Software Requirements and Design Document

For

Group 4

Version 3.0

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.

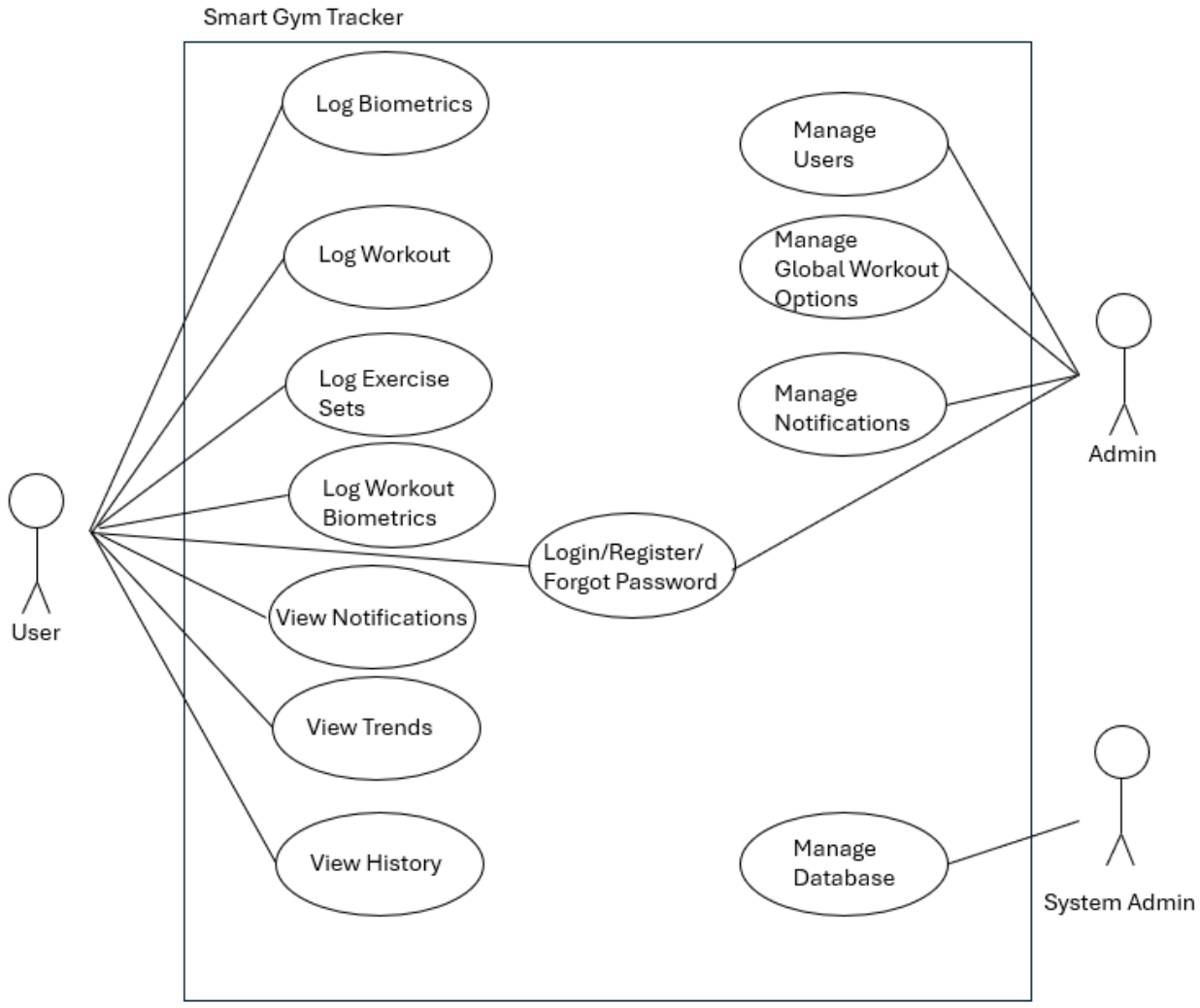
2. Functional Requirements (10 points)

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 the front-end, back-end, and database systems to send data 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 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 relevant messages. **(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 hash 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)



Login

1. Name: Login
2. Description: Describes how a user will enter the main system.
3. Participating actors: User, Admin
4. Preconditions: The user has a valid account in the system.
5. Postconditions: The user is able to see either the admin or user specific pages depending on the privilege level.
6. Basic flow of events:
    a. User reaches the login screen.
    b. The user either enters their username/password.
    c. The user clicks login.
    d. The system either presents a successful login prompt.

    e. The user is redirected to either the user or admin home screen depending on the privilege level.
7. Alternative flows
    a. Invalid username or password in step D.
        i. The use case will loop back to part C or ends with result saying invalid username/password.
8. Special requirements
    a. Password will be stored in an encrypted hash format in the database.
    b. The hashing should be done in the frontend.

Register

1. Name: Register
2. Description: Describes how a user will create a new account.
3. Participating actors: User, Admin
4. Preconditions: The user can access the system/login screen.
5. Postconditions: The user will have a new account created.
6. Basic flow of events:
    a. User reaches the login screen.
    b. The user selects create a new account.
    c. The user inputs the desired information based on the screen.
    d. The user clicks create my account.
    e. The system presents the user with a successful login creation.
    f. The system redirects the user back to the login screen.
7. Alternative flows
    a. Invalid information entered in step D due to duplicate username or unentered required field.
        i. The use case will loop back to step C or ends with result saying invalid data entered.
8. Special requirements
    a. Created password shall be stored in encrypted hash format in the database.
    b. The hashing should be done in the frontend.

Forgot Password

1. Name: Forgot Password
2. Description: Describes how a user will enter the main system
3. Participating actors: User, Admin
4. Preconditions: The user can access the system/login screen.
5. Postconditions: The users password is reset.
6. Basic flow of events:
    a. User reaches the login screen.

    b. The user selects forgot my password.
    c. The user inputs their email.
    d. The user clicks send email.
    e. The system sends an email with information on how to reset their password.
    f. The user enters their new password.
    g. The system redirects the user back to the login screen.
7. Alternative flows
    a. Invalid email entered in step D
        i. The use case loops back to step C or ends with prompt saying invalid email.
8. Special requirements
    a. Created password shall be stored in encrypted hash format in the database.
    b. The hashing should be done in the frontend.
    c. The link sent will expire after a set time.

Log Biometrics

1. Name: Log Biometrics
2. Brief Description: User adds or updates their personal biometric data.
3. Participating actors: User
4. Preconditions: The user is logged into the system.
5. Postconditions: The user's biometric data is timestamped and saved to the system.
6. Basic flow of events:
    a. The user selects add/update biometrics page.
    b. The user enters the intended data.
    c. The user clicks save.
    d. The system sends the data to database and saves them.
7. Alternative flows:
    a. Invalid data entered in step C.
        i. The use case loops back to step B or ends with prompt saying invalid data entered.
8. Special requirements
    a. Timestamp is to be created by system of date entered.

Log Workout

1. Name: Log Workout
2. Brief Description: The user adds a workout session.
3. Participating actors: User
4. Preconditions: The user is logged in.
5. Postconditions: The user's workout data is saved to the system.
6. Basic flow of events:

    a. The user selects the add/update workout page.
    b. The user enters the intended workout data.
    c. The user clicks save.
    d. The system send the data to the database and saves them.
7. Alternative flows
    a. Invalid data ented in step C
        i. The use case loops back to step B or ends with prompt saying invalid data entered.
8. Special requirements
    a. None

Log Workout Biometrics

1. Name: Log Workout Biometrics
2. Brief Description: The user logs workout specific biometrics.
3. Participating actors: User
4. Preconditions: The user is logged in and there is a valid workout specified.
5. Postconditions: The workout specific biometrics are saved.
6. Basic flow of events:
    a. The user selects a workout.
    b. The user navigates to the add/update workout biometrics for that workout.
    c. The user enters their specified data.
    d. The user clicks save.
    e. The system saves the data and sends it to the database.
7. Alternative flows
    a. Invalid data entered in Step D
        i. The use case loops back to step C or ends with prompt saying Invalid data entered.
8. Special requirements
    a. The workout id must be tagged and saved to database to link the workout biometrics.

Log Exercise Set

1. Name: Log Exercise Set
2. Brief Description: The user enters exercise set data that is either of type cardio set or workout set.
3. Participating actors: User
4. Preconditions: The user is logged in and a valid workout exists.
5. Postconditions: The exercise set data is saved.
6. Basic flow of events:
    a. The user selects a workout.

    b. The user navigates to the add/update an exercise set.
    c. The user selects either a cardio or strength set.
    d. The user selects the desired data.
    e. The user selects save.
    f. The system saves the data to the database.
7. Alternative flows:
    a. Invalid data entered in step E.
        i. The use case loops back to part c or ends with prompt saying invalid data entered.
8. Special requirements
    a. System must support multiple sets per exercise and increment set number in the database for strength set.

View Notifications

1. Name: View Notifications
2. Brief Description: The user shows system produced notifications.
3. Participating actors: User
4. Preconditions: The user is logged in.
5. Postconditions: The user is able to see a list of all notifications specific to them.
6. Basic flow of events
    a. The user selects the notification inbox page.
    b. The system obtains a list of all notifications specific to that user and displays them to the user.
7. Alternative flows
    a. None
8. Special requirements
    a. None

View Trends

1. Name: View Trends
2. Brief Description: The user is able to view trends and graphs of their workout and exercise progress over time.
3. Participating actors: User
4. Preconditions: The user is logged in and there is multiple data points available for that user.
5. Postconditions: The data is presented in charts and graphs.
6. Basic flow of events:
    a. The user selects a specific data type to generate a trend of.
    b. The system grabs that data from the database.
    c. The system performs calculations and generates an output.

7. Alternative flows:
    a. There is not enough data to produce a valid trend data.
        i. The use case ends and display error message.
8. Special requirements
    a. There must be charts, graphs, or averages of data depending on the type of data.

View History

1. Name: View History
2. Brief Description: The user is able to view past workouts and other data elements.
3. Participating actors: User
4. Preconditions: The user is logged in and has at least one previous data element.
5. Postconditions: The data is retrieved and displayed.
6. Basic flow of events
    a. The user selects either a previous workout, exercise set, or biometric entry.
    b. The system obtains that data from the database.
    c. The system displays that information.
7. Alternative flows
    a. There is no history to be grabbed.
        i. The use case ends and error message is displayed.
8. Special requirements
    a. None

Manage Database

1. Name: Manage Database
2. Brief Description: The system admin performs needed maintenance and database updates.
3. Participating actors: System Admin
4. Preconditions: The system admin is able to access the MySQL administrator page.
5. Postconditions: The needed database updates and maintenance is completed.
6. Basic flow of events:
    a. The user access the MySQL administrator page.
    b. The user logins in with valid credentials.
    c. The user performs need schema or user updates directly on database.
7. Alternative flows:
    a. Invalid credentials provided for step B.
        i. The use case ends or prompts the user to enter valid credentials.
8. Special requirements
    a. Database access should be local only and be limited to only shared admin accounts with protected password.

Manage Notifications

1. Name: Manage Notifications
2. Brief Description: The admin creates global notifications and messages that users can be subscribed to.
3. Participating actors: Admin
4. Preconditions: The user is logged in and has admin privileges.
5. Postconditions: The notification is either created/updated/deleted.
6. Basic flow of events:
    a. The admin navigates to manage notifications page.
    b. The admin either creates/updates/deletes a notification rule.
    c. The system applies those changes.
7. Alternative flows:
    a. Invalid rule applied in step B.
        i. The use case ends and sends error message.
8. Special requirements
    a. None

Manage Global Workout Options

1. Name: Manage Global Workout Options
2. Brief Description: Additions, deletions, or updates applied to lists of workout types, exercises, or muscles.
3. Participating actors: Admin
4. Preconditions: The admin is logged in.
5. Postconditions: The global values are updated systemwide.
6. Basic flow of events:
    a. The admin navigates to the update workout lists.
    b. The admin selects the desired entry to update.
    c. The admin saves the changes.
    d. The system updates the database and any fields with those entries.
7. Alternative flows:
    a. Entry has existing references and cannot be deleted in step C.
        i. The use case displays error message and ends or loops back to step C.
8. Special requirements:
    a. Data integrity, legacy values, and cascading changes need to be applied to prevent invalid data.

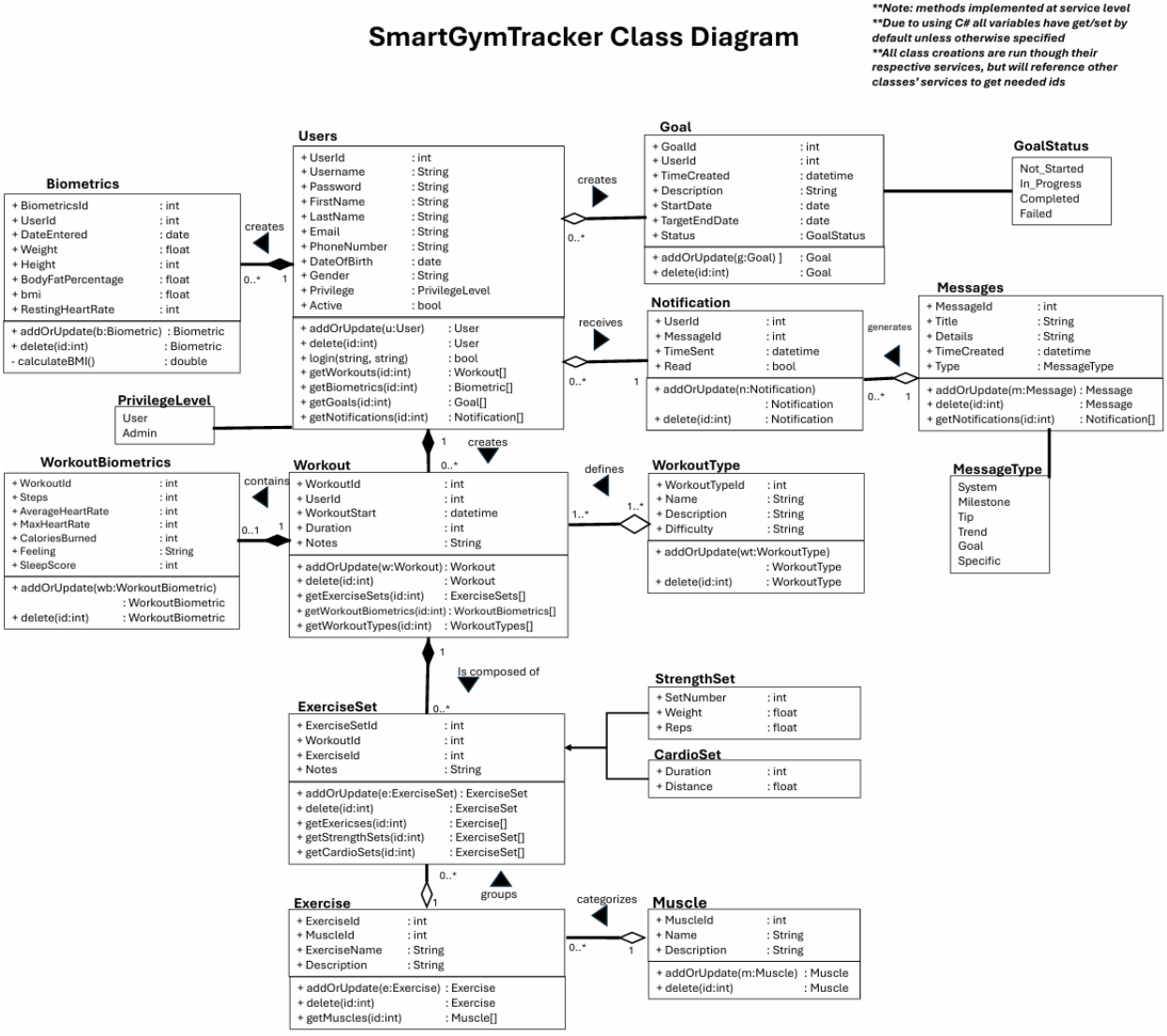
Manage Users

1. Name: Manage Users
2. Brief Description: The admin can update, deactivate, or assign permissions to user.

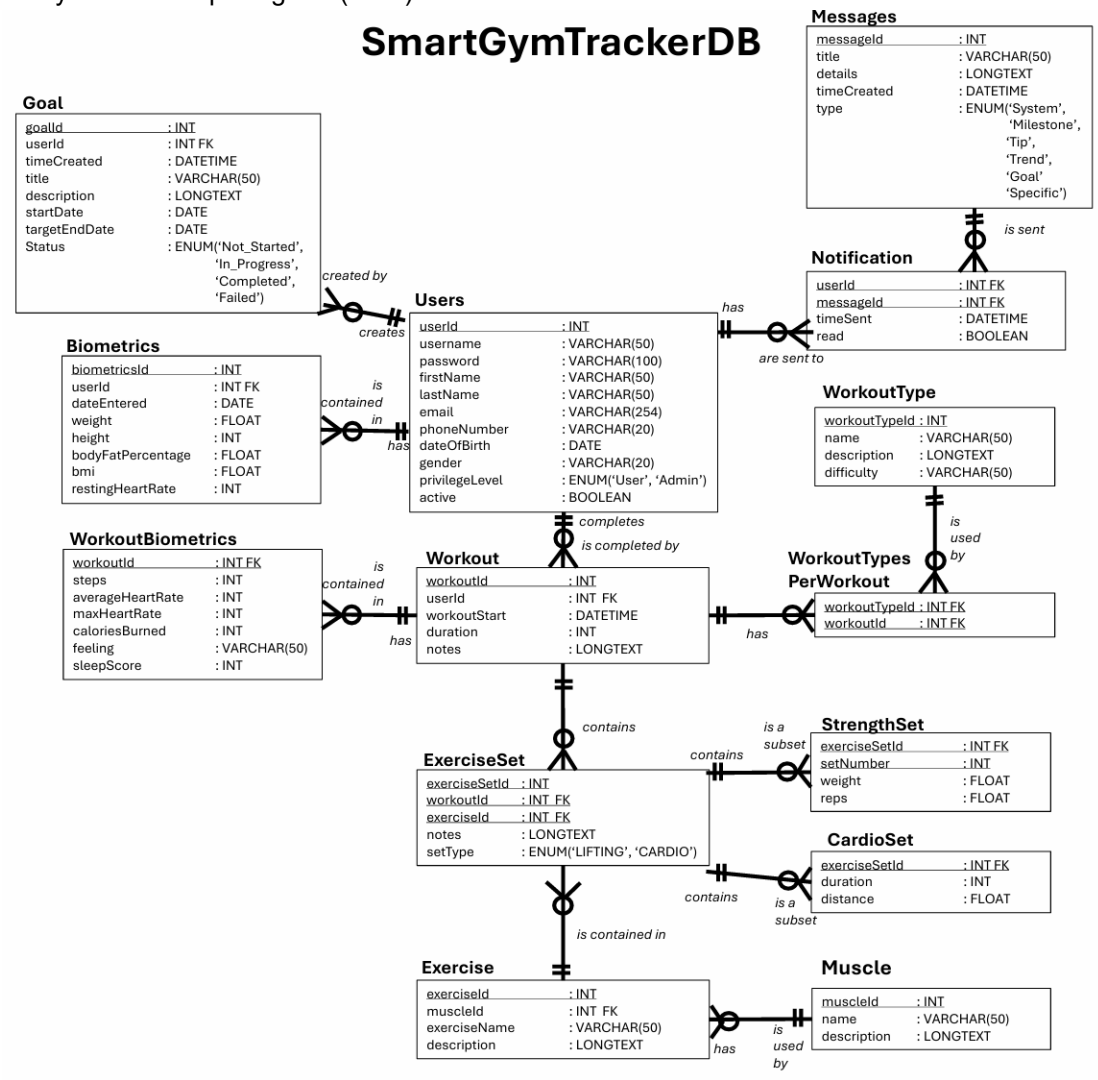
3. Participating actors: Admin
4. Preconditions: The admin is logged in.
5. Postconditions: The users values are updated.
6. Basic flow of events:
    a. The admin navigates to the user management page.
    b. The admin selects a user to update.
    c. The admin edits the details and permission, or deletes the user.
    d. The admin saves the changes.
    e. The system applies those updates to the database.
7. Alternative flows
    a. Invalid change applied in Step D.
        i. The use case loops back to Step C or ends and prompts the admin with invalid update.
8. Special requirements
    a. There must be logs of any updates.
    b. There must be session updates from changing a user's information.

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

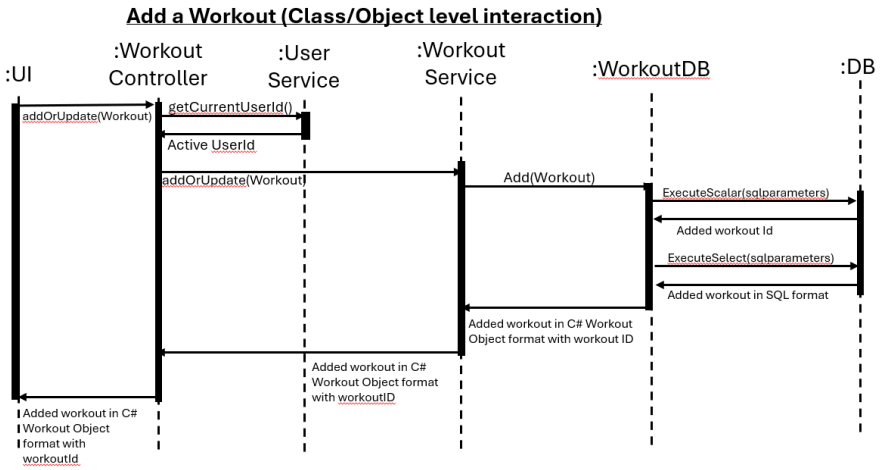
Class Diagram



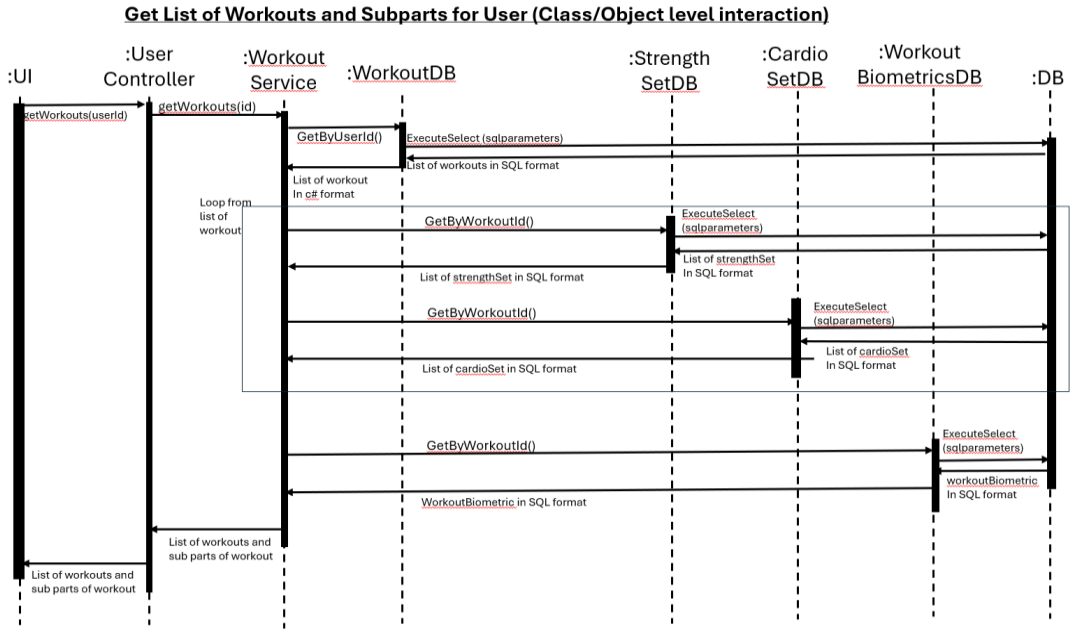
Entity-Relationship Diagram (ERD) for Database



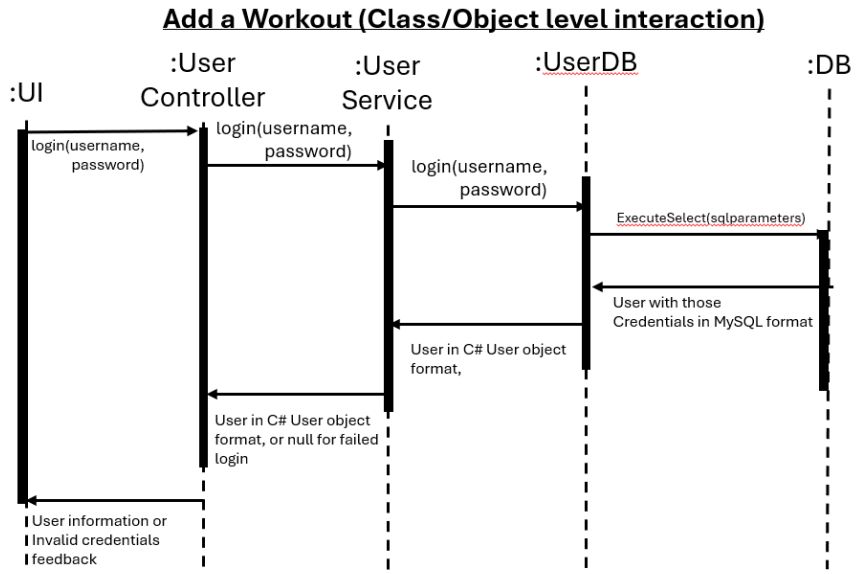
Sequence Diagram 1 (Create a Workout):



Sequence Diagram 2 (View Workouts and subparts):



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. Setup instructions can be found in MySQL.SmartGymTracker/README.md

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.