# Software Requirements Specification

## “OffScore”

Prepared by

Wenhe Xu (Eric)

Zhuoyang Fu (Jeffery)

Yihang Liu（Jessica）

Wenzhou-Kean University

24, February, 2022

# Table of Contents

# 1. Introduction

## 1.1 Purpose

The purpose of this document is to build a music recommendation application -- “Offscore”, which can provide users with songs suitable for their environment and emotions based on their real-time heart rate and location.

## 1.2 Document Conventions

This document follows APA Format. Bold-faced text has been used to emphasize section and sub-section headings. Moreover, this document uses the following conventions.

| **Convention** | **Description** |
|---|---|
| AI | Artificial Intelligence |
| BPM | Beat Per Minute |
| CSS | Cascading Style Sheets |
| HTML | HyperText Markup Language |

## 1.3 Intended Audience and Reading Suggestions

This document is to be read by the development team, testers, and documentation writers. This document informs the reader how they can effectively develop and test this application. The second part of this document illustrates the product's functions and points out the implementation requirements. The third part mentions some specific requirements. Please read this document in the order of the title serial number, and give timely feedback when experiencing problems.

## 1.4 Product Scope

AI technology makes it possible for music listeners to quickly get the music they want in a large number of songs. The goal of “Offscore” is to develop a precise recommendation service, which can provide users recommended music based on real-time heart rate and location. This project consists of a music recommendation system and a wristband which can measure real-time heart rate and location. Wristband provides user data to the recommendation system and the recommendation system can find the qualified music to use.

# 2. Overall Description

## 2.1 Product Perspective

The fundamental part of this recommendation system is a transmission between a smart bracelet and a smartphone. In fact, traditional music recommendations usually read log’s streams and usage data as variables to achieve the functionality. To improve the accuracy of music recommendation, this software aims to generate a music song list for the client in real time that matches the current environment through a sophisticated recommendation algorithm, and transmit it back to the client's bracelet to provide reference information. The designer should carefully consider the related database, hardware interface, software interface and AI recommendation algorithms to implement the whole system.

## 2.2 Product Functions

The core functionality of this recommendation system is to implement the accurate music recommendation by interacting with real-time personal data. The client can provide the basic information through Bluetooth transmission to the software. The whole system will generate a music song list in real-time so that it can match the current environment and person’s emotion. Users will be able to benefit from the formation of efficient and real-time information communication between the user's bracelet and the software.

## 2.3 User Classes and Characteristics

The users of this system are people who hope to get more accurate music recommendation functionalities. These users are able to own the private accounts. Meanwhile, they will have the permission to enter the accounts to select music, and view the records of historical song lists.

## 2.4 Implementation Constraints

The music recommendation system requires a user with a smartphone at least having playback function and a wearable device which can detect the user’s heart rate, location and other necessary data in real time. For the goal of compatibility, the occupied RAM by the system while it is running shall be no more than 512 MB. The storage space of the phone may increase when the users add more music, but the maximum size of the system shall be no more than 150MB. Moreover, the application may use MySQL as the database, and is mainly developed by HTML, CSS, and JavaScript. For the consideration related to the security issue, the system may use data encryption techniques(like SSL) to protect the data. Another important implementation constraint is to select the AI recommendation algorithm. This application needs to design an advanced algorithm which can find music corresponding to the tag.

# 3. Specific Requirement

## 3.1 External Interfaces

Our music recommendation system “OffScore” supports multiple types of interfaces, which are user interface, software interface and hardware interface.

### 3.1.1 User Interface

The user interface of the software is designed by Adobe Illustrator and made by Kivy.

The environmental requirements of Adobe Illustrator and kivy: MacOS 11, multi-core Intel processor, x86-64 architecture, 8 GB RAM, 512 GB SSD, python3.8.10. Or Apple M1 Pro processor, with 14-cores GPU.

* It needs an international internet connection. An Adobe ID, and a license agreement are required to enable and use Adobe Illustrator. This product integrates or allows access to specific Adobe or third-party hosted online services. The browser should use HTTP when accessing the protocol, use port number 80, and require an IPv4 logical system address.

### 3.1.2 Software Interface

1. The software interacts with the user interface, allowing the user to drag or select one or several images.

2. The software uses tag preference as the key to generate recommended music lists.

3. The software uses preference matrix as core mechanism to enable personalized recommendation.

4. The software provides a database interface, driven by MySQL, allowing developers to write users' personal profiles, pulse stats, and preference into the database.

5. The software interacts with the hardware interface, allowing the system to collect, analyze and feedback user data.

6. The software provides a recommendation interface, which is based on collaborative recommendation mode and  to make personalized tag-based music recommendations for users and determine the priority of the next association.

7. The software provides a monitoring interface to allow the superior to monitor and control the lower-level user interface, and the superior has full control authority.

8. At the same time, based on the Android11 operating system, the software will call the application layer interface of the operating system, perform partition storage, save the application proprietary files to the exclusive location, and obtain the permission to automatically remake and access the background location information.

### 3.1.3 Hardware Interface

Since the application must run on the Internet, all hardware that needs to connect to the Internet will be the hardware interface to the system, such as modems, wide area networks, Ethernet cross cables.

1. In terms of hardware device selection, we choose the Xiaomi wristband for data collection as one option. The data format of Xiaomi wristband's open interface is as follows:

| | |
|---|---|
| TYPE_BASAL_METABOLIC_RATE<br><br>(com.xiaomi.micloud.fit.calories.bmr) | Instantaneous value, one-dimensional data type<br><br>Each data point represents the basal metabolic rate of energy consumption when the user is at rest. The unit is kcal/day(kcal/day) |
| TYPE_BODY_FAT_PERCENTAGE<br><br>(com.xiaomi.micloud.fit.body.fat.percentage) | Instantaneous value, one-dimensional data type<br><br>Each data point represents a test result. This measure is the ratio of total fat to body weight |

| | |
|---|---|
| TYPE_CALORIES_EXPENDED<br><br>(com.xiaomi.micloud.fit.calories.expended) | Non-instantaneous, one-dimensional data type<br><br>Each data point represents the number of calories consumed during the time interval between the start and end of the data point, in kilocalories. The data value is of type float.<br><br>Both startTime and endTime require an assignment that represents the interval between calories consumed |
| TYPE_HEART_RATE_BPM<br><br>(com.xiaomi.micloud.fit.heart_rate.bpm) | Instantaneous value, one-dimensional data type<br><br>Each data point represents the instantaneous measurement value of a heartbeat, which is the heartbeat number one minute ahead of the measurement moment |
| TYPE_HEIGHT<br><br>(com.xiaomi.micloud.fit.height) | Instantaneous value, one-dimensional data type<br><br>Each data point represents a measurement of the user's height in meters |

| | |
|---|---|
| TYPE_POWER_SAMPLE<br><br>(com.xiaomi.micloud.fit.power.sample) | Instantaneous value, one-dimensional data type<br><br>Each data point represents an instantaneous measurement of power in watts. Field values are of type float |
| TYPE_WEIGHT<br><br>(com.xiaomi.micloud.fit.weight) | Instantaneous value, one-dimensional data type<br><br>Each data point represents the user's weight in kilograms at the time of measurement |

2. Access to Android Bluetooth API, which allows effective communication between wristband and smartphone. In order to use the Bluetooth feature in your app, you need to declare at least one Bluetooth permission: BLUETOOTH and *BLUETOOTH_ADMIN*. In order to perform any Bluetooth communication, such as requesting a connection, accepting a connection, and transferring data, you must request the BLUETOOTH permission. In order to initiate device discovery or to control Bluetooth settings, you must request the *BLUETOOTH_ADMIN* permission. Most applications require this permission just to be able to discover local Bluetooth devices. Other functions authorized by this permission should not be used unless the app is a "power controller" to modify Bluetooth settings by user request.

3. LILYGO® T-Impulse LORA GPS could be a feasible option as a programmable wear that enable GPS positioning and pulse detection. The devices are highly compatible with Arduino IDE and allow further extensive development.

4. Bracelet Edition is a variant of SoftRF DIY GA proximity awareness device implemented in a small, lightweight wristband form factor. It is supposed to require minimal skills from a user to begin with and gives an opportunity to evaluate if the SoftRF technology is appropriate for your needs.

The Bracelet is based upon a partner's hardware - LilyGO®TTGO T-Impulse product, which is available for direct purchase from the manufacturer at a more or less reasonable price (30+ USD).

5. Arduino IDE is necessary for development based on our programmable wristband. An LK8000 application also needs to be downloaded to ensure the GPS module could work.

6. Location Permission. Our app needs to access the user's location information and must request permission by adding the relevant Android location permission to the app as the description in the link:
https://developer.android.com/reference/android/location/package-summary

Android provides two location permissions: *ACCESS_COARSE_LOCATION and ACCESS_FINE_LOCATION*. This permission determines the accuracy of the location returned by the API. Developers need to request at least one of the Android location permissions:

- *android.permission.ACCESS_COARSE_LOCATION* - Allows the API to use Wi-Fi or mobile data traffic (or both) to determine the device's location. The location accuracy returned by the API is roughly equivalent to a city block.
- *android.permission.ACCESS_FINE_LOCATION* - Allows the API to determine location as accurately as possible using available location providers including Global Positioning System (GPS), as well as Wi-Fi and mobile data traffic.

7. Runtime Permission: Android 6.0 (Marshmallow) introduces a new permission handling model that simplifies the process for users to install and upgrade apps. This app targets API level 23 or higher and is therefore suitable for this new permissions model.

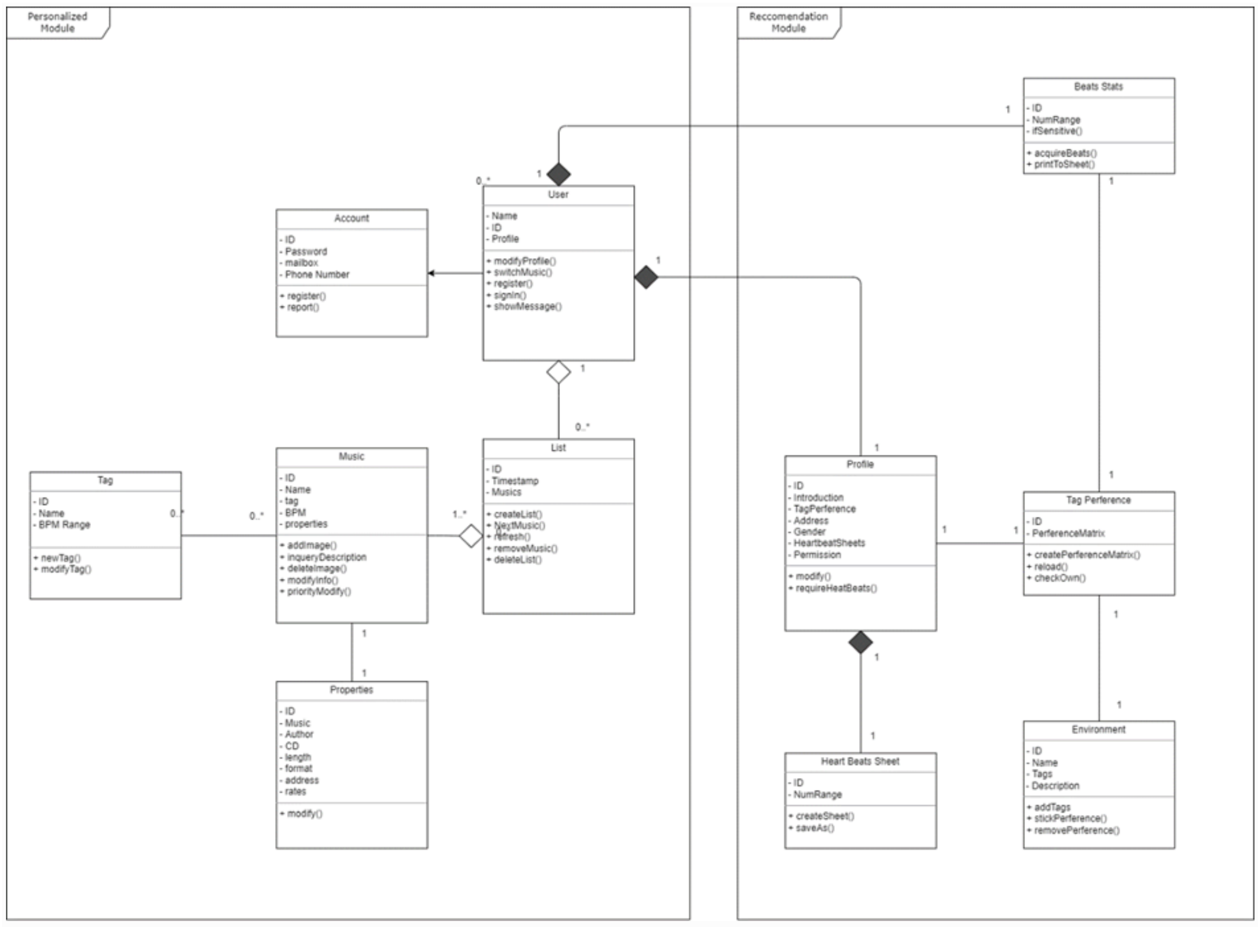
The app must check at runtime to see if it has the necessary permissions, and if not, request the appropriate permissions. The system displays a dialog to the user requesting permission.

For the best user experience, be sure to request permissions in context. If location information is critical for your app to function properly, you should request location permission when your app starts. A good practice is to display a warm welcome screen or wizard to let the user understand why the permission must be granted.

Location permissions should be requested when the app performs an action that requires the permission. Apps must handle cases where the user doesn't grant permissions appropriately. For example, if a specific feature requires the permission, the app can disable that feature. If the permission is essential for the app to function properly, the app can disable all of its functionality and notify the user that the permission needs to be granted.

## 3.2 Functional Requirements

The core function of this system is heart rate-centered music recommendation, and based on the user's personal preferences, combined with the priority system to make composite recommendations. The class diagram of this system is shown below：



### 3.2.1 Account System Function

The system shall provide a visual account login interface.

The system shall provide the modification interface of profiles of personal accounts and allow users to unregister or remove the privacy-relied data consequently.

The system shall automatically remember the preference of the current user and transmit the data to the server.

The system shall provide the unregister function to enable users to delete all privacy-related documents and data.

### 3.2.2 Global Initialization Function

The System shall perform a series of initializations upon first use of the system and check the integrity of the system's local files and whether the Internet is accessible.

### **3.2.3 Pulse Detection Function**

The system shall use the interface of wearable devices such as bracelets to detect the user's heart rate. If necessary, the user's action pattern can be determined by judging the user's cadence. When it is determined that the user is in the jogging stage or the riding stage, different music recommendation feedbacks are made respectively. If the user is relatively still but the heart rate changes greatly, the user is alerted.

### **3.2.4 Pulse Record Function**

The system shall be able to make a continuous heart rate sequence based on the data obtained in 3.2.3, and provide a solution for the visualization of the heart rate sequence.

The system shall be able to synchronize the user's heart rate data to the online database, and allow subsequent music recommendation modules to call the data to make personalized recommendations.

The system shall be able to provide a functional interface for users to save their own heart rate changes.

### **3.2.5 Music Collection Function**

The system shall allow users to upload local songs by themselves, or read the user's local songs as a music library.

The system shall refer to the online music library of other music players or read the online music link provided by the user.

### **3.2.6 Music Tagging Function**

The system shall automatically tag the collected music according to the preset BPM table, and ensure that the BPM corresponding to each tag is relatively close.

The system shall allow system administrators to manually tag music

#### 3.2.7 Music Prioritizing Function

The system should prioritize music based on how long the user listens to, as well as personal factors such as the number of clicks.

The system should set the corresponding tag priority according to the style of music the user listens to the most.

The system should use the label preference matrix to assign different label weights.

The system should automatically recommend music under the corresponding category based on the tag preference matrix.

#### 3.2.8 Tag Define Function

The system shall allow system administrators to set new tags and customize the content of tags and the corresponding range of BPM.

The system shall allow administrators to modify and edit the tag itself.

#### 3.2.9 Environment Define Function

The system should provide a series of preset environmental occasions, corresponding to the labels.

The system should allow administrators to set their own scenarios and associate them with label settings.

The system should determine the corresponding tag preference matrix in different environments, and distinguish from it what type of tags should be recommended in this scenario.

#### 3.2.10 Pulse Mapping Function

The system should provide users with an interface to save their heart rate status.

The system should map the heart rate to the Tag one-to-one.

The system should assign tags to all heart rate states.

### **3.2.11 Environment Detection Function**

The system should call the GPS positioning system to determine the user's location.

The system should use Android's location permissions.

The system should determine the location of the user based on their geographic coordinates.

The system should determine the approximate label based on the user's location.

### **3.2.12 Compound Recommendation Function**

The system should use three tag preference matrices of "environment, user personality, and heart rate" to determine the recommendation algorithm when the playlist is finally generated.

The system should set weights and algorithms for three different tag preference matrices, and finally determine which songs should be included in the playlist.

### **3.2.13 Music List Generation Function**

The system shall generate the music list after analyzing the user's behavior.

The system shall generate a music list which only contains a few or one music.

The system shall generate a music list dynamically and refresh the list every 30 seconds.

### **3.2.14 Music Player Function**

The system should be able to call an external music player to ensure that its music reading, playing and switching functions can work normally.

### **3.2.15 Message Synchronize Function**

The system should be able to communicate with the programmable bracelet and transmit current song playback information to the bracelet.

## 3.3 Performance Requirements

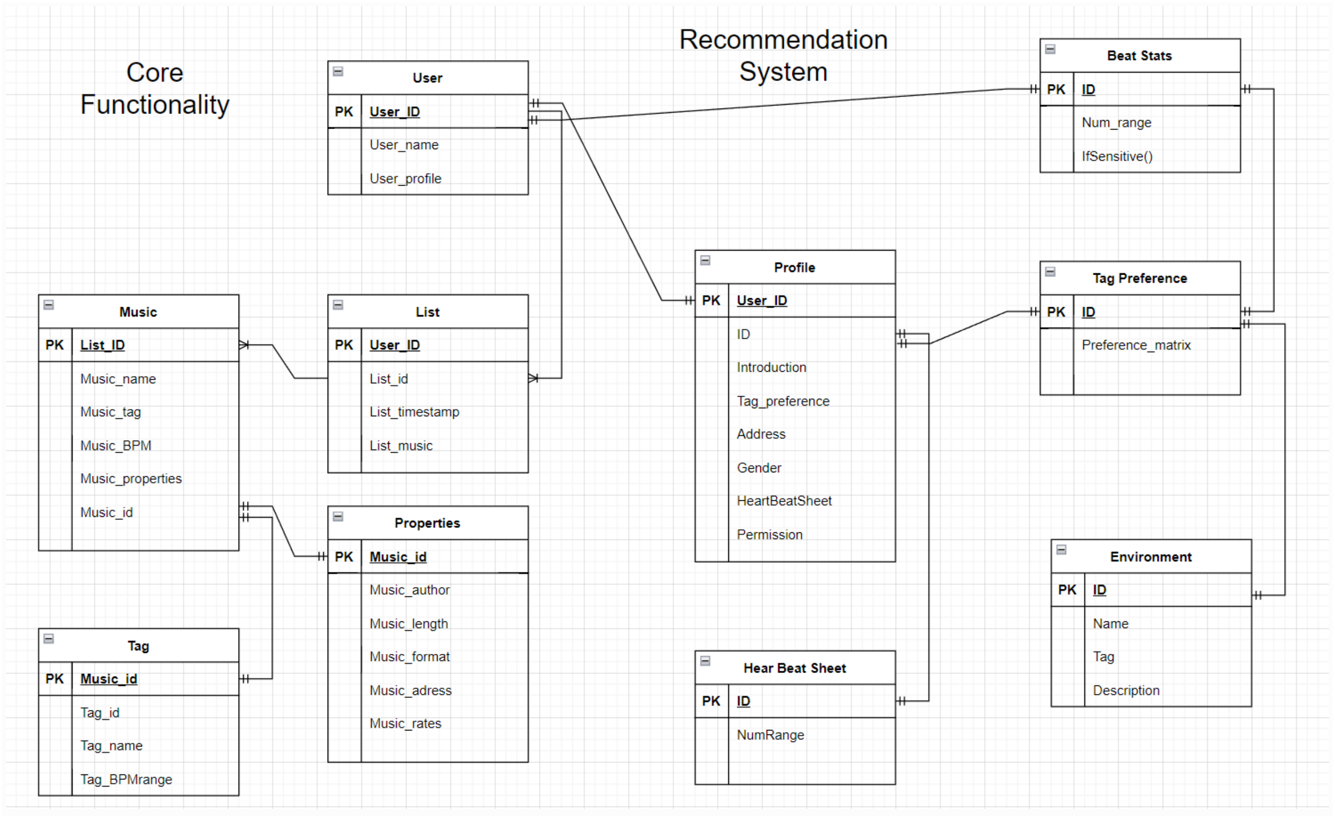
To start with, the designer should consider the timing and capacity of this system. To be more specific, the software will be available 24 hours per day with the Internet and Bluetooth connection. Data transformation should be limited into 1 seconds per song list regardless of the size. Any response time of the operations without data transformation though the Internet should be less than 2 seconds.

Next in line, the designer should also consider the ease of usage of the system. Users may be able to use all the system functions after at most 5 minutes self-exploration because of the convenient  function. Moreover, users can find the assisting instructions from the user interface at any time they want. The average search time of the autism children for the “help button” shall not exceed 2 seconds.

Thirdly, the designer should think about reliability and robustness. Once an error happens, the users shall be able to restart the system within at most 2 times of the time when the users start the system normally.

## 3.4 Logical Database Requirements

The database is deployed on a cloud server and can use any relational or non-relational database. The main classification of the database focuses on the BPM (Beat Per Minute), suitable scene and music. Moreover, the designer should also collect the relevant data and store it in the MySQL database. Here is the entity-relationship diagram:



## 3.5 Design Constraint

### 3.5.1 Standards Compliance

- The application can provide users with sufficient recommended music to enable users to find the best music for them. And, after the user has finished listening, the application should provide ratings to help the system better understand user preferences.
- When the application is modified and refined in the background, all changes are recorded in the database server and all changes are made in the database.

### 3.5.2 Android-based System

- The response time for loading the product should not exceed five minutes.
- Android devices must have wireless protocol support above Bluetooth 2.

## 3.6 Software System Attributes

### 3.6.1 Reliability

Software reliability includes specified time, specified environmental conditions, specified functionality, and the associated necessary software reliability tests. The music recommendation system is not sensitive, there are no restrictions or penalties for users temporary inaccessibility, even up to several weeks. The rate of a failure occurrence from the system server should not exceed 1 time per 50 requests from the users. The hardware of the music recommendation system is also stable and durable to be used in any environment.

### 3.6.2 Security

The music recommendation system requires the data interaction between the bracelet and the software through Bluetooth. Therefore, Bluetooth should use transport authentication and encryption technology. The authentication word is a 128-bit number used for system authentication. The length of the encrypted word is 8-128 bits, which can be changed to meet the upgrade of the algorithm and the encryption hardware system. On the other hand, the software basically should deploy on the Android platform so that the user should be asked to obtain the using permission. More specifically, the application permission contains Positioning, Turn Bluetooth On & Off , Read and write mobile phone storage, Media volume control and Get body movement information. Finally, the system will not leave any cookies containing any confidential and personal  information on the devices.

### 3.6.3 Maintainability

Maintainability is the basic feature of all software. This software will be implemented on some advanced platform to protect the maintainability, such as Visual Studio, Android Studio, etc. The use of standard presentation tools to describe algorithms, data structures, and interfaces in design documents can help maintainers better understand the software.

**3.6.4 Portability**

Since plenty of software development uses Javascript and Go programming language, the software will have the characteristics of cross-platform and high portability. For this software, not much needs to be done when porting to different system platforms, thus reducing the cost of software development and deployment. In order to make software highly portable, programmers need to abstract and modularize the application interface.

## 3.7 Organizing Specific Requirements

**3.7.1 System Mode**

This system will mainly follow the layered architecture. This model can be decomposed into structured programs of subtasks. Each subtask is located at a specific level of abstraction, while each level provides services to the upper level.

**3.7.2 User Class**

There is no specific user category for our application, and the user group is mainly online music listeners.

**3.7.3 Objects**

The main objects of our application are a series of songs, each with some attributes such as BPM, emotional tendency and audio link.

**3.7.4 Stimulus**

N/A

**3.7.5 Response**

This system will be able to return the song list to the bracelet when obtaining the real-time data from the devices. The software will match the suitable songs for users based on the AI recommendation algorithms.

**3.7.6 Functional Hierarchy**

The database server can only be accessed by internal personnel, who can add and modify information in the database at any time for easy maintenance and improvement of applications. The user only has access to the application.

# 4. Change Management Process

To effectively control the changes to the requirements, we will take the following third steps:

## **<u>4.1 Identify What Will Be Improved through group discussion:</u>**

In order to achieve the product easily and successfully, everyone needs to know what to improve, so a group discussion needs to clarify the modified object before the change. In this group discussion, the desired effect that can be achieved after the improvement should be clarified.

## **<u>4.2 Make a Plan for the Change:</u>**

After clarifying the modified object, a detailed and feasible plan is needed for the improvement. The modified plan needs to include the project's completion time and the material or data needed for this change.

## **<u>4.3 Review and Revise:</u>**

After the change, timely supervision is very necessary. The team leader needs to ensure that the changes are made as progress.

## 5. References

[1]Namitha, S.J. (2019). Music Recommendation System. INTERNATIONAL JOURNAL OF ENGINEERING RESEARCH & TECHNOLOGY. Volume 08, Issue 07

[2]Schedl M., Knees P., McFee B., Bogdanov D., Kaminskas M. (2015) Music Recommender Systems. Recommender Systems Handbook. Springer, Boston, MA. https://doi.org/10.1007/978-1-4899-7637-6_13

[3]Android Developers. 2022. Transfer Bluetooth data | Android Developers. [online] Available at:<https://developer.android.com/guide/topics/connectivity/bluetooth/transfer-data> [Accessed 27 February 2022]

[4]GPS & Data Collection » Epic Solutions. (2022). Retrieved 27 February 2022, from https://epicsolutions.us/striping-roadway-solutions/gps-data-collection/

# Appendix: Analysis Models Diagrams

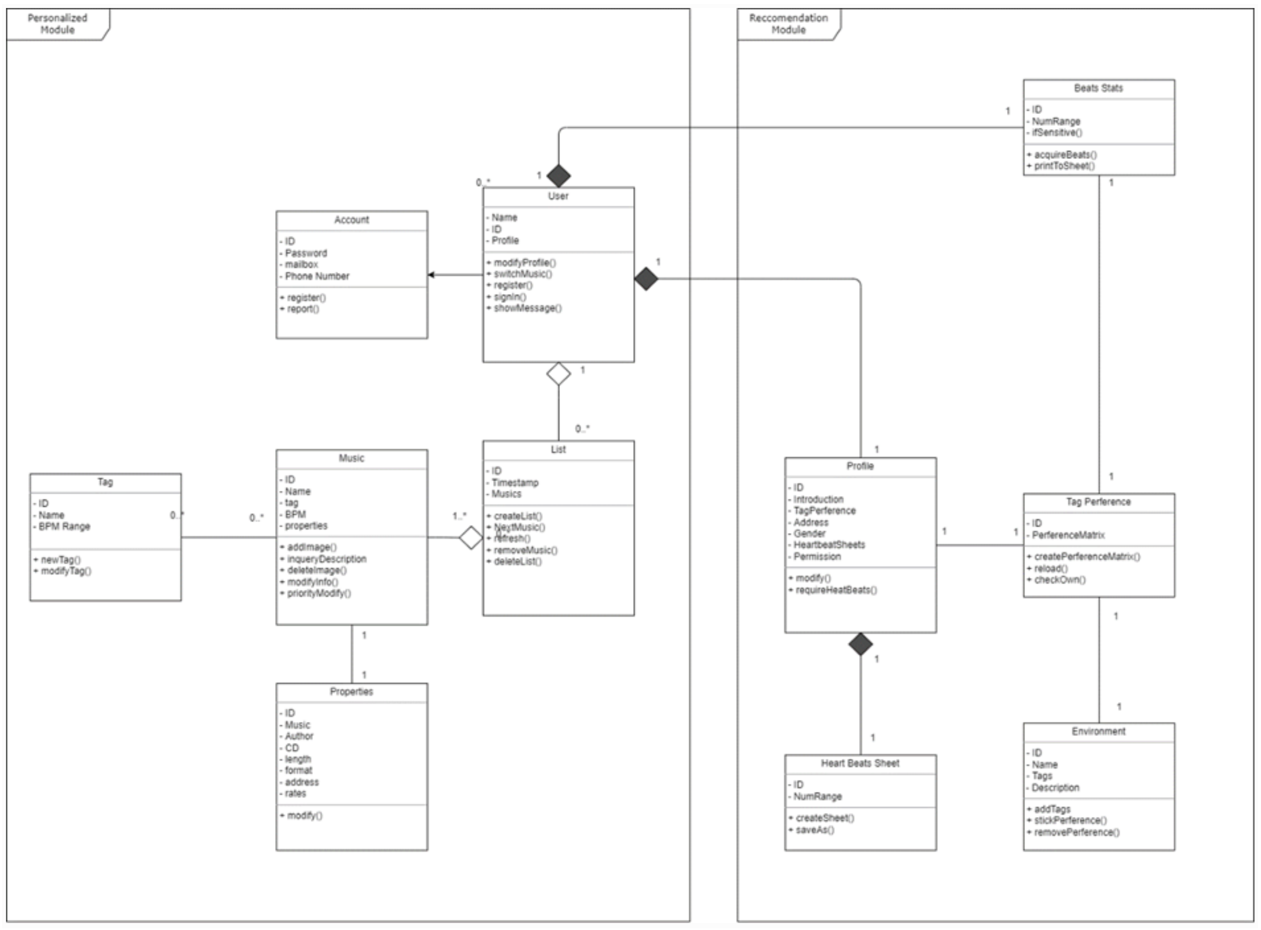


*Figure: Class Diagram*

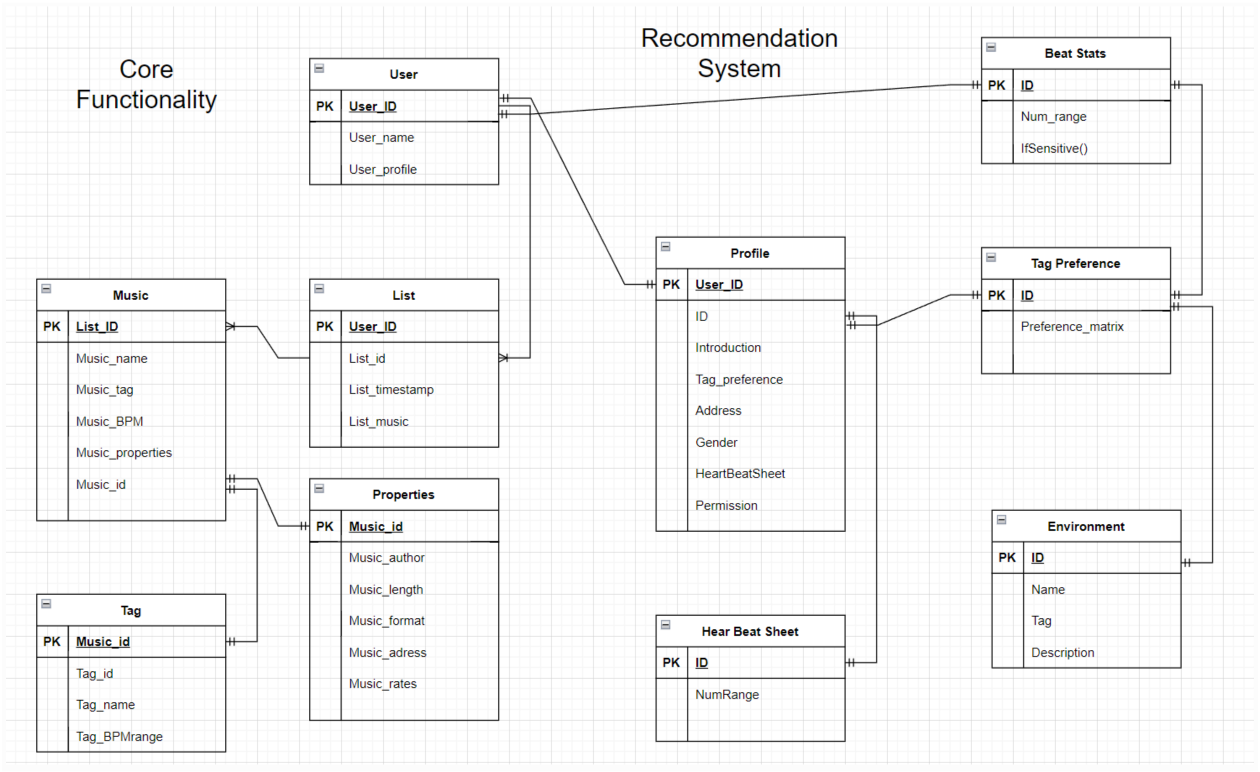


*Figure: Database ER Diagram*