which the hardware used is the local computer or laptop of the administrators.

Accelerometer: The device accelerometer is used as the hardware for calculations requiring accelerometer data.

9.4 Persistent Data Storage

The database we are using is MySQL, but since an Android application does not get access to MySQL, our database is built on SQLite. SQLite is an in-process library that implements a self-contained, serverless, zero-configuration, transactional SQL database engine. When the user registers into the application, their login details (email id and password) are stored into the database. Further when the user uses this detail to log in to the application, they are verified against those stored in the database. Apart from authentication, various components of the application require to store and retrieve data from the database. Thus, the data is separated into certain parts: User, Personal Details, SensorReadings, ExerciseDetails and SleepDetails. They are stored and are persistent on the Android device.

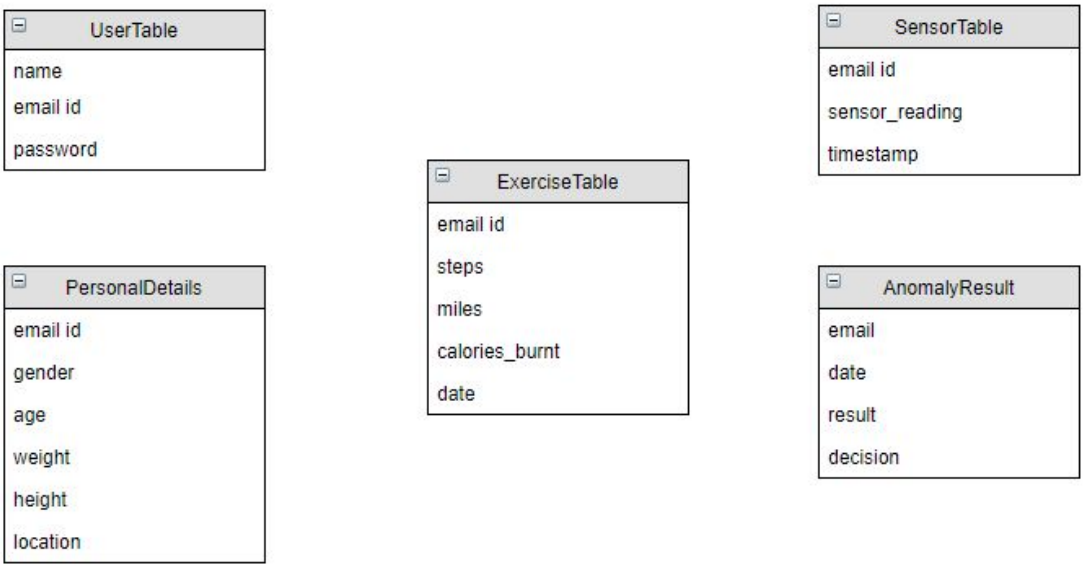


Figure 38: Database Schema

9.5 Network Protocol

Network protocols are formal standards and policies comprised of rules, procedures and formats that define communication between two or more devices over a network. Network protocols govern the end-to-end processes of timely, secure and managed data or network communication. Network protocols include mechanisms for devices to identify and make connections with each other, as well as formatting rules that specify how data is packaged into messages sent and received. Some protocols also support message acknowledgment and data compression designed for reliable and/or high-performance network communication.

Here we have different components of the application which use different network protocols. To accomplish various functionalities in the application, the client has to connect to the server using HTTP protocol. To store and retrieve the data from the database, the system communicates with the SQLite database using queries to create, update and retrieve data. Also, protocols will be required to connect to the Android sensors, through the Android SDK.

9.6 Global Control Flow

1. *Execution orderness:* The system is majorly event-driven, which means it waits in a loop for events. Our application is highly dependent on the user interactions, thus all changes depend on user selection.

9.7 Hardware Requirements

The application requires a Smartphone running Android 6.0 platform (Marshmallow) or higher. The smartphone should have a built-in Accelerometer, so that the steps of the user can be retrieved through the Android SDK. The screen resolution should be high around 480x854 pixels. The hard drive storage should be greater than 1GB. The smartphone should have constant access to the internet with minimum network bandwidth of 3.1Mbps. MySQL database is to be used.

10. ALGORITHM AND DATA STRUCTURES

10.1 Algorithm

Our application "napIT" has features like Exercise and Sleep monitoring. Every feature requires some form of algorithm so that it could work efficiently. For example, the Exercise feature should be able to give the user the distance covered, at what speed and time thus giving out the calories burnt. Sleep monitoring requires sensor readings and to detect whether the sleep is proper or improper, the readings (after analysis and feature extraction) need to be fed into the SVM as input and give out the desired results.

Distance parameter

This parameter will give the distance travelled by the user using the following formula:

Distance= number of steps x distance per step (1)

The distance per step depends upon the speed and height of the 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 2 secs. We use the steps counted in every 2 secs to judge the current stride length.

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	1.2 x Height

Table 75: Stride as a Function of Speed and Height

Speed parameter

Our application also displays the speed to the user along with the distance. The formula for speed is given by:

Speed= distance x time (2)

Our system will calculate speed as:

Speed= steps per 2 s x stride/2 s (3)

Calorie parameter

When it comes to Exercising, the most important parameter that the user is interested in is the amount of calories burnt. Though, 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 thus estimate it using a conventional approximation. The following table shows a typical relationship between calorie expenditure and running speed.

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

16	20
20	25

Table 76: Calories Expended vs. Running

From the above table, we have,

Calories (C/kg/h)= 1.25 x running speed (km/h) (4)

Converting the unit of speed from km/h to m/s, we get,

Calories (C/kg/h)= 1.25 x running speed (m/s) x 3600/1000 =4.5 x speed (m/s) (5)

The calories parameter would be updated every 2 secs with the distance and speed parameters. So, to account for a given athlete's weight, we can convert Equation 5 to Equation 6 as indicated. Weight (kg) is a user input, and one hour is equal to 1800 two- second intervals.

Speed = distance/time, so Equation 3 can be used to get the speed parameter, as steps per 2 s and stride

Calories (C/2 s)= 4.5 x speed x weight/1800 =speed x weight/400 (6)

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 7 since the caloric expenditure is around 1 C/kg/hour while resting.

Calories (C/2 s) = 1 × weight/1800 (7)

Finally, we can add calories for all 2-second intervals together to get the total calories expended.

Sleep Analysis using Support Vector Machine Classifier

The main aim of our application "napIT" is to monitor sleep and perform sleep analysis. We plan to achieve this using a Support Vector Machine classifier. Support Vector Machine is a supervised machine learning algorithm that analyze data used for classification and regression

analysis. Given a set of training example, each marked as belonging to one or the other of the two categories, the SVM will build a model that assigns new examples to one category or the other.

Our application aims to classify the user's sleep into two categories: Proper sleep or Improper Sleep. Thus, SVM classifier suits or needs perfectly. The main components that need to be taken into consideration while doing sleep analysis using SVM are:

1. Training Data
2. Feature Extraction
3. Test Data

Training Data:

The training data was obtained from real sources (i.e. real people) and a graph was plotted and compared to differentiate them into the 2 categories: Proper or Improper Sleep. Enough cases whether considered, so that there is almost equal amount of data for both the categories.

Figure 14: Regular Sleep(Left) and Irregular Sleep(Right)[26]

The above figures show the general trend in people's sleeping pattern. The graph on the left having lesser amount of movement (roughly one peak every 90 min) is that of a person's regular sleep, while that on the right with comparatively lot of movement is that of people's irregular sleep. We have collected real data from our own sources (i.e. family and our friends), and compared the graph and the values with the above graphs and classified them into the two

categories. This data was then used to train the model, with Proper Sleep being "+1" and Improper Sleep being "-1".

Feature Extraction:

Sleep analysis is basically done by tracking the user's movement during his sleep. For this we have recorded the user's readings during his sleep, using accelerometer. During our testing stages, we realized that while sleeping the "x" and "y" values of the accelerometer are not changing much compared to the "z" value. Thus, only it was decided to consider only the z-value for the analysis.

Now, since the readings are in the form of a time-series data (a signal), thus feature extraction was necessary. Feature extraction is an important aspect of any machine learning algorithm. Using the right features is very important to get a correct output. Earlier the features that we had decided to consider was 1. The average of the z- values. 2. The highest z-value and 3. The lowest z-value. But, while testing it was observed that these three features were not sufficient. There was no significant difference between the average, maximum and minimum values for the two cases: Improper and Proper sleep, but there was a significant difference in the range they would fall in. The difference between the average, maximum and minimum was less in case of Proper sleep, and comparatively high for Improper sleep. Thus, a fourth feature: Standard Deviation was taken into consideration.

Thus, the features used for our SVM classifier were Average, Maximum, Minimum and Standard Deviation.

Test Data:

The test data is the user's own data, which is stored in the database when the user's clicks on Start and Stop. This data is collected and used on the model which was built previously using the training data. The test data contains the four features as well. The SVM uses the built model to

classify it as either Proper Sleep (+1) or Improper Sleep(-1) and the result is displayed on the screen.

10.2 Data Structures

The database that we are using to store and retrieve the data is SQLite. The SQLite database will roughly use the following data types:

Data types: The major data types that would be required are integer and string.

1. Sensor Readings (Int): The accelerometer readings will be stored in the data, for every user.
2. User Identification number (user_ID) (Int): It will be auto numbered and auto incremented, and will be used to defined every user.
3. Name, username, email id, location (string): The name, username, email id, location will store the following personal details of the user.
4. Time, miles, calories_burnt, calories_gained (Int): It will be used to store the exercise and food related details of the user.
5. View statistics: An ArrayList is used to store all the exercise and sleep related information, and then display them on the screen.

11. USER INTERFACE DESIGN AND IMPLEMENTATION

The user interface (UI), in the industrial design field of human–computer 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 operator's' decision-making process. In the instance of mobile application, it is in the form of a Graphical User Interface based on Android SDK. To improve the user experience the following changes were made in the user interfaces.

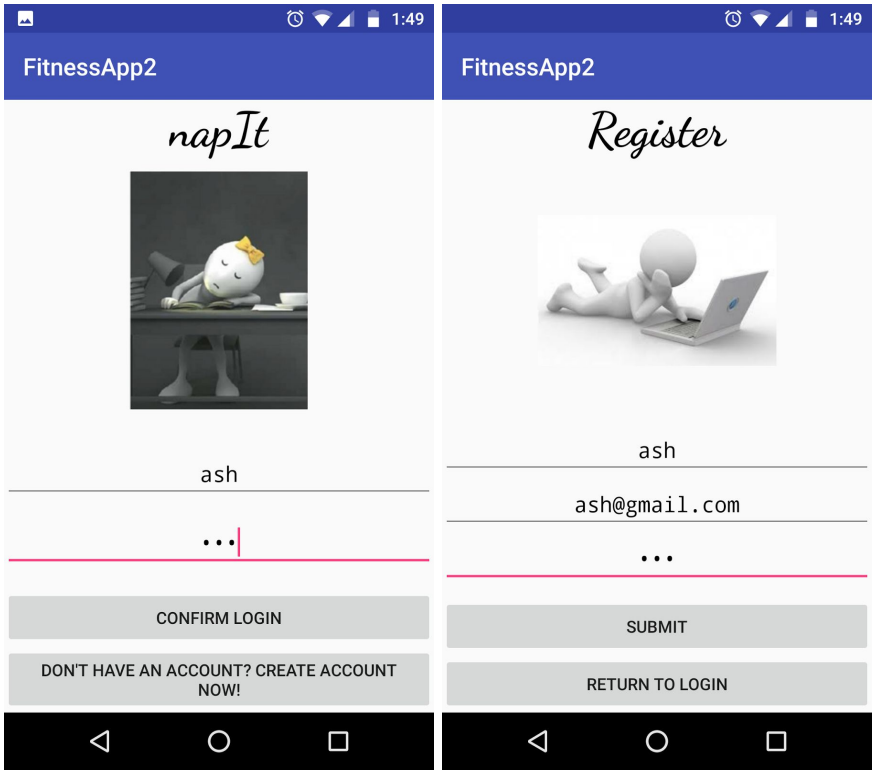


Fig 39: Login Page(Left) and Register Page (Right)

- **Login:** The login page is the main page that opens when the application is started. The user can enter his email id and the password, which he has used to register for the

app, to login. Otherwise, if the user is a new user, he can click on register to create an account.

- **Register:** The register page allows the user to create a new account, by entering his name, email id and password. The user credentials used for registering are stored in the database, and used for authentication and session management.

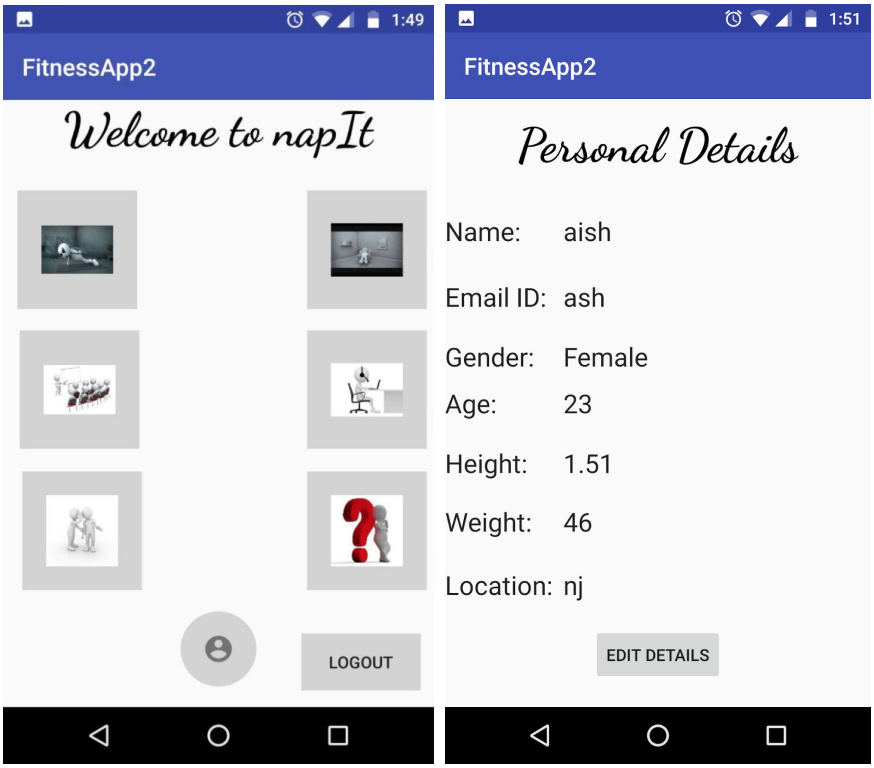


Fig 40: Dashboard (Left) and Manage Accounts(Right)

- **Dashboard**: Once the user logs in, he is directed to a dashboard. From the dashboard, the user can jump to any other page

- **Manage Account:** The user can manage his/her account by clicking on the image button at the bottom of the application. Here, the user can enter his personal details such as name, email id, age, gender, weight, height and location.

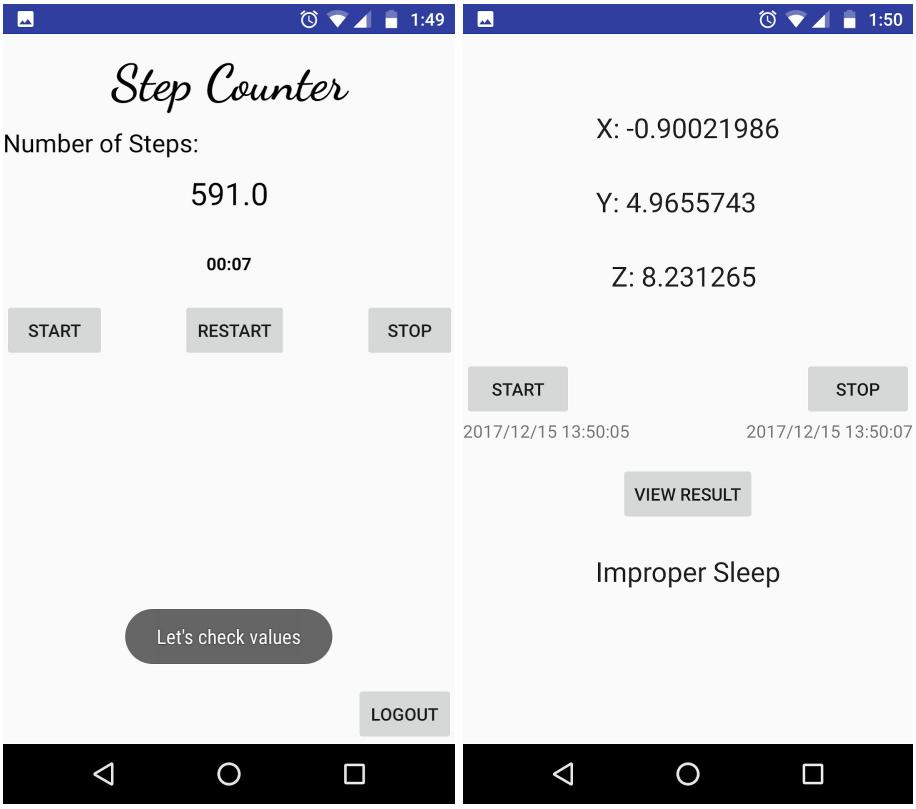


Fig 41: Exercise Page(Left) and Sleep Page(Right)

- **Exercise:** The exercise page allows the user to start and stop his activity. The step counter gets updated simultaneously as the user walks or runs. Additionally, now he can also restart the step counter if he wants using the restart button. The sensor is also ran in the background, so as to minimize the user effort.

- **Sleep:** Similarly, the sleep page monitors the user's sleep when the user starts, and stops monitoring when he clicks on stops. The user can also view his result with just one click, by clicking on "View Results".

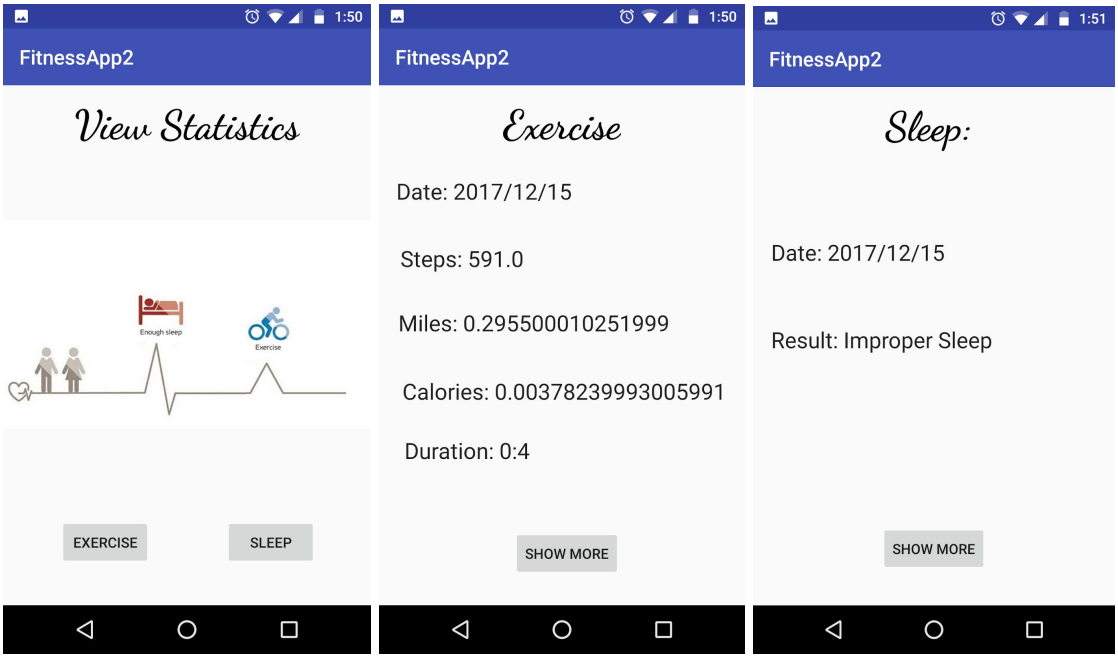


Fig 43: View Statistics- Main page(Left), Exercise (Middle) and Sleep(Right)

- **View Statistics:** View Statistics page allows the user to view his most recent exercise and sleep activity. The exercise page displays the duration, step count, miles walked/ran and calories burnt, and the sleep page displays the result of anomaly detection. Both the pages also have a option to view all the activity statistics, from the time user started using the application up till now.

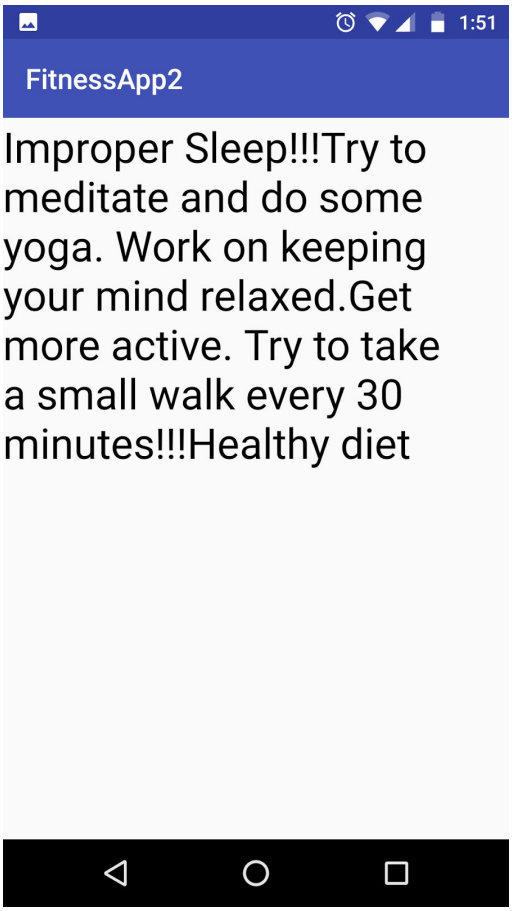


Fig 44: Suggestions Page

- **Suggestions:** Based on various factors, the user will be provided with suggestions on how to improve his sleep and health. The various factors range from step count, sleep result, BMI, age and gender.

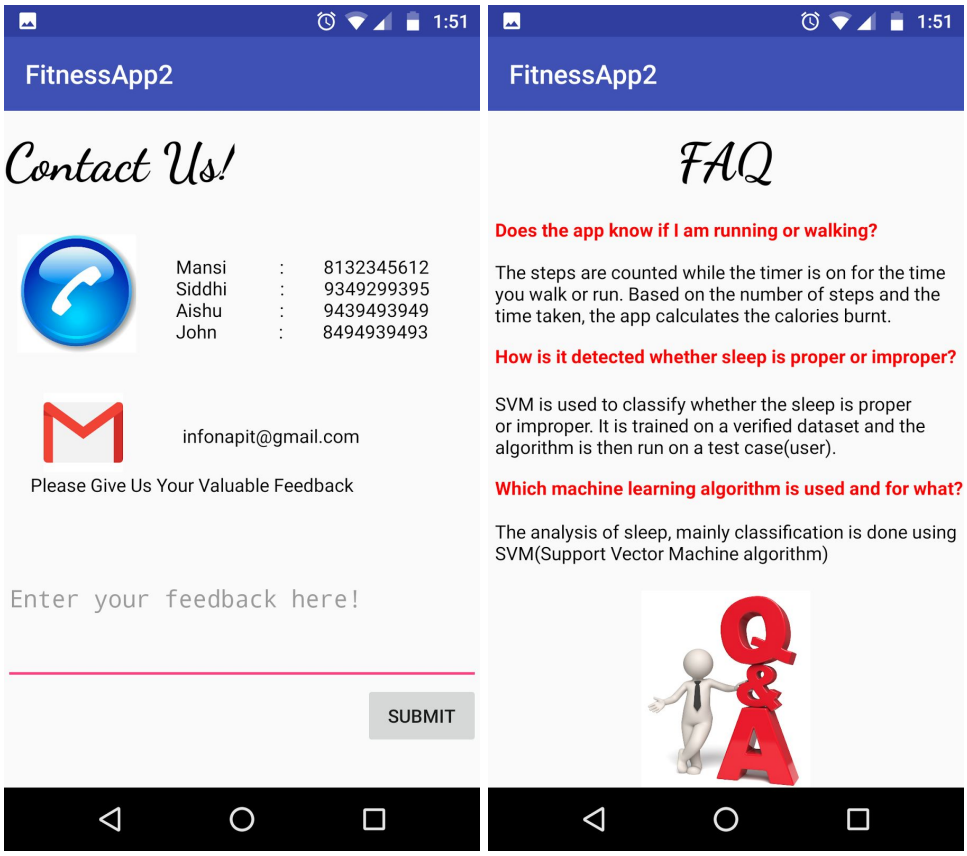


Fig 45: Contact Us(Left) and FAQ(Right)

- **Contact Us and Feedback:** This page has details about the people to be contacted in case the users have any trouble using the application or to clarify any doubts regarding it. It also provides a link to the website associated with the application. Also, it has an option to email. In the case if there are any questions or complaints regarding the functioning of the application, the user can opt to mail us his queries. The contact us page is for the users to be able to contact the developers.

- **Frequently Asked Questions:** This page has some solutions to frequently asked questions. This page is meant to assist the user when they encounter some problem while using the application and to troubleshoot the problem.

Additions in design:

We have included the following pages in design

1. Frequently Asked Questions(FAQ)
2. Contact Us and Feedback

We believe that these pages will help improve user experience and will provide guidance for us to make useful updates to the application.

12. DESIGN OF TESTS

12.1 Test cases and Test coverage

Test-case Identifier:	TC1
Use Case Tested:	UC1 Login
Pass/ Fail criteria:	The test passes if the user enters a key that is contained in the database, with less than a maximum allowed number of unsuccessful attempts
Input Data:	Username and password
Test Procedure:	Step 1. Type in an incorrect password and a valid email id Step 2. Type in the correct password and email id
Expected Result:	1. User enters invalid credentials. 2. Authentication fails. 3. User is prompted to re-enter credentials. a. User is finally able to login b. User fails more than 5 times in under 2 min

Table 76: Test case for UC1: Login

Test Coverage: The test will cover inputs that are null, empty, correct, incorrect.

Test-case Identifier:	TC2
Use Case Tested:	UC2: Registration
Pass/ Fail criteria:	The test passes if the user can register a username that is not contained in the database
Input Data:	Username and password
Test Procedure:	Step 1. Attempt to register username that already exists in the database. Step 2. Attempt to register username that does not exists in the database.
Expected Result:	1. The User clicks create account 2. The user is asked to create a new username and password. a. The username is checked against existing usernames in the database. Password is checked against regex to ensure password meets minimal requirements. i.) If username exists the user is informed and must pick a new unique username. ii.) If the username does not exists the user is allowed to continue. iii.) The password does not match the minimal requirements. E.g. Does not match regex condition. iv.) The password does match regex and user is allowed to continue. 3. User proceeds to Enter Personal details.

Table 77: Test case for UC2: Registration

Test Coverage: The test will cover inputs that are null, empty, correct, incorrect

Test-case Identifier:	TC3
Use Case Tested:	UC3: Manage Account
Pass/ Fail criteria:	The test passes if the user can change details about their account.
Input Data:	User details.
Test Procedure:	Step 1. User enters information in system and information is updated.
Expected Result:	1. User clicks settings 2. User enters settings and personal details 3. User entries are validated a. User entries are valid b. User entries are not valid 4. User entries are valid 5. User saves changes

Table 78: Test case for UC3: Manage Account

Test Coverage: The test will cover inputs that are null, empty, correct, incorrect.

Test-case Identifier:	TC4
Use Case Tested:	UC4: Monitor Sleep
Pass/ Fail criteria:	The test passes if the application can collect sensor reading over the sleep cycle.
Input Data:	Sensor readings
Test Procedure:	Step 1. User clicks the monitor sleep button and the application begins recording the sensor data..
Expected Result:	1. User clicks begin sleep monitoring 2. Application starts recording sensor data. 3. Application filters data. 4. App attaches a timestamp 5. App stores data into database.

Table 79: Test case for UC4: Monitor Sleep

Test Coverage: The test will cover inputs that are null, empty, out of the valid range, and valid readings.

Test-case Identifier:	TC5
Use Case Tested:	UC5: Anomaly detection
Pass/ Fail criteria:	Anomaly detector gives false positive or false negative
Input Data:	Sensor readings from database
Test Procedure:	Step 1. Cross validation data is feed into the anomaly detection algorithm. Step 2. Data labels are compared to the tags output by the detection algorithm.
Expected Result:	Data labels from the cross validation set match those output by the detection algorithm.

Table 80: Test case for UC5: Anomaly Detection

Test Coverage: The test will cover inputs that are null, empty, out of the valid range, and valid readings.

Test-case Identifier:	TC6
Use Case Tested:	UC6: View Statistics
Pass/ Fail criteria:	User is able to view stats about their sleep patterns.
Input Data:	Sensor readings from database
Test Procedure:	Step 1: User selects the view stats button. Step 2: The Stats are displayed.
Expected Result:	The application displays the stats for the user.

Table 81: Test case for UC6: View Statistics

Test Coverage: The test will cover inputs that are null, empty, out of the valid range, and valid readings.

Test-case Identifier:	TC7
Use Case Tested:	UC7: Recommender
Pass/ Fail criteria:	Recommend corrective actions to the user which makes sense
Input Data:	Sensor readings from database, output of anomaly detector.
Test Procedure:	Step 1: Trigger recommender system with a known issue. Step 2: See if recommender gives user correct recommendation .
Expected Result:	The recommender gives the correct corrective action.

Table 82: Test case for UC7: Recommender

Test Coverage: The test will cover inputs that are null, empty, out of the valid range, and valid readings.

Test-case Identifier:	TC8
Use Case Tested:	UC8: Exercise
Pass/ Fail criteria:	App stores the calorie burned to the database
Input Data:	Username, sensor data
Test Procedure:	Step 1: User begins recording data from sensors.
Expected Result:	1. The accelerometer senses the distance covered and the calories burnt.

Table 83: Test case for UC8: Exercise

Test Coverage: The test will cover inputs that are null, empty, out of the valid range, and valid readings.

12.2 Integration testing

Integration testing will be performed between:

- Database and each screen that requires userdata.
- Sensor and screen reading the data
- Sensor screen and the database
- Database and the anomaly detector.
- Database and the recommender.

We will be performing top down integration testing.
Each integration test will first with establishing that communications between each component is possible. Then communication will be check to confirm it is correct. Valid and invalid data will be sent to each component in order to catch edge conditions. Normal operation will be resumed and checked against the sensor readings.

13. PROJECT MANAGEMENT

This project has three major subsystems. Some will be developed in parallel by dividing our team into groups of three. The section below discusses the parts of the subsystems. The group will start working on the functional requirements and the user interface of the project.

13.1 Product Ownership

- Madhura and Manasi will be working on the design of System Architecture, design of data flow of the system, storage of data, detection of anomaly , filtering of sensor, design of exercise recommender.
- John and Siddhi will work on setting up the system to share code, System Architecture design,data flow design, sensor integration, anomaly detection, sensor filtering, backend integrations between components.
- Aishwarya and Preetraj will work on UI design, System Architecture design, data flow design of the system, anomaly detection, sensor filtering, calorie-intake recommender design.

Group	Member 1	Member 2
Subgroup 1	Madhura Daptardar	Manasi Mehta
Subgroup 2	John Grun	Siddhi Patil
Subgroup 3	Aishwarya Srikanth	Preetraj Singh Gujral

Table 84: Division of members

Subgroup	Functionality contributions
Subgroup 1	Setup Document collaboration, System Architecture design, Design data flow of the system, data store, anomaly detection, Research, sensor filtering, exercise recommender.
Subgroup 2	Setup system to share code, System Architecture design,Design data flow of the system, sensor integration, anomaly detection, Research, sensor filtering, backend integrations between components.
Subgroup 3	UI design, System Architecture design, Design data flow of the system, anomaly detection, Research, sensor filtering, calorie-intake recommender.

Table 85: Functionalities of each subgroups

The following table consists of all the deadlines we will meet for this project. The deadlines include report submissions, demos etc. Since we do not have sufficient time, the development phase will be done in parallel.

Milestones	**Description**	**Planned Date**
M0	**Project Proposal**	**Sep. 17, 2017**
	Finalize the project scope after getting feedback	**Sep. 20, 2017**
M1	**First Report**	**Oct. 18, 2017**
	Statement of work and requirements	**Sept 23, 2017**

	Functional requirements Spec and user interface	**Sept 29, 2017**
	Full report submission	**Oct. 6, 2017**
M2	**Second Report**	**Oct 27, 2017**
	Interaction Diagrams	**Oct 13, 2017**
	Class Diagram and System Architecture	**Oct 20, 2017**
	Full report submission	**Oct 27, 2017**
M3	**First Demo**	**Nov 1, 2017**
M4	**Third Report**	**Dec. 8, 2017**
M5	**Second Demo**	**Dec. 13, 2017**
M6	**Electronic Project Archive**	**Dec. 16, 2017**

Table 86: Project Deadlines

13.2 Gantt Chart

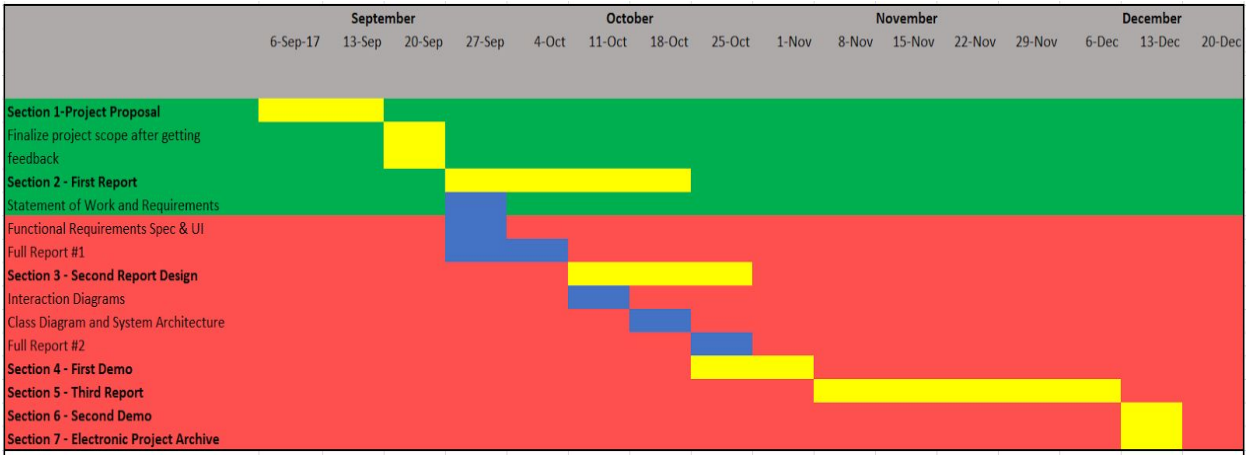


Figure 44: Gantt Chart

13.3 History of Work

13.3.1 Deadlines

With regards to the overarching milestones set in the course we stuck to all the deadlines and submitted everything on time. The work for all these checkpoints in the milestone M0-6 were completed on the planned deadline dates.

13.3.2 Key Accomplishments

The key accomplishments in the project can be enumerated as follows:

1. The application was built from scratch, and was not based on any previous project.
2. The application was designed as a completely pure android application.
3. The UI of the application has been standardized to be uniform over all the pages.
4. The android package has upwards of 10000 lines of code.

13.3.3 Future Work

We plan to work on detecting the abnormility (sleep paralysis, alzheimers, etc) from the improper sleep classification. We also plan to work on improving our user interface, by providing graphical statistics in View Statistics.

REFERENCES

[1] Association between objectively-measured physical activity and sleep, NHANES 2005-2006
http://www.sciencedirect.com/science/article/pii/S1755296611000317

[2] Sleep and Disease Risk
http://healthysleep.med.harvard.edu/healthy/matters/consequences/sleep-and-disease-risk

[3] How Your Sleep Affects Your Heart
http://www.webmd.com/sleep-disorders/features/how-sleep-affects-your-heart#1

[3] The Effects of Sleep Deprivation on Your Body
http://www.healthline.com/health/sleep-deprivation/effects-on-body

[4] Why Is Sleep Important?
https://www.nhlbi.nih.gov/health/health-topics/topics/sdd/why

[5] Relation of Sleep-disordered Breathing to Cardiovascular Disease Risk Factors : The Sleep Heart Health Study.
https://academic.oup.com/aje/article/154/1/50/117333

[6] Sensors Overview
https://developer.android.com/guide/topics/sensors/sensors_overview.html

[7] Anomaly image
https://machine-learning-class-notes.readthedocs.io/en/latest/lecture16.html

[8] Accelerometer
https://en.wikipedia.org/wiki/Accelerometer

[9] Sleep cycle and sleep cycle image
https://en.wikipedia.org/wiki/Sleep_cycle

[10] Possible login screen
https://blog.nativebase.io/react-native-login-logout-animation-722001c8fc55

[11] Possible settings screen
https://www.npmjs.com/package/react-native-root-toast

[13] Monitoring movement during sleep
https://en.wikipedia.org/wiki/Sleep_paralysis

[14]World Sleep Day
http://www.scoop.it/t/nebraska-legal-news/?&tag=Sleep

[15]Domain model
https://en.wikipedia.org/wiki/Domain_model

[16] Support Vector Machine
https://www.analyticsvidhya.com/blog/2017/09/understaing-support-vector-machine-example-code/

[17] Correlation between sleep and exercise
https://blog.bufferapp.com/inactivity-and-the-brain-why-exercise-is-more-important-than-ev

[18] Software Engineering, Ivan Marsic Home Page.
http://www.ece.rutgers.edu/~marsic/Teaching/SE1/

[19] Network Protocol
https://www.techopedia.com/definition/12938/network-protocols

[20] Network Protocol
https://www.lifewire.com/definition-of-protocol-network-817949

[21] SQLite Database
https://www.sqlite.org/about.html

[22]Class Diagram
https://en.wikipedia.org/wiki/Class_diagram

[23] Stride as a Function of Speed and Height
https://www.ijser.org/paper/Personal-Medical-Wearable-Device-for-Distance-Monitoring.html

[24] Calories Expended vs. Running
https://www.ijser.org/paper/Personal-Medical-Wearable-Device-for-Distance-Monitoring.html

[25] Support Vector Machine Classifier
https://www.analyticsvidhya.com/blog/2017/09/understaing-support-vector-machine-example-code/

[26]Regular and Irregular Sleep

https://www.sleepcycle.com/how-it-works/

[27]Model View Presenter
https://code.tutsplus.com/tutorials/an-introduction-to-model-view-presenter-on-android--cms-26162