Software Requirements and Design Document

For

Group 4

Version 1.0

Authors:
Amanda Orama
Matthew Cegala
Nicholas Holguin
Matthew Hummel
Ashton Singpradith

1. Overview (5 points)

SmartGymTracker is a web application for use as a gym metrics tracking application. This web application will track various data points such as sets, reps, weight, body composition, and other values a person would care about when analyzing their health. SmartGymTracker looks to implement a milestone reward tracker and personally curated suggestions to make people more motivated to continue working out. These tracking, trend analysis, and suggestion featured are going to be designed in the intent to make the app simple for those who want a basic app, but also more personalized and in-depth for those who want to see the fine details of their fitness journey.

2. Functional Requirements (10 points)

1. The system shall contain project data including the Requirements and Design Document, Implementation and Testing Document, and any other relevant documentation relating to the architecture and necessary information about the application. **(High)**
2. The system shall have a proper class library and structure setup that will allow developers to easily understand the systems and the necessary information to be sent between them. **(High)**
3. The system shall have a communication setup to allow communication between the front-end, back-end, and database systems to allow for data to be sent between them. **(High)**
4. Users shall be able to record biometric data such as weight, height, body measurements, and step count. Users shall be able to navigate from the dashboard to pages for viewing/adding workouts and biometric data (**Medium)**
5. Users shall be able to add, update, and delete their workout data. This includes general workout data, exercises, sets within an exercise, and biometrics specific to a workout. This should include separate pages in the form to account for all the related information in the class libraries. **(High)**
6. The system shall have a proper User Identity Management, IDM, system in place to allow users of separate privilege level and accounts to be able to see different information. This includes a front-end login screen and back-end communication setup for secure password checking. This also includes the ability to create, delete, and update users. **(High)**
7. Users shall be able to have a searchable library to pick for their selection of muscles worked and exercises. This shall be only editable by the admin users. **(Low)**
8. Users shall be able to see a user management page to update user specific information. For admins there shall be an additional user management page that allows them to edit roles and permissions. **(Low)**
9. Users shall be able to view their workout data in a simple dashboard. **(Low)**
10. Users shall be able to view previous workouts **(High)**
11. The system should be able to properly link workouts and their respective components to users and vice versa. **(High)**
12. The database should be able to rely on the front-end and back-end to provide proper validation techniques to prevent failures. This is a common practice to clean the data before querying and sending data to the database. **(High)**
13. Users shall be able to see a notification page to be used with any messages that the system provides to them. **(Medium)**
14. Users shall be able to set goals about their progress and receive notifications about their progress towards those goals. **(Medium)**
15. Users shall be able to receive notifications about milestones that the system has set as potential milestones in their progress. This includes the ability of the user to be able to filter and fine-tune the milestone sensitivity for their desire to either receive more or none. **(Medium)**

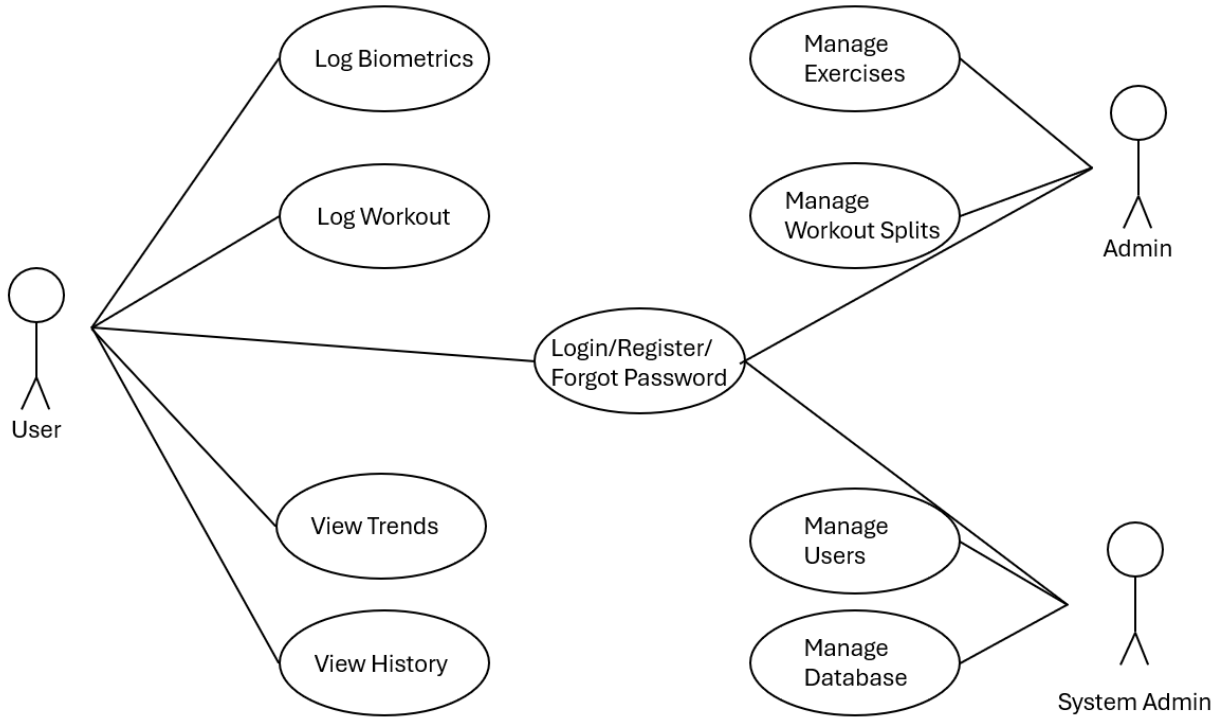
16. Users shall be able to see trends in their workout data. This includes graphs and data trend breakdowns based on their previous data entered including workouts, biometrics, exercises, weight, reps, duration. **(High)**

3. Non-functional Requirements (10 points)

1. The system shall respond to user interactions (for example, clicks) within 2 seconds
2. User data entered in forms shall be validated for correct format (numbers for weight/reps, text for exercise names, etc.)
3. The system shall maintain consistent styling using Tailwind CSS for buttons, forms, and page layouts.
4. Data exchange between frontend and backend shall use JSON format for clarity and reliability.
5. The database should provide fast access to data though efficient queries.
6. The system should be protected against many security vulnerabilities, including cross site scripting and SQL injection attacks.
7. The database shall store the encrypted passwords in the database, and the frontend shall encrypt the password before transmitting the data.
8. The system shall be able to handle multiple concurrent users and ensure that the system can handle the load.
9. The database should ensure data integrity and have transaction processing in place to ensure data corruption does not occur.
10. The database should be able to expect data to be properly sanitized and cleaned by the frontend and backend before inserting queries and data manipulation.
11. The user interface should be clean and easy to navigate to ensure user friendly experience.
12. Multiple accesses to the same information should be prevented in the event a raise condition should occur in order to prevent unintended consequences.
13. Data should be properly secured and protected to ensure that only users with proper privileges can access the data.

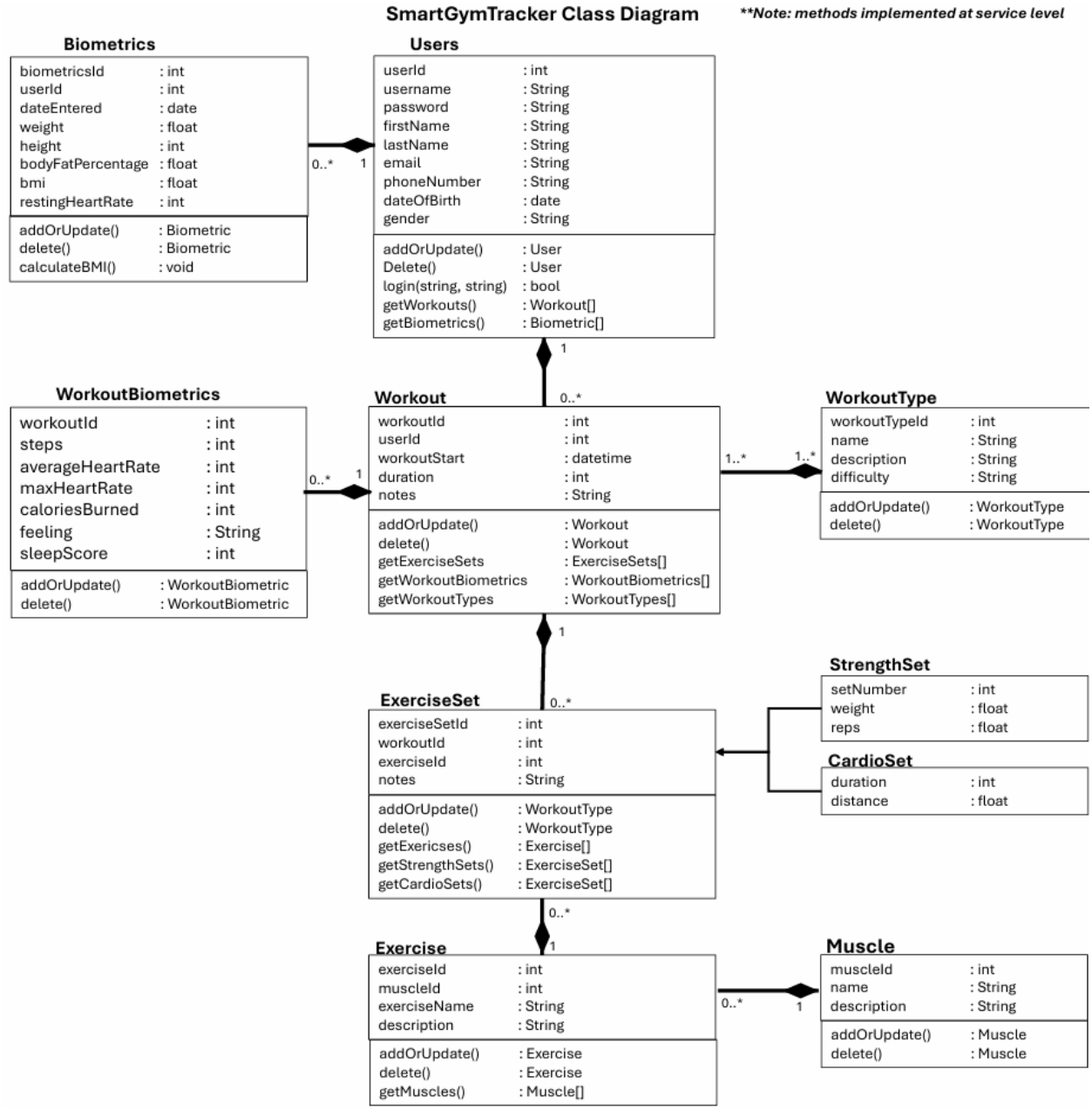
4. Use Case Diagram (10 points)

For the first increment, textual descriptions for the use cases are not required.

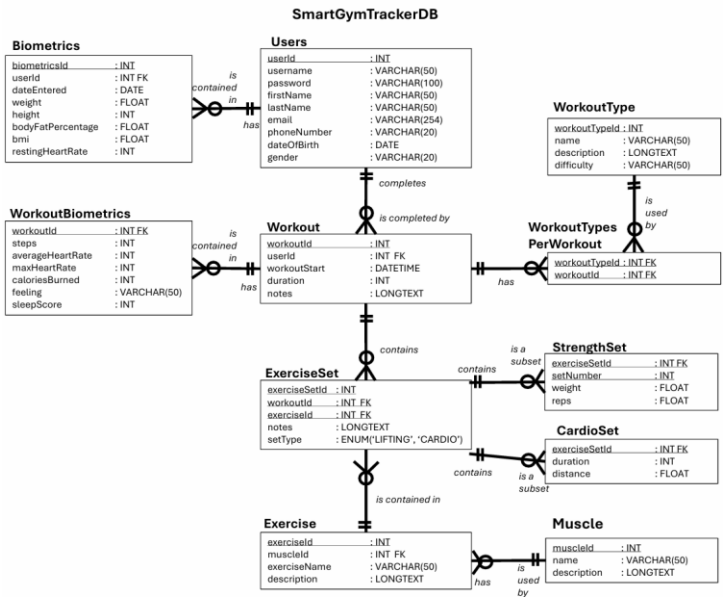


5. Class Diagram and/or Sequence Diagrams (15 points)

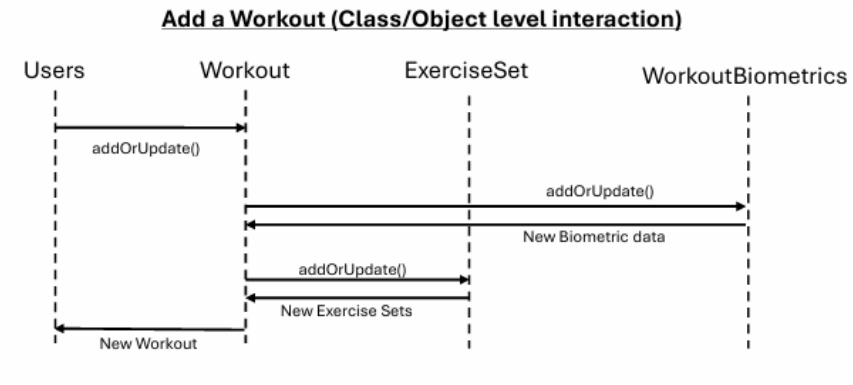
Class Diagram



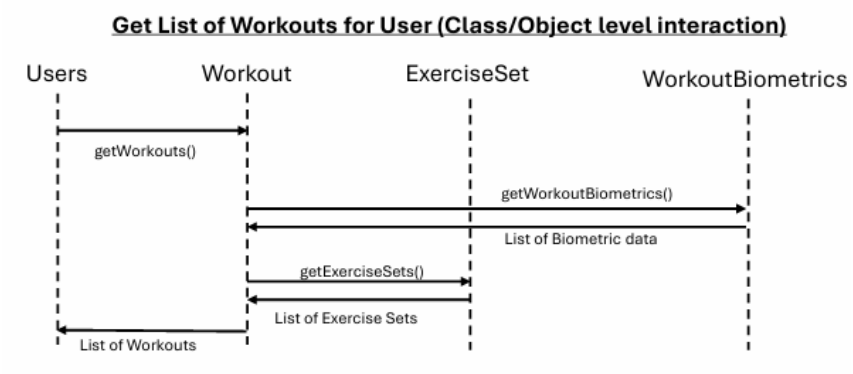
Entity-Relationship Diagram (ERD) for Database



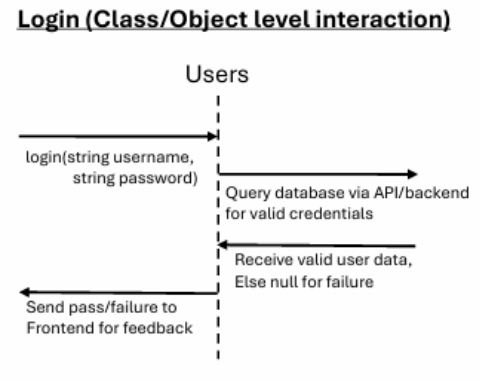
Sequence Diagram 1 (Create a Workout):



Sequence Diagram 2 (View Workouts):



Sequence Diagram 3 (Login):



6. Operating Environment (5 points)

Web application, Frontend: React + Tailwind CSS
Backend: C#, ASP.NET Core Web REST API
Database: 10.4.32-MariaDB is what database is tested on. This is an opensource MySQL database.
OS/Browser: Requires web browser that allows for accessing localhost locations, including Microsoft Edge and Google Chrome.

7. Assumptions and Dependencies (5 points)

1. It is assumed that the user is accessing the frontend via a computer web browser.
2. It is assumed that localhost communication is setup on the device accessing the application with ports 80 and 3306 and 5074 are available. This also includes the assumption that localhost communication is available.
3. It is assumed that frontend, backend, and database communicate with JSON formatted data packets between the system based on the defined class library.
4. Users will enter numeric data correctly (checked by frontend).
5. Third party libraries: React, Tailwind CSS, Maria Database/MySQL, REST API
6. It is assumed that database access will be in the valid formats and available for storing workouts and biometrics
7. It is assumed that the operating environment has the proper memory and applications, MySQL, to run the application properly.
8. user login/authentication is for Increment 2