GetGainz

Team 3

Noya Hafiz, Carlos Avila, Cherishma Jalaparti, Nicholas Brodsky

Software Design Specification

&

Project Delivery Report

Version: (1) **Date: (11/05/2024)**

Table of Contents

1 Introduction....5
1.1 Goals and objectives....5
1.2 Statement of system scope....5
1.3 Reference Material....7
1.4 Definitions and Acronyms....7
2 Architectural design....7
2.1 System Architecture....7
2.2 Design Rational....8
3 Key Functionality design....8
3.1 Weightlifter Request Workout Calendar....9
3.1.1 Request Workout Calendar Graph Use Case....9
3.1.2 Processing sequence for Workout Calendar....10
3.1.3 Structural Design for Request Workout Calendar Profile....11
3.1.4 Key Activities....12
3.1.5 Software Interface to other components....12
3.2 Weightlifter Request Graph....12
3.2.1 Request Strength Relationship Graph Use Case....12
3.2.2 Processing Sequence for Strength Relationship Graph....13
3.2.3 Structural Design for Strength Relationship Graph....15
3.2.4 Key Activities....15
3.2.5 Software Interface to other components....15
3.3 Weightlifter Log Meal....16
3.3.1 Weightlifter Log Meal Use Case....16
3.3.2 Processing Sequence for Log Meal....17
3.3.3 Structural Design for Log Meal....17
3.3.4 Key Activities....19
3.3.5 Software Interface to other components....19
3.4 The Weightlifter Upload Photo....19
3.4.1 Upload Photo Use Cases....19
3.4.2 Processing sequence for Upload Photo....21
3.4.3 Structural Design for Upload Photo....22
3.4.4 Key Activities....23

3.4.5 Software Interface to other components..... 24
3.5 WeightLifter Log In..... 24
3.5.1 Weightlifter Log In Use Cases..... 25
3.5.2 Processing sequence for Log In..... 26
3.5.3 Structural Design for Log In..... 27
3.5.4 Key Activities..... 27
3.5.5 Software Interface to other components..... 28
3.6 Weightlifter Create account..... 28
3.6.1 Create accountUse Cases..... 28
3.6.2 Processing sequence for Create Account..... 29
3.6.3 Structural Design for Create Account..... 30
3.6.4 Key Activities..... 30
3.6.5 Software Interface to other components..... 32
3.7 WeightLifter Edit Profile Census..... 32
3.7.1 Edit Profile Use Cases..... 32
3.7.2 Processing sequence for Edit Profile..... 33
3.7.3 Structural Design for Edit Profile..... 34
3.7.4 Key Activities..... 35
3.7.5 Software Interface to other components..... 36
3.8 Weightlifter Log Strength..... 36
3.8.1 Log StrengthUse Cases..... 36
3.8.2 Processing sequence for Log Strength..... 38
3.8.3 Structural Design for Log Strength..... 39
3.8.4 Key Activities..... 40
3.8.5 Software Interface to other components..... 41
4 User interface design..... 41
4.1 Interface design rules..... 41
4.2 Description of the user interface..... 41
4.2.1 Main Menu Page..... 41
4.2.2 Fitness Enthusiast Login Page..... 41
4.2.3 Fitness Enthusiast Registration Page..... 42
4.2.4 Fitness Enthusiast Meal Logging..... 43
4.2.5 Fitness Enthusiast Workout Logging..... 44
4.2.6 Fitness Enthusiast Account Info..... 45
4.2.7 Fitness Enthusiast Calendar..... 46

5 Restrictions, limitations, and constraints..... 47
6 Testing Issues (SLO #2.v)..... 47
6.1 Types of tests..... 47
6.2 List of Test Cases..... 48
6.3 Test Coverage..... 50
7 Appendices..... 52
7.1 Packaging and installation issues..... 52
7.2 User Manual..... 54
7.3 Open Issues..... 56
7.4 Lessons Learned..... 56
7.4.1 Project Management & Task Allocations (SLO #2.i)..... 56
7.4.2 Implementation (SLO #2.iv)..... 57
7.4.3 Design Patterns..... 61
7.4.4 Team Communications..... 61
7.4.4 Technologies Practiced (SLO #7)..... 62
7.4.5 Desirable Changes..... 62
7.4.6 Challenges Faced..... 63

1 Introduction

This design document highlights all the features to be developed with their implementation details of the GetGains application. It explains data design through elaborating on how information at the level of the users, meal inventory, macronutrient tracking, and strength metrics is structured, stored, and manipulated within the system. It also covers architectural design, showing the overall structure and the interaction between different components for smooth operation and scalability. In this section, it also determines user interface layouts and patterns of interaction, emphasizing the ease of interactions. Finally, component-level design describes each component in the software concerning their responsibilities, algorithms, and interactions. This document outlines the template that shall guide the development of GetGainsfor users who wish to track strength and nutrition.

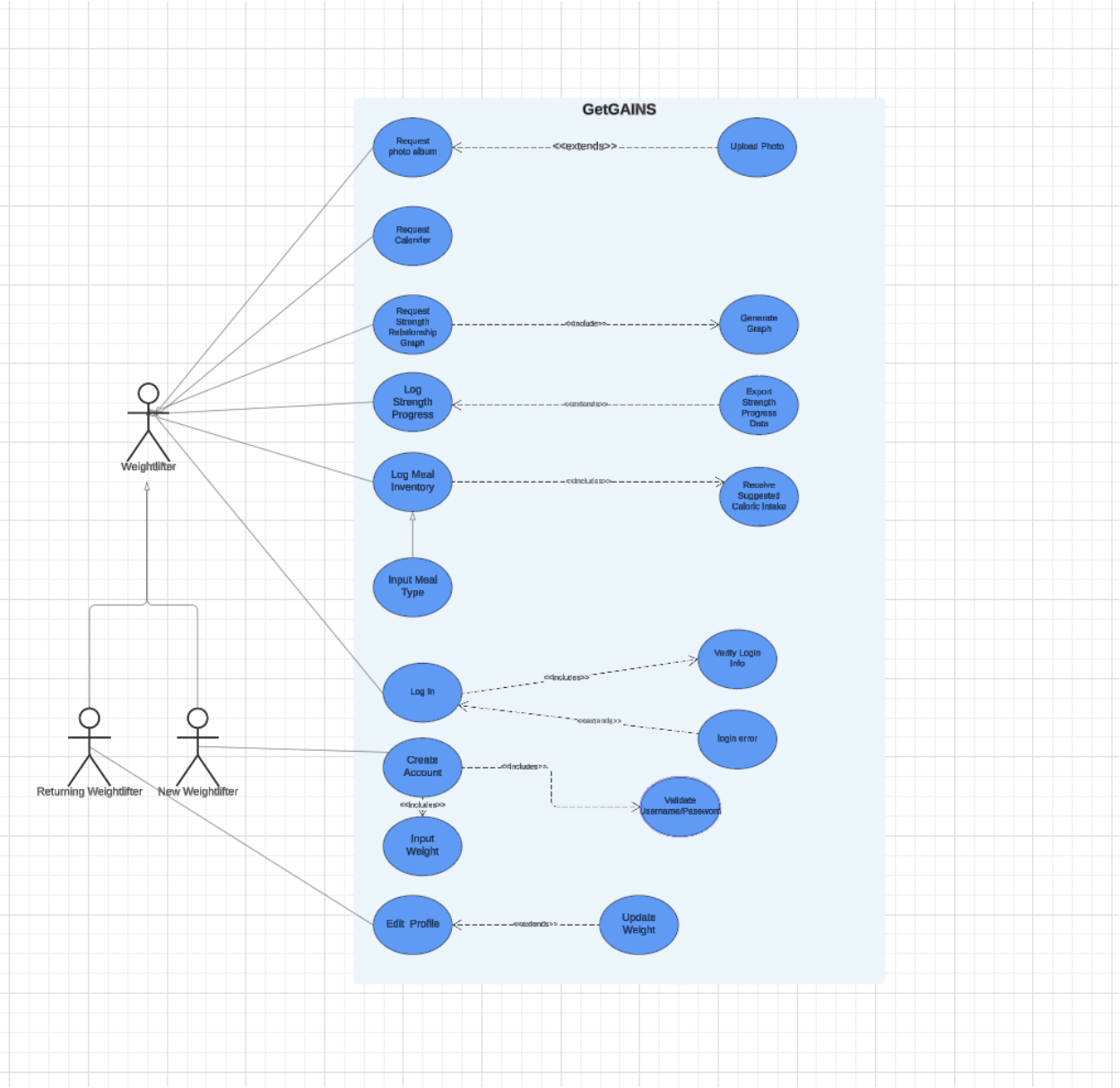
1.1 Goals and Objectives

1.1 Goals and Objectives

The key goal of the GetGains application is to provide users with an all-inclusive tool for monitoring and improving nutrition and strength training, especially for fitness enthusiasts whose goals are muscle gain and performance enhancement. The application shall provide for personalized nutrition tracking, whereby the user will be in a position to log and track the number of calories that they have taken daily. GetGains will also let one track strength progress by easily logging performance in key compound lifts-squats and bench presses-all the way to tracking strength gains. The application will make use of data through algorithms that allow personalized recommendations on intake levels of calories and macronutrients, automatically updating these recommendations in respect to changes made in users' weight and strength metrics. GetGains will also feature meal inventory management in improving usability to let users keep track of their meal and ingredient inventory, hence making meal planning and logging much easier. Allows the user to track their workouts through a calendar function. With focus on these goals and objectives, GetGains hopes to provide the user with an environment that allows the user to take responsibility for nutrition and strength training for better achievement of their fitness goals.

1.2 Statement of system scope

1.2 System Scope Statement



The GetGains application will include functionalities for users to guide their goals to gain muscle and strength by better tracking nutrition and exercise. The system will let users create personal accounts where they can log and keep track of vital information about themselves, such as current weight, macronutrient intake, performance metrics for compound lifts like squats, bench presses, and others. GetGains will be able to allow users to track their daily meals, trace macronutrients consumed for each type of food. It will also have a meal inventory management system to check on what meals and ingredients they have so that they can guide meal planning and avoid missed nutritional targets. GetGainz will also include a user-friendly interface that would ease navigation and improve user engagement, thus enabling people of various proficiency levels in tracking fitness and nutrition to work with it easily. In addition, GetGainz will allow users to log, edit, and view their individual workout schedules through a calendar.

1.3 Reference Material

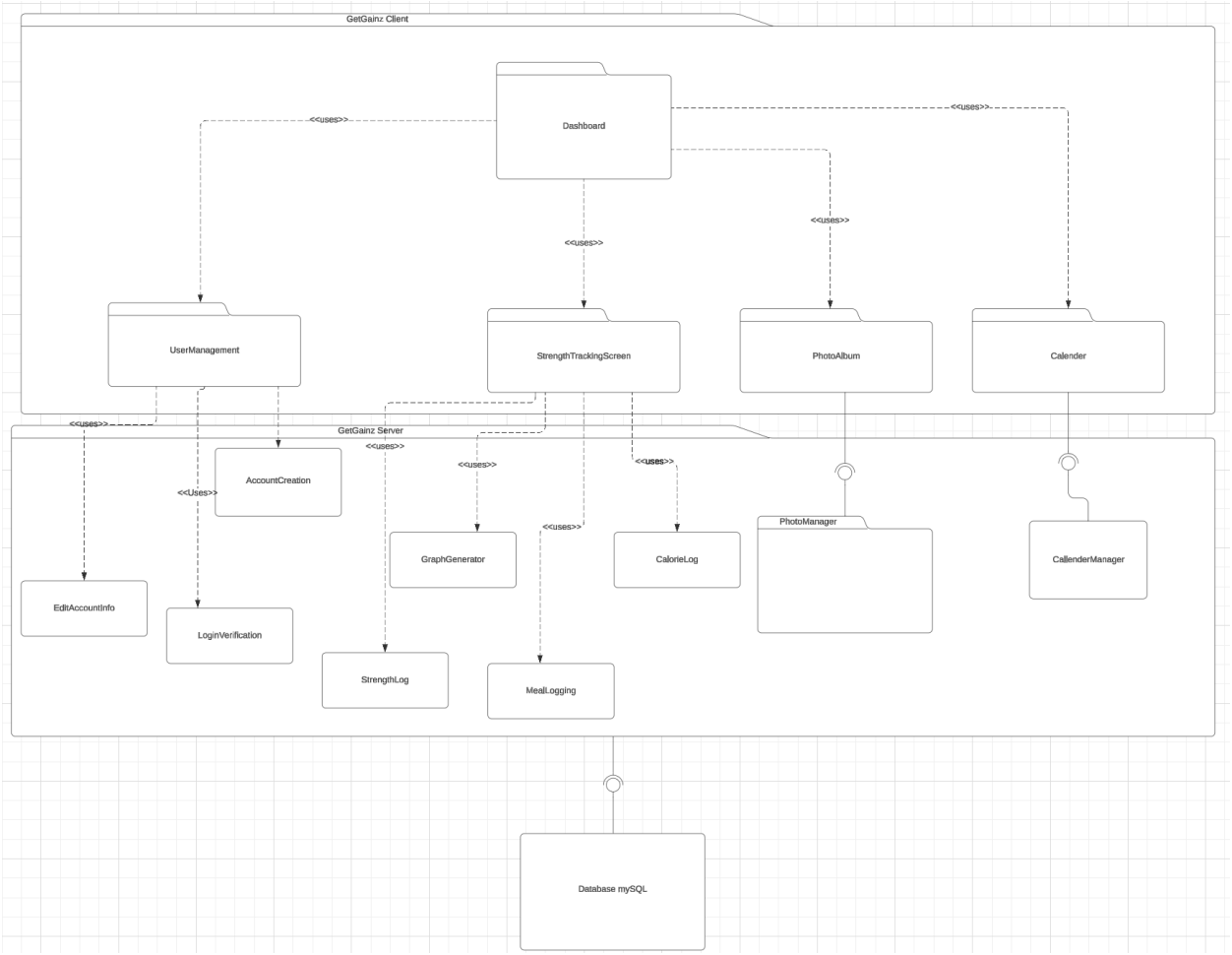
Not applicable.

1.4 Definitions and Acronyms

Not applicable.

2 Architectural design

2.1 System Architecture



UML's package diagram supports the structure that the GetGains system will follow; all components are divided into packages. These packages represent the key subsystems: Client, Server, and Database. In the Client Package there are modules to deal with UserManagement, StrengthTracking, and UI to handle a small portion of the user interface and user experience. UserManagement is responsible for the creation and management of accounts by users, which will include registration, logging in, and updating their profiles. The UI package handles the general user interface, display of information, and direct interaction of users with the system.

Some modules that enable backend processing within the Server package include AccountCreation and LoginVerification provide processing for user authentication and session

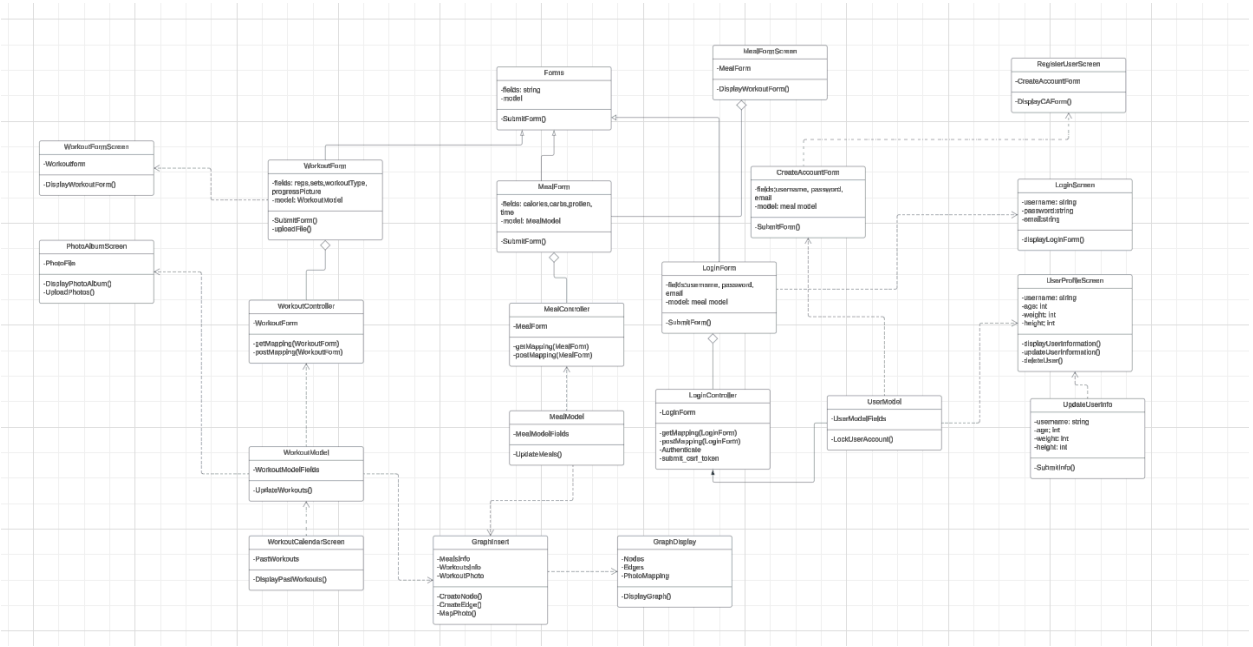
management in the system for secure access. GraphGenerator and CalorieLog create and log representations concerning user progress and daily caloric intake. MealLogging and StrengthLog store logs of meals and exercises correspondingly, while Home Screen and LoginScreen are the main screens of the interface. The Server also interfaces with the Database (H2), where user profiles, meal inventories, workout history, and other important data are held.

This modular architecture will ensure that all components interact with the rest of the components whenever necessary to provide full functionality. Communications between the Client and Server packages are performed through interfaces, whereas the Database package offers secure data storage and retrieval. This architecture is scalable because each of its modules can be independently maintained; any update or expansion of any module will have minimal impact on other modules. All these subsystems enable users to manage their goals of fitness and nutrition, log activities, and then view progress within the GetGains application.

2.2 Design Rational

We chose to have the client-server architecture because this application is made for an unknown number of users. A client-server architecture is useful for service oriented applications and Getgains would fall under this category. By using a client-server model, GetGains benefits from centralized data management, where the server securely stores user data, workout logs, and meal information, and ensures consistency and reliability across multiple user sessions. This architecture also supports modularity, allowing each subsystem—such as user management, meal logging, and strength tracking—to function independently, facilitating easier maintenance and potential enhancements. This approach allows GetGains to provide a seamless, reliable, and scalable experience for all users, aligning with the application's goals of supporting individualized progress tracking and nutritional guidance.

3 Key Functionality design

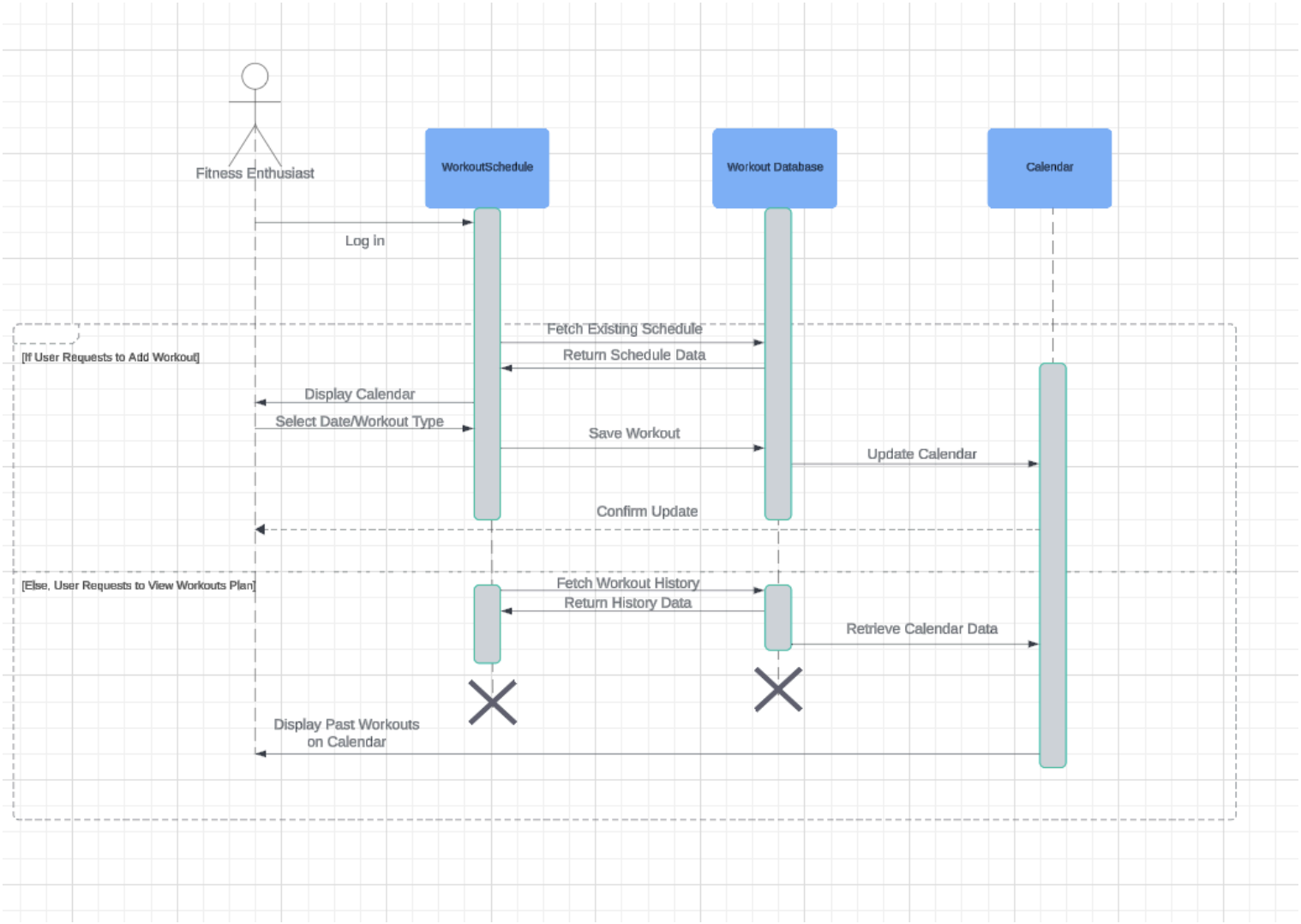


3.1 Weightlifter Request Workout Calendar

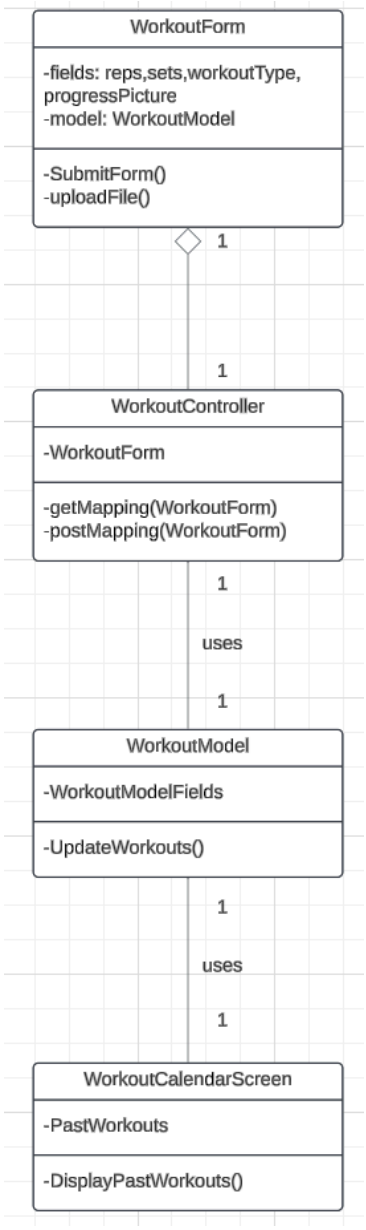
3.1.1 Request Workout Calendar Use Case

Use Case	Request Workout Calendar
Goal in context	Allow users to add, view, and manage their workout schedule with relevant exercise details, helping them stay on track with their fitness goals.
Scope	Enables the user to manage their workout schedule by adding, viewing, and tracking past and future workouts.
Level	Primary user goal level
Primary Actor	Weightlifter (user managing their workout schedule)
Precondition	The user must be logged into their account and have access to the workout calendar feature.
Minimal Guarantee	The user can view the calendar and add or edit workouts. The system will display the current schedule and save any added workouts.
Success Guarantee	The user successfully adds a workout event, and the system updates the workout calendar with the new event details, ensuring that both future and past workouts are easily accessible.
Trigger	The user selects the “Request Workout Calendar” option from the workout management interface.
Success Scenario	1. The user logs into their account and accesses the workout calendar. 2. The system displays the current workout calendar. 3. The user selects a date on the calendar to add a new workout. 4. The user specifies the workout type (e.g., squat, bench press) and inputs details such as the number of sets, reps, weight lifted, and any additional notes. 5. The system validates the entered workout information. 6. The system saves the new workout details to the workout database. 7. The workout calendar is updated, and the new workout is displayed on the calendar for the selected date. 8. The user can view past workout details by selecting previous dates on the calendar.
Extensions	1a. Invalid Data: If the user enters invalid workout data (e.g., unrealistic weight or sets), the system prompts the user to correct the input before submitting. 2a. View Past Workouts: If the user opts to view past workouts, the system fetches historical workout data from the database and displays it on the calendar. 3a. Cancel Workout Addition: If the user decides not to save a new workout, they can cancel the operation, and no new event is added to the calendar. 4a. Update Existing Workout: If the user chooses to update an existing workout, the system prompts them to select the workout and modify details such as sets, reps, or weights.

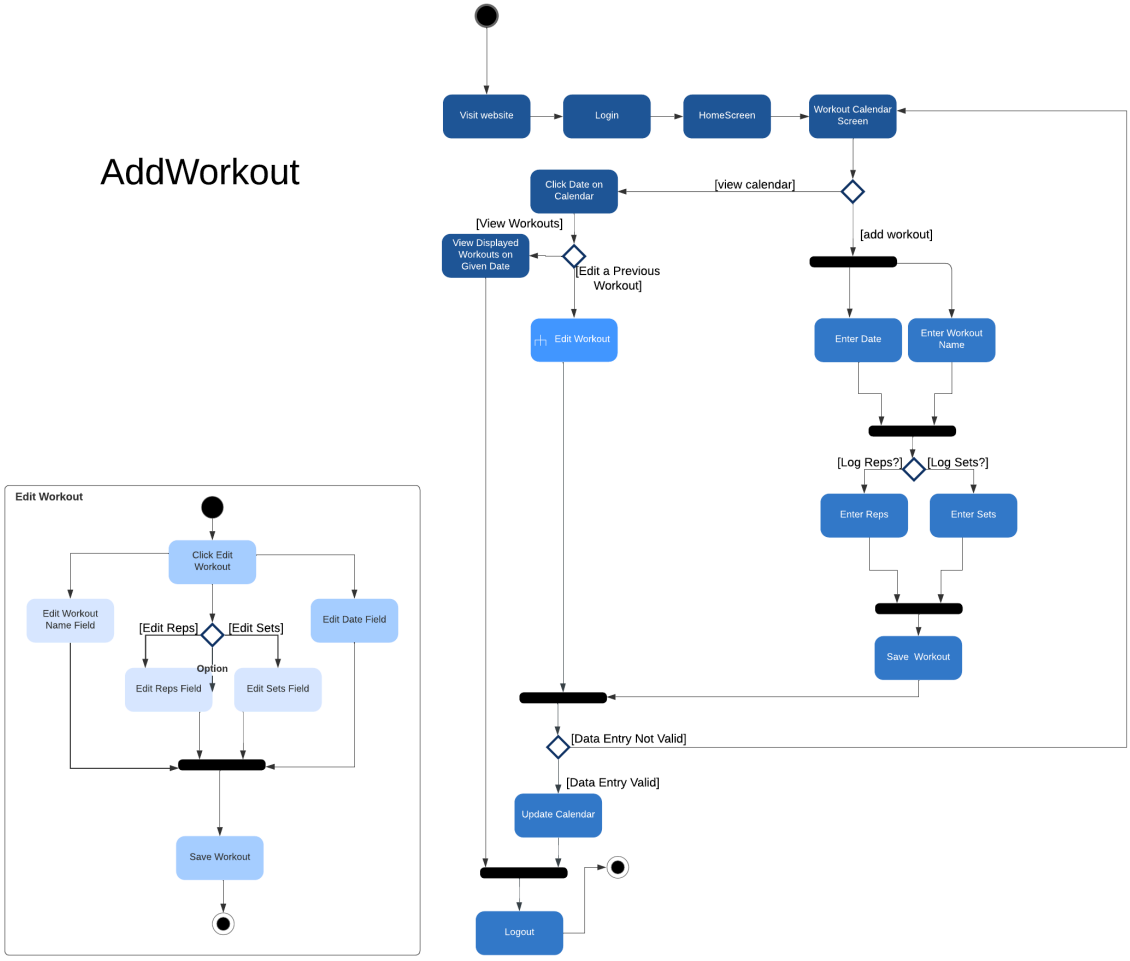
3.1.2 Processing sequence for Workout Calendar



3.1.3 Structural Design for Workout Calendar



3.1.4 Key Activities



3.1.5 Software Interface to other components

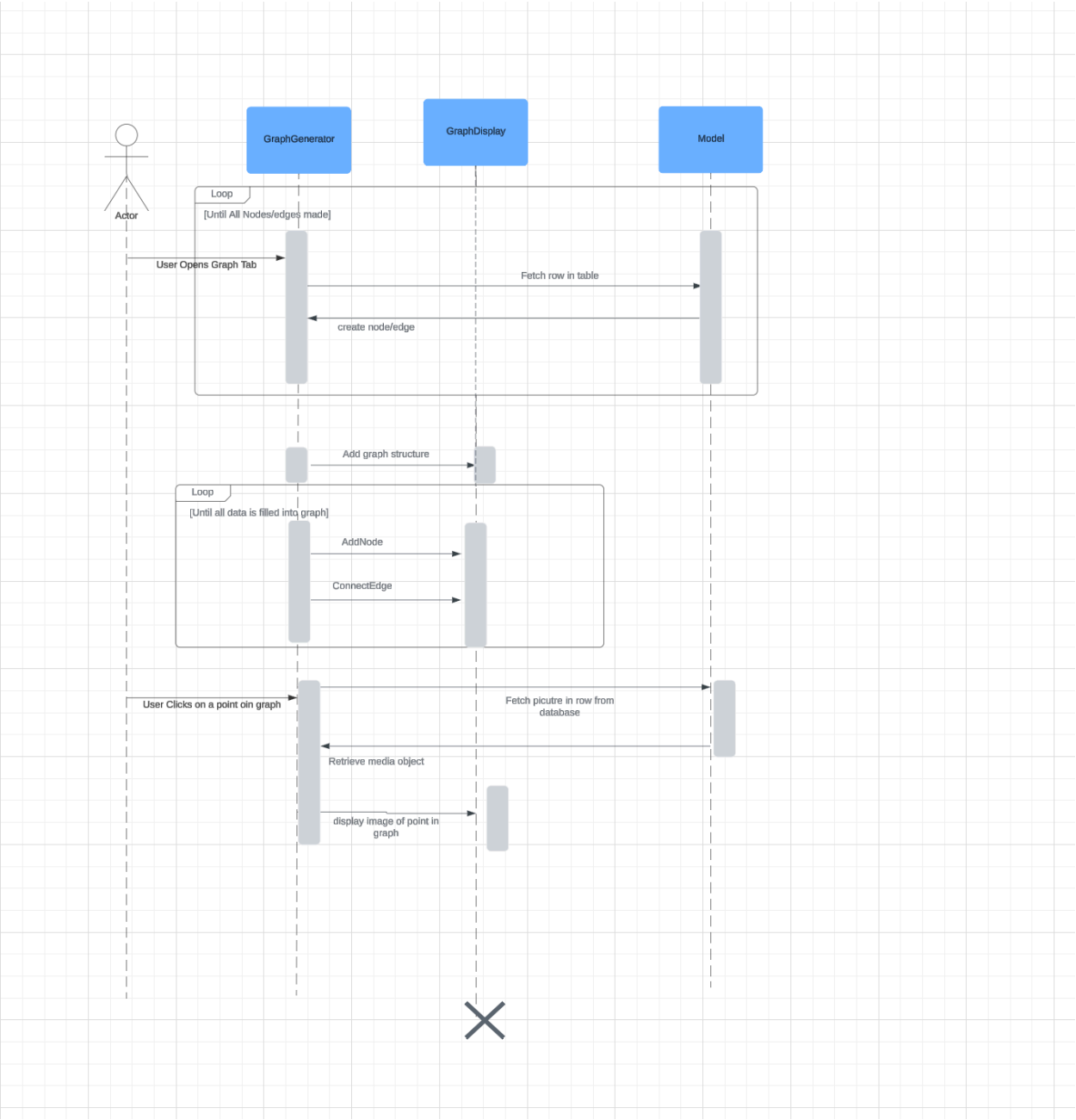
The workout calendar integrates with the user authentication system to ensure that only logged-in users can access and modify their calendar. The calendar is linked to the strength tracking system, as it logs workout performance data, such as weight lifted, sets, and reps.

3.2 Weightlifter Request Graph

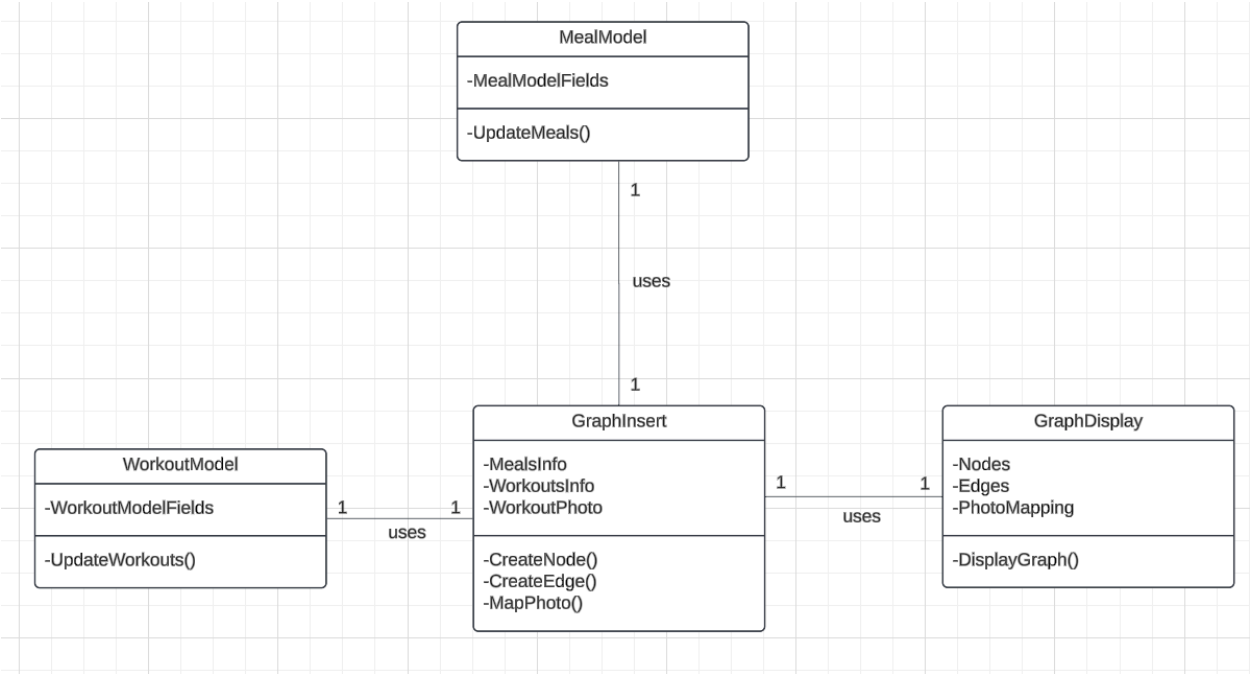
3.2.1 Request Strength Relationship Graph Use Case

Use Case	Request Strength Relationship Graph
Goal in context	Allow users to track their strength performance against caloric intake over time.
Scope	Allows users to analyze the relationship between strength gains and caloric intake to adjust their diet or exercise routine.
Level	Primary user goal level
Primary Actor	Weightlifter (user tracking strength progress)
Precondition	The user must be logged in with valid credentials and have enough data on strength progress and caloric intake.
Minimal Guarantee	The system will always verify user credentials, validate data inputs, and provide feedback if the data is insufficient.
Success Guarantee	The system successfully generates and displays a graph of strength progress against caloric intake.
Trigger	The user clicks the "Request Strength Relationship Graph" button.
Success Scenario	1. User logs in with valid credentials. 2. User navigates to the strength tracking section. 3. User selects the option to request the strength relationship graph. 4. The system checks if sufficient data is available. 5. The system generates the graph using data on strength and caloric intake. 6. The graph is displayed to the user. 7. The user analyzes the graph and makes informed decisions on adjusting their workout or nutrition.
Extensions	1a. Insufficient Data: If sufficient data is not available, the system prompts the user to log more data. 2a. Invalid Input: If the user inputs invalid data, the system prompts for correction. 3a. Data Update: If the user's data changes after generating the graph, the system updates the graph accordingly.

3.2.2 Processing Sequence for Strength Relationship Graph


GraphGenerator
GraphDisplay
Model
Actor
Loop
[Until All Nodes/edges made]
User Opens Graph Tab
Fetch row in table
create node/edge
Add graph structure
Loop
[Until all data is filled into graph]
AddNode
ConnectEdge
User Clicks on a point oin graph
Fetch picutre in row from database
Retrieve media object
display image of point in graph

3.2.3 Structural Design for Strength Relationship Graph



3.2.4 Key Activities

Not implemented.

3.2.5 Software Interface to other components

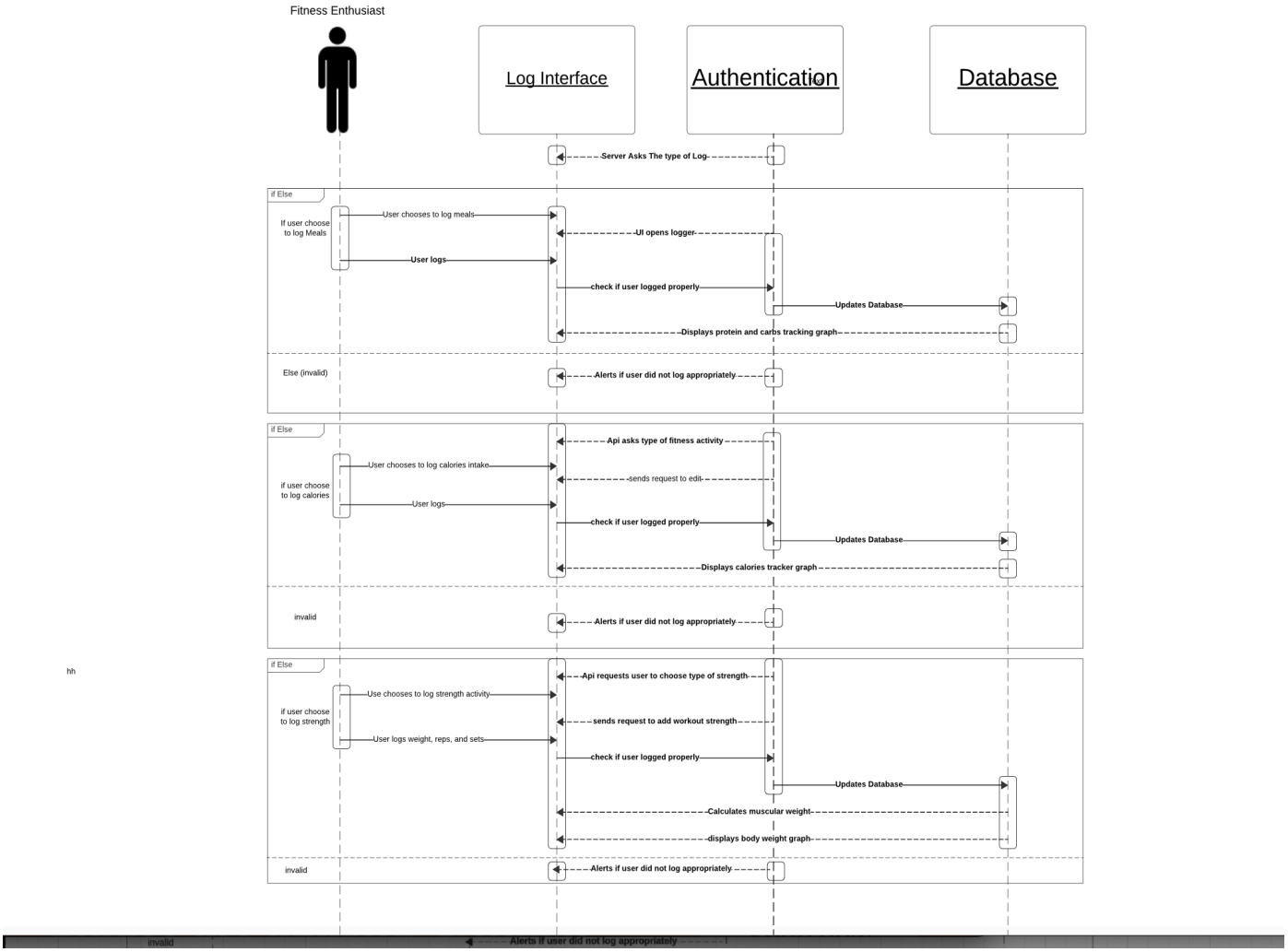
The request strength relationship graph has to work with several components, such as the user authentication system, user profile management, meal tracking system, and strength tracking system. First, the authentication module is important in ensuring that the user has logged in with valid credentials before making a request for the strength relationship graph. Following this, the system retrieves the user's data, such as strength progress and calorie intake, that will show the graph. The user profile management system is responsible for storing and manipulating information about a user's personal information, such as weight, goals, and strength targets. This information is required in order to create the suggested caloric intake and strength progress data reflected in the graph. The meal tracking system integrates into the user's profile, where they log meals, input the macronutrient information of their meals, and track their caloric intake. This system will provide the data required to create the strength relationship graph, since it relates calorie intake to performance of strength-related exercises.The other major component is the strength tracking system, which helps in the generation of the graph for the strength relationship. This system charts the performance of the user’s weight and calorie intake over time. This will be necessary for developing a useful graph showing how personal metric changes with time in relation to caloric intake. The graph creation and visualization module also fetches data from the meal and strength tracking systems. Once the information is processed, the module visually plots the user's strength progression against caloric intake over time. This module also interfaces directly with the UI to display the graph for interaction by the user in trends and making informed decisions based on progress. The Request Strength Relationship Graph feature interfaces to most databases, such as the User Data Database, Meal Log Database, and Workout Log Database, in regard to data flow. This information is recorded on these databases to populate the graph correctly. The system uses such databases to fetch the required data in real time, so that the user always views the most recent information.

3.3 Weightlifter Log Meal

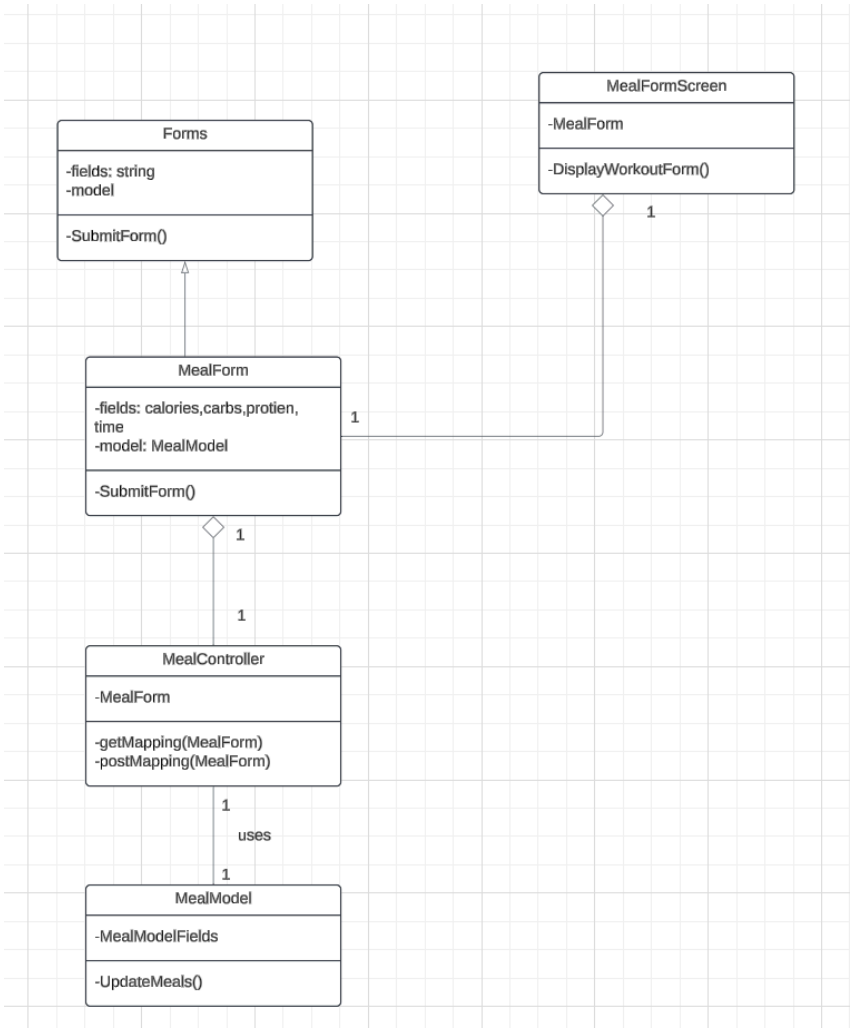
3.3.1 Weightlifter Log Meal Use Case

Use Case	Log Meal
Goal in context	Allow users to log their meals and track their nutritional intake over time.
Scope	Logging meals and calculating macronutrients to help users track their nutrition and adjust as needed.
Level	Primary User goal level
Primary Actor	Fitness Enthusiast (user logging meals)
Precondition	The user must be logged in to access meal logging features.
Minimal Guarantee	The system will store meal data, calculate macronutrients, and update the user's meal log.
Success Guarantee	The user's meal data is successfully logged and the meal log is updated with calories and macronutrients.
Trigger	The user selects the option to log a meal in the meal tracking interface.
Success Scenario	1. User logs into their account. 2. User navigates to the meal logging section. 3. User enters the food types consumed and their quantities. 4. The system retrieves the nutritional data (calories, protein, carbs, fats) for the entered foods. 5. The system calculates the total calories and macronutrients for the meal. 6. The system stores the meal data in the user's log. 7. The meal log is updated, and the user receives insights on their nutritional intake.
Extensions	1a. Invalid Data**:** If the user enters incorrect or missing food data, the system prompts for correction. 2a. Incomplete Meal: If the system cannot find nutritional data for a food item, the user is prompted to enter the data manually or choose an alternative food item. 3a. Meal Overview: After logging, the system displays a summary of the meal's nutritional breakdown.

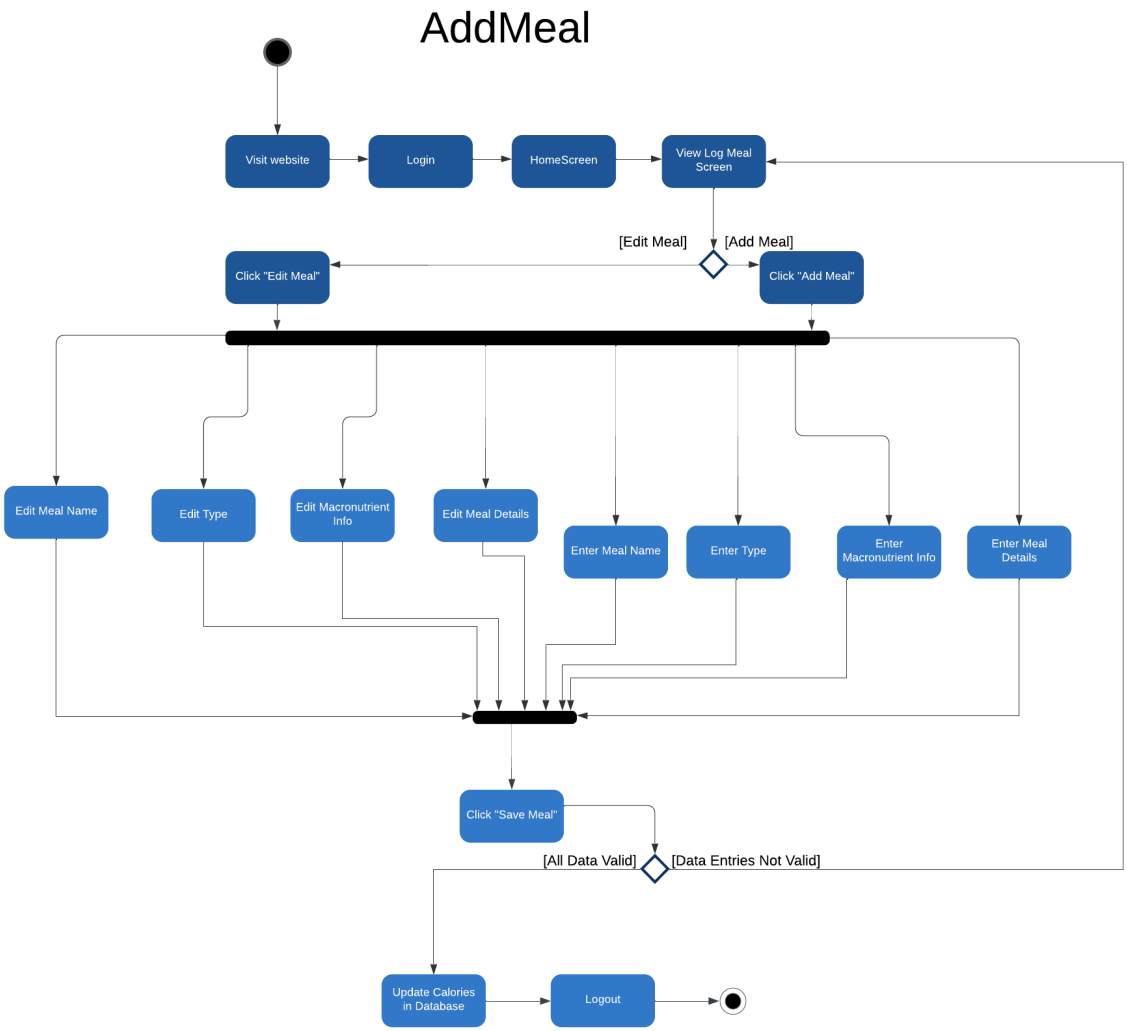
3.3.2 Processing sequence for Log Meal



3.3.3 Structural Design for Log Meal


MealFormScreen
-MealForm
-DisplayWorkoutForm()
1
Forms
-fields: string
-model
-SubmitForm()
MealForm
-fields: calories,carbs,protien,
time
-model: MealModel
-SubmitForm()
1
1
1
MealController
-MealForm
-getMapping(MealForm)
-postMapping(MealForm)
1
uses
1
MealModel
-MealModelFields
-UpdateMeals()

3.3.4 Key Activities



3.3.5 Software Interface to other components

The Meal Database supplies nutritional information for the logged food items, allowing the system to track calories and macronutrients. This data is integrated with the Nutrition Tracking System, which keeps a tally of the user's caloric intake and macronutrient distribution. The Database System stores all meal logs and related data, ensuring that progress is accurately tracked and past meal data is available to view by the user.

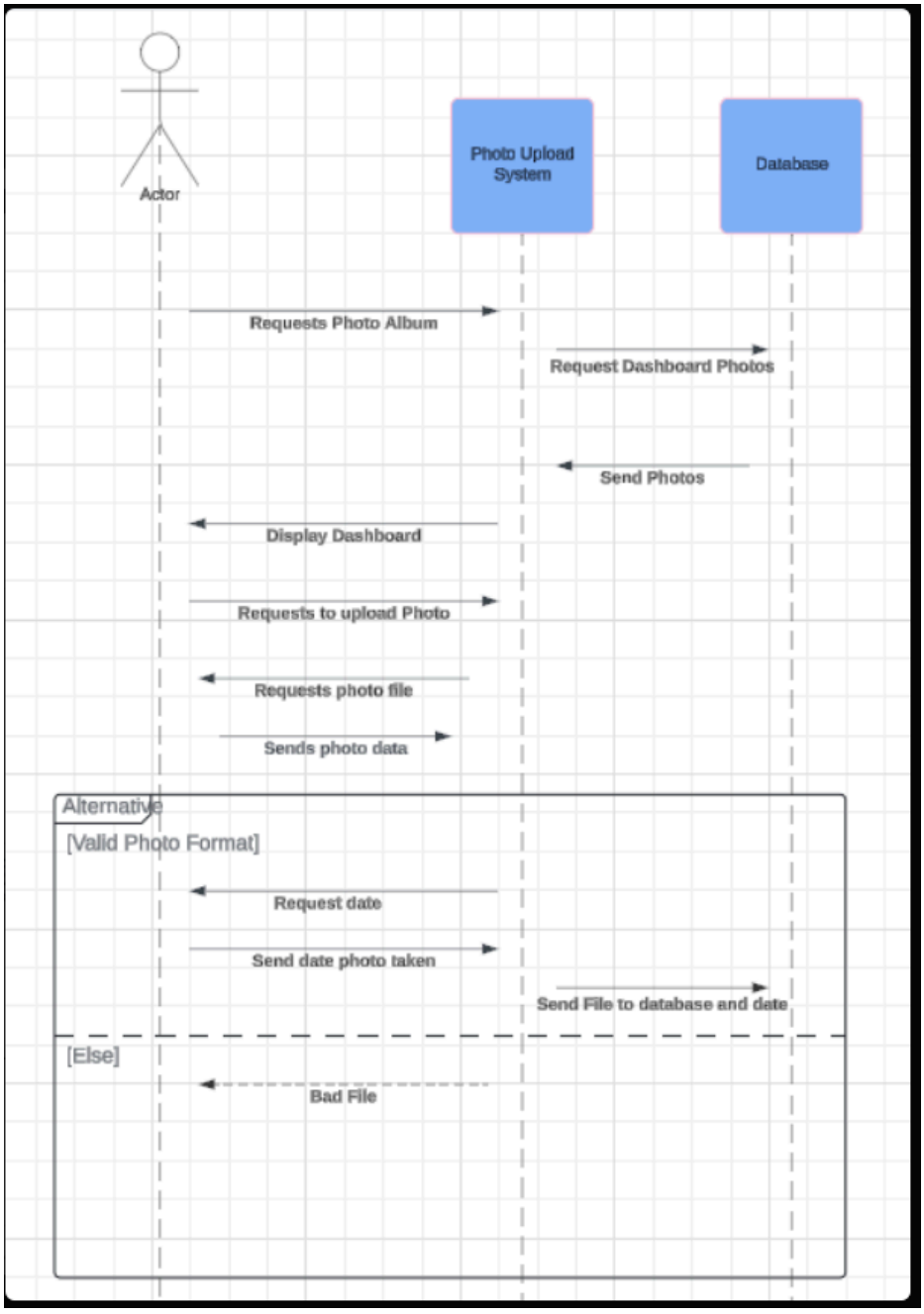
3.4 The Weightlifter Upload Photo

3.4.1 Upload Photo Use Case

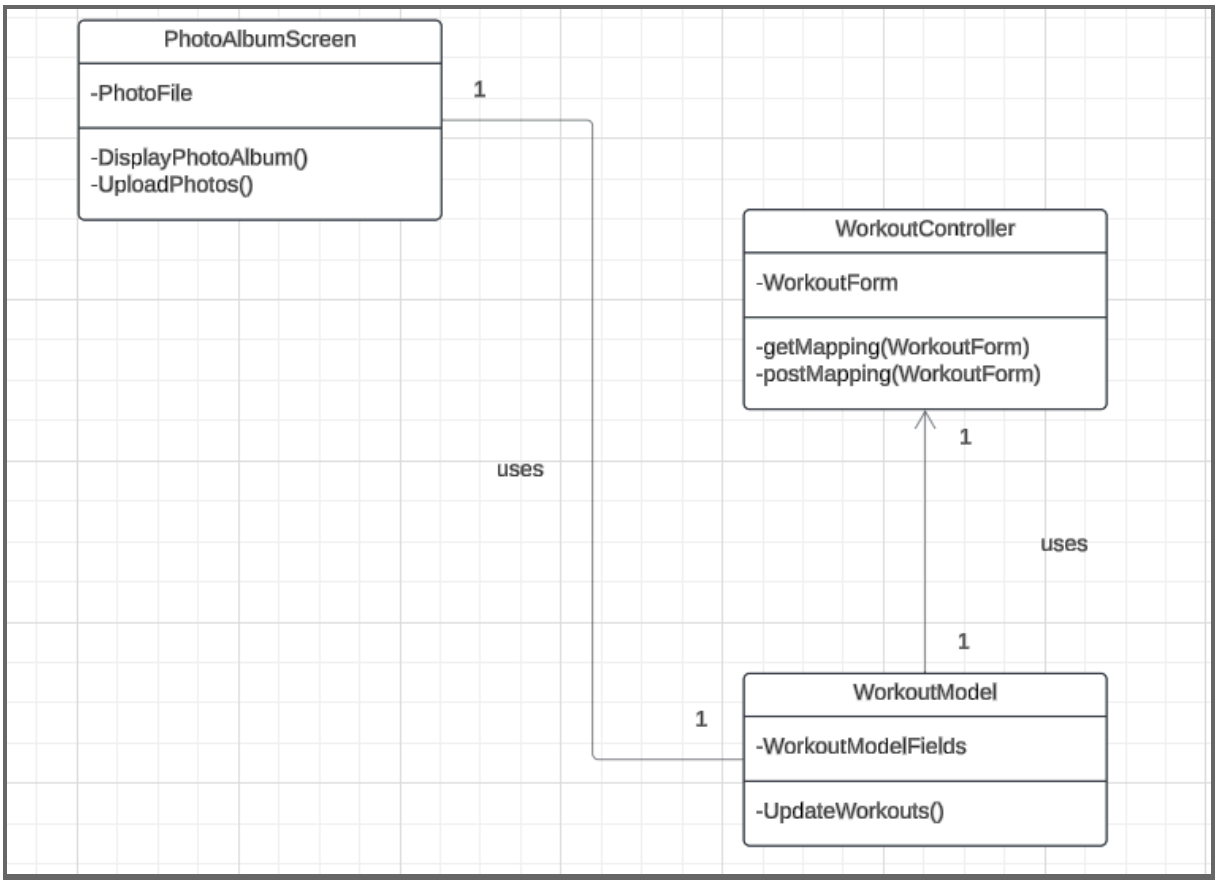
Use Case	Upload Photo
Goal in Context	Allows the users to upload photos of their workout to their album.
Scope	Database updates the photos the user uploads with the correct files and displays in an album format.
Level	Primary User-goal
Primary Actor	Fitness Enthusiast (User uploading photos)
Precondition	The user must be logged in to interact with the upload photos and view feature.
Minimal Guarantee	The database will store their photos and will be notified the photos are successfully uploaded.
Success Guarantee	User will view their current photo album or blank photo album screen if no photos are uploaded
Trigger	The user selects “upload photos” in their album view and can also view their photos.
Success Scenario	1. User logs in and an dashboard opens. 2. Use clicks on Photo Album button 3. User enters the photo Album page 4. There will be an option inside the page called, "Upload Photos". 5. User clicks on "Upload Photos". 6. User can only upload jpg and png files. 7. Once the user uploads their files, the system will give an update stating "Files successfully uploaded" and use presses "Close". 8. System creates an album view for user to view their uploaded photos. 9. User photos are updated and ready to view.
Extensions	1a. Invalid file: If the user uploads a wrong file that does not support in JSON for photos, the system prompts the user to upload the correct file.

	2a. Cancellation: If the user accidentally uploads and decides not to save changes, user can click on the upload folder and choose a different image. 3a. Photo Overview: Once the user uploads the photos, the system will create a photo album for signed in users to view.

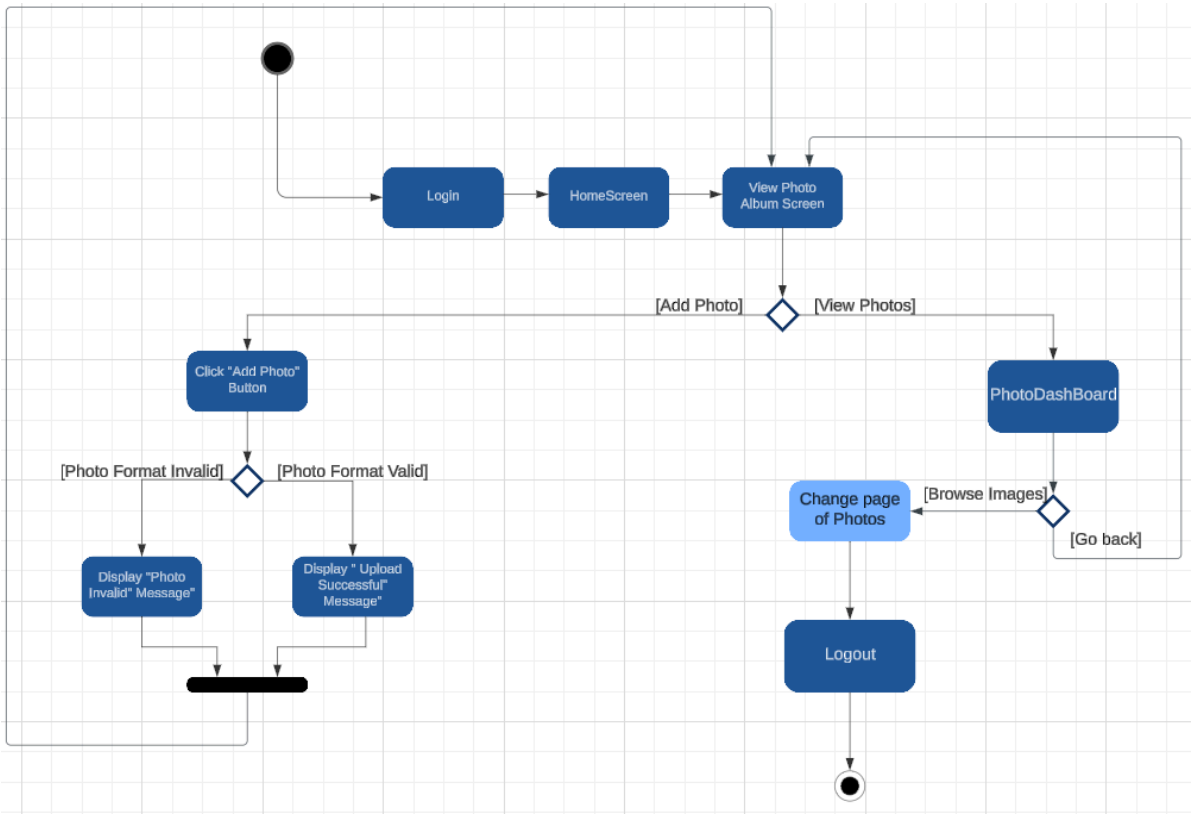
3.4.2 Processing sequence for Upload Photo



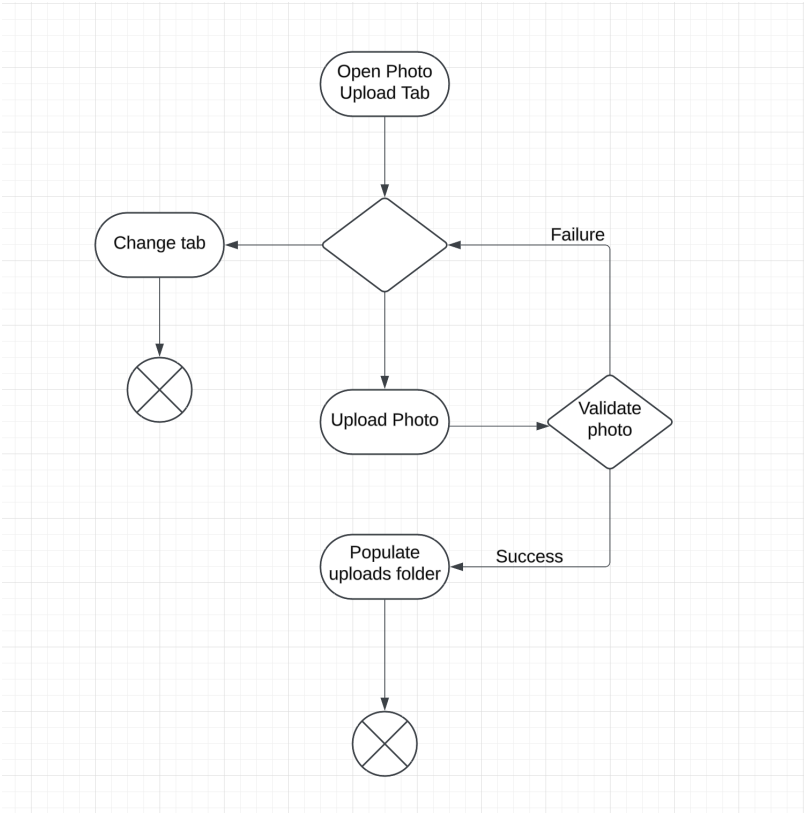
3.4.3 Structural Design for Upload Photo



3.4.4 Key Activities



3.4.4 Key Activities



3.4.5 Software Interface to other components

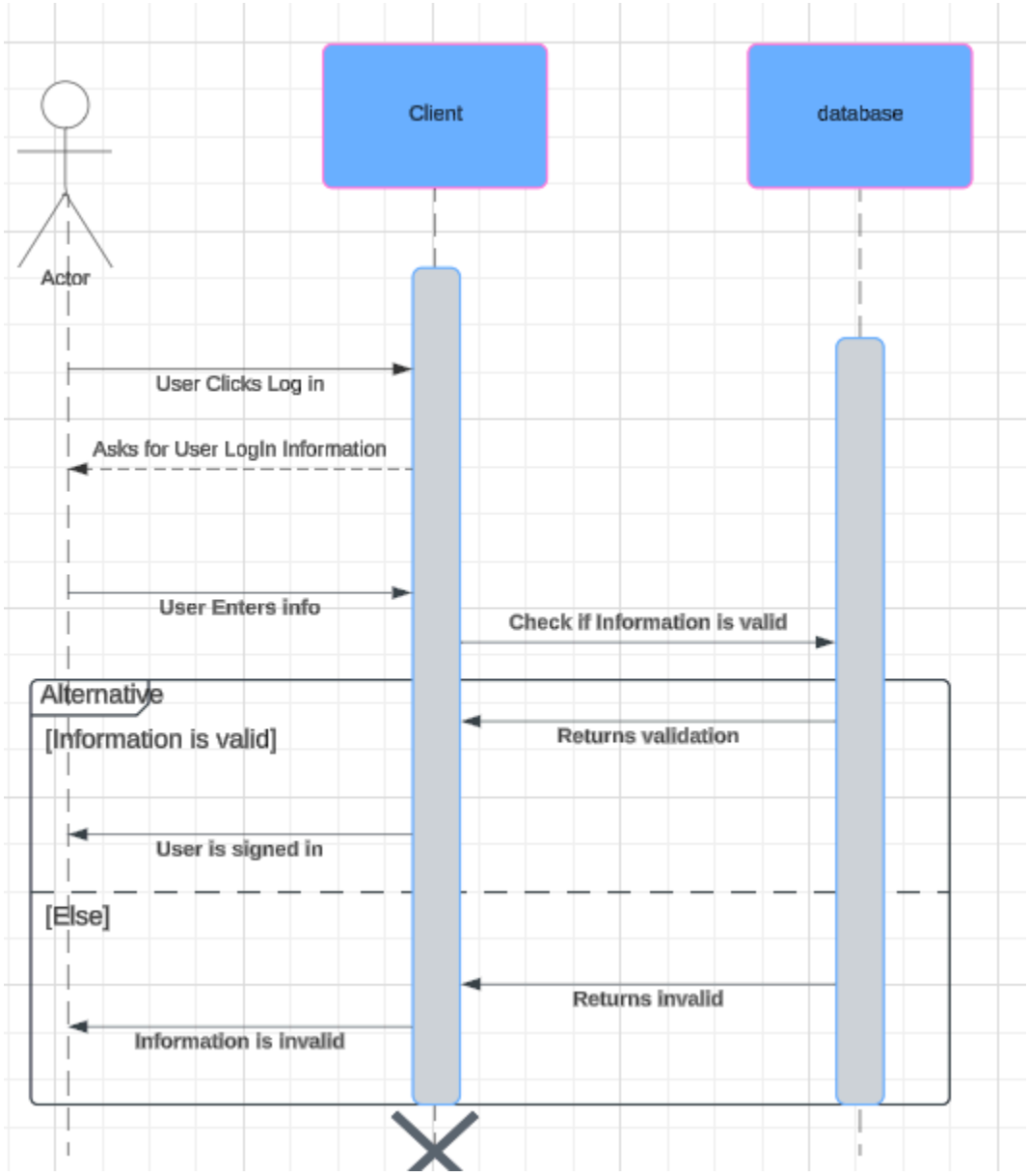
The Log Meal feature interfaces with several key components to ensure smooth operation and data synchronization. First, it interacts with the Authentication System, ensuring that only authenticated users can log meals. Once logged in, the Meal Database provides nutritional data for the food items the user enters, helping to calculate and log macronutrients like calories, protein, carbs, and fats. The User Profile Management system also plays a role, as it helps to store the user's personalized nutritional goals and preferences. This allows the meal logging feature to adjust the meal suggestions and track whether the user is meeting their specific fitness goals. The system then updates the Database System with the logged meal data, ensuring that the meal details are stored and tracked for progress monitoring. Lastly, the User Interface (UI) provides the environment for interacting with the system. It presents the meal logging feature, enables users to input data, and provides feedback on successfully logged meals or any errors, ensuring a seamless experience for the user.

3.5 Weightlifter Log In

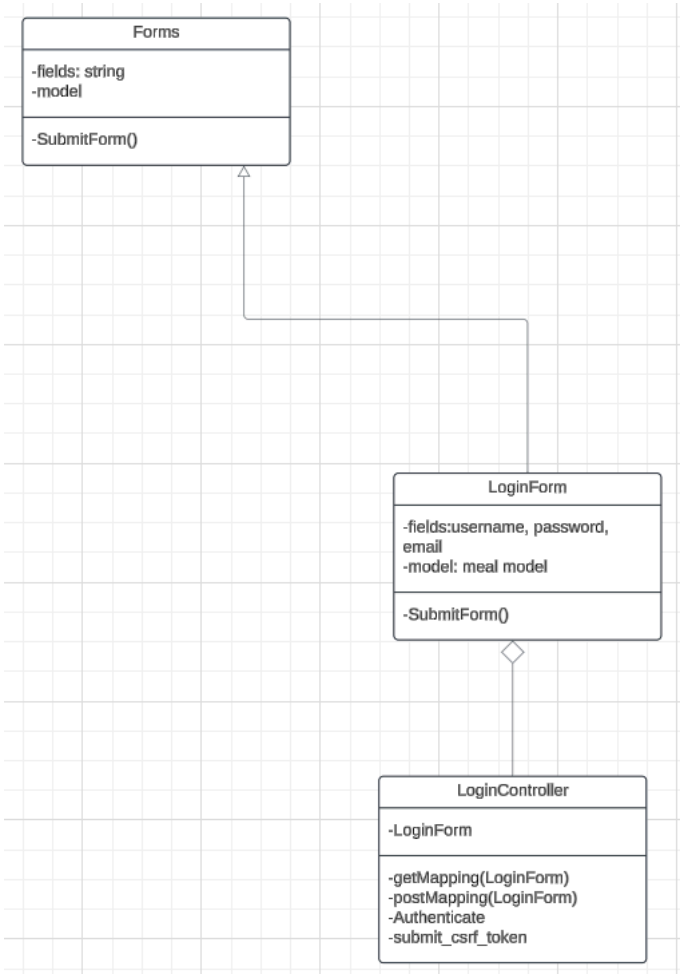
3.5.1 Weightlifter Log In Use Case

Use Case	Log in
Goal in Context	Verify login information
Scope	Security of user accounts and personal data during the login process.
Level	Primary User goal level
Primary Actor	Returning user
Precondition	The user must have already signed up for an account.
Minimal Guarantee	All user input must be securely stored, and the system will provide feedback if the login fails.
Success Guarantee	The user is successfully logged into their account, granting access to the application.
Trigger	“Login” button is pressed
Success Scenario	1. 1. Users enter their ‘Username’. 2. 2. User enters their ‘Password’. 3. 3. System checks if the ‘Username’ is valid. 4. 4. System checks if the ‘Password’ matches the valid ‘Username’. 5. 5. If both are valid, the user is logged into their account.
Extensions	1a. Invalid Username: If the ‘Username’ is invalid, the system displays an error message and prompts the user to re-enter their username. 2a. Invalid Password: If the ‘Password’ is incorrect, the system displays an error message indicating the password is invalid and prompts the user to re-enter their password. 3a. Forgot Username: If the user selects the "Forgot Username" option, the system prompts for recovery options to retrieve their username. 4a. Forgot Password: If the user selects the "Forgot Password" option, the system prompts for recovery options to reset their password.

3.5.2 Processing sequence for Log In

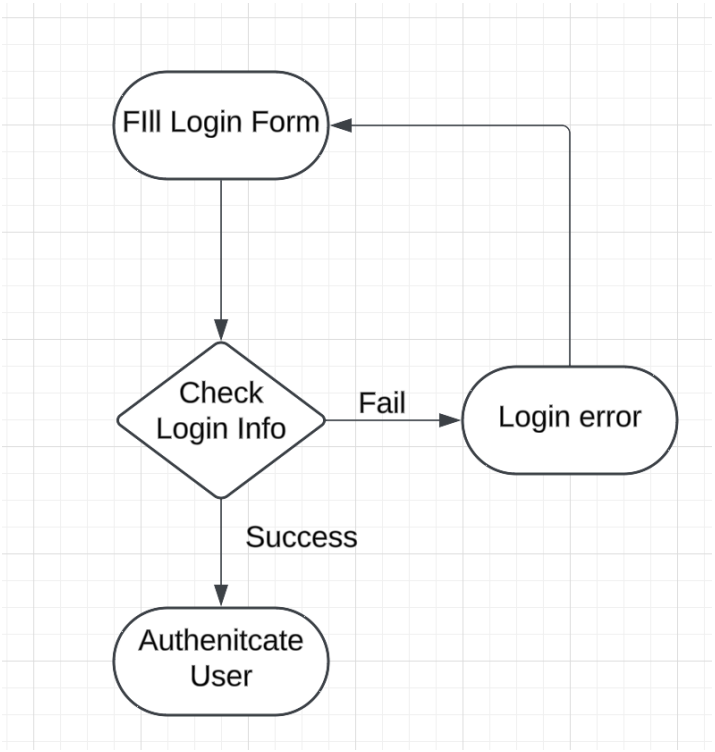


3.5.3 Structural Design for Log in



For login authentication we used spring security called SecurityConfig to do authentication and AppUserCrudServive to do the user authentication that are linked to the user_table in the database.

3.5.4 Key Activities



3.5.5 GetGainz to other components

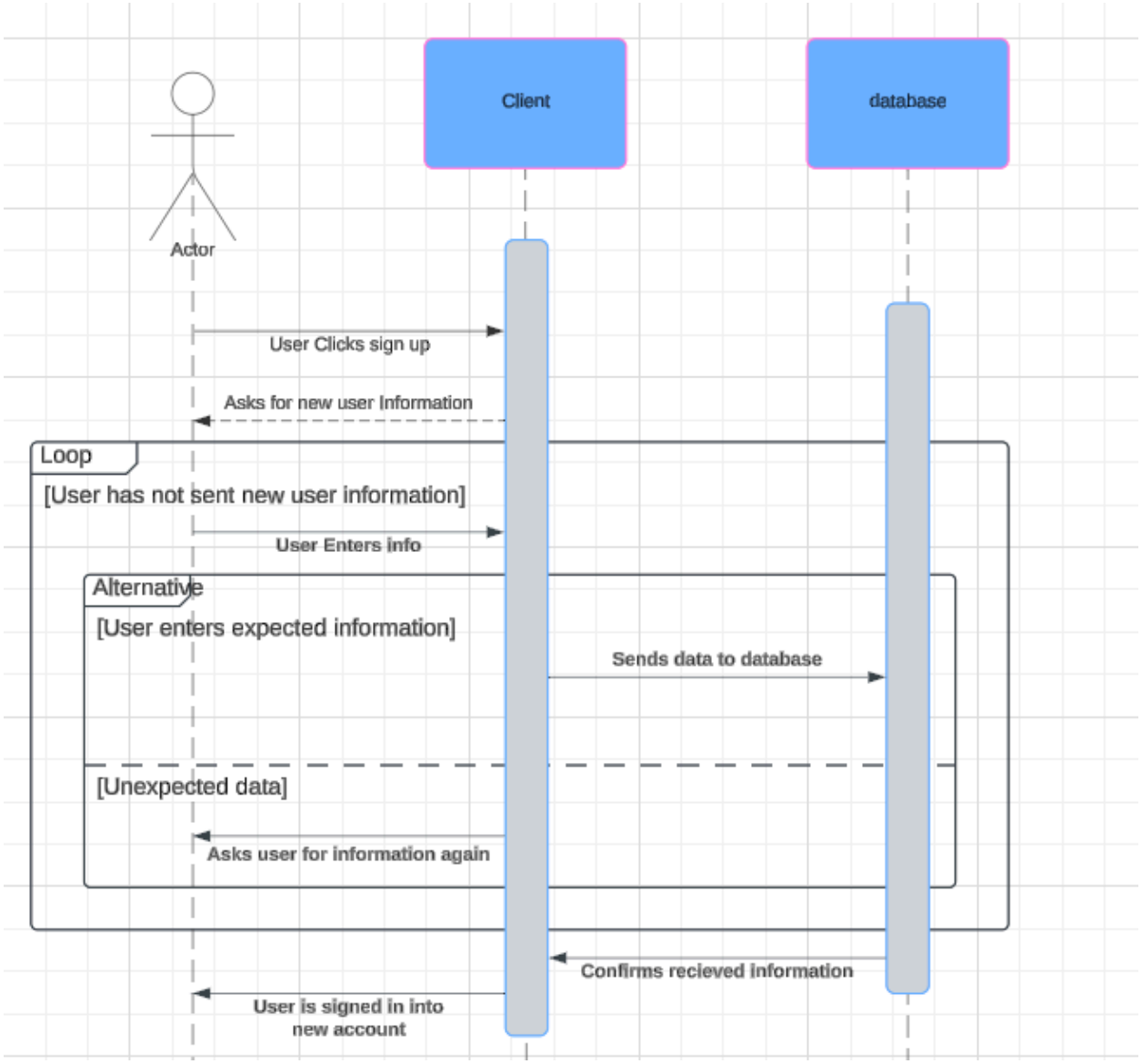
The login interface is for the user to input their details that they have entered when users have created an information. The interface will use an authentication to validate the user's information in the database and respond back to the interface that it's the correct user and they are good to move on in the application.

3.6 Weightlifter Create Account

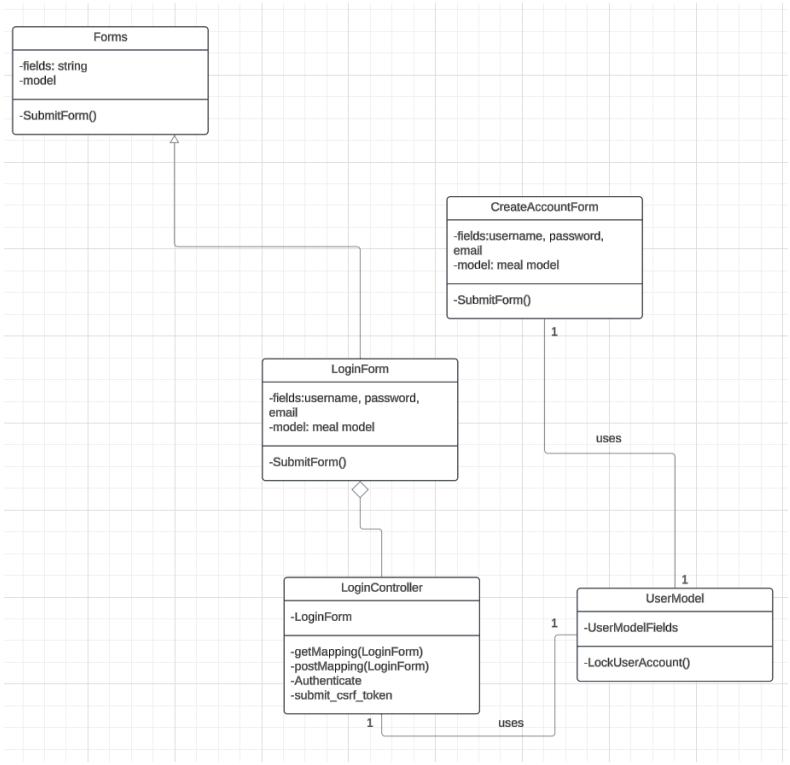
3.6.1 Create Account Use Case

Use Case	Create Account
Goal in context	Allows users to create an account
Scope	Database creates a new user with their log in information
Level	Primary High-level
Primary Actor	Fitness Enthusiast
Precondition	The user must have access to the application.
Minimal Guarantee	User fails to create account
Success Guarantee	User Creates account
Trigger	The user selects the “Sign up” option in the log in page
Success Scenario	1. User navigates to the “Sign up” section. 2. The system displays sign up screen 3. User enters new account information 4. The system validates the inputted data. 5. The system creates account for the user 6. The system is signed into this new account
Extensions	4a. Invalid Data: If the user enters invalid data (e.g., an invalid email format), the system prompts the user to correct the entries.

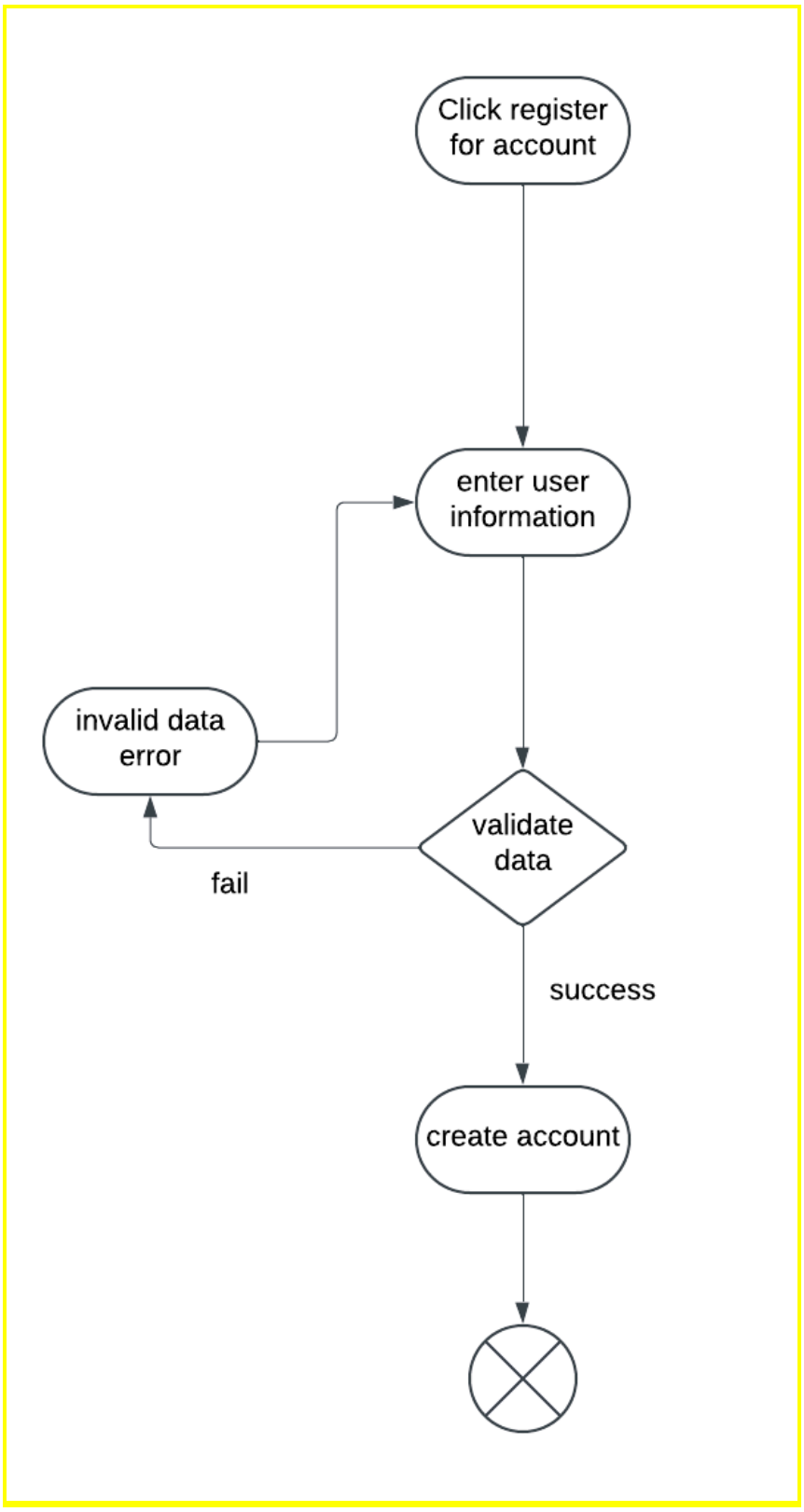
3.6.2 Processing sequence for Create Account



3.6.3 Structural Design for Create Account



3.6.4 Key Activities


Click register for account
enter user information
invalid data error
validate data
fail
success
create account

3.6.5 Software Interface to other components

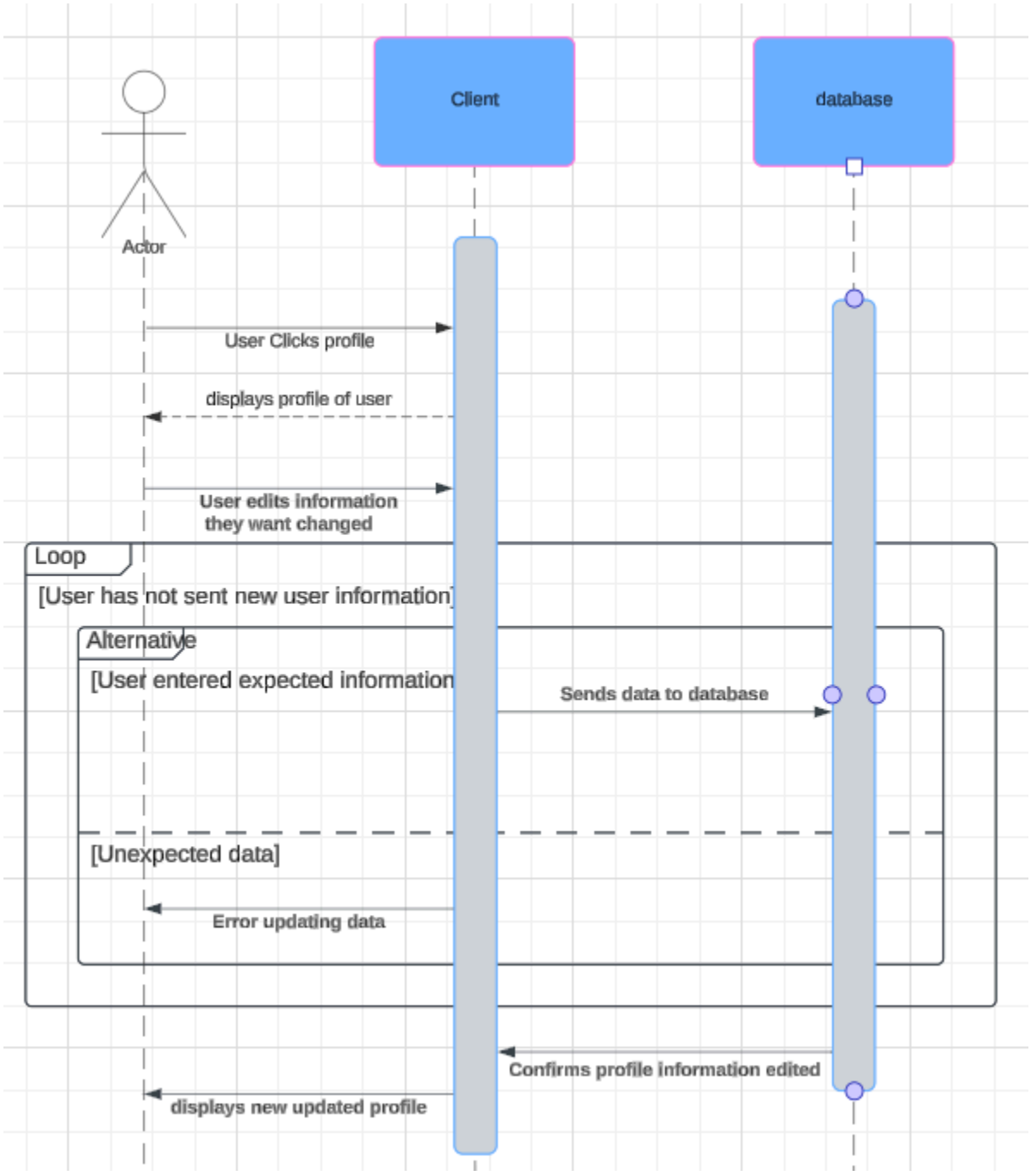
The create account interface is for first time users and new users to create an account in application for the database to store their information and react when the user interacts with the interface. The users created an account and will be sent to the database, which will create a row for the user in our user database table and validates the user have inserted all the information the database needs. Everytime the user creates an account it validates if the user already have an account with the GetGainz and inform them.

3.7 Weightlifter Edit Profile

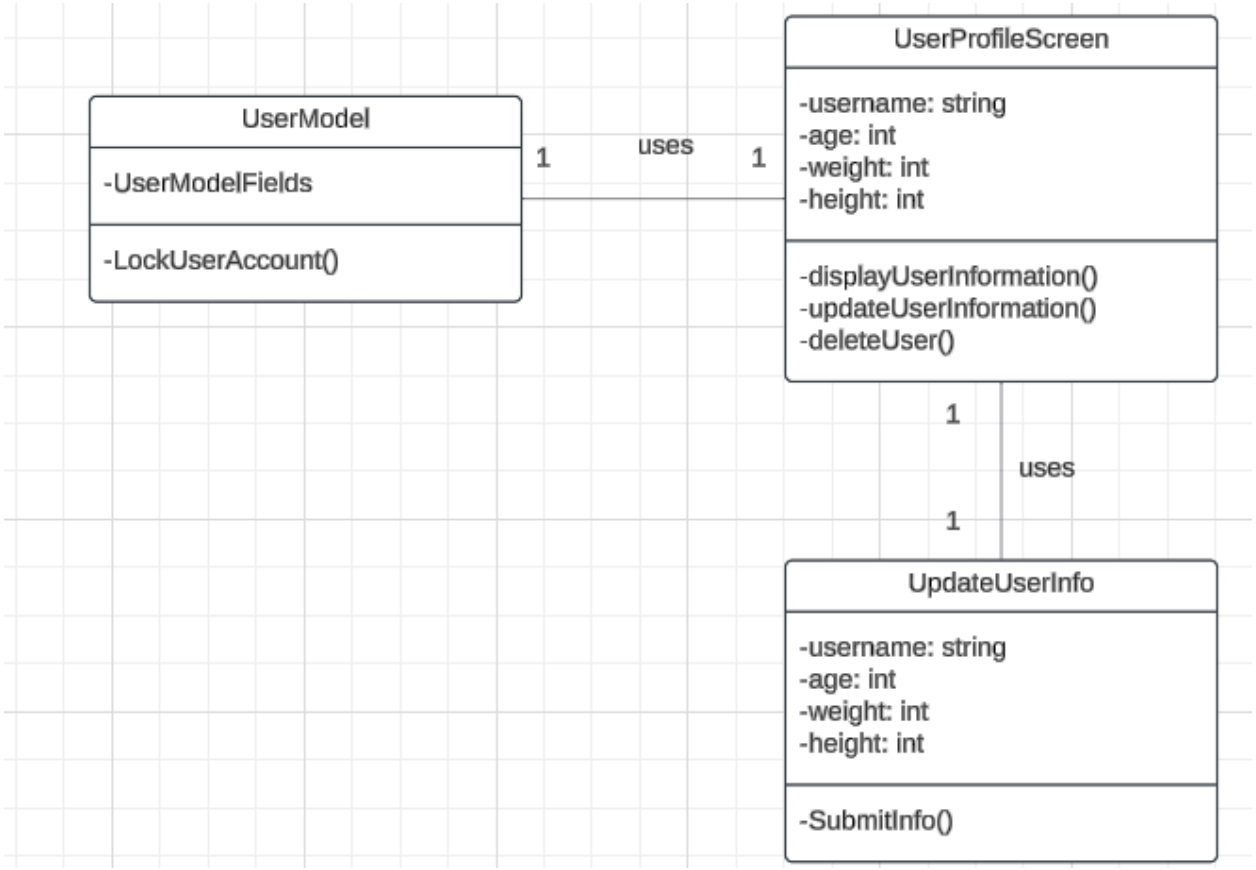
3.7.1 Edit Profile Use Case

Use Case	Edit Profile
Goal in context	Allows users to change their profile data to keep their information current and relevant.
Scope	Database updates to reflect changes made to user profile information.
Level	Primary High-level
Primary Actor	User (individual managing their account information)
Precondition	The user must be logged into their account to access profile editing features.
Minimal Guarantee	User will be able to view Edit Profile screen with fields to modify password
Success Guarantee	The user's profile data is successfully updated in the database, and the system reflects these changes in the user interface.
Trigger	The user selects the “Edit Profile” option in their account settings.
Success Scenario	1. User navigates to the “Edit Profile” section. 2. User updates their profile information (e.g., name, email, fitness goals). 3. User submits the changes by clicking the “Save” button. 4. The system validates the input data. 5. The system updates the user’s profile in the database. 6. The system displays a confirmation message indicating that the profile has been successfully updated. 7. The user’s updated information is displayed in their profile.
Extensions	1a. Invalid Data: If the user enters invalid data (e.g., an invalid email format), the system prompts the user to correct the entries. 2a. Profile Picture Update: If the user chooses to update their profile picture, the system allows the user to upload a new image and checks for valid file types. 3a. Cancellation: If the user decides not to save changes, they can cancel the edit, and no changes will be made. 4a. Password Update: If the user opts to change their password, the system prompts for the current password and the new password, then validates the input before making changes.

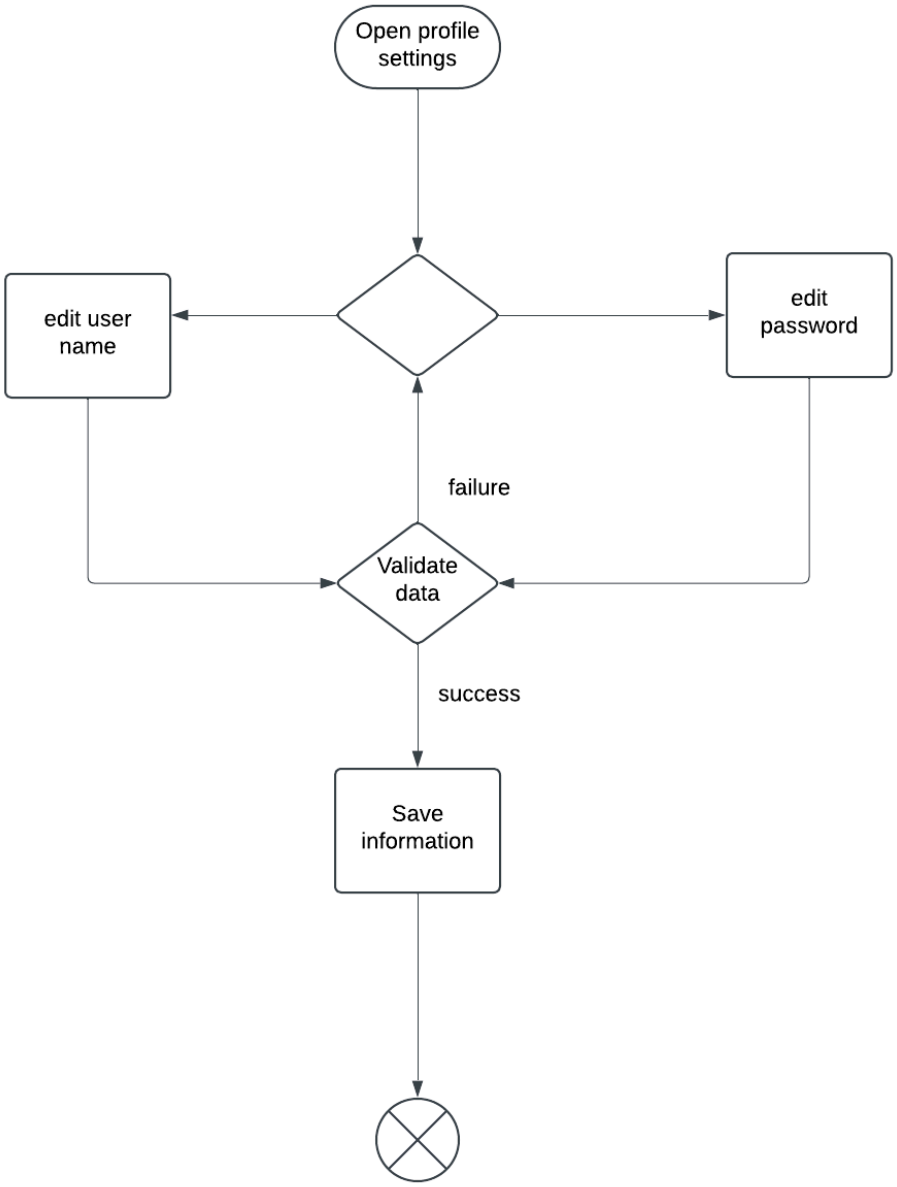
3.7.2 Processing sequence for Edit Profile



3.7.3 Structural Design for Edit Profile



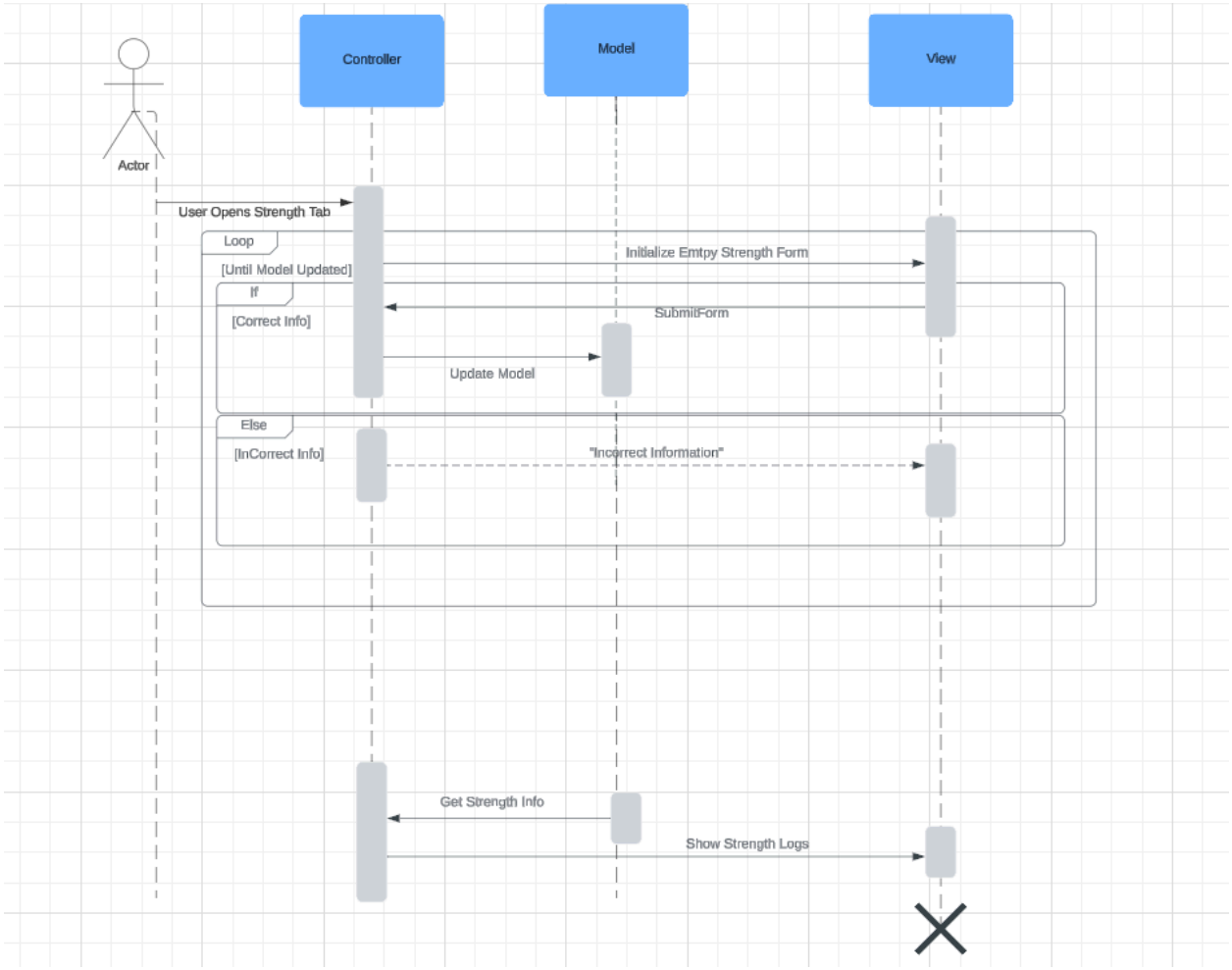
3.7.4 Key Activities



3.7.5 Software Interface to other components

The User Profile Management system stores and retrieves user data, allowing users to update their personal information. Once changes are made, the updated data is sent to the Database System for storage, ensuring that the user's profile is kept up-to-date. The feature also integrates with error handling and validation systems to ensure that the data entered is correct, alerting the user if there are issues. The User Interface provides the environment for the user to interact with the system and view or edit their information.

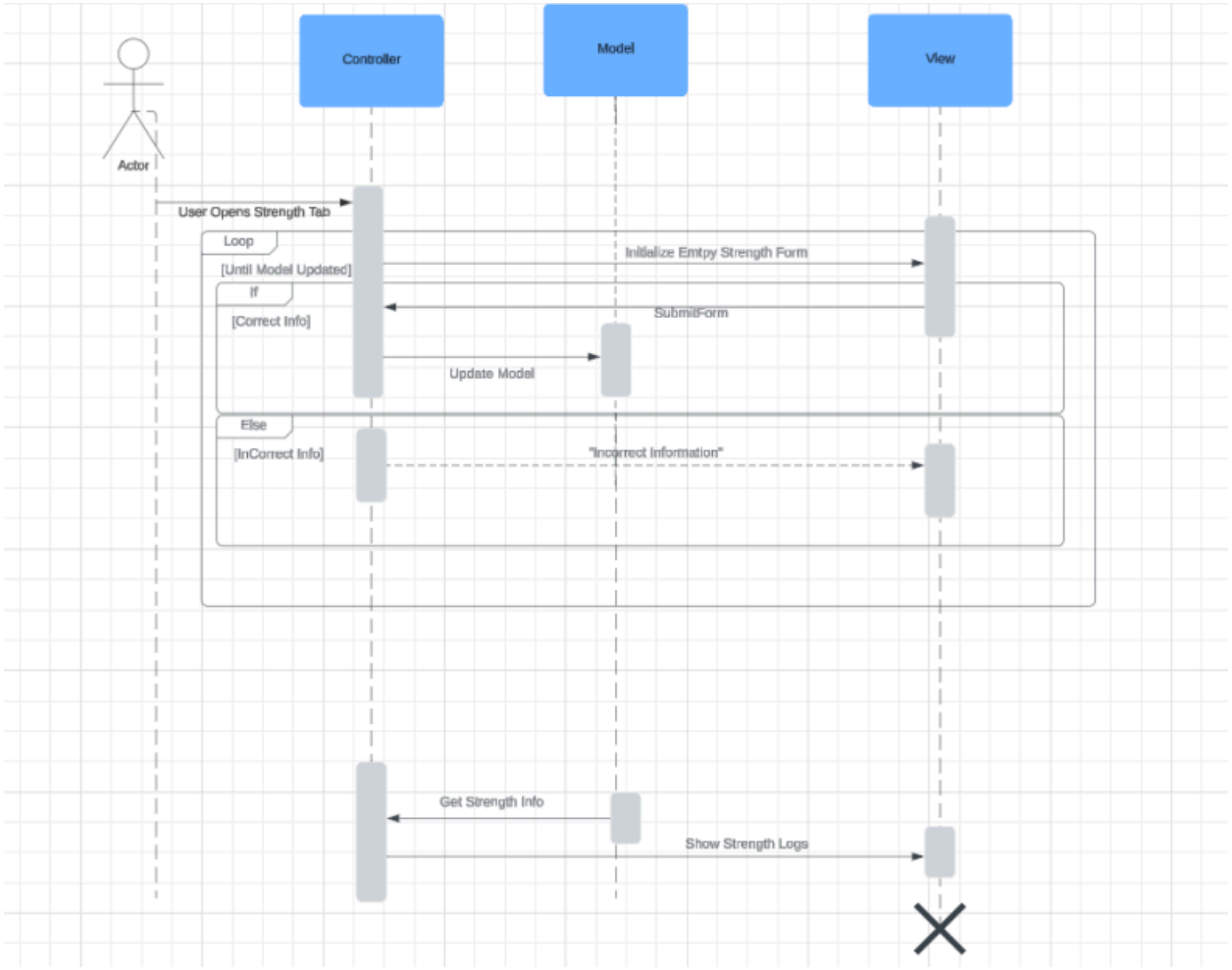
3.8 Weightlifter Log Strength Progress



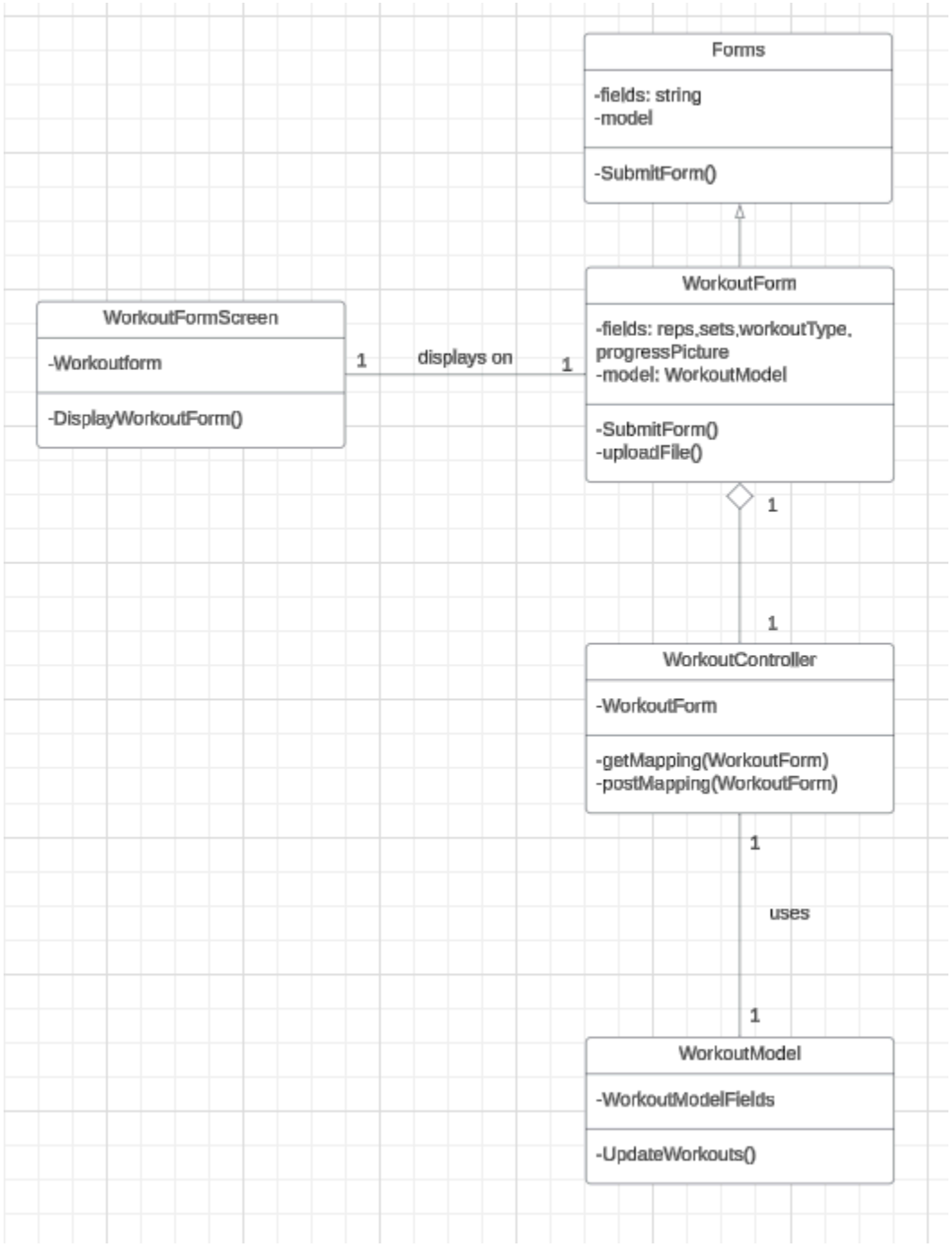
3.8.1 Weightlifter Log Strength Use Case

Use Case	Log Meal
Goal in context	Allow users to log their meals and track their nutritional intake over time.
Scope	Logging meals and calculating macronutrients to help users track their nutrition and adjust as needed.
Level	Primary User goal level
Primary Actor	Fitness Enthusiast (user logging meals)
Precondition	The user must be logged in to access meal logging features.
Minimal Guarantee	The system will store meal data, calculate macronutrients, and update the user's meal log.
Success Guarantee	The user's meal data is successfully logged and the meal log is updated with calories and macronutrients.
Trigger	The user selects the option to log a meal in the meal tracking interface.
Success Scenario	1. User logs into their account. 2. User navigates to the meal logging section. 3. User enters the food types consumed and their quantities. 4. The system retrieves the nutritional data (calories, protein, carbs, fats) for the entered foods. 5. The system calculates the total calories and macronutrients for the meal. 6. The system stores the meal data in the user's log. 7. The meal log is updated, and the user receives insights on their nutritional intake.
Extensions	1a. Invalid Data: If the user enters incorrect or missing food data, the system prompts for correction. 2a. Incomplete Meal: If the system cannot find nutritional data for a food item, the user is prompted to enter the data manually or choose an alternative food item. 3a. Meal Overview: After logging, the system displays a summary of the meal's nutritional breakdown.

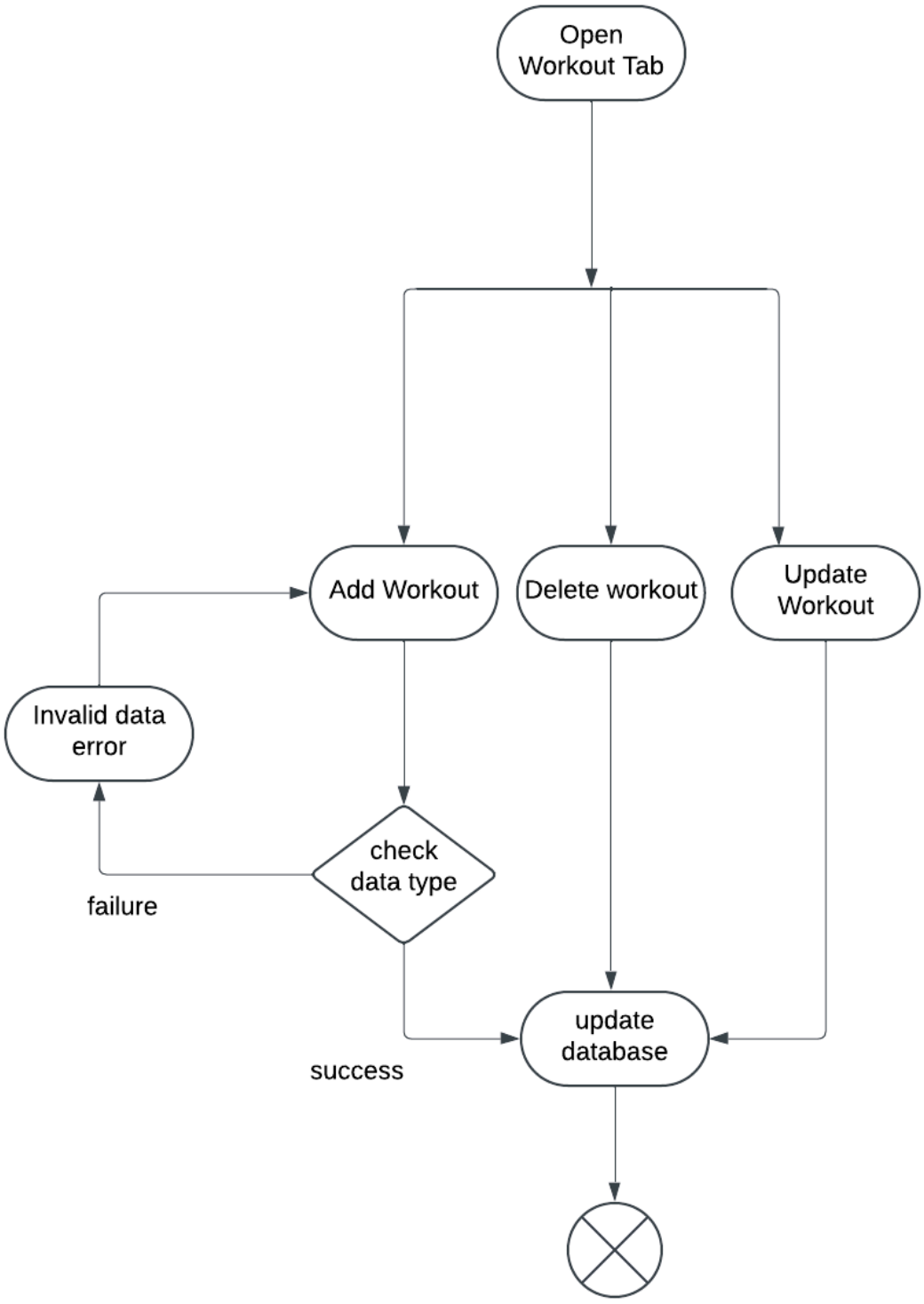
3.8.2 Processing sequence for Log Strength



3.8.3 Structural Design for Log Strength



3.8.4 Key Activities



3.8.5 Software Interface to other components

The Log Strength Progress feature interfaces with several components to track and store strength data. The Authentication System ensures only logged-in users can log their progress. The User Profile Management system stores the user's fitness goals and exercise history, helping customize the strength tracking to their needs. The Database System saves the strength data entered by the user, while the Progress Tracking System generates graphs to visualize improvements.

4 User interface design

4.1 Interface design rules

The GetGains interface design is accessible and an easy-to-use interface. There is consistency in buttons, fonts, color, and icons across the entire software. The color choices will maintain the same color scheme across all user interfaces with engaging fitness related graphics as part of some UI backgrounds. Warnings related to invalid entries are clear to the user through the interface. There is an easy-to-access navigation bar which allows the user to have a central space where they can see all options of the functionalities. The error messages are short and helpful.

4.2 Description of the user interface

The main navigation bar is designed to be the center point of navigation. There are buttons on a navigation bar that lead to various screens based on what the user wishes to use and access. Upon login, users are directed to a screen that allows them to view the current day off the calendar. The Strength Tracking interface will include a section for workout recording, mainly for lift and set data. Profile Settings is the area where one can edit their password. The Log Workout screen allows the user to add personalized workouts that they can add statistics for to fulfill logging functionality.

4.2.2 Fitness Enthusiast Login Page

The Fitness Enthusiast Login Page is designed to provide users with a secure method of accessing their personalized fitness journey. Upon landing on the page, users are prompted to enter their credentials, including their username and password, to authenticate their account. The design will have easy-to-read text fields for input and a prominent login button for submission. For new users, a registration link is available, guiding them through the sign-up process to create an account. This login page acts as the gateway to the user’s customized dashboard, where they can access all features.

4.2.2.1 Screen Images

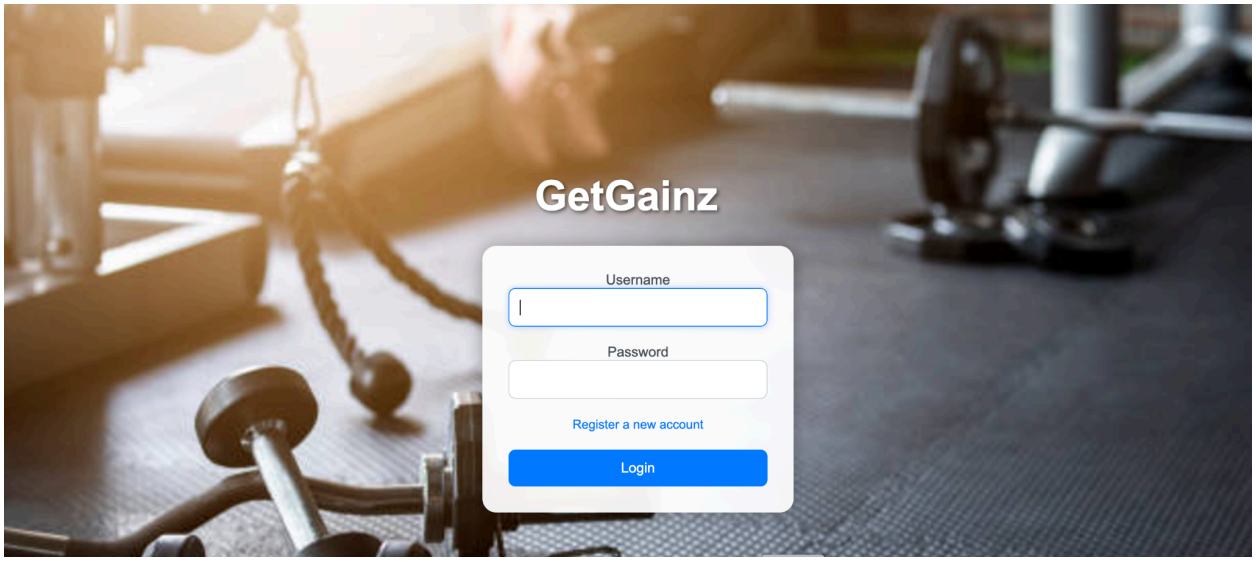


Figure 2 - Login Page

4.2.2.2 Objects and Actions

Key objects include: the Username Field, where a user fills in their registered username; the Password Field, where he or she provides their password for the purpose of authenticating his or her account; and the Login Button, where users submit their credentials for the intent of gaining access to their account. This will include the following actions: Login and Register New Account for new users who will have to sign up.

4.2.3 Fitness Enthusiast Registration Page

The Fitness Enthusiast Registration Page provides an interface for new users to create an account within the GetGainz app. This feature allows a fitness enthusiast to enter essential information, such as username, password, and email to establish their account. Upon submitting their details, the system validates the provided information, ensuring that the username is unique and that the password meets security requirements. Once validated, the account is created, granting the fitness enthusiast access to the app features. If any input errors or validation issues arise, the page displays helpful prompts to guide the user in completing registration accurately.

4.2.3.1 Screen Images

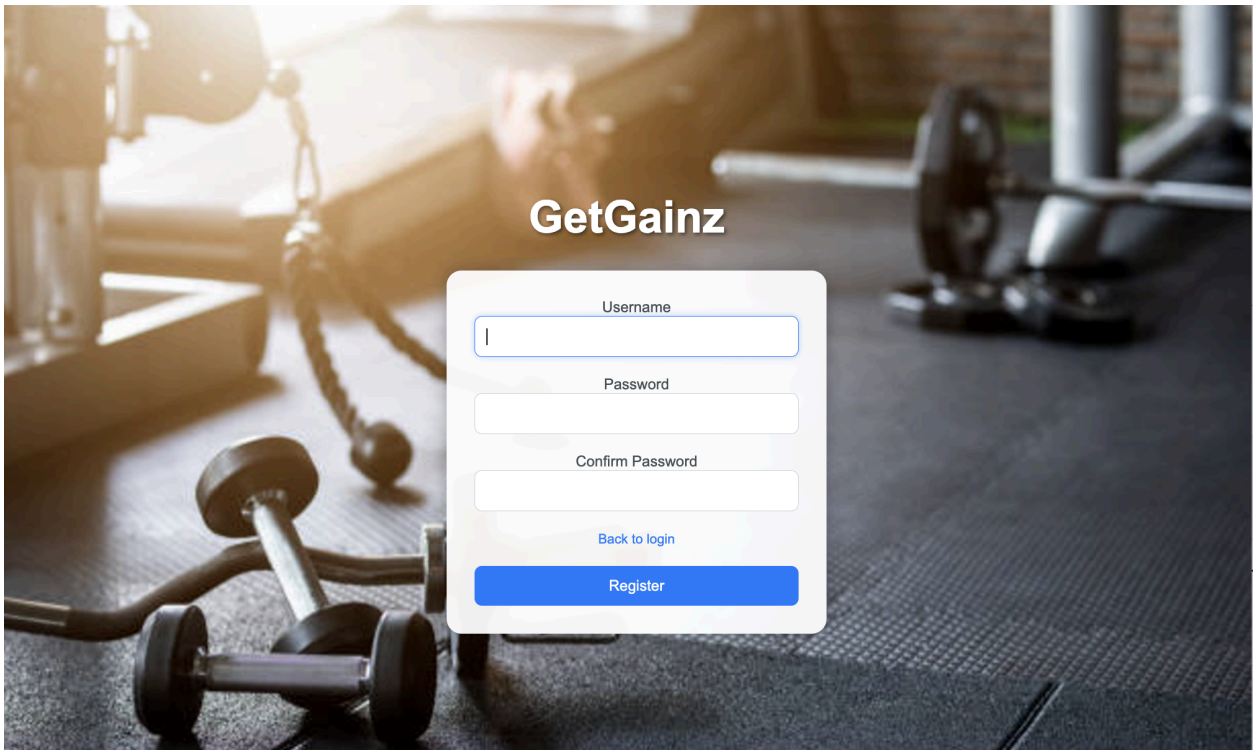


Figure 3 - Sign Up Page

4.2.3.2 Objects and Actions

The Registration allows viewing of a sign-up page for new users who wish to create an account; it takes them through the registration process, keeping the steps simple. This is where the message of error will appear if the inserted credentials are wrong, so that at least the user will be informed about something going wrong in his/her login process.

4.2.4 Fitness Enthusiast Workout Logging

The Fitness Enthusiast Workout Logging feature allows users to track and log their exercise routines, helping them monitor progress and stay consistent with their fitness goals. Users can input specific details about each workout session, such as the type of exercise (e.g., strength training, cardio, flexibility), duration, intensity, and any additional notes or goals associated with the workout. The interface is designed to be user-friendly, providing easy input fields for each workout detail and an option to select from predefined exercises or manually enter custom activities. The workout log can be organized by date, allowing users to view past workouts and track improvements over time. This enables users to track their performance and notice trends or improvements.

4.2.4.1 Screen Images

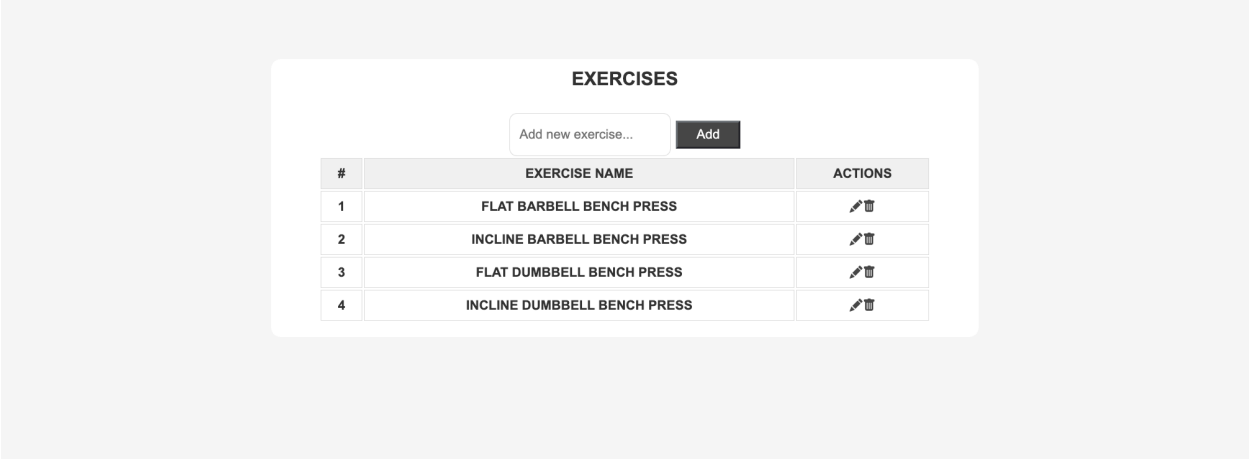


Figure 5 - Workout Log

4.2.4.2 Objects and Actions

The main objects involve a Workout Type Selector that allows users to select the type of exercise they have performed, such as strength training, cardio, flexibility, among others. The Workout Details Input Fields allow users to input relevant information with respect to duration, intensity, sets, reps, and weights lifted according to the given workout type. The Date Picker is for selecting the specific date for the entry of a workout. The actions will include Log Workout for saving the entered workout details and Edit Workout to update an existing entry or modify it.

4.2.5 Fitness Enthusiast Account Info

This will provide a page to the user that allows them to change their password. They can edit this information so that their account will be updated with the new designated password. The new password must be between 4-15 characters, which is checked upon submission.

4.2.5.1 Screen Images

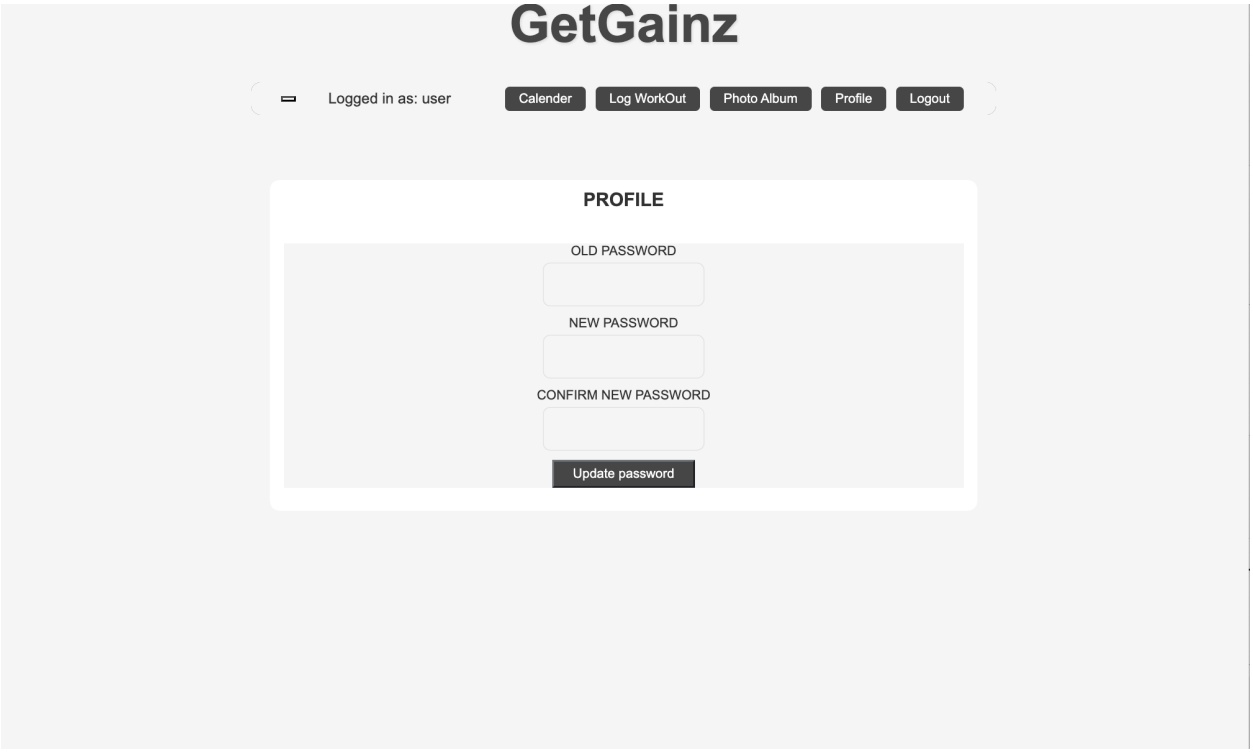


Figure 6 - Account Info Page

4.2.5.2 Objects and Actions

Key objects include the Personal Information Display, which shows the user's username, with options to edit or update these details. Actions include Edit Personal Info to update name, email, or username and Update Password to change login credentials.

4.2.6 Fitness Enthusiast Calendar

The Fitness Enthusiast Calendar feature will be useful to the user in planning, scheduling, and tracking workouts. This feature gives a general overview of the user's workout routine and progress. This calendar, showing the month or week, provides space for each day to record the details of a workout. Also, it will allow the user to insert the workouts for any particular day by setting the date and after that specifying an activity, such as strength, cardio, and other such information like duration and intensity, and notes for that session.

4.2.6.1 Screen Images

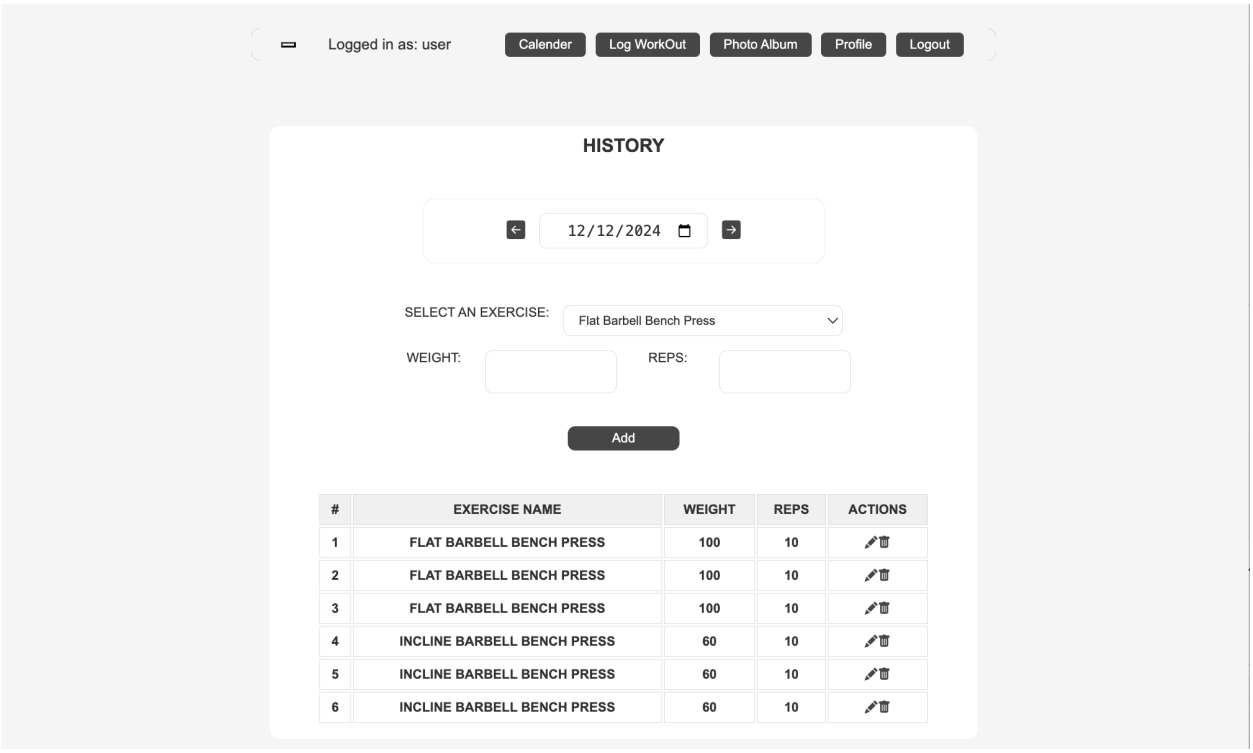


Figure 7 - Calendar Page

4.2.6.2 Objects and Actions

Major objects involved in this project include View Calendar, which would display days either in month or week format; a user can click on any date and schedule workouts accordingly. The Workout Entry Field is the field where the user fills in all details related to his/her workout, like type of exercise, duration, and any specific goals or notes for that session. Actions include Add Workout to input a new workout for a selected date and Edit Workout to modify details of an already existing workout.

4.2.7 Fitness Enthusiast Photo Album

The Photo Album makes it easy for users to track their journey in photos over time. This section gives the user the ability to add, organize, and view photos regarding their progress. The interface is designed for ease of use, enabling users to upload photos directly from their device.

4.2.7.1 Screen Images

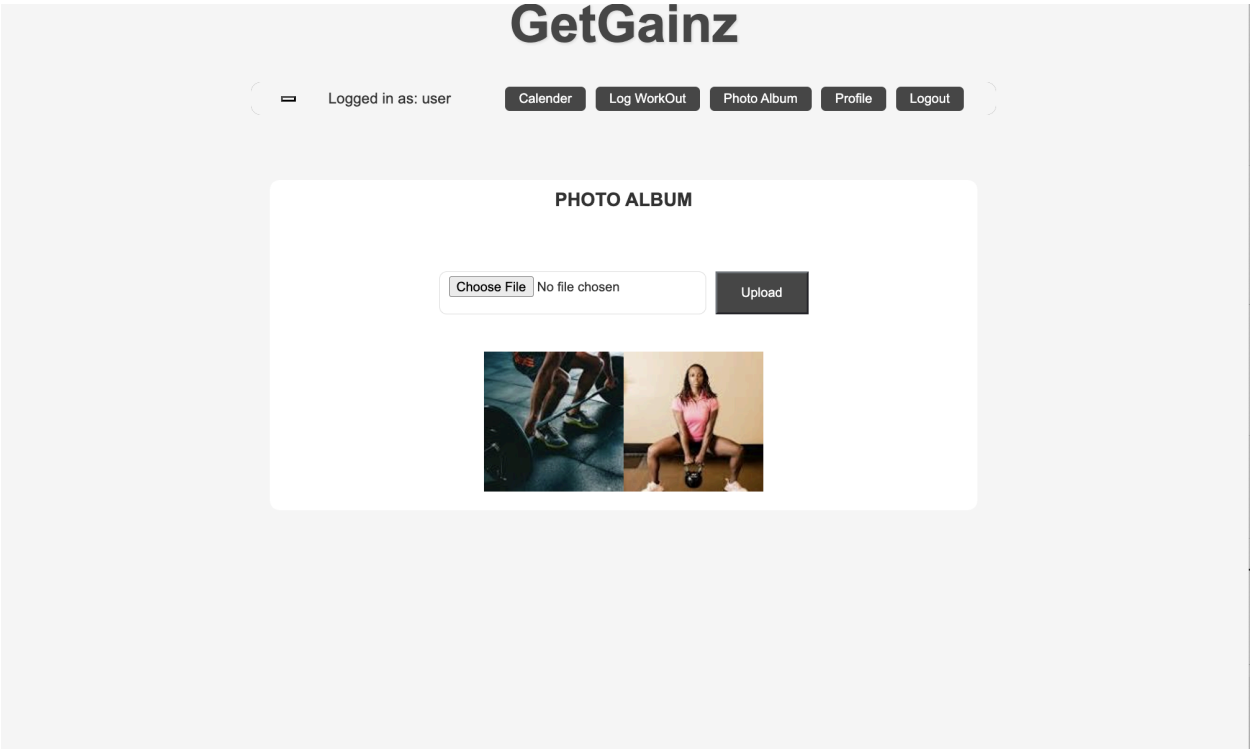


Figure 8 - Photo Album

4.2.7 .2 Objects and Actions

The objects and actions in the Photo Album feature are designed to support users in organizing their visual fitness progress. Key objects include the Photo Upload Button for adding new images directly from a device. The photos are displayed in a grid format. Large files are not accepted for submission.

5 Restrictions, limitations, and constraints

- Java 11
- The information storage will be an H2 database
- Maven must be used for the project build
- Spring Boot version 2.7.4

6 Testing Issues (SLO #2.v)

Test strategy and preliminary test case specification are presented in this section. Additional test cases are located in the Appendix section in spreadsheet format.

6.1 Types of tests

Performance Test – for example, to ensure that the response time for information retrieval is within an acceptable range. You typically should provide a specific performance bounds. For example, the search process should not take longer than 30 seconds.

(1). **Accuracy Test** – for example, to determine if queries return the expected results.

(2). **User Interface Test** – for example, to make sure the user interface is clear and easy to use with all types of users. Unfamiliar users can use the interface with minimal instruction and achieve the desired results.

(3). **Security test** – for example, to ensure that users can only perform the tasks specified for their user group

(4). **Repeatability Test** – for example, the software returns the same result for repeated queries.

6.2 List of Test Cases

1. Login - invalid password /
2. register account requirement - password size must be between 4-15… /
3. username not found (wrong username) /
4. calendar - reps and weight doesn't accept char /
5. add new exercise - character limit /
6. Cannot delete an exercise from logWorkOut if it is being used in calendar. /
7. please fill out this field - no entry for exercises /
8. cannot upload files that are too large for photo Upload - only accepts jpeg, png, webg. Does not accept pdf, docs, HEIC /
9. Cannot change password with an incorrect current password.

Test Type	**Accuracy Test**
Testing range	User login feature
Testing Input	Username and Wrong Password
Testing procedure	Enter username and wrong password Click Login
Expected Test Result	Prompt “Invalid password”
Testers	Nick Brodsky
Test result	Passed

Test Type	**Accuracy Test**
Testing range	Register Account feature
Testing Input	Username and 2 digit password
Testing procedure	Input username and create password of 2 digits Click Login
Expected Test Result	Prompt “Password size must be between 4-15 characters ”
Testers	Noya, Carlos, & Nick
Test result	Passed

Test Type	Accuracy Test

Testing range	Upload Photo Album
Testing Input	Upload a HEIC file or the image width/height > 150px Click Upload
Testing procedure	Upload a HEIC file or the image width/height > 150px Click Upload
Expected Test Result	Prompt “Upload failed due to incorrect file upload or the size is too large.”
Testers	Cherishma Jalaparti
Test result	Passed

Test Type	Username not found
Testing range	Sign in
Testing Input	Input non-register username
Testing procedure	Input non-registered username and password and test
Expected Test Result	Username/password not found.
Testers	Cherishma Jalaparti, Noya Hafiz
Test result	Passed

Test Type	**Accuracy Test**
Testing range	Calendar - Weight & Reps
Testing Input	Input char
Testing procedure	Go to the calendar page and type in char inputs for input that should be int
Expected Test Result	Will not allow any char/string and won’t display anything that is char/string
Testers	Cherishma Jalaparti
Test result	Passed

Test Type	**Accuracy Test**
Testing range	Add New WorkOut
Testing Input	Add WorkOut feature
Testing procedure	Try to input a paragraph and more
Expected Test Result	Prompt “name size must be between 1 and 32”
Testers	Cherishma Jalaparti, Noya Hafiz
Test result	Passed

Test Type	**Accuracy Test**

Testing range	Add WorkOut
Testing Input	Delete WorkOut
Testing procedure	User cannot delete a workout that they logged previously in Calendar
Expected Test Result	Prompt “Exercise cannot be deleted, it's in use.”
Testers	Cherishma Jalaparti, Noya Hafiz
Test result	Passed

Test Type	**Accuracy Test**
Testing range	All input functionalities
Testing Input	Empty Inputs
Testing procedure	User cannot submit blank texts and the forms will warn the user to fill the field
Expected Test Result	Prompt “fill this field” “Please select a photo to upload.”
Testers	Cherishma Jalaparti
Test result	Passed

Test Type	**Accuracy Test**
Testing range	Profile
Testing Input	Incorrect password
Testing procedure	Input current incorrect password and try to change the password
Expected Test Result	Prompt “Old password is not correct.”
Testers	Cherishma Jalaparti, Noya Hafiz
Test result	Passed

6.3 Test Coverage

For each of the functional requirements, describe if it has been achieved by the system or not.

For each of the non-functional requirements, describe if it has been achieved by the system or not.

Functional Requirements:

1. User Registration and Login
    - Achieved: The system allows users to register with an email and password and log in securely. All test cases for successful and failed logins passed.
2. Add WorkOut
    - Achieved: User can add their WorkOut routine and add the types of workOut they would like to log.
    - Not Achieved: Graph visualization.
3. Log WorkOut tools
    - Achieved: User can update their daily workOuts with their weight and reps on a calendar.
    - Not Achieved: Graph visualization.
4. Photo Album
    - Achieved: User can upload their photos and view photos in an album view.

- Not Achieved: Delete images and ability to make images non-public.

5. Profile - Change password
   - Achieved: Users can securely change their password through the profile settings. The feature includes two-step password confirmation for changes. All test c**ases,** including edge cases like invalid current passwords, passed.

Non-functional Requirements:

1. Login under 4 seconds
   a. Achieved
2. Update Password under 5 seconds
   a. Achieved
3. Server starts under 5 seconds, once the application passes through spring boot.
   a. Achieved: You should paste this link in your browser and the website opens: http://localhost:8080/
4. Performance
   a. Achieved: The system processes typical user actions (e.g., logging workouts, updating profiles) within 2 seconds under normal load conditions.
5. Error Handling
   a. Achieved: Most of the application handles error handling perfectly.
6. Accessibility
   a. Achieved: Navigation bar and buttons navigating through the app.

7 Appendices

AI was used to assist with authentication (services) and restful API code.

7.1 Packaging and installation issues

In our project, we used H2. The database configuration was defined in the application.properties file to integrate with our Spring Boot application. Pom.xml includes all our dependency drivers. Through application.properties, we set the spring.datasource.url property to jdbc:h2:mem:db, enabling the use of an in-memory database that resets with each application restart. The username and password were configured as admin for development. The database driver class, org.h2.Driver, was used to allow Spring Boot to manage the connection automatically.

The schema and data initialization were controlled through properties such as spring.jpa.hibernate.ddl-auto=none and spring.datasource.initialization-mode=always so that any SQL scripts would run at application startup. Additionally, we enabled the H2 console with spring.h2.console.enabled=true, making it accessible at /console for debugging.

Our AppUser entity defined the structure of the corresponding table:

```java
@Entity
@Table(name = "app_user")
public class AppUser {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "id")
    private Integer id;

    @Column(name = "name")
    private String name;

    @Column(name = "password")
    private String password;

    @ManyToOne
    @JoinColumn(name = "role_id", referencedColumnName = "id")
    private Role role;
```

Our data.sql file initializes the database with default values and test data for key tables, ensuring the application is ready to use immediately upon startup. It inserts default exercises such as "Flat Barbell Bench Press" into the default_exercise table. Test workout data, including weight, reps, and timestamps, is added to the history table, while default photos are stored in the photo table for testing upload functionality.

```
INSERT INTO exercise
(name, app_user_id)
SELECT name, app_user_id '2'
FROM default_exercise;

INSERT INTO history
(app_user_id, exercise_id, weight, reps, created_on)
VALUES
( app_user_id 2, exercise_id 1, weight 100, reps 10, created_on CURRENT_DATE()),
( app_user_id 2, exercise_id 1, weight 100, reps 10, created_on CURRENT_DATE()),
( app_user_id 2, exercise_id 1, weight 100, reps 10, created_on CURRENT_DATE()),
( app_user_id 2, exercise_id 2, weight 60, reps 10, created_on CURRENT_DATE()),
( app_user_id 2, exercise_id 2, weight 60, reps 10, created_on CURRENT_DATE()),
( app_user_id 2, exercise_id 2, weight 60, reps 10, created_on CURRENT_DATE());

-- Insert default data into the photo table
INSERT INTO photo (filename, path, upload_time, user_id)
VALUES
    ( filename 'fit.jpeg', path '/uploads/fit.jpeg', upload_time CURRENT_TIMESTAMP, user_id 1),
    ( filename 'fitness.jpg', path '/uploads/fitness.jpg', upload_time CURRENT_TIMESTAMP, user_id 1);
```

```
INSERT INTO default_exercise
(name)
VALUES
( name 'Flat Barbell Bench Press'),
( name 'Incline Barbell Bench Press'),
( name 'Flat Dumbbell Bench Press'),
( name 'Incline Dumbbell Bench Press');

INSERT INTO role
(name)
VALUES
( name 'ADMIN'),
( name 'USER');

INSERT INTO app_user
(name, password, role_id)
VALUES
( name 'admin', password '$2a$12$v/mRrRfQ18gpz5ekVluQ8eEEUiN.7gNJ4MtxQCbmWWCavtagHl6cy', role_id 01),
( name 'user', password '$2a$12$.gvecgyfihg.ozfUhC/pGe48uNmgzsYpluQabfwWWvlO4G5xsXQXy', role_id 02);

INSERT INTO exercise
(name, app_user_id)
SELECT name, app_user_id '2'
FROM default_exercise;
```

7.2 User Manual

1. User Registration and Login

- **Registration**:
    - Navigate to the registration page.
    - Fill in required fields: username, password.
    - Submit the form to create a new account. The system validates inputs and assigns the default "USER" role to the new account.
- **Login**:
    - Navigate to the login page.
    - Enter your username and password.
    - If valid, you'll be authenticated and redirected to the Calendar page.

2. Managing User Account Information

- Go to the Edit Profile on the navigation bar at the top
- Type in Old Password
- Type in New Password
- Type in New Passwor again to Confirm Password
- Click Update Password button, which are updated in the database

3. Adding and Managing Exercises

- Navigate to the Log Workout page from the navigation bar, located at the top
- Select exercises from the predefined list or type in a custom exercises and click Add
- Edit or Delete any preexisting workouts

4. Logging Workout History in Calendar

- Go to the Calendar page from the navigation bar, located at the top
- Select an exercise and enter the weight, reps, and date.
- Submit the data to record your workout. The information is stored in the history table.
- Click on a previous date on the calendar, or use the side arrows to find a day in order to view your workout history from that given day

5. Uploading and Viewing Photos

- Navigate to the Photo Album page.
- Upload a fitness-related photo by selecting a file from your device.
- View uploaded photos in a gallery format.

6. Using the H2 Console

- Access the H2 console at http://localhost:8080/console.
- Log in using the default database credential
    - Change url: jdbc:h2:mem:db
    - user: admin
    - password: admin
- Run SQL queries to view or modify the tables directly for testing.

7.3 Open Issues

- Graph
- Delete Photo
- Log Meal

7.4 Lessons Learned

7.4.1 Project Management & Task Allocations (SLO #2.i)

Make sure your response covers the following aspects (refer to the Project Evaluation Rubrics to see how this part will be evaluated).

- How does your team allocate tasks and responsibilities and where could you improve in the future?
- How does your team do project planning activities?
- How much effort does your team perform risk analysis along the process?
- How did your team update each other's progress and adjust your plans?
- How did your team track changes and respond to changes that occurred?

Our team allocated tasks and responsibilities by dividing the project into smaller components and assigning them based on individual strengths and areas of strength. For most of the new knowledge we learned, we would meet as a team and research resources to assist us with learning new concepts, share amongst each other, and try to implement on our own, such as working with the database. We could improve in the future by setting more hard deadlines for due dates based off of what we wanted to accomplish. We could also improve on setting more subtasks, instead of overloading one single person with an entire section of the project. Overall, setting more deadlines in shorter spans of time would be a big improvement.

We did our project planning activities at the beginning of the project by outlining major expectations we had and creating issues on Github then assigning them to a designated team member. While the initial planning was helpful in identifying key deliverables, we could improve by incorporating more frequent planning sessions to address challenges we did not initially expect and adjust our timeline as the project progresses.

Our team could have definitely spent more time on risk analysis. We discussed possible problems, like technical issues or time limits, during our first meetings, but we didn't really plan for them or figure out ways to handle them if they happened. In the future, we should spend more time thinking about what could go wrong, focus on the most important risks, and come up with backup plans to deal with them better, which would allow us to be more time efficient. Our team spent a lot of time troubleshooting problems that arose as we progressed in our programming.

We updated each other's progress during regular meetings, held weekly. These meetings allowed us to share updates, discuss our struggles, and reallocate tasks as needed. There were times we did not communicate to meet, which occurred earlier in the semester. Establishing a more reliable schedule from the beginning and supplementing meetings with text updates through discord could improve our team dynamic.

To track changes, we would have a team member commit code with comments and message the discord when the commit was happening. Each team member then pulled the current working version from Github and worked on new issues. We held quick team meetings to make sure everyone could run the current version. While this approach worked reasonably well.

7.4.2 Implementation (SLO #2.iv)

Make sure your response covers the following aspects (refer to the Project Evaluation Rubrics to see how this part will be evaluated).

- How did your team perform code review and refactoring to improve code quality? Provide a few code snippets (screenshots) that can best represent the team's coding quality.
    - Code Review: We performed a team collaboration to review our code and test our application. We had issues during frontend and backend connection and making the queries work, we would do a pair programming to structure code. We first structured our java folder structure ensuring folders for backend and a section for frontend. We grouped our files together with repositories under one folder, controllers under one folder, entities under one folder, DTOs under one folder.
    - We also used GitHub's pull feature were used to collaborate with issues. We would add issues in the GitHub and perform team code review to look at those issues.
    - Refactoring:
        - We double checked any duplicated code and used IntelliJ's warnings to fix our code and we had to ensure the code matches our applications.properties file where maven and spring boot properties are implemented. We had to ensure the code is applicable with the maven properties and the restful apis.
- How much is your implementation consistent with your system design?
    - Our original idea of the system was to include many screens that would have links on the main screen that would navigate to these screens. However currently our system is different in that we have made a navigation bar which changes the look of the screen on press.

The AppUserRequestDTO class represents inputs for creating or updating the User Data. It encapsulates the data that the frontend sends to the backend, checking validation and preventing direct exposure of the entity structure.

AppUserRequestDTO

```java
package app.getgainz.dto.appuser;

import ...

@Data  37 usages  caravila
@AllArgsConstructor
@NoArgsConstructor
public class AppUserRequestDTO {

    @NotBlank
    @Size(min = 4, max = 15)
    private String name;

    @NotBlank
    @Size(min = 4, max = 15)
    private String password;

    @NotBlank
    @Size(min = 4, max = 16)
    private String rolename;

}
```

App User Update DTO

```java
package app.getgainz.dto.appuser;

import ...

@Data 34 usages caravila
@AllArgsConstructor
@NoArgsConstructor
public class AppUserUpdateDTO {

    @NotBlank
    @Size(min = 4, max = 15)
    private String name;

    @JsonIgnore
    @Size(min = 4, max = 15)
    private String password;

    @NotBlank
    @Size(min = 4, max = 16)
    private String rolename;

    public AppUserUpdateDTO(String name, String rolename) { 10 usages caravila
        this.name = name;
        this.rolename = rolename;
    }

}
```

The entity class AppUser represents the database entity for the app_user table. Includes structure of the table fields.

```java
package app.getgainz.entity;

import ...

@Getter  caravila
@Setter
@ToString
@Entity
@Table(name = "app_user")
public class AppUser {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "id")
    private Integer id;

    @Column(name = "name")
    private String name;

    @Column(name = "password")
    private String password;

    @ManyToOne
    @JoinColumn(name = "role_id", referencedColumnName = "id")
    private Role role;



}

```

and then we added mapping that connects both the entity and DTO classes

```java
@Service  caravila
public class AppUserMapperService implements MapperService<AppUserRequestDTO, AppUserResponseDTO,
        AppUserUpdateDTO, AppUser> {

    private final RoleRepository roleRepository;  3 usages
    private final BCryptPasswordEncoder bCryptPasswordEncoder;  3 usages

    @Autowired  caravila
    public AppUserMapperService(RoleRepository roleRepository, BCryptPasswordEncoder bCryptPasswordEncoder
        this.roleRepository = roleRepository;
        this.bCryptPasswordEncoder = bCryptPasswordEncoder;
    }

    @Override  caravila
    public AppUser requestDtoToEntity(AppUserRequestDTO requestDTO) {
        if (requestDTO == null) {
            return null;
        }
        Optional<Role> optionalRole = roleRepository.findByName(requestDTO.getRolename());
        if (optionalRole.isEmpty()) {
            throw new GetGainzException("Role doesn't exists with name: " +
                    requestDTO.getRolename());
        }
        AppUser appUser = new AppUser();
        appUser.setName(requestDTO.getName());
        appUser.setPassword(bCryptPasswordEncoder.encode(requestDTO.getPassword()));
        appUser.setRole(optionalRole.get());
        return appUser;
    }

    @Override  caravila
    public AppUserResponseDTO entityToResponseDto(AppUser entity) {
```

We also thought it is easier to add services that check validation.

7.4.3 Design Patterns

What are the design patterns used and why?

The GetGainz system follows the Model-View-Controller (MVC) design pattern. In this pattern, the Model represents the data layer, including entities like AppUser and Role, which define the database structure and relationships. The Controller layer handles HTTP requests and serves as the connection between the user interface and logic, with classes such as LoginController and AppUserController managing user actions. The Service layer works between the controller and the model. For example, the LoginService processes user login details and interacts with the AppUserRepository to validate credentials.

7.4.4 Team Communications

How team communications were conducted and where could you improve in the future?

Our team communication was mainly through discord which is where we would settle on dates and times to meet over the weekends and outside of class. We would also use lab time to plan out the week and when we wanted to meet as well as how long based on everyone's availability with

consideration of everyone's workload for that week as well. We could potentially improve on meeting starting earlier in the time span we had to complete our project. We initially began to only meet during lab periods but realized our project was not progressing. By starting earlier, we could have had more time to implement more features and plan incorporating our requirements more efficiently.

7.4.4 Technologies Practiced (SLO #7)

During this project, we learned a lot of new skills that we would not have learned without this project including Docker, MySQL, and Java controllers. Working with Docker introduced us to containerization. We all collectively had to figure out and eventually learned to create Dockerfile configurations to package our Spring Boot application into a containerized environment. This experience taught us benefits of using Docker Compose through terminal to manage our project, such as linking the application with a MySQL database (although we eventually scrapped having to use Docker, it was a useful skill to learn). We also learned how to implement MySQL into a real time application. We had to learning how to design tables, write SQL queries, and configure connections between MySQL and our Java application which caused a lot of difficulty but was definitely a learning experience itself as we had to troubleshoot. Additionally, throughout the semester as we learning about controllers through lectures, we learned to implement controllers in Java as part of the Spring Boot framework. We also learned about constructing a Restful API in the development of our project which is what we initially started without before developing more in depth features.

7.4.5 Desirable Changes

Nicholas

If we had another month to improve our project, I would expand on different ways for the users to be able to track and visually see a representation of their progress. For instance, I would like to give the user the ability to view their data from a graph as a visual representation other than purely numbers and media images. For this to happen we would need to find a way to link the data to a graphing library and configure it to our project. I would also expand on the user settings, meaning I would allow users to be able to change more in their settings. For example a light and dark mode, notifications for when a workout or meal is scheduled, and the ability for a user to delete their account.

Noya

If we had an additional month, I would like to add our calorie intake versus weight graph as well as additional helpful graphs, such as weight lifted versus date. Additionally, I would want to incorporate a more diverse profile for the user to input their characteristics and be able to use that data. For example, create profile would have a height and weight field in which the user would be able to add and adjust their body weight. Another feature that could be added is a comments section for the photo album which would be convenient for users to note down anything they wish across their fitness journey. We did not implement a delete photo function which would make our app more user friendly in the case a user accidentally uploads the wrong photo.

Cherishma

If we had an additional month, we would have an implementation of log meal and graph for each functionality. Our application lacks usage of graph which is one of our key functionality and with a graph we would have an actual fitness application that enables tracking for fitness enthusiast and the photo album would be designated for a user instead of it being viewed universal. The photo album can be viewed by other signed in users from previous users that have uploaded it, it would be nice to have a photo album designated for one user and they have the option to share their photo album. They can also view the progress of the photo album in a graph view. Instead of just logging and tracking our log, we would have a lot of features that alerts users to log their workouts on a daily basis.

Carlos

If we were to have an additional month some features we would be able to implement would include meal log, and the strength graph. These were features that we initially believed we were going to be able to accomplish adding but have since ran out of time to completely implement them. I believe that with our application, while this probably cannot be implemented in one month would like to see some webcam or similar feeding images for the photo album. I would say to change the application to be supported on a mobile device but that would be another project itself. I believe that one month could allow us to implement other features that most fitness apps already have such as some type of social aspect to it where users could interact by comparing results, this feature could be as basic as implementing a way for users to share their progress to their social media accounts.

7.4.6 Challenges Faced

Nicholas Brodsky

The most difficult part of the assignment for me was the system requirements. This is for a few reasons. The first being that we had to make our project in java, which unfortunately none of my group members were familiar with. It took us a lot of researching and trial and error to be able to figure out which specifications our project needed, and also which were best fitted for our project. We faced challenges with either picking frameworks that were too complicated or overkill for what we had in mind, and also IDE's which didn't support the configurations we needed. The second reason which made it difficult to choose requirement specifications for our team was that our lack of experience in working with java made it difficult to predict what technology we may need. For example, we knew that we needed a graph and a database for our project, but we weren't sure what graphing library or which database provider would be the best to use. This ultimately led us to not having time to figure out how to implement the graph into our project.

Noya

Out of requirements specification, system design, and system implementation, I believe requirements specification was the hardest task. During this phase, we set ambitious and unrealistic tasks without fully understanding how we would implement them at the beginning of the semester, especially given that we all had very limited skill sets and had to learn a language alongside working with implementation. Additionally, because we were focused on

brainstorming ideas, we did not consider the complexity of converting those requirements into executable code, which caused significant difficulties later in the project, such as with the graph implementation which was initially a big part of our project, however turned out to be an additional issue we did not have enough time to close.

Cherishma

The hardest task for us was implementing our backend, we took a good decision by first implementing our backend and then work on our frontend. During this phase, many of us are using linux and the others had windows so it was harder for us to directly connect our backend and we realized we had to download additional packages from mySQL and not all of us were able to do that. We also spent few weeks learning docker and using docker to connect sql through it but we were facing huge issues with the administration permissions and there were a lot of steps that had us test the permissions and users would not have proper permissions through the docker and mySQL. We figure h2 uses localhost and its simpler implementing it and it also offers a browser console for us to test our queries and connection.

Carlos

I believe that getting all the requirements initially was one of the hardest portions of all of this. Having so many options and infinite possibilities made us forgot to consider our time constraints. A portion of this error was also found in not knowing exactly how long implementing our design would take. I believe that should we have to do this project again we would have an easier time implementing and realizing what tools to use from the beginning. I also think that having such an open ended question of what to create as a project made us choose certain design aspects that were unnecessary and caused greater pain to try to implement. I believe that we could have achieved a similar product without utilizing the client server aspect which sent us on quite a journey trying to figure out the setup to a system such as this.