

Health Monitoring Application

Group #2

Report #1, Part 1 & Part 2 & Part 3

Project URL : <https://github.com/vpranathy/Health-Monitor>

<https://yuyangchen0122.github.io/Health-Monitoring>

<Softenggroup2healthmonitor-env.aabwmssgtz.us-east-1.elasticbeanstalk.com>

Aniket Anilkumar

Yuyang Chen

Divyaprakash Dhurandhar

Zihao Ding

Malay Shah

Pranathy Veldandi

Table of Contents

0. INDIVIDUAL CONTRIBUTION BREAKDOWN.....	2
1. CUSTOMER STATE OF REQUIREMENTS.....	3
a. Problem	
b. Solution	
i. Music	
ii. Database	
iii. System Architecture Diagram	
iv. Product Usage	
v. Product Ownership	
vi. Devices and Specifications	
2. GLOSSARY OF TERMS.....	9
3. SYSTEM REQUIREMENTS.....	11
a. Enumerated Functional Requirements	
b. Enumerated Nonfunctional Requirements	
c. On-Screen Appearance Requirement	
4. FUNCTION REQUIREMENTS SPECIFICATION.....	16
a. Stakeholders	
b. Actors and Goals	
c. Use Cases	
i. Casual Descriptions	
ii. Use Case Diagram	
iii. Traceability Matrix	
iv. Fully Dressed Descriptions	
d. System Sequence Diagrams	
5. USER INTERFACE SPECIFICATIONS.....	28
a. Preliminary Design	
b. User Effort Estimation	
6. DOMAIN ANALYSIS.....	34
a. Domain Model	
i. Concept Definitions	
ii. Association Definitions	
iii. Attribute Definitions	
iv. Traceability Matrix	
b. System Operation Contracts	
7. Plan of Work.....	39
8. REFERENCES.....	40

0. Individual Contribution Breakdown

Task	Aniket	Yuyang	Divyaprakash	Zihao	Malay	Pranathy	Total
Customer State of Requirements	16.67 %	16.67%	16.67%	16.67 %	16.67 %	16.67%	100.00 %
Glossary of Terms	16.67 %	16.67%	16.67%	16.67 %	16.67 %	16.67%	100.00 %
System Requirements	16.67 %	16.67%	16.67%	16.67 %	16.67 %	16.67%	100.00 %
Functional Requirement Specifications	16.67 %	16.67%	16.67%	16.67 %	16.67 %	16.67%	100.00 %
User Interface Specifications	16.67 %	16.67%	16.67%	16.67 %	16.67 %	16.67%	100.00 %
Domain Analysis	16.67 %	16.67%	16.67%	16.67 %	16.67 %	16.67%	100.00 %
Plan of Work	16.67 %	16.67%	16.67%	16.67 %	16.67 %	16.67%	100.00 %
Project Management	16.67 %	16.67%	16.67%	16.67 %	16.67 %	16.67%	100.00 %
References	16.67 %	16.67%	16.67%	16.67 %	16.67 %	16.67%	100.00 %

1) Customer Statement of Requirements

1.1 Problem

It is a well known fact that a healthy lifestyle depends on three main factors - diet, sleep and exercise. While diet may be dependent on additional factors like income and availability of different produce, sleep and exercise can and should be controllable. Yet, this is not the case due to the fast paced lives people lead and factors like stress, fatigue and lack of motivation acting as barriers.

Lack of education about proper fitness is a widespread problem. Many people in the country would like to exercise and stay in shape, but only a small subset of those people know how to monitor their health in a way that allows them to stay fit. There are several methods out there which people can use to get the proper information; tools such as fitness blogs, the President's Council on Fitness, Sports, and Nutrition, and the classic visit to the doctor's office are all excellent examples. However, many people don't know about those methods or choose not to utilize them, and they do their body a disservice by performing exercises that could be detrimental to their health. With information like this readily available to exercisers, it can be hard to find correct information. And even if one does find correct information, he must check to see if that information applies to a person with his body shape and size. The general problem of finding correct exercise information is that there is no set standard; there is no "one size fits all" set of guidelines which one can follow to have an effective workout. Everybody's body responds differently to different exercises, so the best that the medical community can do is to provide a set of recommendations for people of the most average body type. While this set of recommendations is good in the general, they will never tailor to the needs of one's body and workout. Also, people only have a limited amount of time on their hands to exercise. This is due to the fact their schedule can limit them.

Of all the different metrics for measuring the quality of one's fitness, heart rate is the most important factor in determining whether a workout was effective. Monitoring one's heart rate is useful because it determines whether the exerciser is performing his exercise safely as well as successfully. Experts recommend that one's target heart rate during exercise should be between 60-85% percent of the maximum heart rate, and that anything higher than 85% increases cardiovascular and orthopedic risk to the exerciser. Naturally, the target heart rate varies for people of different ages, so one should always take this into account before starting a fitness regimen. Heart rate is a significant, if not the most important, factor in determining whether a workout was done correctly and effectively, and it must be monitored closely in order to prevent injury.

Unfortunately, there are people who don't know how to correctly monitor their heart rate, and they mistakenly create a certain fitness plan based on wrong information and end up not optimizing their workout. They go to the gym, run on the treadmill at a

light pace, and consider that enough to maintain their health. They do not check their heart rate and make sure they are in the safe region of activity. This critical lack of measurement affects the entire workout. For an exercise to be effective, one must maintain a heart rate that is within the target range for an extended period of time. If not, the exerciser either puts himself at risk of injury or completes a workout that does very little to improve his fitness. Some use exhaustion and soreness after a workout as a judge of an effective workout. Although these methods do give an indication as to how effective the exercise was, they do not provide an insightful and accurate description of one's health. As a result, these people continue bad habits and routines that hinder their progress to stay fit; in fact, they may not be even making progress. A solution to the problem of uninformed exercise must have three main components; it must include all relevant medical data such as heart rate information, create a fitness plan that fits relatively well to the client's body, and provide the client with feedback about the effectiveness of his workout. Once all these components come together, the client will be able to correctly monitor his health during exercise and get the most out of his workout.

Another aspect of a healthy life that is hard to maintain is a good night sleep. Good sleep..

..can eliminate fatigue and resume physical strength, protect the brain, restore energy, enhance immunity, anti-aging and promote longevity. However, many people are suffering insomnia and other sleep related diseases. People are unable to have uninterrupted deep sleep due to stress or unhealthy lifestyle. A good sleep can be quantified and detected based on ECG and heartbeats. When people have a good sleep, their recorded heartbeats are stable without abnormal pulses.

Scientifically there are five stages of sleep:

Stage 1: Beginning of sleep, relatively light stage of sleep, can be considered as a transition

period between awake and sleep. It lasts about 5 ~ 10 mins.

Stage 2: It lasts about 20 mins, heart rate begins to decrease.

Stage 3: Transition period between light sleep to very deep sleep.

Stage 4: Stage 4 is a deep sleep that lasts approximately 30 mins, bedwetting and sleepwalking

usually happen at the end of stage 4 sleep.

Stage 5: Most dreaming occurs during this stage, known as Rapid Eye Movement(REM) sleep.

During this stage, body system become more active, and heart rate is supposed to rise up. On an average, we enter the REM stage approximately 90 mins after falling asleep. The lasting time of REM stage might get longer with each sleep cycle, up to an hour as sleep progresses. In this modern day and age, an ideal sleep cycle has become a myth

due to increased pressures and stress levels. People do not have the required resources to analyse and improve their sleep cycle. As a whole, healthy lifestyles and activities in general are hard to maintain.

1.2 Solution

It has now been well established that heartbeat rate is an essential indicator of an effective workout and a good sleep cycle and in general, an indicator of a healthy lifestyle. Our proposed solution is a web application and a mobile application integrated with a pulse sensor. This will allow us to record and visualize user data and give the user information on how to improve his/her lifestyle. The proposed application would consist of various different aspects, from basic exercise statistics to the use of music to regulate heartbeat to provide better workouts and sleep cycles. Future work includes expansions of the application, including implementations of machine learning to let the app be unique to each user. To generate the huge amount of data required for it, the user would have to sleep with the pulse sensor on their finger and the database connected to our application would thus get their heart rate fluctuations over their entire sleep period. After a while, the data acquired would be enough to perform machine learning on it to give helpful suggestions to improve sleep cycles or play soothing music to help the user relax.

1.2.1 Music

Music and healing once went hand in hand. The Chinese character for medicine includes the character for music. In ancient Greece, music was used to ease stress, promote sleep, and soothe pain. Native Americans and Africans used singing and chanting as part of their healing rituals. Music can powerfully evoke and modulate emotions and moods, along with changes in heart activity, blood pressure (BP), and breathing. Heart rate can be influenced by factors like age, fitness, having a cardiovascular disease, emotions, body-size etc. The heart rate also varies according to the activity the person is involved in. Heart rate acts as a gauge to judge how well the person is involved in that activity. Hence, heart rate should be modulated to get the best out of the activity. Different music tempo can be used to modulate the heartbeat of the person. For example, if the person wants to workout, he can reach optimum heart rate by listening to music with high tempo. Similarly, the tempo varies with the activities like meditating, studying and sleeping.

1.2.2 Database

In this project, we will mainly use the AWS RDS database. First of all, this system will allow new user to sign up a new account, and the account information will be stored directly in database. Whenever existing users log in to the system, the system will detect whether their username and password can match the data to the database. If they match, the system will allow users to log in to the system; otherwise, users will not be allowed to log in.

Users will want to monitor their personal health status, so our project will allow the user to view his heartbeat data graph directly on his phone. This eliminates the inconvenience of having the user log in to a personal account on a website to view his data, because everything he needs will be on the phone itself. But of course, they are allowed to view the data on our web-based system. All the data collected from Arduino will be stored on the database, and the system will perform the necessary database calls to retrieve that data. That data will be processed and formatted into different graphs that will display the correlation between music and heart rate.

1.2.3 System Architecture Diagram

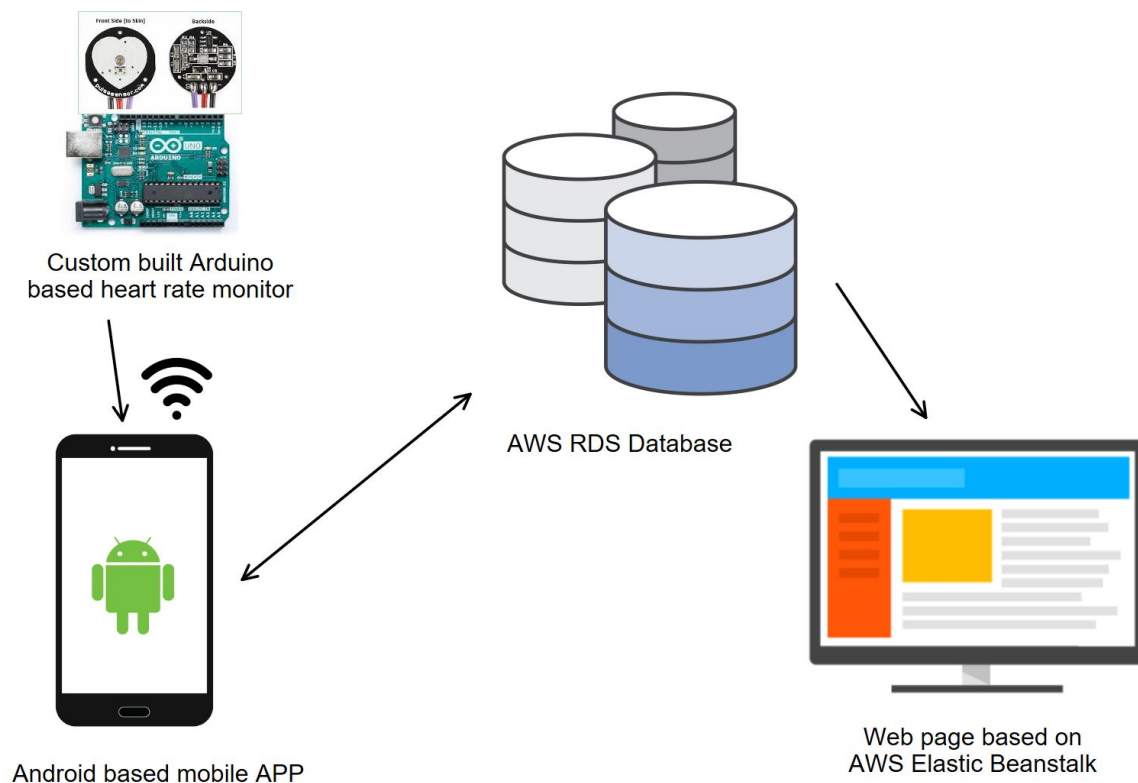


Fig: System Architecture

The mobile application is the core part of our system since it needs to upload the data from the heart rate monitor and present both visual and music feedback to the user. The mobile application part uses the Model-View-Controller architecture. The Arduino heart rate monitor collects the heart rate of the user and uploads the information to the Database via Android smartphone, those data will be processed with AWS Lambda. The graphs and other feedback produced by the AWS Lambda will send back to the APP to let app to play proper music and let the user get to know about their current health status.

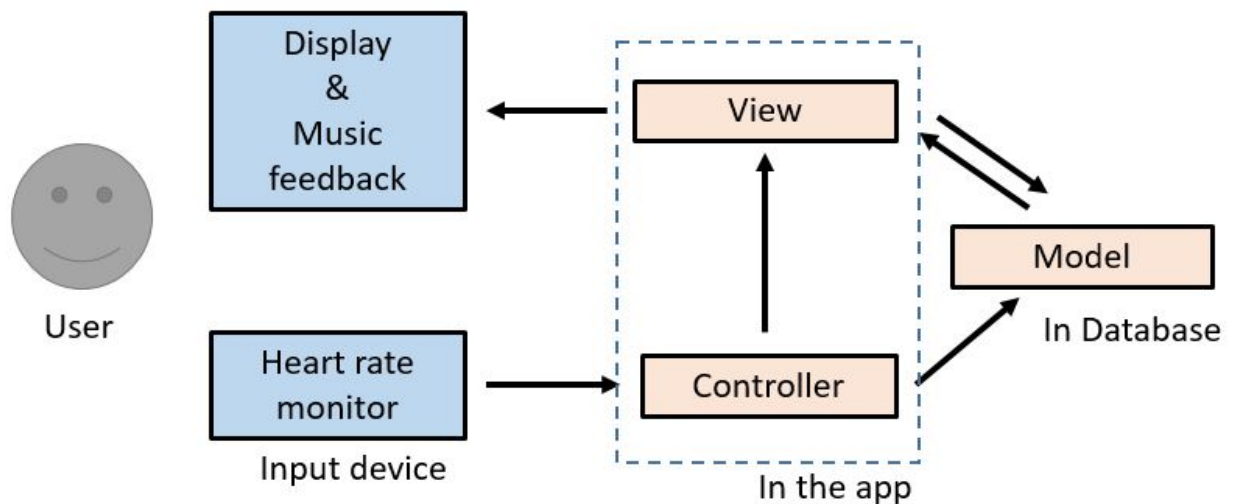


Fig: MVC Architecture for the Mobile APP

The webpage version of the application uses the client/server architectural style, which the data and code to run the application is stored on the server and clients can access their data using all kinds of web browsers. The webpage of the APP is mainly used to display user's histories. It may also provide customized health suggestion according to the user's data.

1.2.4 Product Usage

- The heart rate monitor should only be worn by the user on their finger and it will be linked to an Arduino board. It should be worn while it is in use - only while the user is exercising or the user is sleeping. Although it is safe to wear the heart rate monitor during other times, there will be no benefit unless the application is currently running.

- Users may choose to use the Heart Rate Monitor while not sleeping or exercising if they wish to adjust their heart rate for alternate reasons (possibly for playing video games or preparing for an exam).
- The user will run the android application, and then input a target heart rate. The software will then choose a song based on your current heart rate and begin to either raise or lower it. Once the target heart rate is obtained within a certain tolerance, the software will work to maintain this heart rate rather than increasing/decreasing it.
- Music will be selected from the user's own personal music library (which should be stored on the flash memory of the Android device) to either increase or decrease the user's heart-rate. Music will be played by our software.
- The software will select and play music according to the user's current heart rate in real-time as it receives information from the connected heart rate monitor.
- Music will be delivered through the headphone jack on the Android device or through any bluetooth device.
- Receive information on the songs that are listened to in relation to their usage of the Android device. (What songs were listened to, which songs were the most effective at changing their heart rate, etc.)

1.2.5 Product Ownership (tentative)

Our team will be divided into three smaller sub-teams of three, two and one individuals each, the pairings listed below. Each sub-team will be responsible for hardware, software and web development and provide a brief description of their work on a shared Google drive folder. They will also include the necessary UML diagrams and charts. Every week we will meet twice for 1-3 hours. During the meeting, we will have a specific agenda that primarily involves the week's progress and upcoming deliverable. Our discussion will probably be centered along the following questions: 1) What did you work on this past week? 2) What do you plan on working on next week? 3) Are there any changes that need to be made to the project? Every week, a different team member will take the lead for the next deliverable to ensure that everything is on time.

- Aniket, Pranathy and Malay will develop the Android application which will connect to the Arduino heart rate monitor and provide the user interface as well as all the other user features.
- Yuyang will work on a database that receives, stores, and processes the data from the Android device. The database will export the data to a CSV file, he will use python to do the data visualization based on the CSV file, it will provide users a more intuitive data analysis. He will also work on the web development to synchronize all information between web client and mobile client, making it easier for users to use the system.
- Divyaprakash and Zihao will be responsible for designing and programming the Heart Rate Monitor using the Arduino board as well as configuring it to connect via bluetooth to the android smartphone.

1.2.6 Devices and Specifications

- Heart Rate monitor: Arduino based custom built monitor
- Smartphone: Running Android 5.0+

2) Glossary of Terms

- **Arduino** - An open-source platform used for building electronic and software projects
- **API** - An application program interface (API) is a set of routines, protocols, and tools used for building the software applications
- **AWS** - Amazon web services (AWS) provides cloud computing services
- **AWS RDS Database** - A collection of information that is organized so that it can be easily accessed, managed, and updated
- **Beats per Minute (BPM)** - BPM is the amount of times that the heart beats given one minute of time
- **Encrypted** - A document is encrypted if it can not be accessed by the public and hence only allows specific users to read the data
- **Fetch** - A process in which a computer retrieves the program from its memory to carry out an action that is instructed by the user
- **Histogram** - A bar graph which shows how often something appears within a certain range

- **Interface** - A device or program enabling a user to communicate with the computer
- **Keystrokes** - A single depression of a key on a keyboard in order to measure work
- **Loading Screen** - Provides a plethora of fun facts to keep users engaged between transitions and educate them about general health
- **Module** - A number of distinct but interrelated units from which a program could build up
- **Physical Data** - All data pertaining to the user's step count, heart rate, and type of workout
- **Profile** - A unique set of information pertaining to a particular user, including but not limited to their avatar, lifetime points, and username
- **Query** - A question sent to the database for the purpose of extracting data
- **Sidebar** - A menu of all the modules displayed on the left side of the screen for easy access
- **Smartphone** - A smartphone is a handheld personal computer. It possesses extensive computer capabilities, including high-speed access to the Internet using both Wifi and mobile broadband.
- **System Admin** - Approves community module posts, group and challenge creation, and login
- **UI** - Stands for User Interface. Refers to the general appearance and functionality of the modules, and listens for the user's keystrokes
- **User Effort Estimation** - Refers to the number of keystrokes or mouse clicks needed to navigate and get to the actual context where the user needs to enter the data
- **Visitor** - An outsider who does not yet have an account that is able to create a new account from the "Sign Up" page on the default page of the app.
- **Widget** - An application, or a component of an application, that enables a user to perform a function or access a service

3) System Requirements

3.1 Enumerated Functional Requirements

ID	Priority Weight	Requirement
REQ-1	5	System should play the correct music according to the customer's current activity
REQ-2	4	System should retrieve data from arduino sensor
REQ-3	4	System should transfer data to mobile application
REQ-4	4	System should store data into database
REQ-5	5	System should transfer data to web application
REQ-6	3	System should let the user visualize their data by graphs
REQ-7	2	The user should be able to rank the songs
REQ-8	1	The user should have the ability to change the song if their activity changes
REQ-9	1	The user should have the ability to pause the song if their activity stops

3.2 Non-Functional Requirements

ID	Priority Weight	Requirement
REQ-10	5	The Android Application will be user friendly and require very little user navigation
REQ-11	4	The system shall run constantly, requiring little maintenance
REQ-12	4	The system shall present all information in an orderly manner
REQ-13	3	The system shall recover easily
REQ-14	3	The system will allow multiple users online
REQ-15	3	The system will prevent users from modifying existing system data

3.3 On-Screen Appearance Requirements

This section contains mock-ups of the Android and web application's user interface. This user interface is subject to change. The images presented below do contain essential information for the user. This information will be functionally developed by the team. The input displays the application name, the start button, and the previous session. There is a study mode, sleep mode, and work mode. The minimum heart rate, maximum heart rate, and music being played is also listed within the application.

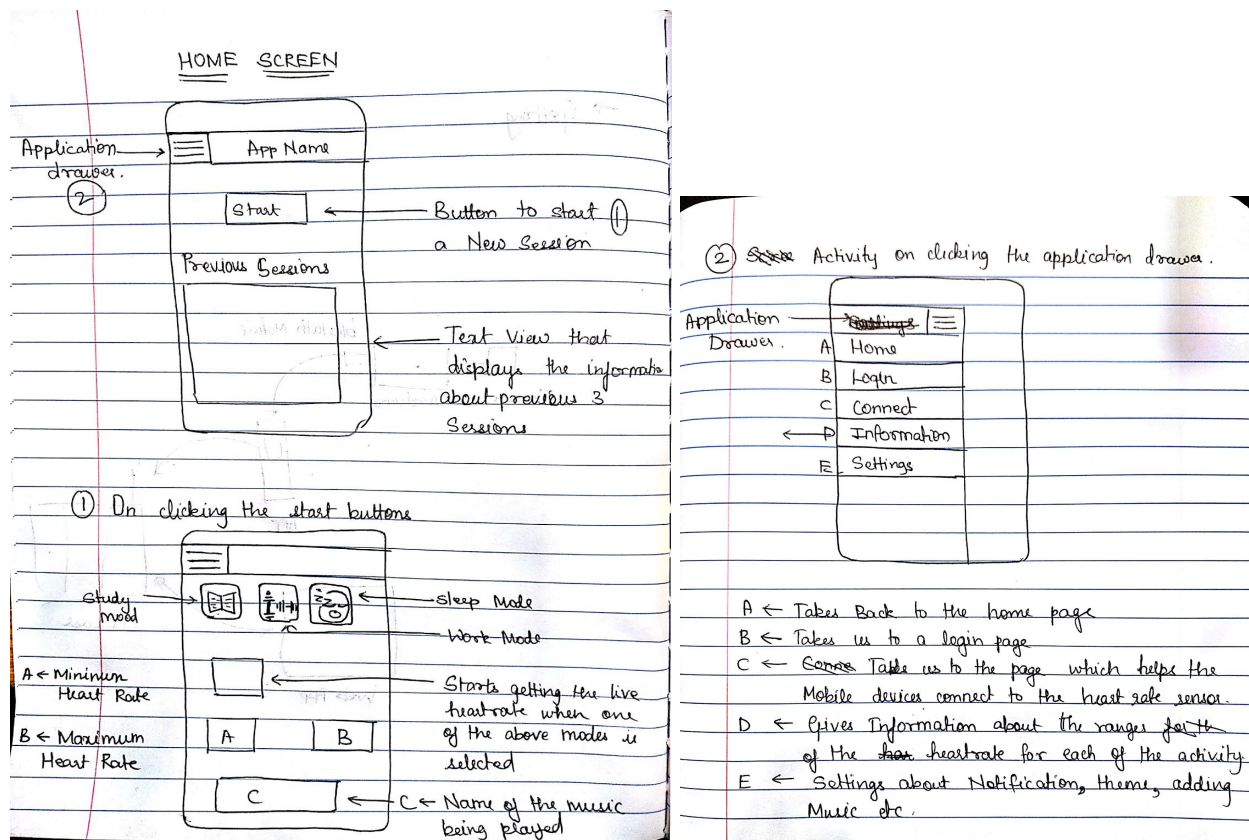


Fig: A basic sketch showing the UI of the Android App

Web-based Dashboard page

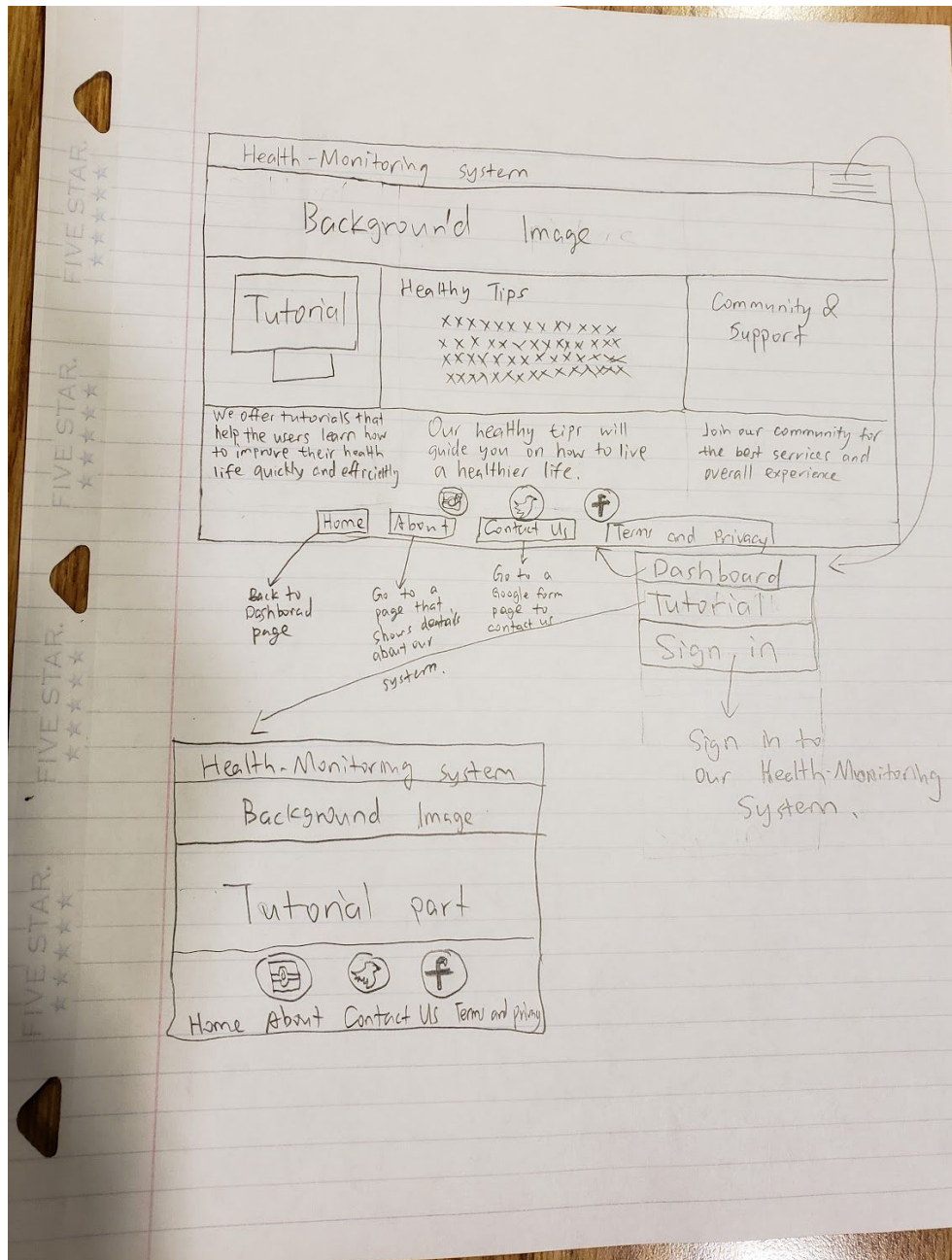


Fig: A basic sketch of the UI of Web Application for data visualisation

The Webpage will be used for data visualization. Graphs depict the heart rate of the user during an activity. Another graph will display the mean of the heart rate for the last n times the same activity was conducted.

Web-based Login page

Each user will have to login with their created username and password to access their data and health suggestions.

Login

Username

Password

Login

Not yet a member? [Sign up](#)

Register

Username

Email

Password

Confirm password

Register

Already a member? [Sign in](#)

Home Page

You are now logged in

Welcome **yuyangchen0122**

[logout](#)

Fig: User login page for web application

4. Functional Requirement Specification

4.1 Stakeholders

Stakeholders include individuals and organizations which are interested in the completion and use of a given product. The amount of stakeholders and different types of stakeholders relies on the versatility and ease-of-use of the product in question. Due to this software's very simple interface and design, stakeholders may include users of all ages and multiple types of organizations who are interested in obtaining easier sleep or a more energetic workout. Examples of potential stakeholders include:

- **Administrator:** The administrator will look into the system to ensure the smooth functioning of the online leaderboard, rewards and the feedbacks from every-user.
- **Cardiologists:** The cardiologists will use this application to get the rest heart-rate and comment on the health of the heart based on the physical structure of the patient.
- **Fitness-instructor:** The instructor can monitor the heart-rate of the of the students during their fitness activity and suggest changes in the warm-up regime to get the heart-rate to the optimal level.
- **Developer:** The developers of the application can work on the data to develop machine learning algorithms which can help to improve the music suggestions, advice changes in daily activities and consult sleep hours.

4.2 Actors and Goals

Actors can be defined as people or devices that will directly interact with the product, and can also be loosely labeled as either "initiators" or "participators". These actors will have a specific goal with the given product, which is what the actors are attempting to achieve by interacting with the system. Actors and their respective goals are:

<i>Actor</i>	<i>Actor's Goal</i>
User(initiator)	To increase heart rate for exercising
User(initiator)	To decrease heart rate for sleeping
User(initiator)	To analyze health information from given graphs

Doctor/Fitness instructor	To consult the changes that should be brought to improve health of the heart
DataBase	A repository of the heart-rate data from previous activities.
Music Suggestion Algorithm	An algorithm that asks users music preference and suggests music from the same genre

This product is one which only requires the interaction of one human actor, the user of the product. While there is the potential for other humans to interact with the user's health information which is produced, only the user himself is considered an actor. The heart monitoring device is a participating actor worn by the user to monitor information and relay the information via Bluetooth back to the smartphone which is running the application. The one exception is that our heart monitoring device may be an initiating actor and notify the user if his/her heart rate is abnormally high or low. In this case, the user would be the participating actor.

4.3 Use Cases

i. Casual Description

UC#1 Collecting Health Data from Arduino

The user data shall be collected and then transferred to the database which will be linked to the web and mobile application.

UC#2 Displaying Heart Rate Data

The heart rate data will be displayed through the web and mobile application.

UC#3 Visualization of Data through a Graphical Representation

The data shall be presented to the user through graphs and be distinguishable through a legend.

UC#4 Music Selection & Playback

The system shall recommend music to the user dependent on their activity and playback that music accordingly.

UC#5 Ranking Songs

Users will give their feedback about the songs after every activity. This feedback will be

in the form of points out 10. This pointing system will help the application to rank the songs.

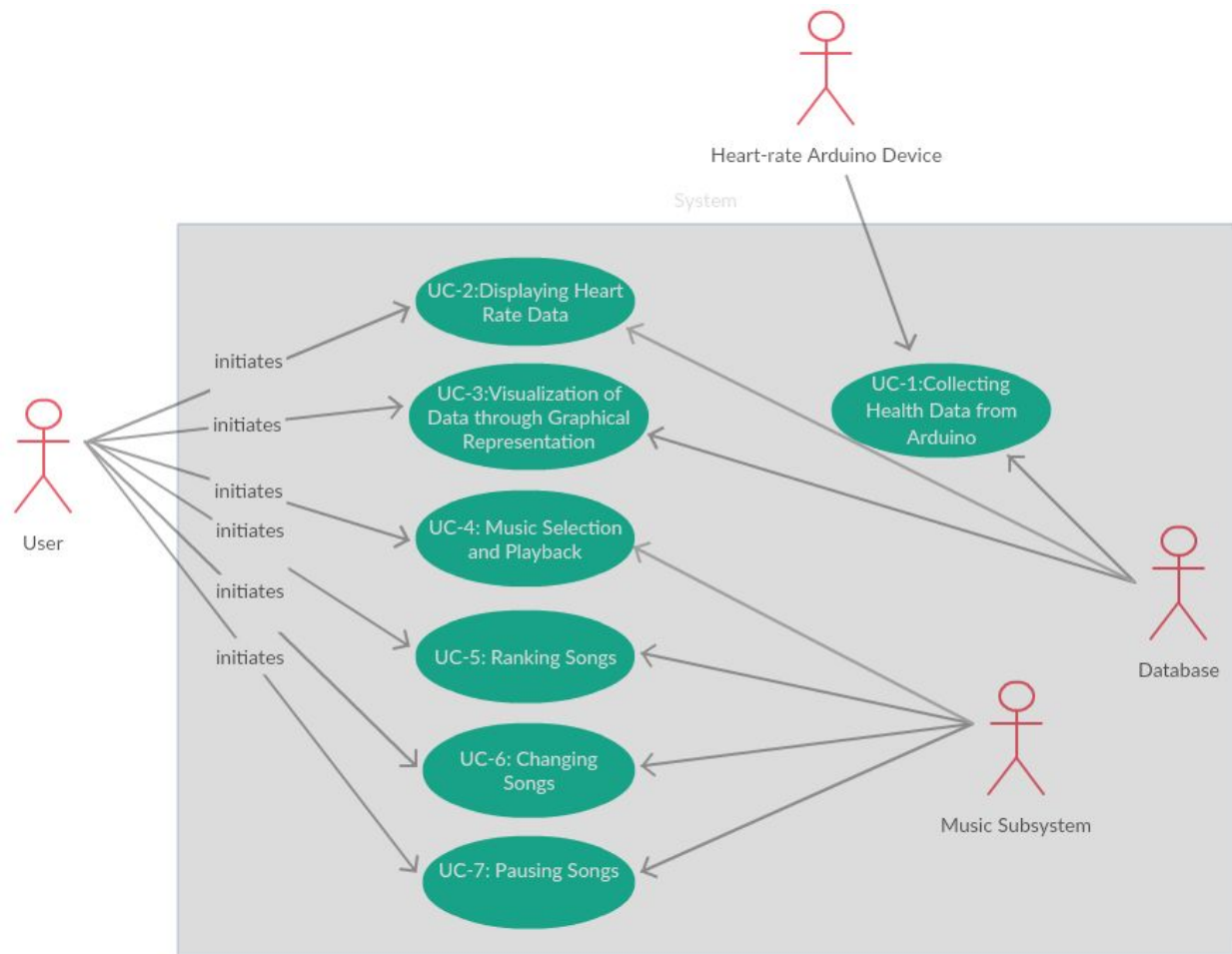
UC#6 Changing Song

Users will have the ability to change the song within the selected genre to their liking. This will help the user reach their desired heart rate.

UC#7 Pausing Song

Users will have the ability to pause the song if they choose to stop the activity or take a break from the activity they are currently performing.

ii. Use Case Diagram



iii. Traceability Matrix

The Traceability Matrix allows the reader to cross the function and non-functional requirements described earlier with the use cases. This will demonstrate which use cases fulfill each requirement.

	UC#1	UC#2	UC#3	UC#4	UC#5	UC#6	UC#7
REQ-1				X			
REQ-2	X						
REQ-3	X	X	X				
REQ-4	X	X					
REQ-5	X	X					
REQ-6			X				
REQ-7					X		
REQ-8						X	
REQ-9							X

iv. Fully Dressed Descriptions

Use Case UC-1:	Collecting Health Data from Arduino
Related Requirements:	REQ-2,REQ-3,REQ-4, & REQ-5
Initiating Actor:	System
Actor's Goal:	To collect the data and transfer it to the database
Participating Actors:	
User Preconditions:	The system collects the data and transfers it to the database.
Postcondition:	User will be able to view data.
Failed End Condition:	Unable to collect data please try again.
Flow of Events for Main Success Scenario: -> 1.Users performs activities and to generate data. <- 2.System collects the data. -> 3.System waits to transfer data. <- 4.System transfers the data to the database.	
Flow of Events for Extensions: 4(a) No data acquired. <- 1.System would display a error page saying the data cannot be acquired due to insufficient data.	

Use Case UC-2:	Displaying Heart Rate Data
Related Requirements:	REQ-3, REQ-4 & REQ-5
Initiating Actor:	Users
Actor's Goal:	To display users' heart rate data
Participating Actors:	
System Preconditions:	The website displays at the Dashboard Page.
Postcondition:	Users get their health index dashboard by month or day.
Failed End Condition:	Unable to show the health index please try again
<p>Flow of Events for Main Success Scenario:</p> <p>-> 1. The system reads data from the database. <- 2. System connect to the mobile applications. -> 3. System waits to connect. <- 4. System displays the heart rate data on the Dashboard Page.</p> <p>Flow of Events for Extensions:</p> <p>4(a) No data acquired. <- 1. System would display a error page saying the data cannot be acquired from database.</p>	

Use Case UC-3:	Visualization of Data through a Graphical Representation
Related Requirements:	REQ-3, REQ-6
Initiating Actor:	System
Actor's Goal:	To display graphs that shows the relationship between heart rate and other factors to users, to explain how these factors affect heart rate
Participating Actors:	
User Preconditions:	The website displays at the Dashboard Page.

Postcondition:	Users get their Heart rate analysis chart by month or day.
Failed End Condition:	Unable to show the Heart rate analysis please try again
<p>Flow of Events for Main Success Scenario:</p> <p>-> 1. The system reads data from the database</p> <p><- 2. System waits to transfer data.</p> <p>-> 3. The system generates heart rate data analysis graphs.</p> <p><- 4. System waits to transfer graphs to Web page, which will be displayed to users.</p>	
<p>Flow of Events for Extensions:</p> <p>3(a) Heart rate analysis chart generation failed</p> <p><- 1. System would display a error page saying Heart rate analysis graph generation failure based on reading data error</p>	

Use Case UC-4:	Music Selection & Playback
Related Requirements:	REQ-1
Initiating Actor:	System
Actor's Goal:	To select the music of their choice and play it at their choice.
Participating Actors:	
User Preconditions:	The application shall allow the user to play music.
Postcondition:	The user will play their music for their current activity.
Failed End Condition:	Unable to play music for the user.
<p>Flow of Events for Main Success Scenario:</p> <p>-> 1. The system allows the user to select their song choice</p> <p><- 2. System waits for users choice</p> <p>-> 3. The system plays the users choice</p> <p><- 4. User continues their activity</p>	
<p>Flow of Events for Extensions:</p> <p>3(a) Failure of Music Selection</p>	

<- 1.System would display a error page saying music selection failure based on the song list of the user.

Use Case UC-5: Ranking Songs

Related Requirements: REQ-7

Initiating Actor: User

Actor's Goal: To pick their favorite songs for their favorite activities.

Participating Actors:

System Preconditions: The user has ranked songs within the application..

Postcondition: The ranked songs will be available for use

Failed End Condition: The user will not be able to play their ranked songs.

Flow of Events for Main Success Scenario:

-> 1. The system allows the user to select their ranked song choice

<- 2.System waits for users choice

-> 3.The system plays the ranked song

<- 4.User continues their activity

Flow of Events for Extensions:

3(a) Failure of Ranked Music Selection

<- 1.System would display a error page saying music selection failure based on the song list of the user.

Use Case UC-6: Changing Songs

Related Requirements: REQ-8

Initiating Actor: User

Actor's Goal: To change the music of their choice and play it at their choice.

Participating Actors:

System Preconditions: This application allows the user to change the music being played

Postcondition: The application changes the music that the user listened to before to broadcast the new music that the user selected.

Failed End Condition: The application cannot change the music that the user has previously listened to and continues to play the music that the user wants to change.

Flow of Events for Main Success Scenario:

-> 1. The system allows the user to change their song choice

<- 2. System waits for users choice

-> 3. The system stop playing the previous song and change to play the user's new choice

<- 4. User continues their activity

Flow of Events for Extensions: System would display a error page saying music selection failure based on the song list of the user.

Use Case UC-7: Pausing Songs

Related Requirements: REQ-9

Initiating Actor: User

Actor's Goal: To pause the music that is playing

Participating Actors:

System Preconditions: This application allows the user to pause the music being played

Postcondition: The application suspends the music being played and does not continue to play any music

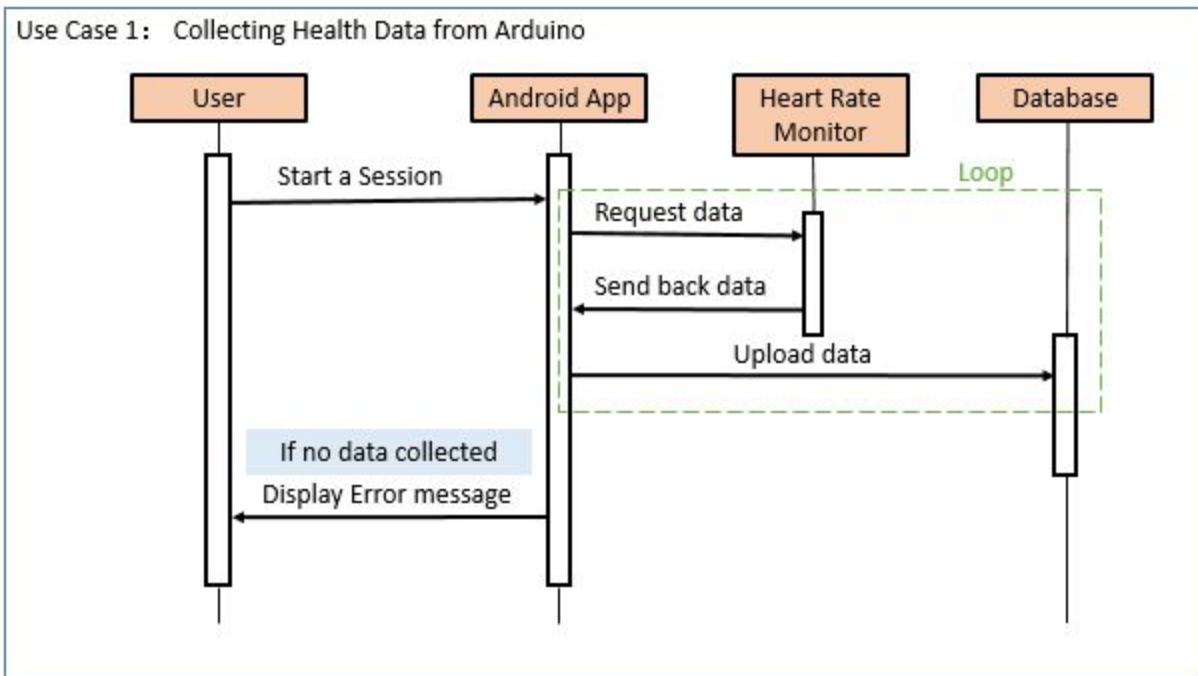
Failed End Condition: The application cannot pause music being played and continue to play music

Flow of Events for Main Success Scenario:

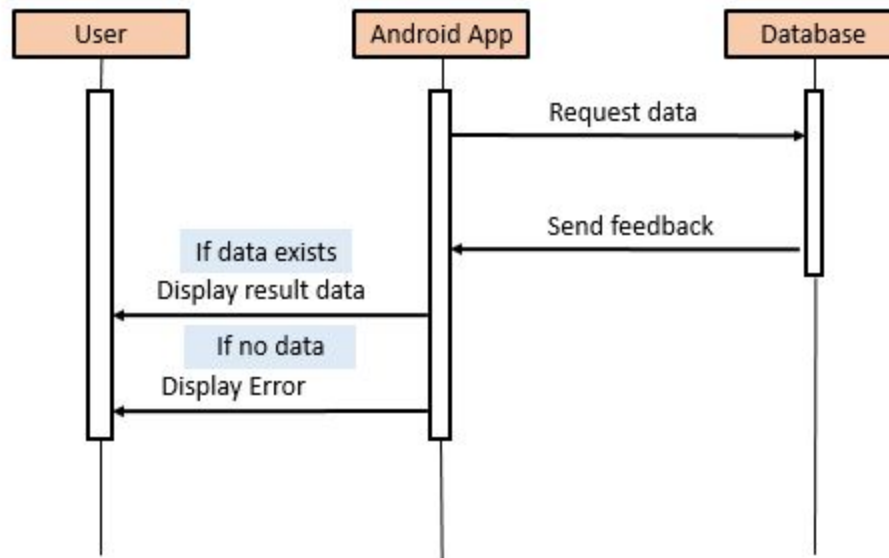
- > 1. The system allows the user to pause the song being played
- <- 2. System waits for users choice
- > 3. The system stop playing the music
- <- 4. User continues their activity

Flow of Events for Extensions: System would display a error page saying music pausing failure based on the error of system.

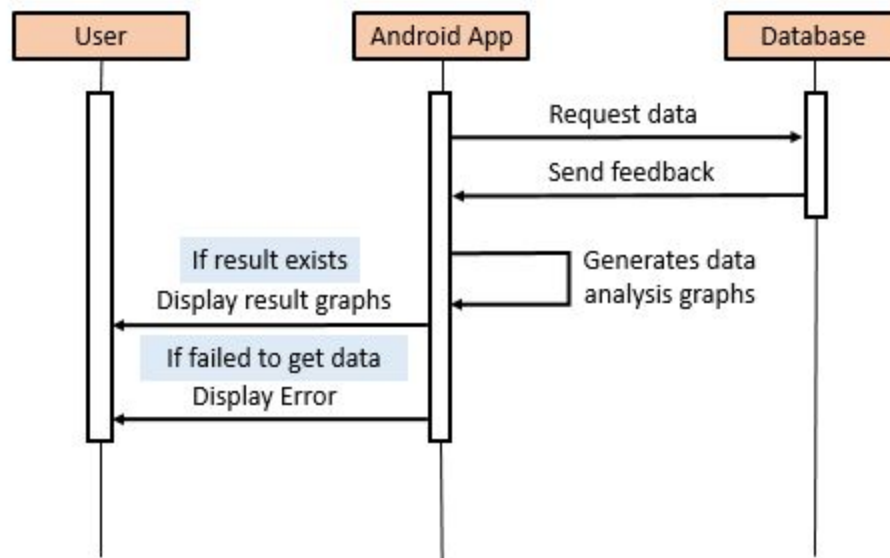
v. System Sequence Diagrams



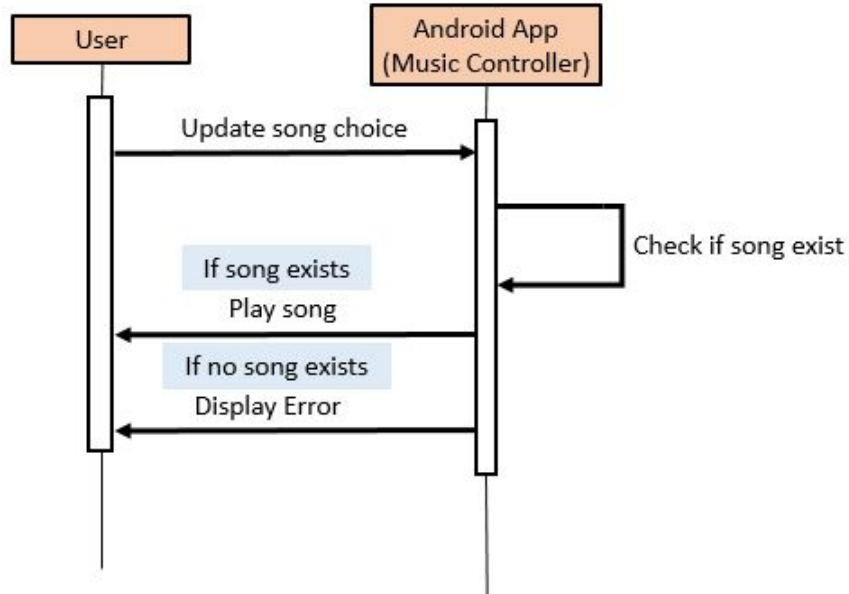
Use Case 2: Displaying Heart Rate Data



Use Case 3: Visualization of Data through a Graphical Representation




Use Case 4: Music Selection & Playback

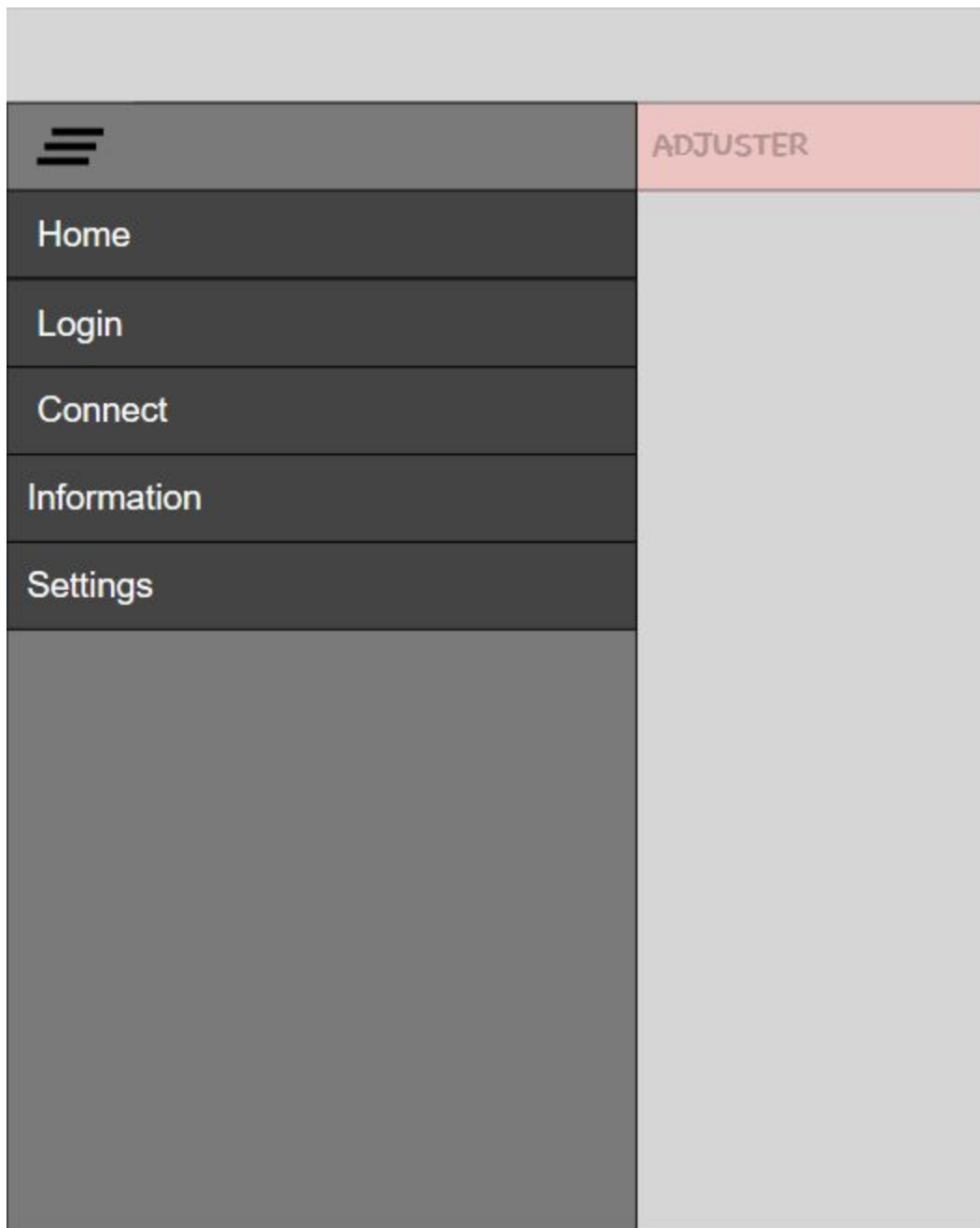


5. User Interface Specification

5.1 Preliminary Design

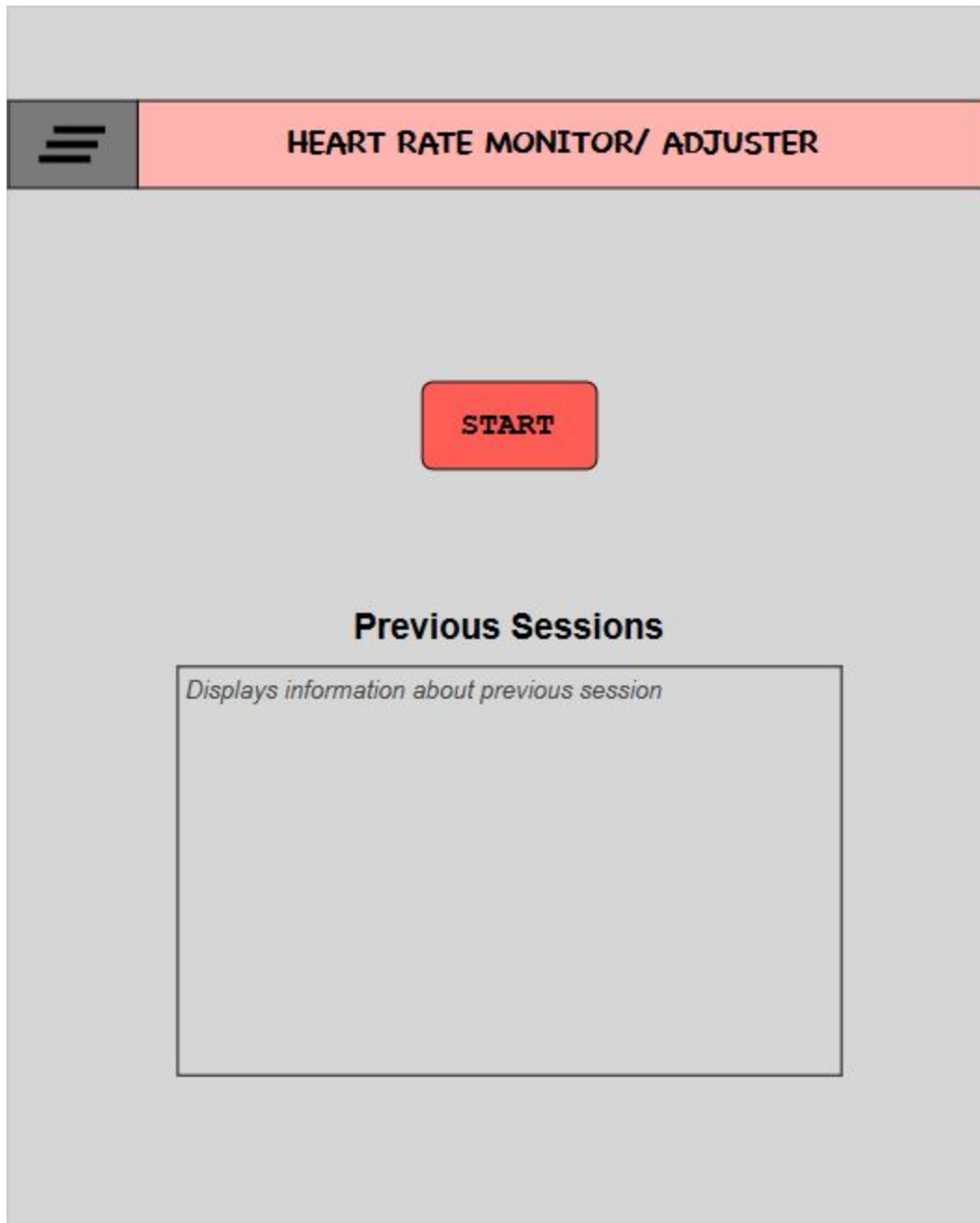
UC#1 Collecting Health Data from Arduino (2 clicks)

- 1) The user has to click on  icon at the top left corner of the screen
- 2) The user must then click on Connect to connect to Arduino via bluetooth

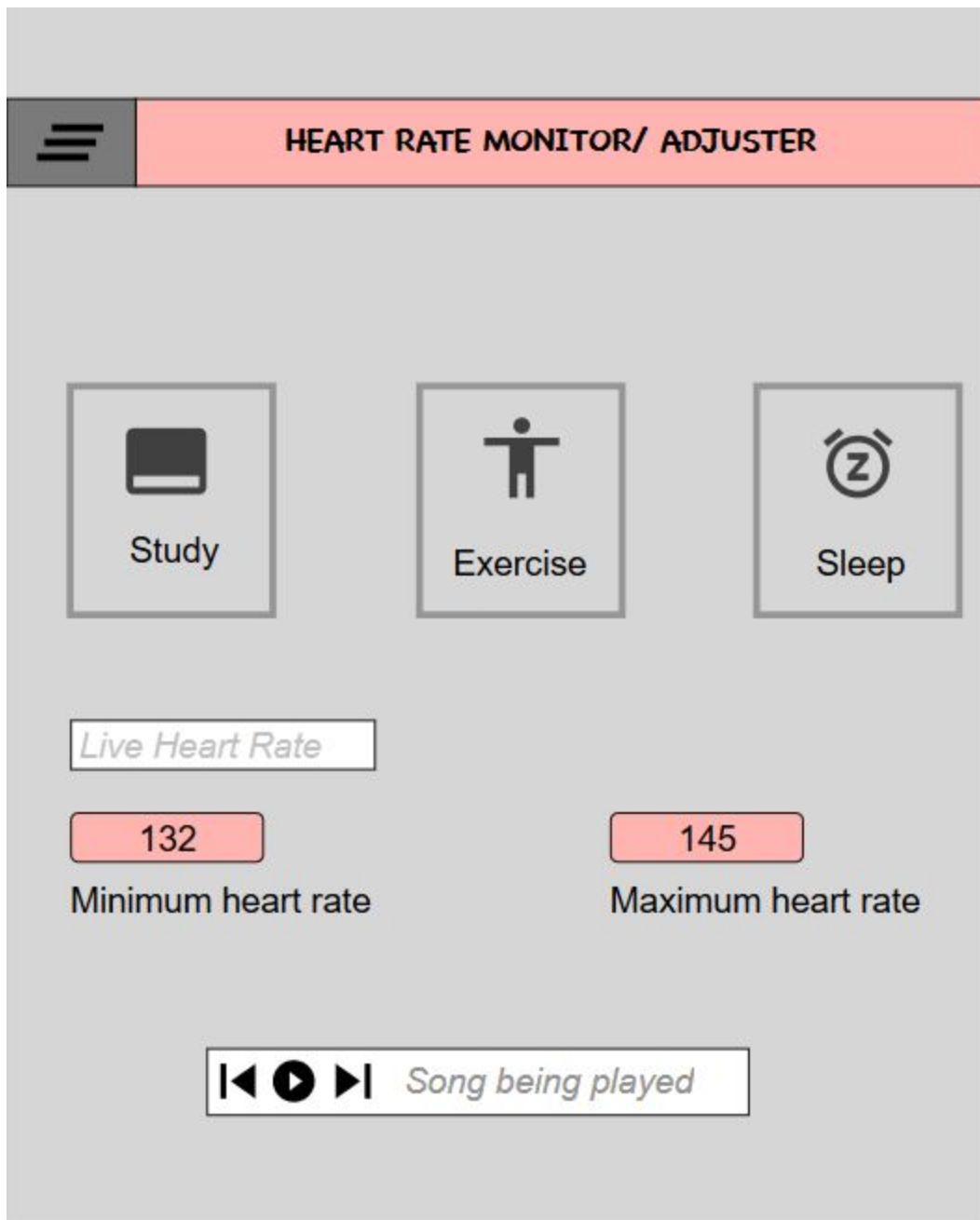


UC#2 Displaying Heart Rate Data (2 clicks)

- 1) The user must click on START on the home page in the app (1 click)




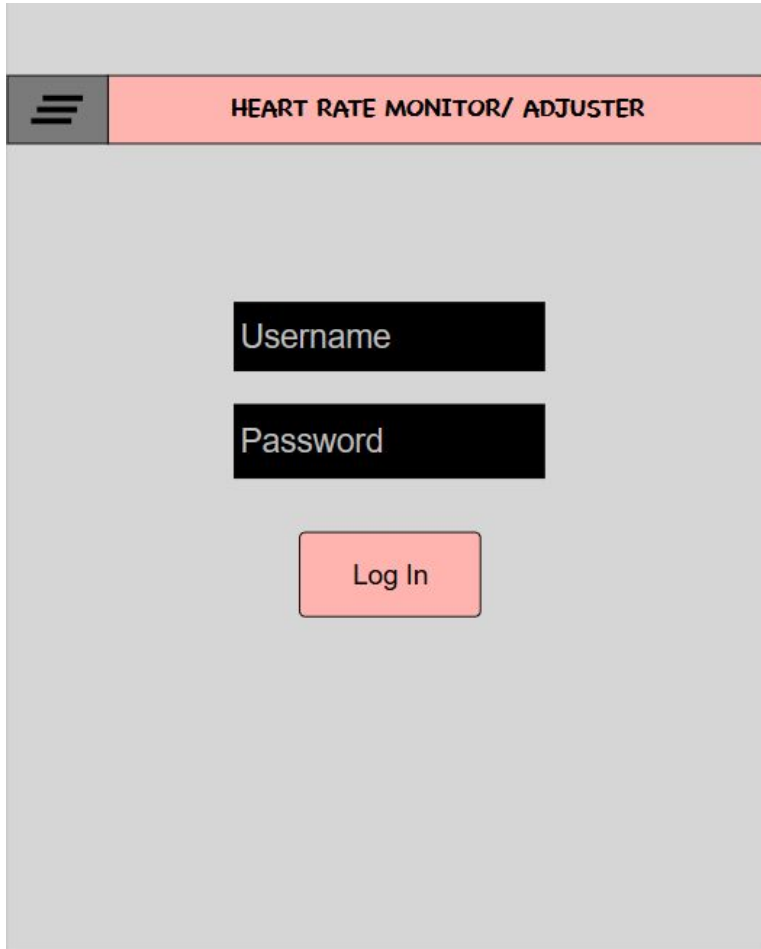
2) Then he/she must choose the mode they are using the app in (1 click)



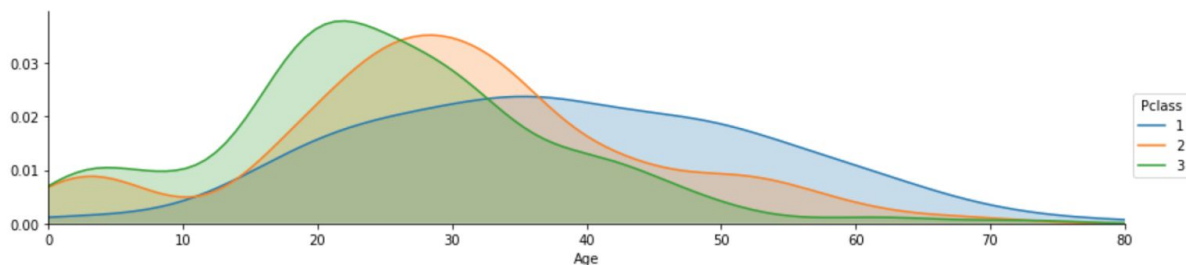
3) Live heart rate will then be displayed in a textView.

UC#3 Visualization of Data through a Graphical Representation


- 1) The user has to click on  icon at the top left corner of the screen. (1 click)
- 2) Then click on Login from the menu bar. (1 click)
- 3) The user must then enter their username and password and click Log In. (12-20 keystrokes and 1 click)

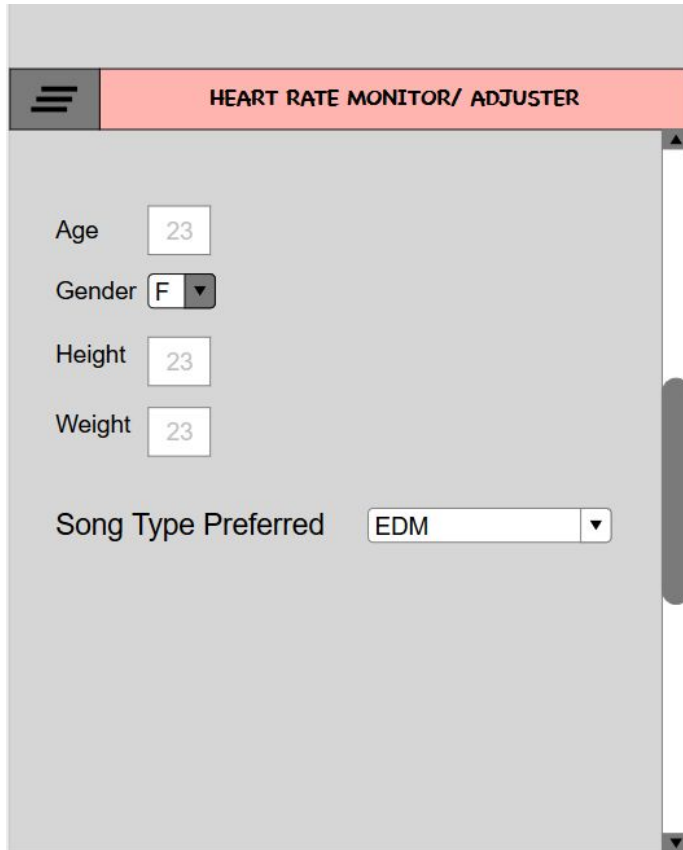


- 4) The user is then linked to their database and can view their Heart beat Data in a graphical representation. This is a similar graph and the actual graphs has to wait until the data is stored into the database, so it cannot be generated at this moment.



UC#4 Music Selection & Playback and UC#5 Ranking Songs (3 clicks)

- 1) The user has to click on  icon at the top left corner of the screen. (1 click)
- 2) Then click on Settings from the menu bar. (1 click)
- 3) The user can then click on Music Type preferred from the scroll bar to play a song and to rank favourite songs. (1 click)



UC#6 Changing Song and UC#7 Pausing Song: (3 clicks)

- 1) The user gets to the app page that displays live heart rate data by following UC#2 (2 clicks)
- 2) The bottom of the screen then has the music player being used to adjust heart rate. By clicking the Play/Pause button, music can be toggled on/off (1 click) and by pressing next/prev, the songs can be changed (1 click).



5.2 User Effort Estimation:

- 1) The User is greeted with a message which appreciates the last successful session with the details with a button which prompts “Continue”. On clicking this button the home activity opens up. (1- click)
- 2) On the home screen the user has 3 options:
 - a) **Start Button:** On one click the application switches to the activity screen where user can select the available activity options(1-click) and start the activity using the play button(1-click).So a total of 3 clicks to start the activity.(3-Clicks)
 - b) **History Button:** On just one click the application switches to an activity which displays the previous activities with their respective time period and the mean of the heart-rate within that time period. (1-click).
- 3) The application drawer has one-click options like Home, settings, information and login.

Of the above mentioned user estimation, roughly 85% of the click actions are for user interface Navigation and the rest is clerical data entry which varies from one user to the other.

6. Domain Analysis

a. *Domain Model*

- i. **Concept definitions:** To analyse the domain model, we first derive domain model concepts and corresponding responsibilities from the formerly defined system use cases. The table below lists all the domain model concepts and corresponding responsibilities.

Responsibility	Type	Concept
Pairing/Communicating with the Heart Rate Sensor	D	HR Manager
Retrieve the sensed data	D	Log Retriever
Musical Playback	D	Music Player
Logging tracks as they are played	D	Tracker
Queueing next track	D	Music Queue
Listen for user input	D	General UI
Recommend Target Heart Rate	D	Range suggester
Setting the Users Heart Rate	D	Rest Setter
Alert the user in case of danger	D	User Alerter
Graphically displaying User's info	D	User UI
Displaying music Information	D	Music UI
Displaying Current Activity Heart Rate Data	D	Workout View
Displaying previous Activity Heart Rate Data	D	History View
Graphically displaying relationship between heart rate data, workout data and music data	D	Data analysis View
User data store for Login pairing	K	User Store

Data Store for Activities	K	Activity Store
Data Store for music Metadata	K	Metadata Store
Data Store for Music Files	K	Music Store

- ii. Association definitions: Some of the concepts defined above as domain concepts have to work in certain pattern to finish some target. The table below gives the corresponding association definitions based on the defined domain concepts.

Concept Pair	Association Description	Use Case
User Info ↔ User data store	System pairing user's login information to allow user to login to system from user info store	data retrieval
music player ↔ metadata store	music player retrieves information about the current track from metadata store	data retrieval
history view ↔ workout store	history view retrieves data about previous workouts from the workout store	data retrieval
track logger ↔ music player	tracks played by music player are logged by track logger	data logging
music player ↔ track queuer	music player retrieves the next track from the track queuer	data retrieval
music player ↔ playback view	playback view displays information based on the data in music player	human data interface
rest setter ↔ HR manager	HRM manager retrieves user's current heart rate to set as resting	human data interface
user alerter ↔ HRM manager	HRM manager retrieves user's current heart rate and activates user alerter if in dangerous levels	human data interface
hrm manager ↔ general UI	general UI pairs and reports hrm status based on hrm manager	human data interface

heart beat view ↔ hrm manager	retrieves and displays heart rate data from the Heart-rate manager	human data interface
log retriever ↔ workout store	log retriever fetches logs from the workout store and used to display the graphs	data logging
music playbaker ↔ music store	music playbaker plays songs from the music store	data retrieval
Data analysis↔music store, workout store and activity store	Retrieves music data, workout data and heart rate data to do the data visualization for analyzing data.	data retrieval

iii. Attribute definitions

Concepts	Attributes	Attribute Definition
User Info	Data logging	Data logging has to with the storage or retrieval of logged data or the logging of data.
HRM Manager		
Log retriever		
Track logger		
Data Analysis		
Music Playbaker	Human data Interface	Human data interfaces deal with the interaction between the user and the data
Track queuer		
Rest setter		
Peak calculator		
General UI		
User Alerter		
Playback View		
Heart beat view		

Workout view		
History View		
User Data Store	Data storage	Data storage deals with the storage of data.
Workout Store		
Metadata Store		
Music Store		

Iv. Traceability matrix

	HRM	Log Retriever	M u s i c P l a y e r	T r a c k e r	M u s i c Q u e u e	G e n e r a l U I	R a n g e s u g g e s t e r	R e s t S e t t e r	U s e r A l t e r	U s e r U I	M u s i c U I	W o r k o u t V i e w	H i s t o r y V i e w	D a t a a n a l y s i s V i e w	U s e r S t o r e	A c t i v i t y S t o r e	M e t a d a t a S t o r e	M u s i c S t o r e
UC-1	X		X			X				X					X	X	X	
UC-2		X		X		X		X		X		X	X			X	X	
UC-3		X			X	X	X		X	X		X	X	X		X	X	
UC-4				X			X			X	X						X	
UC-5	X		X			X					X				X			X
UC-6	X		X			X					X				X			X
UC-7	X		X			X					X				X			X

6. Part B: System Operation Contracts

System operation contracts for the operations of the fully-dressed use cases.

Collecting Health Data from Arduino

1. Pre Condition: The system collects the data and transfers it to the database.
2. Post Condition: User will be able to view data.

Displaying Heart Rate Data

1. Pre Condition: The website displays at the Dashboard Page.
2. Post Condition: Users get their health index dashboard by month or day.

Visualization of Data through a Graphical Representation

1. Pre Condition: The website displays at the Dashboard Page.
2. Post Condition: Users get their Heart rate analysis chart by month or day.

Music Selection & Playback

1. Pre Condition: The application shall allow the user to play music.
2. Post Condition: The user will play their music for their current activity.

Ranking Songs

1. Pre Condition: The user has ranked songs within the application.
2. Post Condition: The ranked songs will be available for use.

Changing Songs

1. Pre Condition: The application allows the user to change the music being played.
2. Post Condition: The application changes the music that the user listened to before to broadcast the new music that user selected.

Pausing Songs

1. Pre Condition: This application allows the user to pause the music being played.
2. Post Condition: The application suspends the music being played and does not continue to play any music.

7. Plan of Work

Although we are a team of 5 students working on a project, we have divided ourselves into 3 sub-teams of three, two and one individuals. Each sub-team will be responsible for hardware, software and web development.

The software team will be responsible for the development of the Android application and the user interface. Different activities for the android application consists of the home screen activity, connection screen activity, select mode screen activity, login screen activity, settings screen activity and the information screen activity. After completion of report 3 i.e October 7, we plan on getting the activities working in one week and then preparing the layouts in 2 weeks. By the end of the 2nd week, we plan on getting the software requirements to connect the application to the arduino device.

The hardware team will be responsible for the development and testing of the heart-rate monitor. The heart-rate monitor consists of a pulse sensor, a bluetooth module and an arduino board. Each of these components is independant and requires separate testing. We achieved the preliminary testing of the pulse sensor by October 5. We hope to get the bluetooth module working by October 14. Complete integration with the android app is expected to be done after the 1st demo i.e 31st October.

The web-based database development consists of using the Amazon Relational Database Service. Amazon Relational Database Service (Amazon RDS) is a web service that makes it easier to set up, operate, and scale a relational database in the cloud. It provides cost-efficient, resizable capacity for an industry-standard relational database and manages common database administration tasks. We already have the AWS RDS database which is based on the MySQL. Our database is called "fall2018softenggroup2health". We have created the user's table using MySQL's commands. This table would store the user's login information. When they set up a new account, the login name, password and email information are all gonna stored in the database. When the existing user logs into our system, the system will automatically match the login information entered by the user with the stored user login information in the database. If so, users will be allowed to log in.

We will create a new table to store user's heart rate information, but we need to finish setting up Arduino and wait for the android app to be developed. We expect to create this new table set up by November. We will extract the data from the database and export them to a file in CSV format, through which we can use python for data visualization analysis. All these tasks are expected to be complete by mid-November.

We expect to complete the full fledged working of the arduino heart rate device before the first demo i.e October 31, android app integration by the first week of November and database integration by mid November leaving at least two weeks for additional testing and debugging before the final submission.

8. References

- [1] Stavros Asimakopoulos , Grigorios Asimakopoulos , and Frank Spillers, Motivation and User Engagement in Fitness Tracking: Heuristics for Mobile Healthcare Wearables
- [2] Stefan Koelsch and Lutz Jancke, Music and the heart, European Heart Journal (2015) 36, 3043–3048
- [3] Using music to tune the heart URL:
https://www.health.harvard.edu/newsletter_article/using-music-to-tune-the-heart
- [4] Heart rate change from awake stage to sleep stage URL:
<http://www.livestrong.com/article/105256-normal-heart-rate-sleeping/>
- [5] Masao Yaso, Atsuo Nuruki, Seiichi Tsujimura, and Kazutomo Yunokuchi, Detection of REM sleep by heart rate, Proceedings of The First International Workshop on Kansei
- [6] Previous project:
<http://www.ece.rutgers.edu/~marsic/books/SE/projects/HealthMonitor/2014-g12-report3.pdf>

Health Monitoring Application

Group #2

Report #2, Part 1 & 2 & 3

Project URL : <https://github.com/vpranathy/Health-Monitor>

<http://healthmonitoringhomepage-env.5ndteffiz2.us-east-2.elasticbeanstalk.com/>

<http://healthmonitoringsystem.us-east-2.elasticbeanstalk.com/>

Aniket Anilkumar

Yuyang Chen

Divyaprakash Dhurandhar

Zihao Ding

Malay Shah

Pranathy Veldandi

Table of Contents

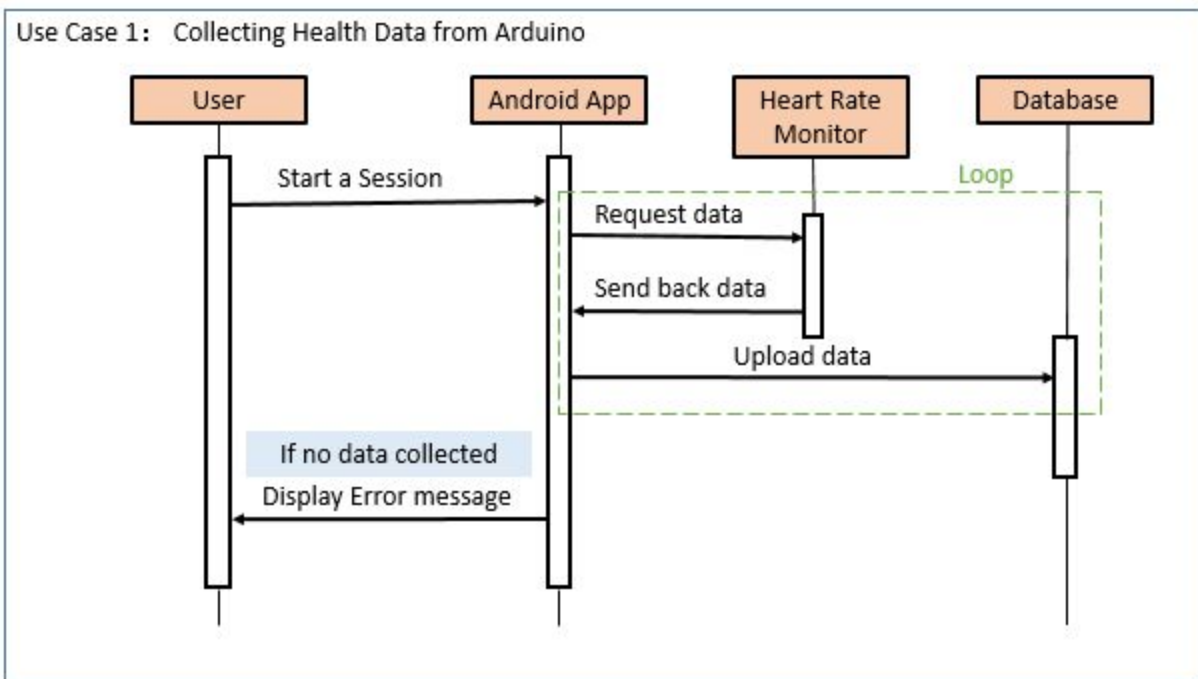
0. INDIVIDUAL CONTRIBUTION BREAKDOWN.....	2
1. Interaction Diagrams.....	2
2. Class Diagram and Interface Specification.....	7
3. System Architecture and System Design.....	21
4. Algorithms and Data Structures.....	26
5. User Interface Design and Implementation.....	27
6. Design of Tests.....	28
7. Project Management and Plan of Work.....	34
8. Cyclomatic Complexity.....	36
9. REFERENCES.....	38

0. Individual Contribution Breakdown

Task	Aniket	Yuyang	Divyaprakash	Zihao	Malay	Pranathy	Total
Interaction Diagrams	16.67%	16.67%	16.67%	16.67%	16.67%	16.67%	100.00 %
Class Diagram and Interface Specification	16.67%	16.67%	16.67%	16.67%	16.67%	16.67%	100.00 %
System Architecture and System Design	16.67%	16.67%	16.67%	16.67%	16.67%	16.67%	100.00 %
Algorithms and Data Structures	16.67%	16.67%	16.67%	16.67%	16.67%	16.67%	100.00 %
User Interface Design and Implementation	16.67%	16.67%	16.67%	16.67%	16.67%	16.67%	100.00 %
Design of Tests	16.67%	16.67%	16.67%	16.67%	16.67%	16.67%	100.00 %

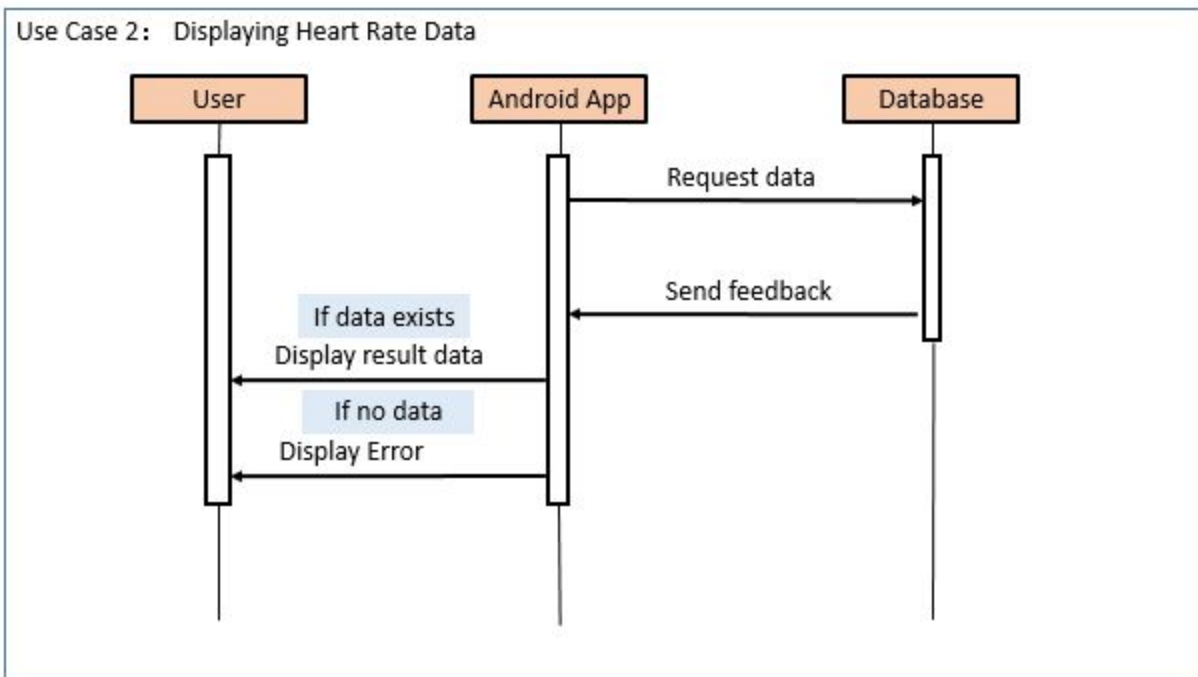
Project Management and Plan of Work	16.67%	16.67%	16.67%	16.67%	16.67%	16.67%	100.00 %
Cyclomatic Complexity	16.67%	16.67%	16.67%	16.67%	16.67%	16.67%	100.00 %
References	16.67%	16.67%	16.67%	16.67%	16.67%	16.67%	100.00 %

1. Interaction Diagrams

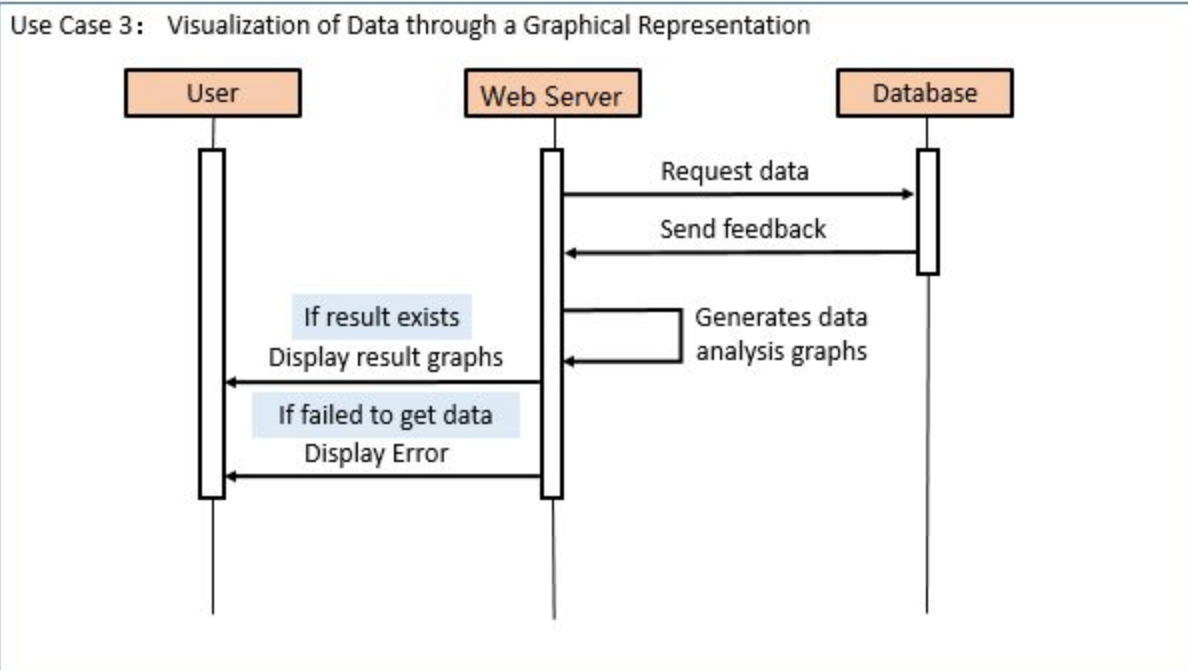


UC1: For our first use case, we wish to record information about the user's heart rate by using Arduino. This includes a timestamp and date of when the heart rate data was recorded, and could also include some of the song metadata such as artist, album, or genre. We also have a Android app to connect the Arduino by Bluetooth, whenever the Arduino sends heart rate data to Android App, the App will automatically upload the data with date, time, and music informations to our database. For our main success scenario, the user interacts with the UI which communicates with the Database Manager. As long as user does not tell the UI to stop recording, the Database Manager will continually ask the Android App for BPM Data and a timestamp for every piece of data received. When the user tells the UI to stop, we break out of our loop, and the Database Manager returns to the UI and displays updated information. Alternate scenarios could occur

when the Arduino is not functioning correctly and reports an error in measurement, and when user move out his/her figure out from the sensor of Arduino, it will also stop recording.

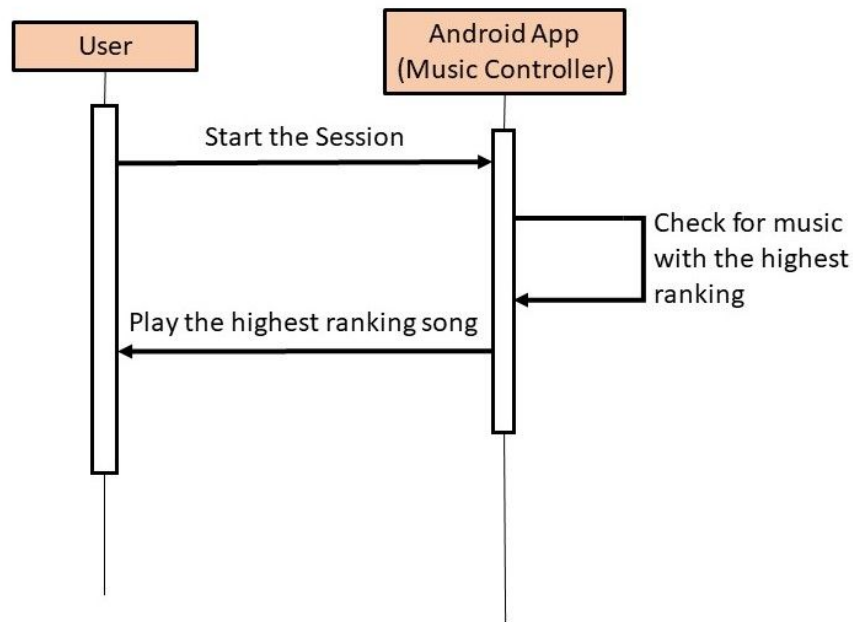


UC2: For our second use case, we are displaying the heart rate data. The database requests the data from the android application. The database sends feedback to the android application. The android application, if the data exists, displays the resulting data to the user. If no data exists, the android application displays an error the user. This is important for the connection between the database, android application, and user. The android UI will display the data to the user. We applied the Expert Doer Principle to guide our use case design. This states that an object who knows should do the task.

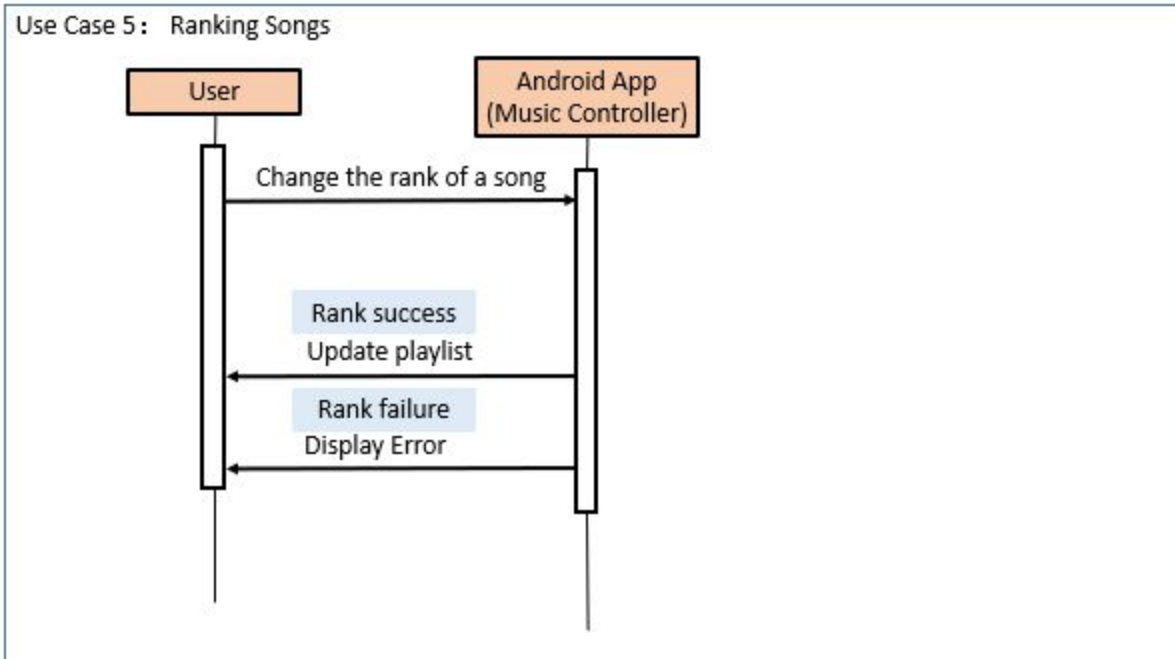


UC3: The User has the hardware which detects the heart-beat at the tip of his finger. This hardware is connected to the mobile application via bluetooth. On receiving the data from the hardware, the application displays it on the activity screen as well as uploads it to the server. The Same server is used by the web-application which can be used by the user for visualization of the efficiency of the activity. The mobile application also displays the graph of the present activity. But in order to get an idea of the previous activities the user has to use the web-application.

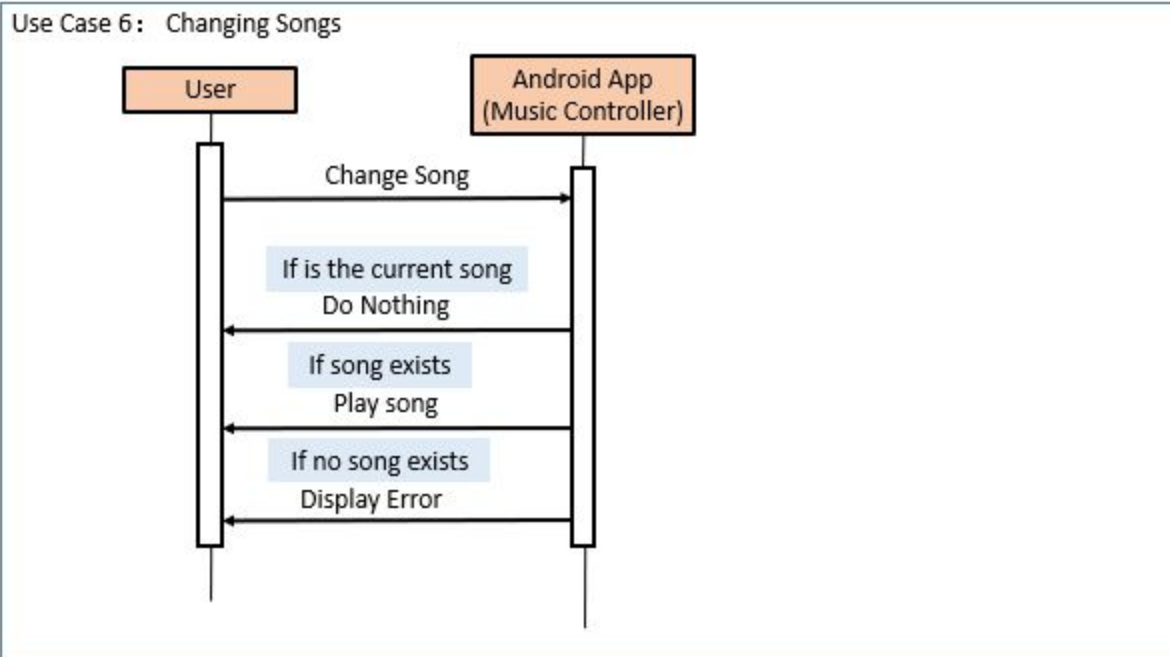
Use Case 4: Initiate music



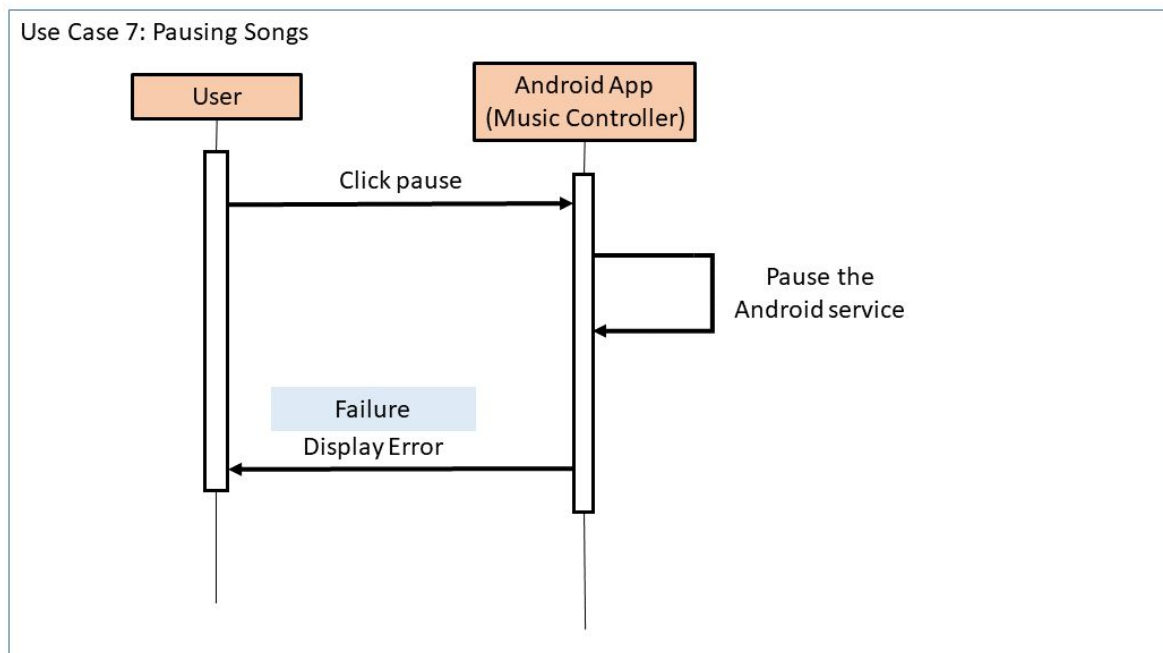
UC4: The user sets the preferred activity. According to the preferred activity the heartbeat range is set that will be ideal for the user. Based on the current heartbeat of the user the song is selected. If the heartbeat is lower than the lower limit of the range the song is selected such that the heartbeat increases and goes beyond the lower range. Such a situation arises when the user will be starting a workout session. For this the purpose the song with the highest rank is played first. Following songs are selected in the descending order of their respective ranks. Whereas in the scenario where the user want to sleep or start studying the heartbeat might be higher than the upper limit since the the range of the heartbeat should be in the lower region. In such an occurrence, songs will be selected such that the mood of the user is lightened which results in lowering the heartbeat.



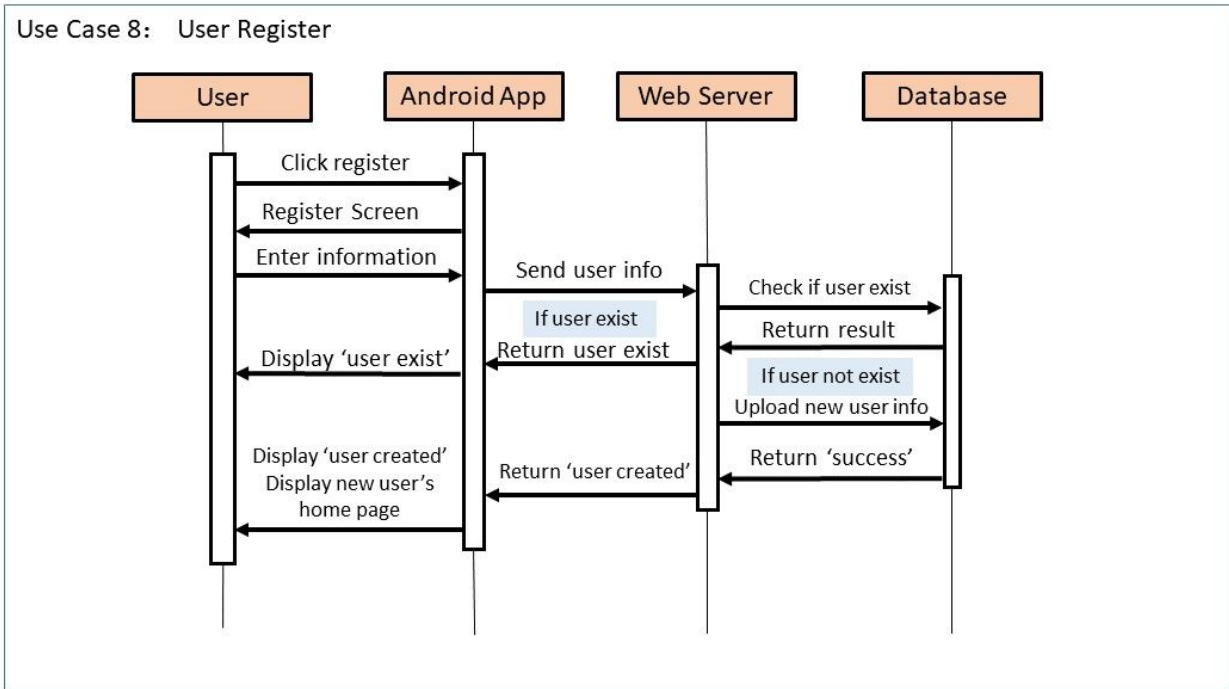
UC5: For this use case, the users are allowed to rank the songs in their library that based on their preferences. Users are able to give their feedback about the songs after every activity. This feedback will be in the form of points out 10. This pointing system will help the application to rank the songs. So, basically when user change the rank of a song in the form of points out of 10, the system will update the playlist in database, then feedback to user with a successful message. If there is an error in the update, then feedback to user with a error message.



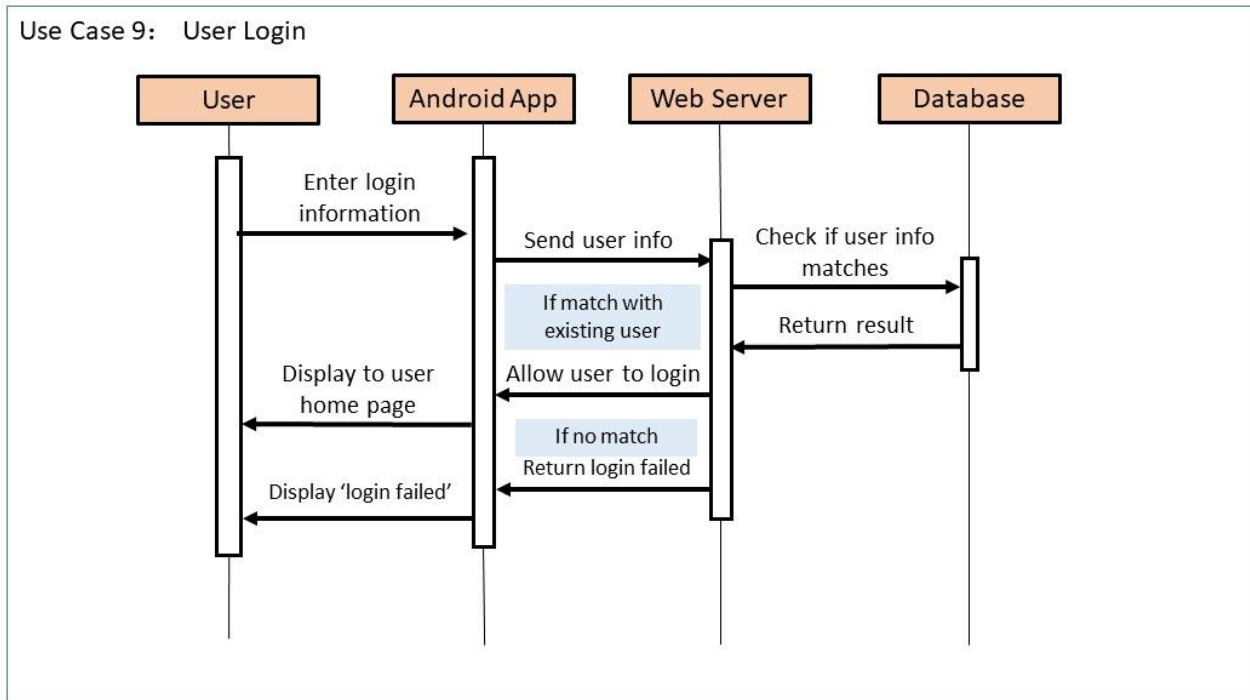
UC6: For the sixth use case, the user has the option to change the song to play in accordance with their preferences. When the user changes a song, the app checks if the selected song is the same as the existing one. In this case, nothing is done. If the song is different, then the app checks for the selected song. If the song exists, the selected song is loaded and can be played by clicking on the play option in the music player widget. If the song doesn't exist, an error message is shown.



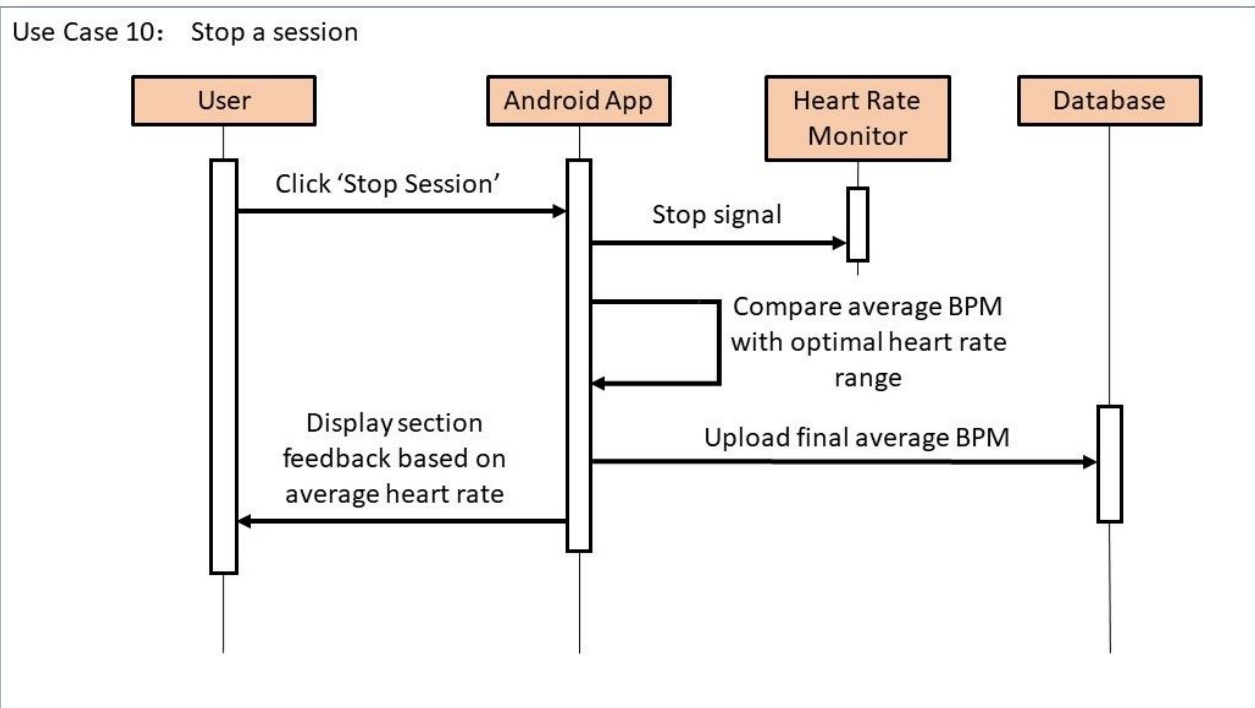
UC7: For the seventh use case, the user has the option to pause the song if there are any interruptions to their routine. When the user pauses a song, the app first checks if a song is even being played. If there is a song playing, the song is stopped and playback is resumed from that point again when the user presses play if the same session is being continued. If there is no song playing, an error message is shown.



UC8: For the eighth use case, the user can create an account by pressing “Register Here” button in the login screen, after entering the information and click “Submit”, the Android APP will send the user information to the web server. The webserver will check the user exist or not. If exist it will return “User already exist.” If not exist, it will insert the data to the database, and return a “Success” back to user.



UC9: For the ninth use case, the user can login their existing account on the homepage of the APP. Then the APP will send the login information to the web server. The web server compare the login information with the user information stored in the database. If the credentials are valid then the user is redirected to the homepage of their account. If the credentials are incorrect the users is asked to retype the credentials.



UC10: Stop a Session

This use case is initiated when the user clicks on the stop button in the mobile application. During the session, a variable keeps on adding the value of the current heartbeat to get the cumulative value of the heartbeat and another variable tracks the current heartbeat value. When the stop button is clicked the cumulative value is used to calculate the average heart rate value. This average value is then compared to the range of the optimal heart rate. If the average value is within the range then the activity is completed with optimal heart rate and is considered as a successful session. A popup window open informing the summary of the activity. Parallely the arduino is asked to stop the transmission of the live heart rate data. The average heart rate is uploaded to the database in the remote server.

2. Class Diagram and Interface Specification

a. Class Diagrams

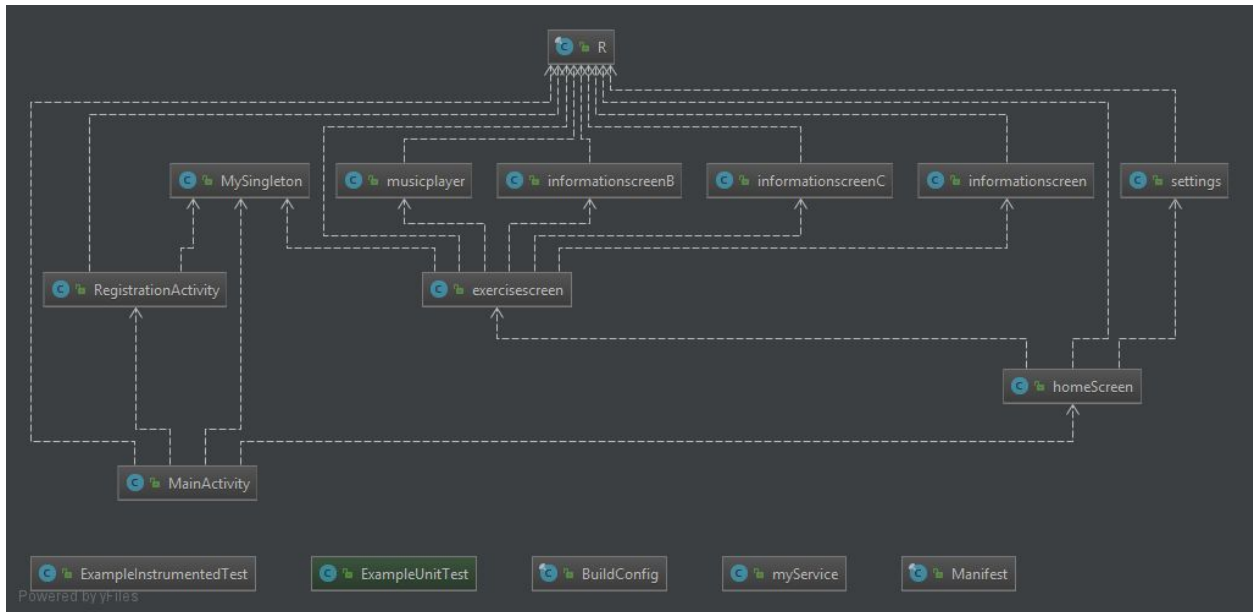


Figure: Class Diagram in general for the main class of the Android app.

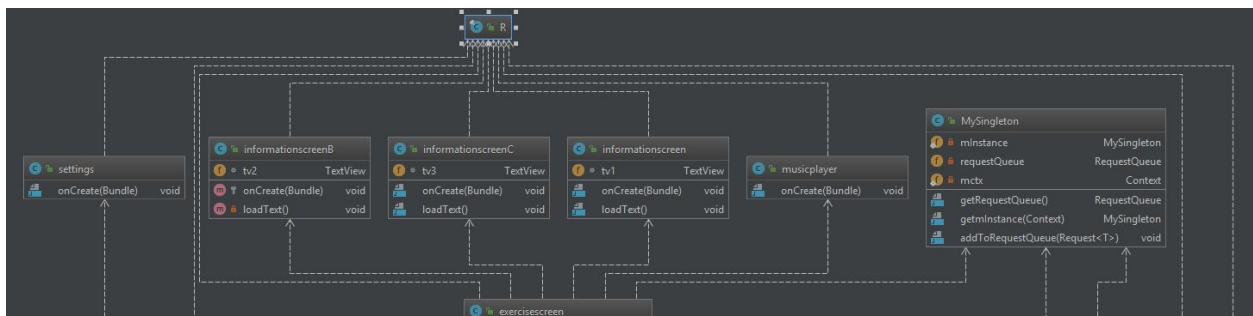


Figure: The first part of Class Diagram for the main Android app in detail.

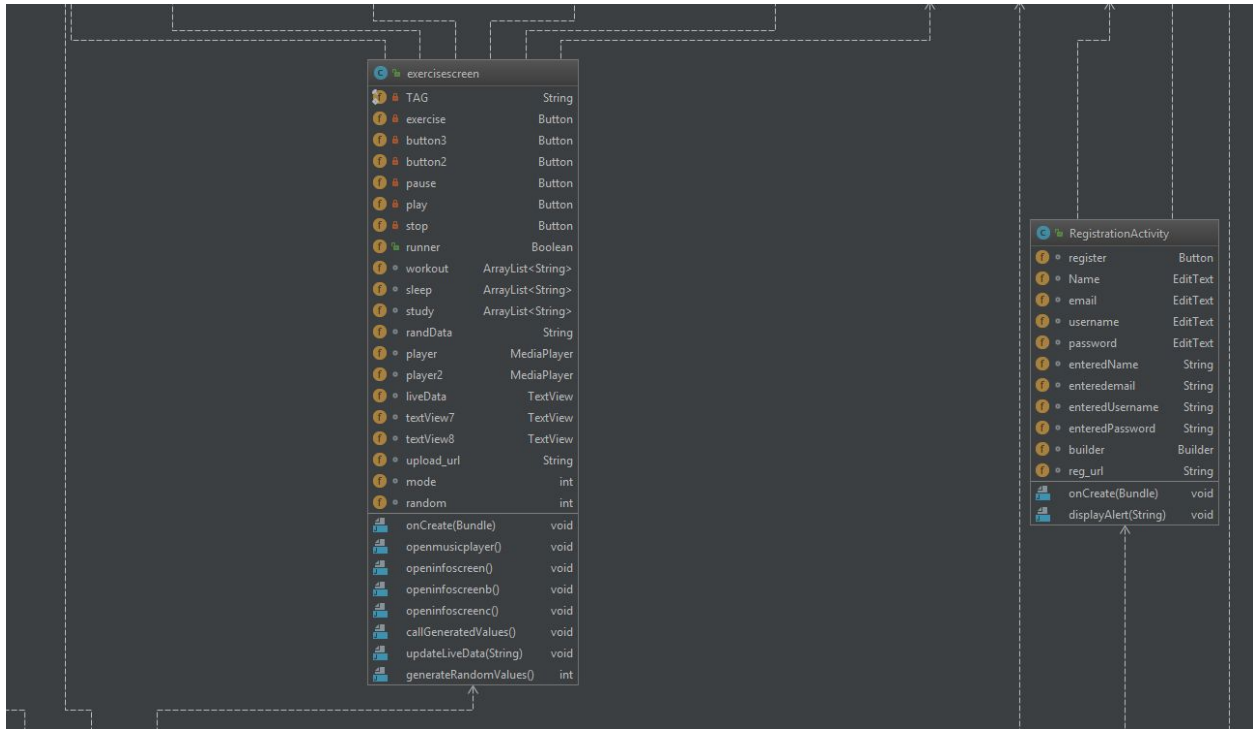


Figure: The second part of Class Diagram for the main Android app in detail.

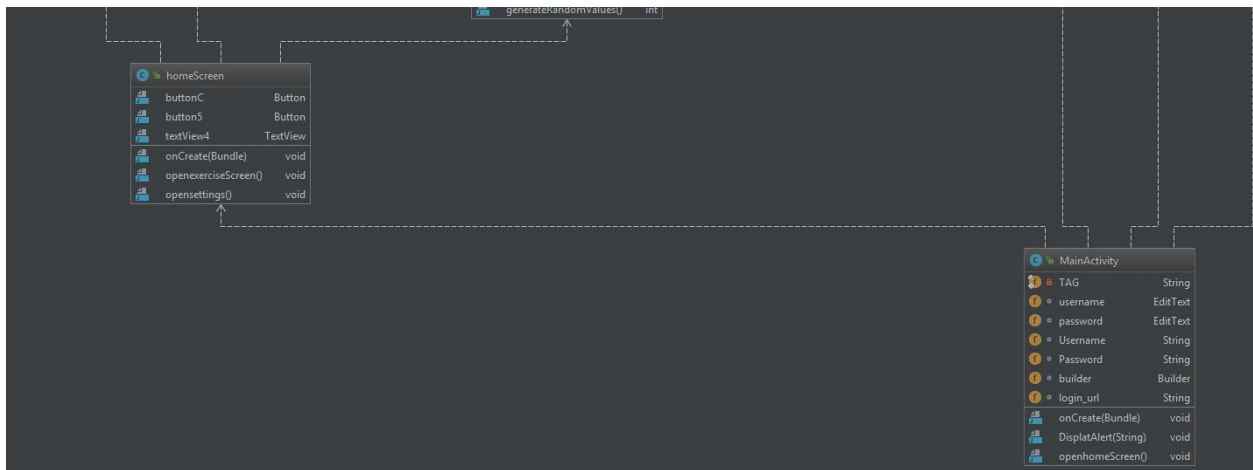


Figure: The third part of Class Diagram for the main Android app in detail.

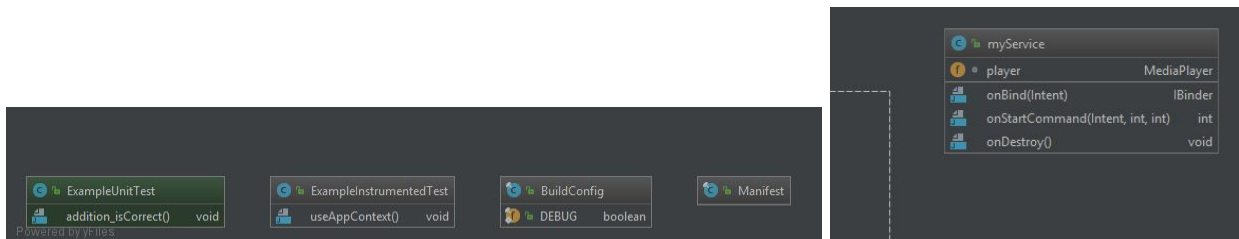


Figure: The final part of Class Diagram for the main Android app in detail.

The android application for bluetooth communication till this stage was developed separately. It was tested for its communication with the arduino. In the later stages of the project, this application will be integrated within the other application which communicates with the databases on the remote server.

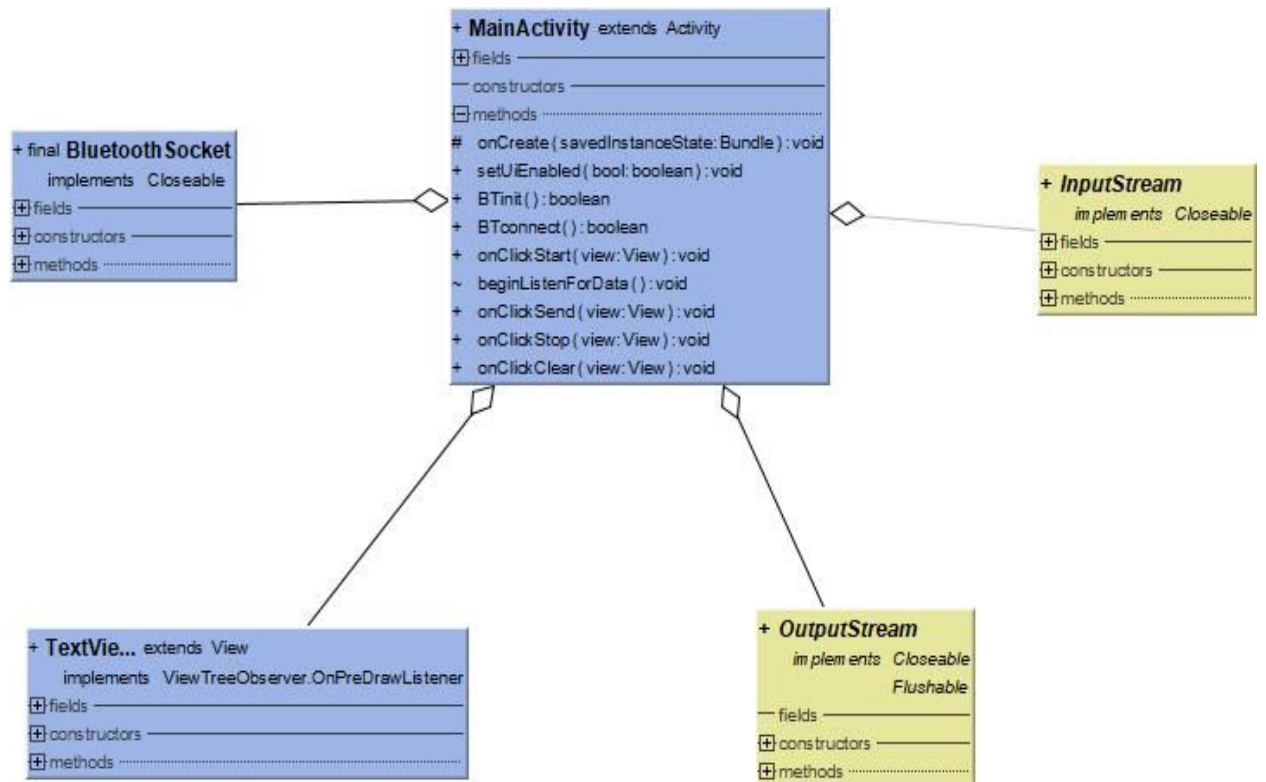


Figure: Class Diagram for Bluetooth Functionality

b. Data Types and Operation Signatures

Settings
<p>Functions:</p> <ul style="list-style-type: none">• <i>onCreate(Bundle) : void.</i> Every activity has a <i>onCreate()</i> method which is run when the activity is created. Whatever code we want to run at the start must be placed inside it. Reading from the device is asynchronous, meaning it will keep running in the background. This is done so that the data is received as soon as possible .

Information Screen
<p>Functions:</p> <ul style="list-style-type: none">• <i>onCreate(Bundle) : void.</i> Every activity has a <i>onCreate()</i> method which is run when the activity is created. Whatever code we want to run at the start must be placed inside it. Reading from the device is asynchronous, meaning it will keep running in the background. This is done so that the data is received as soon as possible .• <i>loadText() : void.</i> This function load Text from the text box.

Music player
<p>Functions:</p> <ul style="list-style-type: none">• <i>onCreate(Bundle) : void.</i> Every activity has a <i>onCreate()</i> method which is run when the activity is created. Whatever code we want to run at the start must be placed inside it. Reading from the device is asynchronous, meaning it will keep running in the background. This is done so that the data is received as soon as possible .

MySingleton
<p>Functions:</p> <ul style="list-style-type: none"> • <i>getRequestQueue() : RequestQueue.</i> This function is to post a request with query to get information from the DataBase. • <i>getInstance(Context): MySingleton.</i> This function is use to get the instance. • <i>addToRequestQueue(Request<T>): void.</i> This function is called when the a new request is to be made. On being called, the new request is added to the request queue.

Exercisescreen
<p>Functions:</p> <ul style="list-style-type: none"> • <i>onCreate(Bundle) : void.</i> Every activity has a onCreate() method which is run when the activity is created. Whatever code we want to run at the start must be placed inside it. Reading from the device is asynchronous, meaning it will keep running in the background. This is done so that the data is received as soon as possible . • <i>openmusicplayer(): void.</i> This function call up the internal music player. The user will able to use the basic music player functions in the newly called activity. • <i>openinfoscreen(): void.</i> This function is called to display an activity which has all the health related information in a tabular form which can act as a source of information for the user. The information will depend on the activity the user is indulged in. • <i>callGeneratedValues(): void.</i> This function is called to get the random values back which are generated in the generateRandomValues function. The returned value is displayed in the textView in the exercisescreen Activity. • <i>updateLiveData(String): void.</i> This function is called to upload the current heartbeat data to the server. • <i>generateRandomValues(): int.</i> This function generates a Random integer.

HomeScreen

Functions:

- *onCreate(Bundle) : void.* Every activity has a onCreate() method which is run when the activity is created. Whatever code we want to run at the start must be placed inside it. Reading from the device is asynchronous, meaning it will keep running in the background. This is done so that the data is received as soon as possible .
- *openexerciseScreen(): void.* This function is called on tapping the Start button. This function basically uses the intent function to call the exercise screen where the user can start collecting the heartrate data and play the music.
- *opensettings(): void.* This function is called on tapping the setting button. Using Intent the new activity is opened where the user can add the personal details like age, sex, weight etc.

MainActivity

Functions:

- *onCreate(Bundle) : void.* Every activity has a onCreate() method which is run when the activity is created. Whatever code we want to run at the start must be placed inside it. Reading from the device is asynchronous, meaning it will keep running in the background. This is done so that the data is received as soon as possible .
- *DisplayAlert(String): void.* This function is called to display a pop up Alert about the information that be put in. User can click “OK” button to dismiss the alert.
- *openhomeScreen(): void.* This is function is called when the user logs in successfully and by using Intents the function opens a new Activity for the user to either start a new routine or change the settings.

Main Activity for Bluetooth

Functions:

- *onCreate() : void*. Every activity has a *onCreate()* method which is run when the activity is created. Whatever code we want to run at the start must be placed inside it. Reading from the device is asynchronous, meaning it will keep running in the background. This is done so that the data is received as soon as possible .
- *BTinit() : boolean* . This function checks whether the bluetooth adapters for the android phone are set or not and therefore initialises them.
- *BTconnect() : boolean*. If adapters are available, this function is used to initiate the connection to the arduino bluetooth adapter.
- *beginListenForData() : void*. This function constantly checks the input stream for incoming data.
- *OnClickStart() : void, OnClickStop() : void, OnClickSend() : void, OnClickClear() : void* These functions are used for debugging purposes to check whether the input data is being displayed or not and whether we can remotely trigger the arduino device or not.

Bluetooth Socket

Functions:

- *connect() : void*. This function opens the bluetooth socket in the android phone for bluetooth connection.
- *isConnected() : boolean*. This function checks whether the status of bluetooth connection is true or false.
- *getInputStream() : InputStream*. After getting the BluetoothDevice, a socket has to be created to handle the incoming connection. Here a RFCOMM socket is used. RFCOMM--also known as Serial Port Profile--is essentially a Bluetooth protocol to emulate an RS232 cable.
- *getOutputStream() : OutputStream*. After getting the BluetoothDevice, a socket has to be created to handle the outgoing connection. Here a RFCOMM socket is used. RFCOMM--also known as Serial Port Profile--is essentially a Bluetooth protocol to emulate an RS232 cable.
- *close() : void*. This function closes the bluetooth socket in the Android phone to terminate the connection.

InputStream

Functions:

- `read() : int`. This function reads the input stream and data
- `read(bytes:bytes[]) : int`. This function reads the input stream and data in bytes
- `available() : int`. This function makes the data available to write on android screen
- `close() : void`. This function closed the input read stream.

OutputStream

Functions:

- `write() : int`. This function writes the input stream to android textview
- `write(bytes:bytes[]) : int`. This function writes the input stream to android textview.
- `flush() : void`. This function refreshes the screen for new data to be displayed or else the data stays on the screen.
- `close() : void`. This function closes the output stream to android textview.

TextView
<p>This method contains various pre-built functions to process data as the data received will be in the form of raw bytes. We will have to re-encode it into a readable format like UTF-8. Then it is appended to the TextView using a custom method named tvAppend(). This is done because any change to the UI can only happen on the UI thread. Since this Callback will be running as a background thread, it can't affect the UI directly.</p>

c. Traceability Matrix

	Class								
Domain Concepts	Main Activity	Home Screen	Settings	Information Screen	My Singleton	Registration Screen	Exercise screen	Web Server	Bluetooth Connect
HRM Manager							X	X	X
Log Retriever	X	X			X			X	
Track Logger	X	X	X		X	X	X	X	X
Music Playerbacker							X		
Track Queuer							X		
General UI	X	X	X	X	X	X	X	X	
Playback View							X		
Heartbeat View	X	X					X		
Workout View	X	X		X	X	X	X	X	
History View	X	X						X	
Workout Store	X	X						X	

Metadata Store	X	X	X		X		X		X
Music Store							X		
Rest Setter			X				X		
Peak Calculator			X				X		
User Alerter							X	X	

Based on the Domain Concepts, We designed multiple classes for our Mobile Application as that is the major part of this project and one Web Server to deal with the Web Application and the database.

Main Activity - This is the Main class that ties the web app and the mobile app and logs in the user based on their registered account and displays the main home screen

Home Screen - Interacts with the database and web server to get the data of previous sessions and

Settings - Will be used to input user information and is connected to the database on the webserver and calculates optimal heart rate

My Singleton - Is the main interface for interacting with the web server. Allows HTTP requests to be made by the application in a synchronized manner.

Registration Screen - Is connected to the database to store user information and registers a user so the database can be accessed later while logging in

Exercise Screen - Has the major functionality of the mobile app and is used to get live heart rate data via bluetooth and is used to get the peak calculated by settings and stored in the database. It also is used to alert the user based on whether the target of that session has been reached. It can also start/stop a session and register the results in the database in webserver.

Web server - Stores all information of the user in databases and displays data visually through graphs

Bluetooth Connect - Is used to connect with the Heart Rate Monitor via bluetooth and can get real time heart rate data

3. System Architecture and System Design

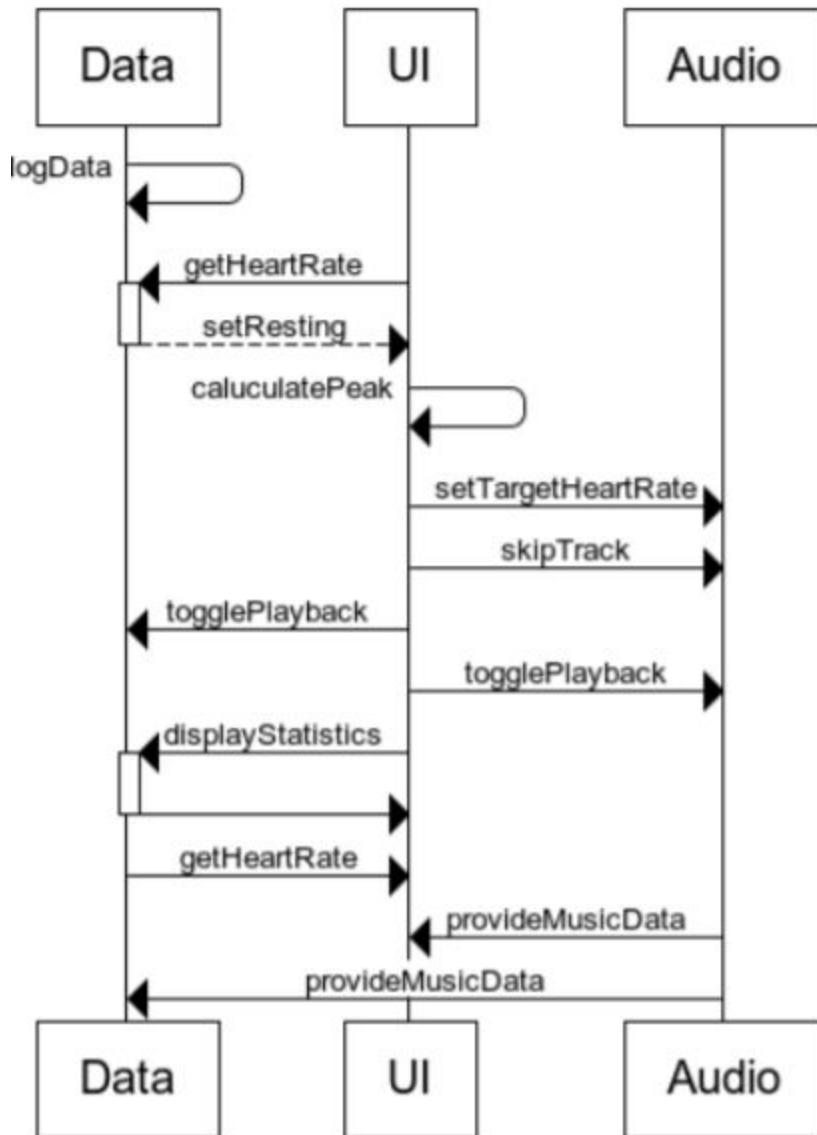
a. Architectural Styles

Our system has a 3-tier architecture style which has 3 different layers. The first layer starts with the android app user interface and web based interface which is the presentation layer. This is the mobile interface and web interface allows the user to visualize the information presented to them. They have inputting commands and present outputs to the user. All of this information is readable and uses logical analysis. The next layer is the data storage layer. The database stores the information given by the user and is put together into a table. This table is visualized and analyzed through graphical representation through the web based application. The final tier is the

application tier which is built by android and web based development. There is plenty of code within these tiers which allows the applications to interact with the database and represent information successfully to the user and system. The application tier is very system and user friendly and allows for ease of access between the 3 different layers. All of our tiers require powerful connections and databases because they have to handle a lot of data relatively quickly.

b. Identifying Subsystems

The software is designed through three primary subsystems. The main subsystem is the UI Subsystem. This subsystem is represented through the mobile and web application. The user has easy access through these two different applications. The second subsystem is the data subsystem and the third subsystem is the audio subsystem. The data subsystem acquires and analyzes data from the mobile and web application. The data is retrieved, stored, and analyzed. Often the subsystem requires user input and system analysis. The audio subsystem is the basis of how the different elements connect with each other in terms of music. This includes the the data, user interface, web application, and mobile application. They audio subsystem allows for the accurate interaction between different elements because the music is a big factor in their analysis.



c. Mapping Subsystems to Hardware

The Audio subsystem contains two primary components in this application, which are server component and client interface. Clients and servers exchange messages in a request–response messaging pattern. The client sends a request, and the server returns a response. This exchange of messages is an example of inter-process communication. To communicate, the computers must have a common language, and they must follow rules so that both the client and the server know what to expect. The language and rules of communication are defined in a communications protocol. All client-server protocols operate in the application layer. The application layer protocol defines the basic patterns of the dialogue. To formalize the data exchange even further, the server may implement an application programming interface (API). The API is an abstraction

layer for accessing a service. By restricting communication to a specific content format, it facilitates parsing. By abstracting access, it facilitates cross-platform data exchange.

d. Persistent Data Storage

We are using the AWS RDS MySQL database for this entirely project. MySQL is the world's most popular open source relational database and Amazon RDS makes it easy to set up, operate, and scale MySQL deployments in the cloud. With Amazon RDS, you can deploy scalable MySQL servers in minutes with cost-efficient and resizable hardware capacity. Amazon RDS for MySQL frees you up to focus on application development by managing time-consuming database administration tasks including backups, software patching, monitoring, scaling and replication. The wide variety of fully-developed features allows us to focus more on the actual organization and management of the data in relation to the other modules. All that is needed is a simple call to the database to retrieve the raw data, and the custom designed objects illustrated in the Class diagram then do their own processing on the data. MySQL allows us to store all the data specific to application on the device and system itself, which is advantageous for a mobile application and web application such as ours. The goal is for the user to be able to record and view his workout data, sleeping data, studying data without having to use any external devices other than his phone and browser, and internal data storage via the MySQL database allows our application this benefit.

The database will be accessible only to the Data Manager and the Data Assembler. In regards to the Data Manager, the only interactions with the database will be to store the initial state of the system, store the current music track, and store the current heart rate. It will not retrieve anything from the database, because that is the purpose of the Data Assembler. The Data Assembler is the other object that will interact with the database. It will issue requests for the various data that the UI would like to graph, which include the heart rate, the current date and times, and the songs. Thus, the Data Manager and Data Assembler are the only objects that have direct access to the database. The reference implementation of Audio subsystem utilizes RDS database for storing information about the available music library.

e. Network Protocol

The Audio subsystem is implemented entirely over HTTP. HTTP (Hypertext Transfer Protocol) is the set of rules for transferring files (text, graphic images, sound, video, and other multimedia files) on the World Wide Web. As soon as a Web user opens their Web browser, the user is indirectly making use of HTTP. HTTP is an application protocol that runs on top of the TCP/IP suite of protocols (the foundation protocols for the Internet).

f. Global Control Flow

i. Execution Orderliness

The execution order is a mix of procedure-driven and event-driven. From a broad view, the use of the program follows the same steps: the user starts the

system and selects a mode, the music plays, then the user stops the music or music is automatically stopped when target heart rate is reached based on the mode. However, the system provides a variety of interface options to activate events during the execution: a user may pause or skip playback, and view their statistics, at any time. Internally, data transfer between the app and web server is procedural. Clients make requests and the content of those requests determines the information the server returns.

ii. Time dependency

The system is real-time, with a timer firing once a second. This timer triggers the fetching of the heart rate from the monitor, and triggers the logging of this data. With respect to the other subsystems the Audio subsystem is event-driven. It can be told to pause or play and it will provide an audio stream except for the sleep mode which is real time and turns off the audio subsystem based on whenever the current heart rate reaches its target rest rate.

iii. Concurrency

The Android standard concurrency model is that the main thread handles UI, so lengthy tasks must be performed on a background thread else the UI becomes unresponsive. Synchronization is unnecessary as there are no shared resources. The Audio subsystem is responsible storing meta information about the audio files, and serving audio streams. These operations are all performed asynchronously from one another. The beauty of this design is that concerns about such things are abstracted away from our implementation

g. Hardware Requirements

The system requires:

- Touch screen display with minimum resolution of 640 x 480 pixels
- Storage space for music library, minimum size of 100 MB
- Bluetooth for communication with a heart rate monitor 96
- Network connection for communicating with music selection service
- Audio playback capabilities

All of these requirements are met by most Android phones on the market

4. Algorithms and Data Structures

1. ArrayList :

a) A class named “user activity” is created which has the following attributes.

- UserName;
- User Activity Number
- Activity type
- Type and the name if the music that was played
- A Boolean which indicates whether the target heartrate was achieved

The arrayList of the type “User activity” is the data structure used to store the previous 3 activities and its related information. This information is retrieved from the database and added to the arraylists. This filled arrayList is used by the arrayAdapter to populate the listView on the Homescreen which display the information about the previous 3 activities.

b) Arraylists are also implemented to store the names of the music files that will be used in the project. This implementation comes in handy when the user click the next track button to change the music. On clicking the “NextTrack”, the next index of the arrayList is used is used to retrieve the name of the next track that must be played. 6 arrayLists of this type will be used.

2 Singleton Pattern : The Singleton Pattern is a software design pattern that guarantees a class has one instance only and a global point of access to it is provided by that class. Anytime multiple classes or clients request for that class, they get the same instance of the class. This Singleton class may be responsible for instantiating itself, or you can delegate the object creation to a factory class.

- a. Let's use the example of a cell phone and its owner. A phone is typically owned by a single person, while a person can own many phones. Anytime one of these phones rings, the same owner picks it up
- b. It is used to implement the volley requests that is used by the application. The advantage of using this class is that it avoids generation of race condition between the requests. Race condition is harmful when working with http requests. Since the singleton class is instantiated just once, it helps our system to execute without any defects.

3. HashMap<String, String>: This data structure is used to send data from the mobile application to the php files on the server. The key and value are of the type “String”. The name of the key should be exactly the same as the name mentioned in the php file.

5. User Interface Design and Implementation

Since we spent the time to make high quality mockups early on in this project, the UI was already well thought out and designed with standard Android UI elements such that it does not have to change much in implementation.

One significant difference with regards to our initial design was the removal of a few features. We removed the sidebar menu from our original mockups and the user directly goes to each screen - Connect, Information and Settings, by clicking on buttons on the main screen.

This simultaneously decreases the user effort in some cases and makes no difference in others:

It makes no difference to the user effort when user has to go from one of these screens to another, for example from information to settings, as originally the user would have to click on the menu icon to open the menu and then click on whichever screen option they wish to go to. Now, the user has to go back and then click the button for whichever screen they want to go to so the total clicks remains two.

It decreases user effort by one click when the user has to go to any other screen from the main screen because originally, the user would have to click on the menu icon to open the menu and then click on whichever screen option they wish to go to. Now, the user has to just click on the button on the home screen to go that particular screen.

One other UI feature that has been implemented is an alert that lets the user know if their target heartbeat rate has not been reached in that current session. The user has the option to stop their current session at any time but the alert lets them know if their session was productive so the user can make their decision as to whether they would like to start another session till they reach their target or stop there.++

6. Design of Tests

a. Test Cases used for Unit Testing

i. Test Cases used for Custom built heart rate monitor:

Test case id	Heart Rate collection
Unit to test	Arduino Heart rate monitor
Assumption	The device can display the correct heart rate data including instant result and the average beats per minute.

Steps to be executed	Press the start button on the Arduino unit. Put finger on the sensor, check the result display on the LCD screen, then compare the result with the BPM reading from Fingertip Pulse Oximeter bought from market.
Expected result	The BPM reading from our heart rate monitor should be close to the BPM reading from the Fingertip Pulse Oximeter bought from market.
Pass/Fail	The result from our custom built heart rate monitor is +/-2% of the result from the Fingertip Pulse Oximeter bought from market.

Test case id	Bluetooth Data Transmission
Unit to test	Arduino Heart rate monitor
Assumption	The device can receive the signal from the Android Phone in order to turn on/off a LED light and send the current result (1/0) back to the Phone.
Steps to be executed	Open the Android APP, click the ON/OFF button, check the Status of the LED, check the information section in the APP.
Expected result	While click the ON button in the APP, the LED on the Arduino Board should turn on and "1" will appear on the information section in APP. While click OFF button in the APP, the LED will turn off and "0" will appear in the information section in APP.
Pass/Fail	The LED turned on and off successfully and the result shown in the APP info section correctly / The LED did not perform as predicted.

ii. Test Cases used for Android APP:

Test Case	Testing the Php files
Unit to test	Integration Manager
Assumptions	A browser with working internet connection

Steps to be executed	Browse to the following URL http://healthmonitoringsystem.us-east-2.elasticbeanstalk.com/mobile_connect.php
Expected Result	A blank echo from the php file
Pass/Fail	A blank echo from the php file / A connection error from the browser.

A similar test can be conducted for the rest of the php files by replacing the mobile_connect.php with mobile_login.php, mobile_register.php and mobile_heartrate.php. An echo from the php files is expected for the proper functioning. An error from the browser indicates a problem with the connection with the server.

Test Case	Account Registration
Unit to Test	Account Manager and Database Manager
Assumptions	The program has displayed the registration screen and is waiting for the user's input.
Steps to be executed	Input the preferred account number and password. Input the basic personal information. Click "finish" button to complete registration
Expected Result	A new User is registered in the database
Pass/Fail	A pop up window which says user Registered / A pop window which says Error while registering.

Test Case	Account manager and Database Manager
Unit to test	User Login
Assumptions	The program has displayed the login Screen and is waiting for the user's input
Steps to be executed	Input the preferred account number and password. Input the basic personal information. Click “finish” button to complete registration
Expected Result	User should be redirected to the homePage
Pass / Fail	Homepage is displayed / Popup activity which says invalid credentials

iii. Test Cases used for Database and Web Page:

Test Case ID	Database testing
Unit to test	Database Manager
Assumptions	The application stores the user's information in the application database and displays them correctly to the user.
Steps to be executed	This type of testing involves validating the schema, database tables, columns, keys and indexes, stored procedures, triggers, database server validations, validating data duplication.
Expected Result	No information is lost in the process. No partially performed or aborted operation information is saved by the application. No unauthorized individual is allowed

	to access the users information.
Pass/Fail	No data lost in the process/Data lost in the process.

Test Case ID	Account Registration
Unit to test	Account Manager, database Manager
Assumptions	The program has displayed the registration screen and is waiting for the user's input.
Steps to be executed	Input the preferred account username and password. Input the email address. Click "Register" button to complete the registration.
Expected Result	A new user account has been added to the database, and it will be allowed to login to our System by using this account information.
Pass/Fail	A new user account shows up in the database/No account has been added to the database.

Test Case ID	Log in/ Log out
Unit to test	Account Manager, database Manager
Assumptions	The program has displayed the registration screen and is waiting for the user's input.
Steps to be executed	Input a valid user account and the corresponding password. Click "login" button. If step 2 successes, Click "logout" button.
Expected Result	The user successfully logs in and

	logout.
Pass/Fail	The user log in the system, and then logout / the user fails to log in.

Test Case ID	Show Heart Rate table
Unit to test	GUI, database Manager
Assumptions	The user has logged in the system and clicked the “Table” button on the table page.
Steps to be executed	When user click the button to table page, the page will be automatically reflash the system to grab data from our database, and then show all the heart rate data based on different activities as table format.
Expected Result	The heart rate data is displayed in the multiple tables based on their activities.
Pass/Fail	The table is successfully displayed / the table does not show up

Test Case ID	Show Heart Rate graph
Unit to test	GUI, database Manager
Assumptions	The user has logged in the system and clicked the “typography” button on the typography page.
Steps to be executed	When user click the button to typography page, the page will be automatically reflash the system to grab data from our database, and then display the heart rate data in the x-y coordinates as a function of time.

Expected Result	The heart rate data is displayed in the multiple graphs based on their activities.
Pass/Fail	The graph is successfully displayed / the graph does not show up

b. Test Coverage : The above unit testing is a mandatory step in the development and deployment of the system. It tests the basic units of the system and the interaction between them. Any error in the tests can cause complete system breakdown. The system was tested with correct as well as incorrect data to check its reliability. These testes cover all possible sub systems that contribute to the smooth functioning of the system.

c. Integration Testing Strategy

Mobile Application:

- 1) The integration for this part is communication between the application and the database on the remote server
- 2) This can be tested by trying to register the new user in the application .
- 3) After you have filled the details and clicked the register button. You will be routed to the login page.
- 4) Now Try logging in
- 5) If the above steps are successfully followed, then the system has been integrated successfully

Web Application:

The integration flow test we carried out is described below.

- 1) Login with unknown user account -> Fail
- 2) Login with user account with wrong password -> Fail
- 3) Login with user account with correct password -> Success
- 4) Add user -> Success
- 5) Logout -> must show the index page, which is the dashboard page
- 6) Login with new user -> success, must show the loginsystem.php and wait 3 seconds then redirect to index page, which is the dashboard page.
- 7) Click button “Table” -> Success, User’s information and heart beat data must be shown as
multiple tables.
- 8) Click button “User” -> Success, User’s information must be shown, and able to allow

user to modify their personal information.

- 9) Click button “Typography” -> Success, Heart Rate Graph must be shown as graph function, and they are based on different activities.
- 10) Click button “Maps” -> Success, Google maps must be shown, and with multiple markers shown.
- 11) Logout -> must show the index page, which is the dashboard page.

7. Project Management and Plan of Work

a. Merging the Contributions from Individual Team Members

We compiled the final copy of the report from everyone's work. We ensured consistency, uniform formatting, and appearance. Constantly throughout the project one of our team members would be in charge of formatting and appearance. Some issues that occurred would be working the last couple hours and working on the formatting at the same time. Our group members would constantly be updating the formatting so our project would have a standard. To add on, the merging of the contributions was done through Google Docs. Google Docs let us work on the report at the same time and consistently add to the report when necessary.

b. Project Coordination and Progress Report

The use cases that have been implemented include:

UC #2 Displaying Heart Rate Data

UC #3 Visualization of Data through a Graphical Representation

UC #4 Initiating Music

UC #6 Changing Song

UC #7 Pausing Song

UC #8 Register

UC #9 Login

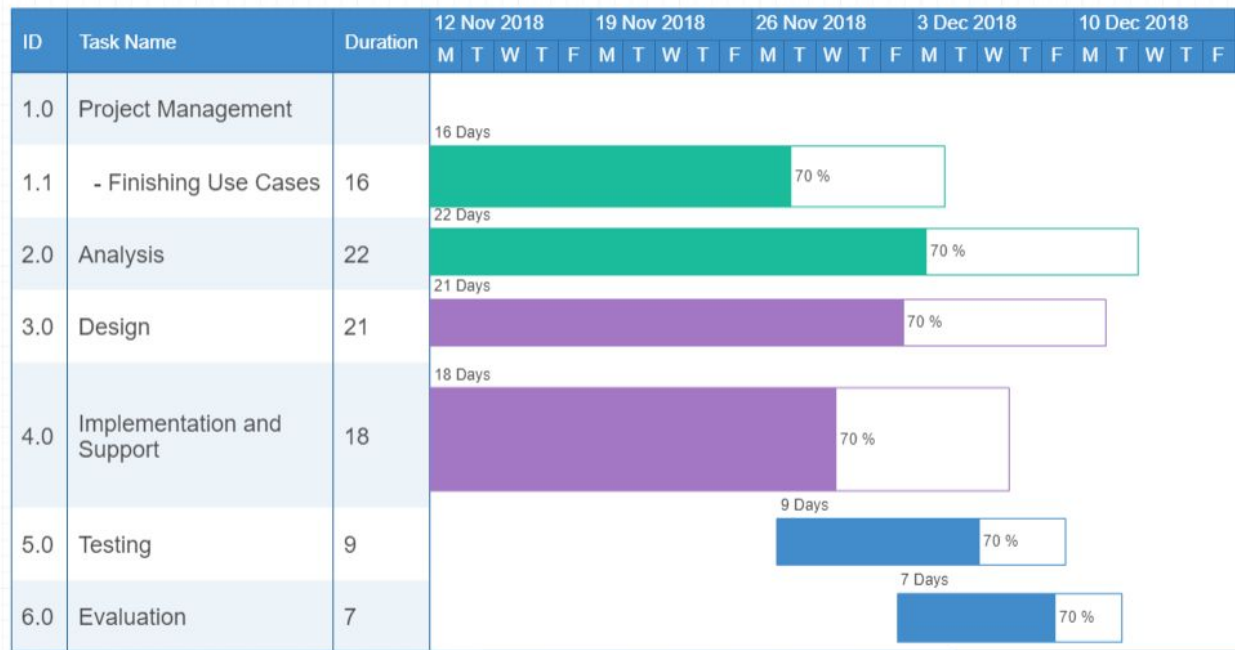
UC #10 Stop Session

These uses cases are currently functional and we are currently tackling **UC #1 Collecting Health Data from Arduino & UC #5 Ranking Songs**. We are working together as group to implement these. We are constantly communicating with the group member in charge of the arduino and waiting to finalize things as he finishes the communication and sensor. The android app development team is currently finishing ranking the songs with the users preference. This use case requires data so it will take time to tackle because we have to acquire large amounts of data. Data acquisition is a time intensive process. Overall, project coordination is at high level currently. The diagrams are clear and intelligible. There is a high consistency and traceability between the requirements, use cases, use interface, and the domain model. This is perceived through Report #1 and Report #2. Connections are constantly drawn and referenced. Our project has a plethora of resources and references to back all of our findings. We made a clear effort to

make sure things are readable and provide adequate information about any platform we are using.

c. Plan of Work

The project milestones and dates by which we plan to accomplish them are listed below:



d. Breakdown of Responsibilities

Aniket, Pranathy and Malay will implement the music selection process due to the heart rate monitoring. This will be influenced by the current heart rate and activity of the user. For the next part, we intend to gather the data from the arduino. Yuyang will also work on the web development to synchronize all information between web client and mobile client, making it easier for users to use the system. The data will be stored in the database with timeline and date, he will use javascript to do the data visualization based on the timeline, then user will be allowed to select a time range to view his/her heart rate on the website. Divyaprakash and Zihao will be responsible for designing and programming the Heart Rate Monitor using the Arduino board as well as configuring it to connect via bluetooth to the android smartphone. The integration will be done as a team and will be coordinated by Aniket and Zihao. They will make sure all components of the project are connected together. The integration testing will be done by the whole team. We will work together and break tasks up accordingly to who developed that task.

8. Cyclomatic Complexity

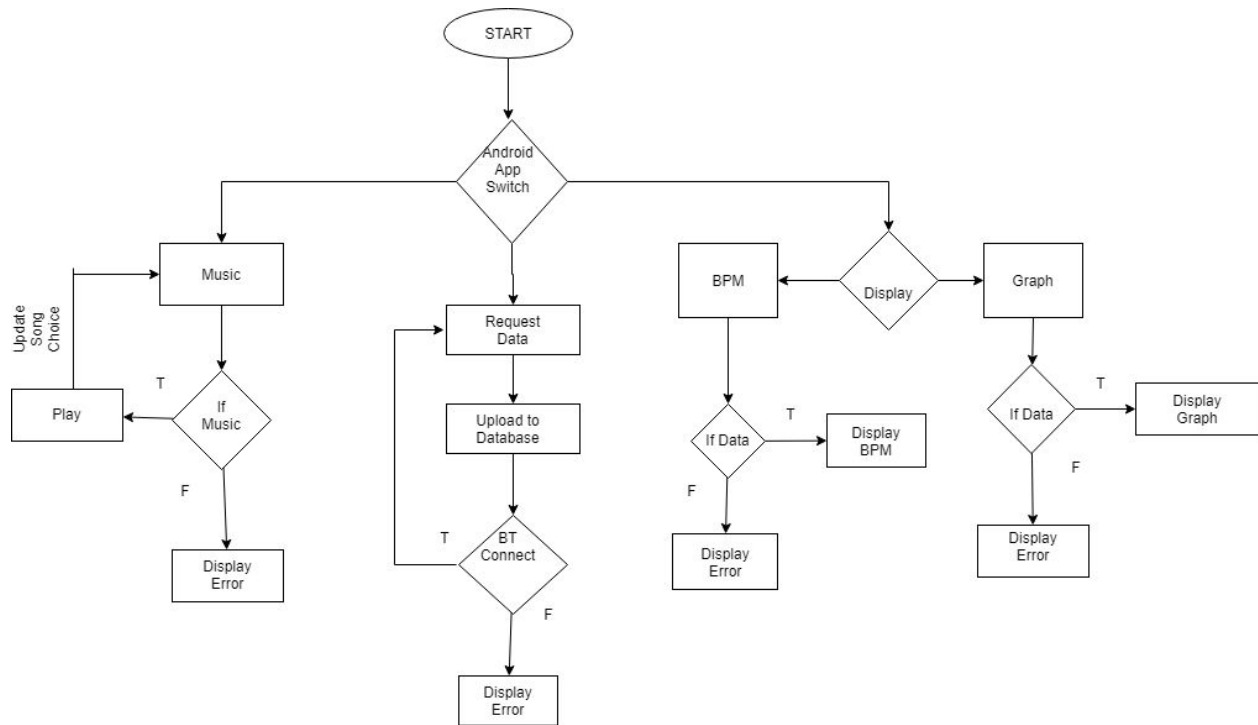


Figure: Flowchart derived from the system sequence diagrams

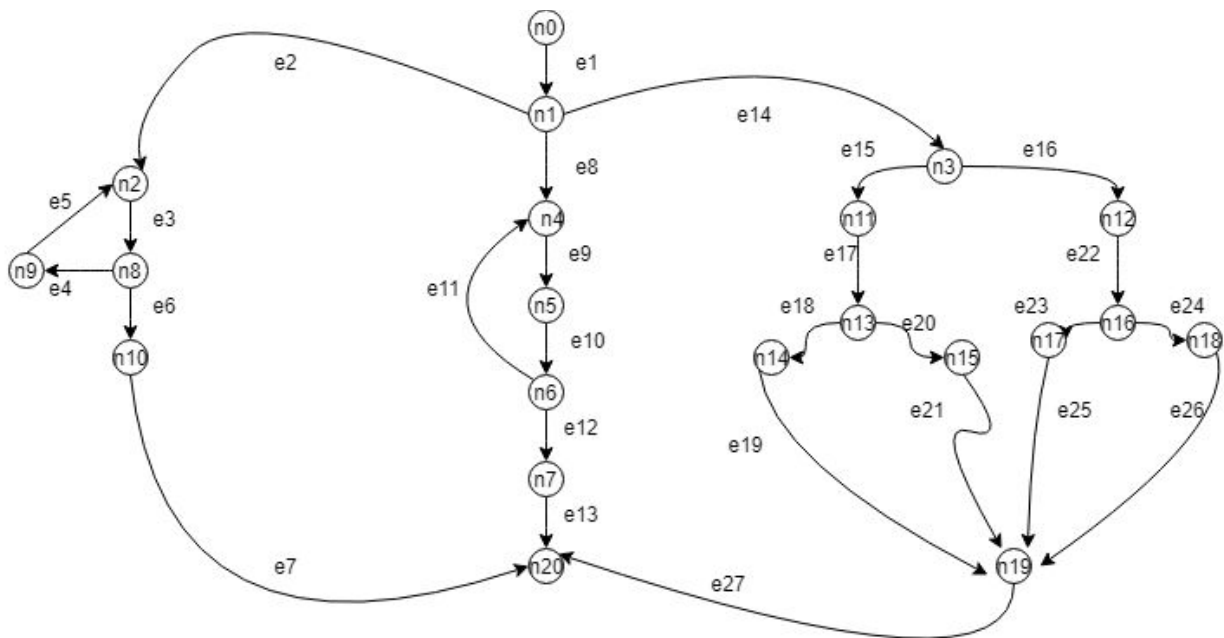


Figure: Graphical representation of program flowchart.

The program graph contains 21 nodes and 27 edges. The nodes and the edges represent the program functional blocks in the flowchart which is in turn derived from the System Sequence Diagrams.

Cyclomatic Complexity of a graph G is given by the relationship “#edges - #nodes + 2”

i.e $V(G) = e - n + 2$.

Plugging the values of e and n to 27 and 21 respectively, we get a cyclomatic complexity of 8. Therefore, Cyclomatic Complexity for the main program of Heart Rate Monitor is 8.

When calculating the UCP, the perceived value of T4 in TCF was set to 4. But as multiple frameworks and modules in the project were factored in, the cyclomatic complexity as seen has risen to 8. This sets the final Technical value to 45.5 and the new TCF becomes 1.055.

Thus the new UCP of the project becomes

$$\text{UCP} = \text{UUCP} \times \text{TCF} \times \text{ECF} = 105 \times 1.055 \times 1.00 = \mathbf{110.775}$$

9. References

- [1] Stavros Asimakopoulos , Grigorios Asimakopoulos , and Frank Spillers, Motivation and User Engagement in Fitness Tracking: Heuristics for Mobile Healthcare Wearables
- [2] Stefan Koelsch and Lutz Jancke, Music and the heart, European Heart Journal (2015) 36, 3043–3048
- [3] Using music to tune the heart URL:
https://www.health.harvard.edu/newsletter_article/using-music-to-tune-the-heart
- [4] Heart rate change from awake stage to sleep stage URL:
<http://www.livestrong.com/article/105256-normal-heart-rate-sleeping/>
- [5] Masao Yaso, Atsuo Nuruki, Seiichi Tsujimura, and Kazutomo Yunokuchi, Detection of REM sleep by heart rate, Proceedings of The First International Workshop on Kansei
- [6] Previous project:
<http://www.ece.rutgers.edu/~marsic/books/SE/projects/HealthMonitor/2014-g12-report3.pdf>
- [7] Harmat L., Takács J., Bodizs R. (2008). Music improves sleep quality in students. URL:
<https://doi.org/10.1111/j.1365-2648.2008.04602.x>
- [8]Wagner, Richard, "The Ride of The Valkyrie"
URL:http://ia802301.us.archive.org/31/items/RideOfTheValkyries/ride_of_the_valkyries_2.mp3
- [9]Macaroni Union, "Weightless" URL: <https://www.youtube.com/watch?v=UfcAVejslrU>

- [10]Debussy, Claude, "Clair De Lune" URL:
<http://www.orangefreesounds.com/clair-de-lune-piano/>
- [11]Chopin, Frederic, "Nocturne in E Flat Major Op. 9 No. 2" URL:
<https://www.youtube.com/watch?v=5ZUw78FXpG4>
- [12]Mozart, Wolfgang Amadeus, "Canzonetta Sul-aria" URL:
<https://www.youtube.com/watch?v=Fc3fmSSUwck>
- [13]Beethoven, Ludwig van, "Moonlight Sonata(1st mvt)" URL:
https://www.8notes.com/school/mp32/piano/moonlight_sonata.mp3
- [14]Offenbach, Jacques, "Can Can" URL: "<https://www.youtube.com/watch?v=4Diu2N8TGKA>"
- Kunzel, Erich, "William Tell Overture Finale" URL:
"<https://www.youtube.com/watch?v=c7O91GDWGPU>"
- [15]Beethoven, Ludwig, "Fur Elise" URL: https://www.youtube.com/watch?v=k_UOuSk1NL4
- Bach, Johann Sebastian - Suite No. 2 in B minor URL :
<https://www.youtube.com/watch?v=4ufehp7gULA>
- [16]La Stravaganza, Op. 4, Concerto No. 2 in E Minor, RV 279: I. Allegro URL:
<https://www.youtube.com/watch?v=tVQkrEY2isI>
- [17]Bach, Johann Sebastian, "Air" URL: <https://www.youtube.com/watch?v=pzlw6fUux4o>
- [18]Cockerton, T., Moore, S., & Norman, D. (1997). Cognitive Test Performance and Background Music. *Perceptual and Motor Skills*, 85(3_suppl), 1435–1438.
<https://doi.org/10.2466/pms.1997.85.3f.1435>
- [19]DeLoach, Alana G., Carter, Jeff P. and Braasch, Jonas, "Tuning the cognitive environment: Sound masking with "natural" sounds in open-plan offices", DOI link:
<https://doi.org/10.1121/1.4920363>
- [20]Baker, Max, "How Music Could Help You Study Better", URL:
<https://www.independent.co.uk/student/student-life/Studies/how-music-could-help-you-to-conce ntrate-while-studying-a6907341.html>
- [21]American Roentgen Ray Society. "Baroque Classical Music In The Reading Room May Improve Mood And Productivity." ScienceDaily. ScienceDaily, 26 April 2009.
www.sciencedaily.com/releases/2009/04/090423132615.htm
- [22]Costas Karageorghis and David-Lee Priest – Brunel University, "Music in Sport and Exercise : An Update on Research and Application", URL:
<https://thesportjournal.org/article/music-sport-and-exercise-update-research-and-application/>
- [23]Jabr, Ferris, "Let's Get Physical: The Psychology of Effective Workout Music", URL:
<https://www.scientificamerican.com/article/psychology-workout-music/>
- [24]Thakare AE, Mehrotra R, Singh A. Effect of music tempo on exercise performance and heart rate among young adults. *Int J Physiol Pathophysiol Pharmacol*. 2017;9(2):35-39. Published 2017 Apr 15.
- [25]Oura Crew, "Heart Rate While Sleeping", URL:
<https://ouraring.com/heart-rate-while-sleeping/>

- [26]Fan Feng, Yingshi Zhang, Jun Hou, Jiayi Cai, Qiyu Jiang, Xiaojuan Li, Qingchun Zhao, Bo-an Li, “Can music improve sleep quality in adults with primary insomnia? A systematic review and network meta-analysis”, International Journal of Nursing Studies, Volume 77, 2018, Pages 189-196, ISSN 0020-7489, URL: <https://doi.org/10.1016/j.ijnurstu.2017.10.011>.
- Pickut, Walt, ”What Is a Normal Heart Rate While Sleeping”, URL: <https://www.livestrong.com/article/105256-normal-heart-rate-sleeping/>
- [27]Dr. Michael Breus, “The Power of Music for Sleep and Performance”, URL: <https://www.thesleepdoctor.com/2018/06/04/the-power-of-music-for-sleep-and-performance/>