Software Engineering
Software Requirements Specification (SRS) Document

Fitness Gym-Bud

https://github.com/bilalv14/Gym-Bud

09/23/2023

[Final]

By: Jacob Crews, Rohan Waheed, Bilal Zahid

[Honor Code]

Table of Contents

1. Introduction

1.1. Purpose

The goal of the Fitness Gym-Bud is to help connect not only the average person to a personalized trainer but also make it easy for the gym itself to offer its services to its members while also making it easy for the trainers themselves to connect with their clients and offer better training and dietary suggestions to make actual progress with clients further increasing the trainer's reputation and the gym's influx of members.

1.2. Document Conventions

The purpose of this Software Requirements Document (SRD) is to describe the requirements for the developers and the clients for the Fitness Gym-Bud app. In it, we will describe the application from the perspective of each type of user the app aims to service, and describe the application from a developers perspective, including descriptions of data, performance and other elements.

1.3. Definitions, Acronyms, and Abbreviations

[Include any specialized terminology dictated by the application area or the product area. For example:]

Java	A programming language originally developed by James Gosling at Sun Microsystems. We will be using this language to build the Restaurant Manager.
MySQL	Open-source relational database management system.
.HTML	Hypertext Markup Language. This is the code that will be used to structure and design the web application and its content.
SpringBoot	An open-source Java-based framework used to create a micro Service. This will be used to create and run our application.
MVC	Model-View-Controller. This is the architectural pattern that will be used to implement our system.
Spring Web	Will be used to build our web application by using Spring MVC. This is one of the dependencies of our system.
Thymeleaf	A modern server-side Java template engine for our web environment. This is one of the dependencies of our system.
NetBeans	An integrated development environment (IDE) for Java. This is where our system will be created.

API	Application Programming Interface. This will be used to implement a function within the software where the current date and time is displayed on the homepage.
PR	Personal record. A gym member can log their personal records on lifts.

1.4. Intended Audience

[Describe which part of the SRS document is intended for which reader. Include a list of all stakeholders of the project, developers, project managers, and users for better clarity.]

1.5. Project Scope

The goal of the application is to connect people with gyms and personal trainers through an easy-to-use app to help gyms reach more people. This aligns with the overall goal of the gyms that use this app, as it will allow the gym to grow, and better provide their services.

The benefits of this app to Gyms include:

- Increase in trainer and client connection
- Physical hassle of creating workouts and meal plans removed
- Better understanding of calories and food intake
- Easy method of setting availability for trainers

1.6. Technology Challenges

We will be using software that we are not very familiar with

1.7. References

2. General Description

2.1. Product Perspective

Fitness Gym-Bud was founded with the idea of making it easier for regular people to connect with a gym and a personal trainer. From personal experience from the founders, staying on track to meet fitness goals can be hard, this app aims to keep is clients motivated with easy access to a gym and a trainer.

2.2. Product Features

The product features include the ability for a gym admin, trainer, and gym member to create accounts with different features. The gym admin has the ability to assign trainers to clients, remove or add the gym to the app, pay trainers, and remove clients, The trainer has the ability to create a meal plan, create a workout plan, set availability, and set food/workout reminders for their clients. The client/gym member has the ability to pay gym fees, request a new trainer/remove a trainer, leave the gym, and also create a workout/meal plan.

2.3. User Class and Characteristics

Our website application only expects users to know how to use a web browser. Trainers will be expected to be knowledgeable on fitness and healthy eating, and the gym admin will be expected to know how to manage multiple trainers and clients. The clients will need no prior knowledge in anyway to be successful with this app.

2.4. Operating Environment

The application is designed to operate on the web across many different devices.

2.5. Constraints

We are learning to use much of what is required in the project on the go, limiting our time to work on certain parts of the project

2.6. Assumptions and Dependencies

The software will be dependent on Spring Web and Thymeleaf in order to create and execute the MVC architecture that will be developed within NetBeans. The application will also use the My Fitness Pal API (http://myfitnesspalapi.com/) to http://myfitnesspalapi.com/) to allow users to track their nutrition in the foods they eat throughout the day.

3. Functional Requirements

[Statements of services the system should provide, how the system should react to particular inputs and how the system should behave in particular situations.]

The system first has a login feature and sign up feature for what type of user you are. For example if a gym member tried logging in as a trainer it would not work. The system also responds to food and workouts the trainer or client inputs. The gym admin also has the ability to remove gym members and pay trainers while clients/ members can pay their fees and add/remove a trainer.

3.1. Primary

Admins can add new trainers and clients to the system.

3.2. Secondary

Members have their own space in the app where they can see and change their own gym schedules and trainer appointments. We will use a food api because it allows for easier input.

4. Technical Requirements

4.1. Operating System and Compatibility

The application will be compatible with any operating system that is able to view and to interact with traditional web pages.

4.2. Interface Requirements

4.2.1. User Interfaces

Every page will feature a footer with the name of the app and the year it was established, and a header that will display different text based on the page the user is on.

4.2.2. Hardware Interfaces

The web application will run on any hardware device that has access to the internet, the ability to display webpages, and the ability to interact with web pages. This includes, but is not limited to, smartphones, tablets, desktop computers, and laptops.

4.2.3. Communications Interfaces

[Determination of all the communication standards to be utilized by the software as a part of the project]

4.2.4. Software Interfaces

The front end was built with html and css, the backend was made with java. They were connected by springboot web services. Data is stored in an sql database

5. Non-Functional Requirements

5.1. Performance Requirements

The project must store data to an outside source and be able to access and change this data. The project must also have a well layed out ui.

5.2. Safety Requirements

List out any safeguards that need to be incorporated as a measure against any possible harm the use of the software application may cause.

5.3. Security Requirements

The project must have a way of keeping the different actors from being able to access the other's pages.

5.4. Software Quality Attributes

5.4.1. Availability

There is an easy sign-up process for new users, and the app can be accessed at any time.

5.4.2. Correctness

The developers will take extra precautions to ensure there are no bugs and everything available to the users works as intended

5.4.3. Maintainability

The code will be well documented so the developers can easily make changes or maintain the app

5.4.4. Reusability

The client, trainer, and gym admin will all be in constant contact with each other through the app, Giving all users reason to continue using the app

5.4.5. Portability

The app will be available to download on multiple devices, and can be used anytime the user has access to the internet.

5.5. Process Requirements

5.5.1. Development Process Used

Software Process Model

5.5.2. Time Constraints

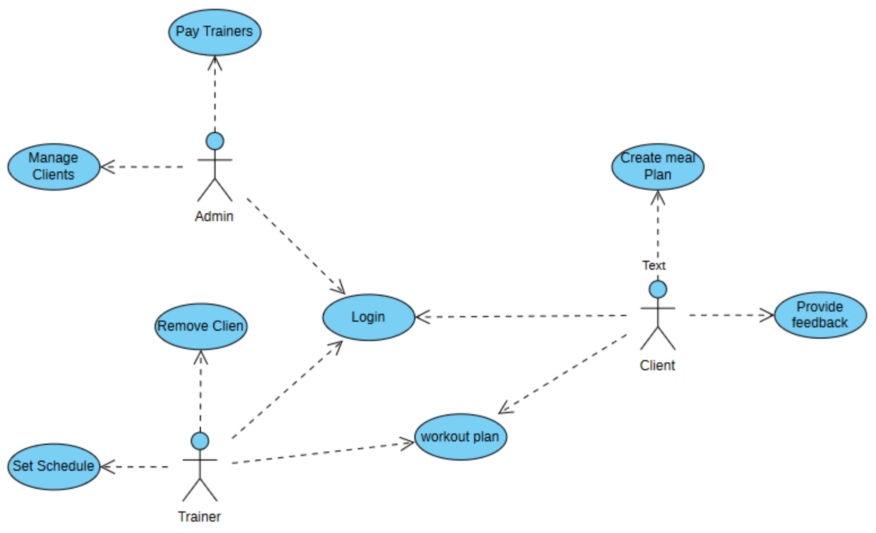
Due December 5th

5.5.3. Cost and Delivery Date

Due December 5th

5.6. Other Requirements

5.7. Use-Case Model Diagram



5.8. Use-Case Model Descriptions

5.8.1. Actor: Bilal Zahid

- **Client/Member**: Muhammed Zahid is responsible of Client (pay gym fee, request new trainer, leave gym, create meal plan, create workout plan, login).

5.8.2. Actor: Rohan Waheed

- **Use-Case Name**: Khawaja waheed is responsible off Gym Admin(Manage Clients, pay Trainers)

5.8.3. Actor: Jacob Crews

- **Use-Case Name**: Jacob Crews is responsible for Trainer (remove client, login, create meal plan, create workout plan, set availability

5.9. Use-Case Model Scenarios

5.9.1. Actor: Bilal Zahid

- **Use-Case Name**: client/ gym member
  - **Initial Assumption**: Needs to be able to work in tandem with trainer.
  - **Normal**:
  - **What Can Go Wrong**: Conflict between trainer food and workout logs and client ● **Other Activities**: login and log off feature ● **System State on Completion**:
- **Use-Case Name**:
  - **Initial Assumption**:
  - **Normal**:
  - **What Can Go Wrong**:
  - **Other Activities**:
  - **System State on Completion**:

5.9.2. Actor: Trainer (Jacob Crews)

- **Use-Case Name**: Create Workout Plan
    - **Initial Assumption**: Trainer creates workout plan for client.
    - **Normal**: Client agrees with and follows the plan.
    - **What Can Go Wrong**: Client disagrees with the plan, causing the trainer to accommodate the client and make adjustments.
    - **Other Activities**: The client can adjust the workout plan themselves and discuss the changes with the trainer.
    - **System State on Completion**: The trainer has created a workout plan that satisfies the client.
- **Use-Case Name**: Set avalibility
    - **Initial Assumption**: The Trainer sets their avalibilty for their clients to be able to reach them.
    - **Normal**: The trainers avalability
    - **What Can Go Wrong**: The Trainers avalability does not line up with the avalibility of a client. The trainer has the option to drop this client to reslove this issue for both of them.
    - **Other Activities**: The trainer and clients can communicate with each other to come to an arrangement that works for them both.
    - **System State on Completion**: The trainer has set their avalibility and it works with their clients.

5.9.3. Actor: Actor Name (Responsible Team Member) -

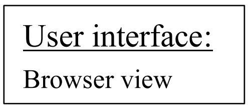
Use-Case Name:

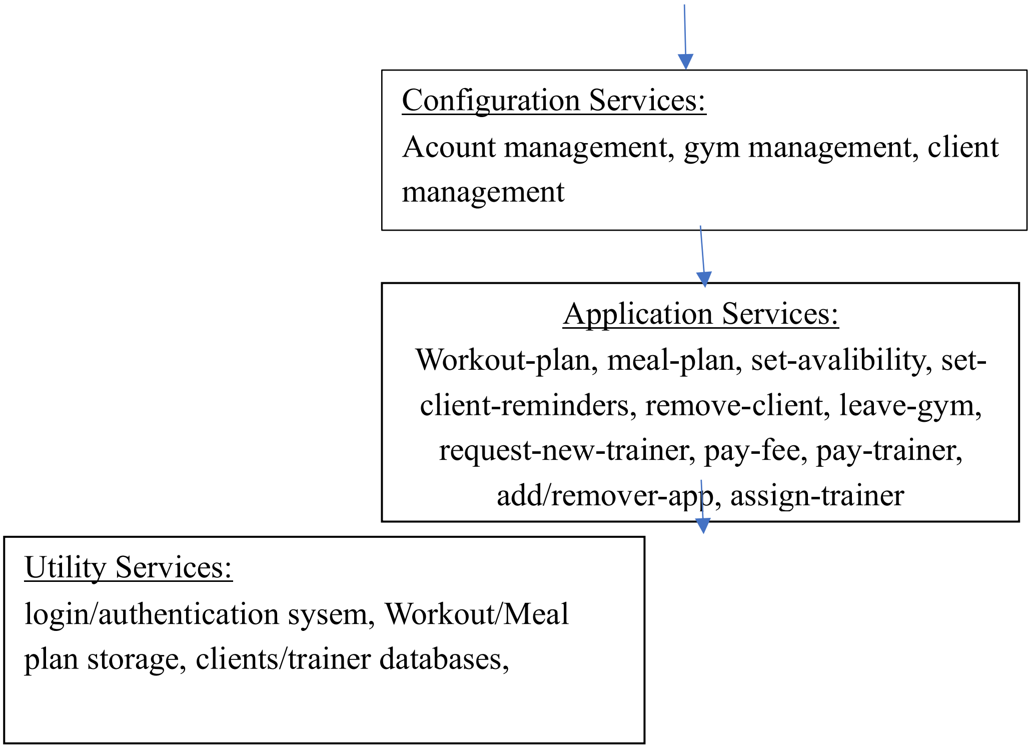
- **Initial Assumption**:
- **Normal**:
- **What Can Go Wrong**:
- **Other Activities**:
- **System State on Completion**:

- **Use-Case Name**:
    - **Initial Assumption**:
    - **Normal**:
    - **What Can Go Wrong**:
    - **Other Activities**:
    - **System State on Completion**:

6. Design Documents

6.1. Software Architecture


User interface:
Browser view



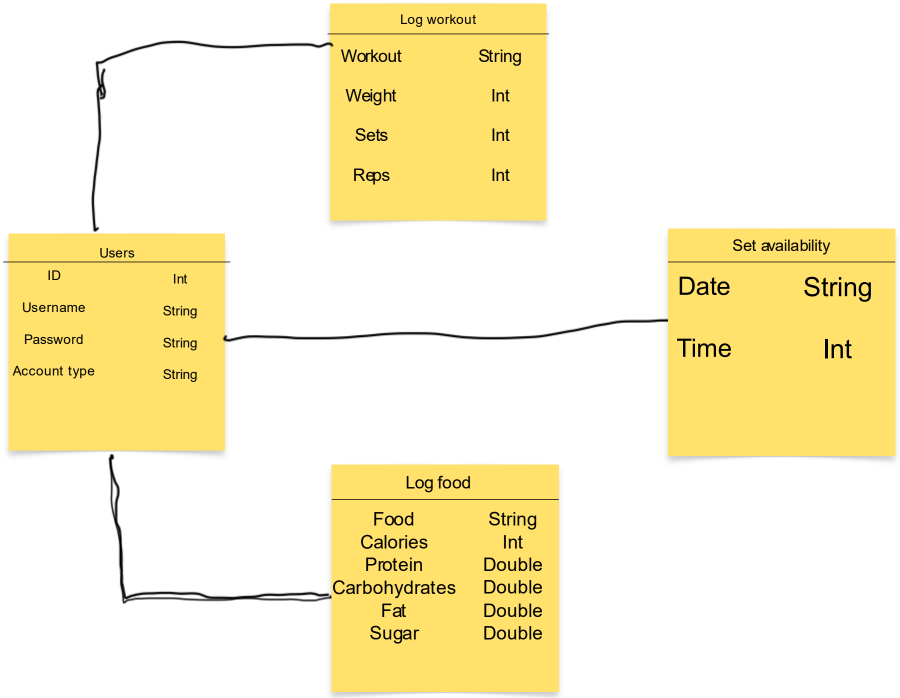
Configuration Services:
Acount management, gym management, client management

Application Services:
Workout-plan, meal-plan, set-avalibility, set-client-reminders, remove-client, leave-gym, request-new-trainer, pay-fee, pay-trainer, add/remover-app, assign-trainer

Utility Services:
login/authentication sysem, Workout/Meal plan storage, clients/trainer databases,

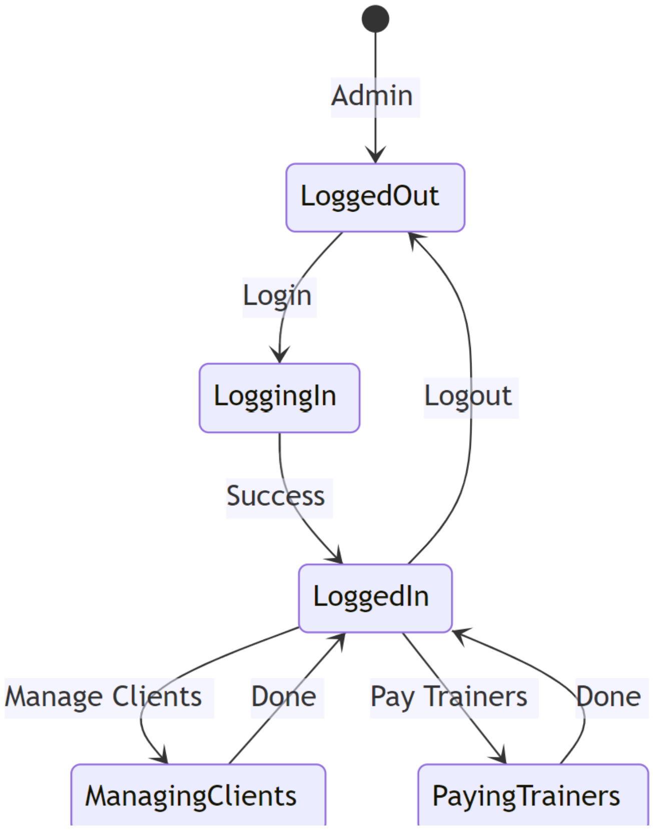
6.2. High-Level Database Schema

Fitness Gym Bud stores user account details, food logs, and workout logs in a data base.

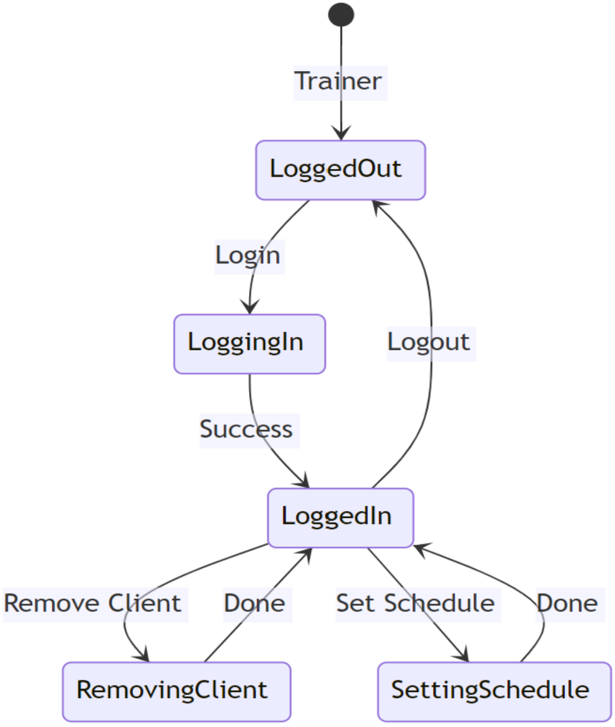


6.3. Software Design

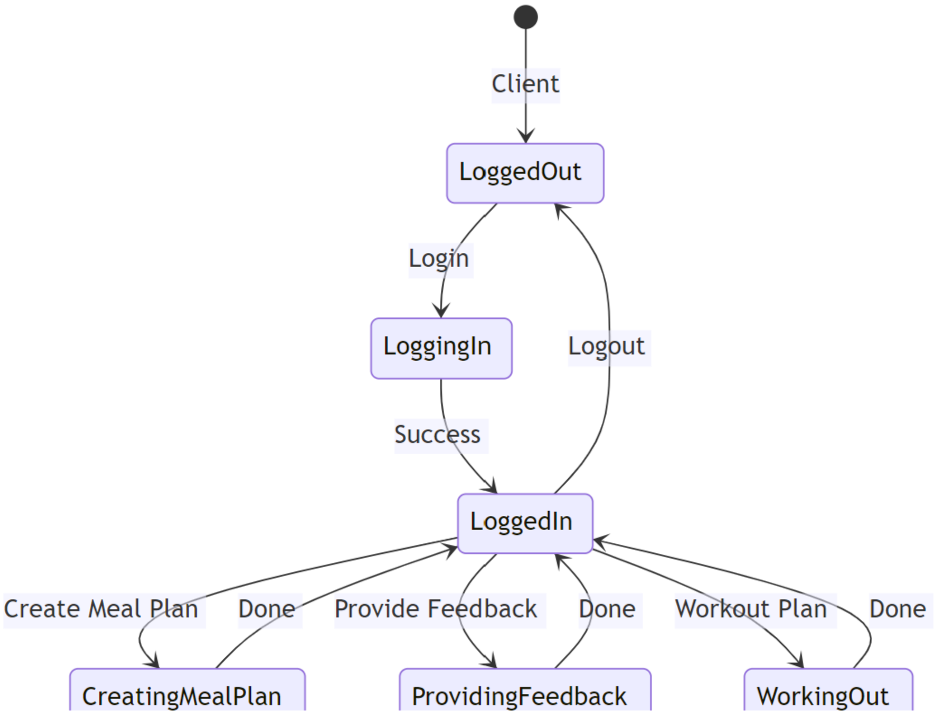
6.3.1. State Machine Diagram: GymAdmin (khawaja waheed)



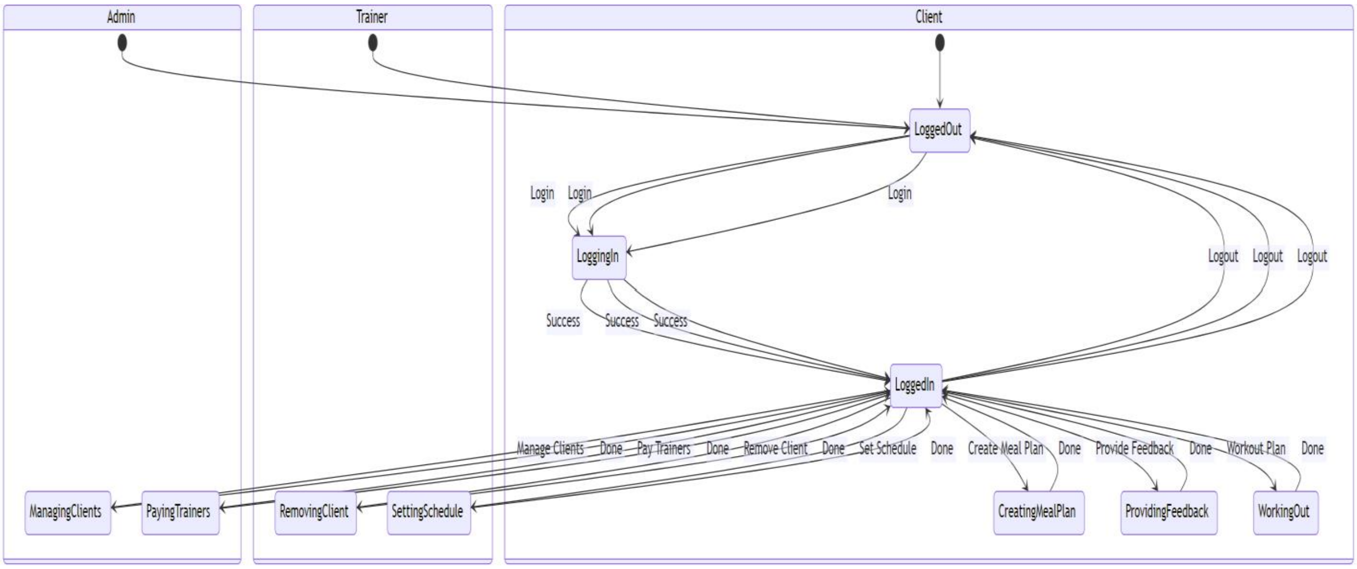
6.3.2. State Machine Diagram: Trainer (Jacob Crews)



6.3.3. State Machine Diagram: Client (Muhammad Zahid)



6.4. UML Class Diagram



7. Scenario

7.1. Brief Written Scenario with Screenshots

Actors:

Admin(Rohan): Manages the system, trainers, and clients.

Trainer(Jacob): Logs Workouts, Set Schedule, Remove client

Client(Bilal): Log workouts, Log Meals, and provides feedback.

Use Cases:

Admin:

Manage Users:

View System Statistics:

Trainer:

Create Workout Plans:

Manage Schedule:

Client:

Client logs in and logs workouts and meals

Client then views past workouts and meals

Client provides feedback to trainer

Scenario Flow:

Admin A1 logs in:
Trainer T1 logs in:
Creates workout plan for client
updates schedule to change availability for tuesday
Removes former client that left the gym
Client C1 logs in:
Logs workouts and logs meals
Checks previous meals and workouts
Leaves feedback to trainer
Data Storage and Retrieval:
All actions by A1, T1, and C1 are recorded in the system's database.
Each login, workout log, meal log, and feedback submission is stored and can be retrieved for review.
the trainers schedule is stored and can be viewed or updated later, trainer can also see list of clients and remove if needed

Implementation:

Admin Component: Allows the admin to manage users and view statistics.
Trainer Component: Enables trainers to create workout plans and manage schedules.
Client Component: Clients can log workout and view past workouts , log meals and view past meals, and provide feedback.
Backend Integration: application is properly communicating with the backend
State Management:
Front End: Sending post requests to backend