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

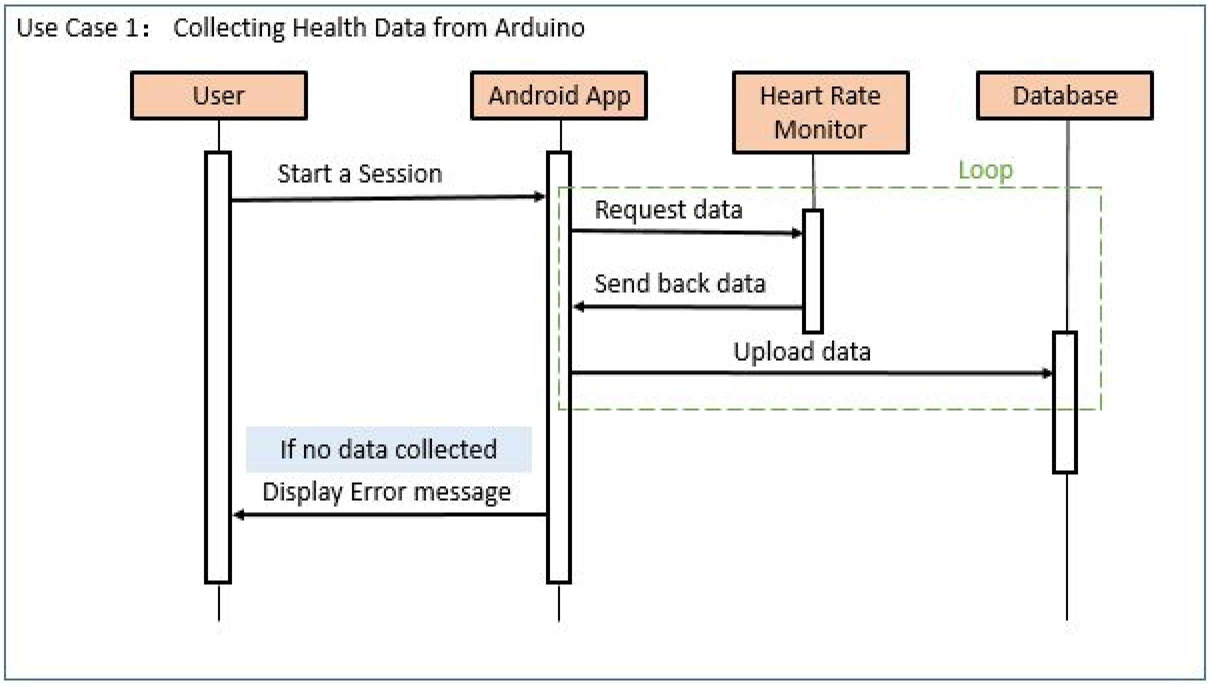
Table of Contents

0. Individual Contribution Breakdown

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 %

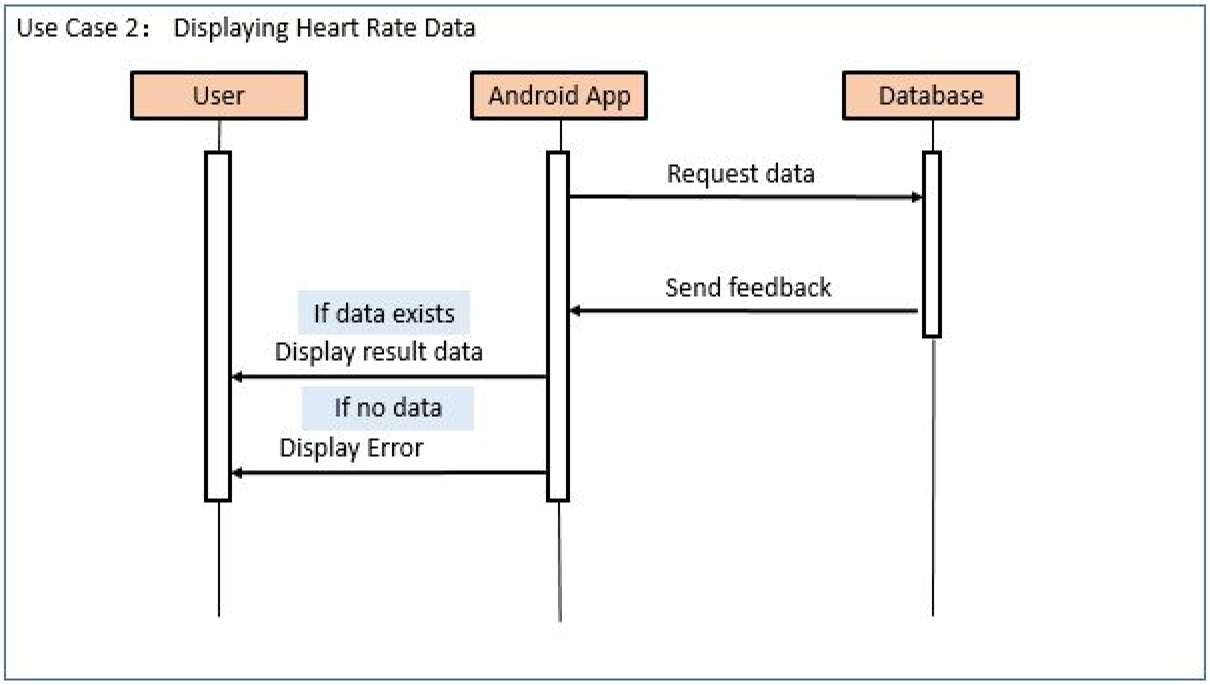
Project Management and Plan of Work	16.67%	16.67%	16.67%	16.67%	16.67%	16.67%	100.00 %
Cyclomatic Complexity	16.67%	16.67%	16.67%	16.67%	16.67%	16.67%	100.00 %
References	16.67%	16.67%	16.67%	16.67%	16.67%	16.67%	100.00 %

1. Interaction Diagrams

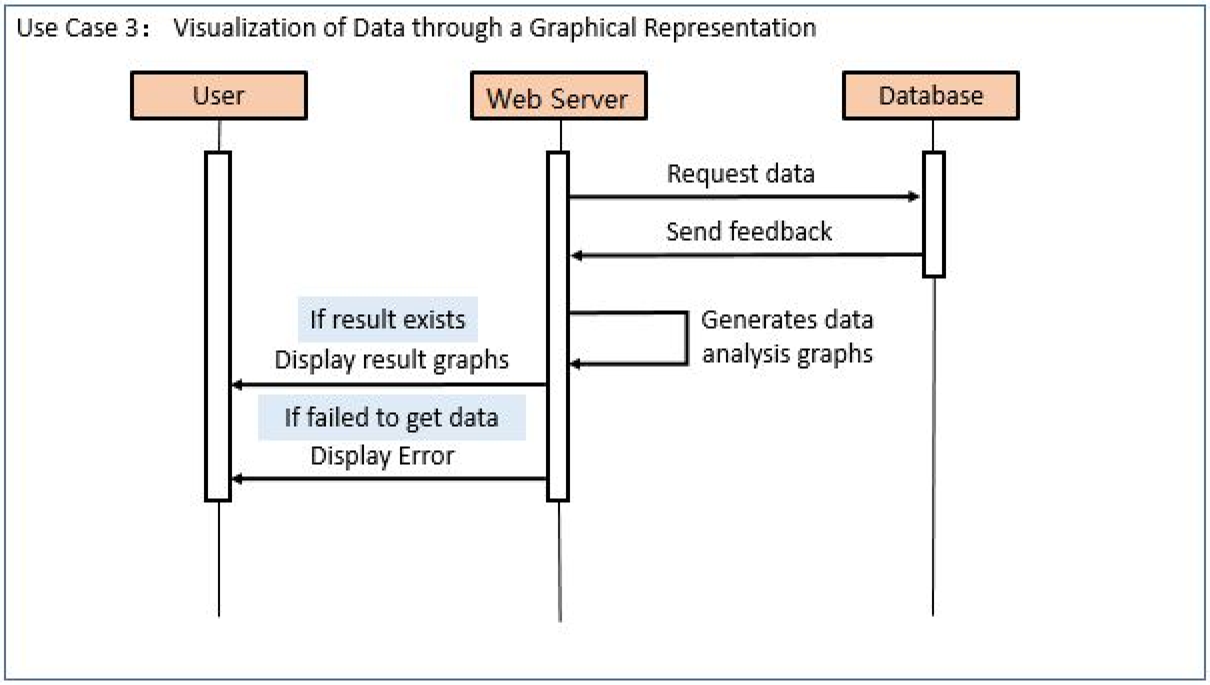


UC1: For our first use case, we wish to record information about the user's heart rate by using Arduino. This includes a timestamp and date of when the heart rate data was recorded, and could also include some of the song metadata such as artist, album, or genre. We also have a Android app to connect 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

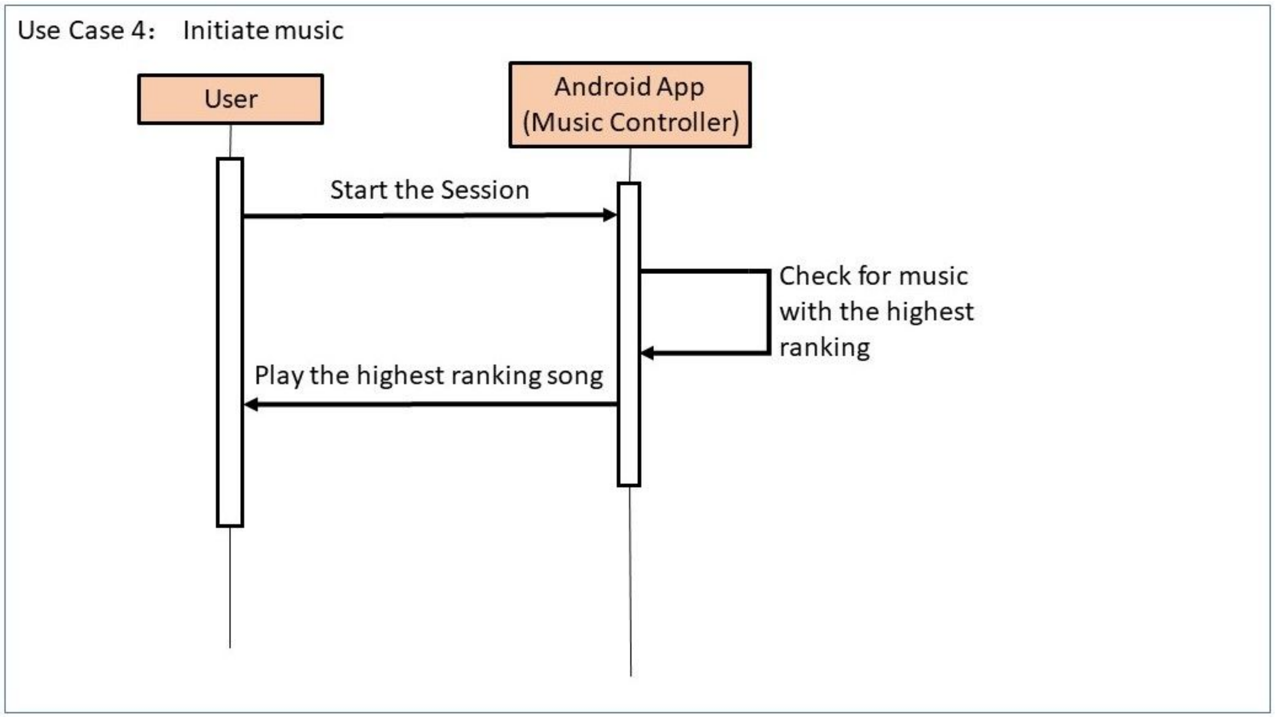
when the Arduino is not functioning correctly and reports an error in measurement, and when user move out his/her figure out from the sensor of Arduino, it will also stop recording.



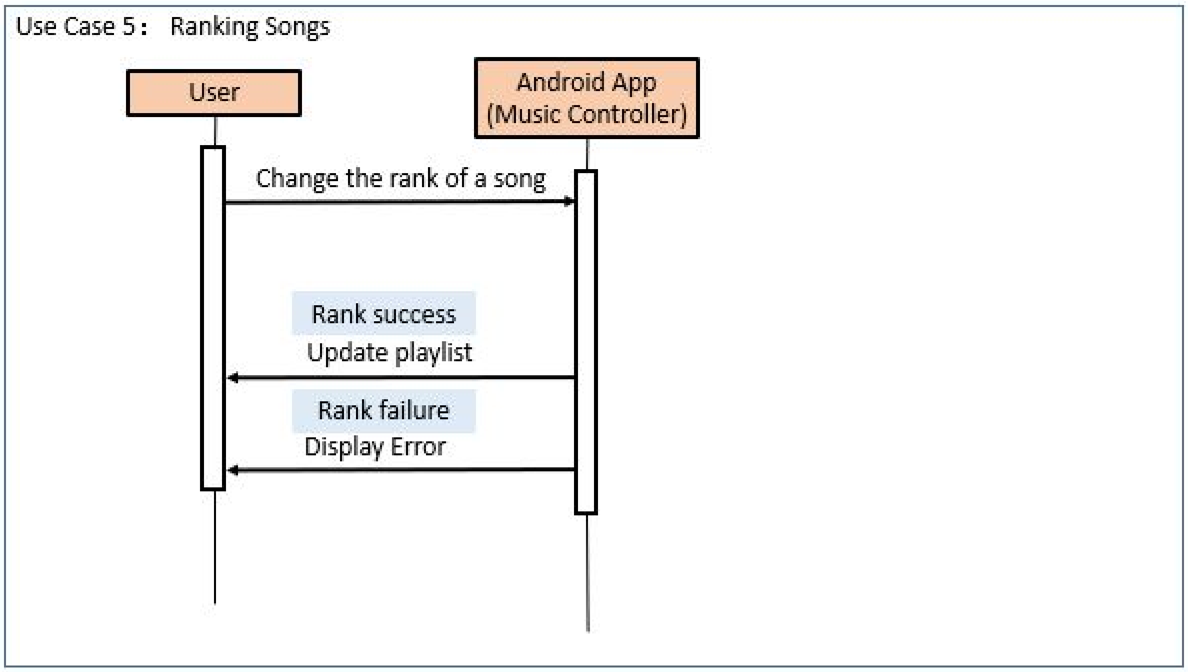
UC2: For our second use case, we are displaying the heart rate data. The database requests the data from the android application. The database sends feedback to the android application. The android application, if the data exists, displays the resulting data to the user. If no data exists, the android 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.



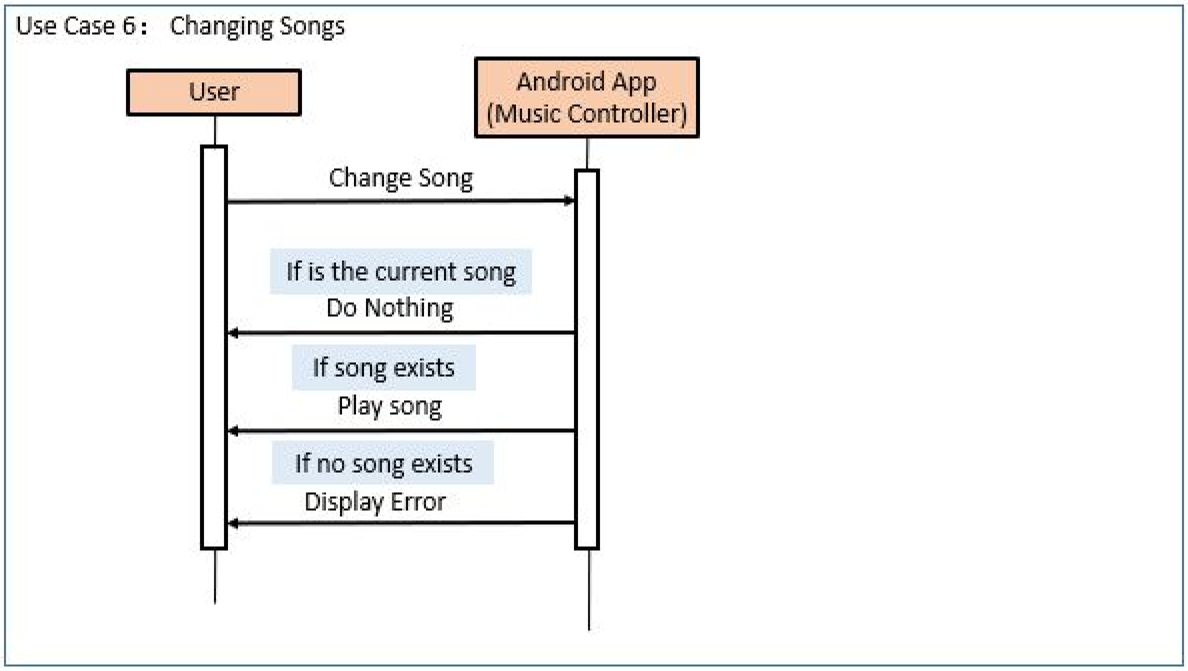
UC3: The User has the hardware which detects the heart-beat at the tip of his finger. This hardware is connected to the mobile application via bluetooth. On receiving the data from the hardware, the application displays it on the activity screen as well as uploads it to the server. The Same server is 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
User
Android App (Music Controller)
Start the Session
Check for music with the highest ranking
Play the highest ranking song

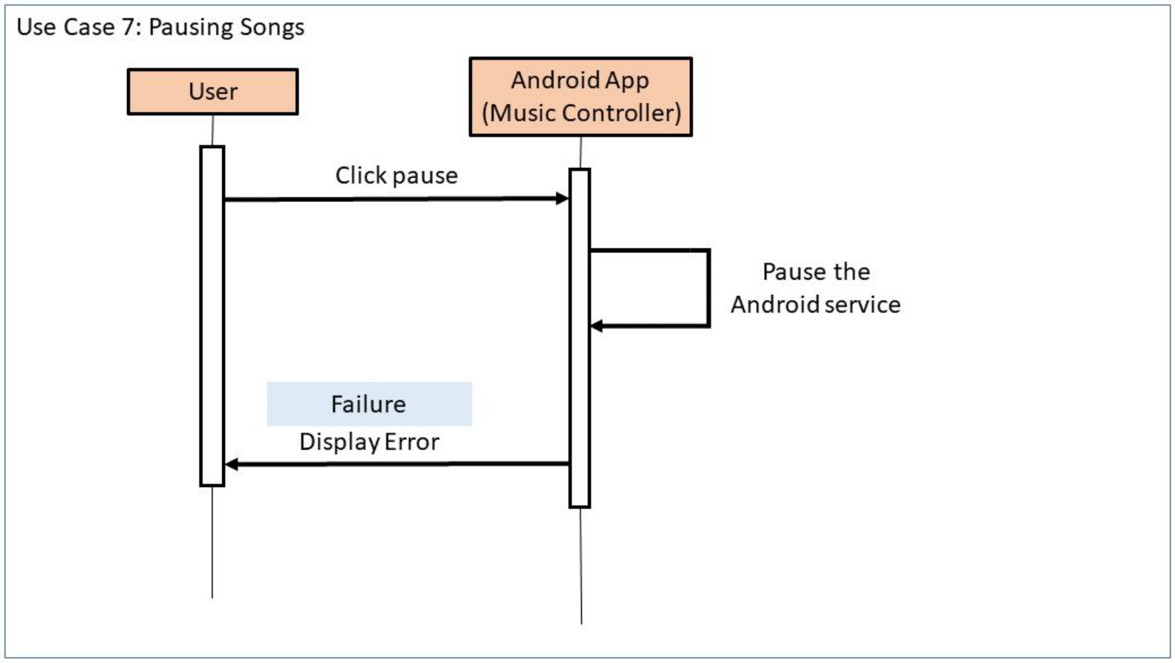
UC4: The user sets the preferred activity. According to the preferred activity the heartbeat range is set that will be ideal for the user. Based on the current heartbeat of the user the song is selected. If the heartbeat is lower than the lower limit of the range the song is selected such that the heartbeat 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.


Use Case 5： Ranking Songs
User
Android App (Music Controller)
Change the rank of a song
Rank success
Update playlist
Rank failure
Display Error

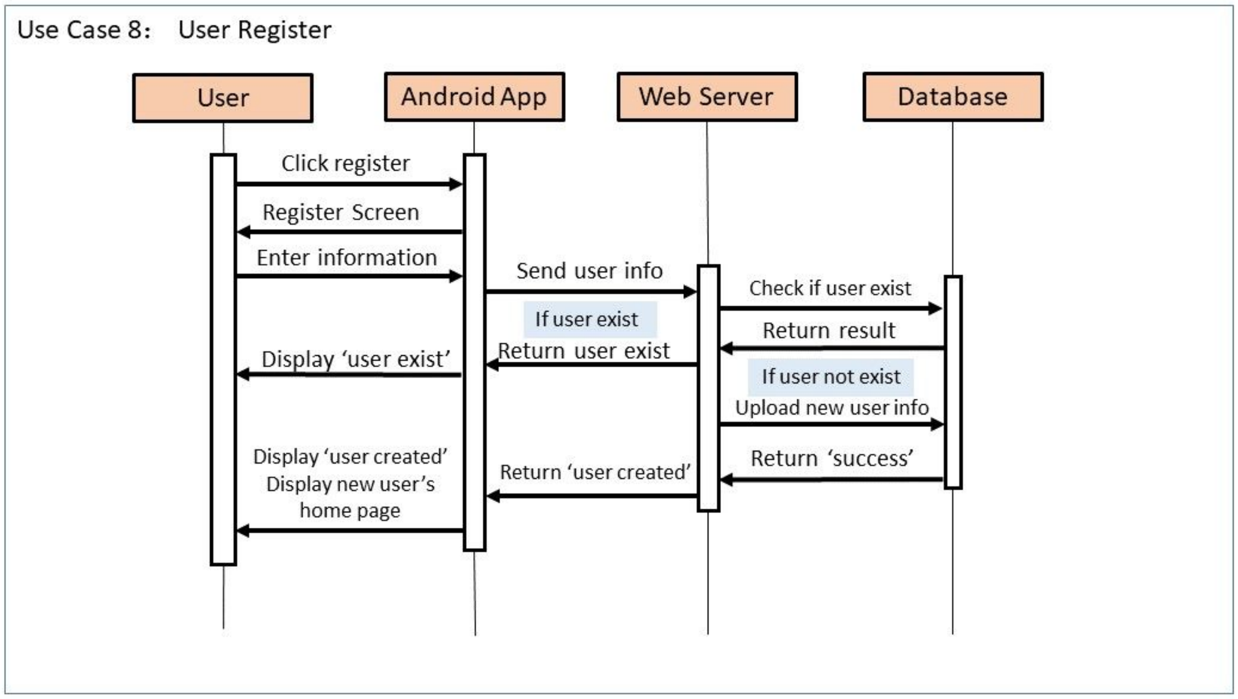
UC5: For this use case, the users are allowed to rank the songs in their library that based on their preferences. Users are able to give their feedback about the songs after every activity. This feedback will be in the form of points out 10. This pointing system will help the application to rank the 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.



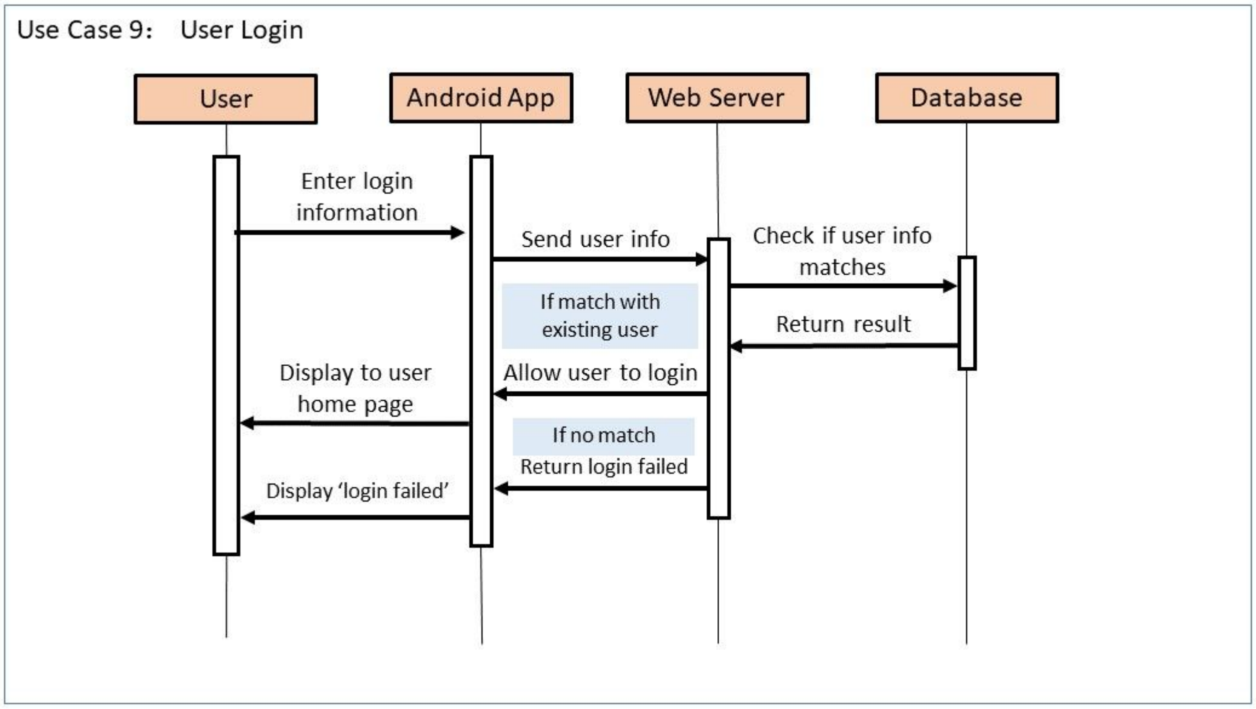
UC6: For the sixth use case, the user has the option to change the song to play in accordance with their preferences. When the user changes a song, the app checks if the selected song is the same as the existing one. In this case, nothing is done. If the song is different, then the app checks for the 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.


Use Case 7: Pausing Songs
User
Android App (Music Controller)
Click pause
Pause the Android service
Failure
Display Error

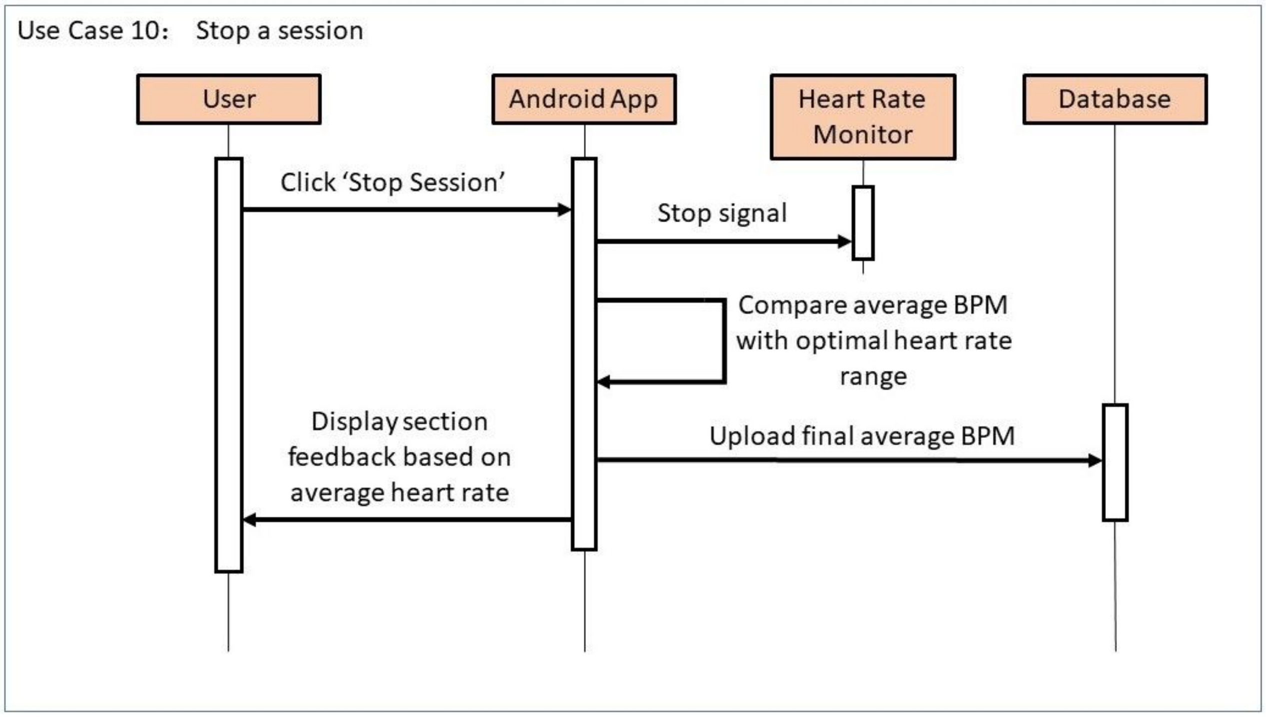
UC7: For the seventh use case, the user has the option to pause the song if there are any interruptions to their routine. When the user pauses a song, the app first checks if a song is even being played. If there is a song playing, the song is stopped and playback is resumed from that point again when 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.


Use Case 10: Stop a session
User
Android App
Heart Rate Monitor
Database
Click ‘Stop Session’
Stop signal
Compare average BPM with optimal heart rate range
Display section feedback based on average heart rate
Upload final average BPM

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

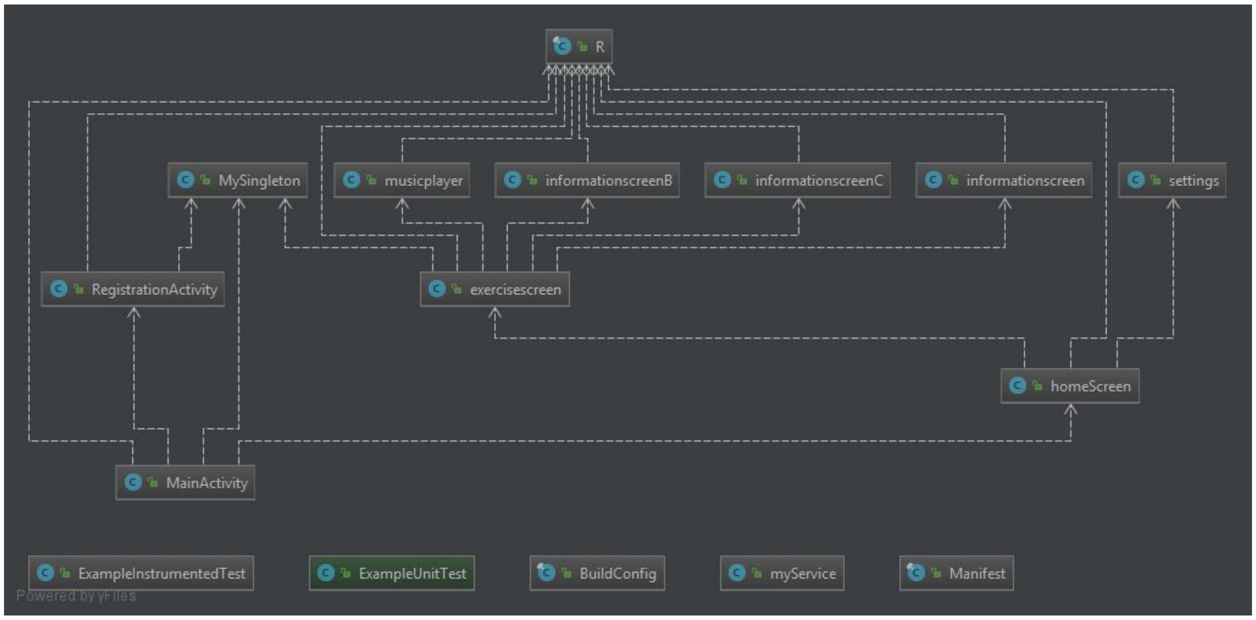


Figure: Class Diagram in general for the main class of the Android app.

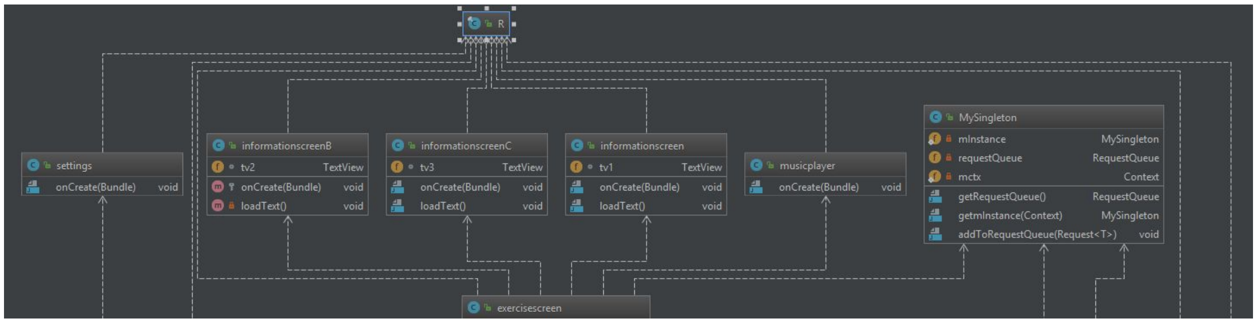


Figure: The first part of Class Diagram for the main Android app in detail.

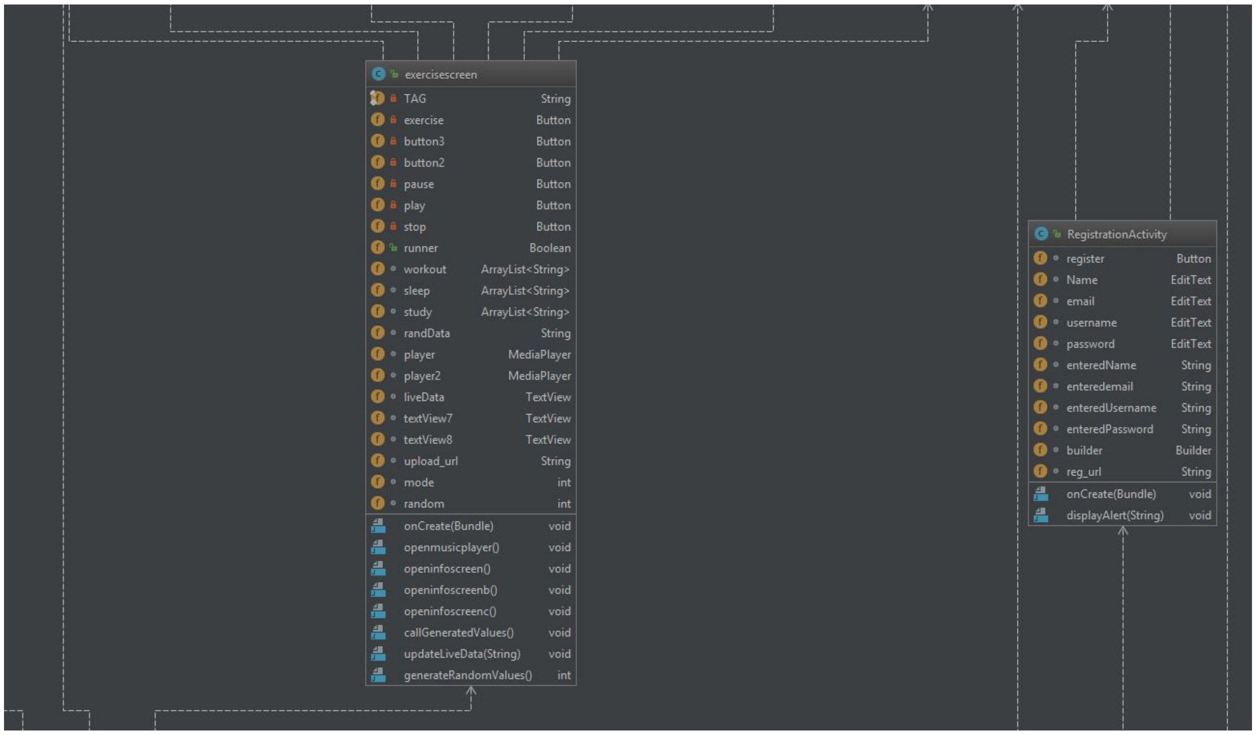


Figure: The second part of Class Diagram for the main Android app in detail.

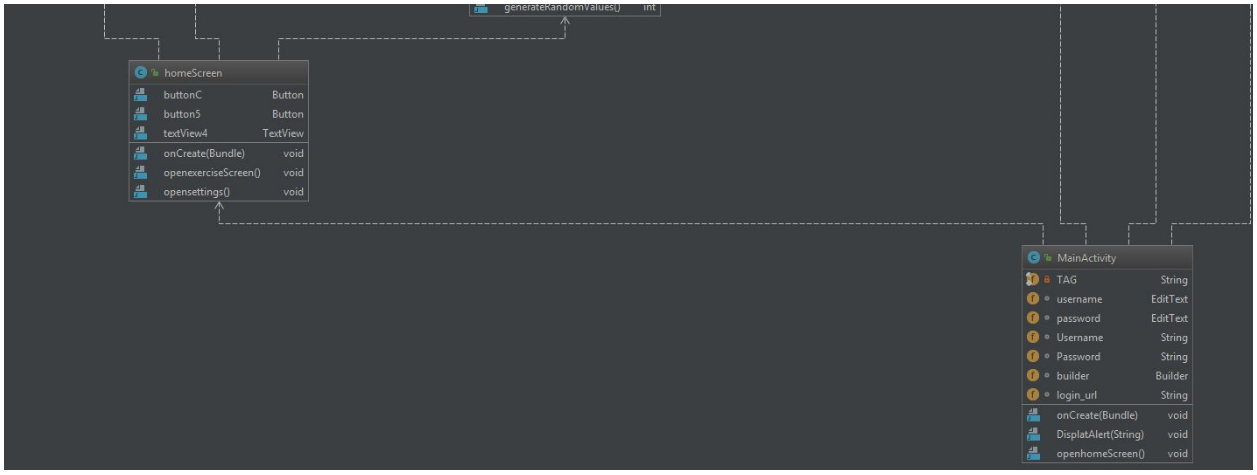


Figure: The third part of Class Diagram for the main Android app in detail.

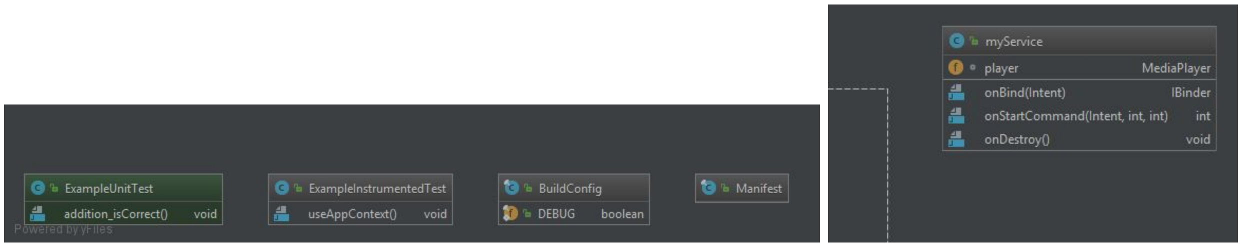
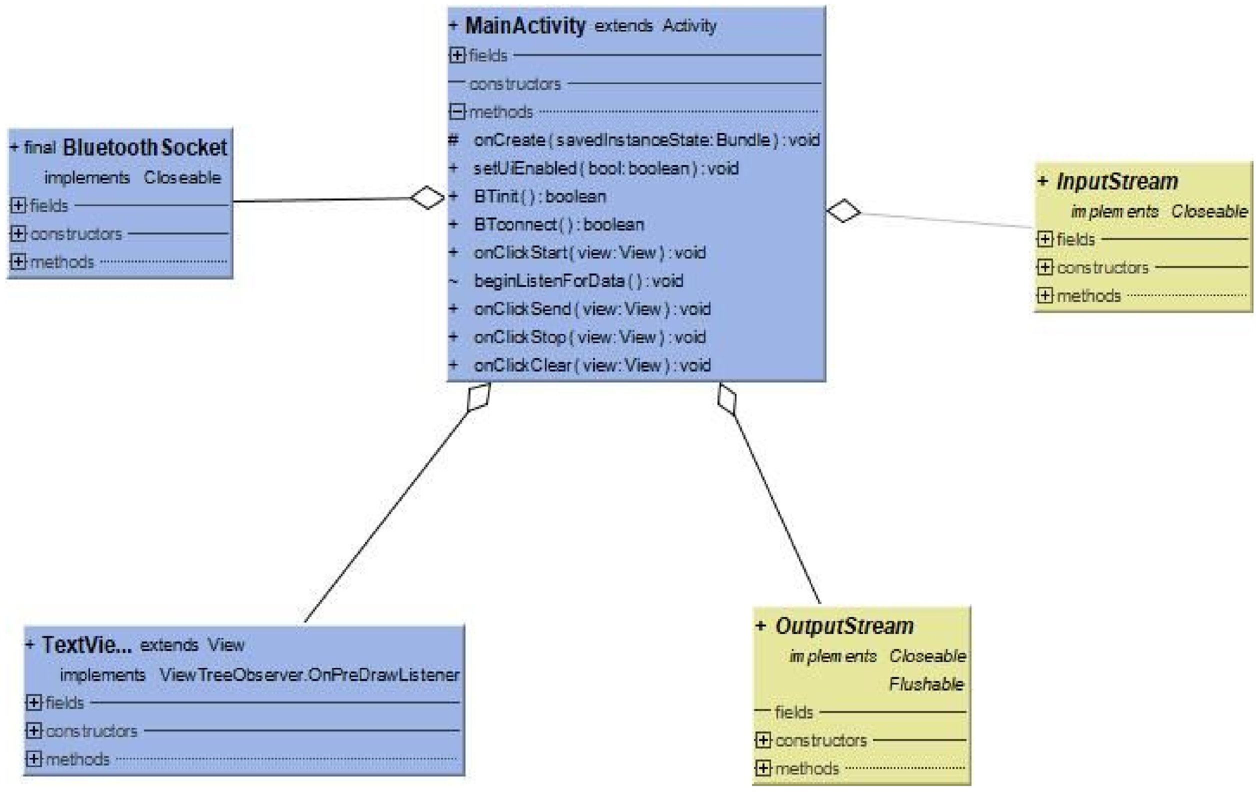


Figure: The final part of Class Diagram for the main Android app in detail.

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.



+ MainActivity extends Activity
fields
constructors
methods
# onCreate (savedInstanceState: Bundle) : void
+ setUiEnabled (bool: boolean) : void
+ BTinit ( ) : boolean
+ BTconnect ( ) : boolean
+ onClickStart (view: View) : void
~ beginListenForData ( ) : void
+ onClickSend (view: View) : void
+ onClickStop (view: View) : void
+ onClickClear (view: View) : void
+ final BluetoothSocket implements Closeable
fields
constructors
methods
+ InputStream implements Closeable
fields
constructors
methods
+ TextVie... extends View implements ViewTreeObserver.OnPreDrawListener
fields
constructors
methods
+ OutputStream implements Closeable Flushable
fields
constructors
methods

Figure: Class Diagram for Bluetooth Functionality

b. Data Types and Operation Signatures

Settings
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 .

Information Screen
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 . • *loadText() : void*. This function load Text from the text box.

Music player
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 .

MySingleton
Functions: • *getRequestQueue() : RequestQueue*. This function is to post a request with query to get information from the DataBase. • *getmInstance(Context): MySingleton.* This function is use to get the instance. • *addToRequestQueue(Request<T>): void.* 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
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 . • *openmusicplayer(): void.* This function call up the internal music player. The user will able to use the basic music player functions in the newly called activity. • *openinfoscreen(): void.* 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. • *callGeneratedValues(): void.* 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. • *updateLiveData(String): void.* This function is called to upload the current heartbeat data to the server. • *generateRandomValues(): int.* 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
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.

c. Traceability Matrix

	Class								
Domain Concepts	Main Activity	Home Screen	Settings	Inform-ation Screen	My Singl-eton	Registr -ation Screen	Exerci -se 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

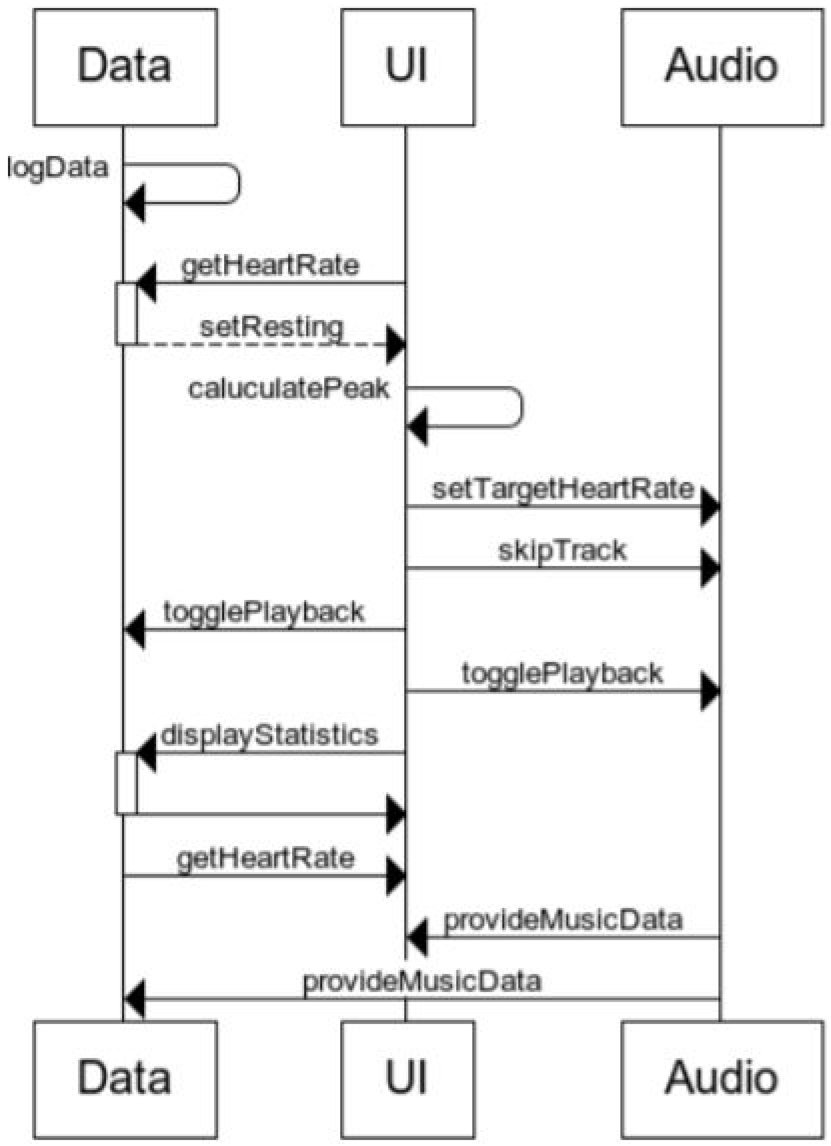
3. System Architecture and System Design

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 the 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

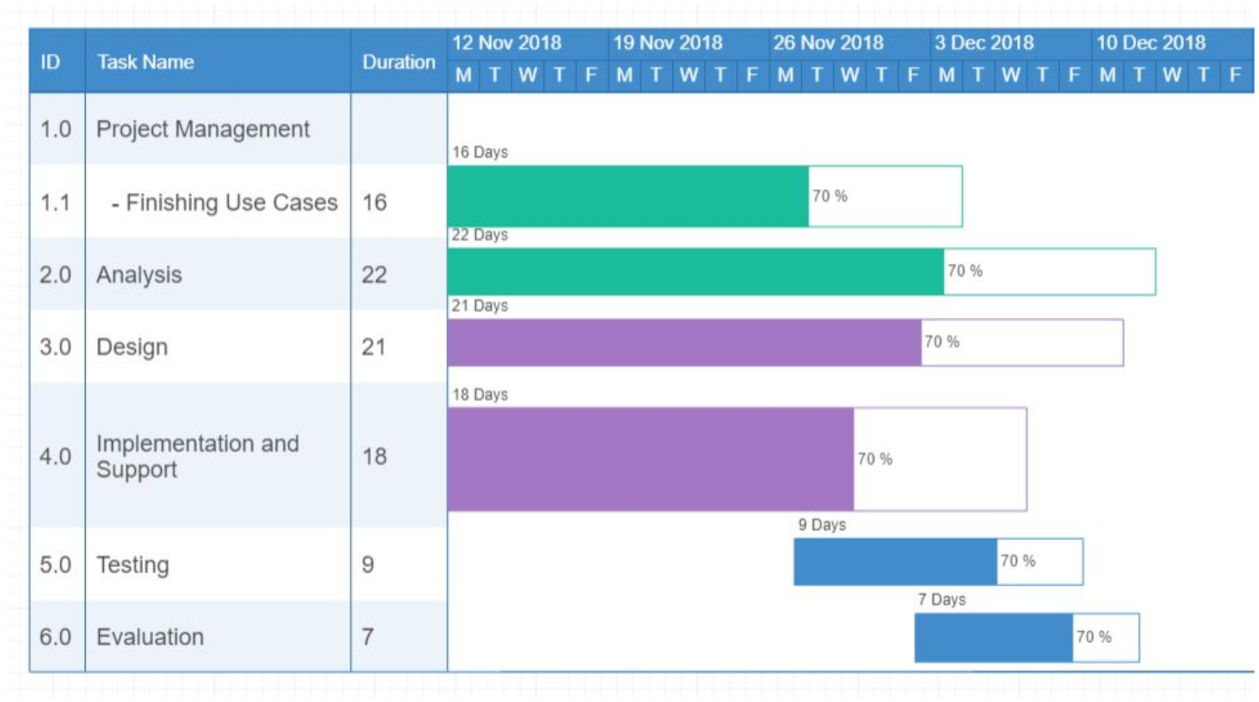
The use cases that have been implemented include:

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:



ID	Task Name	Duration
1.0	Project Management	
1.1	- Finishing Use Cases	16
2.0	Analysis	22
3.0	Design	21
4.0	Implementation and Support	18
5.0	Testing	9
6.0	Evaluation	7

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

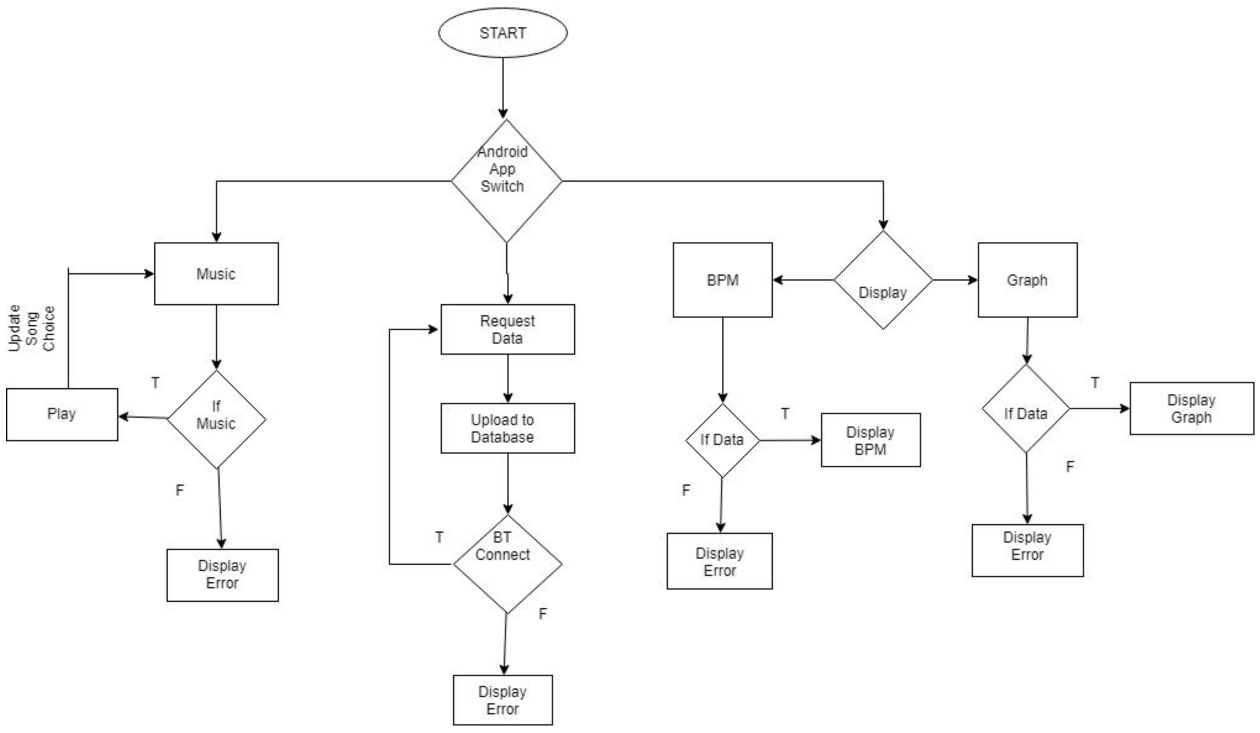


Figure: Flowchart derived from the system sequence diagrams

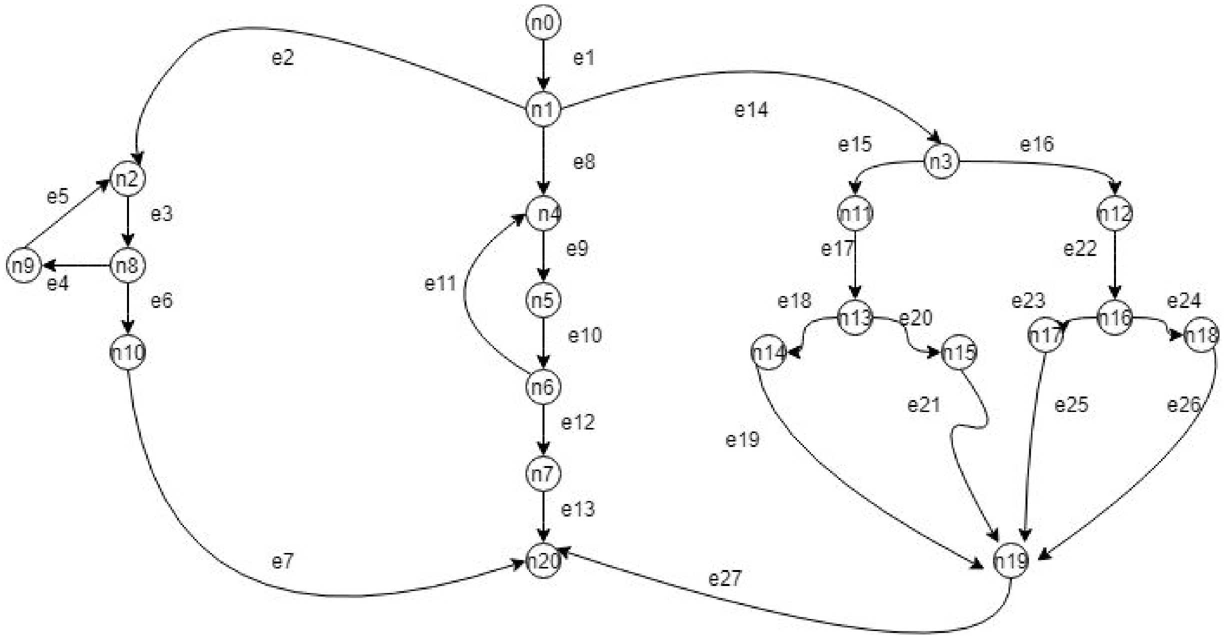


Figure: Graphical representation of program flowchart.

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
UCP = UUCP x TCF x ECF = 105 x 1.055 x 1.00 = **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_UOuSklNL4
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-concentrate-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/