

Software Engineering
Software Requirements Specification
(SRS) Document

Progress

<https://github.com/James1056/Progress/tree/main>

02/14/2023

V3

By: Dallin Pierce, James Copestake, Devin Holcomb, Gary Li,

**I have abided by the UNCG Academic Integrity Policy on this
Assignment**

Table of Contents

1.	Introduction	4
1.1.	Purpose	4
1.2.	Document Conventions	4
1.3.	Definitions, Acronyms, and Abbreviations	4
1.4.	Intended Audience	5
1.5.	Project Scope	5
1.6.	Technology Challenges	6
1.7.	References	6
2.	General Description	6
2.1.	Product Perspective	6
2.2.	Product Features	6
2.3.	User Class and Characteristics	7
2.4.	Operating Environment	7
2.5.	Constraints	7
2.6.	Assumptions and Dependencies	7
3.	Functional Requirements	7
3.1.	Primary	7
3.2.	Secondary	7
4.	Technical Requirements	8
4.1.	Operating System and Compatibility	8
4.2.	Interface Requirements	8
4.2.1.	User Interfaces	8
4.2.2.	Hardware Interfaces	8
4.2.3.	Communications Interfaces	8
4.2.4.	Software Interfaces	9
5.	Non-Functional Requirements	9
5.1.	Performance Requirements	9

5.2.	Safety Requirements	9
5.3.	Security Requirements	9
5.4.	Software Quality Attributes	9
5.4.1.	Availability	9
5.4.2.	Correctness	9
5.4.3.	Maintainability	10
5.4.4.	Reusability	10
5.4.5.	Portability	10
5.5.	Process Requirements	7
5.5.1.	Development Process Used	7
5.5.2.	Time Constraints	7
5.5.3.	Cost and Delivery Date	7
5.6.	Other Requirements	10
5.7.	Use-Case Model Diagram	11
5.8.	Use-Case Model Descriptions	11
5.8.1.	Actor: Beginner User (Devin)	11
5.8.2.	Actor: Intermediate User (Devin)	11
5.8.3.	Actor: Advance User (James)	12
5.8.4.	Actor: Personal Trainer (Dallin)	12
5.8.5.	Actor: Gym Owner (Gary)	12
5.9.	Use-Case Model Scenarios	12
5.9.1.	Actor: Beginner User (Devin)	8
5.9.2.	Actor: Intermediate User (Devin)	9
5.9.3.	Actor: Advance User (James)	9
5.9.1.	Actor: Personal Trainer (Dallin)	8
5.9.1.	Actor: Gym Owner (Gary)	8
6.	Design Documents	18
6.1.	Software Architecture	18
6.2.	High-Level Database Schema	18

6.3. Software Design21

6.3.1. State Machine Diagram: Users (Devin/James)21

6.3.2. State Machine Diagram: Personal Trainer (Dallin)22

6.3.3. State Machine Diagram: Gym owner (Gary)23

6.4. UML Class Diagram24

7. Scenario25

7.1. Brief Written Scenario with Screenshots25

1. Introduction (James Copestake)

1.1.Purpose

The goal of Progress is to allow gym users of all abilities to have a community which is easily accessible that acts as a catalyst to help everyone to progress in their fitness journey. Users can talk to one another; ask whatever questions they may have and use the knowledge of others and what they do to adjust their routine to reach the goals that they have set themselves quicker and more efficiently.

Another big goal of Progress is to encourage beginners who may have been to the gym previously but been intimidated or people who have never been before to go to the gym and by having this software available it will show these users to speak to other people who go to the gym thus allowing them to have a easier time when they begin the gym as they have been able to ask questions and get to know people prior to going.

1.2.Document Conventions

The purpose of this Software Requirements Document (SRD) is to explain and portray the features and functions that will be within the gym social media platform “Progress”. In the application we aim to include the following features and functions:

- Private messaging between users of the platform
- Users creating different posts, be these questions, videos
- Users can react to other users posts and comment on them
- Certain users can publicize their services (for example personal training appointments)
- A level system for the users (beginner, intermediate, advanced) to distinguish what roles and features they have available to them on the application.

1.3.Definitions, Acronyms, and Abbreviations

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.

1.4.Intended Audience

Stakeholders

- Gym Users
- Gym Owners
- Personal Trainers
- UI/ UX Designers
- Software Engineers
- Project Manager

SRS Document and Readers

1. Introduction – UI/UX Designers, Software Engineers & Project Manager
2. General Description - UI/UX Designers, Software Engineers & Project Manager
3. Functional Requirements – UI/UX Designers & Software Engineers
4. Technical Requirements – UI/UX Designers & Software Engineers
5. Non-Functional Requirements - Software Engineers
6. Design Documents – Software Engineers
7. Scenario - UI/UX Designers & Software Engineers

1.5.Project Scope

The goal of the software is to provide a new environment to users that can use it as a source of information for everything related to fitness and personal development. The environment will contain an easy-to-use UI

with UX at the forefront and simple to use features. This aligns with the overall business goals of a social media platform as these platforms would fail if the purposes of the intended software are difficult to use with a confusing design therefore resulting in a rapid decline in users and no to minimal activity on the software.

The benefits of the project to business include:

- In helping alleviate the stress new gym goers face when thinking about starting the gym, this software should hopefully increase the intake gyms due to removing this fear.
- Not only will this software help people get into the gym but will also expose them to everything that goes hand in hand with the gym such as dieting, supplements and other physical activities, thus helping other industries by exposing these products to a wider audience.
- By promoting all of these industries on the software this then opens up the option for the software to take advantage of advertisements as an avenue of profiting from the platform.

1.6.Technology Challenges

For the posting aspect of the prospective software a user may refresh the page so any new posts that may have been posted in that time will have to be shown, so one technical requirement may be to be able to live update the application when its running to ensure that the user is able to view the most up to date content.

1.7.References

[Mention books, articles, web sites, worksheets, people who are sources of information about the application domain, etc. Use proper and complete reference notation. Give links to documents as appropriate. You should use the APA Documentation model (Alfred, 2003, p. 144).]

2. General Description (Dallin Pierce)

2.1.Product Perspective

-It's far easier to keep motivated about personal health and fitness when surrounded by a community of like-minded individuals. That's where Progress as a concept shines. By providing a platform for new and experienced users to ask questions and find help, it promotes greater health among all its users.

2.2.Product Features

-The product features include the ability for all users to ask questions, and for intermediate and advanced users to answer questions from users of lower levels than their own. All users will also have the ability to request a workout partner of a given proficiency level, or respond to a partner request. Intermediate users will be able to upload their workout splits and advanced users will be able to upload their meal plan. Specially marked accounts for certified personal trainers will be able to upload training videos and will

have specific support to advertise their services. Finally, gyms will be able to give updates and news, and they will have the ability to set dates for special classes and events occurring at the gym.

2.3.User Class and Characteristics

-While little technical knowledge will be required to operate our application, likely just the ability to operate a computer and run a web browser, the app does sort users by proficiency level. Users of any level of experience level should be supported on the platform, however certain features will be locked off till they prove a level of competency in health and fitness. Likewise, proof of certification, and proof of ownership will be required to be recognized as a personal trainer or gym owner, respectively.

2.4.Operating Environment

-As a web application, Progress should be able to run on a wide variety of devices, such as PC or Mobile.

2.5.Constraints

?

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. It will also be dependent on the YouTube Data API in order to upload and display training videos.

3. Functional Requirements (Gary Li)

3.1.Primary

- Facilitate communication and collaboration between users of different fitness levels in a gym environment.
- Provide a messaging system for private conversations between users.
- Provide public posts for sharing tips and progress.
- Offer social sharing features for achievements.
- Allow users to publicize personal training services.
- Provide different features based on the user's fitness level.
- Be designed to reduce the intimidation felt by new gym users and encourage engagement with fitness.

3.2.Secondary

- A secure login and user authentication system.
- A user profile page, displaying information about the user's fitness level and preferences.

- A dashboard page, displaying relevant information to the user, such as workout routines, goals, and progress tracking.
- A search function for finding workout partners, based on criteria such as fitness level, location, and availability.
- A notification system for new messages, posts, and updates on progress.
- An administrative system for managing user data, including the ability to delete user accounts if necessary.

4. Technical Requirements (Gary Li)

4.1. Operating System and Compatibility

The operating system and compatibility requirements for the "Progress" platform must support a wide range of devices and browsers to ensure maximum accessibility and usability for all users. The platform must be compatible with popular web browsers such as Google Chrome, Mozilla Firefox, Safari, and Microsoft Edge on desktops, laptops, and mobile devices.

4.2. Interface Requirements

4.2.1. User Interfaces

The UI should be intuitive, accessible, and visually appealing. It should have a simple layout, clear navigation, and fast loading times. The platform should encourage social interaction, allowing users to create and share workout plans, training tips, and fitness challenges. It should be customizable, allowing users to track their progress and set fitness goals.

4.2.2. Hardware Interfaces

The "Progress" platform does not require any specific hardware interface. As a web-based platform, it can be accessed through any device with an internet connection and a web browser. However, to fully utilize the platform's features, it is recommended that users have access to a device with a stable internet connection, a display with high resolution, and sufficient processing power to ensure a smooth user experience.

4.2.3. Communications Interfaces

The communication standards to be utilized by the "Progress" platform include HTTPS for secure data transmission, OAuth 2.0 for user authentication, and REST API for interaction between the front-end and back-end components of the platform. Additionally, the messaging system should be designed to handle large amounts of data and support multimedia content such as images, videos, and audio files. Overall, the communication standards should be robust, reliable, and secure to provide a seamless and safe user experience.

4.2.4. Software Interfaces

The software interface for the "Progress" platform should be designed using React and Thymeleaf for the front-end, and JPA for the back end, with Spring Boot as the connecting framework. The front-end should be responsive, providing a seamless user experience across different screen sizes. The back end should store user data securely using JPA, and follow standardized architecture and coding practices to ensure maintainability and scalability. RESTful APIs should be used to enable communication between the front-end and back-end.

5. Non-Functional Requirements (Devin Holcomb)

5.1.Performance Requirements

- NFR0(R): The local copy of the vehicle violation database will consume less than 20 MB of memory
- NFR1(R): The system (including the local copy of the vehicle violation database) will consume less than 50MB of memory
- NFR2(R): The novice user will be able to create and print a ticket in less than 5 minutes.
- NFR3(R): The expert user will be able to create and print a ticket in less than 1 minute.

5.2.Safety Requirements

5.3.Security Requirements

- The software only asks for relevant user personal data, not excessive or unnecessary.
- The software is only able to be accessed by users with an account.
- The software only asks for the permissions that are required

5.4.Software Quality Attributes

5.4.1. Availability

The software needs to be able to cope and deal with any possible errors/ flaws that it may encounter, so therefore error catching will need to be implemented into relevant sections as to ensure that an error does not break the software. Additionally, sufficient testing will need to take place as to identify areas where errors occur to then correct these.

5.4.2. Correctness

The software will need to perform the exact tasks/ processes that we have defined in this document.

5.4.3. Maintainability

The software will need to be able to be updated and improved as time progresses, once the software is released then users may have ideas and suggestions to improve the product by this informing about bugs and problems or features that they want to have added to provide a better service. Therefore, the code of the software needs to be able to be added to easily in order for this to be a possibility.

5.4.4. Re-usability

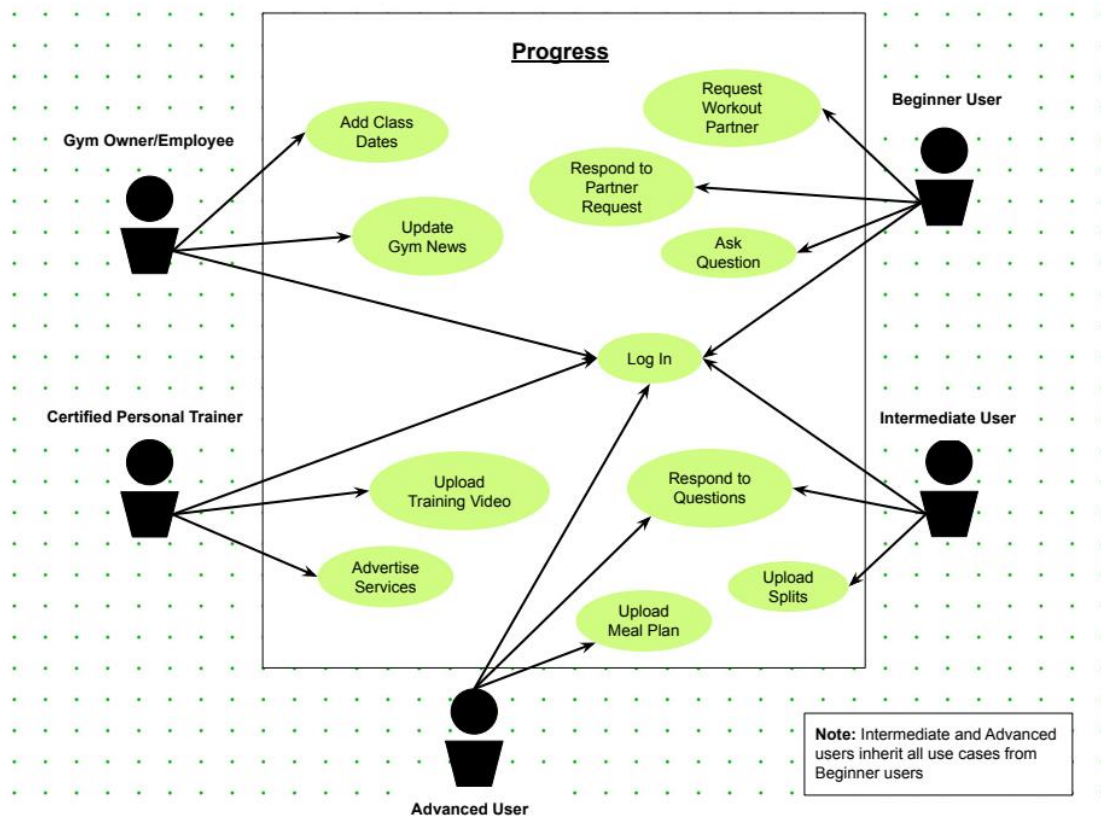
Using code that is already available. Such as code libraries and APIs. Also by splitting the code created into different modules and sections as so that it can be used by multiple areas of the software, thus bringing down cost and time.

5.4.5. Portability

The software will initially be created using a web platform and will be designed initially as a desktop product, the software will be responsive in order to fit to different sized screens in order to maximize the number of devices that it can be clearly viewed on. In the future in order to maximize portability the product will be adapted into a mobile application as this will allow for a larger number of users to use the platform.

5.5. Other Requirements

5.6. Use-Case Model Diagram



5.7. Use-Case Model Descriptions

5.7.1. Actor: Beginner user (Devin Holcomb)

- **Request Partner:** User can send a request to match up with a workout partner
- **Respond to Partner Request:** User can respond with yes or no to a request they've received
- **Ask Question:** User can ask a question related to working out
- **Login:** User can log in with their own credentials

5.7.2. Actor: Intermediate User (Devin Holcomb)

- **Request Partner:** User can send a request to match up with a workout partner
- **Respond to Partner Request:** User can respond with yes or no to a request they've received
- **Ask Question:** User can ask a question related to working out
- **Login:** User can log in with their own credentials
- **Respond to Questions:** User can provide their answer to an open question from another user

- **Upload Splits:** User can upload their very own workout that they're following for other users to view

5.7.3. Actor: Advanced User (James Copestake)

- Has the same functions as both the beginner and intermediate
- **Answer questions:** To those asked by both beginner and intermediate and also other advanced.
- **Upload meal plan:** Can upload their meal plan with nutritional info and pictures for other users to gain inspiration from.

5.7.4. Actor: Personal Trainer (Dallin Pierce)

- **Upload training videos:** This user can upload videos of them completing workouts for others to follow
- **Publicize their services:** This user can advertise their personal training business on the app and other users can then respond to this.

5.7.5. Actor: Gym Owner (Gary Li)

- **Add Class Dates:** User can add dates for specific classes for the gym
- **Update Gym News:** User can add news about the gym for other users to see
- **Login:** User can log in with their own credentials

5.8. Use-Case Model Scenarios

5.8.1. Actor: Beginner User (Devin Holcomb)

- **Use-Case Name: Request Workout Partner**
 - **Initial Assumption:** The user has a function that can send a request to another user to be partners.
 - **Normal:** The user will go to another's profile and use a button that will send a request to partner up.
 - **What Can Go Wrong:** The request can be denied due to the other user's privacy settings.
 - **Other Activities:** The user can request to follow/befriend the user so the request will be allowed.
 - **System State on Completion:** The two users are now partners. They can view this from either of their profiles.
- **Use-Case Name: Respond to partner request**
 - **Initial Assumption:** The user has an inbox, in which a request to partner is contained. They are prompted to accept or deny the request.

- **Normal:** The user will choose to accept or deny the request.
 - **What Can Go Wrong:** They could choose the wrong option
 - **Other Activities:** They can simply go to the other user's profile and send a request back.
 - **System State on Completion:** The user will be awaiting the response of the other user.
- **Use-Case Name:** Log in
- **Initial Assumption:** The user has an account to login to the system. The account is stored in the database.
 - **Normal:** The user enters a name and password to login to the account.
 - **What Can Go Wrong:** The user may enter the wrong password or forget theirs.
 - **Other Activities:** The user can use a "forgot password" function to reset their password.
 - **System State on Completion:** The user is logged in to their account. They can view other accounts and see their front page.
- **Use-Case Name:** Ask Question
- **Initial Assumption:** The user can post a question or ask for advice on a discussion board.
 - **Normal:** The user types out their question and posts it to the board.
 - **What Can Go Wrong:** The user gets no response to their question.
 - **Other Activities:** The user can choose to repost so that people have another chance to see the question.
 - **System State on Completion:** The user's question is posted on the discussion board for other users to see.

5.8.2. Actor: Intermediate User (Devin Holcomb)

-**Use-Case Name:** Request Workout Partner

- **Initial Assumption:** The user has a function that can send a request to another user to be partners.
 - **Normal:** The user will go to another's profile and use a button that will send a request to partner up.
 - **What Can Go Wrong:** The request can be denied due to the other user's privacy settings.
 - **Other Activities:** The user can request to follow/befriend the user so the request will be allowed.
 - **System State on Completion:** The two users are now partners. They can view this from either of their profiles.
- **Use-Case Name:** Respond to partner request
- **Initial Assumption:** The user has an inbox, in which a request to partner is contained. They are prompted to accept or deny the request.

- **Normal:** The user will choose to accept or deny the request.
- **What Can Go Wrong:** They could choose the wrong option
- **Other Activities:** They can simply go to the other user's profile and send a request back.
- **System State on Completion:** The user will be awaiting the response of the other user.

-Use-Case Name: Log In

- **Initial Assumption:** The user has an account to login to the system. The account is stored in the database.
- **Normal:** The user enters a name and password to login to the account.
- **What Can Go Wrong:** The user may enter the wrong password or forget theirs.
- **Other Activities:** The user can use a "forgot password" function to reset their password.
- **System State on Completion:** The user is logged in to their account. They can view other accounts and see their front page.

-Use-Case Name: Ask Question

- **Initial Assumption:** The user can post a question or ask for advice on a discussion board.
- **Normal:** The user types out their question and posts it to the board.
- **What Can Go Wrong:** The user gets no response to their question.
- **Other Activities:** The user can choose to repost so that people have another chance to see the question.
- **System State on Completion:** The user's question is posted on the discussion board for other users to view.

-Use-Case Name: Respond to Question

- **Initial Assumption:** User can see a question on the discussion board and choose to answer it or comment.
- **Normal:** The user types their answer and posts it to the board.
- **What Can Go Wrong:** The user mistypes in their answer, so their answer is misunderstood.
- **Other Activities:** User can choose to edit any of their past posts.
- **System State on Completion:** The user's answer is posted correctly to the discussion board for other users to view.

-Use-Case Name: Upload Split

- **Initial Assumption:** User can upload a workout split for other people to see and download.
- **Normal:** The user creates their workout split and posts it to the feed/ their profile.
- **What Can Go Wrong:** The user decided to change their workout so the one they posted is incorrect.
- **Other Activities:** User can opt to post new split, or edit the existing one.

- **System State on Completion:** The user's split is now posted for other users to view and download.

5.8.3. Actor: Advanced User (James Copestake)

-Use-Case Name: Request Workout Partner

- **Initial Assumption:** The user has a function that can send a request to another user to be partners.
- **Normal:** The user will go to another's profile and use a button that will send a request to partner up.
- **What Can Go Wrong:** The request can be denied due to the other user's privacy settings.
- **Other Activities:** The user can request to follow/befriend the user so the request will be allowed.
- **System State on Completion:** The two users are now partners. They can view this from either of their profiles.

-Use-Case Name: Respond to partner request

- **Initial Assumption:** The user has an inbox, in which a request to partner is contained. They are prompted to accept or deny the request.
- **Normal:** The user will choose to accept or deny the request.
- **What Can Go Wrong:** They could choose the wrong option
- **Other Activities:** They can simply go to the other user's profile and send a request back.
- **System State on Completion:** The user will be awaiting the response of the other user.

-Use-Case Name: Log In

- **Initial Assumption:** The user has an account to login to the system. The account is stored in the database.
- **Normal:** The user enters a name and password to login to the account.
- **What Can Go Wrong:** The user may enter the wrong password or forget theirs.
- **Other Activities:** The user can use a "forgot password" function to reset their password.
- **System State on Completion:** The user is logged in to their account. They can view other accounts and see their front page.

-Use-Case Name: Ask Question

- **Initial Assumption:** The user can post a question or ask for advice on a discussion board.
- **Normal:** The user types out their question and posts it to the board.
- **What Can Go Wrong:** The user gets no response to their question.
- **Other Activities:** The user can choose to repost so that people have another chance to see the question.

- **System State on Completion:** The user's question is posted on the discussion board for other users to view.

-**Use-Case Name:** Respond to Question

- **Initial Assumption:** User can see a question on the discussion board and choose to answer it or comment. Respond to beginner, intermediate and advanced
- **Normal:** The user types their answer and posts it to the board.
- **What Can Go Wrong:** The user mistypes in their answer, so their answer is misunderstood.
- **Other Activities:** User can choose to edit any of their past posts.
- **System State on Completion:** The user's answer is posted correctly to the discussion board for other users to view.

-**Use-Case Name:** Upload Split

- **Initial Assumption:** User can upload a workout split for other people to see and download.
- **Normal:** The user creates their workout split and posts it to the feed/ their profile.
- **What Can Go Wrong:** The user decided to change their workout so the one they posted is incorrect.
- **Other Activities:** User can opt to post new split, or edit the existing one.
- **System State on Completion:** The user's split is now posted for other users to view and download.

- **Use-Case Name:** Upload Meal plan

- **Initial Assumption:** User can upload their meal plan with relevant nutritional info for other users to see download and implement into their routine.
- **Normal:** The user creates their meal plan and uploads it to their profile
- **What Can Go Wrong:** User can accidentally input the incorrect dietary information
- **Other Activities:** Users can update and edit their meal plans they previously posted
- **System State on Completion:** The user's meal plan is now available for other users to view and download.

5.8.4. Actor: Personal Trainer (Dallin Pierce)

- **Use-Case Name:** Upload training videos:

- **Initial Assumption:** The Trainer is logged in and has a video of an accepted format to upload, through use of the YouTube Data API
- **Normal:** The video is uploaded and displayed to their clients or on their page.
- **What Can Go Wrong:** The video may fail to upload due to a miscommunication with YouTube, or the failure of the API.

- **Other Activities:** the user may want to manage or view videos already uploaded to their account.
- **System State on Completion:** The video is uploaded and displayed to their followers/clients and on their account.
- **Use Case Name:** Publicize their services:
 - **Initial Assumption:** The trainer will create a special kind of post that can be seen by users of various levels, which can be specified by the trainer.
 - **Normal:** the post is successful and stored on the database, where it can be served to users of appropriate levels.
 - **What Can Go Wrong:** there may be an error when saving the post causing it to not be stored in the database, or the user may make an error when posting causing for a need to be able to remove their post.
 - **Other Activities:** The user may want to delete one of their advertisements, or edit info on said advert.
 - **System State on Completion:** The post is saved to the database and can be viewed by users of appropriate levels.

5.8.5. Actor: Gym Owner

-**Use-Case Name:** Add class dates

- **Initial Assumption:** User can add a class schedule so users know when they can attend
- **Normal:** The user adds the hours for a class the gym is having
- **What Can Go Wrong:** The user accidentally adds the incorrect times for the class
- **Other Activities:** User can edit their schedule posting to show the correct times
- **System State on Completion:** Within the gym's information, users can see where the class schedule is posted

-**Use-Case Name:** Update Gym News

- **Initial Assumption:** User can add updates/news story for current events for the gym
- **Normal:** The user updates the gym's page to show that they will be getting new equipment
- **What Can Go Wrong:** User posts information before it has been confirmed and the new equipment has been delayed
- **Other Activities:** The user can remove the news post to prevent other users from being misinformed
- **System State on Completion:** The gym's news only shows what was added previously, since the new incorrect story was removed

-**Use-Case Name:** Log In

- **Initial Assumption:** The user has an account to login to the system. The account is stored in the database.
- **Normal:** The user enters a name and password to login to the account.
- **What Can Go Wrong:** The user may enter the wrong password or forget theirs.
- **Other Activities:** The user can use a “forgot password” function to reset their password.
- **System State on Completion:** The user is logged in to their account. They can view other accounts and see their front page.

6. Design Documents

6.1. Software Architecture (Gary Li)

- The platform will be implemented as a web application.
- Users will be able to create accounts and log in to access the platform's features.
- Different user levels will have access to different features and functionality within the platform.
- The platform will include private messaging, question/answer posting, partner matching, and personal trainer ads as its core features.

Classes

Controller

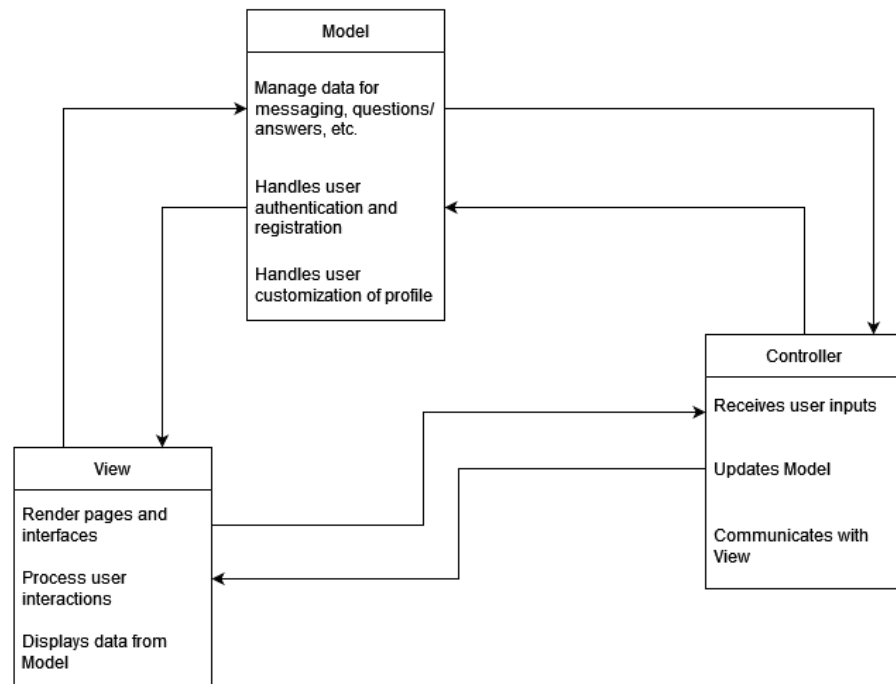
- **UserController:** Handles user inputs and communicates with the Model and View to display relevant information to the user.
- **MessageController:** Handles the sending and receiving of private messages between users.
- **PostController:** Handles the posting of questions, answers, and personal trainer ads.
- **PartnerController:** Handles the request and matching of workout partners.

Model

- **UserModel:** Stores and manages user data, including authentication and registration information.
- **MessageModel:** Stores and manages private messages between users.
- **PostModel:** Stores and manages questions, answers, and personal trainer ads.
- **PartnerModel:** Stores and manages requests and matches for workout partners.

View

- **UIView:** Renders pages and interfaces related to user authentication, registration, and account settings.
- **MessageView:** Renders pages and interfaces related to private messaging between users.
- **PostView:** Renders pages and interfaces related to posting questions, answers, and personal trainer ads.
- **PartnerView:** Renders pages and interfaces related to finding and matching workout partners.

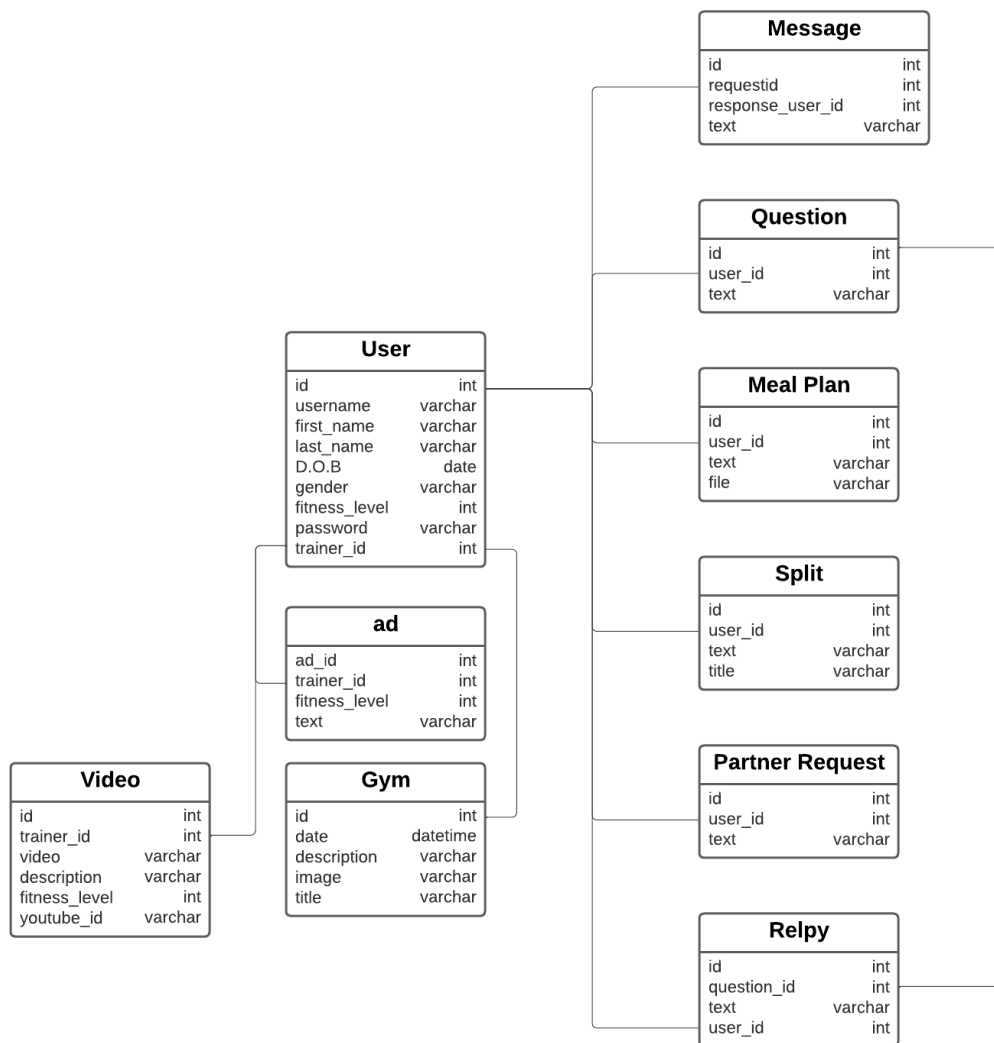


6.2. High-Level Database Schema (James Copestake)

- **User**
 - Id – unique identifier for each user
 - Username – the user's unique username within the software
 - First_name – user's first name
 - Last_name – user's last name
 - D.O.B – user's date of birth
 - Gender – user's gender

- Fitness level – user's fitness level to determine what features they have available within the software
- Password – the user's password that is encrypted for them to log in
- Trainer_id – the trainer id to link a user to their trainer
- Split
 - Id – unique identifier for each split post
 - User_id – links to the user that created the post
 - Text – the text that is within the post, the exercise that the user does
 - File – any files that the user may have uploaded alongside the text they entered
- Meal Plan
 - Id – unique identifier for each meal plan post
 - User_id – links to the user that created the post
 - Text – the text that is within the post, the breakdown of meals that they eat
 - File - any files that the user may have uploaded alongside the text they entered
- Question
 - Id – unique identifier for each question post
 - User_id – links to the user that created the post
 - Text – the text that is included within the question post
- Reply
 - Id – unique identifier for the reply
 - Question_id – links to the question that the response is to
 - Text – the text included within the reply
 - User_id – the user id of the user replying to the question
- Partner Request
 - Id – unique identifier of the partner request
 - User_id – the user id of the user who posted the partner request
 - Text – the text included within the partner request
- Message
 - Id – unique identifier for the message
 - Requestid – the user id of the user who will receive the message
 - Response_user_id – the user id of the user who responded to the message
 - Text – the text included within the message
- Gym
 - Id – unique identifier for the gym
 - Image – the images that are included within the gym page
 - Date – the date the info was posted
 - Title – the title of the info that has been added
- Video
 - Id – unique identifier for each training video
 - Trainer_id – the id of the trainer that posted the video
 - Video – the video that is on the software
 - Description – the description of the video
 - Fitness_level – indicates what fitness level is able to see the video
 - Youtube_id – the id of the video required for the API
- Ad
 - Ad_id – the unique id for the ad
 - Trainer_id – the id of the trainer that posted the ad

- Fitness_level – indicates what user is able to see the ad based on their level of fitness
- Text – the text that will be included in the add.

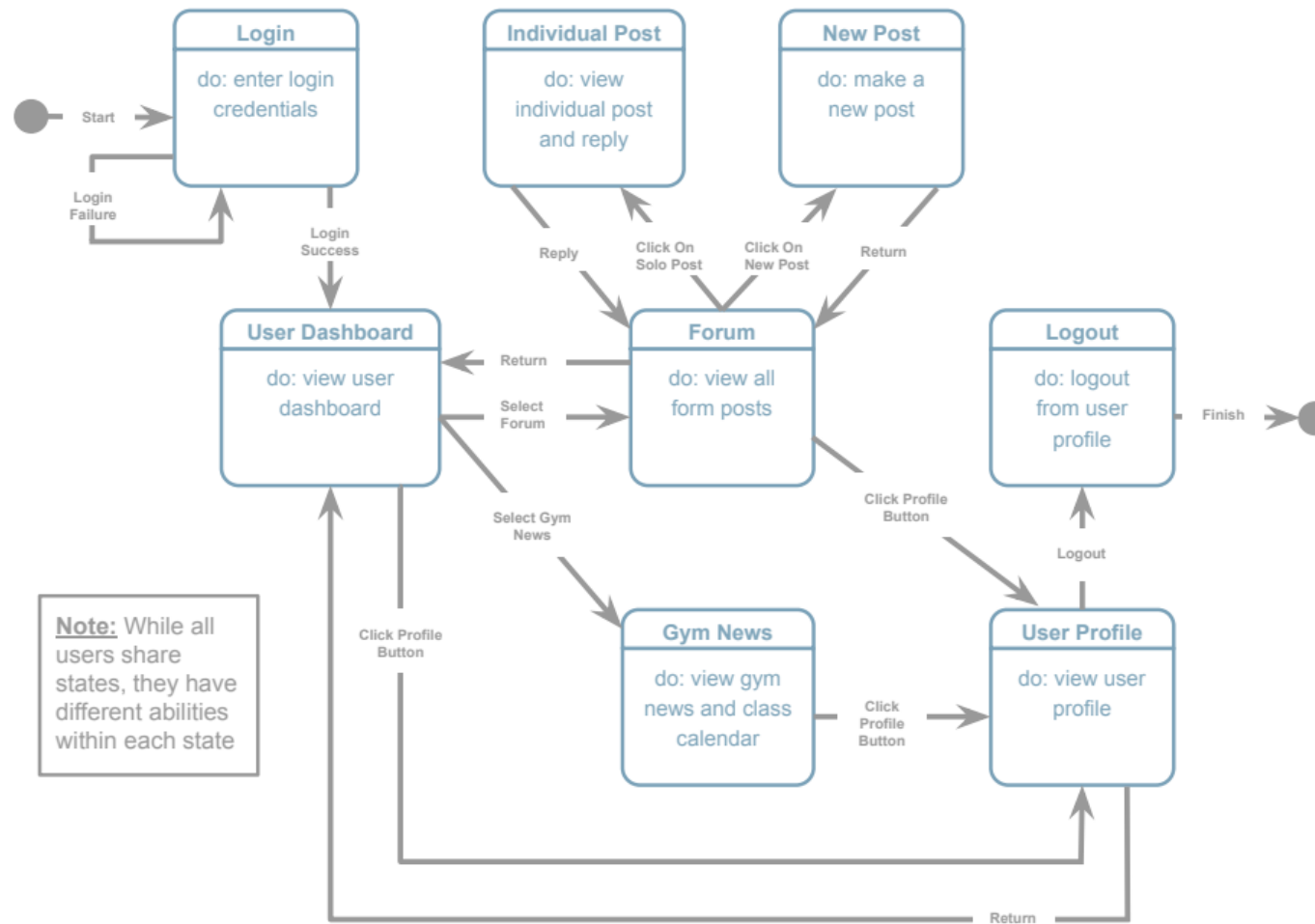


Many of the tables are linked together via the use of the user_id as all the posts and content on the system are all linked to the user_id be this posts, messages or client of the trainers, only one part of the system does not rely on the user_id, that is the gym table which is linked via the trainer as each trainer belongs to a certain gym. The trainer table relies on inheritance as the majority of the data stored within the user table is data that is also required for the trainers, thus by making use of inheritance the database system runs less risk of having data redundancies. The trainers are also classed as users. What differentiates the users from one another is the fitness level, 1 will be beginner, 2 will be intermediate, 3 will be advance and 4 will signify that the user is a trainer and thus will then link to the trainer table which will contain information that is only related to the trainers.

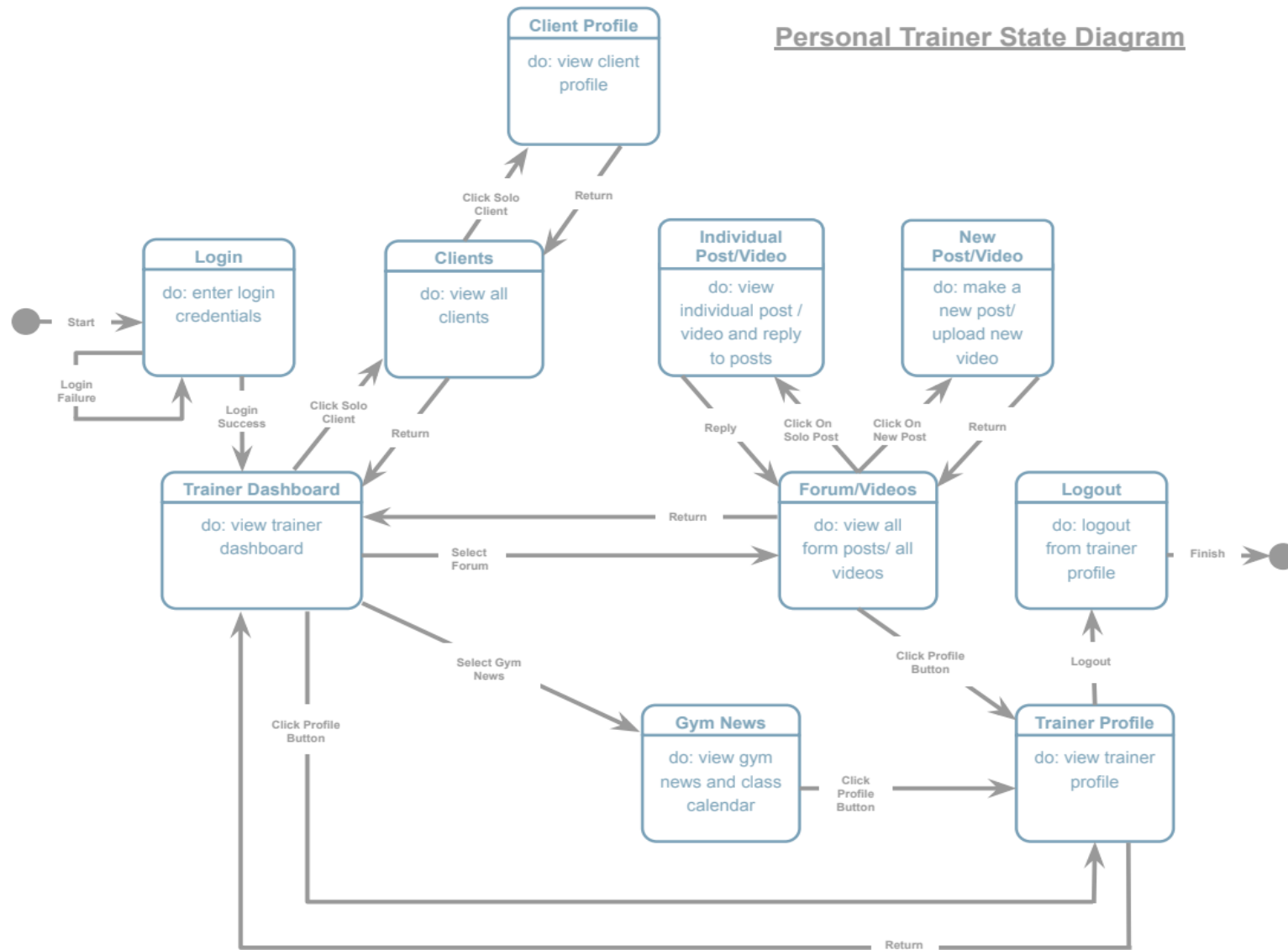
6.3. Software Design

6.3.1. State Machine Diagram: User < General > (Devin Holcomb/James Copestake)

User State Diagram

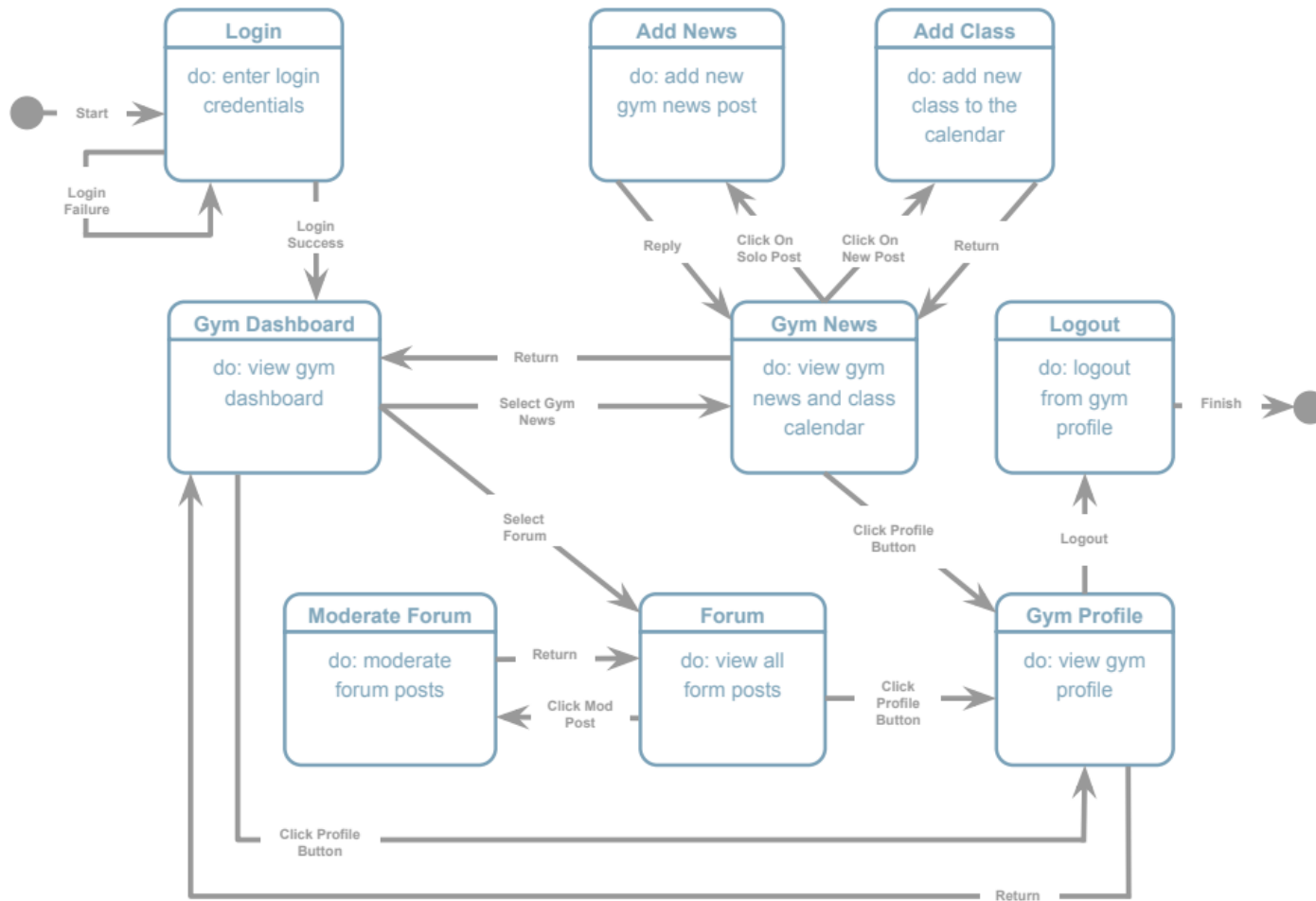


6.3.2. State Machine Diagram: Personal Trainer (Dallin Pierce)

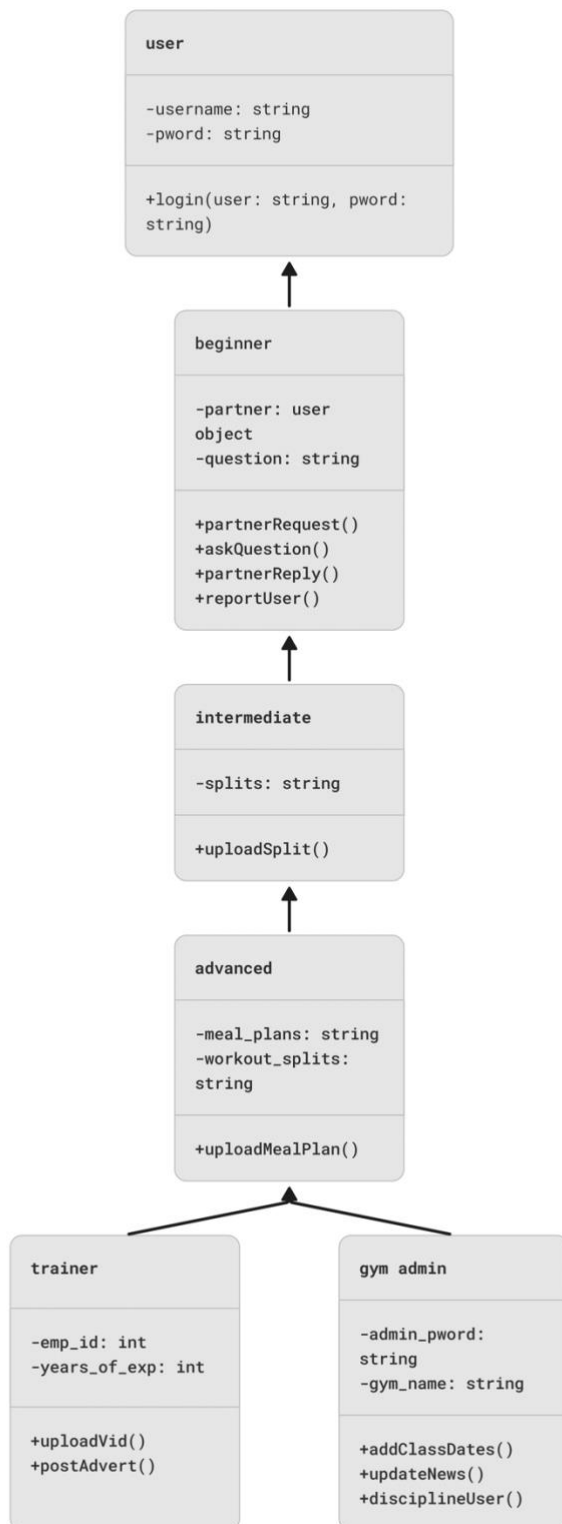


6.3.3. State Machine Diagram: Gym Owner/Employee (Gary Li)

Gym Owner State Diagram



6.4.UML Class Diagram (Devin Holcomb)

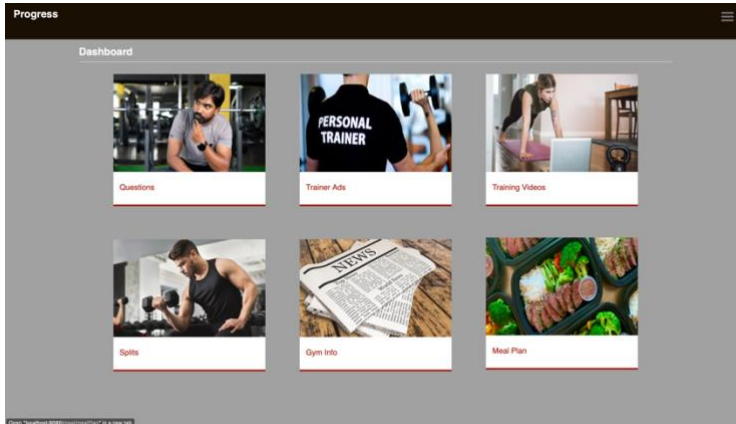


7. Scenario

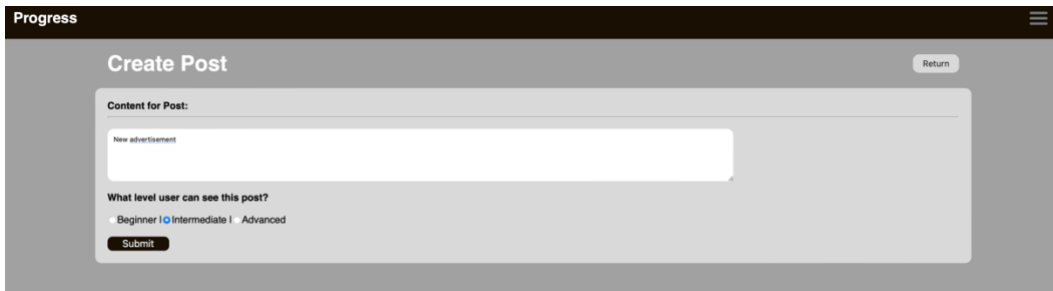
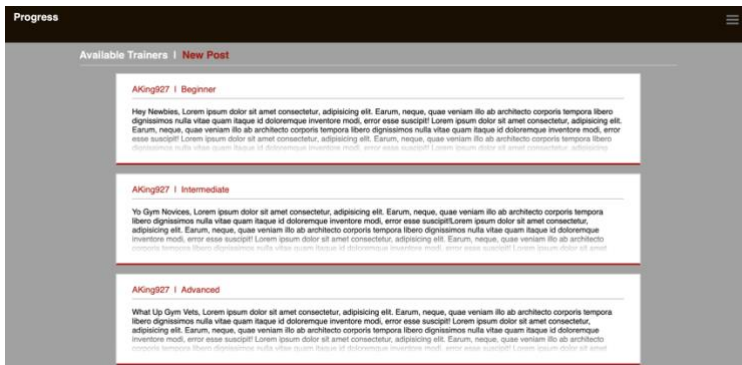
7.1. Brief Written Scenario with Screenshots (All)

Trainer

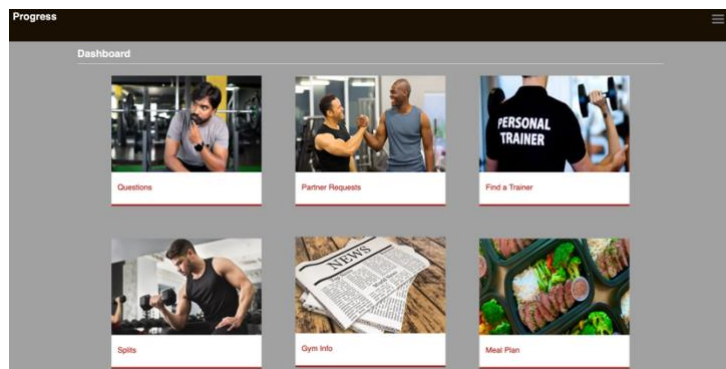
1. Trainer T1 logs in and uploads a new video.



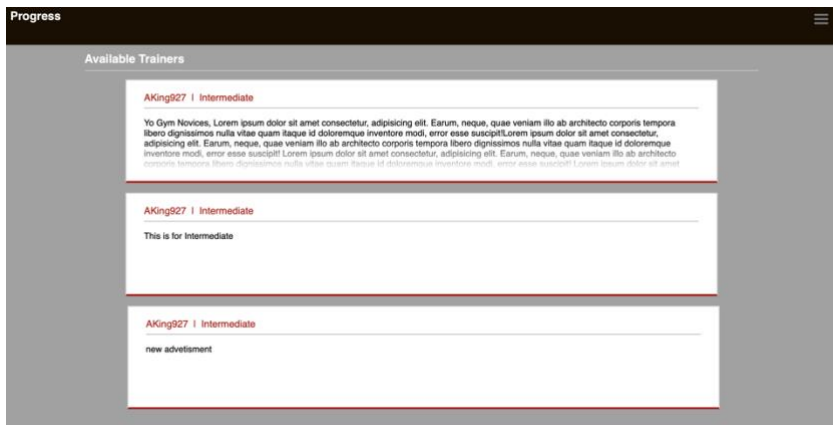
2. Trainer T1 then posts a new Intermediate advertisement. T1 then logs out.



3. User UB that is connected to the trainer logs in and can see that a new video has been uploaded by their trainer. UB logs out.

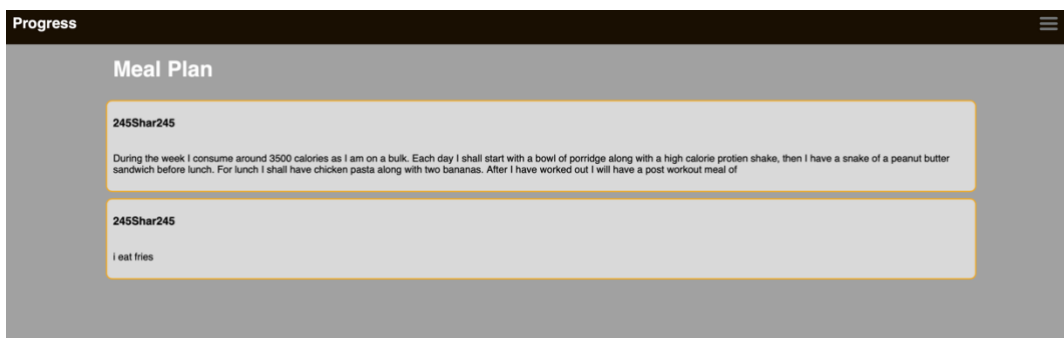


4. User UI logs in and goes to the trainer section where they can then view the advertisement from trainer T1. UI logs out.



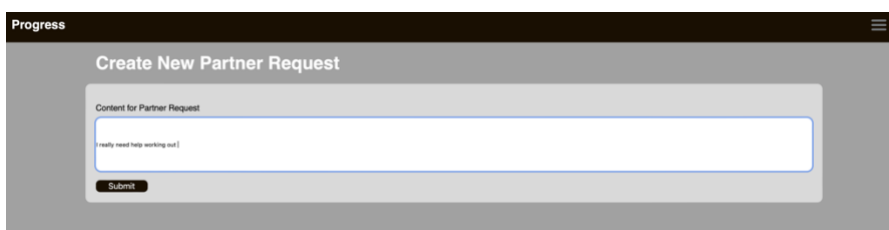
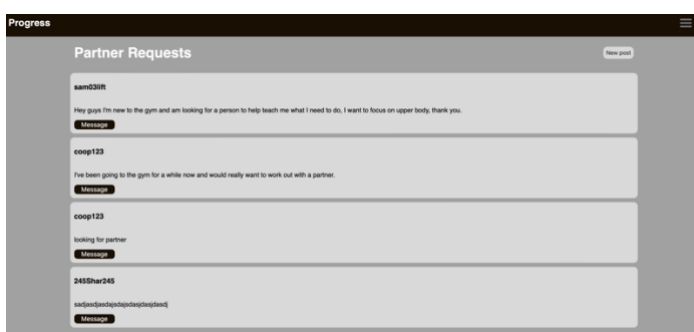
Beginner

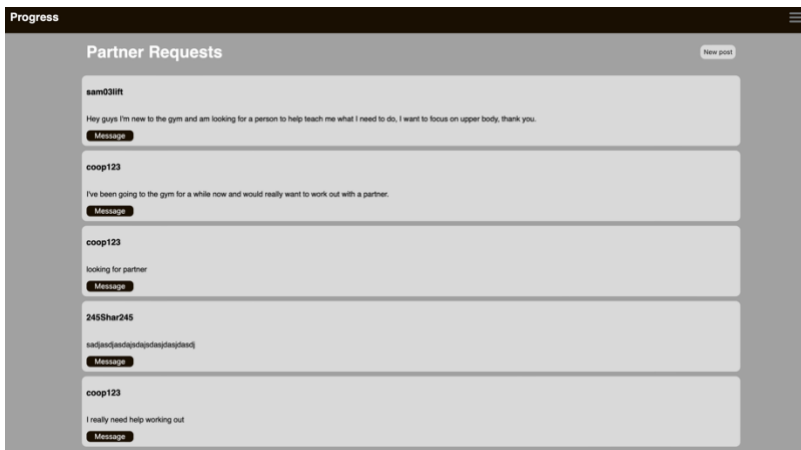
1. User UB logs in and views meal plans, they can only view and not upload due to their access level. User logs out.



Beginner

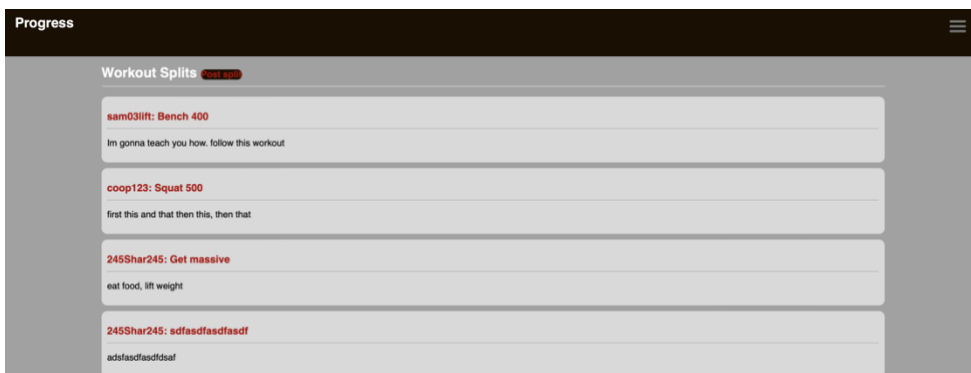
1. User UB logs in and goes to partner requests, they then post a new request to the page by pressing the new post button, stating what they want help in and where, they can then see their post on the page. User then logs out.





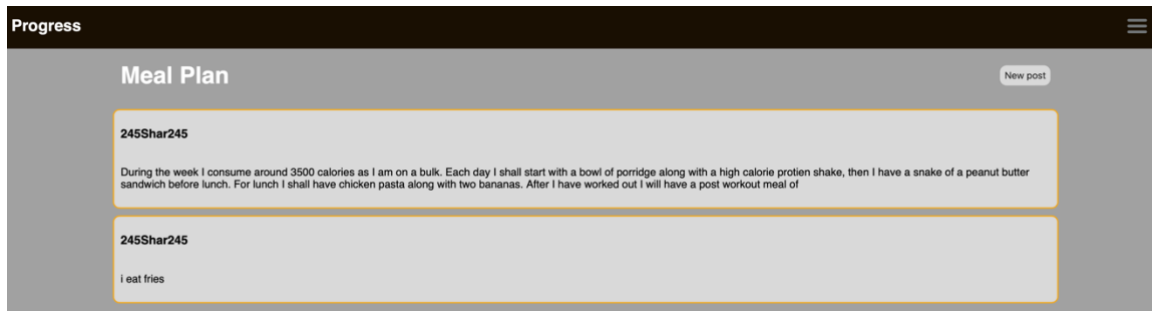
Intermediate

1. User UI logs in and views splits they then upload a new split as they have this access level. Once they have uploaded their new split it's available to see on the page. User then logs out.

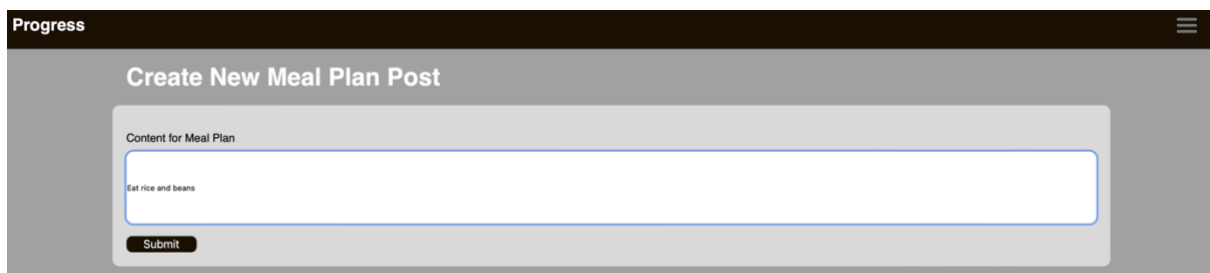


Advanced

1. User UA logs in and heads to the meal plan section of the website, they can view and also upload to this page as they have the correct access level.



2. User creates a new meal post and presses submit



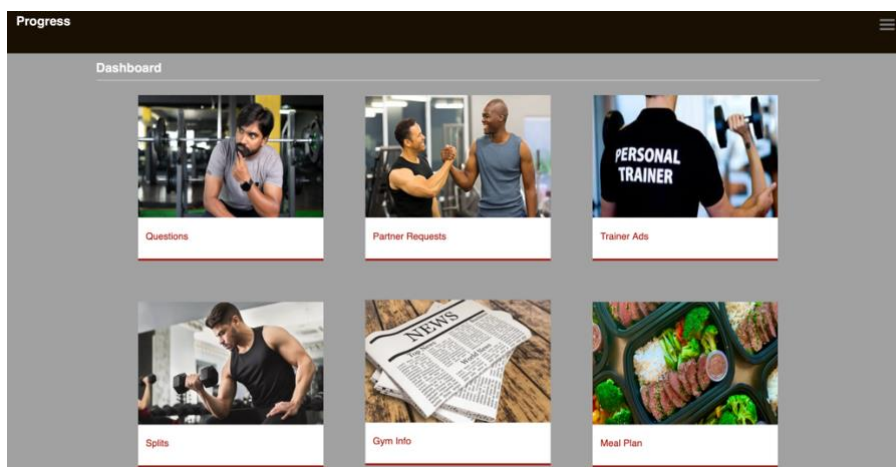
3. User can then see the post on the page









4. User then logs out.

Gym User

1. Gym User GU1 logs in and heads to gym info page, here they can create new info relating to their gym.



Gym Announcements					
ID	Title	Description	Image	Date Posted	
6	announcement	announcement		2023-04-18 10:30:10.0	View Details
5	announcement	announcement		2023-04-17 10:30:00.0	View Details
4	announcement	announcement		2023-04-17 10:30:00.0	View Details
3	announcement	announcement		2023-04-17 10:30:00.0	View Details
2	About Our Facility	announcement		2023-04-16 10:30:00.0	View Details
1	Welcome to _____ Gym	announcement		2023-04-17 10:30:00.0	View Details

Create New Info

Title:

test

Description:


this is a test for new gym info

Image URL:

4969 characters remaining

<https://images.unsplash.com/photo-1534438327276-14e5300c3a48?ixlib=rb-4.0.3&ixid=MnwxMjA3fDB8MHxzZWZyY2h8M3x8Z3ltfGVufDB8fDB8fA%3D%3D&w=1000&q=80>

[Back to List](#) [Create](#)

Gym Announcements					
ID	Title	Description	Image	Date Posted	
7	test	this is a test for new gym info		2023-04-25 10:30:55.0	View Details

- Once they have uploaded this, they can see this on the gym info page, however they have made a mistake so need to update the info they do this by pressing the edit button, on a new page they can now amend what was wrong. GU1 then logs out

Update announcement

Title:

7

Title:

test

Description:

this is a test for new gym info

Image URL:

4969 characters remaining

<https://images.unsplash.com/photo-1534438327276-14e5300c3a48?ixlib=rb-4.0.3&ixid=MnwxMjA3fDB8MHxzZWZyY2h8M3x8Z3ltfGVufDB8fDB8fA%3D%3D&w=1000&q=80>

[Update](#)