Smart Barbell:
Software Design Description

Team SEnSE
석주영 (20190310)
김정민 (20200145)
김창희 (20200172)
김민성 (20220099)

Table of Contents

1. Overview

1.1 Scope

The scope of this software design description document covers the overall system design - including data design, architecture design, interface design, and procedural design. Also, different design concerns are discussed for multiple design viewpoints in this document. UML diagrams are included to provide more detailed and intuitive feature specifications of each design.

1.1.1 Product Quality

The main goal of the Smart Barbell System is to provide feedback on the workout of the users in an efficient manner. There are mainly two kinds of feedback; real-time feedback on the workout pace, and the workout progress feedback on the workout history of the user. By detecting the user's workout with sensors, we aim to give real-time feedback through music selection so that the user will keep motivated. With the data retrieved during the real-time feedback, the system would analyze the progress of the workout, visualize and show the user, and provide further recommendations to improve the workout.

1.1.2 Time

The final project report should be submitted on 12th June, 2022. Thus the project should be completed by 11th, June, 2022. The detailed timeline for the project is explained below.

- *2022.05.10: SDD ver.1* - MVC pattern selected for the architecture, and then created component-level design based on the use case models of SRS.
- *2022.05.21: SDD ver.2* - Contextual viewpoint described, and data classes added.
- *2022.05.31: SDD ver.3* - Information, Interaction viewpoint described, and component-level design transferred into class diagrams. Start development based on the architecture and classes.
- *2022.06.07: SDD ver.4* - Overview, Logical Viewpoint, Interface viewpoint described. Critical functional requirements implemented.
- *2022.06.11: SDD ver.5* - Criteria for evaluation described, and prototype testing conducted. Entire SDD reviewed.

The project aims to design the architecture and models for the key requirements from the SRS document. Among the initial set of requirements, until the project's due date, we aim to design all the "Critical" requirements, along with some additional features. Detailed explanation is described in [3.3 Functional Requirements]

1.1.3 Cost

There are 4 members for the project. Members of the project do not have to be paid, so the cost of the project is only the material cost.

The cost for the Smart Barbell Device is estimated to be about 50,000 KRW. For the Barbell Device, an Arduino board, a gyro sensor, and jumper cables. During the early development and testing, buzzers are used to replace the print statements and music selection.

1.2 Purpose

The purpose of this software design description document is to demonstrate the design of the whole Smart Barbell software system which will meet the key functional and nonfunctional requirements stated in the software requirement specification document of Smart Barbell. This document will provide initial guidelines for the developer team to successfully implement the software.

1.3 Intended Audience

The audience of this software design document includes all the stakeholders. Each part may have specific target stakeholders. The target audience for each part is described for section 4 to section 8, where it explains details of the system.

Throughout the whole document, the main audience are the development teams; the development team should understand all the elements of this document. Besides the developers, this document aims to provide understanding of the system to all the stakeholders described in [3.2 Stakeholders and Relationships].

2. Definitions, Acronyms, and Abbreviations

Following is the table consisting of terms used in this document and following definitions.

Term	Definition
Smart Barbell	Name of our software, or the entire system
SEnSE	Team Name: Software Engineering and Software Engineering
SRS	Software Requirements Specification
IoT	Internet of Things
Barbell Device	The IoT based barbell which could detect the motion of itself and transmits the motion data to the app server.
Workout	Activity of using the Barbell Device; lifting and lowering the barbell.
Workout User	Someone who interacts with the web page and does the workout using the Barbell Device.
Web page	A web application where the Workout User interacts with Smart Barbell and views the feedback on their workout.
Lift	A unit of the workout using the barbell. We call it a lift when the barbell rises from the lowest point to the highest point and then comes back to the lowest point.
top_timestamp	The time displacement from the start of the workout to the highest point.
bottom_timestamp	The time displacement from the start of the workout to the lowest point.
volume	The volume of workout; (weight) (number of repetitions)
frequency	The number of times you worked out for more than 30 minutes in the last week
MET	MET stands for Metabolic Equivalent of Task, which represents how much oxygen(converted to calorie) is consumed per 1kg, 1min.
API	Application Programming Interface
AWS	Amazon Web Services

DBMS	Database Management System
Flask	Python Web Framework
HTTP	HyperText Transfer Protocol
REST	Representational State Transfer
Arduino	A microcontroller board
MPU-6050	Six-Axis (Gyro + Accelerometer) MEMS MotionTracking™ Device
ESP-8266	SOC with integrated TCP/IP protocol stack that provides a microcontroller access to the WiFi network

3. Requirements and Use Case Models

3.1 General Overview

Smart Barbell is an attachable IoT device provided with a web interface which helps people to be more concentrated and motivated to their workout. The Smart Barbell provides feedback on the user's workout in two ways; by real-time music feedback, and by record analysis after the workout. First, the Smart Barbell automatically changes the music based on the user's workout pace, calculated by lifts per minute, through the data retrieved from the sensor on the Barbell Device. With the workout data stored in the database, Smart Barbell also provides feedback and recommendation by analyzing and visualizing the workout history.

The Barbell User can interact with the system in two main ways. One is to work out with the Barbell Device and receive real-time feedback during the workout. Here, the user can manage their pace to remain constant without being lowered. Another way is to check their workout status and prescriptions before and after the workout. Here, the user can easily track their workout progress and receive recommendations based on their workout data.

1) **Functional Requirements**

   From the user's view, the goals of using the Smart Barbell System are the following:
   - Get Real-Time feedback during the workout.
   - Check the workout history, and progress.
   - Get recommendations on workout routines.

   And in order to use the Smart Barbell System, all the users shall be signed up for the system web page. This leads to the 4 main features of the system:

   - Feature #1: User Account Management
   - Feature #2: Real-Time Page Management
   - Feature #3: Workout Record Visualization
   - Feature #4: Workout Record Analysis & Prescription

   These four main features are the basis for the functional requirements, and the composition of the system is also made accordingly. Detailed use cases, requirements are [explained in 3.3. Functional Requirements].

2) **Non-Functional Requirements**

   The two main components that make up the system are the Web Page, and the Smart Barbell. The non-functional requirements for the hardware and software of these two components are explained in [3.4. Non-Functional Requirements]. Also, since this system captures private health data, the security related requirements are also explained in 3.4 with other constraints.

3.2 Stakeholders and Relationships

Barbell Users

- **Individual Barbell Users**
  Individual barbell users are the customer and the end-users of the Smart Barbell System. They usually purchase the Barbell Device to attach it to their own barbells or other weight equipment for their personal use. Individual barbell users mostly use the Barbell Device on its own, without integrating it with other external gym systems.
- **Gym Barbell Users**
  Gym barbell users are the end-users who use the Smart Barbell System in a gym. They usually use the Barbell Device as a part of the smart gym system. They use barbells within their workout routine along with other equipment, rather on their own.

Gym Managers

- Gym Managers are the owners of the gyms. They are likely to have a lot of barbells in the gym. Gym Managers purchase the Barbell Device to support their customers to do their workout more efficiently and keep their motivations.
  Small-sized-gym managers purchase barbell devices directly to attach to their equipment. However, large scale gyms who try to apply a smart gym system may purchase the Barbell Device as a part of a smart gym.

Smart Gym Service Company

- Smart Gym Service Companies are the biggest customers of the Smart Barbell System. The companies design and develop smart gym systems for the Gym Managers. Many gym equipment are integrated into a single smart gym system, and the Barbell Device may be a part of it.

Software Engineer Team

- **Requirement Engineers**
  Requirement engineers wrote the SRS document for this Smart Barbell System. The requirement engineer team gathers, elicits, and validates the requirements. In the early stages they gathered new requirements to build the basis of the system. After the release, the requirement engineer team interacts with the above stakeholders to receive feedback and gather new requirements. Then the team goes through a screening process based on the SRS and this SDD document, and requests for modification.
- **Developers**
  In the early stages, developers implement new functions and develop the Smart Barbell System based on this SDD document. After the release, the development team keeps track of the system and updates minor bugs. When the system needs modification, the developers update the software.
- **System Administrator**
  After release, system administrators manage the saved data and storage of the system.

3.3 Functional Requirements

This section introduces a list of functional requirements. The requirements in the table below are the initial set of requirements from the Software Requirement Document. The functional requirements are then explained with the use case models for each system feature. The use case models for 4 subsystems cover all the functional requirements in the table.

In this ver.5 of this SDD document, all the “Critical” functional requirements are modeled along with some other priority functions. The highlighted rows indicate the modeled features until now. Other functional requirements will be added through further iterations.

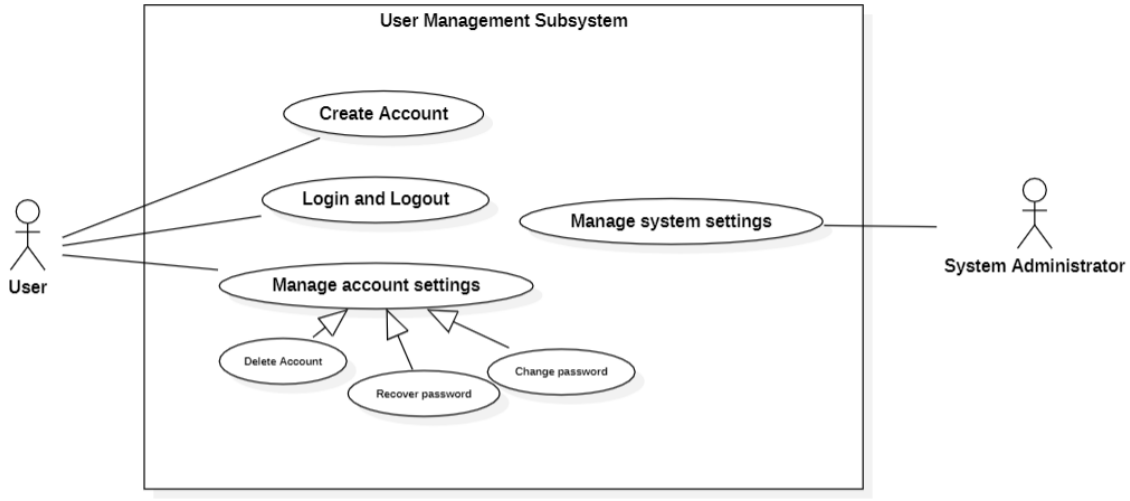
Code	Title	Requirement Description	Priority
FR-1	Create A New Account	A new work out user shall create a new account.	Critical
FR-2	Log In	A registered work out user shall get access to their account	Critical
FR-3	Manage Account Settings	A work out user shall change his/her account information, which is username and password. (Information such as weight, goal of exercising, etc can be added).	High
FR-4	Manage System Settings	A system administrator should access all user accounts in order to prevent malicious users or dormant accounts.	Low
FR-5	Barbell Connection	The Barbell Device shall start its Real-Time Pace Management System when the user connects to the Barbell Device.	Critical
FR-6	Set Configuration	A work out user shall set the weight and target pace before starting a set, and also shall be able to change these information during the workout.	Critical
FR-7	Set Modification	When the user changes the set information, the system shall modify the values into the new ones.	Critical
FR-8	Lift Detection	The Barbell Device shall detect the lifts using the acceleration values retrieved from the accelerometer sensor.	Critical
FR-9	Data Transmission from Barbell Device to Cloud	The Barbell Device shall send requests to the app server with top_timestamp and bottom_timestamp every time a lift is detected and the data shall instantly save to the DB with the user information.	Critical
FR-10	Pace Decision	The system shall be able to determine the Workout User’s workout status by comparing the current pace to the target pace.	Critical
FR-11	Music Selection	The system shall select a corresponding music when the workout pace changes.	Critical

FR-12	Barbell Disconnection	The user shall be able to disconnect from the Barbell Device after the workout and	High
FR-13	Display Table	The system shall generate and display a table of workout indicators according to the date range.	Medium
FR-14	Display Summary Graph in the Welcome Page	The system shall generate and display a line graph of workout indicators according to the date range.	High
FR-15	Display Pace	The system shall generate and display a pace log representing every lifts with its pace in sets and rest time.	Medium
FR-16	Retrieve Workout Records	The system shall query and retrieve the workout record with given range and filters.	High
FR-17	Share a Record to Others	The system shall share the specified record page utilizing platform-specific share methods.	Low
FR-18	Update Objective	A workout user shall select their objective between ‘gain muscle’, ‘lose weight’, and ‘maintain health’, and submit.	High
FR-19	Update Goal	A workout user shall enter a goal, related to their objective.	Medium
FR-20	Change Start Date	A workout user shall change the start date which will be used to calculate progress.	High
FR-21	Change Latest Result	A workout user sets a goal, related to their objective.The system shall check if the latest analysis result stored in Workout User DB is up-to-date.	High
FR-22	Analyze Workout Record	The system analyzes the workout record of the user. The result is composed of a welcome message, progress report, pace feedback, and workout prescription.	Critical
FR-23	Display Welcome Message	The system should display welcome messages on the main page, based on the analysis.	Medium
FR-24	Display Analysis Result	The system shall display analysis results, containing progress report, pace feedback, and workout prescription on the record analysis page.	Critical

3.3.1 User Account Management

In order to use the system, the Barbell Users shall be signed up for the system. The Barbell User can create an account to log in, and manage their accounts through the web page. The users can modify their account settings: User ID and Password. Since this system has user

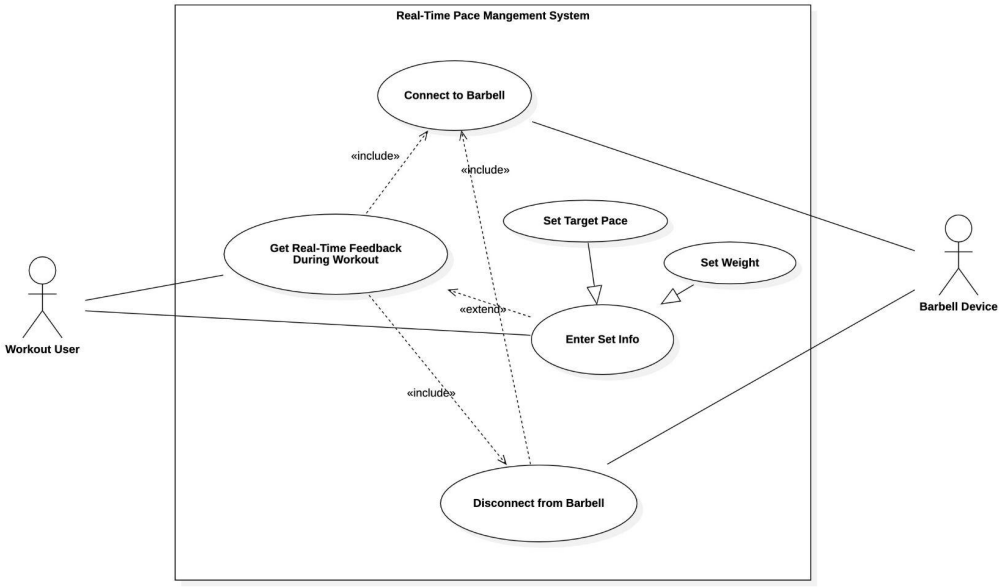
health information in the DB, the user can also delete the account. When they delete their account, the related workout record, account information, and the health information of the user are all deleted.



[Figure 3.1: Use Case Model #1. User Account Management]

3.3.2 Real-Time Pace Management

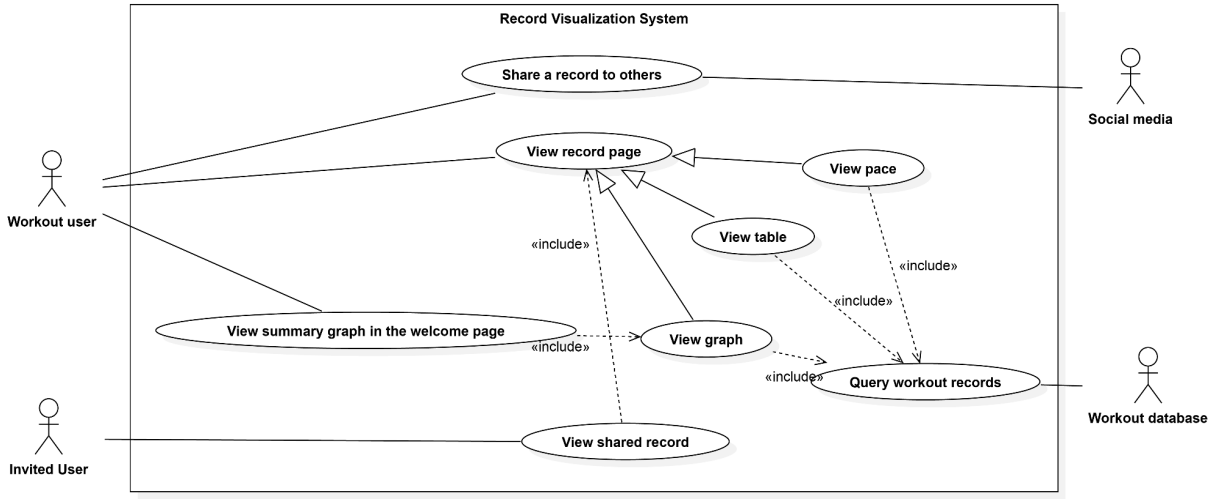
Before the workout, the Barbell User can connect to the Barbell Device to get real-time feedback on their workout. Then, after the user enters the weight of the barbell device and their target pace, the system keeps track of their workout to help them maintain their pace. During the workout, users can listen to music through the web page, and the system selects the right music for the user based on their workout pace or intensity. If the current pace is too slower than the target, the system will change to an exciting music.



[Figure 3.2: Use Case Model #2. Real-Time Pace Management]

3.3.3 Workout Record Visualization

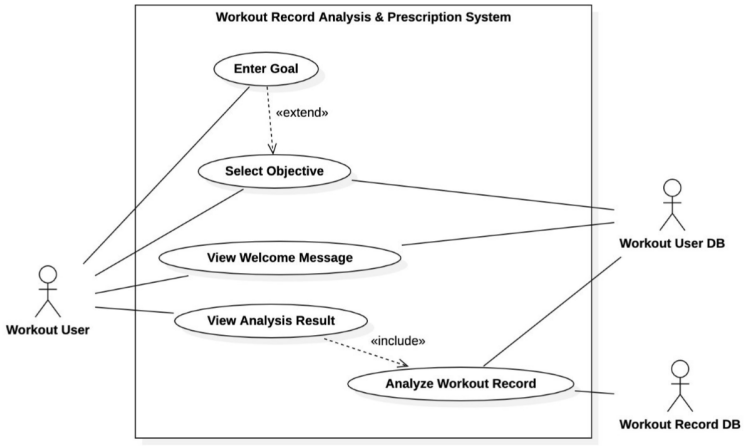
The Record Visualization subsystem is designed to utilize measured real-time data and process it into various forms for future queries to provide convenience to the users. This extends the opportunity of motivation through workout records to those who work out in a light manner.



[Figure 3.3: Use Case Model #3. Workout Record Visualization]

3.3.4 Workout Record Analysis & Prescription

Workout Record Analysis and Prescription System provides analysis based on the user's objective. The Workout User shall select objectives between ‘gain muscle’, ‘lose weight’, and ‘maintain health’. For each objective, there exists different goals that require the user to enter. The system will collect user’s Workout Record form Workout Record DB, and save analysis results at Workout User DB.



[Figure 3.4: Use Case Model #4. Workout Record Analysis & Prescription]

3.4 Non-Functional Requirements

Based on the logical non-functional requirements produced in the SRS, detailed implementation elements for the interfaces are described in this section. The below tables contain non-functional requirements for the system; hardware and software composition, quality attributes, and security.

3.4.1 Hardware

The main hardware of the Smart Barbell System is the Barbell Device. Since the Barbell Device shall detect the user's weight lifting based on the movement of the barbell, gyro sensors are used for detection. Also, since the Barbell Device shall be attachable to any weight equipment, it shall provide WiFi connection through WiFi modules.

Code	Requirement Description	Priority
HIR-1	**Barbell Device - Processor** The Barbell Device shall be composed of a processor, Arduino Nano.	Critical
HIR-2	**Barbell Device - Sensor** MPU-6050 shall be used for the Barbell Device sensor.	Critical
HIR-3	**Barbell Device - WiFi module** ESP-8266 shall be used for the connection between the Barbell Device and the Server.	Critical

3.4.2 Software

Code	Requirement Description	Priority
SIR-1	**Database Management System (DBMS)** MySQL 8.0 shall be used as the database management system.	Critical
SIR-2	**Operating System (Cloud OS)** The cloud server shall have linux (Ubuntu 20.04). IoT devices shall have no OS since they use simple microprocessors.	Critical
SIR-3	**AWS Server** The cloud server shall be driven by AWS server. This cloud server shall provide a web page to interact with users.	Critical
SIR-4	**Web Service** React.js, REST API and Django shall be used for web service.	Critical
SIR-5	**Data Transmission Format** IoT device software will send a message of 'processed real-time sensor value' to Cloud Server in below format. [(top_timestamp, bottom_timestamp), …]	Critical
SIR-6	**HTTP Protocol** For communication, http shall be used as a protocol with REST API.	Critical

SIR-7	**Barbell Device - Sensor** Adafruit MPU6050 library is used for the GyroSensor operations.	High

3.4.3 Quality Attributes

This section describes the criteria for assessment on the system, based on the 8 quality attributes; performance, features, reliability, conformance, durability, serviceability, aesthetics, and perceived quality.

Code	Requirement Description
QR-1	(Performance) The connection to the Barbell Device shall take less than 30 seconds when a Workout User already knows that they should scan the QR code for connection.
QR-2	(Performance) Whether the username is already taken or not, verification should take less than 10 seconds.
QR-3	(Performance) Matching Login information and the one in Work Out DB should take less than 10 seconds.
QR-4	(Performance) Record Analysis page should be loaded in less than 5 seconds when arrived.
QR-5	(Performance) Verifying if the user record and objective exist in DB should take less than 10 seconds.
QR-6	(Performance) Saving new account information while creating an account should take less than 20 seconds.
QR-7	(Performance) When a request to the app server fails, it shall try 2 more times before informing the user of the failure.
QR-8	(Reliability) Searched results for Workout User account in the Workout User DB during the Login procedure should be always right. When 100 searches are done, all 100 searches should return the right user information.
QR-9	(Reliability) Detection of a lift shall always be accurate. When 50 lifts are done, 48 or more lifts shall be detected with the data saved on the Workout Record DB.
QR-10	(Reliability) The recognition of the top point and the bottom point of the barbell shall not be delayed by 0.2 seconds. When testing the lift recognition with a beeper, after reaching the top point, the beeper shall beep before coming back to the bottom point.
QR-11	(Serviceability) More than two users shall be able to simultaneously connect to Smart Barbell.
QR-12	Web Interface should be accessible to all mobile phones, or tablets, despite their different display sizes.

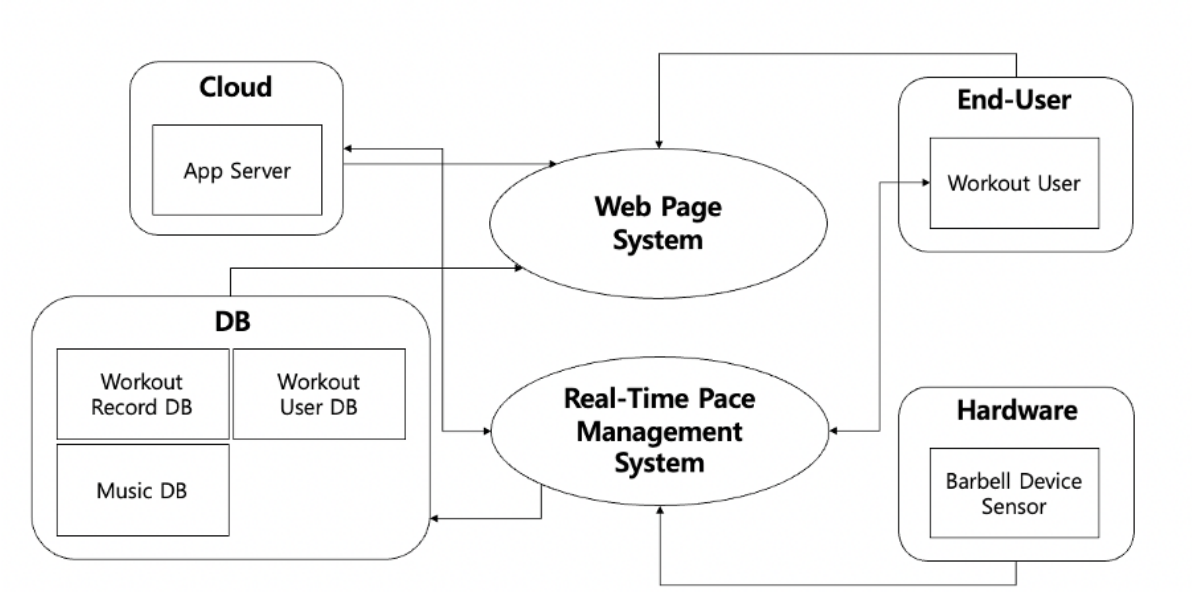
3.4.4 Security

The Smart Barbell System stores and utilizes not only the workout record, but also the user's health information such as age, weight, etc.. Since the system handles private information of the users, it is important to keep the database inaccessible to entities other than the System Administrators.

Code	Requirement Description
SR-1	Workout Data that is older than 1 year should be deleted from the DB.
SR-2	Only administrators should have access to other work out user accounts.
SR-3	Users shall be able to delete their record and health information whenever they want, through the web page. Then the data shall also be deleted from the database.

4. Contextual Viewpoint

[Figure 4.1] is a contextual overview of the whole Smart Barbell System. The Smart Barbell System can be divided into two major contexts based on the interaction of the system with the external entities.



[Figure 4.1: Overall relationship between the Smart Barbell System and the external entities]

First is the Real-Time Pace Management System, which provides real-time feedback on the user's workout by adjusting the music playlist according to the user's exercise pace. This system mainly consists of the Barbell Device's software which detects the user's workout, the workout feedback system, and the workout data storage system which stores the workout record in the database.

The second context is the web-page system, which is responsible for the user's interaction with the web-page. Here, the user can manage their accounts, check their workout record, and receive analysis or prescription on their workout routines.

4.1 Design Concerns

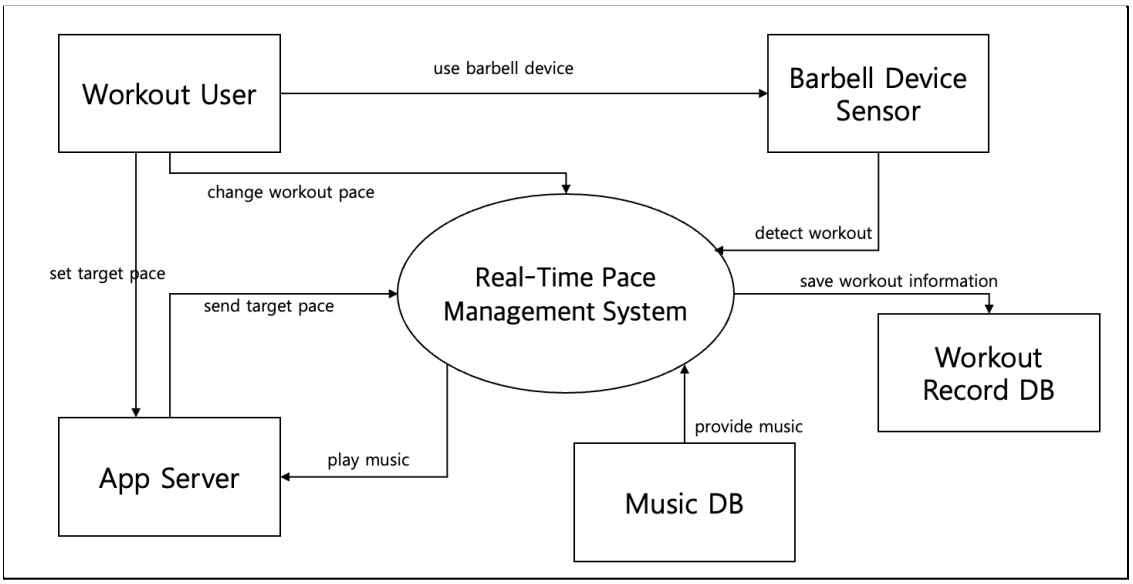
This section describes the context of the entire Smart Barbell System, so it covers all the functional requirements listed in [3.3 Functional Requirements]. As explained above, the system is divided into two contexts, and the related functional and non-functional requirements for each context are the following:

- **Real-Time Pace Management**: FR-5, FR-6, FR-7, FR-8, FR-9, FR-10, FR-11, FR-12, SIR-5, SIR-6
- **Web Page**: FR-1, FR-2, FR-3, FR-4, FR-13, FR-14, FR-15, FR-16, FR-17, FR-18, FR-19, FR-20, FR-21, FR-22, FR-23, FR-24, SIR-6

4.2 Design Elements

4.2.1 Real-Time Pace Management System

[Figure 4.2] is a context diagram of the Real-Time Pace Management System interacting with external entities. The external entities and the interactions are described below.



[Figure 4.2: Context Diagram of Real-Time Pace Management System]

1) External Entities

Workout User
A Workout User is the end-user of our system. The Workout User uses the Barbell Device to do the workout and is willing to get feedback on their workout data. The Workout User initiates the system by connecting to the Barbell Device by web, and setting environments for the feedback. Then they receive real-time feedback from the system and disconnect from the system after the workout is done.

Smart Barbell Sensor
Smart Barbell is the main device for the Real-Time Pace Management System. The Barbell Device consists of an Arduino board with a gyro sensor, which is considered as an external entity. The Barbell Device software receives the workout status in real-time through the Barbell Device sensor in it. Then the Barbell Device software, or the Real-Time Pace Management System processes the data before sending it to the App Server.

App Server
App Server is where the web-page is handled. Here, the Workout User can set their target pace, and deliver it to the system. Also, this is where the Workout User can listen to the music selected by the system.

Workout Record DB
Workout Record DB is a database where the workout information will be saved in. Every time the Real-Time Pace Management System detects a lift from the Barbell Device, it will save the Workout information in the Workout DB after processing. The form of the Workout DB is explained in [Figure 7.1].

Music DB
Music DB is where different kinds of songs are saved with their corresponding labels. The songs inside this DB are labeled in either calm, neutral, or exciting. The system selects and plays the music within this Music DB in the early stages. However, we plan to expand our system to collaborate with music streaming applications later, so that music streaming can be held in external applications while using our system.

2) Attributes

The main feature of the Real-Time Pace Management System is to give real-time feedback on the workout pace of the users based on the data retrieved from the accelerometer sensor of the Barbell Device. If the Workout User's workout pace is slower than the target pace, it notifies the user to keep the pace up.
As the Barbell Device detects each lift of the barbell, it stores the workout record into the Workout Record DB.

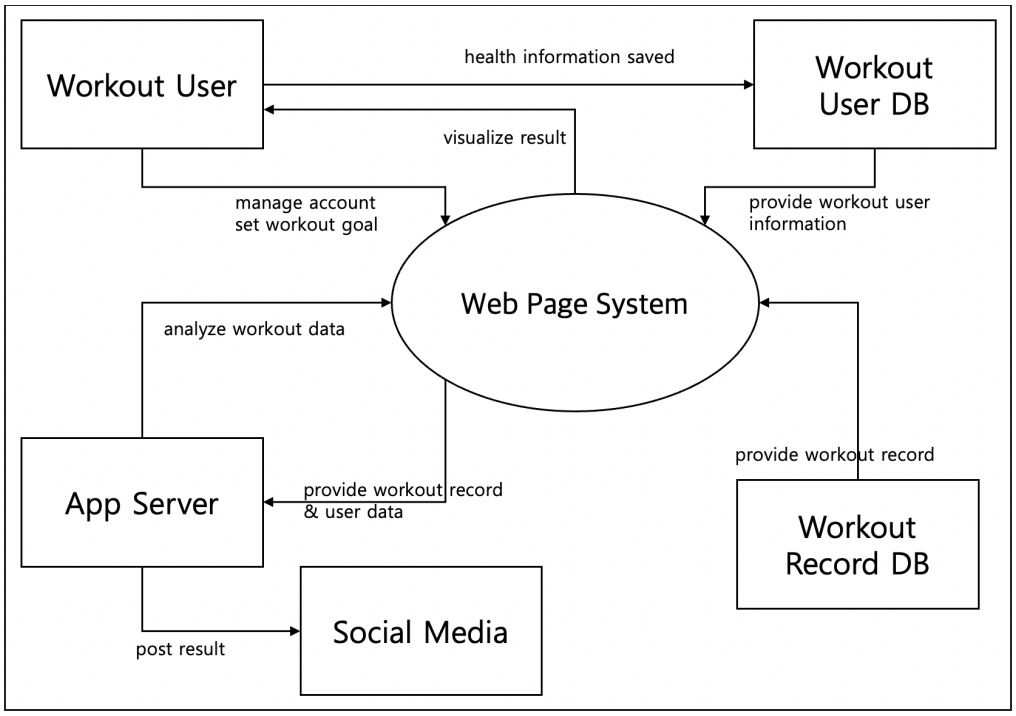
3) Relationships

The relationships between entities are described in [Figure 4.2].

- The Workout User sets their target pace in the web-page through the App Server.
- The system receives the target pace from the App Server.
- The Workout User uses the Barbell Device to do their workout and get feedback.
- The sensor on the Barbell Device detects the workout of the Workout User and saves the workout information in the Workout Record DB.
- Throughout the workout, the workout pace of the user may change, which triggers the system to respond to it.
- From the workout data detected by the Barbell Device and the target pace, the system decides the workout state of the user.
- In reference to the workout state, the system selects music from the Music DB and plays it through the web-page in the App Server.

4.2.2 Web-Page System

[Figure 4.3] is a context diagram of the web page interacting with external entities. The external entities and the interactions are described below.



[Figure 4.3: Context Diagram of Web-Page System]

1) External Entities

Workout User
A Workout User is the end-user of the system. Through the web-page on the App Server, the Workout User can sign up for the system, view their workout record, and receive feedback or prescription on their progress.

App Server
App Server is where the web-page is handled and displayed to the Workout Users. Here, the Workout User can manage their account, view their workout records, and get recommendations on their workout progress.

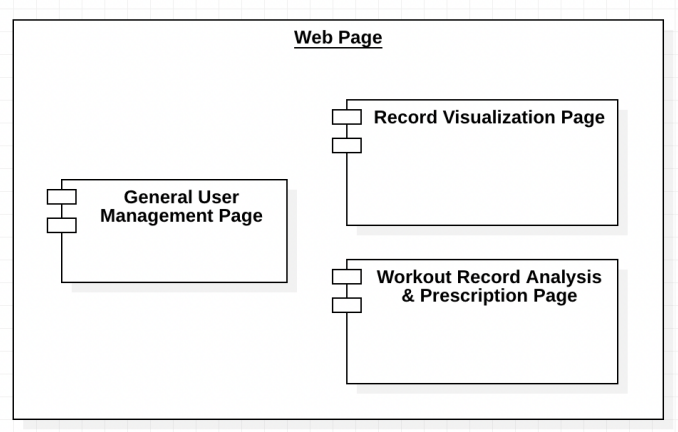
Workout Record DB
Workout Record DB is a database where the workout information is saved in. When the web page requires the workout record of a Workout User, the App Server can receive the information of the workout history from the Workout Record DB. The Workout Record DB is updated frequently as the user exercises. Detailed form of the Workout Record DB is described in [Figure 7.1].

Workout User DB
Workout User DB is a database where the static information of each Workout User is saved in. When the user first signs up for the system, a new set of Workout User data is stored in the Workout User DB, including user ID, password, and health information such

as age, height, weight, etc. The users can also modify this information through this system. Detailed form of the Workout User DB is described in [Figure 7.1]

2) Attributes

The Web-Page consists of three main pages which correspond to the three main attributes of the Web Page System. According to these pages, the Web Page System is divided into three subsystems; User Account Management System, Record Visualization System, and Workout Record Analysis & Prescription System.



[Figure 4.4: Main components of the Web Page System]

User Account Management
The user can create an account, login, change their ID/PW, and enter their health information. This information is saved in the Workout User DB.

Record Visualization
The user can view their workout record and history in the web-page. The system retrieves data from the Workout Record DB and visualizes the workout history with visual methods such as graphs and tables.

Workout Record Analysis & Prescription
Based on the health information of the user, the system analyzes the workout data in the Workout Record DB. The user can set their long-term workout goal, and the system analyzes the workout progress or gives recommendations based on the goal.

3) Relationships

The relationships between entities are described in [Figure 4.3].

- To use the Smart Barbell System, the Workout User should create an account for the web-page through the Web Page System. As an account is created, the user information is saved in the Workout User DB.

- The Workout User can modify and manage their information through the web-page, and it will modify the corresponding information in the Workout User DB.
- After the workout, the Workout User can log into the web page and request to visualize the workout history.
- The system queries the workout data from Workout User DB and processes them on the app server.
- Through the web page on the app server, the visualization results can be shared through the social media of the account user.

4.3 Criteria for Evaluation

In the contextual viewpoint, the system is divided into two contexts based on the interaction between the user and the entities. Even when a new feature or a function is added during the iteration, no more entities should be added in the context diagram. The details can be added, but the context itself should not become more complex than it is now.

4.4 Viewpoint Source

This contextual viewpoint provides an overview of the Smart Barbell System from a broad perspective, focusing on the relationship with external entities. The two contexts of the system are divided mainly based on the user's interaction with the system, but the models also include system-level entities involved in each context. All the stakeholders, especially the development team shall be able to have a whole understanding of the system through this chapter.

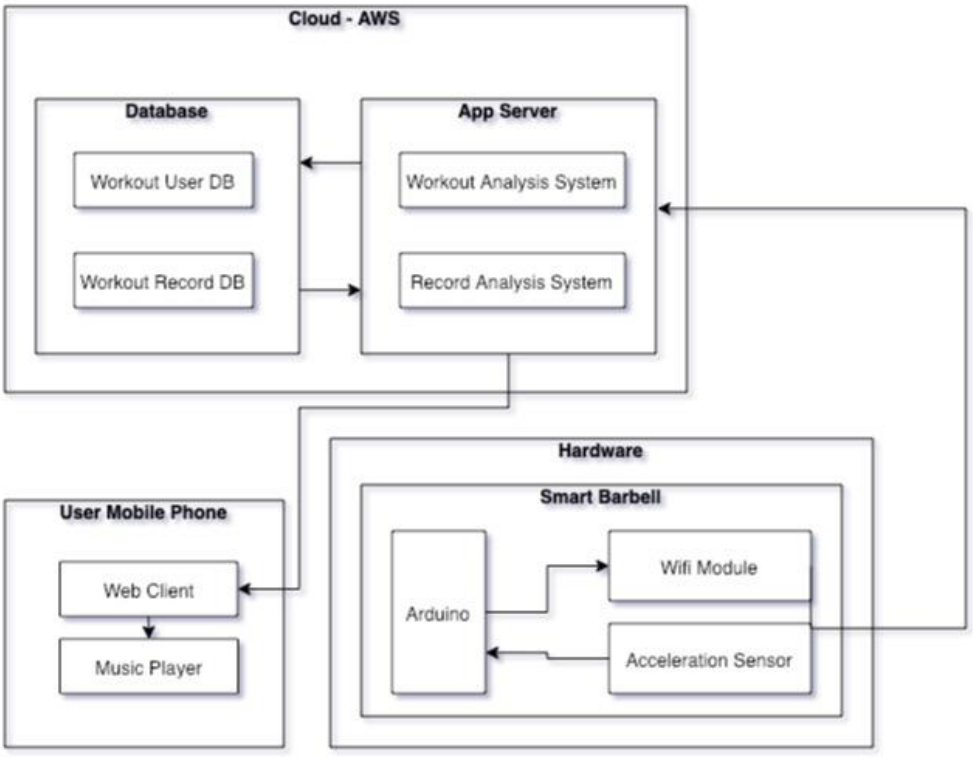
4.5 Design Rationale

The context of the Smart Barbell system is divided into two based on the Barbell Device, which is the main hardware device of our system. There are two main behaviors that our users can do. One is to get real-time analysis of one's exercise through interaction with this device, and the other is to check the analysis of one's exercise on the web-page. Therefore, the context of the system was largely divided into two depending on whether or not this main device was included. Barbell Device Sensor is included as externality in the first case and not in the second case. Therefore, in the contextual viewpoint looking at the relationship with externality, it was divided into two according to the type of externality that interaction is performed.

5. Patterns Viewpoint

Smart Barbell is an IoT software, with the utilization of Cloud computing. There are multiple components and relationships between them that build up the system, and these are called software architecture. In this section, the software architecture pattern of Smart Barbell will be discussed and represented from various perspectives. Note that Krutchen 4+1 View Model was used as a reference to represent our architecture.

5.1 Logical View



[Figure 5.1: Logical View - entities of the system]

[Figure 5.1] shows the key entities that compose the whole Smart Barbell system – Cloud, Hardware (Smart Barbell Device), and User Mobile Phone. Cloud contains database and app server. There are two databases, Workout User DB and Workout Record DB. They store information of users, and workout records of every lift, respectively. App server has four main features and functional requirements that belong to each feature are described in the Software Requirement Specification. Barbell Device uses Arduino for processor, Wi-Fi module to connect internet, and acceleration sensor to detect workout. Users will access our web client by their mobile phone, and our service will open the music player of their phone to play suggested music.

5.2 Implementation View

5.2.1 Design Concerns

According to the Software Requirement Specification document of Smart Barbell, there are four main functionalities: User Account Management System, Real-Time Workout Pace Management System, Record Visualization System, and Workout Record Analysis & Prescription system. However, the most crucial and complicated part of the Smart Barbell is the Real-Time Workout Pace Management System, as it is the distinguished functionality that Smart Barbell serves compared to other workout-related softwares. Also, in terms of complexity, IoT devices, Cloud, and Client interaction are all involved and the functionality should satisfy strict reliability and availability requirements stated in QR-10, QR-11 in [3.4.3]. In the aspect of data, multiple types of data entities flow through the whole software, so careful management is required.

Based on the overall functional requirements and non-functional requirements, we tried to focus on the following two concepts, **efficiency** and **modularity**.

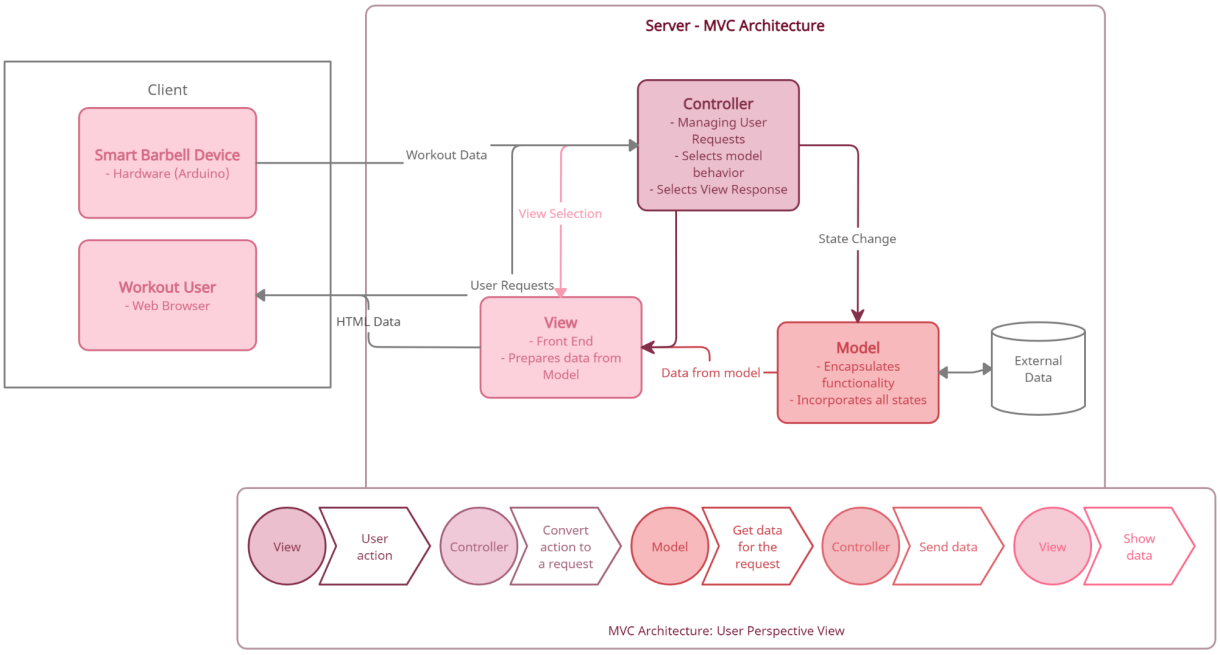
- **Efficiency**
  According to QR-10, QR-11, Smart Barbell should be able to serve multiple clients simultaneously, with high response time. Therefore, efficiency is one of the crucial goals that the Smart Barbell system has to achieve.

- **Modularity**
  Smart Barbell is an Cloud-based IoT Software, involving multiple components. Also, there are numerous data entities that flow through the whole system. In order to handle each concern with ease, modularizing each part is necessary.

In the aspect of technology, developers in Smart Barbell are friendly to Python, especially Django for developing applications. The fact that Django uses MTV design pattern, a design pattern that is a modified version of MVC design pattern, was also considered in the process of architecture decision.

5.2.2 Overall Architecture

Overall architecture has the following structure([Figure 5.2]). Generally, it follows the client-server pattern, and server specifically is designed as an MVC pattern.



[Figure 5.2: Overall architecture of the system]

To benefit in efficiency, the client-server pattern is selected so that it can serve multiple clients at a time. Within the server, based on the structure of the AWS server, described in SIR-3.

5.2.3 Design Rationale

The architecture of Smart Barbell uses a combination of MVC design pattern and client-server pattern. Some properties of MVC pattern is that it isolates business logic from user interface, and makes it easier to modify applications either their visual appearance or the underlying business rules without affecting others. Client-server pattern, on the other hand, has the server component listening to client requests, and providing services to multiple client components.

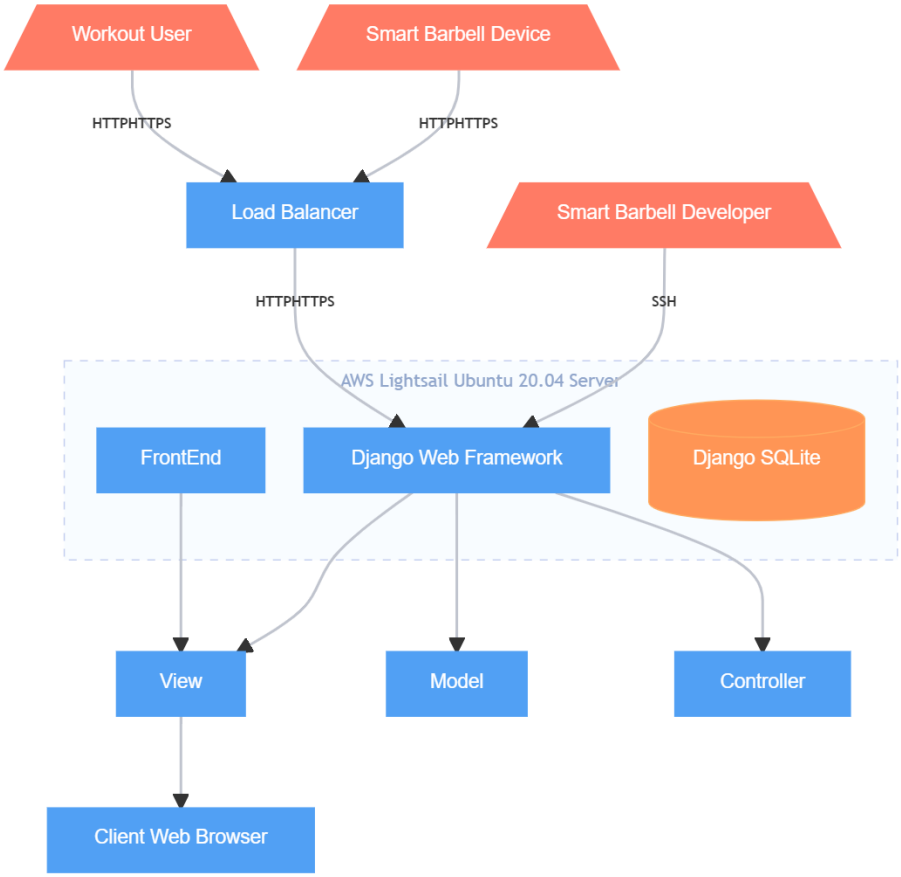
Based on the two concepts that were introduced previously at [5.2.1], the connection between the concept and design pattern can be the following.

- **Efficiency - Client-Server design pattern**
  Client-server pattern is known to be scalable and effective in serving numerous clients, as there are some modern techniques that handle the scalability issue in client-server architecture: load-balancing, sharding, and partitioning.

- **Modularity - MVC pattern**

  In MVC pattern, changes in a certain section i.e. Model, View, Controller, of the application will never affect the entire architecture. This is an indicator that modularization can be achieved using MVC pattern, also allowing fast and incremental development.

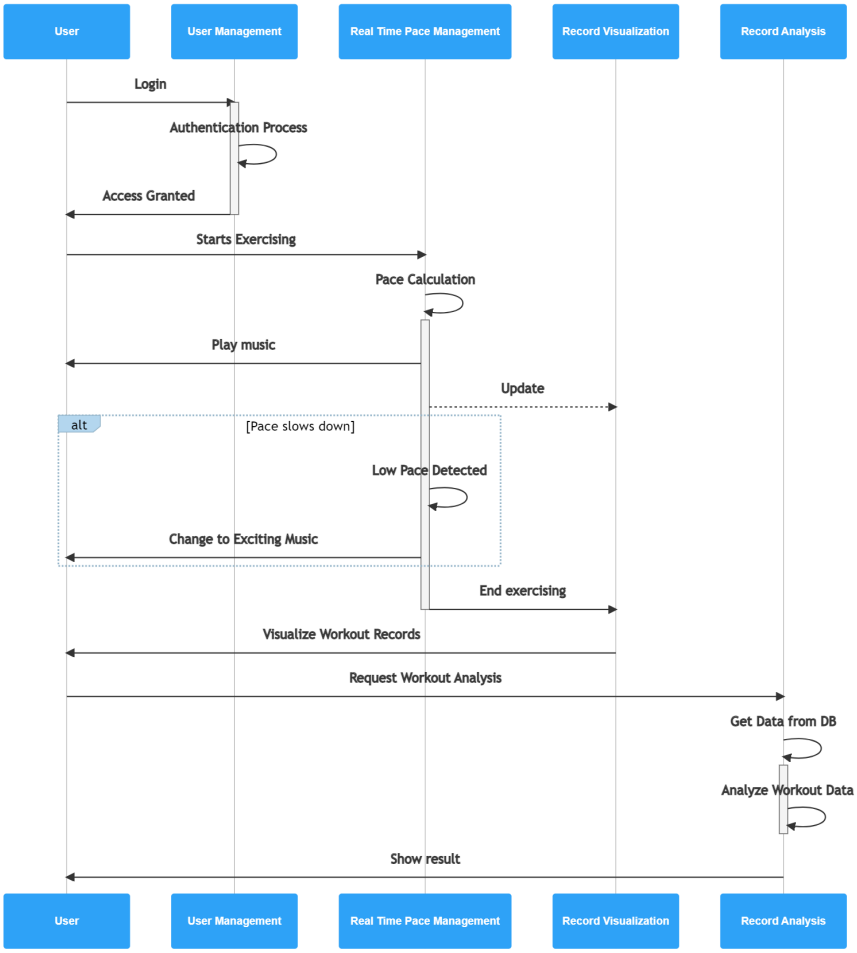
5.3 Deployment View



[Figure 5.3: Deployment view of the system]

The Smart Barbell software will be deployed and served using AWS lightsail server, for which operating system is Linux (Ubuntu 20.04), with x86-64 bit architecture. IoT devices shall have no OS since they use simple microprocessors. Django, React.js with REST API shall be used for web service. Note that there is no external DB as Django provides SQLite internally. The website will be published using Github Pages, with a domain .github.io. Finally, HTTP connection between IoT devices and the cloud server will be activated when the data is transmitted.

5.4 Process View



[Figure 5.4: Sequence Diagram for the Smart Barbell System]

Upper diagram is a simplified sequence diagram that shows the flow of Smart Barbell at a high-level point of view. We expect workout users to utilize subsystems in the following order: User Management - Real-Time Pace Management - Record Visualization - Record Analysis, and the whole software is developed with this order in mind. Subsystems that are located aside in terms of use order have more connections to each other. Detailed procedures for each use case are described at [9. Interaction Viewpoint].

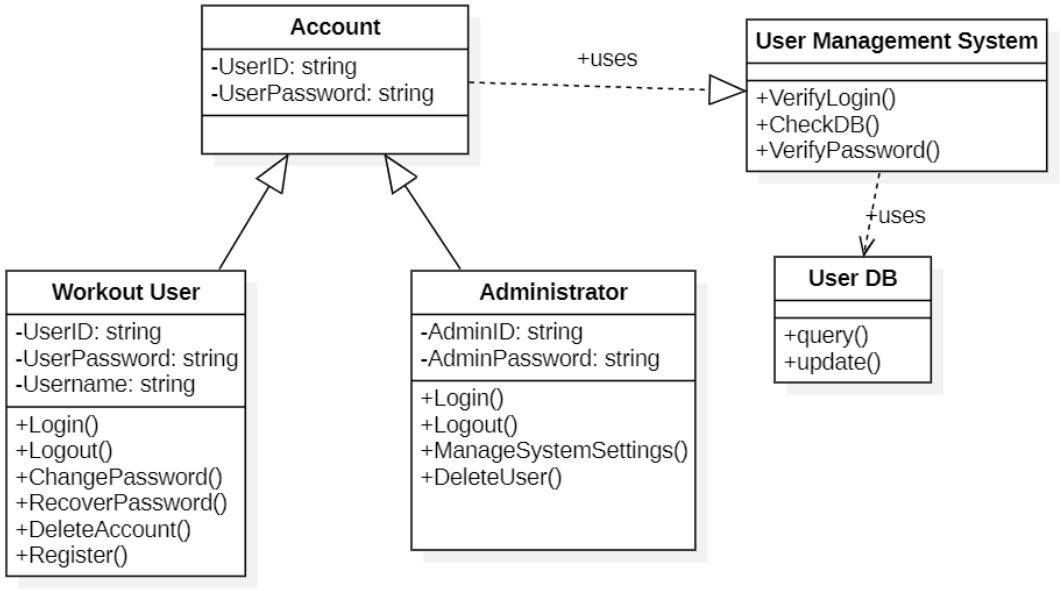
6. Logical Viewpoint

6.1 User Management System

6.1.1 Design Concerns

User management system is in charge of giving authority to registered workout users to start using the Smart Barbell software. Related functional requirements are FR-1, FR-2, FR-3, FR-4.

6.1.2 Design Elements



[Figure 6.1: Class Diagram of User Management System]

- **Workout User**

  The Workout User class holds information of a registered and non-registered (but about to register) user, which is UserID, UserPassword, and Username. It also has the main functions a user can do, for example Login, Logout, ChangePassword, and etc.
- **Administrator**

  Administrator class holds information of an administrator, such as AdminID and AdminPassword. It includes Login and Logout that Workout User also has. Moreover, it has the critical functions that only administrators can do: ManageSystemSettings and DeleteUser.
- **Account**

  Account is a class that contains Workout User and Administrator. It is an abstraction indicating people who use the Smart Barbell.
- **User Management System**

User Management System class is a class that contains functions that are used in the process of verification of account information.

- **User Database (i.e. User DB)**

  User database is a class where account information is saved. Query is used in the login process, to check whether the UserID, UserPassword pair is valid, and Update is used whenever there is a change in password or new user is registered.

6.1.3 Criteria for Evaluation

The main goal of the User Management System is to give authority to registered workout users to start using the Smart Barbell software. Evaluating the correctness is directly associated with QR-8.

6.1.4 Design Rationale

User management system of Smart Barbell is a common function that exists in other softwares but also is one of the critical parts of the whole system. We tried to follow the principles of login systems yet emphasizing that the user account information is secured. That is why we keep a separate database for users.

6.2 Real-Time Pace Management System

6.2.1 Design Concerns

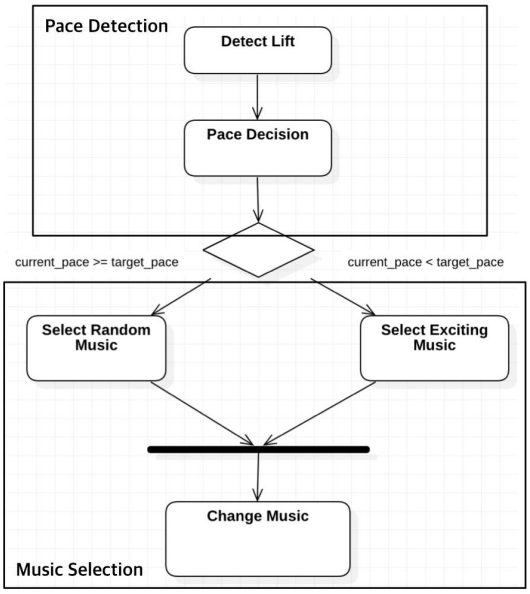
Among the contexts of the system explained in [5. Contextual Viewpoint], the Real-Time Pace Management System is responsible for providing the real-time feedback, as explained in [5.2.1 Real-Time Pace Management System]. The related requirements are listed below.

- **Pace Recognition**: FR-5, FR-6, FR-7, FR-8, FR-9, FR-10, FR-12
- **Music Selection**: FR-11

6.2.2 Design Elements

1) Overall Procedural Logic

[Figure 6.2] is a state chart diagram of an overall flow of the Real-Time Pace Management System. First, it detects a single weight lift and measures the time, or pace, for the lift(FR-8). Then, the system compares the current pace to the target workout pace(FR-10) and selects an appropriate music depending on the result of comparison(FR-11).



[Figure 6.2: State Chart Diagram of the Real-Time Pace Management using music change]

The Real-Time Pace Management System can largely be divided into two parts; the pace recognition which is held in the Barbell Device to the cloud app server, and the music selection which is held on the app server. The pace recognition logic detects the pace of the Barbell User and decides if the pace is satisfactory. Based on the decision made during the pace recognition, the next song is selected through the music selection logic.

For the pace detection, the Barbell Device detects each lift by using pitch as the criteria for detecting the lowest and highest point. Currently, the range of pitch (-45 ~ -65) is decided by default through iterations of tests, however in further iterations, the system will have an adjustment function to adjust the barbell detection for each user's postures.

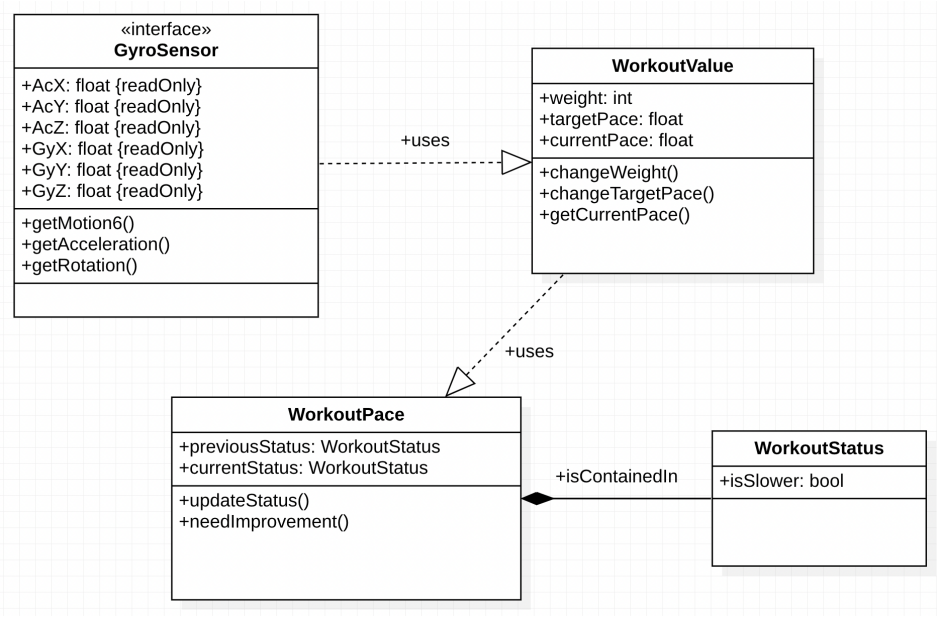
Algorithm #1: Barbell Device - DetectLift

```
1   while True do
2       pitch = calculate pitch from from GyroSesor values: gyroGet
3       if pitch > -45 then
4           bottom_timestamp = time for the lowest point
5           while True do
6               calculate roll, pitch from GyroSensor values: gyroGet
7               if pitch < -65 then
8                   top_timestamp = time for highest point
9                   return top_timestamp - bottom_timestamp
```

[Figure 6.3 Algorithm for Lift Detection in the Barbell Device]

2) Pace Detection

[Figure 6.4] is a class diagram for the pace detection. The GyroSensor interface is an Arduino library for MPU-6050 sensor(SIR-7).



[Figure 6.4: Class Diagram for Pace Detection]

- **GyroSensor**

  GyroSensor is a class of type: *MPU6050.* Only the used methods and attributes of the class MPU6050 are explained in [Figure 6.3]. The methods of the MPU6050 library are used in detecting the lift of a Barbell Device(FR-8) eventually to decide the workout pace(FR-9).

- **WorkoutValue**

  WorkoutValue class contains measures for the workout; weight, targetPace, and currentPace. When a user configures or modifies the set information of the workout(FR-6, FR-7), the changeWeight(), changeTargetPace(), and getCurrentPace() operations change the values of the attributes.

- **WorkoutPace**

  WorkoutPace class contains the WorkoutStatus information of currentStatus. Every time a lift is detected(FR-8), the status of current workout is decided(FR-9) and previousStatus, currentStatus is updated with the updateStatus() operation.

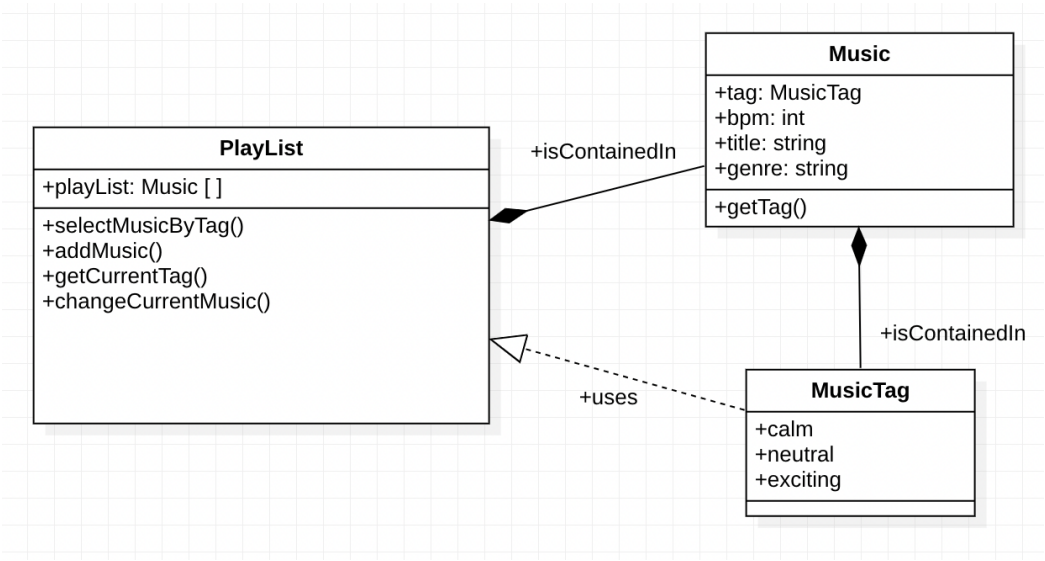
  Currently, as explained in [Figure 6.4], the workout pace is decided only by comparing it to the target pace. However, in further iterations the analysis of real-time workout pace will be improved to compare more factors.

- **WorkoutStatus**

  WorkoutStatus is a class that includes different kinds of workout status. Based on these workout status, the WorkoutPace is decided. Currently, the only measure of workout status is to compare current pace to the target pace. In further iterations, different status measures will be implemented in this workout status class.

3) Music Selection

[Figure 6.2.4] is a class diagram for the Music Selection. The GyroSensor interface is an Arduino library for MPU-6050 sensor(SIR-7).



[Figure 6.2.4: Class Diagram for Music Selection]

- **Music**
  In the current iteration level, all the songs, or music, are stored in the Music DB as explained in the contextual viewpoint ([4.2.1 Real-Time Pace Management]). Each music has tags based on the bpm and genre.
- **MusicTag**
  Each Music in the Playlist has a tag of they MusicTag. Currently, the tags are generated according to the bpm of the music. The songs of bpm under 70 are tagged as ‘calm’, those in the range 70~120 are tagged as ‘neutral’, and those over 120 are tagged as ‘exciting’. The MusicTag is used for selecting an appropriate music for feedback,
- **PlayList**
  PlayList is the class that contains all the Songs. When the system plays the music, it selects the music within the PlayList. The system can search for a specific music with a specific MusicTag.

6.2.3 Criteria for Evaluation

The classes of the system are designed in a way that is compatible with further iterations. Through iterations, the system is expected to improve by adding more features and attributes. Thus, the design should try to maintain the current structure as they go through iterations. However, when a new class should be added, there should be no coupling over the parts. (e.g.

When a new class is added to [Figure 6.2.4], there shall not be coupling with the classes in [Figure 6.4].)
Also, every time a new class is added, it should go through the equivalence class partitioning.

6.2.4 Design Rationale

- **WorkoutStatus**

At this moment, the WorkoutStatus only has the ‘isSlower’ attribute. In this state, ‘previousStatus’ and ‘currentStatus’ in the ‘WorkoutPace’ class can be specified immediately, but the ‘WorkoutStaus’ class is built regarding further iterations. In the next version of the software, in addition to comparing the currentPace to targetPace, new measures will be added for deciding the WorkoutStatus. Thus, there should exist some space(class) for more attributes and operations to be added.

- **MusicTag**

In [Figure 6.2.5] the songs are stored in the PlayList class, and the system selects a song within the PlayList. Currently, the number of songs are small and the songs are stored in the database. Since the WorkoutStatus is only either ‘isSlower’ or not, the tags of the musics are also limited to ‘calm’, ‘neutral’, and ‘exciting’. However in future iterations, we aim to integrate the music selection system with external music streaming apps. Then, the system will look up songs in the streaming app with improved methods of calculating tags. Therefore, for further improvement, the system keeps the MusicTag class separately for calculating the tags.

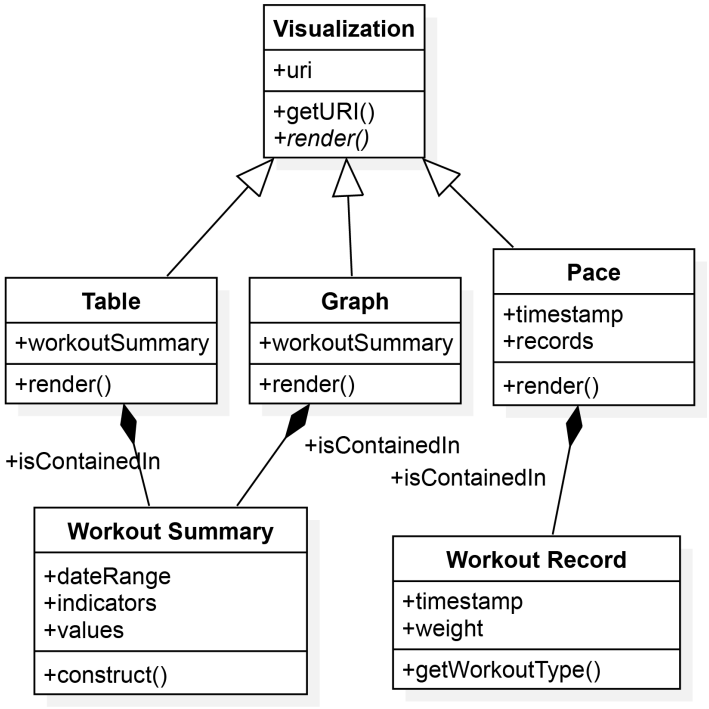
6.3 Record Visualization

6.3.1 Design Concerns

Considered requirements are as follows.

- FR-13, FR-14, FR-15, FR-16, FR-17

6.3.2 Design Elements



[Figure 6.5: Class Diagram for Record Visualization]

- **Visualization**

  Visualization has an attribute and an operation holding URI of the visualization, which is used to share it to other users. The class has an abstract method indicating there should exists individual rendering method for every visualization class.
- **Table & Graph**

  Table and Graph has an attribute workoutSummary, and an operation render to render the actual visualization with its own method. Every time when Workout User changes the visualization filter including indicator and date range, new table or graph has to be constructed and re-rendered.
- **Pace**

  Pace is a class representing a visualization of records within a workout. A workout and corresponding Pace is identified by its starting timestamp. The render operation generates a frame showing break times and lift-ups during sets.
- **Workout Summary**

  Workout Summary acts as a summary of multiple workout records distributed in specific date range. It has a list of values; each representing a value for specified indicator, and can be constructed with visualization filter.
- **Workout Record**

  Workout Record acts as a small and simple unit of record, representing one lift-up. Detailed description can be found in the Workout Record section of [7.2 Design Elements].

6.3.3 Criteria for Evaluation

The goal of Record Visualization System is to visualize historical records measured in past so Workout User can notice the progress of development in a glimpse. Therefore, no matter which input is given, which records were retrieved from the database, readability should be maintained.

6.3.4 Design Rationale

- **Visualization**

Class Visualization works as a base class for all kinds of visualizations: Table, Graph, Pace. An URI referring to a visualization is specified when Workout User explicitly share it. This URI is then delivered to platform specific social media service, regardless of the type of visualization.

Render operation is an abstract method, and derived class implements the detail logic for rendering the actual visualization with its own method.

- **Workout Summary**

Class Workout Summary works as an abstraction, or summary of individual workout records. When such an instance is constructed, target date range and indicators should be provided by Workout User and calculated from the data inside Workout Record DB. Then information collected within a day is aggregated, and those values are put in a list to form a Workout Summary. This class is especially useful when rendering Table and Graph.

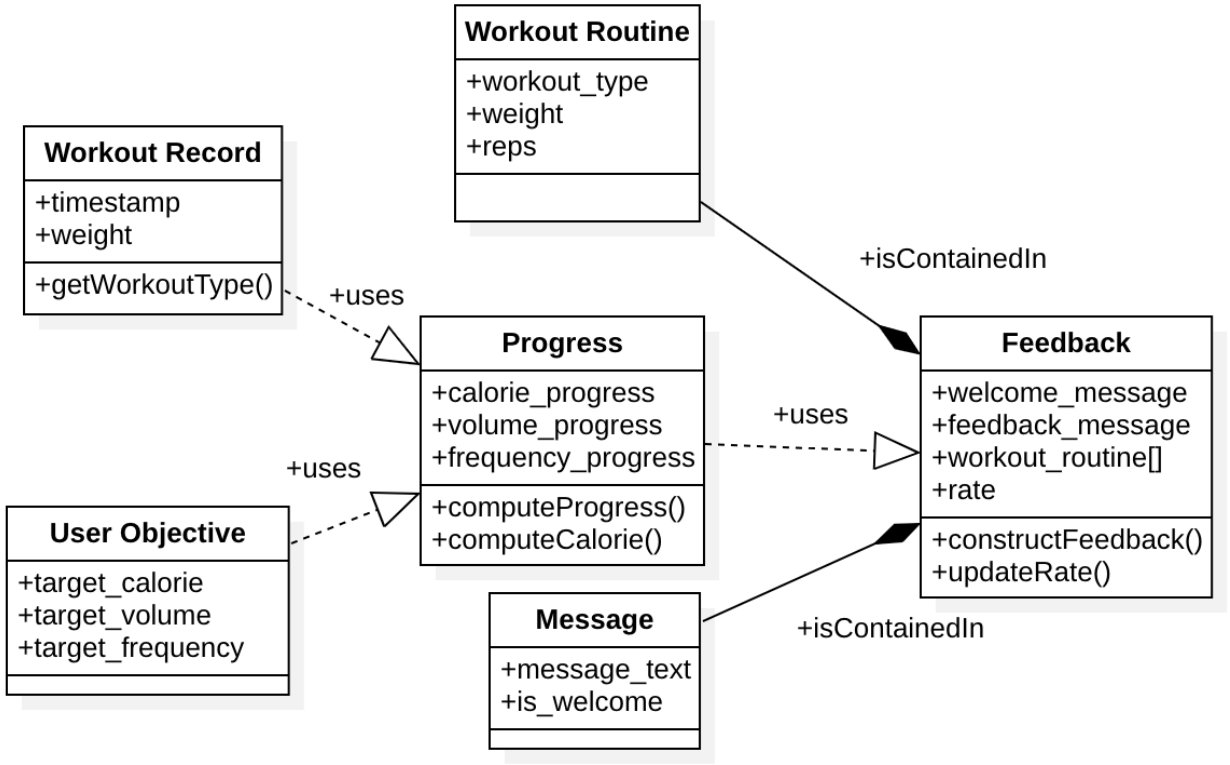
6.4 Record Analysis & Prescription System

6.4.1 Design Concerns

The related functional requirements are as below:

- FR-18, FR-19, FR-20, FR-21, FR-22, FR-23, FR-24

6.4.2 Design Elements



[Figure 6.6: Class Diagram for Record Analysis]

The Record Analysis System produces the feedback as an analysis result. To construct the feedback, progress of the workout is calculated. Progress could be calculated by collecting the user's workout record, considering the user's objective. After progress is produced, the system constructs feedback by joining suggested workout routines and feedback messages.

6.4.3 Criteria for Evaluation

The main goal of the Record Analysis System is constructing appropriate feedback. The attribute 'rate' is assigned by the user's review of the feedback. The rate will be evaluated dichotomously, asking the user if the feedback was helpful or not. The feedback algorithm will be updated by a certain period, reflecting the rate of each feedback.

6.4.4 Design Rationale

- Progress

The class progress helps unify the criteria of each objective. For now, there are three major workout objectives (lose weight, gain muscle, and maintain health). We can implement a single algorithm to construct feedback since every workout record is processed to the same form, the progress. If the alternative workout objective or criteria is added in further iteration, we can easily implement it. The workout routine and messages are modularized independently, which makes it easy to edit and add contents.

- Feedback

Feedback is a major product of the Record Analysis System, which is composed of welcome message, feedback message, and workout routine. Since the id is assigned for every constructed feedback, we can manage the past feedback. The results of analysis are displayed in various web pages, so feedback class helps store and manage recent results of analysis.

6.5 Viewpoint Source

The logical viewpoint provides basic structures of classes, logics for the systems, and algorithms for key functions. The main source of this logical viewpoint is the development team. The software development team shall implement the software based on the above structures and logics. Furthermore, in order to integrate the Smart Barbell System into the Smart Gym System, teams of the Smart Gym Development Companies should refer to the structure of the system via this section.

7. Information Viewpoint

The purpose of the Information Viewpoint is to describe the model of persistent data used within the framework. There are four groups of data entities colligated by the characteristics: user, workout, feedback, and music. [Figure 7.1] shows the Entity-Relation Diagram for the data model used by the framework.

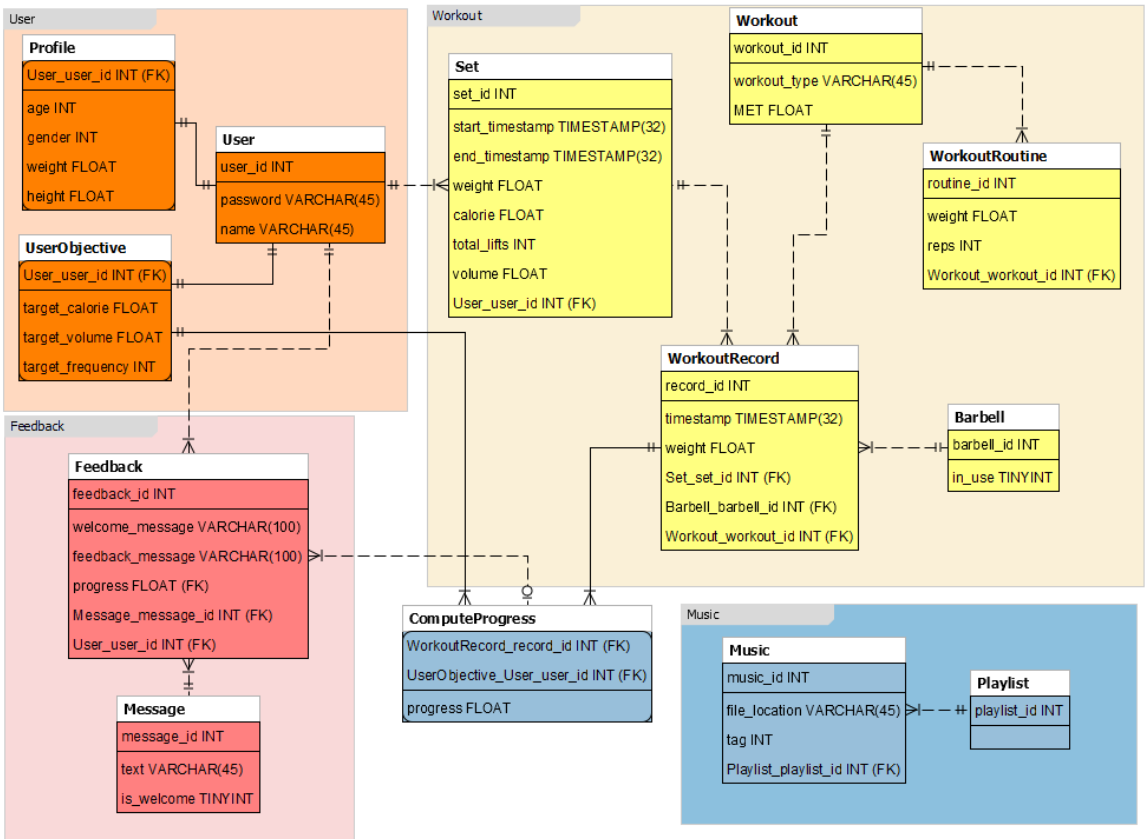
7.1 Design Concerns

This section describes the data of Smart Barbell System, so it covers all the functional requirements listed in [3.3 Functional Requirements]. The non-functional requirements which have considered importantly are as following:

- SIR-1, SIR-5

7.2 Design Elements

[Figure 7.1] is an ER(entity-relation) diagram for the entire Smart Barbell System. This diagram contains the information for the data model. Further descriptions of each design element are written below.



[Figure 7.1: ER diagram for Smart Barbell System]

User

- Type: Data Entity
- Description: A basic information of the user to register. Detailed information of the user belongs to other data entities, such as Profile and Objective. The user_id and password attributes exist for identification and authentication. The name is for representing users, but not key information.
- Attributes:
    - user_id (int)
    - password (string)
    - name (string)
- Relations:
    - each User has an associated Profile data entity
    - each User can have at most one User Objective data entity
    - each User can have zero or more Feedback data entity
    - each User can have zero or more Set data entity
    - each User can have zero or more Workout Record data entity

Profile

- Type: Data Entity
- Description: Detailed information of the user, mainly about the physical characteristics. The data is used to calculate consumed calories, and to provide appropriate feedback. Profiles can be empty(null) until users submit their information.
- Attributes:
    - age (int)
    - gender (int)
    - weight (float)
    - height (float)

User Objective

- Type: Data Entity
- Description: Information of user's objective about workout. To use feedback service, the purpose and value of corresponding attributes should exist. Purpose is the user's purpose of workout, expressed as one of three strings (lose_weight /gain_muscle /maintain_health). Each purpose matches attributes target_calorie, target_volume, target_frequency respectively. For example, if the user's purpose is to lose_weight, there must exist a target_calorie to get valid feedback.
- Attributes:
    - purpose (string)
    - target_calorie (float)
    - target_volume (float)
    - target_frequency (int)

Workout Record

- Type: Data Entity
- Description: A table of workout records, including who, which barbell is used, when the lift is done.
- Attributes:
    - record_id (int)
    - user_id (int)
    - barbell_id (int)
    - workout_id (int)
    - timestamp (timestamp)
    - weight (float)
- Relations:
    - each Workout Record is associated with one Barbell data entity
    - each Workout Record is associated with one User data entity

Barbell

- Type: Data Entity
- Description: Data entity that manages each barbell.
- Attributes:
    - barbell_id (int)
    - in_use (boolean)

Set

- Type: Data Entity
- Description: One set of the workout. Created by processing the workout record of the user, after the user finished one set. The start and end timestamp is created by the oldest and newest timestamp of the queried Workout Record data entity. The attribute total_lift is derived by counting numbers of rows in the queried Workout Record data entity. Volume is calculated by multiplying weight and total_lifts.
- Attributes:
    - set_id (int)
    - user_id (int)
    - workout_id (int)
    - start_timestamp (timestamp)
    - end_timestamp (timestamp)
    - weight (float)
    - calorie (float)
    - total_lifts (int)
    - volume (float)

Feedback

- Type: Data Entity
- Description: Data entity that represents all contents of feedback. It contains which user is getting feedback, the messages of feedback, calculated workout progress, and which workout routine is suggested. Progress is calculated with User Objective and Set of users.
- Attributes:
    - feedback _id (int)
    - user_id (int)
    - routine_id (int)
    - welcome_message (string)
    - feedback_message (string)
    - progress (float)

Workout Routine

- Type: Data Entity
- Description: The Workout Routine contains information on what kind of workout and how many reps will be done.
- Attributes:
    - routine _id (int)
    - weight (float)
    - reps (int)
- Relations:
    - each Workout Routine can contain one or more Workout data entity

Workout

- Type: Data Entity
- Description: Represents the type and MET of workout. For example, deadlift ,squat, bench-press can be the workout_type.
- Attributes:
    - workout _id (int)
    - workout_type (string)
    - MET (float)

Message

- Type: Data Entity
- Description: Predefined set of feedback message. Welcome message is noted with is_welcome, which has a shorter length.
- Attributes:
    - message _id (int)
    - text (string)

    - is_welcome (boolean)

Playlist

- Type: Data Entity
- Description: List of music, which has information of who created the list
- Attributes:
    - playlist _id (int)
    - user_id (int)
- Relations:
    - each Playlist can contain one or more Music data entity

Music

- Type: Data Entity
- Description: Information of music including tempo. Tag is labeled with three classes: calm, neutral, and exciting.
- Attributes:
    - music _id (int)
    - file_location (string)
    - tag (int)

7.3 Criteria for Evaluation

In this viewpoint, the structure of each data entity, relationship and dependency between each entity are critical points. Since structural integrity is most important in this viewpoint, increasing the variety of test datasets is more efficient than increasing the number of control flows. The test data will contain invalid form, duplicated data, insufficient data to register, and other kinds of data to test the database management.

7.4 Viewpoint Source

The informational viewpoint provides definitions of all data entities and relationships between them. It helps understanding the whole system from the perspective of data. Mostly, the development team should be able to understand information flow and the data structure of the whole system through this viewpoint.

7.5 Design Rationale

The informational viewpoint has many design elements. We classified the Data Entity to four categories, each is related to user, workout, feedback, and music. Each element is related but represents different kinds of data. User contains information of user account, and profile which represents physical information. Workout is composed of workout record, type, routine, set, and barbell. Music is an independent component, since it is used only when real-time feedback is given.

The primary key and foreign keys are well defined in this viewpoint, which makes it easier to implement and modify an actual database in future.

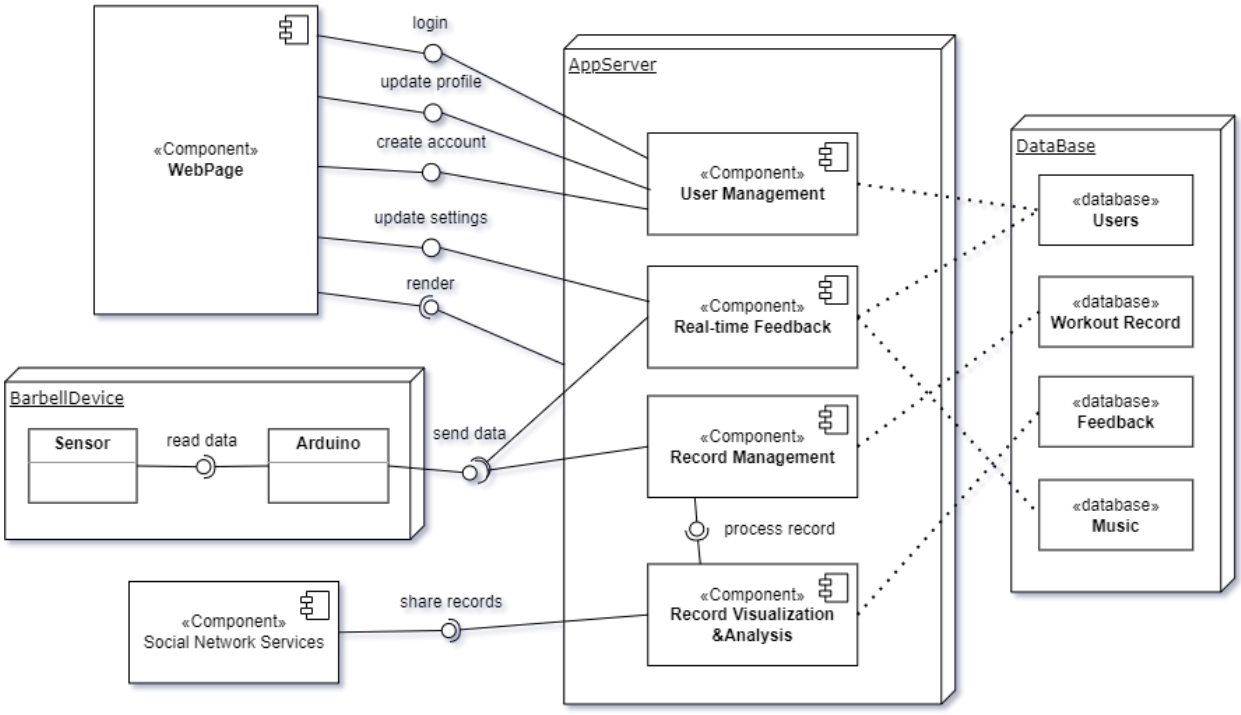
8. Interface Viewpoint

8.1 Design Concerns

The purpose of the Interface Viewpoint is to describe the interaction between system components. Components within the framework may interact with each other, and have access to databases. To simplify the interface between webpage and app server, app server block contains 4 main software components. [Figure 8.1] shows the diagram of the overall system interface. Since this section covers the whole system, all the functional requirements listed in [3.3 Functional Requirements] are related. The non-functional requirements which have considered importantly are as following:

- HIR-3, SIR-1, SIR-4, SIR-5, SIR-6
- QR-1, QR-2, QR-3, QR-4, QR-5, QR-6, QR-7, QR-8, QR-9

8.2 Design Elements



[Figure 8.1: Interface Component Diagram of the System]

User Management

- Exposes
    - Create account - permit a user to create account
    - Login - permit users to log into the service
    - Update profile - updates when user change their profile information
- Requires
    - DBA(Users) - database access to create, update, delete, and fetch basic user information and profile

Webpage

- Exposes
    - Create account
    - Login
    - Update profile
    - Update Settings
- Requires
    - Render - get all data from app server to render each pages

Real-Time Pace Management

- Exposes
    - Update settings - update current workout settings such as weight, target pace
- Requires
    - DBA (Users) - database access to modify user information and update profile
    - DBA (Music) - database access to get music and playlist

Record Management

- Exposes
    - Process record - collect workout data from workout record database, process, and return the results to appropriate form so record visualization system and record analysis system could use
- Requires
    - DBA (Workout Record) - database access to save new workout record, read existing workout record, and

Record Visualization & Analysis

- Requires
    - DBA (Feedback) - database access to collect prepared feedback components

8.3 Criteria for Evaluation

To evaluate interface viewpoints, each interaction between components should be verified. First, the connection of the Barbell Devices and app server will be tested to match the QR-1. For now, there is only one prototype device. After further iterations, the number of Barbell Devices matching the QR-1 will be used as criteria as well as the connection time. Second, the control flow test cases based on QR-2, QR-3, QR-4, QR-7 will be executed to test the interface between the web page and the app server. The database access will be tested until it satisfies QR-5, QR-6. Finally, the flow from the web page and Barbell Devices to the app server and database are tested with automated testing tools, such as selenium, matching QR-9.

8.4 Viewpoint Source

Interface viewpoint focuses on the interaction of software components. Development team should understand this viewpoint to implement proper interaction, especially accessing databases, connecting IoT devices, and connecting web clients. Customers, for example the gym owners could refer to this chapter to understand interaction of device and the server.

8.5 Design Rationale

Interface viewpoint shows interaction between each component. We made groups of components, which are BarbellDevice, Appserver, and DataBase. Therefore, the interface relationship between web page and software components (subsystems) are easily defined and shown. If new components are added in future, we could easily understand the interface by classifying into these groups.
Moreover, we defined a new component ‘Record Management’ to clarify the flow of data from the Arduino to the database and subsystems.

9. Interaction Viewpoint

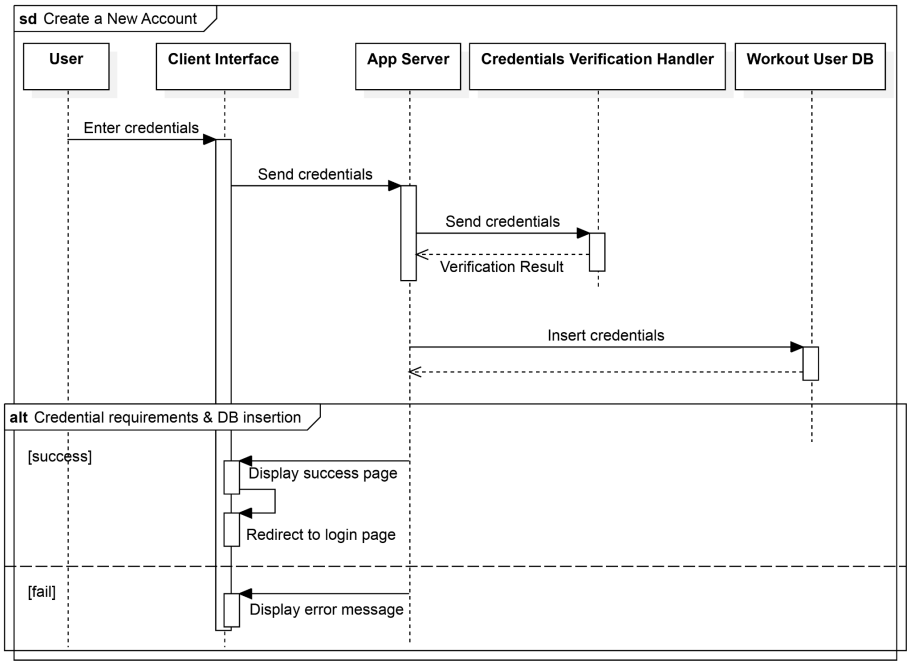
9.1 Design Concerns

The Interaction Viewpoint defines interaction among entities by discovering flow of information or messages, and control logic of related objects.

9.2 Design Elements

- 9.2.1 Create a New Account

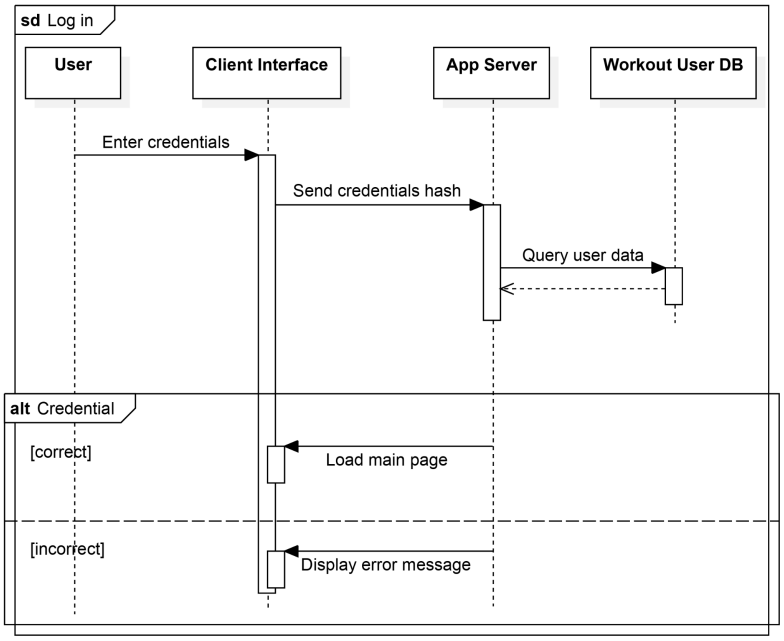
[Figure 9.1] illustrates interaction occurs when the Workout User give credentials in order to create a new account. Credentials Verification Handler is responsible for verifying types and formality of user inputs, including check of character allowance, length, and strength of the password. Control branch derived from the database insertion error is designed to ensure the consistency and atomicity of the database.



[Figure 9.1: Sequence Diagram for Create a New Account interaction]

- 9.2.2 Log in

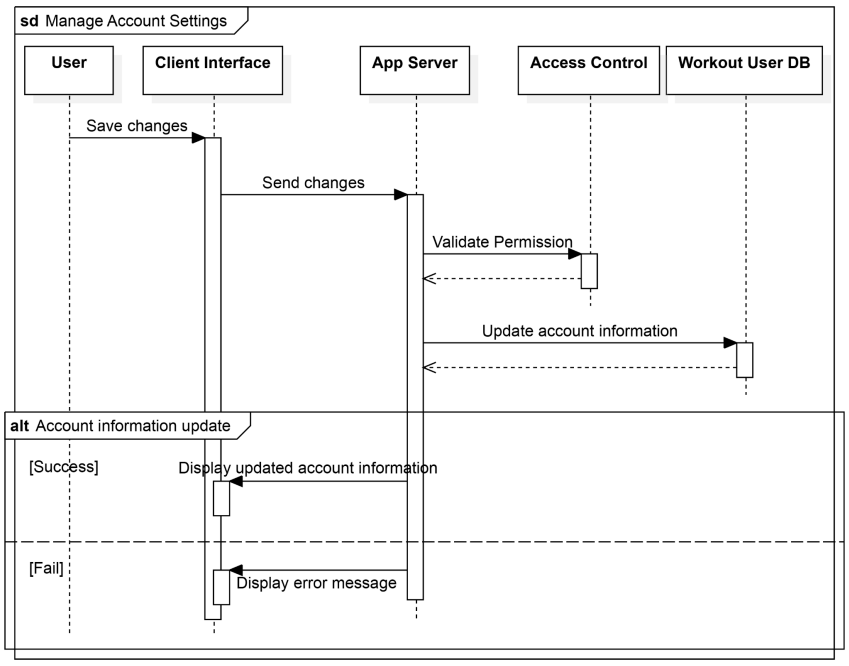
[Figure 9.2] illustrates interaction of user log in. Credentials are cryptographically hashed, and sent to the App Server. If there exists an account matching with given credentials, App Server generates an auth token and delivers it. When the operation was successful, the user should see the main page customized with personal information.



[Figure 9.2: Sequence Diagram for Log in interaction]

- 9.2.3 Manage Account Settings

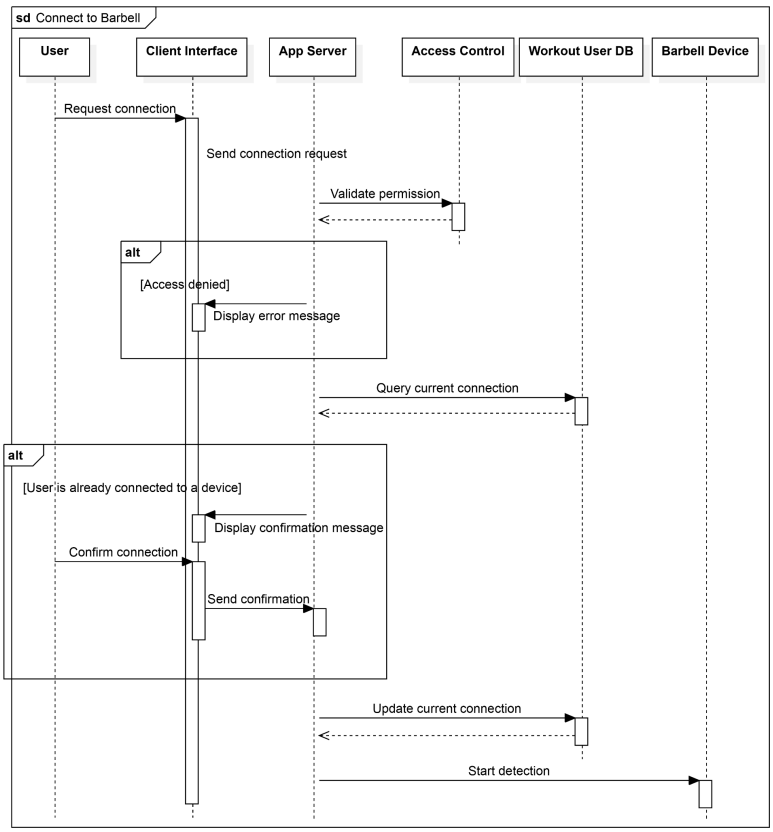
[Figure 9.3] illustrates the interaction of the user and the system when the user tries to edit account profiles. When the App Server receives a request, it validates the authenticity with Access Control. This happens in almost every interaction described in the current viewpoint. When the operation is successful, account information in Workout User DB must be revised, and updated information should appear in Client Interface accordingly.
Managing System Settings, done by System Administrator can be described in a similar way, thereby omitted.



[Figure 9.3: Sequence Diagram for Manage Account Settings interaction]

- 9.2.4 Connect to Barbell

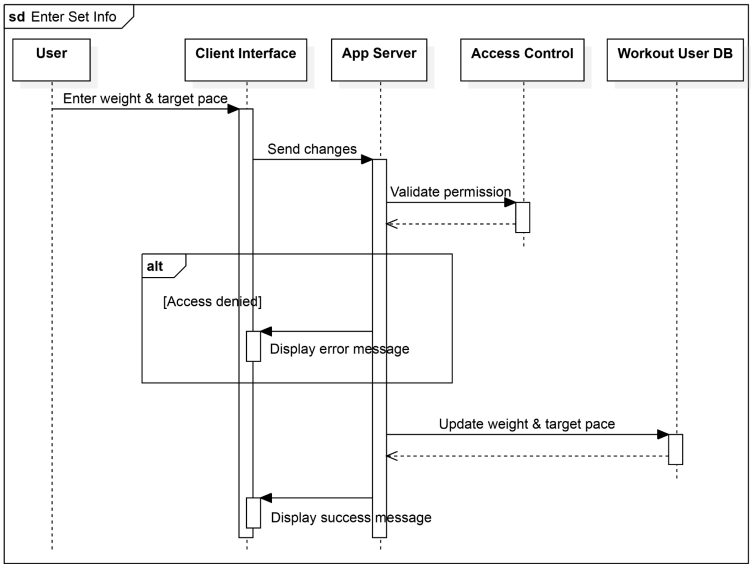
[Figure 9.4] illustrates interaction occurs when a Workout User connects to a Barbell Device. Unlike other interactions, there exists an extra confirmation branch when the user is known to be using another device already. If the operation was successful, the Barbell Device gets a message instructing it to start detection.



[Figure 9.4: Sequence Diagram for Connect to Barbell interaction]

- 9.2.5 Enter Set Info

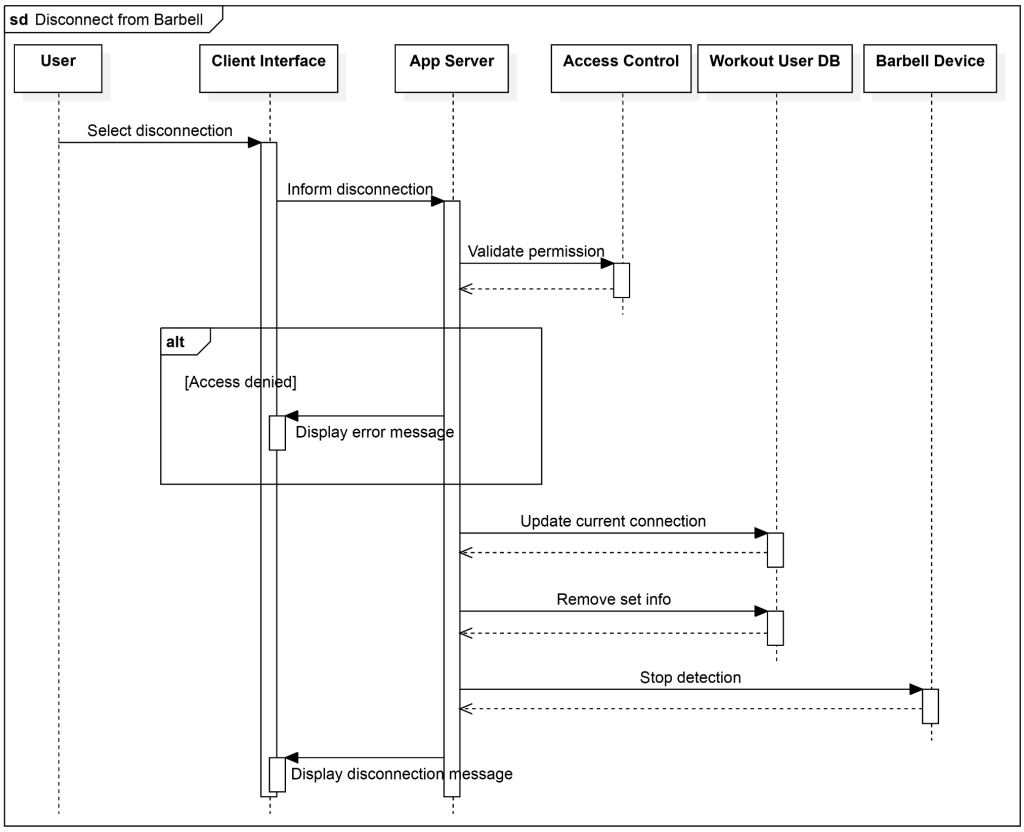
[Figure 9.5] illustrates the interaction of entering Set Info. If the operation was successful, updated weight and target pace gotten from Workout User should be applied to Workout User DB.



[Figure 9.5: Sequence Diagram for Enter Set Info interaction]

- 9.2.6 Disconnect from Barbell

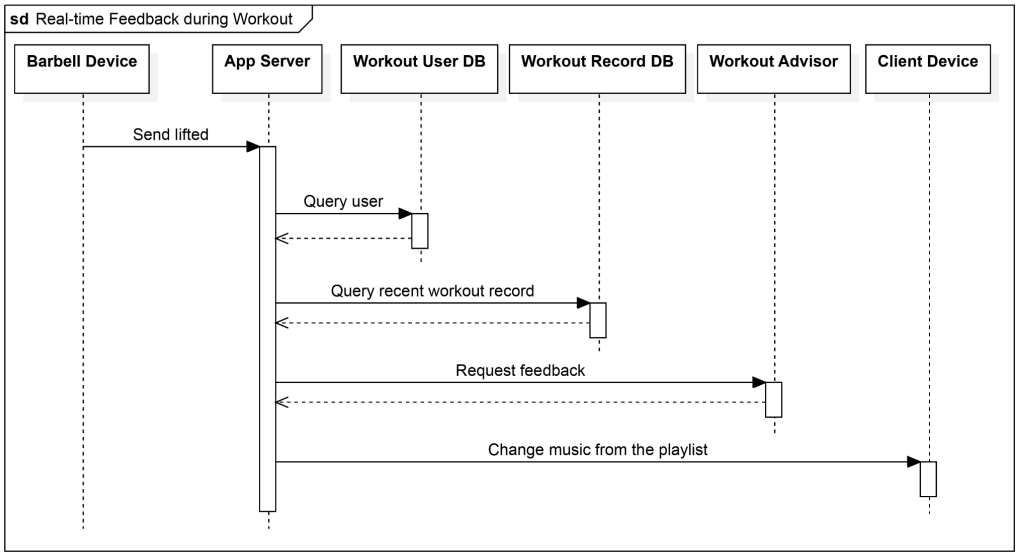
[Figure 9.6] illustrates interaction of Workout User and the system when the user disconnects themself from the Barbell Device. App Server sends a message to Workout User DB to remove set info, which was specified at the beginning of the workout, and also sends a message to Barbell Device to stop detection. These can be shown as a clean-up process.



[Figure 9.6: Sequence Diagram for Disconnect from Barbell interaction]

- 9.2.7 Real-time Feedback during Workout

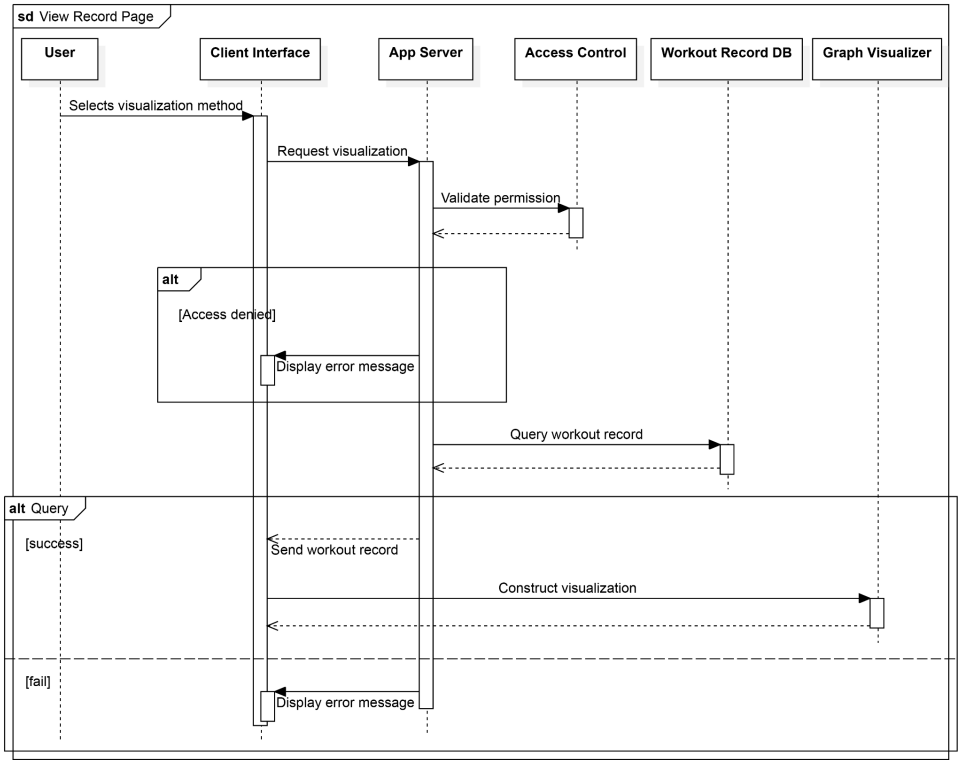
[Figure 9.7] illustrates interaction of real-time feedback given by the system during the workout. It starts with a lift signal provided by Barbell Device. Both of two databases participate in the interaction, one for querying the Workout User using the device, another for querying their recent workout record to provide real-time feedback. Determining the value of the feedback is responsible by the Workout Advisor. At the end of the operation, App Server sends a message to Client Device to change music according to the feedback.



[Figure 9.7: Sequence Diagram for Real-time Feedback during Workout interaction]

- 9.2.8 View Record

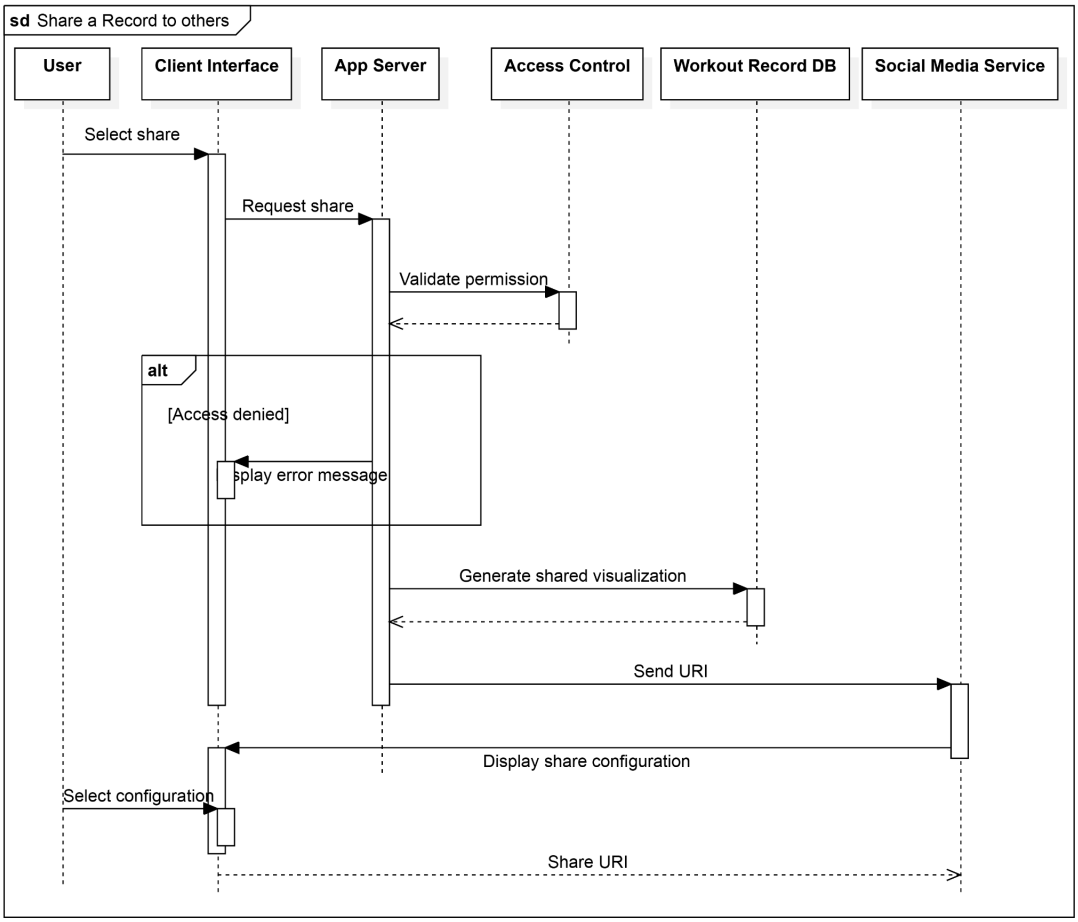
[Figure 9.8] illustrates interaction occurs when Workout User requests a visualized record. Workout record of the specified user is delivered in an usual way. Note that Client Interface is responsible for calling external service which renders tables and graphs in the frame. This separates the concern of detailed visualization from the system.



[Figure 9.8: Sequence Diagram for View Record Page interaction]

- 9.2.9 Share a Record to others

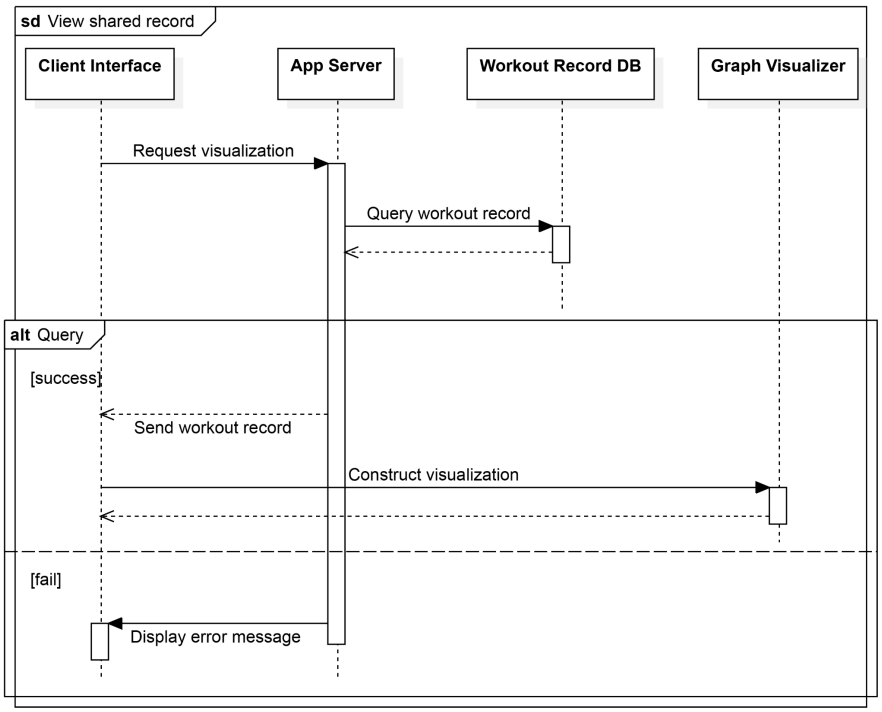
[Figure 9.9] illustrates the interaction of sharing a visualized Workout Record. Shared visualization is saved in Workout Record DB, and expected to act as a cache for frequent future inquiries. After that, the generated URI of the shared visualization is sent to the platform-specific Social Media Service, and the Workout User configures which social media and whom to share it.



[Figure 9.9: Sequence Diagram for Share a Record to others interaction]

- 9.2.10 View shared record

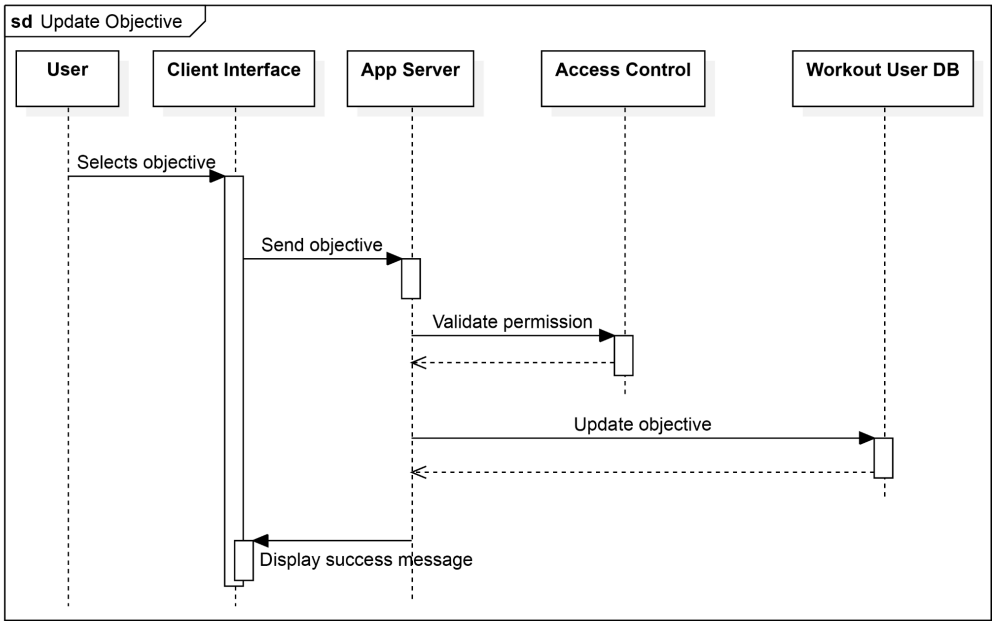
[Figure 9.10] illustrates the interaction of viewing visualization of the record shared in advance. The interaction starts with arbitrary user's access to the shared visualization URI. Note that there isn't a process of checking permission since shared visualization has become public in the sharing stage. The rest is the same as [9.2.8 View Record].



[Figure 9.10: Sequence Diagram for View shared record interaction]

- 9.2.11 Update Objective

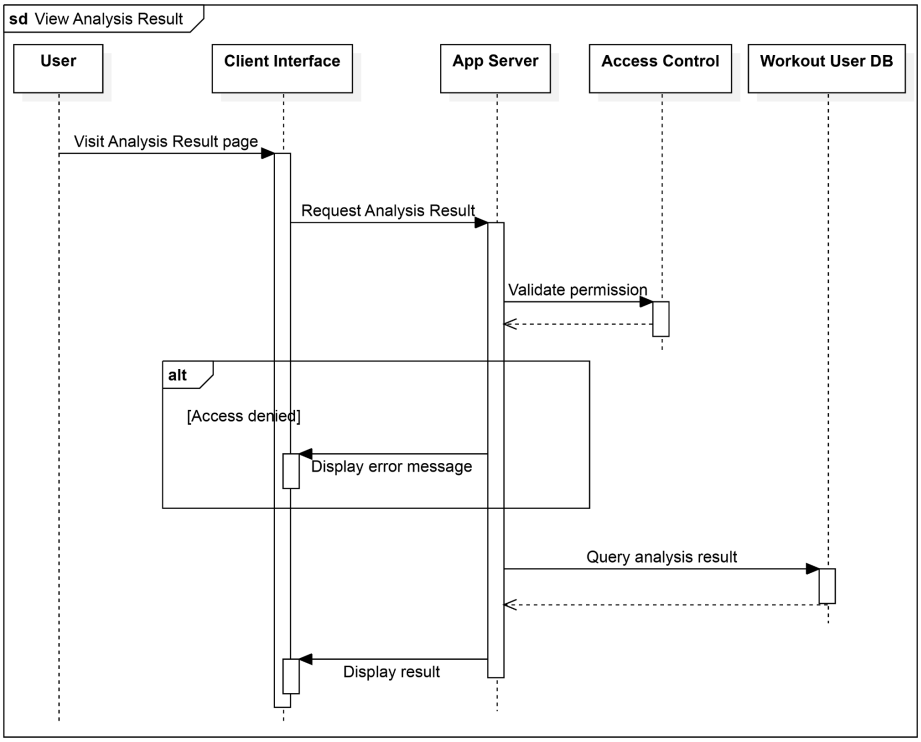
[Figure 9.11] illustrates that interaction occurs when a Workout User updates their objective. Updating Goal follows the equivalent flow.



[Figure 9.11: Sequence Diagram for Update Objective interaction]

- 9.2.12 View Analysis Result

[Figure 9.12] illustrates interaction occurs when Workout User requests analysis of their historical workouts. Workout analyses for all users are done regularly in advance, then inserted to Workout User DB. Thereby Workout User can retrieve the analysis result directly from the database.



[Figure 9.12: Sequence Diagram for View Analysis Result interaction]

9.3 Criteria for Evaluation

The main point of the viewpoint is to describe all entities and messages they send and receive in the entire framework. Possible flaws are missing entities participating in an interaction, unconsidered branches and unexpected exceptions. Extensive testing activities conducted over functional implementations will discover the majority of those flaws. The sequence of interaction can be verified if it meets the security requirements described in [3.4.4 Security].

9.4 Viewpoint Source

The interaction viewpoint emphasizes interaction between entities covering the entire framework. The development team should refer to this viewpoint in order to find out which object participates in a specific task, and what are the roles of it during the interaction.

9.5 Design Rationale

Interaction sequences designed are based on use cases and functional requirements, which ensures and validates why and when those interactions happen. Candidates for possible entities and their message flows were considered to ensure compliance with non-functional requirements for all interaction sequences.
If possible, a component was introduced to clarify the message between entities, particularly corresponding to the interface proposed in [8 Interface Viewpoint].

10. Implementation (Progress & Plan)

Our team worked on implementing the very initial stage of the Smart Barbell, and here we show the current progress of the implementation.

10.1 User Interface Design

Based on critical functional requirements (except login, register), our team developed user interface and flow using a tool called Figma.
Currently, ten screens exist, and connections between the screens are defined using Figma Prototype Interactions. Our Figma design can be accessed here.

10.2 Implementation Progress

As mentioned in section 4, our architecture is chosen with the emphasis on modularization, so implementation is done separately on the four main functionalities. Most of our implementation is maintained as a git repository, and it can be viewed here.

- 10.2.1 User Management System

User Management functionality is a very typical functionality that also exists in various softwares. We found a Django-based minimal register, login implementation and integrated it into our software.

- 10.2.2 Real-Time Pace Management

Based on pseudocode provided on [6.2.2], we developed a program that is able to detect the highest and lowest point on Arduino. Also, we managed to play music depending on the target pace by simple python script. In terms of functional requirements ID, we currently implemented FR-8 and FR-10.

- 10.2.3 Record Visualization System

A prototype capable of displaying a graph along with several options has been developed. The prototype utilizes Django's template feature and a graph is integrated in the frame by using Chart.js, a Javascript library for charting.

- 10.2.4 Record Analysis System

No progress is made to this part yet. There will be more study to find algorithms to calculate consumed calories precisely, and implement the progress calculating algorithm. Sample arbitrary workout routines and feedback messages are prepared, but further be updated based on the advice of expert athletes or trainers.

10.3 Plan for Iterations

For the next iteration, the highest priority is to finish the implementation of the Real-Time Pace Management system. Plans related to associated functional requirements are the following.

- **FR-10: Pace Decision**
  As explained in [6.2.4 Design Rationale] of the Real-Time Pace Management System, the calculation for current pace decision is simple, comparing the current pace to the target pace. In further iterations, the workout status of the user will be calculated with more measures and parameters.
- **FR-11: Music Selection**
  As explained in [6.2.2 Design Elements] and [6.2.4 Design Rationale] of the Real-Time Pace Management System, the music selection is in its early stage. The tags for each music will become diverse, and the music selection method will change from changing the music immediately when a request comes, which is the current level.
  Furthermore, in future iterations, we aim to integrate this feature with external music streaming applications.

Next, the development of front end tasks and back end tasks will be done. Deployment to the server will be done incrementally, synchronizing with our team's sprint interval.

11. Summary

This software design description document of Smart Barbell explains the Smart Barbell system in a multiple viewpoint - Contextual viewpoint, Pattern viewpoint, Logical viewpoint, Information viewpoint, Interface viewpoint, and Interaction viewpoint. For the ease of explanation, we have included critical functional requirements and non-functional requirements - a summary content of our Software Requirements Specification document. We tried to include explicit information of Smart Barbell including sequence diagrams, deployment diagrams, class models, and the actual pseudo code. We strongly believe that a developer team can have a solid understanding of what Smart Barbell is and how it acts by looking at this software design description document.