



CP317 – Software Engineering


Project Title: Workout Logger

All team members read and approved this report and acknowledge that the information provided is not a reuse or recycling of any previous project. All team members have contributed equally to the work presented in this report.


Member 1 Name: Nick Lalonde

Signature 


Member 2 Name: Benjamin Schmid

Signature 

Member 3 Name: Benjamin Clark

Signature 

Member 4 Name: Aidan Espiritu

Signature 

Date submitted: 07 – 12 – 2024

Table of Contents

1	44
1.1	44
1.2	44
2	6
2.1	6
2.2	6
2.3	88
2.4	88
2.5	99
3	1010
3.1	1010
3.1.1	1010
3.1.2	1515
3.2	1516
3.2.1	1616
3.2.2	1616
3.2.3	1616
3.3	1717
3.3.1	1717
4	1818
5	Error! Bookmark not defined. 19
5.1	1919
5.1.1	2020
6	2626
7	2727

List of Figures

Figure 2.1: Context Diagram

Figure 3.1.1.1: Login Interface

Figure 3.1.1.2: Sign Up Interface

Figure 3.1.1.3: Navbar Interface

Figure 3.1.1.4: Quiz Interface

Figure 3.1.1.5: Workouts Interface

Figure 5.1: Use Case Diagram

Figure 5.1.1.1: UML Class Diagram

Figure 5.1.1.2: Sequence Diagram: Login

Figure 5.1.1.3: Sequence Diagram: Sign Up

Figure 5.1.1.4: Sequence Diagram: Quiz

Figure 5.1.1.5: Sequence Diagram: View Workouts

Figure 5.1.1.6: Sequence Diagram: Sign Out

■ Figure 5.1.1.7: Sequence Diagram: Home Button

Figure 5.1.1.8: Component Diagram: Workout Page

1 Introduction

The increasing prevalence of obesity, currently affecting 30% of Canadians and continuing to rise, highlights the urgent need for accessible tools to promote physical activity and fitness. Physical exercise has been known as a cornerstone for improving not only physical health but also cognitive functioning and overall well-being. According to Mandolesi et al. (2018), regular physical activity enhances mental health, cognitive abilities, and emotional resilience, contributing to an individual's overall quality of life [2]. Despite these benefits, many individuals struggle to maintain consistent exercise habits, often lacking effective and engaging workout plans as a barrier.

This project aims to address this challenge by developing a Workout Routine Logger, a web application that creates personalized workout plans based on user inputs such as age, gender, fitness goals, and training experience. By offering customized and accessible fitness solutions, this platform aims to reduce common barriers to exercise, support long-term fitness, and maximize the physical benefits of regular activity. Our motivation stems from the significant health, and cognitive benefits associated with consistent physical activity. Personalized workout plans can help users choose a workout plan better suited for their goals, resulting in improved fitness outcomes and greater motivation to maintain an active lifestyle.

1.1 Problem Definition

The primary problem addressed by this project is the lack of accessible and personalized fitness tools to help individuals create effective workout plans. Many people struggle with the boredom and ineffectiveness of generic routines not suited to them or their goals, leading to inconsistent exercise habits and diminished fitness progress [3]. This workout logger not only delivers a workout plan based on user preferences, but allows users to create multiple workout plans and track progress by updating the weight achieved on each exercise.

The solution integrates React for an intuitive user interface, Flask for backend logic, and MongoDB for secure data storage. Key features include personalized workout generation, the ability to save, retrieve, and update workout plans, and a visually appealing quiz interface to gather user preferences.

The top-level acceptance criteria for the solution include:

- The ability to create and display tailored workout routines based on quiz responses.
- Persistent data storage for user accounts and workout plans.
- A user-friendly interface that supports viewing, updating, and managing saved workouts.
- Error handling to manage invalid inputs, missing data, and server issues.
- Effortless site navigation and secure session management

1.2 Literature Survey

Personalized workout plans are critical for improving consistency and dedication to exercise routines and achieving meaningful results. According to a study published by the National Center for Biotechnology Information (NCBI), "Personalized training programs lead to improved adherence and better health outcomes compared to standardized approaches" [4]. The study highlights the effectiveness of tailoring fitness regimens to individual needs but also notes a lack

of accessible tools to implement such strategies effectively. Moreover, it emphasizes the need for additional resources to validate the long-term benefits of personalized workout solutions, showing the gap in existing offerings and the demand for user-focused platforms.

■

■ Existing Platform: MyFitnessPal

What It Does Well:

- **Comprehensive Tracking:** MyFitnessPal does well in tracking various fitness metrics, including caloric intake, and physical activity, which supports users in maintaining a balanced fitness plan.
- **Community and Engagement:** The app fosters a social aspect through community features like friend connections, shared progress updates, and challenges.
- **Goal Customization and Progress Tracking:** MyFitnessPal allows users to set specific fitness goals, such as weight loss, muscle gain, or maintenance, and provides detailed progress tracking over time.

Room for improvement:

- **Lack of Personalized Workouts:** While MyFitnessPal is excellent for tracking, it does not generate individualized workout routines tailored to user goals, preferences, or fitness levels.
- **Limited Guidance for Beginners:** The app assumes users have a foundational understanding of fitness, leaving beginners without clear, actionable steps for starting or improving their routines.

2 Project Requirements

2.1 Product Scope

The workout Logger is a web-based application that generates personalized workout routines for users, enables users to track their progress, and saves data for future workout sessions. It included user authentication for user login, a quiz to generate a workout routine for users, and a workout logging interface. All of this combined seamlessly integrates to a database for saving and loading user data.

The primary goal of the Workout Logger is to address a common challenge faced by gym goers; maintaining motivation and consistency. While many individuals start their fitness journey with enthusiasm, the lack of a structured routine and an effective medium to track progress often leads to disengagement and loss of motivation. This application aims to mitigate this shortfall in the gym industry by providing users with a personalized, engaging, user friendly platform that encourages consistency, and simplified progress tracking.

Objectives include: provide secure login system to manage user accounts, deploy a quiz that generates tailored workout plans, develop workout logging feature that saves and loads user data, and ensure all relevant data is stored and retrieved using a database

Benefits of this application include

- **Personalization:** Users receive workout routines tailored to their preferences and goals through a dynamic quiz
- **Progress Tracking:** Allows users to log weights during workouts and review their progress over time.
- **Ease of Use:** A simple, intuitive interface ensures accessibility for all users
- **Improve Gym Motivation:** by displaying progress and personalized plans, the app helps user stay motivated as they see their progress improving

2.2 Product Perspective

○ The Workout Logger is a new, self-contained product designed to address challenges of personalized fitness tracking. It is not part of an existing product family, or a replacement for any current systems, but rather a new solution.

The application comprises three major components that interact to deliver its functionality:

1. **Front-end (React):** This is the user interface, written in Javascript, utilizing CSS/HTML, and leveraging the power of the React front-end framework, allows users to log in, create and account, complete the quiz, and interact with the workout logging system. Additionally, the UI also allows users to access previously saved workouts to pick up and log weights where they left off.

Major Components of Front-end:

- **Login Page:** Allows user to login, and for the application to save and retrieve user specific data
- **Create Account:** User can create account to utilize application

- **Navbar:** Displayed on every single page of our application (excluding the login and create account page), the navbar serves as a basic method of navigation. Component included in the Navbar include
 - **Signout:** logs users out of application and directs them to the login page
 - **Home Page:** Navigates users to home page to take quiz again
 - **View Workouts:** Allows users to select saved workouts that the quiz logic previously determined. Users can also delete said workouts
 - **Home page:** Houses the quiz for user to take to generate a workout and launch the workout page
 - **Workout Page:** Displays the workout routine. Within the workoutpage
 - **Workout Tables:** Formatted tables of workout routine. Within the tables, user can enter weights to log progress
2. **Back-end (Flask):** Written in python, the back-end handles the core application logic. It processes quiz results to generate personalized workout plans, manages user authentication, and acts as a bridge between frontend and the database. Communication is facilitated through REST APIs, ensuring efficient and reliable data exchange.
- Major Components of the Back-end:**
- **Classes**
 - **Driver Class (Server)**
 - handles all data going in and out from back-end and front-end
 - **User Class**
 - Handles validating user data from the database and creates user credentials and stores it into the database
 - **Workout Class**
 - handles all workout related data
 - Determines workout routine based on user input from the front-end and stores it into the database
 - Gets saved workouts and weights from database and sends it to the front-end
3. **Database (MongoDB):** Stores user profiles, personalized workout routines, and logged progress. It ensures all data is persistent and retrievable for future sessions, enabling users to maintain a consistent workout history.

As shown in figure 2.2 below, the major components front-end, back-end and database are shown within the center as they are the core of all operation of the application. As shown, user interacts with components of the front-end for user input, and front-end interacts with user to display information

The flask server, housing the driver class that handles all information being sent into the back-end and out to the front-end or database, also houses two main classes; The user class, represented to the right of the flask server component, and the workout class, represented to the left of the server component.

All components, front-end, back-end and database, seamlessly communicate with each other to facilitate all of our products requirements.

```
graph TD
    User[User] -- "User Interacts with Front-end" --> FrontEnd[Workout Logger React.js Front-End]
    FrontEnd -- "Front-end sends user data to back-end" --> BackEnd[Flask Back-end Server]
    BackEnd -- "Back-end sends information to back-end" --> FrontEnd
    BackEnd -- "Back-end sends user data to Database" --> MongoDB[(MongoDB Database)]
    MongoDB -- "Database sends user data to back-end" --> BackEnd
    FrontEnd -- "Components sends data to front-end" --> BackEnd
    BackEnd -- "Get saved workouts from Database" --> FrontEnd
    BackEnd -- "Determine Workout Routine VIA quiz" --> FrontEnd
    FrontEnd -- "Save Workout into Database based on user" --> BackEnd
    BackEnd -- "Validate User Credentials by checking database" --> FrontEnd
    BackEnd -- "Create Account: Store User data into database" --> FrontEnd
    BackEnd -- "Hash Password (bcrypt)" --> FrontEnd
```

The primary audience of the application will be fitness enthusiasts, or future fitness enthusiasts. They seek a tool to help track workout progress and personalize routines based on their preferences and goals. Users will interact with the app several times a week, depending on the frequency of their workout routines. The application caters to a diverse user base as the interface is designed to be intuitive, therefore, no technical expertise is required.

The workout logger must account for the following constraints:

[illegible]

- The Database utilizes MongoDB, therefore, design is constrained to NoSQL schemas.
- Development tools must support React, Python, and MongoDB.

2.5 Assumptions and Dependencies

Major Assumptions

- End-users will have a reliable internet connection, and a modern computer running a modern web browser that supports JavaScript and react.
- The development team will have the expertise of the various tools and frameworks used to develop the project, or will be able to quickly learn said tools.
- The applications front-end, back-end, and database will function as intended on a wide variety of machines and operating systems
- Sufficient data storage space will be available for the database
- All third-party libraries (Flask, pymongo, MUI, bcrypt, react) will remain supported, updated, and compatible with the projects requirements
- Encryption protocols for password hashing will meet industry standards
- Application will be hosted on a secure server

Dependencies

- The project depends on the **React library** and **MUI components** for UI design.
- The back-end relies on **Flask, pymongo, bcrypt, and Python** programming language
- MongoDB will be the database of choice for maintaining data persistence, and user credentials. Any outage will affect the usability of the application.
- The application will require a **hosting service** to host the app for deployment

Impact of Incorrect or Changing Assumptions

- If users do not have access to a modern web browser or stable internet, the application will be rendered useless.
- Any breaking changes in third-party libraries or tools could delay development or require a substantial redesign.
- MongoDB downtime could result in data loss and inability to use the application.
- Significant increase in the userbase beyond initial assumptions will lead to degradation in performance.

3 Product Specific Requirements

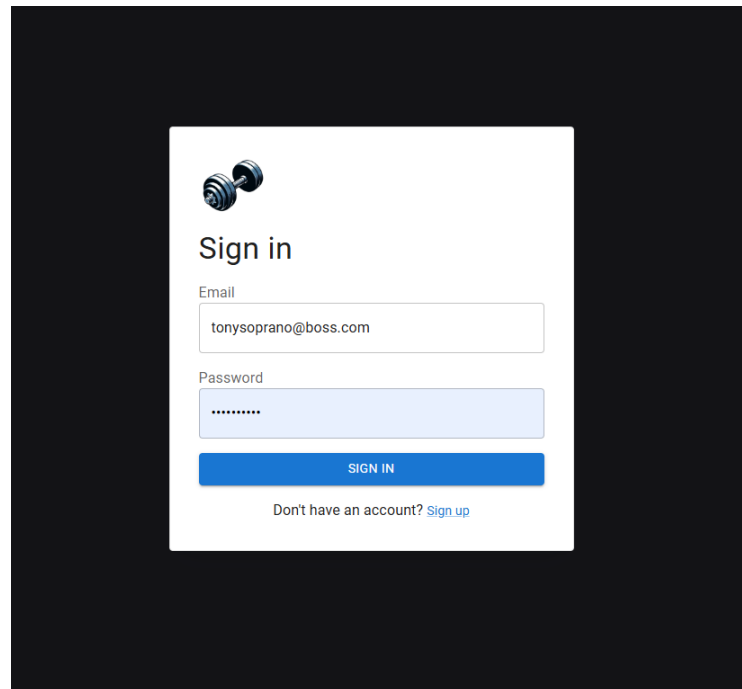
3.1 External Interface Requirements

3.1.1 User Interfaces

The user interfaces for the Workout Logger web application adheres to modern design standards by utilizing Material UI components to ensure accessibility, responsiveness, and usability across modern web browsers.

Login Interface

Figure 3.1.1.1



- **Purpose**
The login interface is used to authenticate users by providing a valid email and password. On successful authentication the user is then able to access the Home page. If a user does not already have a valid account, they are able to sign up and make one. Errors that occur are displayed under the input fields.
- **Components**
 1. **Email Field** – accepts user input for their email.
 2. **Password Field** – accepts user input for their password
 3. **Sign In Button** – allows users to submit their email and password to the back end to be verified.
 4. **Sign Up Link** – allows users to be redirected to the Sign Up page where they can create a new account.

- **Error Messages**

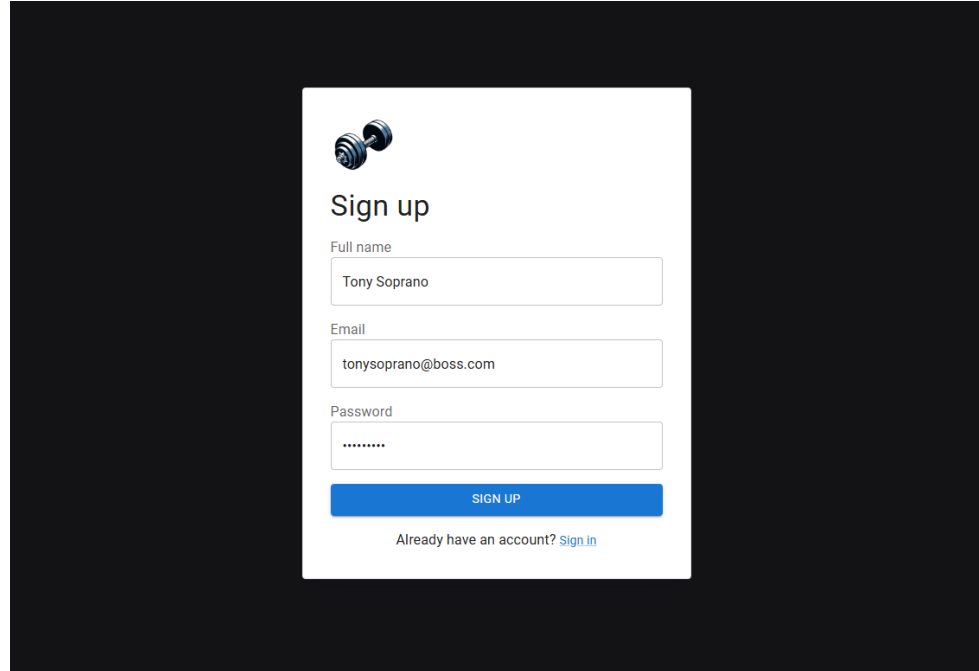
Error messages are used to notify issues such as empty fields or invalid input from users. They are displayed underneath their corresponding fields.

- **GUI Design Standards**

The Login interface follows the application's minimalist design theme. It contains a white container for key information on top of a dark black background. The white container and its components were sourced from Material UI.

Sign Up Interface

Figure 3.1.1.2

A screenshot of a 'Sign up' form. The form is a white rectangle centered on a dark background. At the top left of the form is a logo consisting of two interlocking spheres. Below the logo is the title 'Sign up'. There are three input fields: 'Full name' with the text 'Tony Soprano', 'Email' with the text 'tonysoprano@boss.com', and 'Password' with masked characters '*****'. Below these fields is a blue button labeled 'SIGN UP'. At the bottom of the form, there is a link that says 'Already have an account? Sign in'.

- **Purpose**

To collect essential user information (name, email, password) and create them a secure account.

- **Components**

1. **Name Field** – accepts input for the user's full name.
2. **Email Field** – accepts input for the user's email.
3. **Password Field** – accepts the input for the user's password.
4. **Sign Up Button** – allows users to submit their details for account creation. Information for name, email, and password is sent to the back end where it is stored for the next time they log in.
5. **Login Link** – redirects the users back to the login page in case they already have an account.

- **Error Messages**

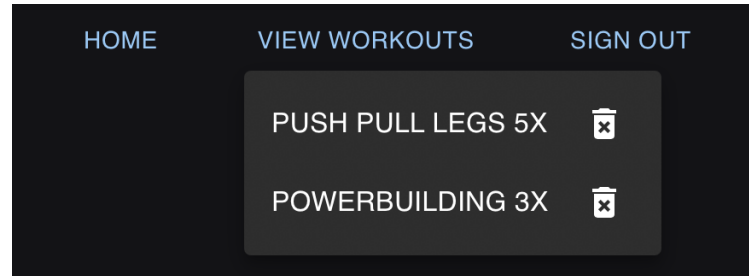
Error messages are displayed directly below the input fields if the user fails to meet valid credential input.

- **GUI Design Standards**

Like the Login page, the Sign Up interface follows the same design of the white container for key information on top of a dark black background. The white container and its components were sourced from MUI.

Navbar Interface

Figure 3.1.1.3



Located in the top right corner of the Home and Workouts pages.

- **Purpose**

The navbar interface is located on the Home page and the workouts page. It is used to direct the user around the application. The user can use the navbar to view workouts, delete workouts, logout or go back to the Home page.

- **Components**

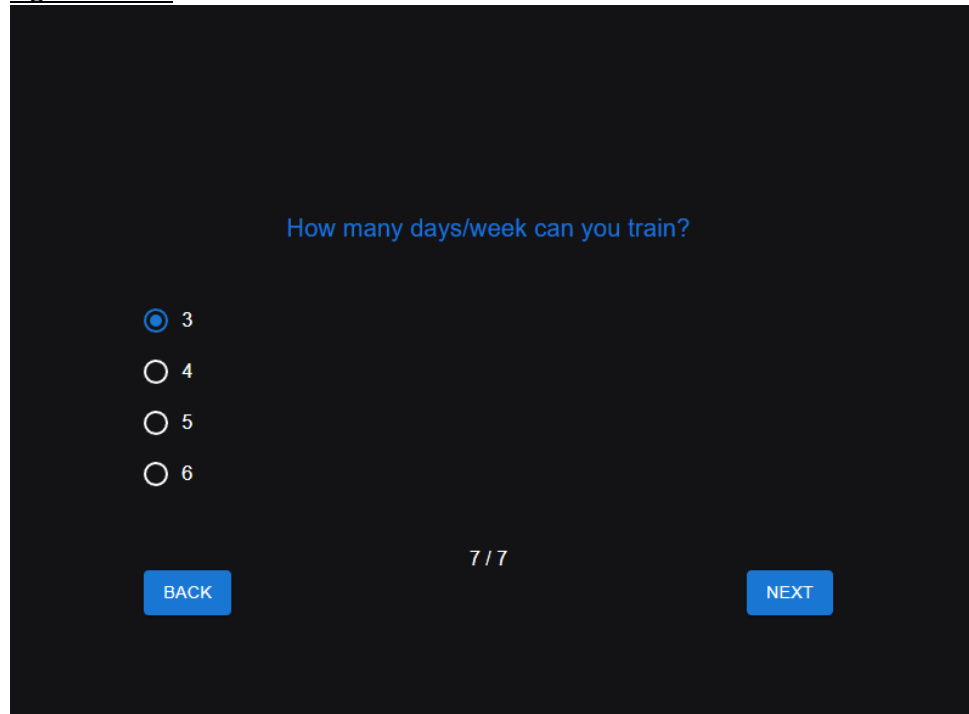
1. **Home Button** – it takes the user back to the home page.
2. **View Workouts Button** – it allows the user to see a dropdown menu of workouts they have available. Each workout in the dropdown menu is its own button. The user may click the workout to go to that page. On all the workout buttons there is a delete workout button. This delete button allows the user to delete the workout from the dropdown menu and their account.
3. **Sign Out** – allows the user to sign out of their account. They are redirected back to the login page.

- **GUI Design Standards**

The navbar is fixed at the top of the home and workouts pages. This ensures access to navigation regardless of where you've scrolled onto the page. The design elements of the navbar are consistent on both pages (black background with blue button). The nav bar was created using HTML and CSS and imported buttons from MUI.

Quiz Interface

Figure 3.1.1.4



How many days/week can you train?

☒ 3

☐ 4

☐ 5

☐ 6

7 / 7

BACK NEXT


- **Purpose**
The purpose of the quiz interface is to allow users to receive a workout based on their preferences and information from the quiz. It stores the answer of each question in the quiz in an array which it uses to pass to the back end.
- **Components**
 1. **Quiz Radio Buttons** – each question on the quiz has its own corresponding radio button. This button is used to select the answer the user wants to select.
 2. **Back Button** – used to go back to the last question in the quiz. It is disabled when the user is on the first question.
 3. **Next Button** – used to go to the next question in the quiz. When the user is on the last question of the quiz, the next button acts as a submit button where it sends the quiz information to the back end. For every question, the button is disabled until the user selects an answer.
- **GUI Design Standards**
All the buttons were sourced from MUI. It was designed with HTML/CSS.. The design of the quiz integrates with the rest of the application featuring a black background with blue buttons. The design was implemented to be as simple, user and error friendly as possible.

Workouts Interface

Figure 3.1.1.5

POWERLIFTING

6X PER WEEK



DAY 1: HEAVY SQUAT

Exercise	Sets	Reps	Weight
Squat	5	3	
Pause Squat	3	4	
Leg Press	4	8	
Hanging Leg Raise	3	12	

DAY 2: HEAVY BENCH

Exercise	Sets	Reps	Weight
Bench Press	5	3	
Close-Grip Bench Press	3	5	
Dumbbell Row	4	10 (per arm)	
Face Pulls	3	12	

- **Purpose**
The purpose of the workouts interface is to allow the user to view the exercises in their workouts (which are tailored based on their answers to the quiz) and input the weights they've used so they can track progression.
- **Components**
 1. **Table** – allows the user to view each day's workout in a clean and easy to read way.
 2. **Weight Input Field** – allows the user to enter in the weights they are currently using for each exercise in order to track their progression.
- **GUI Design Standards**
The design consists of white, blue and gray tables on top of a black background. The table was sourced from MUI and styled with CSS. The tables are aligned with the middle of the page. The design is again implemented to be simple, minimalistic and user-friendly.

3.1.2 Software Interfaces

The workout Logger web application is a multi component system designed with modularity in mind to ensure seamless communication between components and efficient data handling. Communication between the front-end and back-end is handled through REST APIs, enabling structured interaction.

Communication

To facilitate communications from front-end to back-end, Flask creates REST APIs and the front-end sends HTTP POST, GET, or DELETE request that contains user data, to execute the following functions

- Login
- Create Account
- Generate workout based on quiz results
- Delete Workout
- Get saved workouts
- load saved workout and weights
- Save logged weights

See figure 5.1.1.1 for API endpoints

In the back-end, the library **pymongo** is used to interact with the MongoDB database to perform these functions and store user specific data.

Tools and Libraries

Several tools and libraries are used. First, the front-end used React.js for rendering components, along with Material-UI for creating consistent user friendly graphical interfaces. In the back-end, Flask is used to handle server-side logic, REST API creation, and bridging the communication between front-end and the database. **Bcrypt** is used for hashing sensitive user information before storing it into the database to increase security of our end users. As mentioned, Pymongo is used to facilitate connecting and interacting with the MongoDB database.

System Requirements:

System requirements for our app are very lax, since it is a web application with little resource dependency. The front-end requires a modern browser that supports JavaScript, to run React components. The back-end requires Python 3, along with the libraries mentioned, to execute server-based logic. Ideally, the back-end would run on a UNIX-based system to increase stability, however, all operating systems will be supported. Finally, MongoDB is used for persistent data storage, and is currently hosted using MongoDB Atlas for remote connection.

Nature of Communication:

Communication between front-end and back-end is structured through HTTP requests with JSON-formatted payloads. Using JSON lists allows seamless and structured communication that both subsystems can understand. On either end, front-end or back-end, that data is then extracted from the JSON payload, and either loaded to the front-end to display information, or stored in MongoDB.

3.2 Product Functional Requirements

3.2.1 Functional Requirement # 1: Account Management

Description: The system must provide functionalities for users to create an account, login and manage their information.

Operations:

- Sign up: Users must be able to create a new account by providing their email, password and name.
- Login: Users must be able to login into the application using the email and password they signed up with.
- Error Handling: The system must return appropriate messages if the user signs up with an already in use email or tries to login with incorrect information.

Functional Areas:

- Create Account
 - Collect user data including email, password and name
 - Validate the user input for uniqueness in MongoDB
 - Store the user information in the DB
- User Authentication
 - Verify user input
 - Allow successful login or provide error message if login invalid

3.2.2 Functional Requirement # 2: Quiz and Workout RoutineGeneration

Description: The system must enable users to take the quiz to generate a routine that fits their goals and time and experience.

Operations:

- Take Quiz: Users answer multiple questions related to their experience, availability and goals
- Generate Routine: Based on the users answers, a suggested routine is generated and saved for that user
- Save Workout Routine: Save the generated routine to the DB with the users ID for future reference.

Functional Areas:

- Quiz Handling:
 - Accept user responses from the front end
 - Determine the users fitness profile using the quiz responses
- Workout Routine Generation:
 - Based on the quiz results, generate a routine type (e.g. bodybuilding, ppl).
 - Store workout plan information including user ID and associated exercises in MongoDB

3.2.3 Functional Requirement # 3: Workout Tracking and Weight Management

Description: The system must provide users the ability to track their workout progress by updating the weights used in their exercises

Operations:

- View Workout Routine: Users must be able to view their respective routine with exercise names and details such as sets, reps and weights used in the exercise
- Update Weights: Users can update the weight they can use for each exercise, and this updated information must be saved to the database

Functional Areas:

- Workout Routine Retrieval
 - Retrieve and display the saved workout routine to the user.
- Weights Update Functionality
 - Allow users to update weights for exercises where its needed.
 - Store the updated weights associated with the user's routine and exercise in MongoDB.

3.3 Product Non-Functional Requirements

3.3.1 Performance Requirements

The following performance requirements ensure that the application operates efficiently.

1. Registration and Login

The system must process user registration and login requests within a short time in order for the user to gain quick access to which enhances user satisfaction and improves user retention.

2. Custom Workout Generation

The system must generate a workout based on quiz results within a short time. Users expect immediate results

3. Quiz

The system must load a personalized workout plan based on the quiz results. Quiz results are bundled into a single request before being sent to MongoDB to optimize database performance. Users expect immediate results when setting up their workouts.

4. Login Weights

The system must record user-submitted weights and workout data in the backend. A delimiter is used when sending weights to the MongoDB to ensure all data is sent in one chunk. This avoids MongoDB performance issues.

5. Backend and API

All database operations must be completed within a very short time so that all other functionalities can also be completed within a very short time. A delimiter was added on the data being sent to the back end so that the data could be sent in one chunk instead of letter-by-letter. This allows MongoDB a chance to keep up, ensuring that it can handle multiple simultaneous requests.

4 System Solution

The Workout Logger project provides a solution to address the need for personalized fitness plans and progress tracking. The system combines the use of front end back end and database components to create a seamless experience for users of the application.

The solution is designed as a full web-based application that assigns the user a workout plan based on their quiz inputs. The solution uses modern technologies, such as React for the front end, Flask for the back end, and MongoDB for data storage.

The front-end, which uses React and Material-UI (MUI), offers a simple and visually appealing user interface. The key components of the front end such as the quiz, workout logging tables, and the navigation bar ensure smooth interactions between the application and the user. The design caters to both fitness enthusiasts and complete beginners.

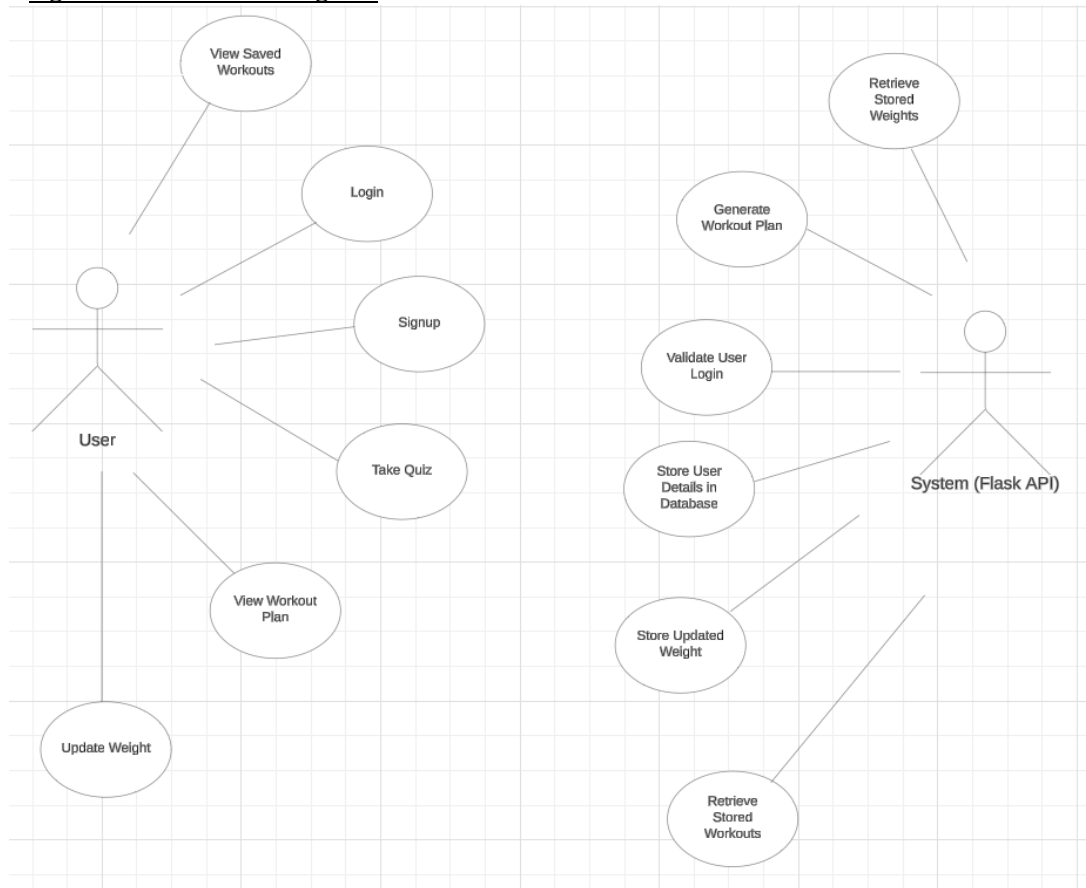
The back-end, which uses Flask, acts as the brain of the system. It is the bridge between the front-end and MongoDB. Using REST APIs, the back-end can handle functionalities such as user authentication, quiz results, and workout weights tracking. MongoDB is the database in the system. It provides a flexible and scalable database that is used for managing user information. The system ensures that users can quickly retrieve their workout plans and progress on exercises.

The connection between the front-end, back-end, and database allows the Workout Logger to be responsive even when it is under heavy usage. The system presents a solution to the unfortunate challenges of inconsistent fitness habits. It offers each user a personalized platform which they can use to get motivated and stay consistent and committed to their fitness goals.

5 Product Design

5.1 System Architecture

■ Figure 5.1: Use Case Diagram



The system is structured into two main subsystems: the User Subsystem and the System Subsystem (Flask API). Each subsystem is assigned specific roles and responsibilities that collectively enable the functionality of the overall product.

User Subsystem

The user subsystem represents the actions and interactions of the user with the system.

- Logging in
- Signing up, using a new account
- Taking quizzes to generate a workout plan
- Viewing generated and saved workout plans
- Updating specific weights for exercise

The user subsystem initiates requests to the other main subsystem (Flask API) to perform these actions. Its role as a primary external actor interacts with the system to provide inputs and receive the corresponding outputs.

System (Flask API) Subsystem

Core driver of the system, responsible for processing user requests and managing the logic behind the requested operations.

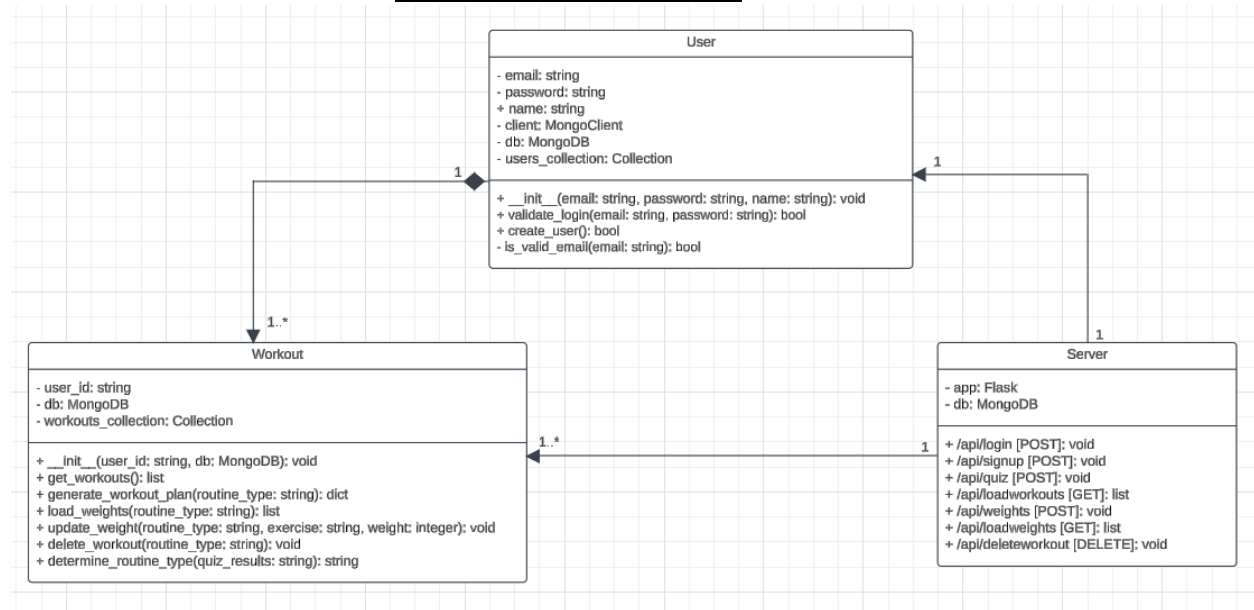
- Validating user logins
- Storing user details in the database
- Generating workout plans
- Storing and retrieving data

The system subsystem acts as the bridge between the user and the data stored within the database. Processes user requests utilizing the proper back-end logic via user and workout classes.

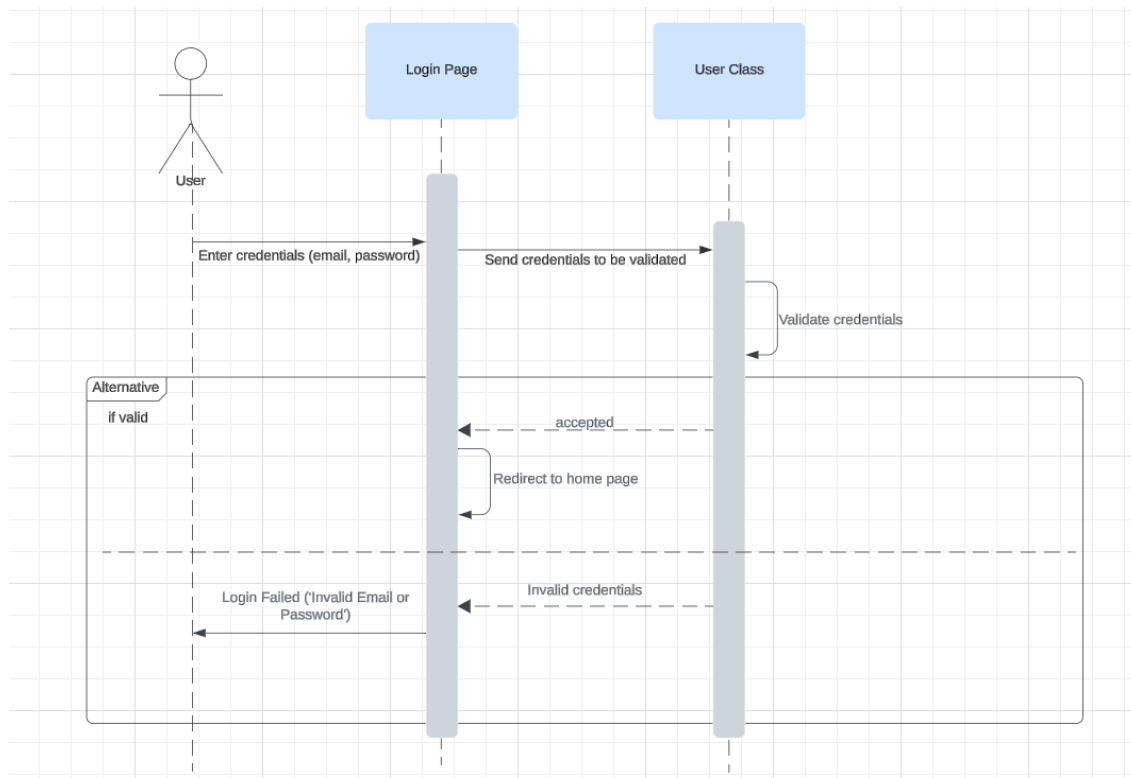
The User Subsystem acts as the primary external actor, initiating requests such as logging in, signing up, or interacting with workout plans. These requests are handled by the System Subsystem (Flask API), which acts as the central driver. The Flask API processes user input, validates it, and manages tasks to the appropriate classes (User and Workout) for execution. The System Subsystem then interacts with the Database (MongoDB) to store or retrieve user data, such as account details, workout plans, and progress updates. Once the operations are completed, the System Subsystem formats and sends a response back to the User Subsystem, which ensures all features remain functional.

5.1.1 Decomposition\Sub System Description

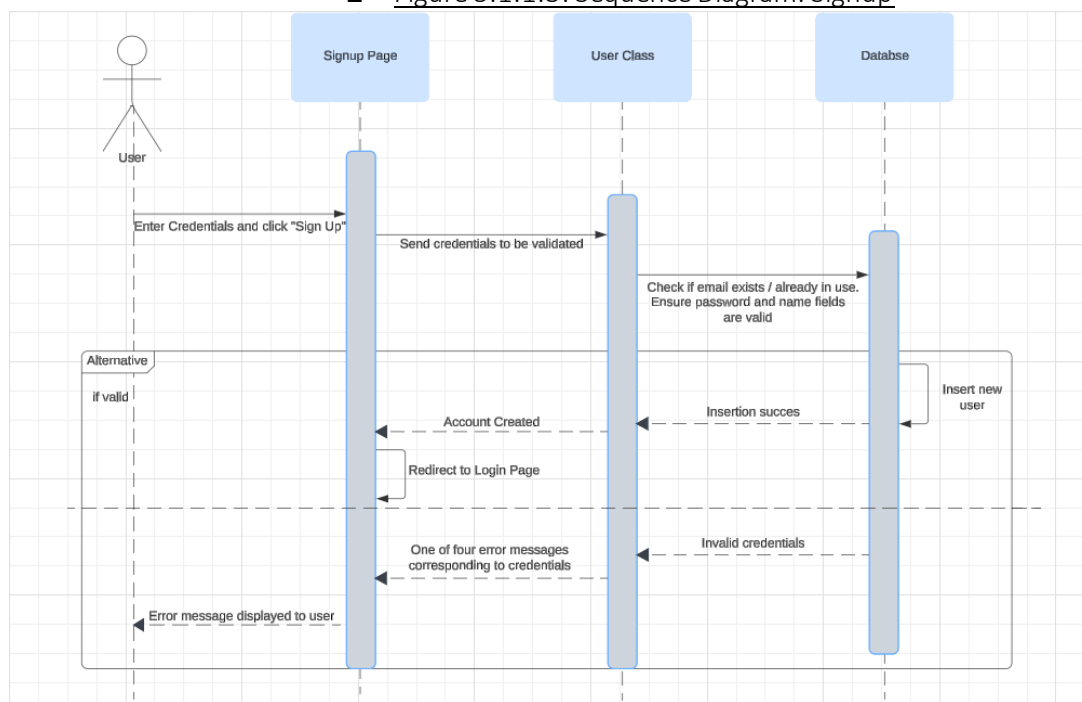
■ Figure 5.1.1.1: UML Diagram



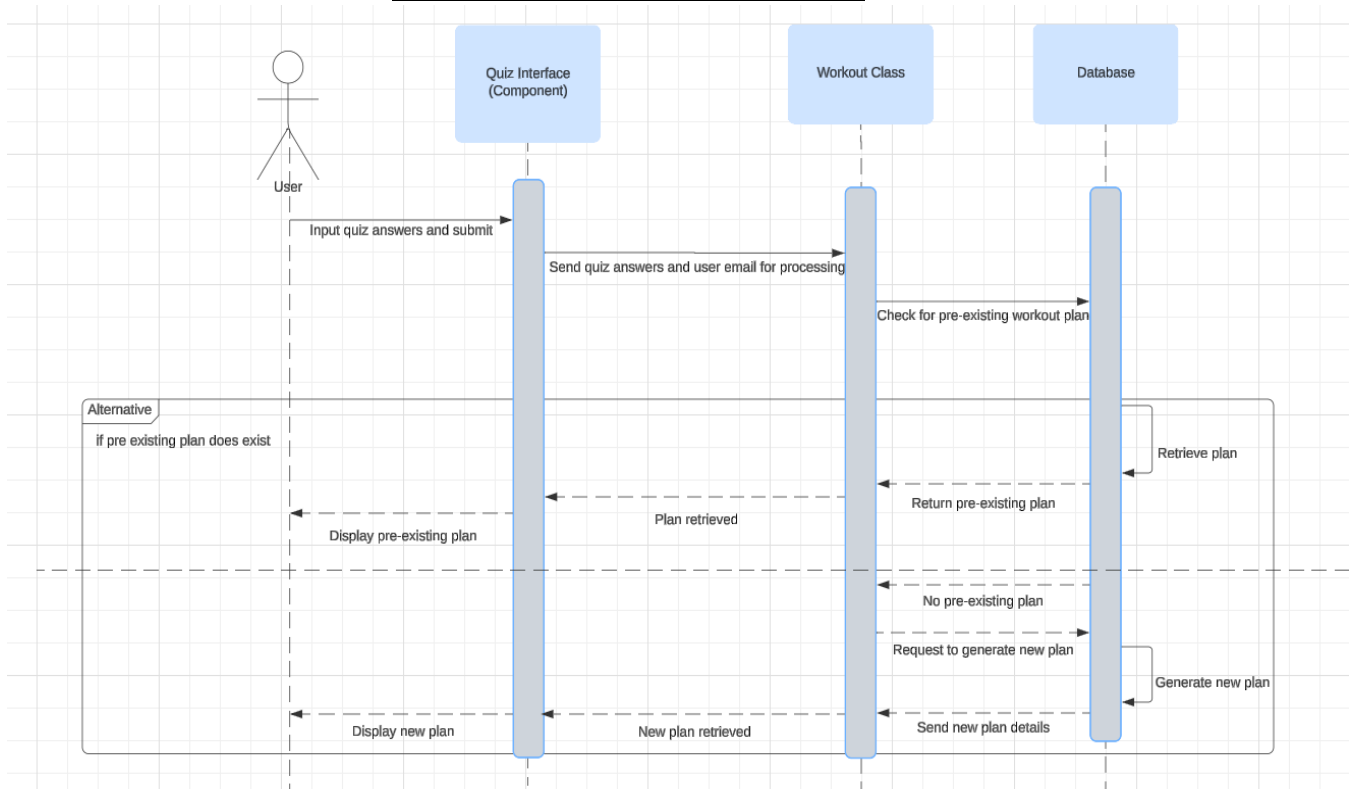
■ Figure 5.1.1.2: Sequence Diagram: Login



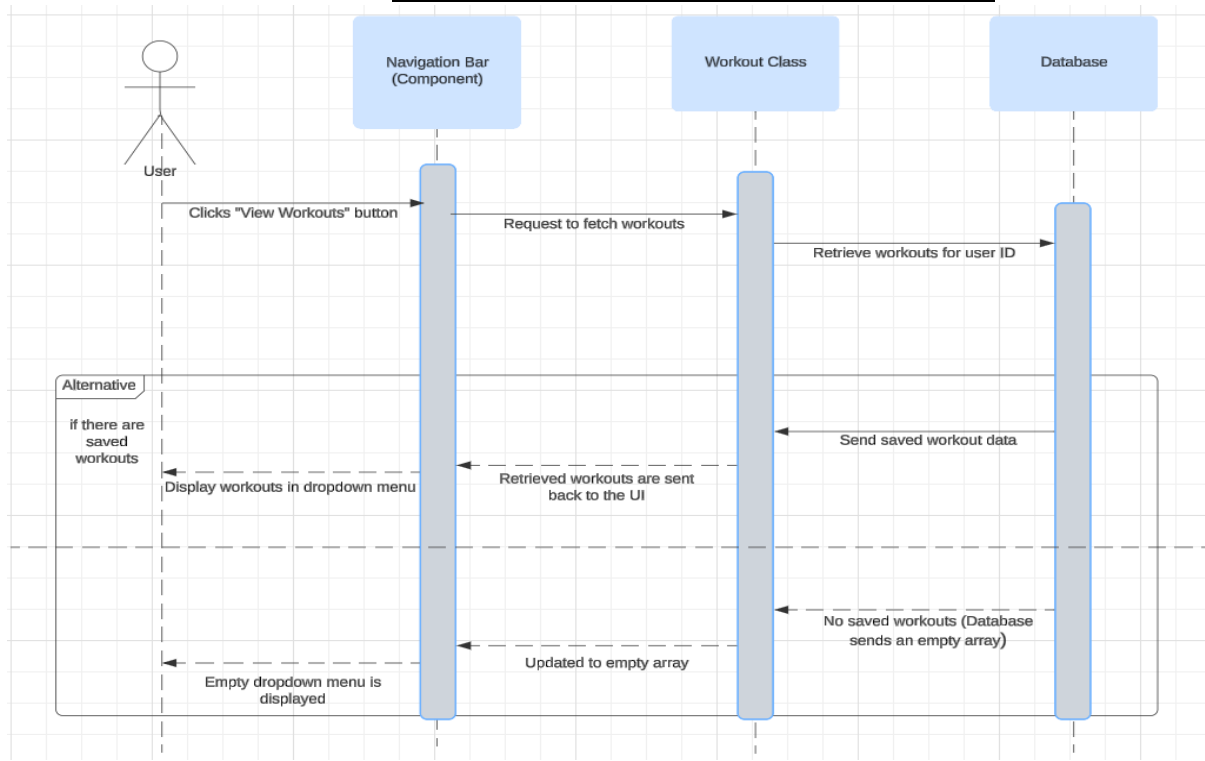
■ Figure 5.1.1.3: Sequence Diagram: Signup



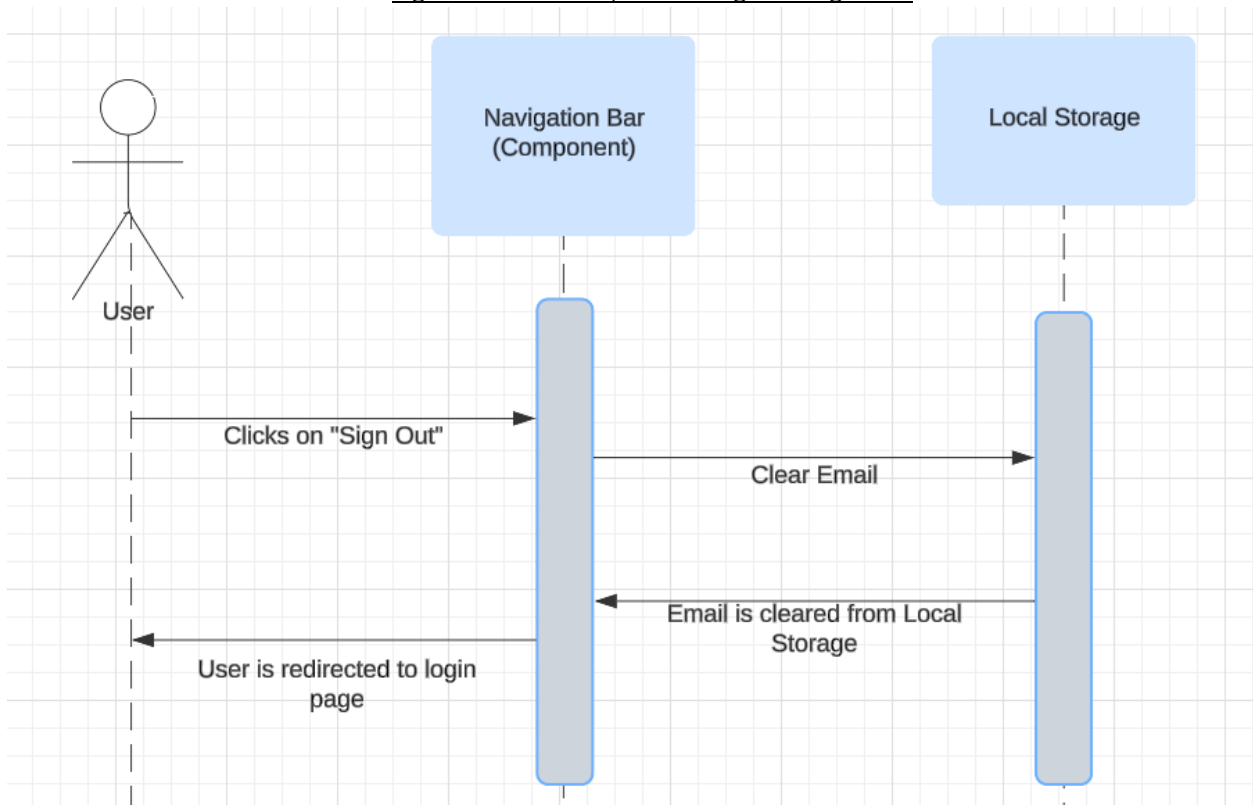
■ Figure 5.1.1.4: Sequence Diagram: Quiz



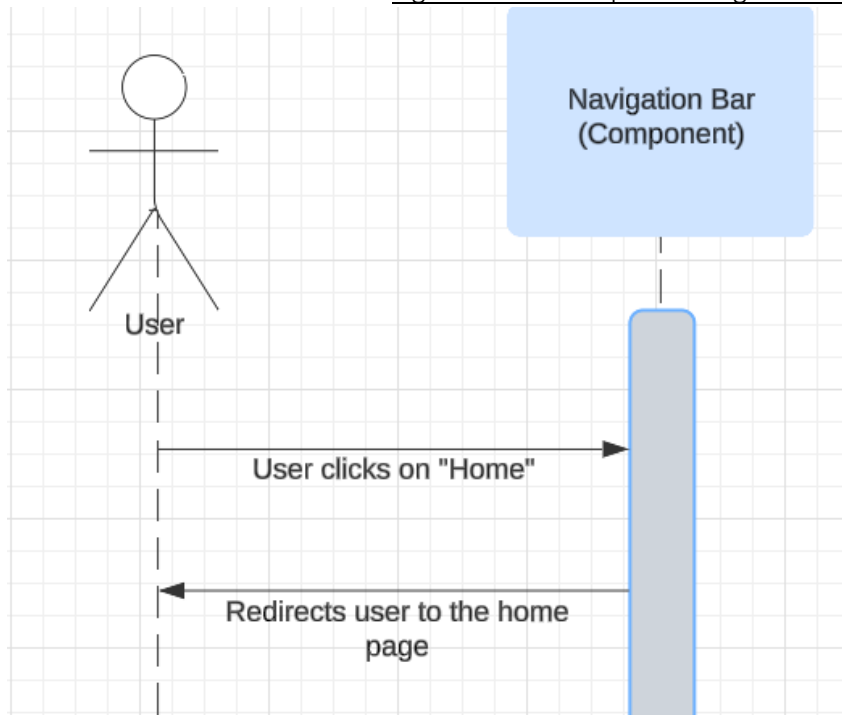
■ Figure 5.1.1.5: Sequence Diagram: View Workouts



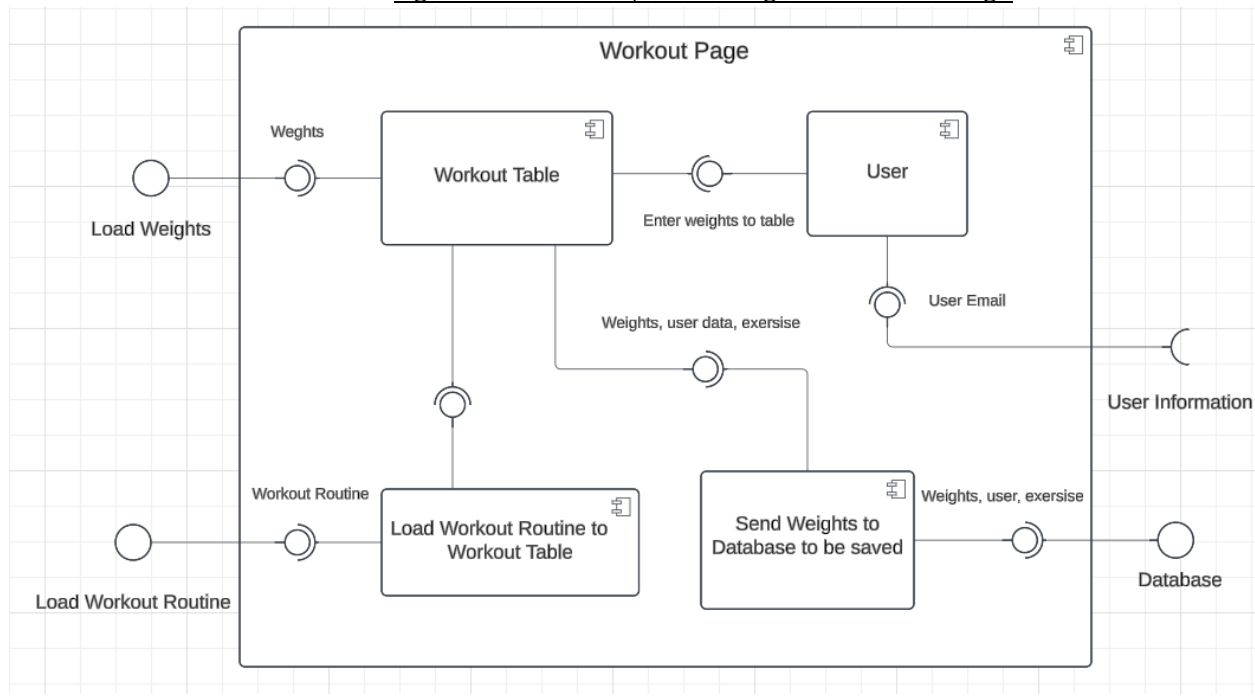
■ Figure 5.1.1.6: Sequence Diagram: Sign Out



■ Figure 5.1.1.7: Sequence Diagram: Home Button



■ Figure 5.1.1.8: Component Diagram: Workout Page



Interface Specifications

Navigation Bar Component

The navigation bar acts as the primary gateway for user interactions with the system. Users can navigate to the home page, view saved workouts, or sign out. Upon clicking "View Workouts," the navigation bar fetches saved workouts by sending a GET request to `/api/loadworkouts` with the user's email from local storage. If workouts exist, they are displayed in a dropdown menu. If none are found, an empty dropdown is displayed. Clicking "Sign Out" clears the email from local storage and redirects the user to the login page. Errors during these actions are logged to the console, and missing user emails prompt a redirection to the login page.

Workout Table Component

The workout table displays the details of a selected workout routine, including exercises, sets, reps, and weights. Upon loading, the component fetches saved weights for the selected routine from `/api/loadweights`. Users can update weights directly in the table, which are sent to `/api/weights` using a POST request. The table uses a loading state until data is retrieved and defaults to empty fields for weights if no data is available. Errors during API calls are logged to the console.

Quiz Interface Component

The quiz interface generates personalized workout plans based on user responses. As users progress through the quiz, selected answers are saved in the component's state. Upon submission, the quiz sends the responses and user email to `/api/quiz` to process and generate a workout plan. If successful, the user

is redirected to the generated workout plan page. Missing user emails redirect users to the login page, and submission errors are logged to the console.

The Navigation Bar Component, Workout Table Component, and Quiz Interface Component each serve distinct roles in interacting with the backend, resulting in a seamless user experience. The Navigation Bar acts as a central hub, and the quiz and workout table component are the driving features of the user interface.

6 Conclusion and Future Work

This Workout Logger project has successfully addressed the challenge of providing a structured and user-friendly platform for people looking to improve their health and wellness by delivering personalized workout routines, tracking their progress, and staying motivated. Through the usage of React for the frontend, Flask for the backend, and MongoDB for database management, we achieved the following objectives while satisfying the constraints:

- **Data Integrity:** The system emphasizes data validation at multiple levels. For instance, invalid login credentials or duplicate email addresses are caught early by the server, ensuring that only accurate and complete data is stored in MongoDB.
- **Usability:** The frontend, built with React and Material-UI, focuses on an intuitive user experience. Features such as progress tracking and workout updates, allows users to update their workout details, including weights and routines, and retrieve stored data to monitor their progress over time. Sequence diagrams and specifications ensured that the design aligned with expected user interactions
- **Personalized Workout Generation:** A dynamic quiz delivers workout routines tailored to user preferences and goals, making the platform more engaging and effective.
- **Workout Tracking:** The application allows users to log their workout progress by updating weights and viewing saved workout routines while ensuring consistent data storage and retrieval in the database.

While the project is a solid foundation for a personalized fitness platform, there are some current limitations that present room for future improvement. One of the key areas for improvement in this project is expanding the workout database to offer more personalized and diverse workout plans. Currently, the database is limited in which it restricts the ability to fully tailor plans to meet users' unique goals, preferences, and fitness levels. By using a broader range of exercises, routines, and training methodologies, such as yoga, pilates, HIIT, or sport-specific training—the platform could cater to a wider audience.

7 References

- [1] S. C. Government of Canada, “An overview of weight and height measurements on World Obesity Day,” *www.statcan.gc.ca*, Mar. 04, 2024. <https://www.statcan.gc.ca/o1/en/plus/5742-overview-weight-and-height-measurements-world-obesity-day>
- [2] L. Mandolesi *et al.*, “Effects of physical exercise on cognitive functioning and wellbeing: Biological and psychological benefits,” *Frontiers in Psychology*, vol. 9, no. 9, Apr. 2018, doi: <https://doi.org/10.3389/fpsyg.2018.00509>.
- [3] H. K. Larson, K. McFadden, T.-L. F. McHugh, T. R. Berry, and W. M. Rodgers, “When you don’t get what you want—and it’s really hard: Exploring motivational contributions to exercise dropout,” *Psychology of Sport and Exercise*, vol. 37, pp. 59–66, Jul. 2018, doi: <https://doi.org/10.1016/j.psychsport.2018.04.006>.
- [4] T. Davergne, P. Meidinger, A. Dechertres, and L. Gossec, “The effectiveness of digital applications providing personalized exercise videos: a systematic review with meta-analysis (Preprint),” *Journal of Medical Internet Research*, vol. 25, Dec. 2022, doi: <https://doi.org/10.2196/45207>.