SOFTWARE ENGINEERING 1
REPORT 1

---

napIT

October 08, 2017

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

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

CONTENTS

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 smartphones. 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 anomalies. These anomalies can take the form of 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 anomalies detected, remedies will be suggested. Most of the anomalies will be apparent the next morning for common problems. The anomalies may be lack of sleep, frequent wakeups, lack of sleep, or large amount of movements during the night. Two possible reasons that hinder a user from getting proper sleep are lack of physical exercise and improper food habits. Lack of exercise and the amount of food we eat has been linked with a lower quality of sleep. 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 analyse a person's sleep pattern. This application helps detect an anomaly in user's sleep pattern 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.

Further, sleep is also influenced by eating habits. We aim to incorporate a feature which will serve as a remedy to ensure that the user is follows a healthy diet. If an anomaly in the user's sleep has been detected, he/she will be requested to vary his/her calorie intake accordingly. On the basis of the food consumed and the workout schedule for that particular day, a smart alarm will be designed by us, which will set a sleep timing and will automatically trigger when the time expires.

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

This application came as a boon at this stage. After registering, there were quite a things which caught my attention as sections for Food, Exercise and Sleep. I realised that these were the things that I needed to look after as I was neglecting them a lot. This app allows me to enter my everyday food intake and gives me the amount of calorie intake consumed. It also helps me track my everyday activity, for example when I am running, it works as a treadmill in a way, where it gives me the amount of time I worked out and the meters covered along with the calories I burnt for the session. Further it provides the most important function where I can track my sleeping pattern and come across the anomalies based on the same. The application analyses as to why I am possibly having that particular medical condition and based on that provides a suggestion as to how I 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.

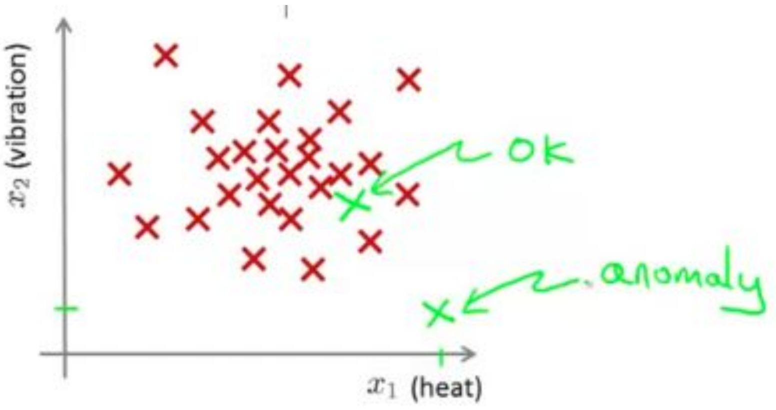


Fig 1: Anomaly

- **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].

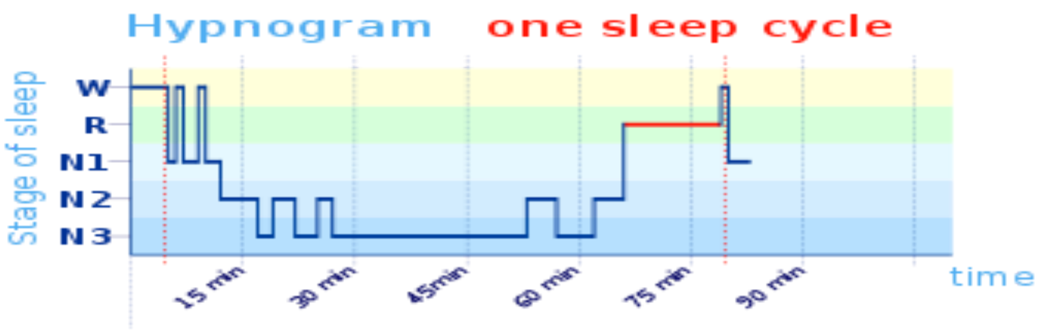


Fig 2: Sleep cycle

- **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. Every window will have a logout button so that the user can logout during any session.
- **Food:**
    1. *Enter food:* The user can select from among a set of food items that will be provided to the user through a dropdown menu.
    2. *Quantity:* The user can enter the quantity of the food item consumed.
    3. *View Statistics:* Here, the user can view his overall calorie intake.
- **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. *Smart Alarm:* Once the sleeping activity starts, there will be an alarm set in the background suggesting when is the appropriate time for the user to get up. The alarm goes on, giving a pop-up to either snooze or dismiss the

alarm. If by chance the user sleeps through the alarm, there will be a manual stop to terminate the activity once the user gets up.

3. *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.
4. *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 1 - 5. 1 is highest, 5 is lowest.

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

Table 1: Priority weights for requirements

3. ON SCREEN APPEARANCE REQUIREMENTS

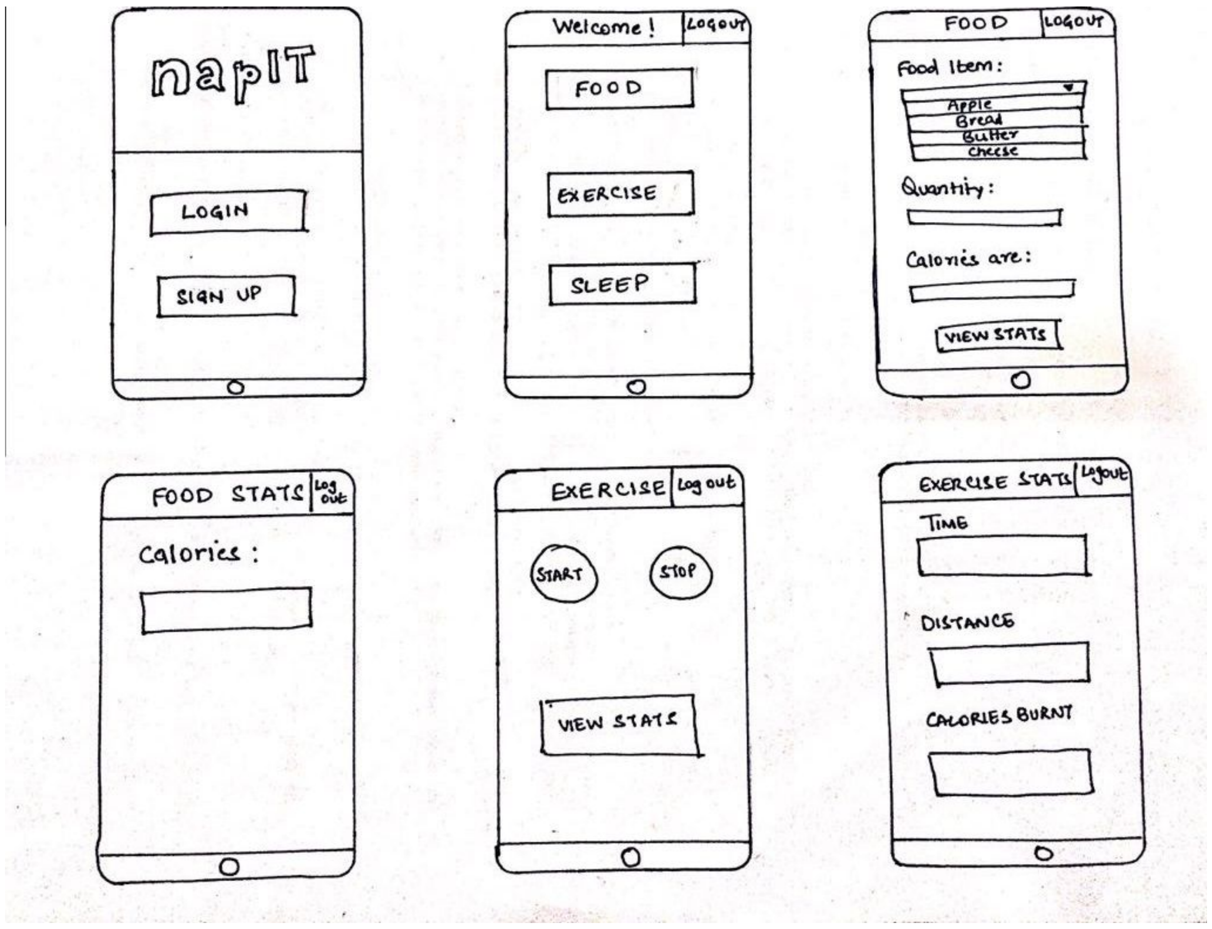


Fig 3: Paper prototyping for user stories

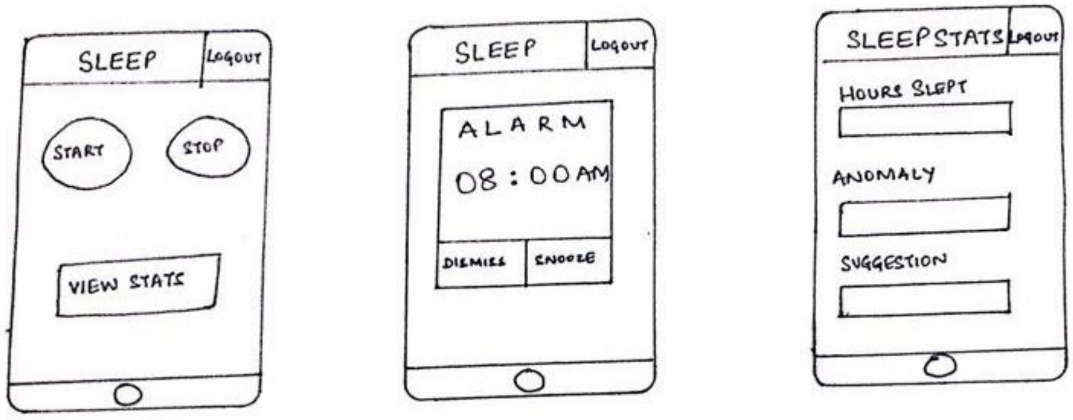


Fig 4: Paper prototyping for user stories

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 health care	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, 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 food consumed, 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: Smart Alarm - Once the activity has started, the smart alarm suggests the appropriate time for the user to wake up, and goes off at the particular time. The user will get an option to snooze or dismiss it.

UC-9: Eat - Whenever the user consumes some food, the user can also choose between the food choices in the dropdown menu and the quantity he consumes. The corresponding calories are added to the user's database.

UC-10: 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

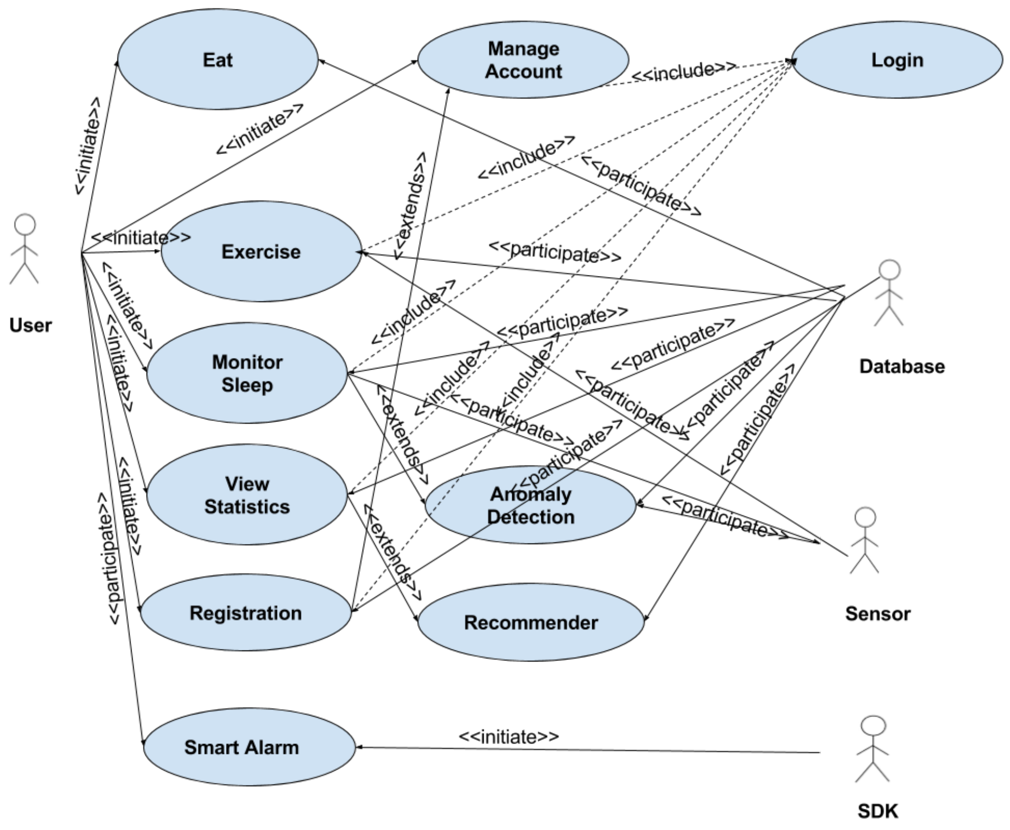


Fig 5: Use Case Diagram

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	UC-9	UC-10	UC-11
REQ-1	1	x	x	x	x							
REQ-2	5	x								x		
REQ-3	4	x									x	
REQ-4	1	x				x						
REQ-5	5	x				x				x		x
REQ-6	3	x				x	x					x
REQ-7	2	x				x		x				x
REQ-8	3	x							x			x
MAX PW		5	1	1	1	5	3	2	3	5	4	5
TOTAL PW		24	1	1	1	11	3	2	3	10	4	13

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 forMain 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
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 forMain 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.

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 analyse 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 an anomaly is detected, the user is informed otherwise nothing new happens.
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 a minimal amount of data has been collected.
Post conditions	User sleeping pattern sensor data along with food and exercise data is logged to the database
Flow of events for Main Success Scenario	1. User elects to view food, 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 a particular function stats. 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 a minimal amount of data 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: Smart Alarm

Use Case	**Functionality**
Use Case 8	Smart Alarm
Related Requirements	REQ-1, REQ-4, REQ-5
Initiating Actors	Android SDK
Actor Goals	1. To wake the user up at a particular time. 2. To decide how much sleep is most efficient for the user based on his activities.
Participating Actors	User, Database
Pre Conditions	The user should be logged into the system.
Post Conditions	
Flow of events for Main Success Scenario	1. The system starts recording the user's sleep data with a sensor. 2. Based on the user's daily activity, the application will set the best time for waking up. 3. At the particular time, the alarm will ring. 4. The user can choose to either snooze or dismiss it.

Table 12: Use Case 8: Smart Alarm

Use Case 9: Eat

Use Case	**Functionality**
Use Case 9	Eat
Related Requirements	REQ-1, REQ-2, REQ-5.
Initiating Actors	User
Actor Goals	To keep a check on calories consumed.
Participating Actors	Database
Pre Conditions	1. The user should be logged into the system. 2. The UI should display a dropdown menu which would contains possible food items.
Post Conditions	The consumed food and it's quantity should be stored in the database.
Flow of events for Main Success Scenario	1. The user selects the consumed food from the dropdown menu. 2. The user enters the quantity of food consumed. 3. The entered data is stored in the database. 4. The amount of calories gained is calculated and stored in the database.

Table 13: Use Case 9: Eat

Use Case 10: Exercise

Use Case	**Functionality**
Use Case 10	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. The UI displays an option to start the activity. 2. The user chooses to start the activity. 3. The accelerometer senses the distance covered and the calories burnt.

Table 14: Use Case 10: 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.

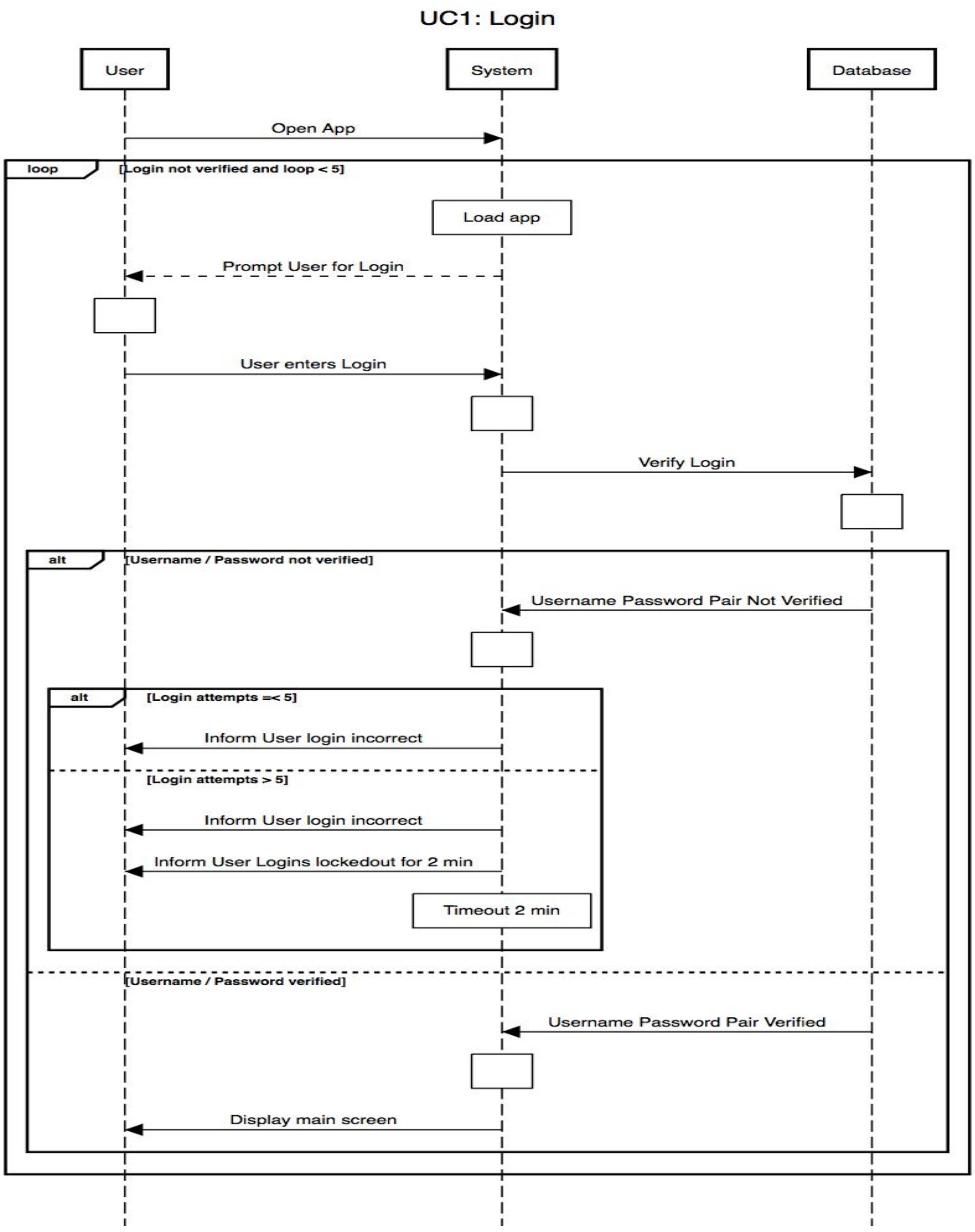

UC1: Login
User
System
Database
Open App
loop
[Login not verified and loop < 5]
Load app
Prompt User for Login
User enters Login
Verify Login
alt
[Username / Password not verified]
Username Password Pair Not Verified
alt
[Login attempts =< 5]
Inform User login incorrect
[Login attempts > 5]
Inform User login incorrect
Inform User Logins lockedout for 2 min
Timeout 2 min
[Username / Password verified]
Username Password Pair Verified
Display main screen

Fig 6: UC 1: Login

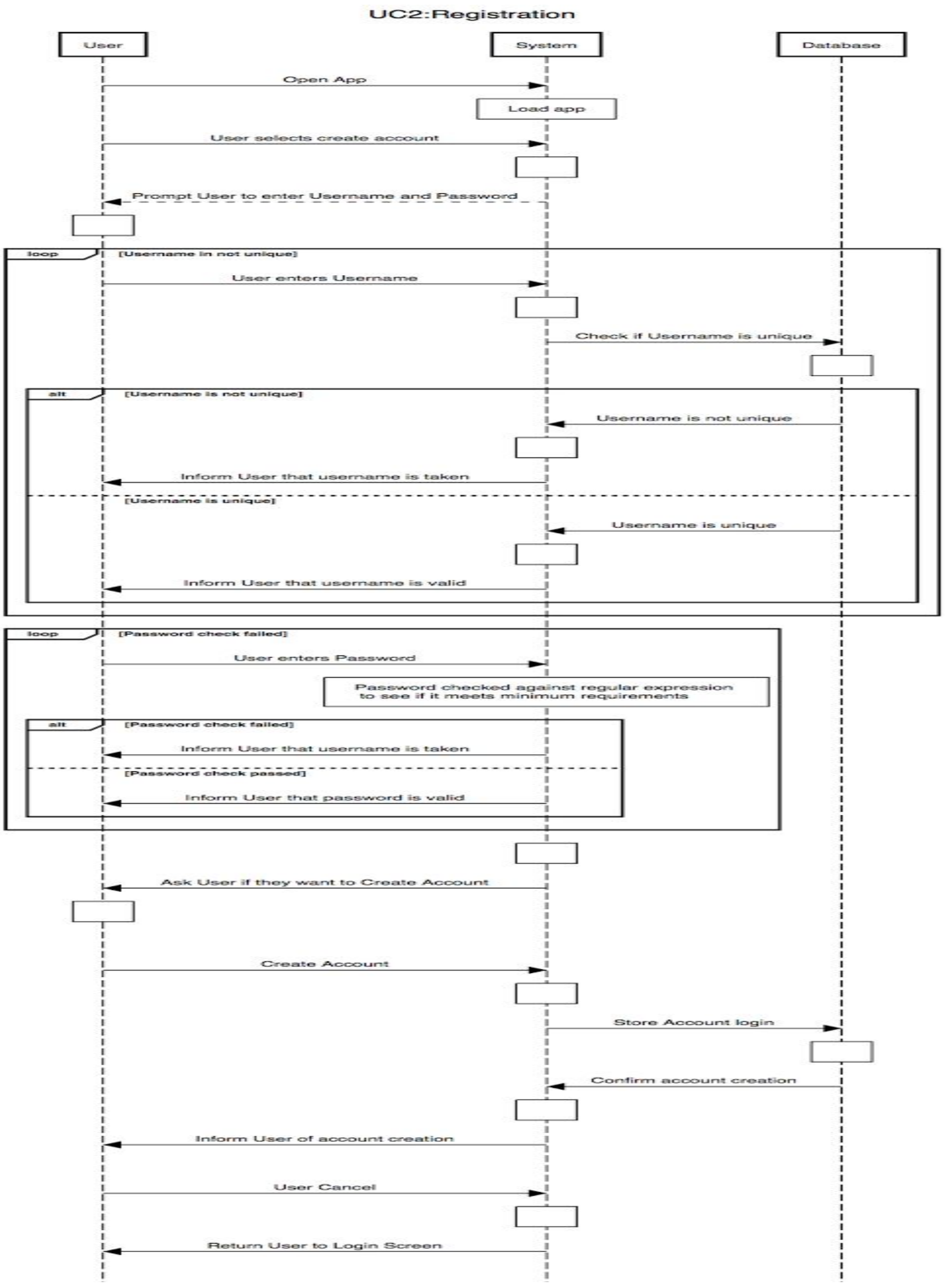

UC2:Registration
User
System
Database
Open App
Load app
User selects create account
Prompt User to enter Username and Password
loop [Username in not unique]
User enters Username
Check if Username is unique
alt [Username is not unique]
Username is not unique
Inform User that username is taken
[Username is unique]
Username is unique
Inform User that username is valid
loop [Password check failed]
User enters Password
Password checked against regular expression to see if it meets minimum requirements
alt [Password check failed]
Inform User that username is taken
[Password check passed]
Inform User that password is valid
Ask User if they want to Create Account
Create Account
Store Account login
Confirm account creation
Inform User of account creation
User Cancel
Return User to Login Screen

Fig 7: UC2: Registration

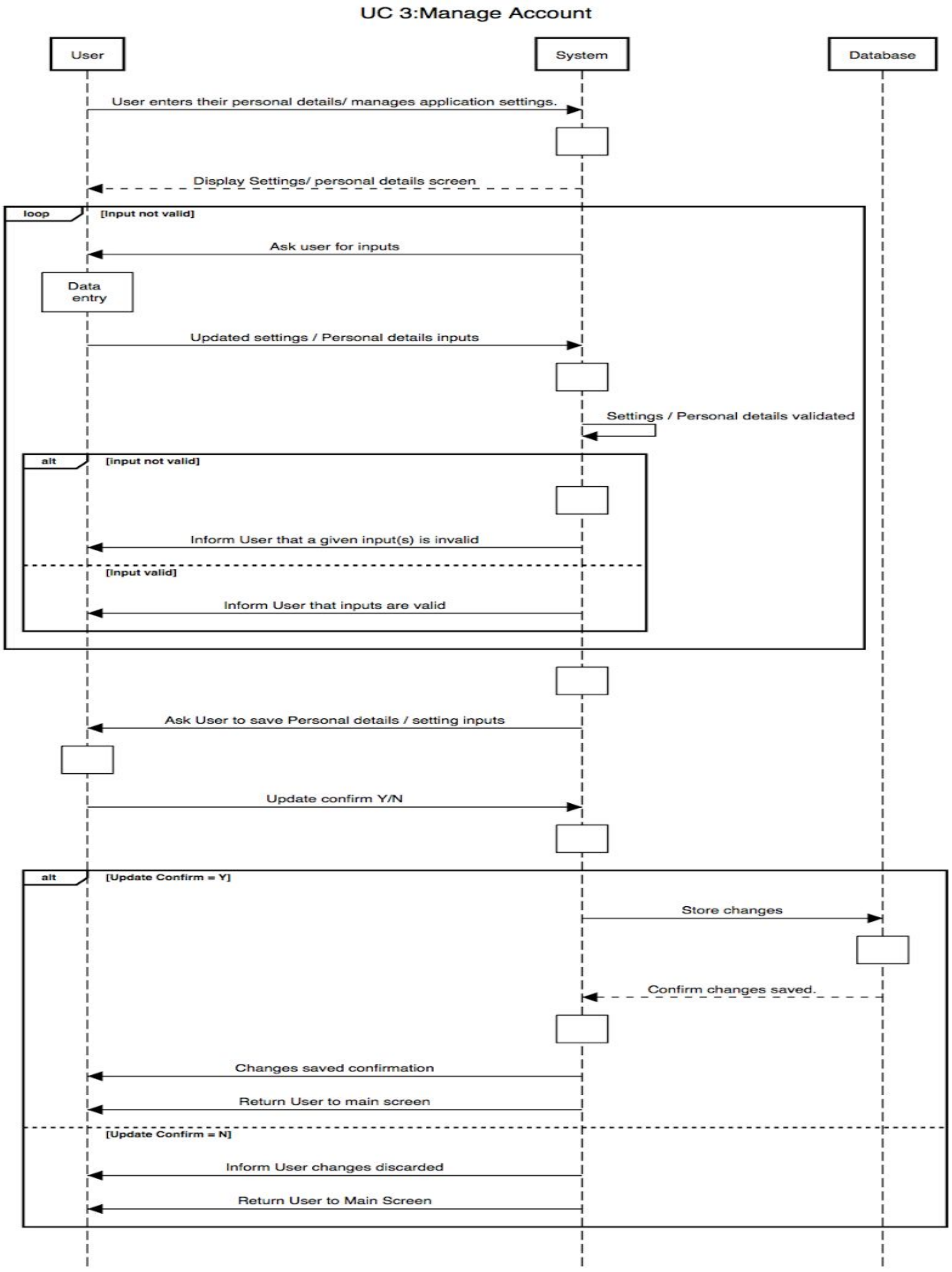

UC 3:Manage Account
User
System
Database
User enters their personal details/ manages application settings.
Display Settings/ personal details screen
loop
[Input not valid]
Ask user for inputs
Data entry
Updated settings / Personal details inputs
Settings / Personal details validated
alt
[input not valid]
Inform User that a given input(s) is invalid
[Input valid]
Inform User that inputs are valid
Ask User to save Personal details / setting inputs
Update confirm Y/N
alt
[Update Confirm = Y]
Store changes
Confirm changes saved.
Changes saved confirmation
Return User to main screen
[Update Confirm = N]
Inform User changes discarded
Return User to Main Screen

Fig 8: UC3: Manage Account

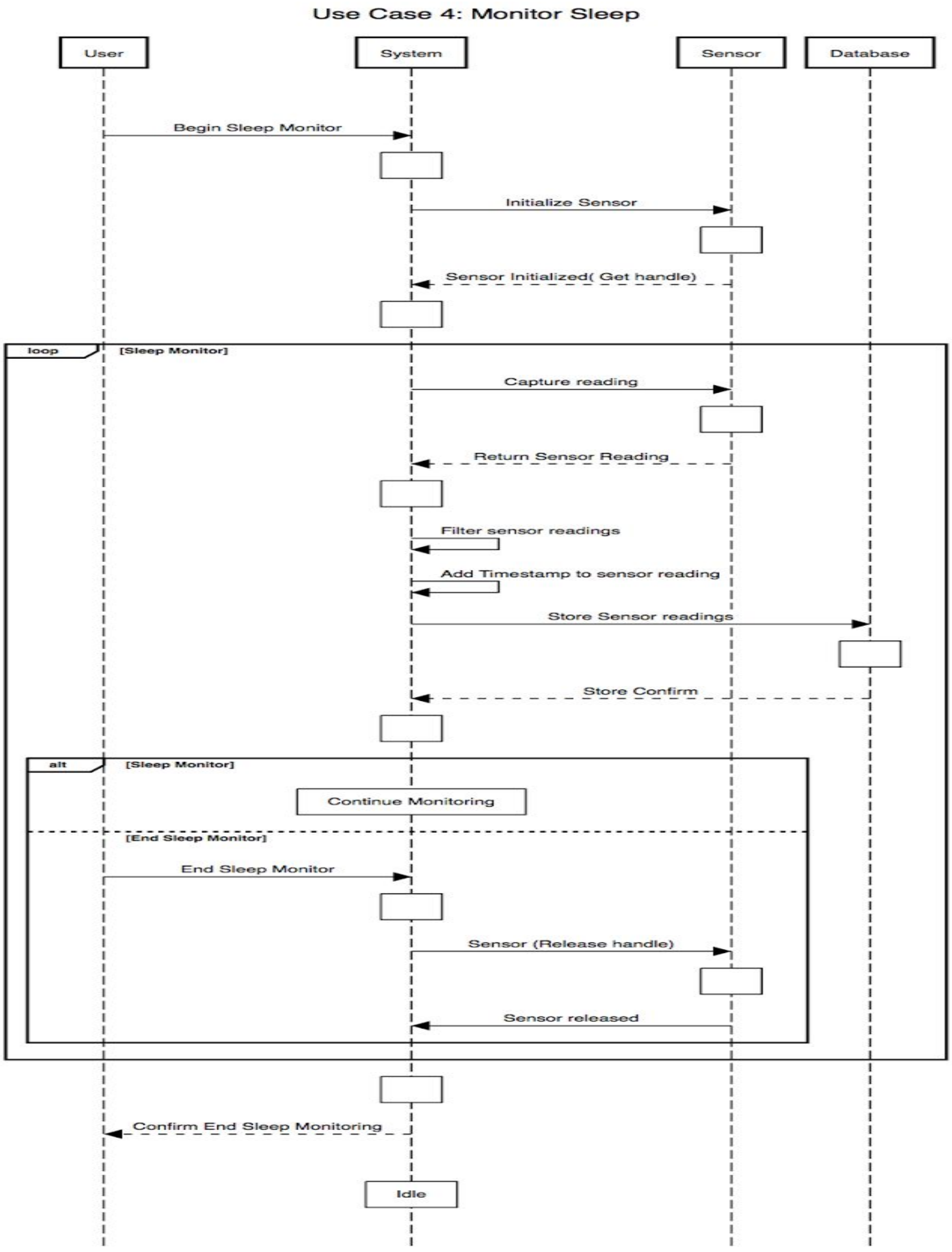


Fig 9: UC4: Monitor sleep

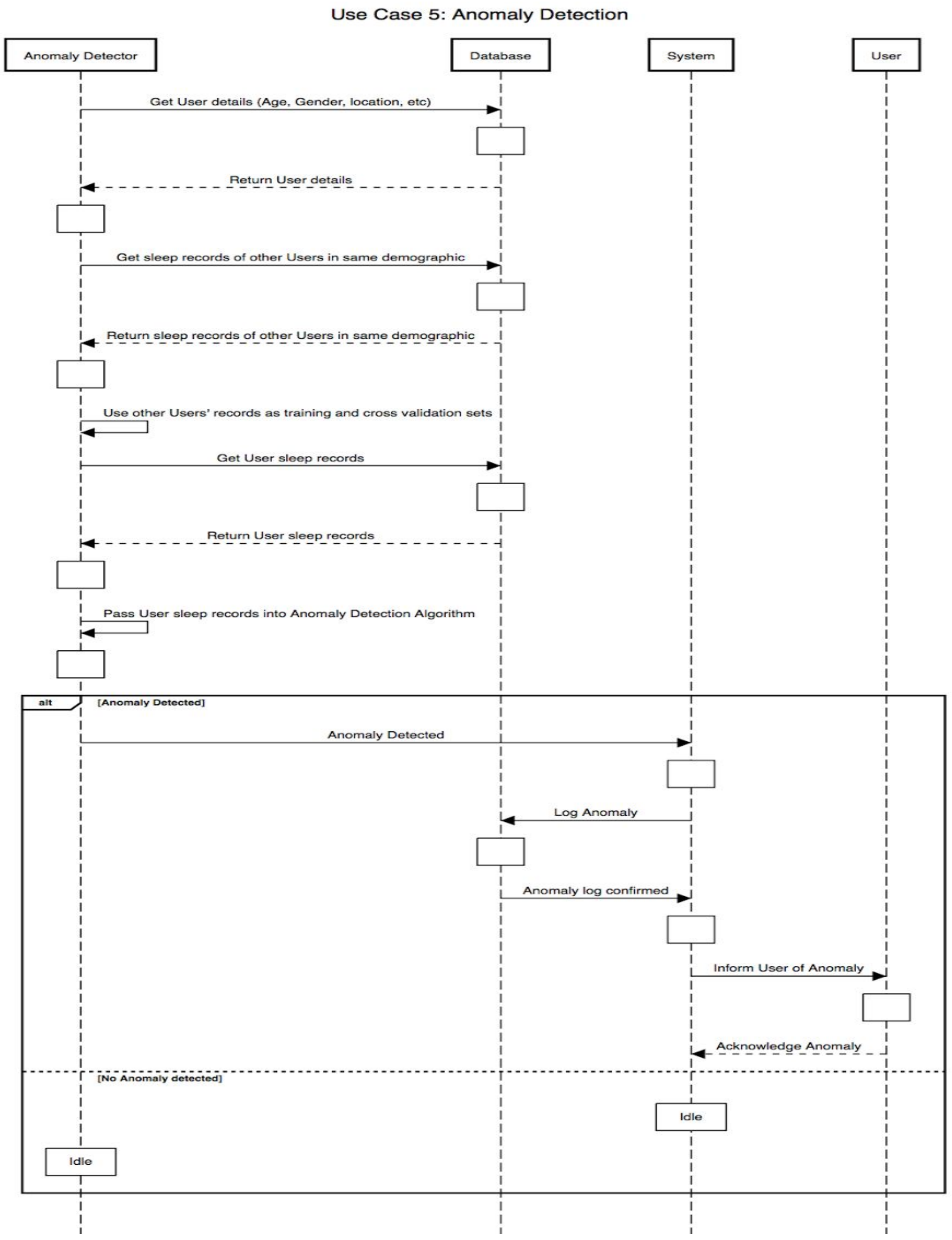

Use Case 5: Anomaly Detection
Anomaly Detector
Database
System
User
Get User details (Age, Gender, location, etc)
Return User details
Get sleep records of other Users in same demographic
Return sleep records of other Users in same demographic
Use other Users' records as training and cross validation sets
Get User sleep records
Return User sleep records
Pass User sleep records into Anomaly Detection Algorithm
alt
[Anomaly Detected]
Anomaly Detected
Log Anomaly
Anomaly log confirmed
Inform User of Anomaly
Acknowledge Anomaly
[No Anomaly detected]
Idle
Idle

Fig 10: UC 5: Anomaly Detection

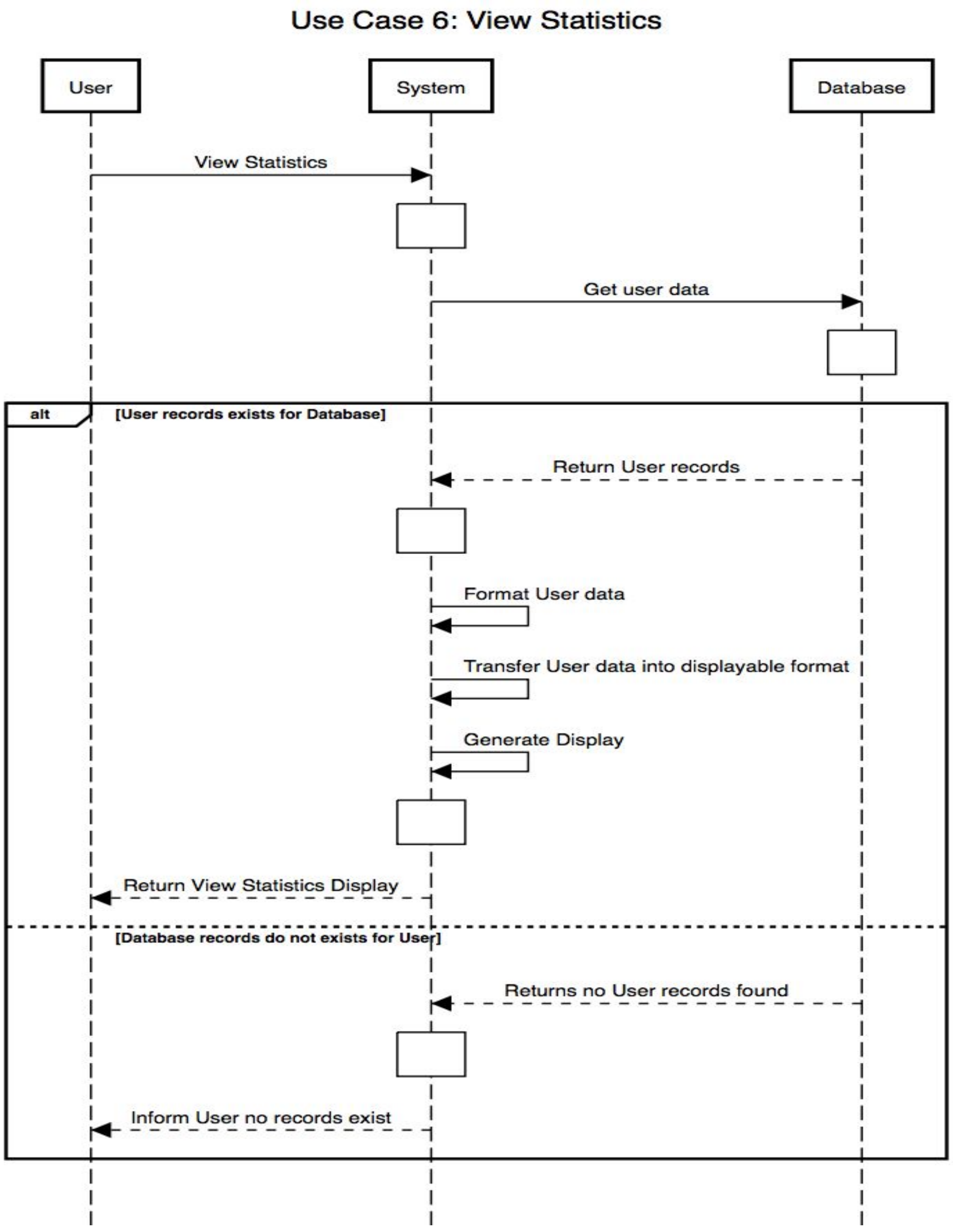

Use Case 6: View Statistics
User
System
Database
View Statistics
Get user data
alt
[User records exists for Database]
Return User records
Format User data
Transfer User data into displayable format
Generate Display
Return View Statistics Display
[Database records do not exists for User]
Returns no User records found
Inform User no records exist

Fig 11: UC 6: View statistics

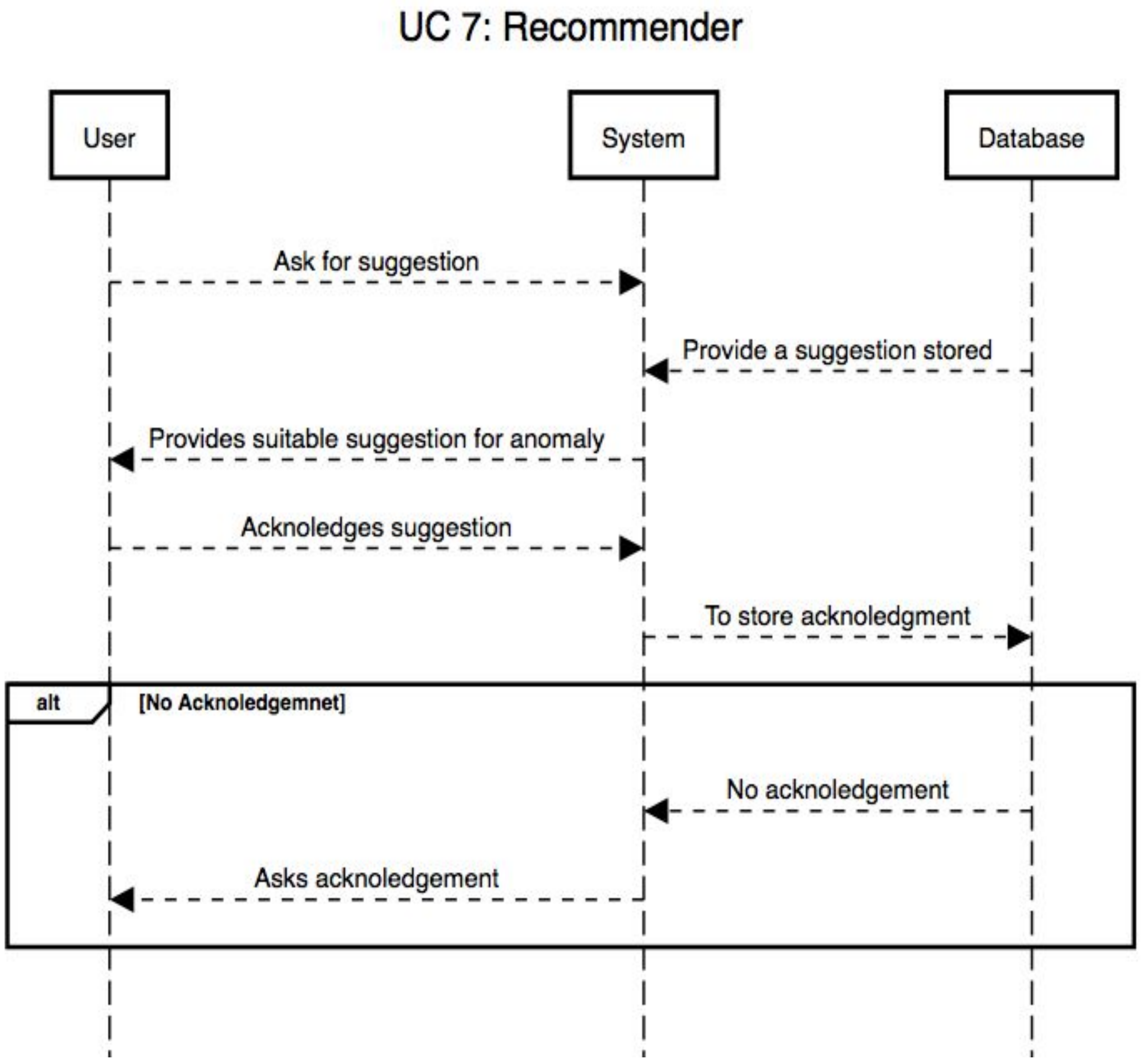


Fig 12: UC 7: Recommender

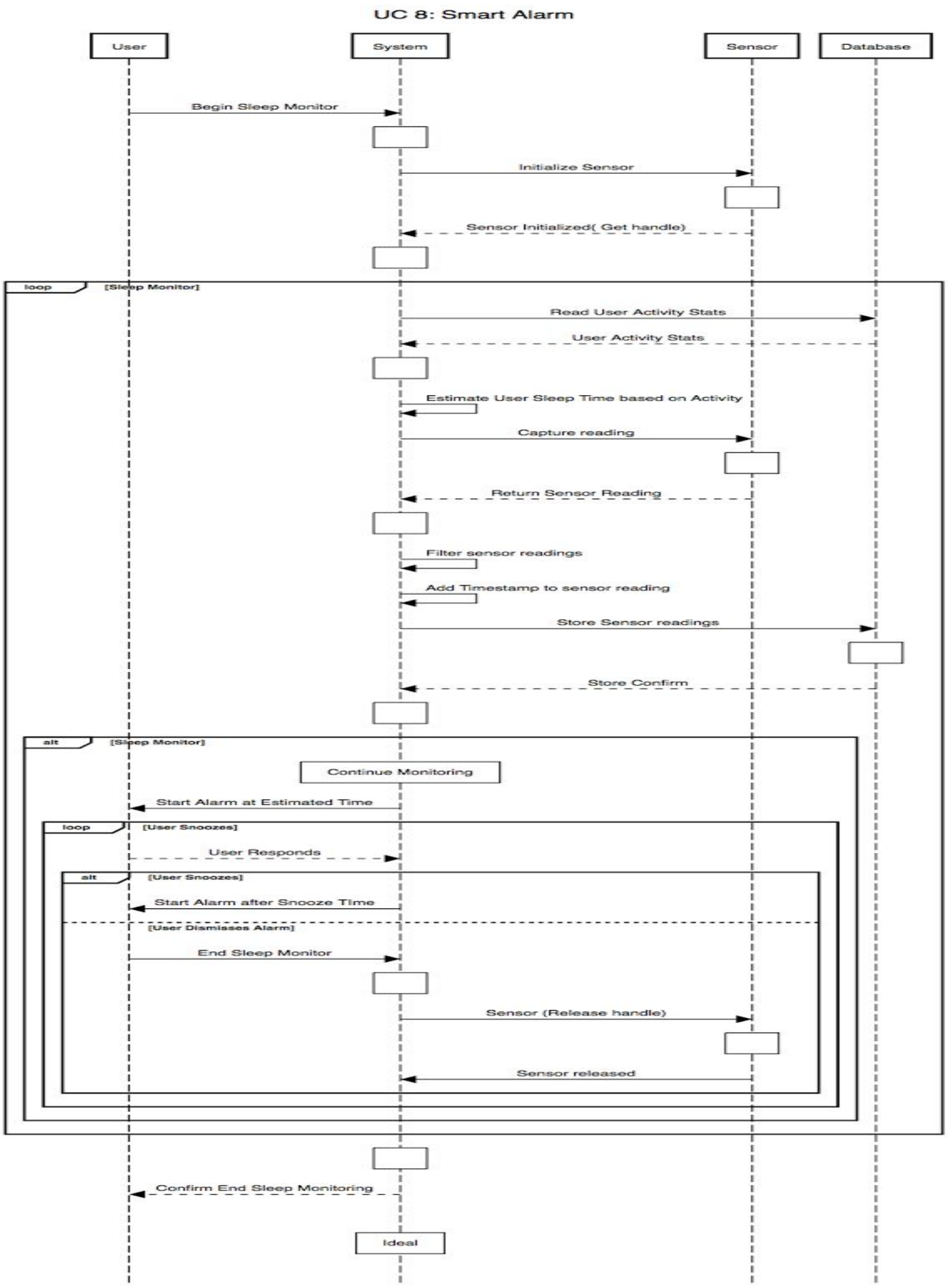


Fig 13: UC 8: Smart Alarm

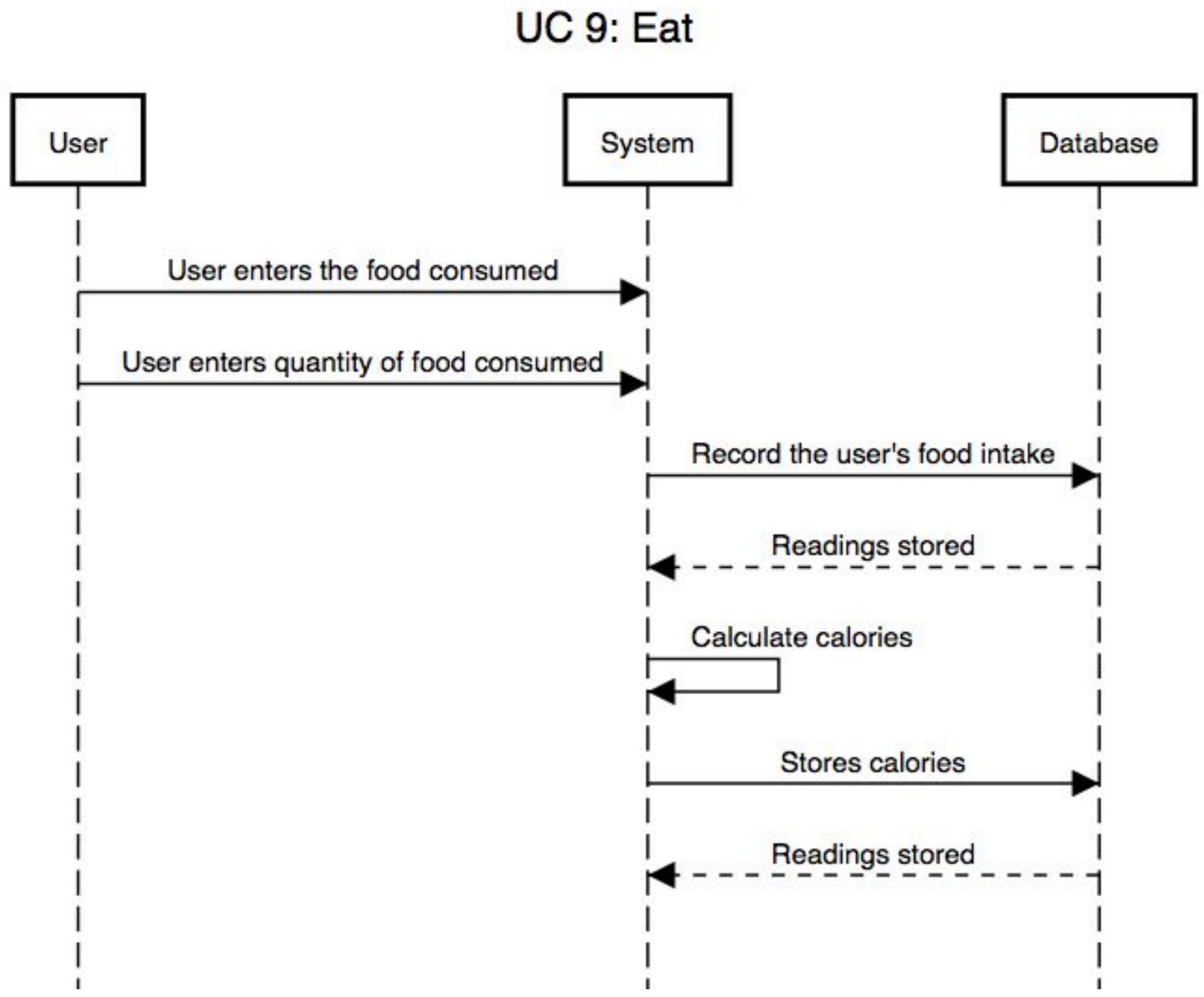

UC 9: Eat
User
System
Database
User enters the food consumed
User enters quantity of food consumed
Record the user's food intake
Readings stored
Calculate calories
Stores calories
Readings stored

Fig 14: UC 9: Eat

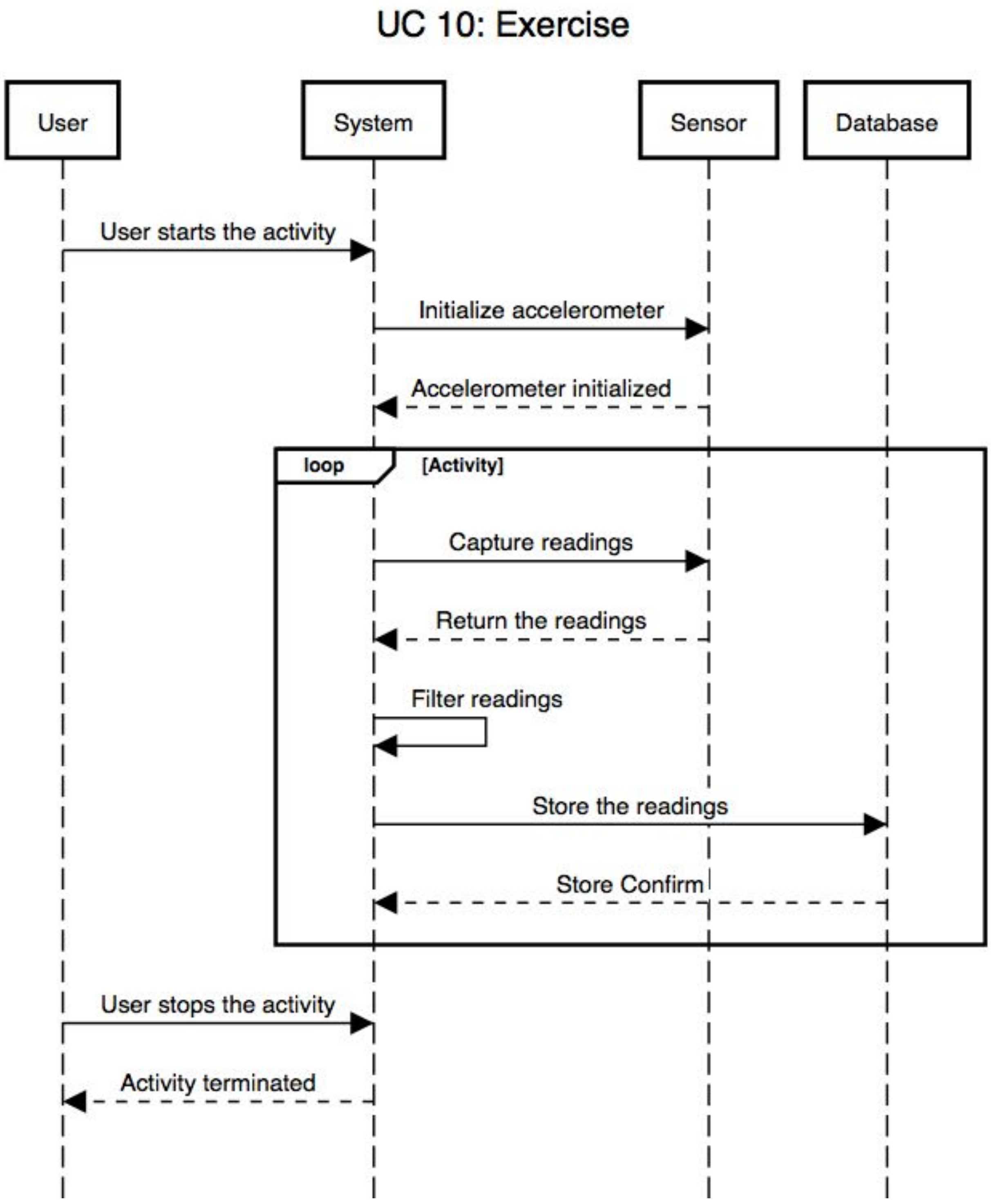


Fig 15: UC10: Exercise

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 textbox 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 textbox, 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:** The main menu has different buttons like food, 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.

- **Smart Alarm:** A smart alarm will set the wake-up time for the user based on his physical activity for that particular day and the type of food the user consumes. The user can either snooze or dismiss the alarm. Snooze will trigger the alarm again in ten minutes. Dismiss will shut down the alarm.

- **Food intake:** When the user clicks on the food button in the main menu, he/she will be directed to a page where there is an option to enter the type of food and quantity consumed by the user. The type of food can be chosen from a drop down list box and quantity can be entered in the textbox. Once the user hits the “Submit” button, these details get saved in the database.

- **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 and the number of calories gained during food intake for that particular day. The user can log out anytime by clicking on the “logout” button.

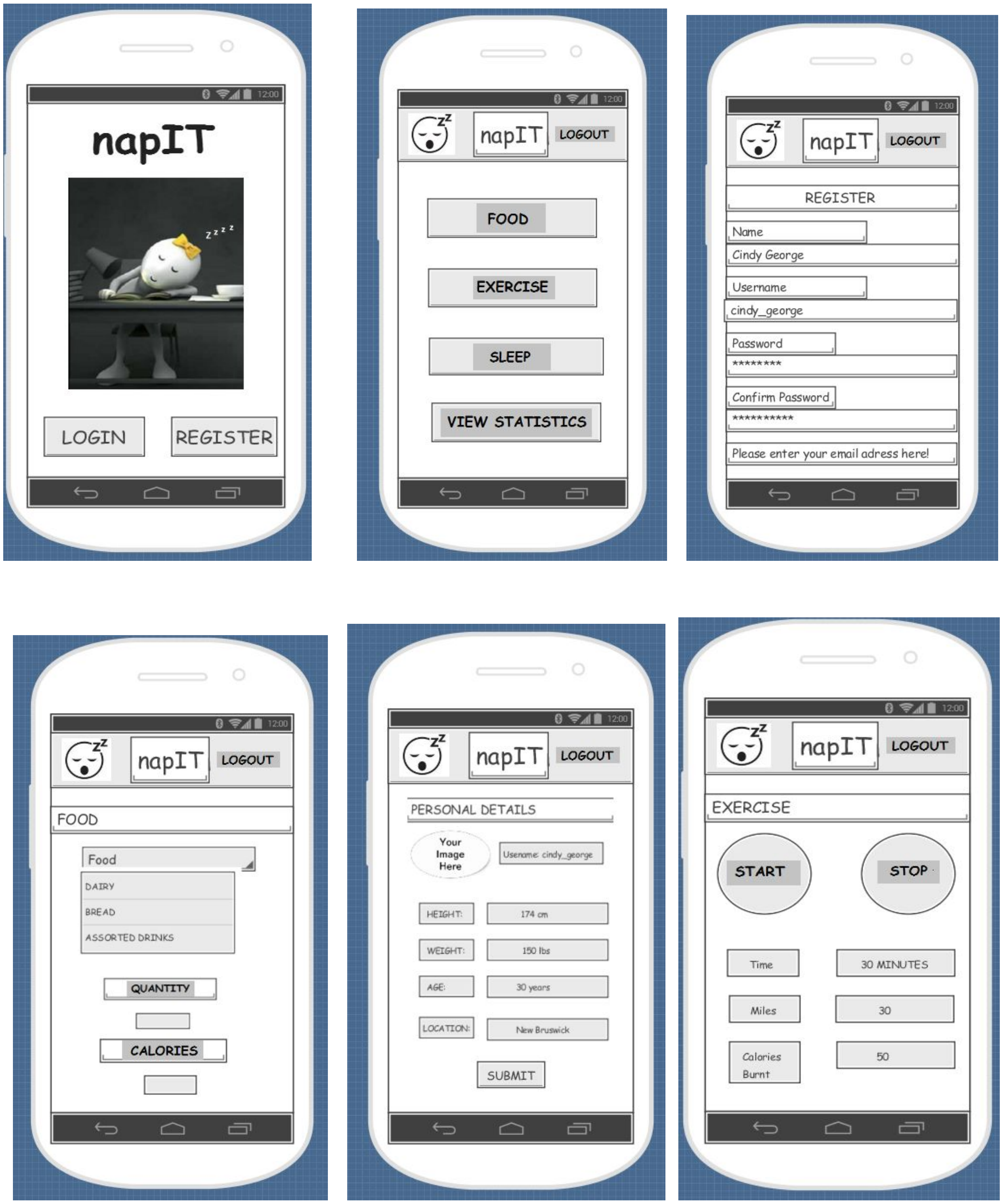


Fig 16: Mock-ups: Login, Main Menu, Register, Food, Personal Details, Exercise

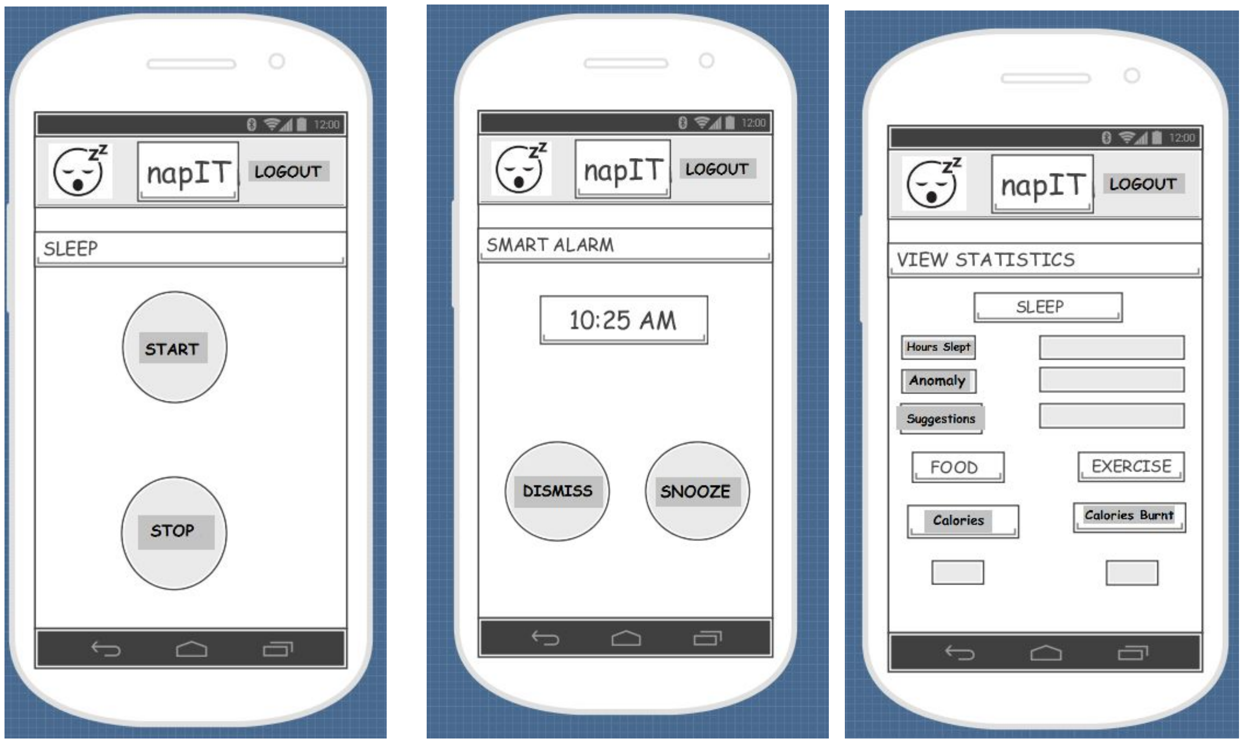


Fig 17: Mock-ups: Sleep, Smart Alarm, View-Statistics

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. Food

- Navigation - 2 mouse clicks
    - ➔ Click on the “Food” button in main menu.
    - ➔ Click on the “Submit” button after completing data entry.
- Data Entry - 3 mouse clicks and one keypress minimum.
    - ➔ Click on the Drop down List and then click on the food type.

- ➔ Click on the “Quantity” field.
- ➔ Enter the quantity(keys>1)

8. View Statistics

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

9. 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.

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 user's 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: Smart Alarm

Responsibility Description	**Type**	**Concept Name**
Acts as a link by controlling the interaction between different parts of the system.	D	MainController
Decides how much sleep is accurate for the user	D	SleepDecider
Rings the alarm at the time given by the timer.	D	AlarmRinger

Sets the time at which the alarm should off	D	Timer
Allows the user to decide whether to snooze or dismiss	K	DecisionMaker
Snoozes the alarm and sets the timer for a set period.	D	Snoozer
Dismisses the alarm and turns off the timer	D	Dismisser

Table 22: Concept Definition of Use Case 8: Smart Alarm

Concept Definition of Use Case 9: Eat

Responsibility Description	**Type**	**Concept Name**
Responsible for generating Eat GUI	D	GUI Generator
Detects the selection of the food item from the drop down menu and Detects user input for quantity	D	SelectionDetector
Readings from Calorie meter stored in Database	D	FoodTracker

Table 23: Concept Definition of Use Case 9: Eat

Concept Definition of Use Case 10: 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	D	AccelerometerController

speed of the walk or run once the “Start” button is pressed and release access to the sensors on pressing the “Stop” Button.		
Stores the distance covered, step count and calories burnt in the database.	K	ExerciseInfoKeeper

Table 24: Concept Definition of Use Case 10: 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
PendingRecommendationInfoKeeper	Unacknowledged Recommendation	The unacknowledged recommendation from the user

Table 31: Attribute Definition of Use Case 7: Recommender

Attribute Definition for Use Case 8: Smart Alarm

Concept	**Attributes**	**Attribute Description**
SleepDecider	User details implying the amount of sleep needed.	To decide how much sleep a user requires
AlarmRinger	Time duration from the database	Turns on the alarm and rings it.
Timer	Time	Sets the timer on/off during wakeup

		or snooze
DecisionMaker	Snooze/ Dismiss Button	To allow the user to decide whether to snooze or dismiss the alarm
Snoozer	Snooze button	To allow the user to snooze the alarm
Dismisser	Dismiss button	To allow the user to dismiss the alarm.

Table 32: Attribute Definition of Use Case 8: Smart Alarm

Attribute Definition for Use Case 9: Eat

Concept	**Attribute Name**	**Attribute Description**
GUIGenerator	List	List for selection for food details
SelectionDetector	List Button	Detects the selection of food item and the selection of quantity
FoodTracker	Food data	Keeps the food details and calories consumed in the database.

Table 33: Attribute Definition of Use Case 9: Eat

Attribute Definition for Use Case 10: 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 10: 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 compare it against the other user's in the same demographic	Anomaly Connection
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↔	The Database Manager	populates

PendingRecommendationInfoKeeper	populates the PendingRecommendationInfoKeeper from the database.	
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 : Smart Alarm

Concept Pair	**Association Description**	**Association Name**
MainController ↔ AlarmRinger	The MainController displays the AlarmRinger pop up when the timer goes off	displays
SleepDecider ↔ Timer	SleepDecider calculates the best amount of sleep a user requires and sets the Timer accordingly.	calculates
Timer ↔ AlarmRinger	When the Timer goes off based on the calculated sleep amount, it informs the Alarm Ringer	informs
DecisionMaker ↔ Snoozer	When the users chooses snooze button displayed by the Decision Maker, it informs the Snoozer.	informs
DecisionMaker ↔ Dismisser	When the users chooses dismiss button displayed by the Decision Maker, it informs the Dismiss.	informs
Snoozer ↔ Timer	The Snoozer informs the Timer to set the wake up time to a set period.	informs

Table 42: Association Definition of Use Case 8: Smart Alarm

Association definition of Use Case 9: Eat

Concept Pair	**Association Description**	**Association Name**
SelectionDetector ↔ GUIGenerator	Lists to enter food details	Food info
FoodTracker ↔	Saves the data to the database	Keep info

SelectionDetector		
FoodTracker ↔ GUIGenerator	Makes changes in GUI	Change GUI

Table 43: Association Definition of Use Case 9: Eat

Association definition of Use Case 10: 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 10: 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 Genera-tor	Login Interface	Discon-nector	Main Controller	Registration Interface	Valida-tor	DatabaseMan-ager	UserDetail Keeper	SelectionDete-ctor
Login	x	x	x						
Registration				x	x	x	x		
Manage Accounts	x						x	x	x
Monitor Sleep	x								x
Anomaly Detection	x						x		
View Statistics	x								
Recommender									
Smart Alarm									
Eat	x								x
Exercise	x								x

	AccountManager	Respo--nse Contro-l	Sleep Contro-l	AnalysisManager	ObserveControl	StatsControl	AnomalyControl	AnomalyConnection	AnomalyDisplayer
Login									
Registration									
Manage Accounts	x								
Monitor Sleep		x	x	x	x	x			
Anomaly Detection							x	x	x
View Statistics									
Recommender									
Smart Alarm									
Eat									
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						
Smart Alarm					x	x	x	x	x	x
Eat										
Exercise										

	FoodTracker	RunController	AccelerometerController	ExerciseInfoKeeper	
Login					
Registration					
Manage Accounts					
Monitor Sleep					
Anomaly Detection					
View Statistics					
Recommender					
Smart Alarm					
Eat	x				
Exercise		x	x	x	

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:

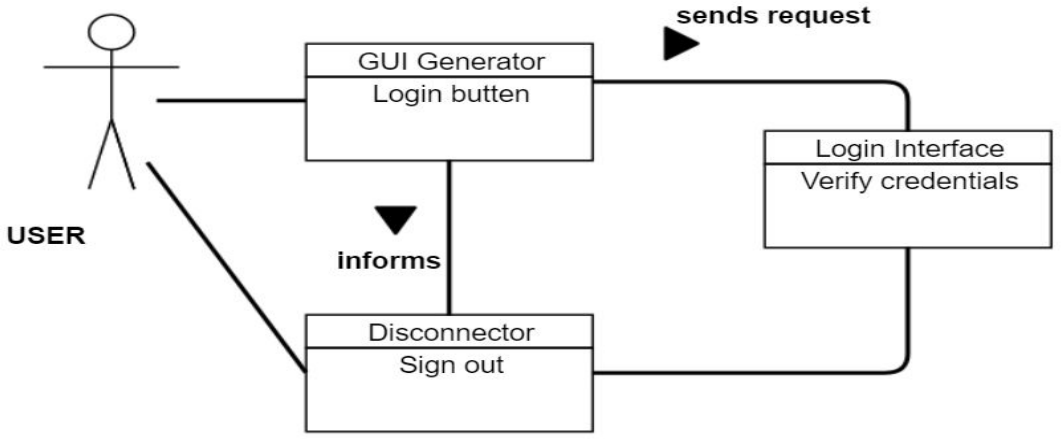


Fig 18: Domain Model: Login

Registration:

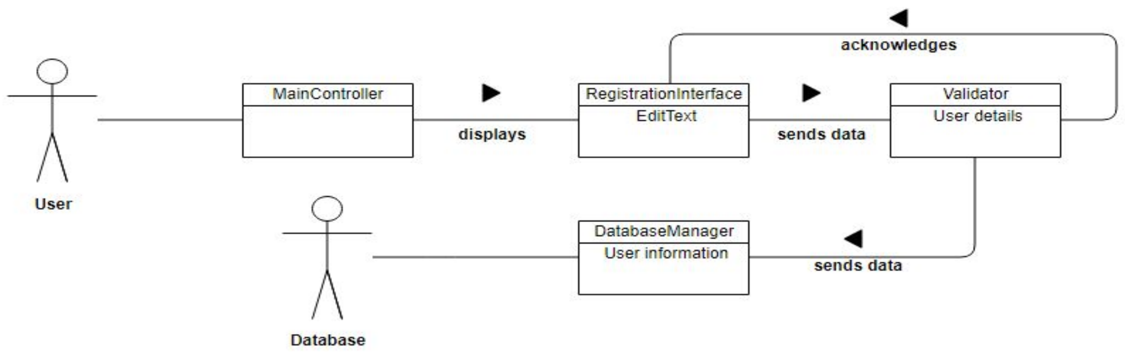


Fig 19: Domain Model: Registration

Manage Accounts:

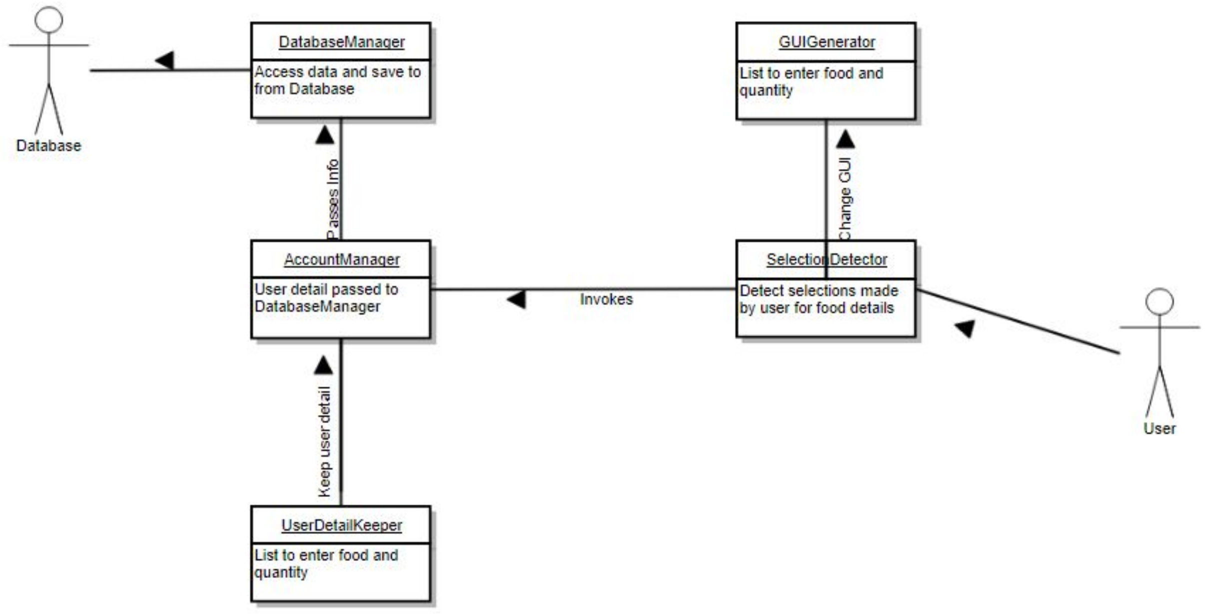

Database
DatabaseManager
Access data and save to from Database
Passes Info
GUIGenerator
List to enter food and quantity
Change GUI
AccountManager
User detail passed to DatabaseManager
Invokes
SelectionDetector
Detect selections made by user for food details
User
Keep user detail
UserDetailKeeper
List to enter food and quantity

Fig 20: Domain Model: Manage Accounts

Sleep Monitor:

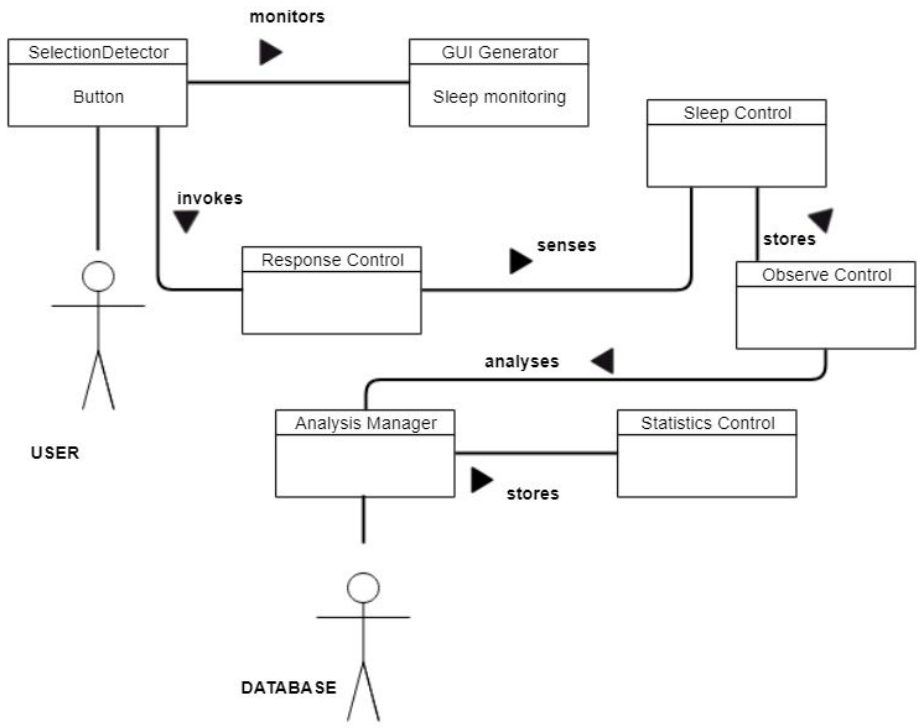


Fig 21: Domain Model: Sleep Monitor

Anomaly Detection:

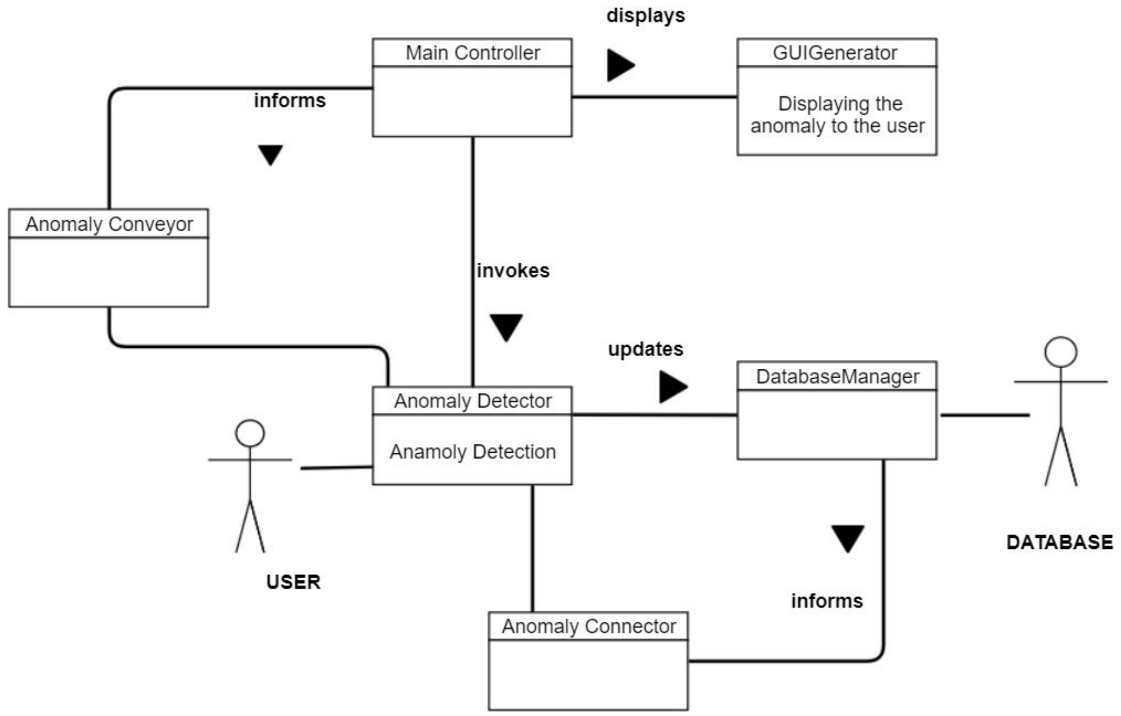


Fig 22: Domain Model: Anomaly Detection

View Statistics:

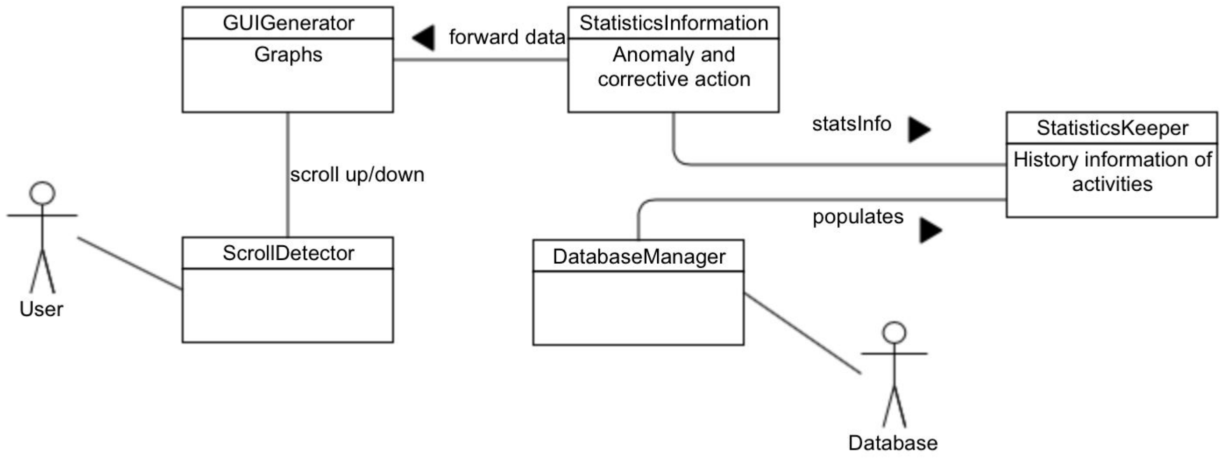


Fig 23: Domain Model: View Statistics

Recommender:

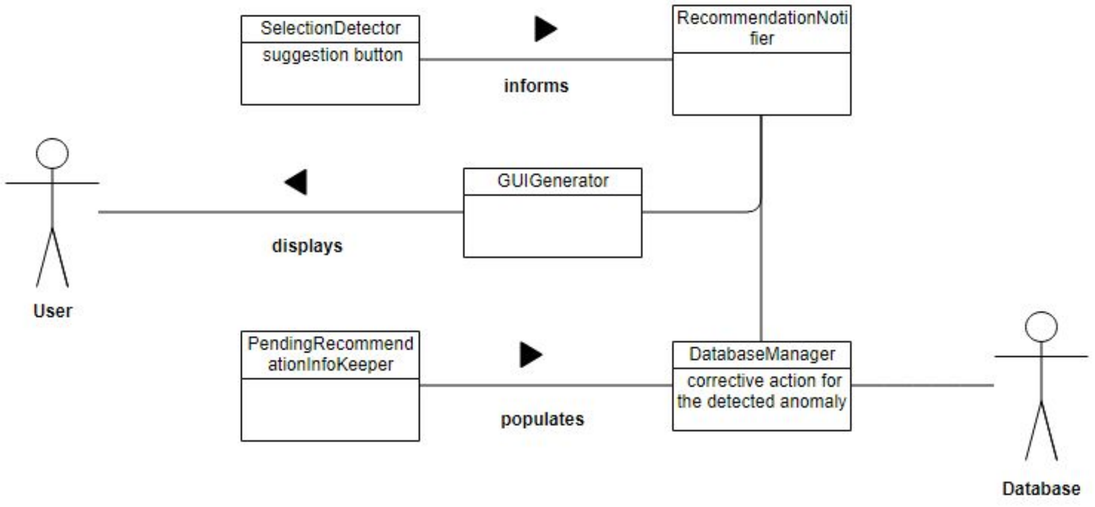


Fig 24: Domain Model: Recommender

Smart Alarm:

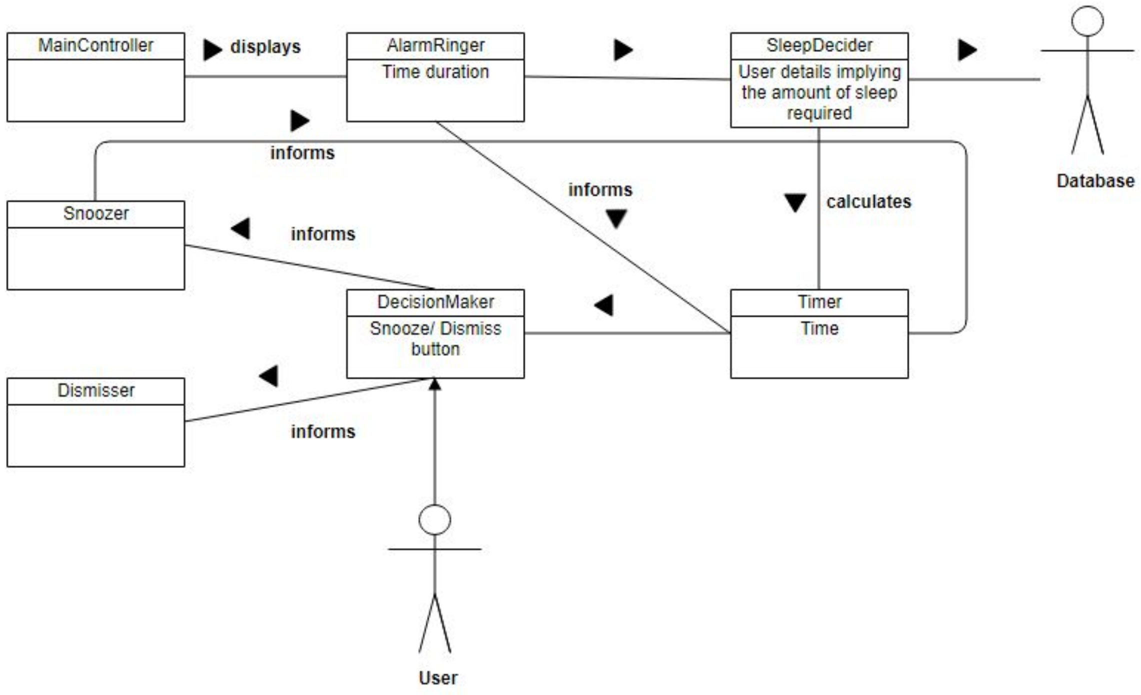


Figure 25: Domain Model: Smart Alarm

Exercise:

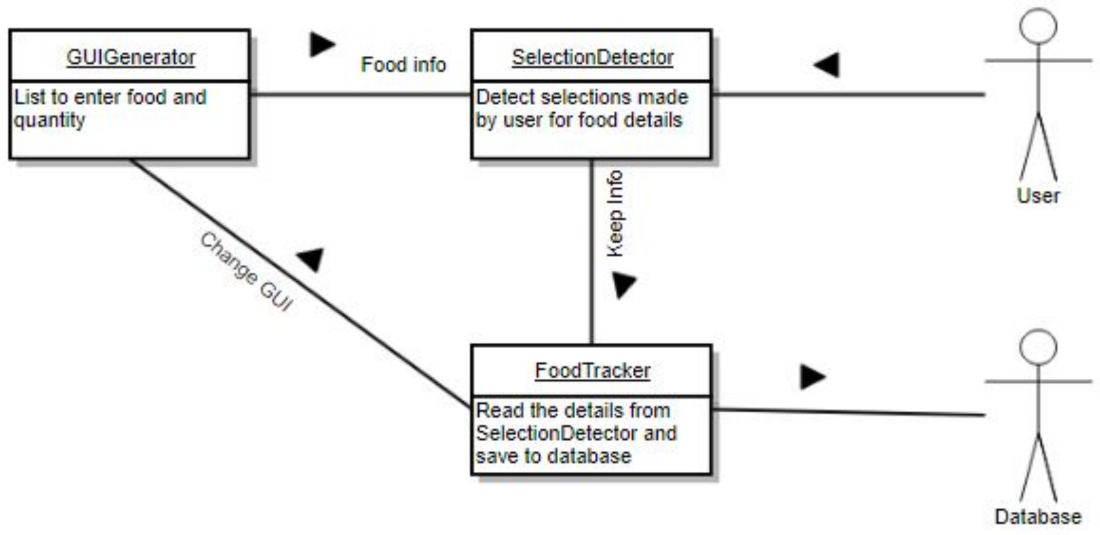

GUIGenerator
List to enter food and quantity
Food info
SelectionDetector
Detect selections made by user for food details
User
Keep Info
Change GUI
FoodTracker
Read the details from SelectionDetector and save to database
Database

Figure 26: 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: Smart Alarm**
Pre conditions	Monitor sleep button = “pressed”
Post conditions	Dismiss button = “pressed” and user is directed to Recommendation screen

Table 53: System Operation Contracts for Use Case 8: Smart Alarm

Operation	**UC9: Eat**
Pre conditions	Drop down menu with food items
Post conditions	Consumed food and quantity stored in database

Table 54: System Operation Contracts for Use Case 9: Eat

Operation	**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 10: 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;

}
```

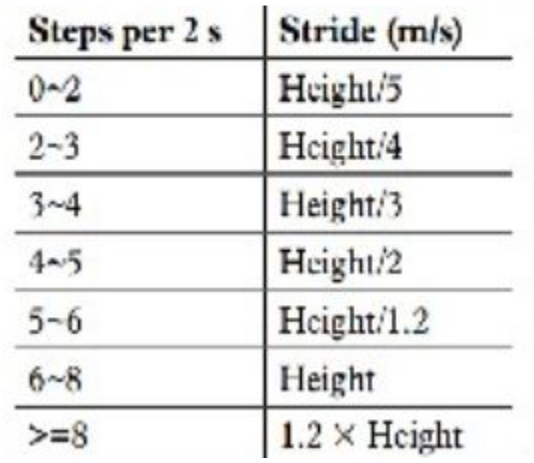
First, we have to make sure we sample the accelerometer frequently enough. Then we're going to have to make sure that we are calculating correctly about what your threshold should be. This is going to require a lot of trial and error. We will graph out what the length is over time and using those values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

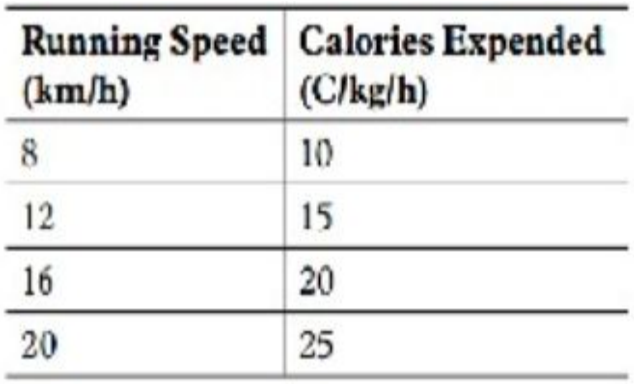
Speed parameter:

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 27: 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 sensor's 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 hyperplane 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 hyperplane that would give the best accuracy, which can be done by treating the hyperplane as a convex optimization problem and finding the optimum set.

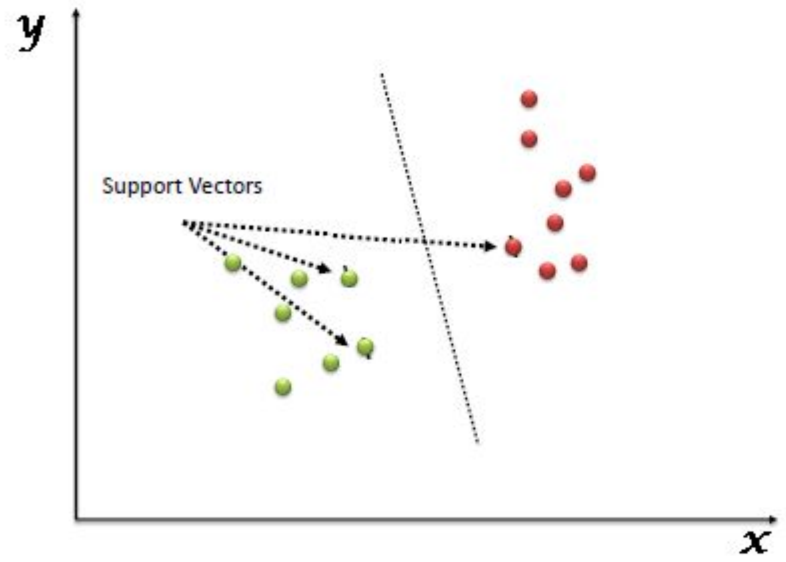


Fig 28: Support Vector Machine Classifier

7. 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.

7.1 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 56: 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 57: 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. 15, 2017**

Table 58: Project Deadlines

7.2 Gantt Chart

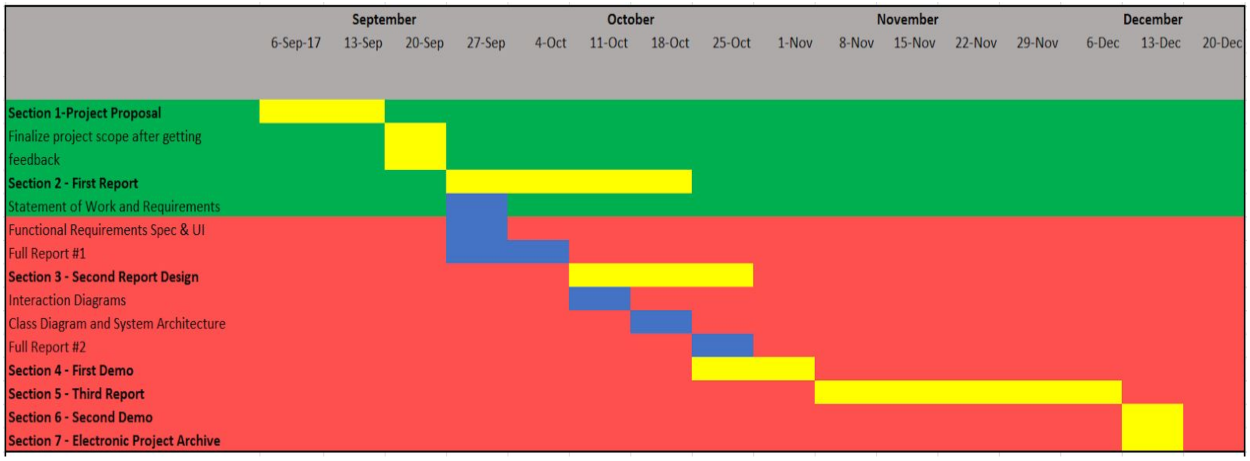


Fig 59: Gantt Chart

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/