Final Iteration

By team.java (Mac Johnson, Dakota Hernandez, Faith Ota, Joshua Carroll, Eli Hall, Sofia Amador)

team.java - PawPlates

team.java - PawPlates

Our Vision

Version <2.0>

Revision History

Date	Version	Description	Author
06/02/2025	1.0	Initial document	Mac Johnson, et al
19/02/2025	1.1	Formatting revisions	Faith Ota
06/05/2025	2.0	Final Edits	Faith Ota

Table of Contents

Vision (Small Project)

1. Introduction

The purpose of this document is to collect, analyze, and define high-level needs and features of the fitness app PawPlates. It focuses on the capabilities needed by the stakeholders and the target users, and **why** these needs exist. The details of how PawPlates fulfills these needs are detailed in the use-case and supplementary specifications.

2. Positioning

- **Problem Statement**

The problem of	People forgetting details about their health/workouts and having no good place to find workout plans.
affects	Everyday people, health enthusiasts, professional trainers, athletes
the impact of which is	These things directly affect public health, and having an organized plan for healthy living is important for society
a successful solution would be	A place where users can organize their health, encourage others to live healthily, and take steps to live better themselves.

- **Product Position Statement**

For	Everyday people, athletes, personal trainers, health enthusiasts
Who	Need a place to organize their health and can create healthier habits
The (product name)	Is a health/fitness app
That	Helps users live a healthy life by logging and keeping track of their lifestyle
Unlike	Fitbit
Our product	Allows users to manually log details, and focuses more on community

3. Stakeholder and User Descriptions

- **Stakeholder Summary**

Name	**Description**	**Responsibilities**
App owners	Individuals responsible for ownership of the company/application. They are also the software engineers responsible for the app's development.	Ensures the product will be released in a timely manner; in charge of software development and project management. Approves resources, approves features, approves app functionalities.

Professional trainers	The trainers host sessions and post workouts on the platform.	Ensures that users have content to continuously drive them to the app. Ensures a consistent user count.

- **User Summary**

Name	Description	Responsibilities	Stakeholder
General public	Anyone who takes their health seriously; anyone who wants a place to track workout progress	Uploads and tracks details about workouts; provides a community/audience for professional trainers on the platform	Professional trainers

- **User Environment**

  The app will originally be developed for a desktop environment, where the user will use their mouse and valid input device to navigate through. As a precondition to accessing all of the app's content and features, all users will be prompted to log in once the app is launched, or to create an account if no account is found.

  A user logs on to see a dashboard, where they can enter several different interfaces. One page is where they can log food, water, and calories. Another page is where they can log workouts, workout times, record lifted weights and set workout reminders. One interface is a large page where they can view workout plans posted by other trainers on the platform.

  A trainer can log in and see the same things that a typical user can see but also have options to post a workout plan. Choosing the option to post a workout plan navigates the user to an environment where they can compose a long message, where they can apply filters pertaining to certain workout styles. These messages can be step by step textual instruction, or a link to a video showcasing it visually.

- **Summary of Key Stakeholder or User Needs**

Need	Priority	Concerns	Current Solution	Proposed Solutions
Ability to track food/calories	10/10	General user	Feature is planned	Feature needs to be built into the app
Ability to create workout plans	10/10	Trainer	Feature is planned	Feature needs to be built into the app, in order to bring an audience to the platform
Workout tracking – weights, reminders, stats	10/10	General user/trainer	Feature is planned	Core feature needed in final app

- **Alternatives and Competition**

  Many users manually track these things on their own, such as using a note app on their devices, or by writing it down physically in a handbook. Establishing a routine can be harder using this method, as well as sharing details with others. The main appeal of PawPlate is to bring a more cohesive design to tracking fitness, making it more appealing to more audiences and allowing users to share these things with each other.

  Many users use Fitbit for this reason. The appeal of PawPlate is that no device is required to use the platform, making it more accessible to all audiences.

4. Product Overview

- **Product Perspective**

This fitness app is independent, and self-contained. This system is designed to help users monitor and track their health and fitness. The app will allow users to log health data, track workouts, sets, and goals. The app will allow users to analyze their progress over time. Trainer-led exercises will also be available, giving users the ability to carry out various workout plans. This system enables users to track progress, interact with trainers, and other users. Major components of this system would include a log in screen, user dashboard, workout tab, sleep tab, workout tab with a trainer led option, and a social tab. These various sections in the system will allow the user to be able to navigate easily and find the features that they want. This app is designed to be user friendly and adaptable to the individual's fitness goals.

- **Assumptions and Dependencies**

The application is designed for web browsers, some changes may be adjusted to allow more platforms during development. To track steps the user will have to manually log their steps. Most features will work offline with the exception of social features. The app relies on the user to log their workouts, progress, and setting goals. If this engagement is low, a generated goal can be created. This system will comply with all regulations regarding the user’s data privacy and will adjust with the current regulations if there are any changes. Any major changes to these assumptions will need updates to the vision document and the app.

5. Product Features

- **User Authentication and Access Control**

Description: The system will provide secure authentication mechanisms, including user registration, login, and role-based access control to ensure data security. **Priority:** High **Attributes:** Stable, High Benefit, Medium Effort, Low Risk

- **Dashboard and User Interface**

Description: A user-friendly dashboard displaying key metrics,and system status updates will be available for efficient navigation. **Priority:** High **Attributes:** High Benefit, Medium Effort, Medium Risk

- **Data Management and Storage**

Description: The system will allow users to input, update, retrieve, and delete relevant data while ensuring data integrity. **Priority:** High **Attributes:** Stable, High Benefit, High Effort, Medium Risk

- **Reporting and Analytics**

Description: Users will be able to generate and export various reports, including trend analysis and performance metrics, for informed decision-making. **Priority:** Medium **Attributes:** High Benefit, Medium Effort, Medium Risk

- **Notifications and Alerts**

Description: The system will provide real-time notifications via in-app alerts for critical events and system updates. **Priority:** Medium **Attributes:** High Benefit, Medium Effort, Medium Risk

- **Scalability and Performance Optimization**

Description: The system will be designed to handle increased user loads and data growth without performance loss. **Priority:** High **Attributes:** Stable, High Benefit, High Effort, Medium Risk

- **Customization and Personalization**

Description: Users will have the ability to configure system settings, dashboards, and preferences to suit their individual needs. **Priority:** Low **Attributes:** Medium Benefit, Medium Effort, Low Risk

These features will serve as the basis for further detailing in the use-case model and contribute to project scope management.

6. Other Product Requirements

The app must be compatible with iOS (version 12 and above) and Android (version 8.0 and above). It should run smoothly on devices with at least 2 GB of RAM and 16 GB of free storage.
The system should integrate with popular fitness trackers (e.g., Fitbit, Apple Watch, etc.). It should optimize battery usage and not exceed 20% battery drain during continuous use for 1 hour.

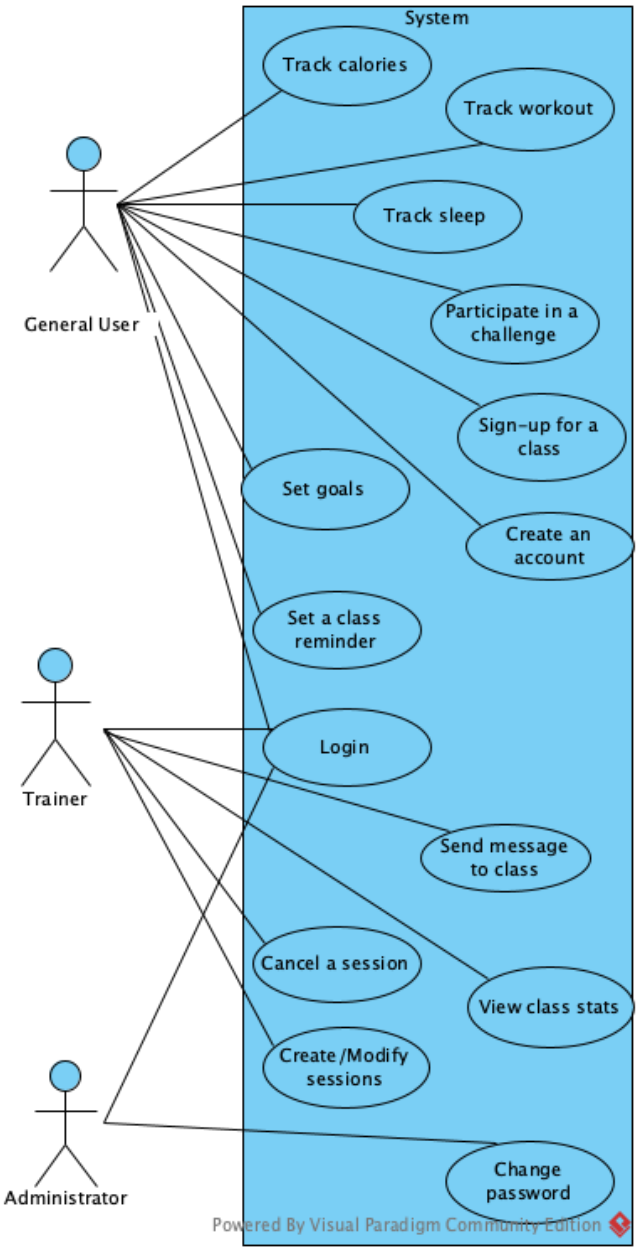
The app must support offline functionality for basic features (e.g., calorie tracking, workout history), with data syncing when the internet connection is restored. The system should function on Wi-Fi and mobile networks, with limited features available during low-bandwidth conditions. Any failure, such as during data sync, should display a user-friendly error message and allow retry without data loss.

The app will require cloud storage services, which must be secure and scalable. The application must meet the Apple App Store and Google Play Store guidelines for health and fitness apps.

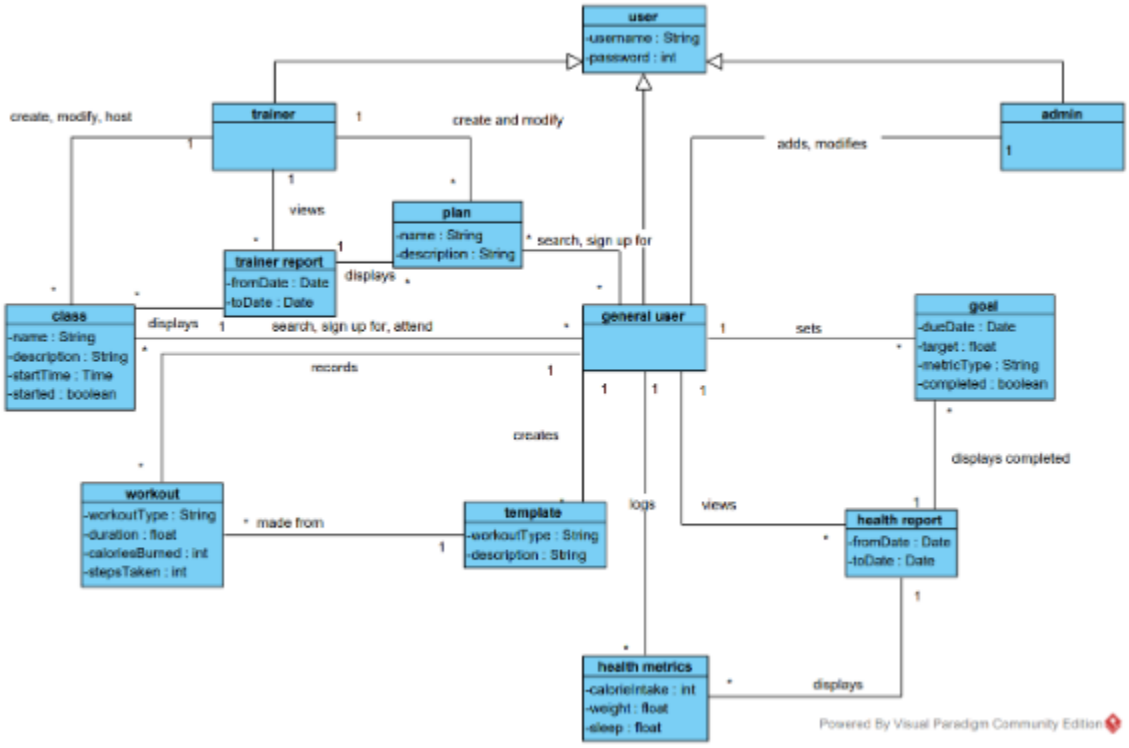
The system should provide a digital user guide accessible from within the app that covers setup, features, and troubleshooting, as well as in-app FAQs and support contact information. The application should include terms of service agreement and privacy policy displayed clearly upon first login.

High Priority: Data security and privacy compliance, performance, and offline functionality.
Medium Priority: Integration with third-party devices, documentation, and cloud storage scalability.
Low Priority: Third-party app dependencies.

7. Use-Case Diagram



8. Domain Model



9. Use Cases

UC1 - Trainer Creates an Exercise Plan-Dakota Hernandez

Scope: Health & Fitness Application

Level: User Goal

Stakeholders and Interests:

- Trainer – Wants to create and manage exercise plans effectively.
- General User – Wants access to trainer made exercise plans.
- System Administrator – Maintains the system's functionality and resolves issues.

Preconditions:

- Trainer is authenticated and logged into the system.

Postconditions:

- The exercise plan is successfully saved in the system.
- The plan is available for users to view and enroll in.

Main Success Scenario:

1. Trainer logs into the system.
2. Trainer navigates to the exercise plan creation page.
3. Trainer selects the type of plan (self-paced or scheduled class).
4. Trainer enters plan details:
    a. Name of the plan
    b. Description
    c. Required equipment
    d. Recommended fitness level
    e. Session length
    f. Date, time, days of the week, and duration (for scheduled classes)
    g. Prerequisites (if any)
5. Trainer submits the plan for creation.
6. System validates the entered details.
7. System saves the plan and makes it available for users.
8. Trainer receives confirmation of successful plan creation.

Alternate Paths:

2.a If the trainer is not authenticated:

- System prompts the trainer to log in.

4.a If required details are missing:

- System highlights missing fields and prompts trainer to complete them.

6.a If system validation fails:

- System displays an error message and the necessary corrections.

7.a If the system encounters an error while saving the plan:

- System notifies the trainer of the issue.
- Trainer can retry saving the plan.

8.a If the trainer wishes to modify the plan after creation:

- Trainer navigates to the plan management section.
- Trainer updates details and submits changes.
- System validates and saves the modifications.

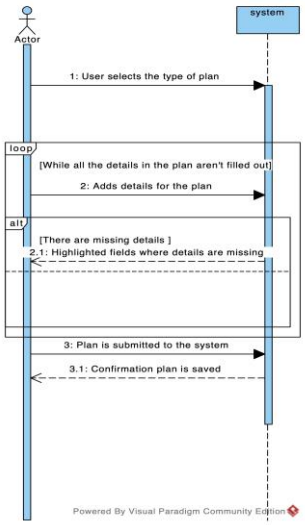
Exceptions:

- If the system crashes during plan creation, trainer must reattempt after system recovery.
- If the network connection is lost, trainer must retry when connectivity is restored.

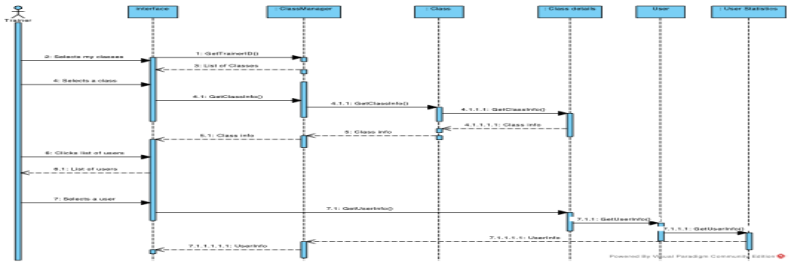
Business Rules:

- Only authenticated trainers can create exercise plans.
- Self-paced plans must contain at least session length, and required equipment details.
- Scheduled classes must include date, time, and prerequisites.
- Trainers can update or delete their plans but not those created by other trainers

SSD:



SD:



g

UC2 – General User wants to track daily statistics - Faith Ota

Scope: Health & Fitness Application

Level: User Goal

Stakeholders and Interests

- General User – person who is interested in logging down their health habits on this app
- System Administration – person who oversees the server and has access to all general users and trainers

Precondition: User has a valid login username and password.

Postcondition: Workout duration and activities are stored, and user can see progress towards goal.

Main Success Scenario:

1) User goes to “Log Daily Workout” page
2) User inputs:
    a) Date
    b) Workout Focus
    c) Duration
3) User presses “OK”
4) System will save the information and use it to calculate current progress
5) System will update the past information and graph
6) User can see their new progress

Alternate Paths:

2.a) User wants to store sleep schedule instead of a workout

1) There will be an option to input sleeping hours and will have separate progress

2.b) User wants to track caloric intake

1) There will be an option to track meals

2) User will have to estimate number of calories and put the amount in the system

3.a) Not all workout fields are filled out

1) All fields will have an asterisk, "*" to signify that it must be filled out to be a valid input

2) Workout will not be saved

3) User will be prompted to fill out required fields

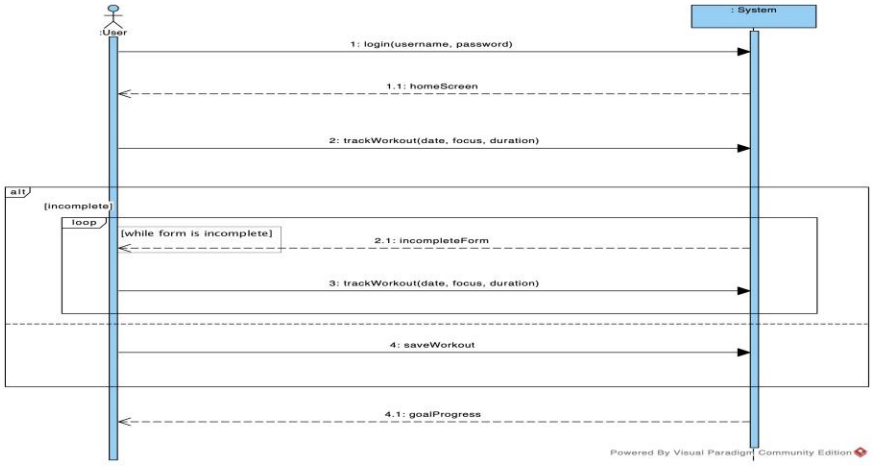
5.a) The progress is negative

1) The user's goals will be pushed back accordingly

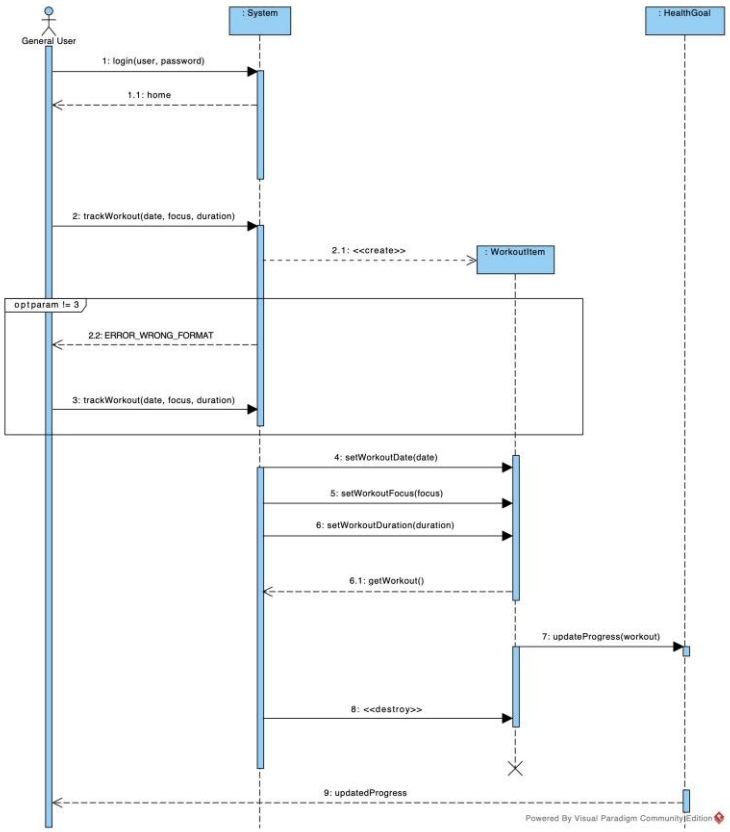
6.a) The User wants to see their progress over different intervals of time (days, weeks, months)

1) The User can select a zoom to change the axes of the progress graph

SSD:


: System
User
1: login(username, password)
1.1: homeScreen
2: trackWorkout(date, focus, duration)
alt
[incomplete]
loop
[while form is incomplete]
2.1: incompleteForm
3: trackWorkout(date, focus, duration)
4: saveWorkout
4.1: goalProgress
Powered By Visual Paradigm Community Edition

SD:



UC3 – User Sets Reminder(s) for a Workout – Mac Johnson

Scope: Health and Fitness Application
Level: User Goal
Stakeholders and Interests:

- General User – person who wants to set reminders for a workout

Precondition: User has a registered account with an email and password
Postcondition: User receives a push notification at a certain time to their device to remind them of a workout or healthy habit
Main Success Scenario:

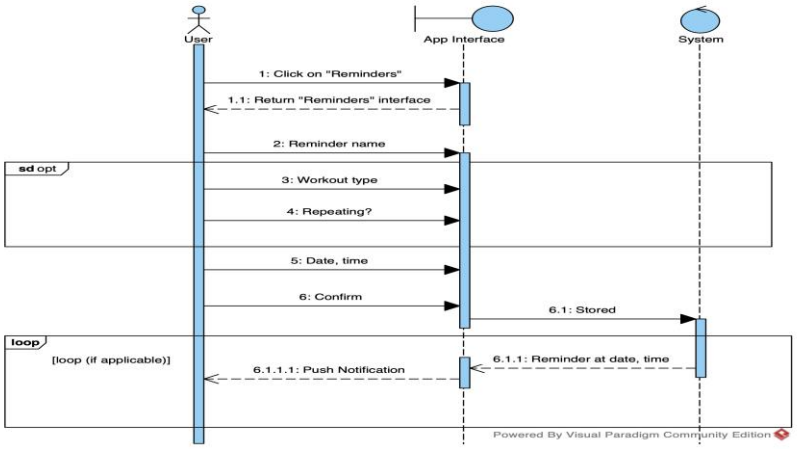
- User enters the app and lands at the homepage
- User goes to “Reminders” section
- User inputs:
  - Reminder/Workout name
  - Workout type
  - Date, time
  - Checkbox for repeating reminders
- User clicks “Confirm”
- Data is sent to system and stored
- User receives a notification at the time and day they selected, which will repeat if

applicable

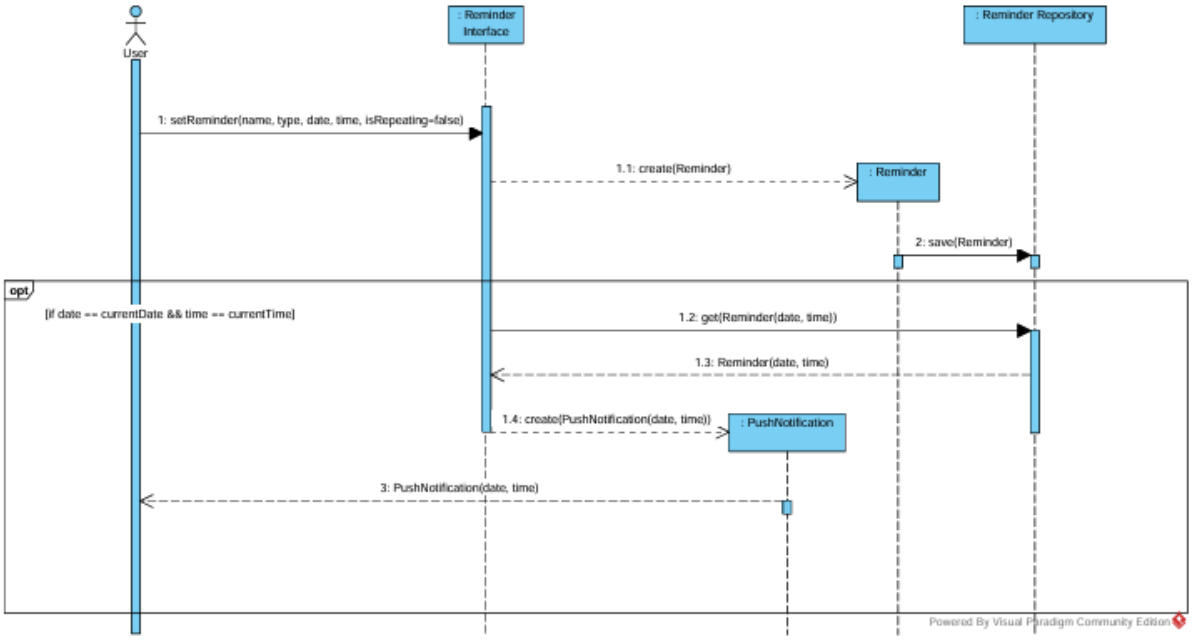
Alternate Paths:

- User does not want repeating reminders
  - The user will leave the repeating reminders checkbox unchecked
  - The user will not receive a notification for another time than what they specifically selected
- User leaves name or another field blank
  - An asterisk (*) will be left beside the field name to indicate it is required, and will not allow the user to continue

SSD:



SD:



UC4 – User Tracks Caloric Intake – Faith Ota

Scope: Health & Fitness Application
Level: User Goal
Stakeholders & Interests:

- User – person who uses the app to keep track of their meals

Precondition: General User has a valid username and password
Postcondition: The calories are stored, and the user's health goals are reevaluated
Main Success Scenario:

1. User clicks on "Record a Meal" page from the "Home" screen
2. User inputs:
    a. Meal name (ie Breakfast, Lunch, Dinner, Snack)

   b. Food items
   c. Estimated calorie count for each item
3. User presses the “Save” button
4. Meal is set and stored in the System
5. User can see their progress towards their previously set goal

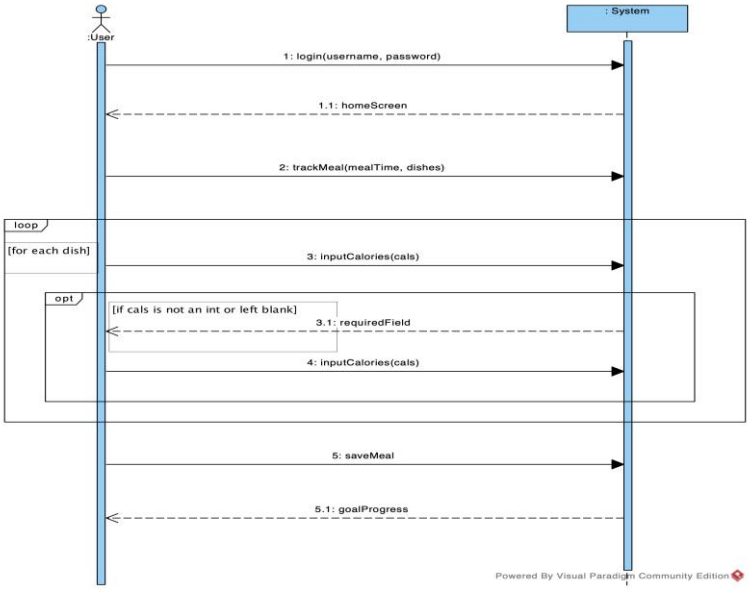
Alternate Paths:

3.a) User does not fill out calories section for an item

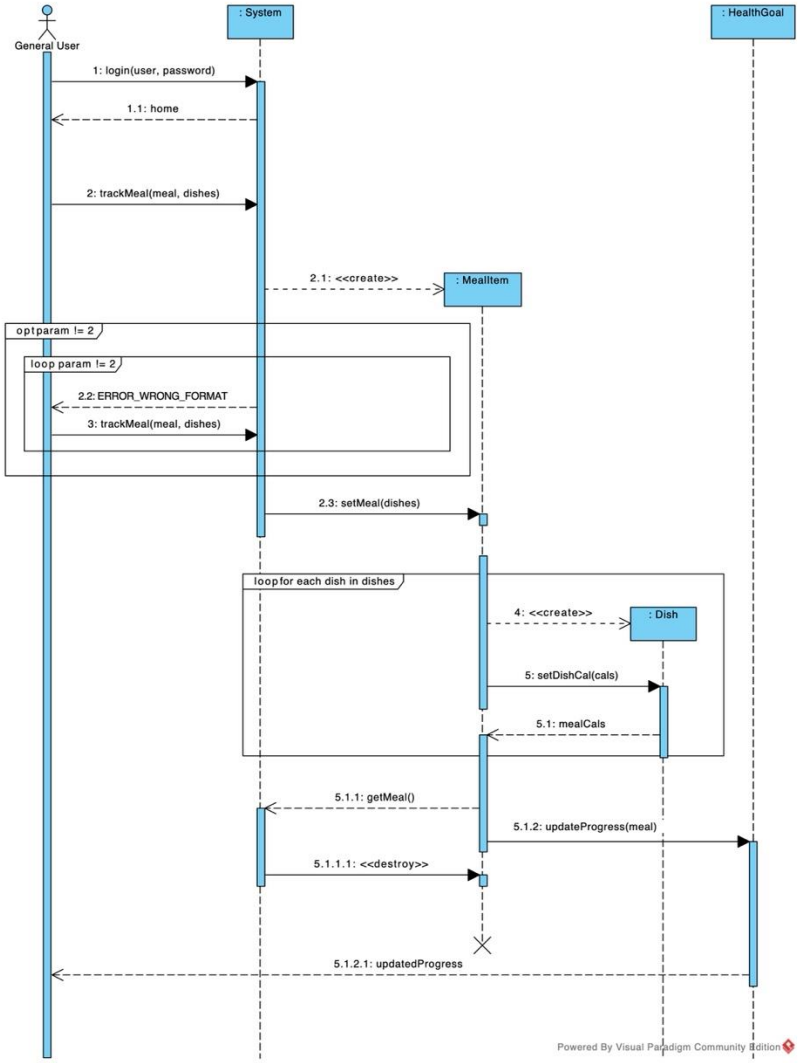
1) All fields will have an asterisk, “*” to signify that it is a required field

2) The missed field(s) will be highlighted until filled with proper information (ie Calorie” field will only accept positive integer values)

SSD:



SD:



UC5 – Administrator Resets Password – Sofia Amador

Scope: Health & Fitness Application
Level: User Goal
Stakeholders & Interests:

- System Administrator – person who has access to general user's account information that is not privately regulated, and has the ability to reset passwords.
- General User – person who wants to reset their account password.

Preconditions:

- Administrator is logged in with their credentials and has administrative permissions.
- General user has a valid pre-existing username and password, and knows their username.

Postconditions:

- General user's password is reset if the security question condition is met.

- If the password reset is unsuccessful, the account is flagged for suspicious activity.
- If the password reset is successful, the date of this event is logged.

Main Success Scenario:

1. User submits a request to reset their password.
2. An administrator verifies that there is a registered account with the given username.
3. The administrator can access relevant information associated with the account, such as name, the security questions and answers, if the password has been recently reset or has flagged activity.
4. The administrator checks that the account is not flagged and the password has not been reset at least within the past week.
5. The administrator asks the general user their security question to verify their identity.
6. The general user answers the security question correctly based on their answer when they first created their account.
7. The general user provides their new password.
8. The administrator resets the general user's password.

Alternate Paths:

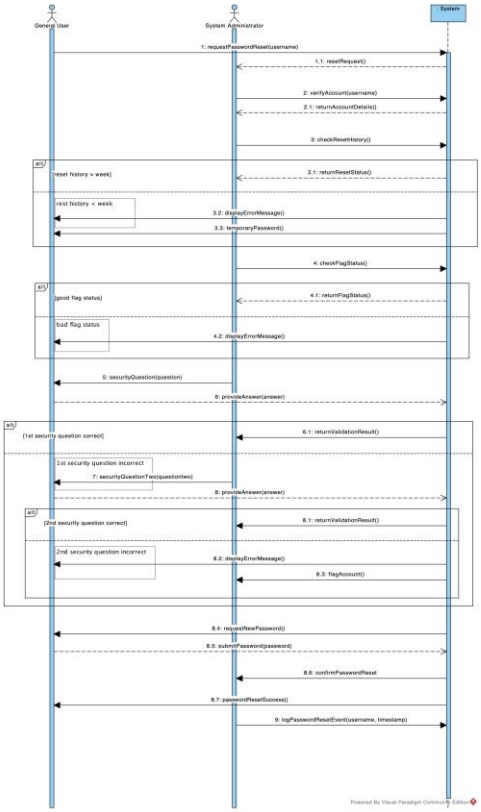
4a) General user's password has been reset within the past week.

- The user is asked their security question and answers it correctly.
- The user is given a temporary one time use password, but has to wait until a week has passed since the last time they changed their password to reset it again.

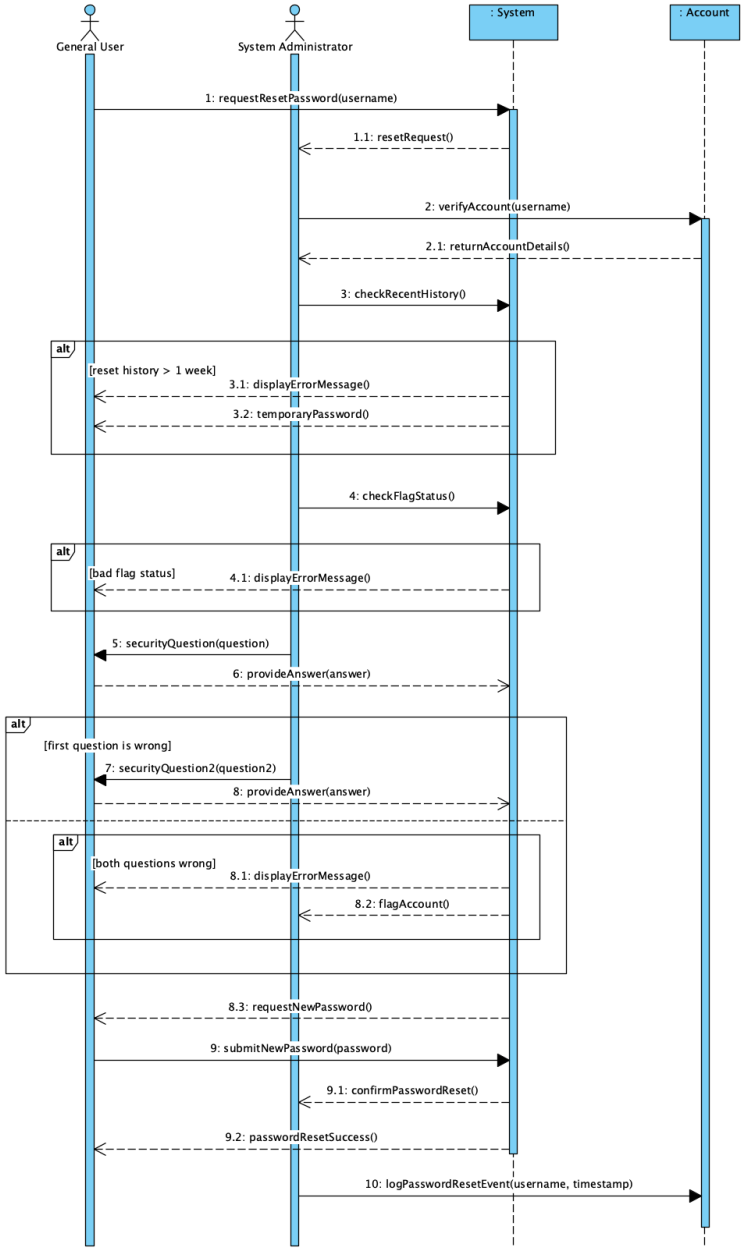
6a) General user incorrectly answers their security question.

- The user is asked a second security question, but also answers incorrectly.
- The user is unable to reset the password and may create a new account.
- The administrator flags this event as suspicious activity.

SSD:


General User
System Administrator
System
1: requestPasswordReset(username)
1.1: resetRequest()
2: verifyAccount(username)
2.1: returnAccountDetails()
3: checkResetHistory()
alt
[reset history > week]
3.1: returnResetStatus()
[reset history < week
3.2: displayErrorMessage()
3.3: temporaryPassword()
4: checkFlagStatus()
alt
[good flag status]
4.1: returnFlagStatus()
bad flag status
4.2: displayErrorMessage()
5: securityQuestion(question)
6: provideAnswer(answer)
alt
[1st security question correct]
6.1: returnValidationResult()
1st security question incorrect
7: securityQuestionTwo(questionTwo)
8: provideAnswer(answer)
alt
[2nd security question correct]
8.1: returnValidationResult()
2nd security question incorrect
8.2: displayErrorMessage()
8.3: flagAccount()
8.4: requestNewPassword()
8.5: submitPassword(password)
8.6: confirmPasswordReset
8.7: passwordResetSuccess()
9: logPasswordResetEvent(username, timestamp)

SD:



UC6 – User Tracks Sleep Hours – Eli Hall

Scope: Health & Fitness Application

Level: User Goal

Stakeholders & Interests:

- General User – person who wants to record their daily number of hours slept.

Preconditions:

- General user is logged into the system with valid username and password.
- The system initializes the sleep hours to 0 each day.

Postcondition:

- The daily tracked number of hours slept is changed.

- The system generates graphs and statistics based on long term data.

Main Success Scenario:

1. General user navigates to sleep tracking tab.
2. The system prompts the general user to input the number of hours they slept.
3. The user is given the option to edit the total number of hours recorded or add additional hours to the total for that given day.
4. The user inputs the positive number of hours ($0 < n < 24$). The system records this change.
5. The system updates the long term sleep data and generates the related graphs/stats which can be reviewed by the user.

Alternate Paths:

4a) The user inputs a negative number for number of hours slept.

- The user is prompted again to input a permissible number, otherwise their number is denied.

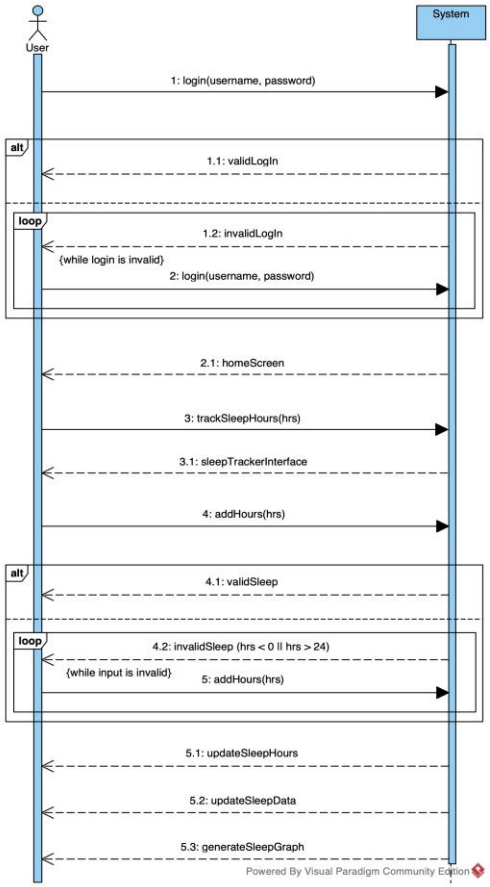
4b) The user inputs an additional number less than 24 that brings the total number of hours slept to higher than 24.

- The user is prompted again to input a permissible number, otherwise their number is denied.

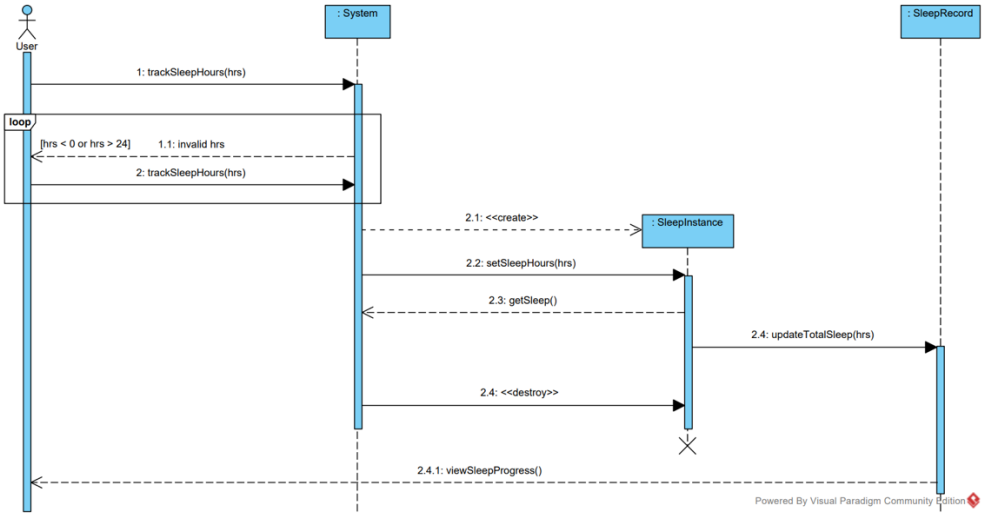
4c) The user inputs 18 hours but they meant to input 8 hours.

- The user can go back to step 3 to make edits to the total number of hours to correct this mistake.

SSD:



SD:



UC7 – Registering for Scheduled Class – Joshua Carroll

Scope: Health & Fitness Application

Level: User Goal

Stakeholders & Interests:

- General User – person who wants to register for a class.
- Trainer – person who creates and manages the classes.

Preconditions:

- General user is logged into the system with a valid username and password.
- There are classes that are created by trainers

Postcondition:

- The general user is added to the class roster
- The trainer's schedule is updated
- The general user's schedule is updated
- Sends confirmation message to trainer and general user

Main Success Scenario:

1. General user navigates to the classes tab
2. General user selects register for a class
3. General user selects an event based on the time and day
4. The system provides a recommendation based on the general user's fitness level and prerequisites whether they should take the class.
5. The user clicks register for class
6. The system verifies logistics, including the availability, class size, and equipment availability.
7. General user is registered for the class and is updated on the user's and trainer's schedules

Alternate Paths:

4a) The class size of participants is full

- The system notifies the user that the class is full and gives the user an option to join a waitlist
- The general user joins the waitlist
- The system updates the waitlist

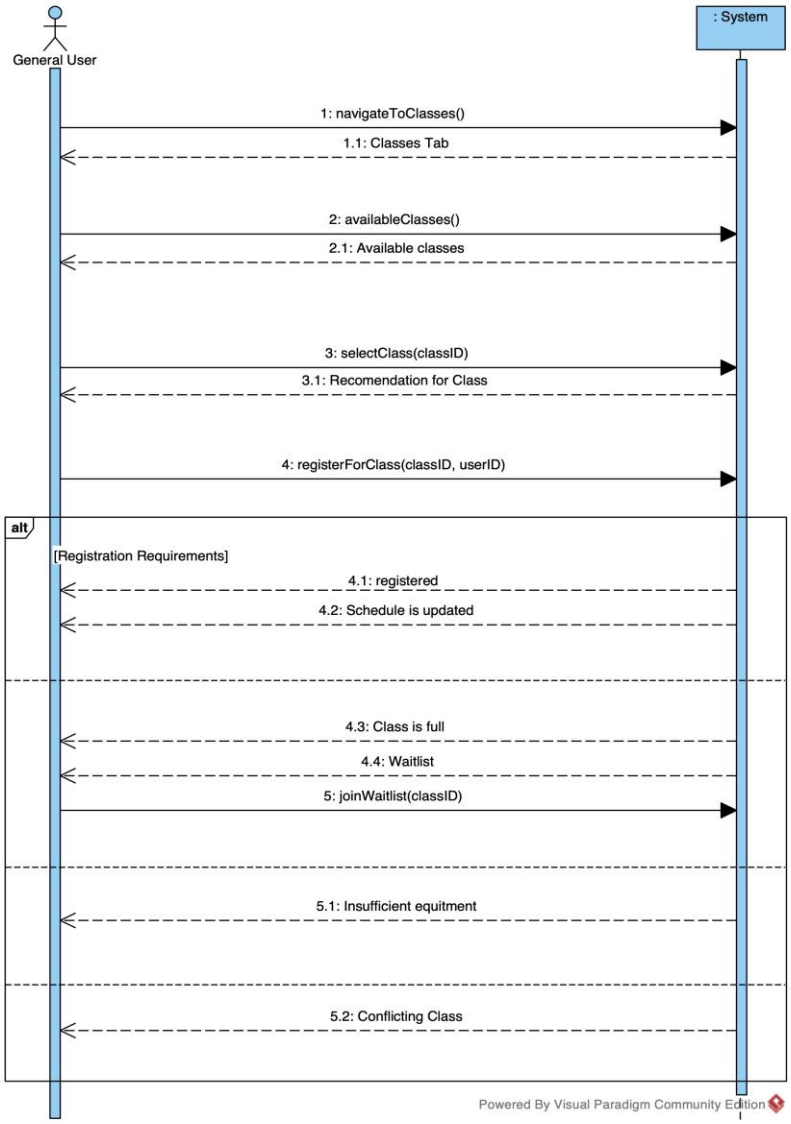
4b) The general user does not have the equipment for the class

- The system prevents the general user from registering
- The system shows other classes that the general user can register for

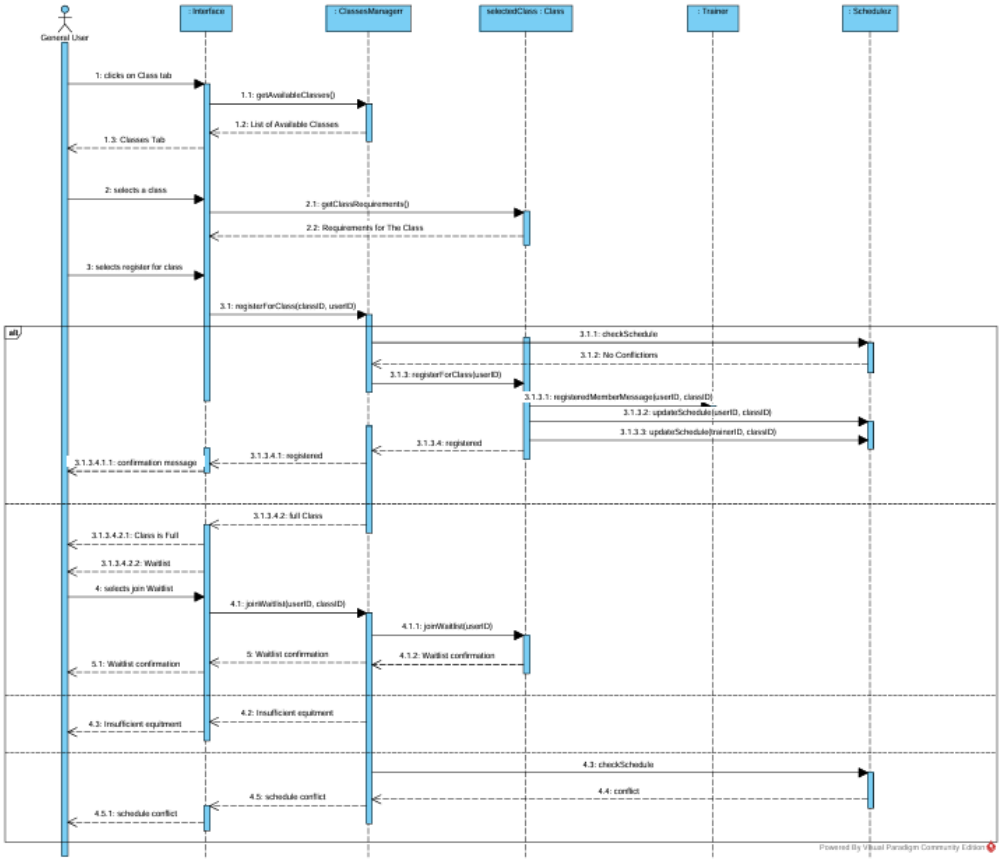
4c) The general user has a conflicting class when registering

- The system prevents the general user from registering
- The system notifies the general user that they must drop the previous class in order to register for this class

SSD:


General User
: System
1: navigateToClasses()
1.1: Classes Tab
2: availableClasses()
2.1: Available classes
3: selectClass(classID)
3.1: Recomendation for Class
4: registerForClass(classID, userID)
alt
[Registration Requirements]
4.1: registered
4.2: Schedule is updated
4.3: Class is full
4.4: Waitlist
5: joinWaitlist(classID)
5.1: Insufficient equitment
5.2: Conflicting Class
Powered By Visual Paradigm Community Edition

SD:



UC8 – Setting Up a Workout Goal – Faith Ota

Scope: Health & Fitness Application

Level: User Goal

Stakeholders & Interests:

- General User – a person who wants to use the app to achieve their health goals

Preconditions:

- General User has a valid login and password

Postconditions:

- Progress towards goal is updated and saved to the system

Main Success Scenario:

1. User navigates selects the “Set a Goal” page from the main home interface
2. The system prompts the user to pick a category in which their goal falls into (ie weight loss, muscle gain, increased cardio, etc)
3. The system prompts the user to specify their goals by inputting the “goal” (ie lifting x-pounds, running an x-minute mile, etc) and date to achieve goal by.
4. After filling out the required fields, the user presses “Save” to prompt the system to store the goal data.

5. The system will return a short summary of a daily workout and diet plan to reach said goal.

Alternate Paths:

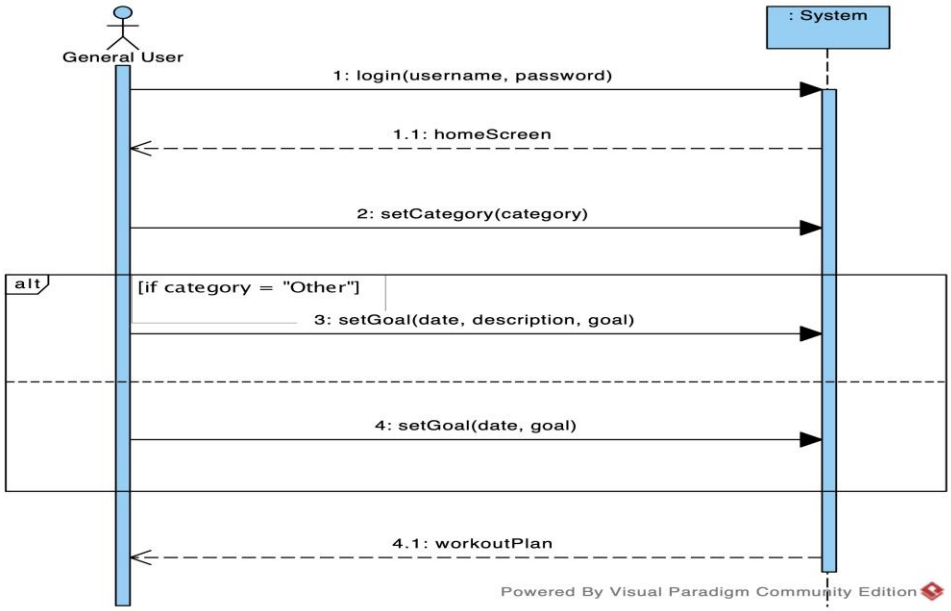
2.a) The user's goal does not fit into any of the given categories.

- The user can select "other" and provide a written description of their goal
- All of the fields become optional to be able to maximize the types of goals that can be set

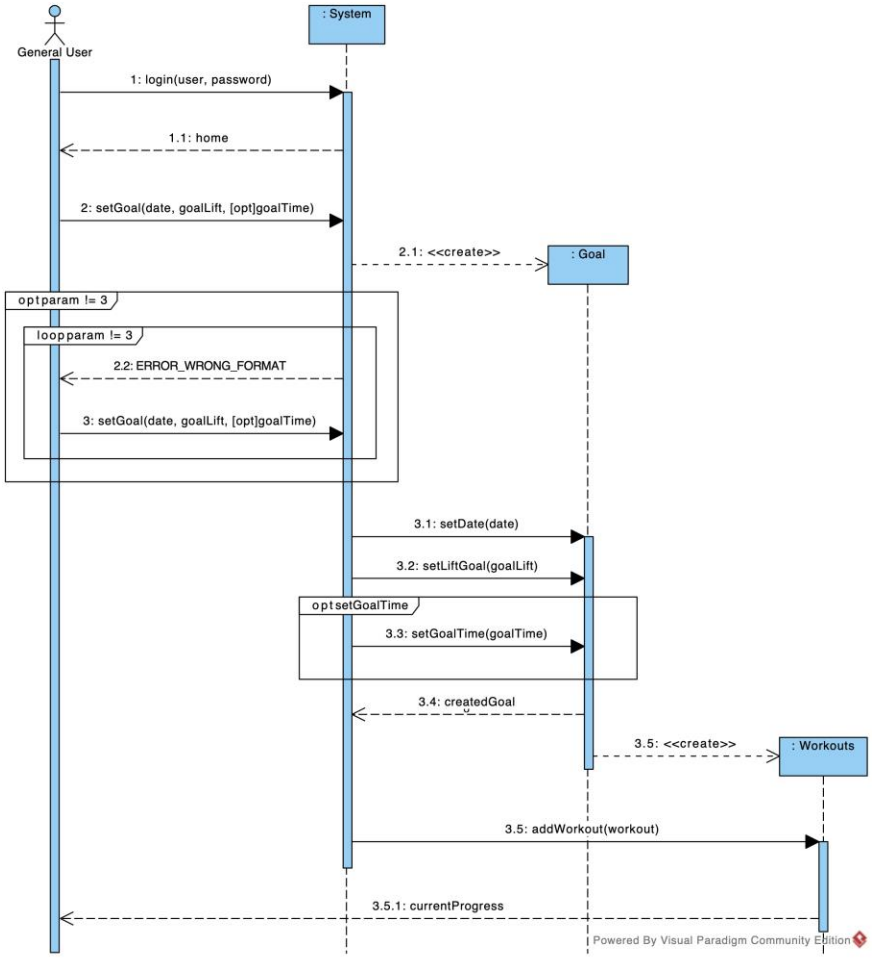
3.a) The user does not fill in all the fields to set a goal

- The missed fields will be marked with a "*"
- The fields that appear will be based on the goal category

SSD:



SD:



UC9 – User Follows a Trainer/Other User – Mac Johnson

Scope: Communication and Media
Level: User Perspective

Stakeholders & Interests:

- General User – a person who wants to follow a fitness trainer/content creator they enjoy
- Fitness Trainer – a content creator who wants to grow their platform/audience

Preconditions:

- General user has a login and password.
- Fitness trainer has a login and password.

Postconditions:

- User follows trainer; user receives notifications and updates on workout plans/posts by the trainer.

Main Success Scenario:

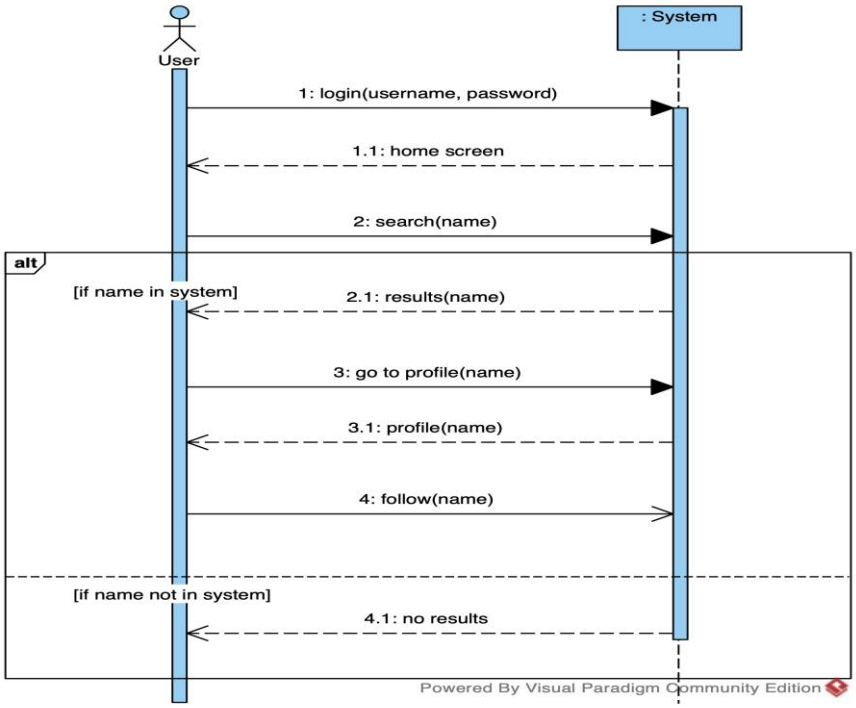
1. User logs into the application.
2. User navigates to the search bar at the top of the home menu screen.

3. User types in username of trainer they want to follow.
4. System returns results of user with that specific username.
5. User clicks on trainer's profile.
6. User clicks the follow button.
7. The user will now receive post/workout notifications from the user they followed.

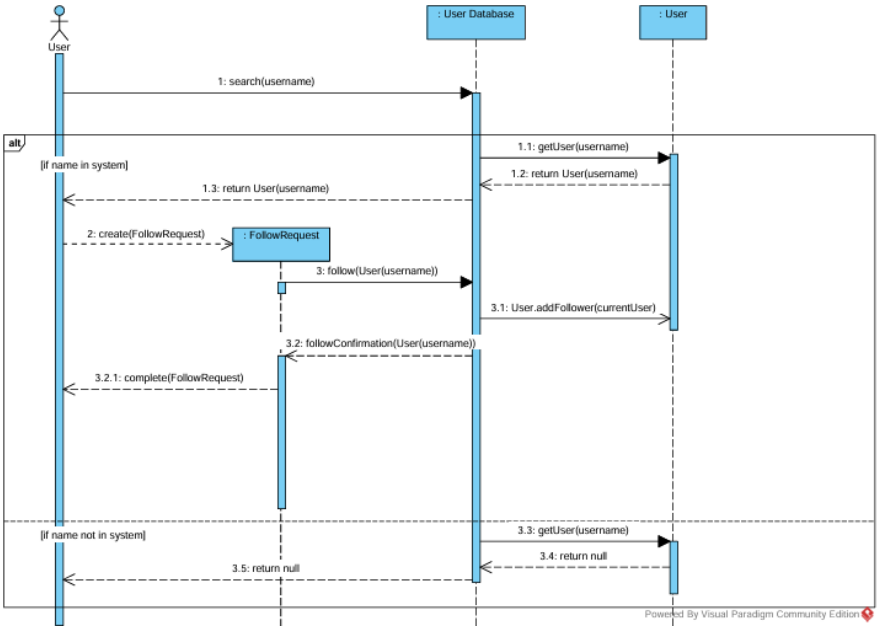
Alternate Paths:

3. a) The user searches for a username that does not exist.
    - The system will return false if an account with the given username does not exist in the database.
    - The search results will be empty, and a message will appear saying "No usernames match your search. Check for any spelling mistakes."

SSD:



SD:



UC10 – User Makes a Post on Their Profile – Mac Johnson

Scope: Communication and Media

Level: User Perspective

Stakeholders & Interests:

- General User – a person wanting to update their circle about their health and fitness journey
- Trainer/Content Creator – a person who wants to grow an audience on our platform by interacting with other users

Preconditions:

- User has a login and password.

Postconditions:

- A new post is added to the user profile for others to see either on the profile or the "feed" tab.

Main Success Scenario:

1. User logs into application.
2. From the homepage, the user selects the "Feed" area.
3. From the feed page, the user selects "New Post" at the top of the screen.
4. User inputs:
   - A stream of text with a limit of 1000 characters.
   - (optional) An image from their device with a file size limit of 5 mb.
5. User clicks "Post" and their post is added to their "Feed" tab and their profile. All other **users who follow this user are notified of a new post.**

Alternate Paths:

4. a) User leaves text box blank when pressing "Post".

- The text box will flash red and notify the user that a post cannot contain empty text.

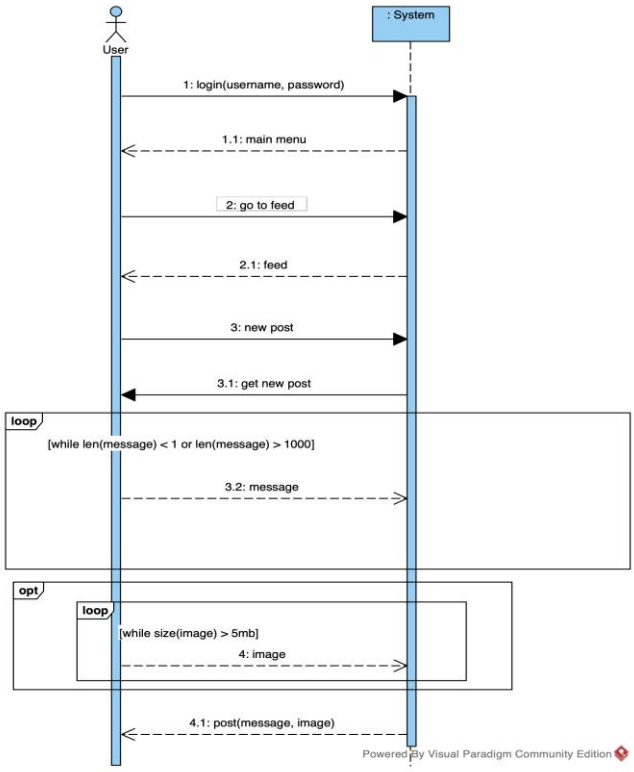
4. b) User inputs text longer than the 1000-character limit.

- The text box will flash red and notify the user that the post is over the 1000-character limit.

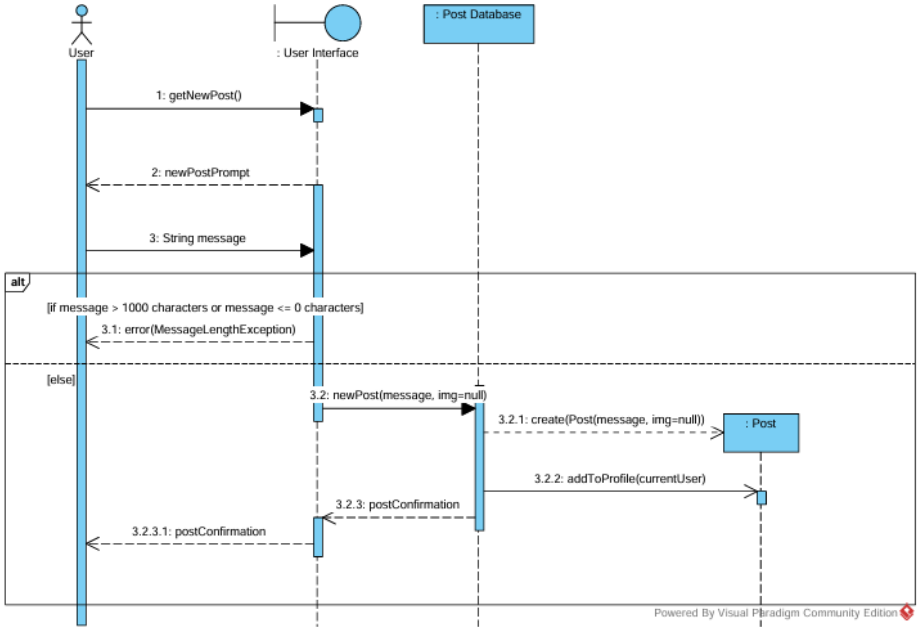
4 .c) User attempts to upload an image that exceeds the 5 mb file size limit.

- The upload will not commence and an error message stating the file limit will be pushed to the user

SSD:



SD:



UC11 – User Views their Historical Health Data – Eli Hall

Scope: Health and Fitness Application

Level: User Goal

Stakeholders & Interests:

- General User – a person using the app who wants to view their past health data.

Preconditions:

- General user has a valid login and password

Postconditions

- A report of the user's data over the specified timeframe is displayed

Main Success Scenario:

1. General user navigates to the health history tab.
2. System prompts the general user to specify the timeframe of historical data they want to view.
3. User inputs a start date, and then a finish date that is set to the current day by default.
4. System returns a report of the user's data from the given period, including their net weight loss, total sleep, total step count, and total calories consumed/burned.
5. System gives user the option to save the report as an image, post the report, or return to the home tab.

Alternate Paths:

3.a) The user does not input a start or end date for the report.

- The system by default will return a report of the user's date from the last week.

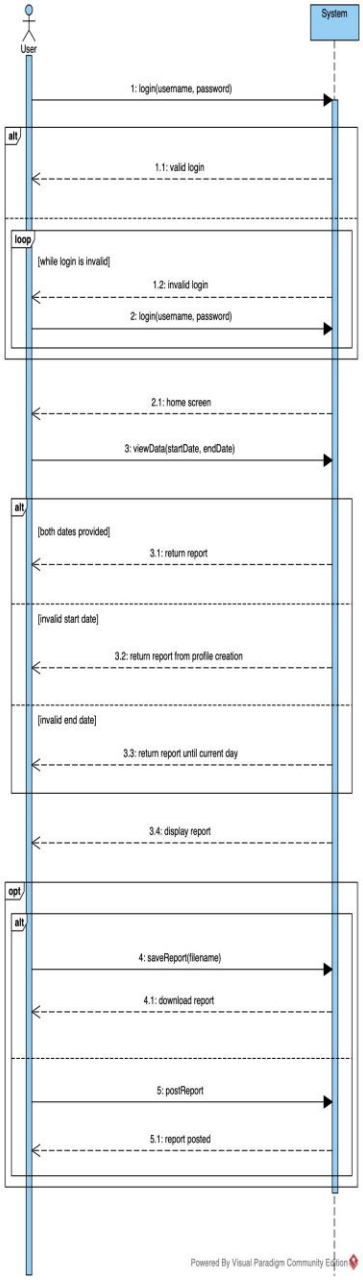
3.b) The user enters a start date that is older than their profile has existed.

- The system will return a report from the creation of the user's profile to the specified end date.

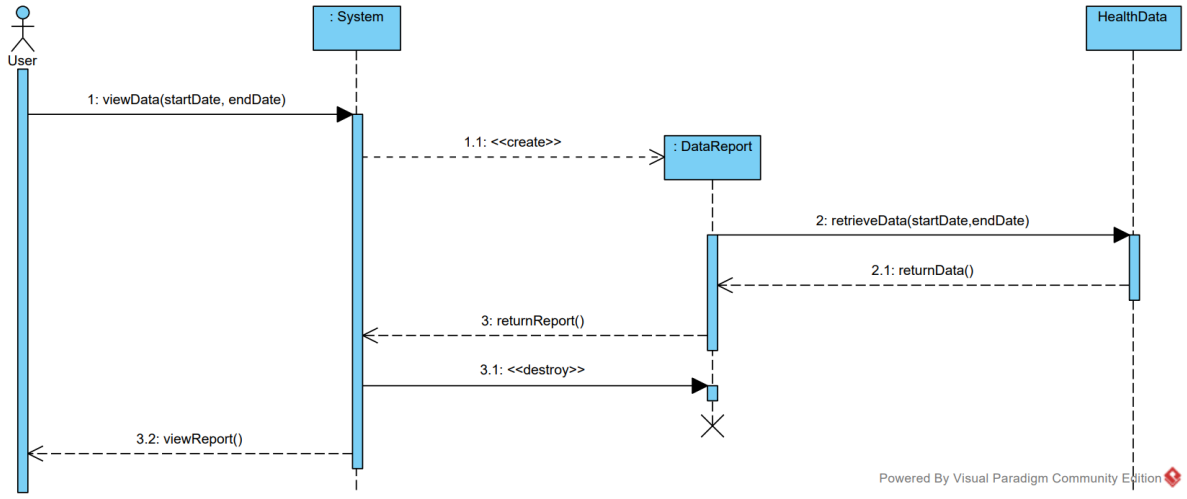
3.c) The user enters an end date that has not happened yet.

- The system will return a report from the specified start date to the current day.

SSD:


User
System
1: login(username, password)
alt
1.1: valid login
loop
[while login is invalid]
1.2: invalid login
2: login(username, password)
2.1: home screen
3: viewData(startDate, endDate)
alt
[both dates provided]
3.1: return report
[invalid start date]
3.2: return report from profile creation
[invalid end date]
3.3: return report until current day
3.4: display report
opt
alt
4: saveReport(filename)
4.1: download report
5: postReport
5.1: report posted
Powered By Visual Paradigm Community Edition

SD:



UC12 – Trainer Views Statistics for their Plan – Eli Hall

Scope: Health and Fitness Application

Level: User Goal

Stakeholders & Interests:

- Fitness Trainer – a person using the app who wants to view the stats of the self-paced plan they've created.

Preconditions:

- Fitness trainer is logged in with a valid username and password
- Fitness trainer has created at least one self-paced plan with at least one registered user.

Postconditions:

- The stats of the trainer's self-paced plan are displayed

Main Success Scenario:

1. Trainer navigates to the exercise plan tab.
2. Trainer navigates to and selects the self-paced plan they want to view.
3. Trainer clicks "view statistics"
4. System displays information about the trainer's plan, including:
    a. How many total users have signed up
    b. How many users are currently signed up
    c. How many total users have completed the plan
    d. The percentage of users who complete the plan
5. System gives the trainer the option to export the statistics (either as an image or a post on their profile), or return to the home tab.

Alternate Paths:

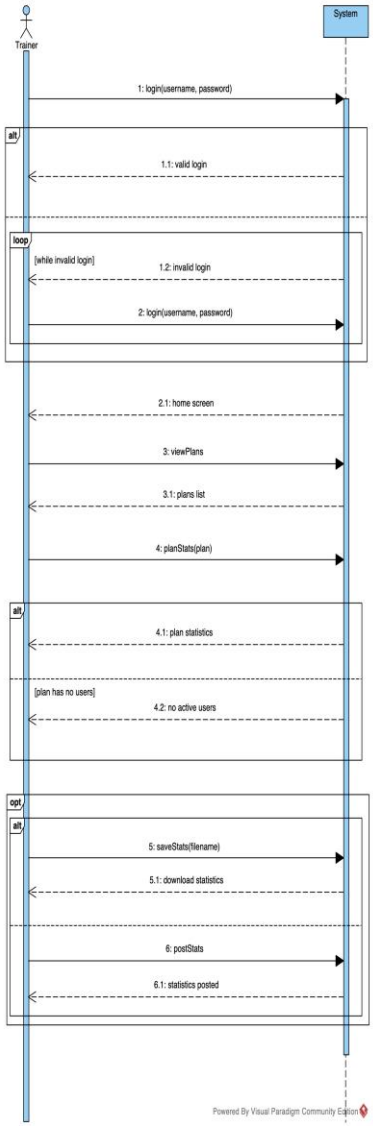
3.a) Trainer selects a plan with no active users

- System displays the message "No active users"
- System prompts trainer to return to the home tab

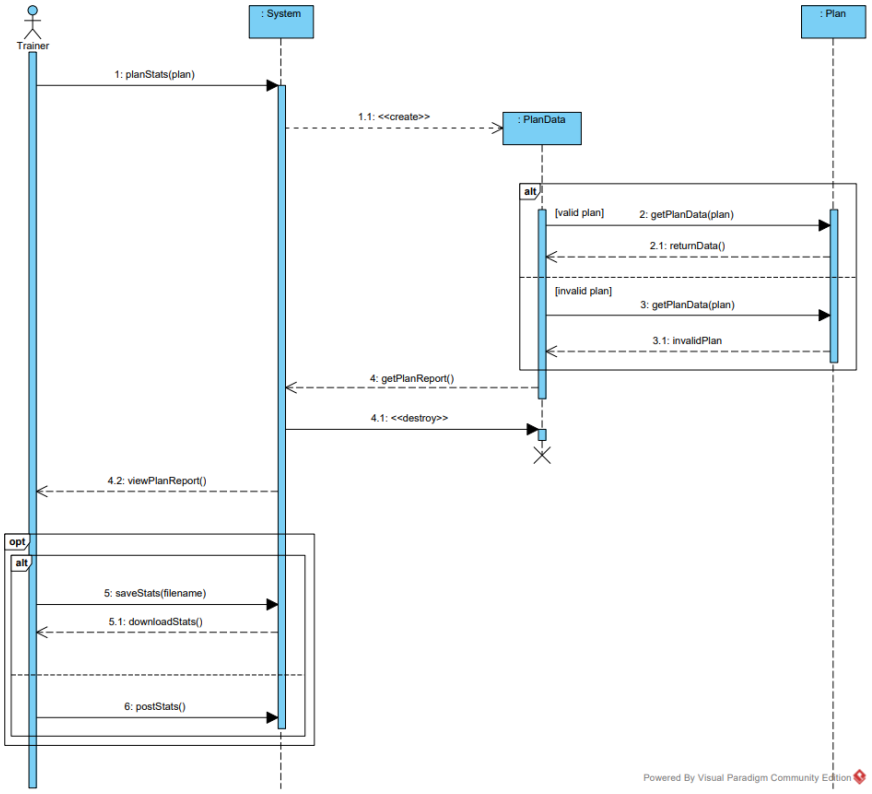
4.a) Trainer has no active plans

- System displays the message "No active plans" with the option to create a new plan

SSD:


Trainer
System
1: login(username, password)
alt
1.1: valid login
loop
[while invalid login]
1.2: invalid login
2: login(username, password)
2.1: home screen
3: viewPlans
3.1: plans list
4: planStats(plan)
alt
4.1: plan statistics
[plan has no users]
4.2: no active users
opt
alt
5: saveStats(filename)
5.1: download statistics
6: postStats
6.1: statistics posted
Powered By Visual Paradigm Community Edition

SD:



UC13 – Trainer Cancels a Class – Sofia Amador

Scope: Health & Fitness Application

Level: User Goal

Stakeholders & Interests:

- Trainer – person who manages scheduled classes by creating, editing, and canceling them.
- General User – person who attends and registers for scheduled classes.

Preconditions:

- Trainer is logged into the system with their valid username and password.
- A scheduled class is already created.
- The scheduled class is upcoming.

Postconditions:

- The class is removed or labeled as cancelled.
- Users that previously registered for the class are notified of the change.

Main Success Scenario:

1. Trainer navigates to manage classes tab.
2. Trainer selects the class they want to cancel.
3. Trainer confirms the selected class for cancellation.
4. The system marks the class as cancelled.
5. Previously registered general users are notified of this change.

Alternate Paths:

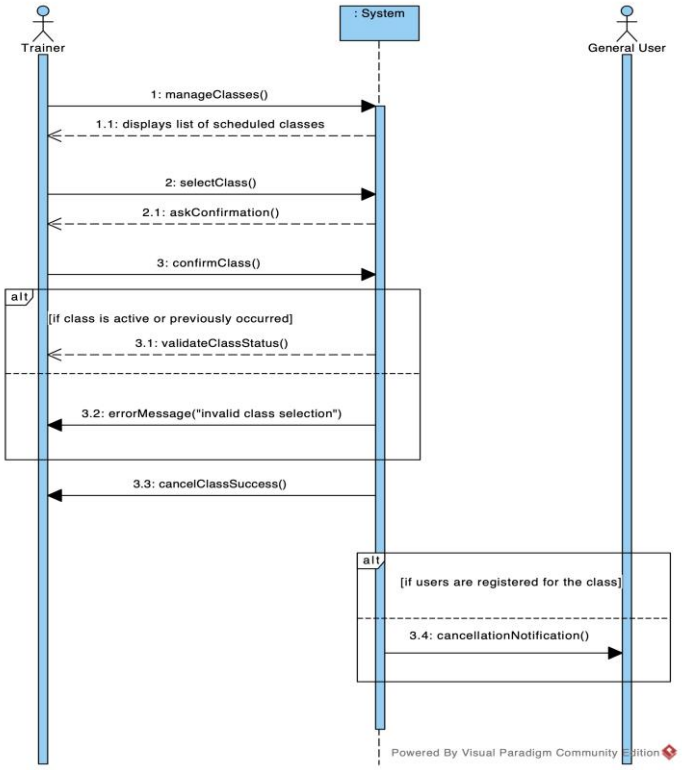
5a) No users were registered for the class.

- No users were notified of the class cancellation.

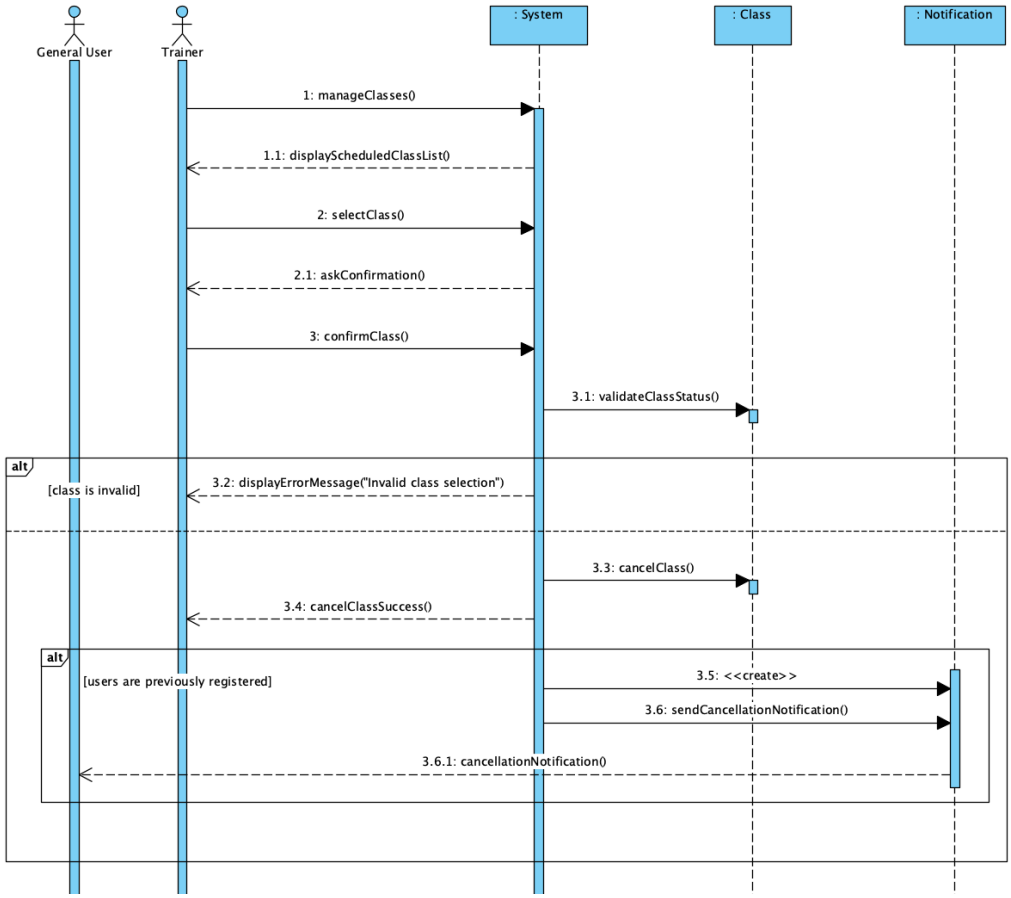
3) Trainer attempts to cancel a class that has already occurred or is currently active.

- The trainer is given an error message and cannot complete this action.

SSD:



SD:



UC14 – General User Creates a Profile - Sofia Amador

Scope: Health & Fitness Application
Level: User Goal
Stakeholders & Interests:

- General User – person who wants to set up their profile.

Preconditions:

- User is logged into the system with their valid username and password.

Postconditions:

- The user's information is saved privately to their user profile.

Main Success Scenario:

1. The user navigates to the edit profile tab.
2. The system prompts the user to enter various information.
    a. First and last name.
    b. Security questions for password reset.
        i. Security question answers.
    c. Fitness level (beginner, intermediate, advanced, elite).
3. The user confirms their information.
4. The system saves the user's profile information.

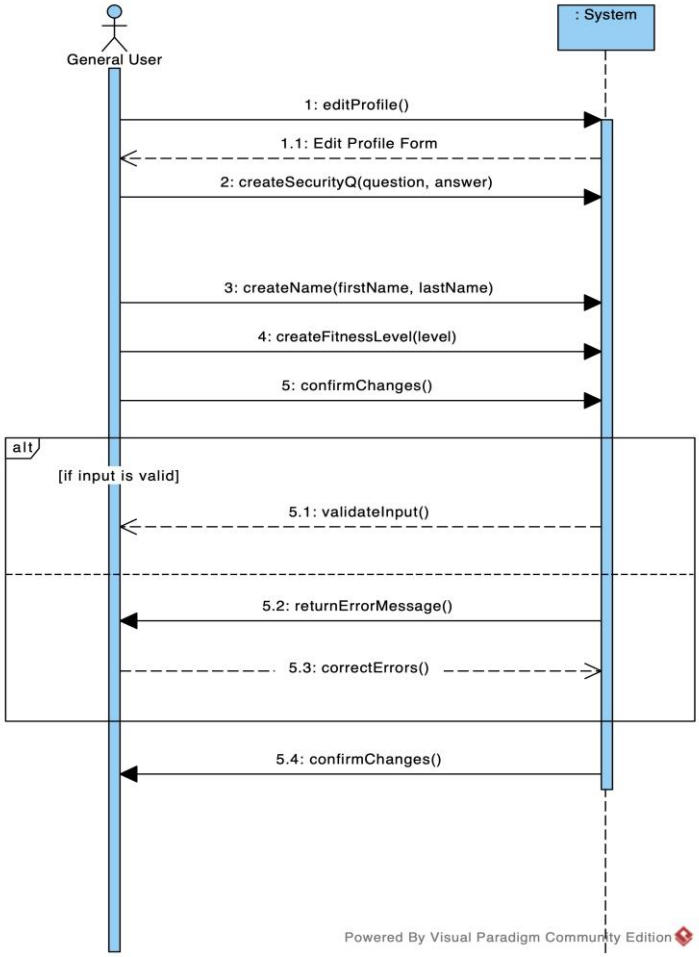
Alternate Paths:

2a) The user inputs a name that includes characters other than alphabetical.

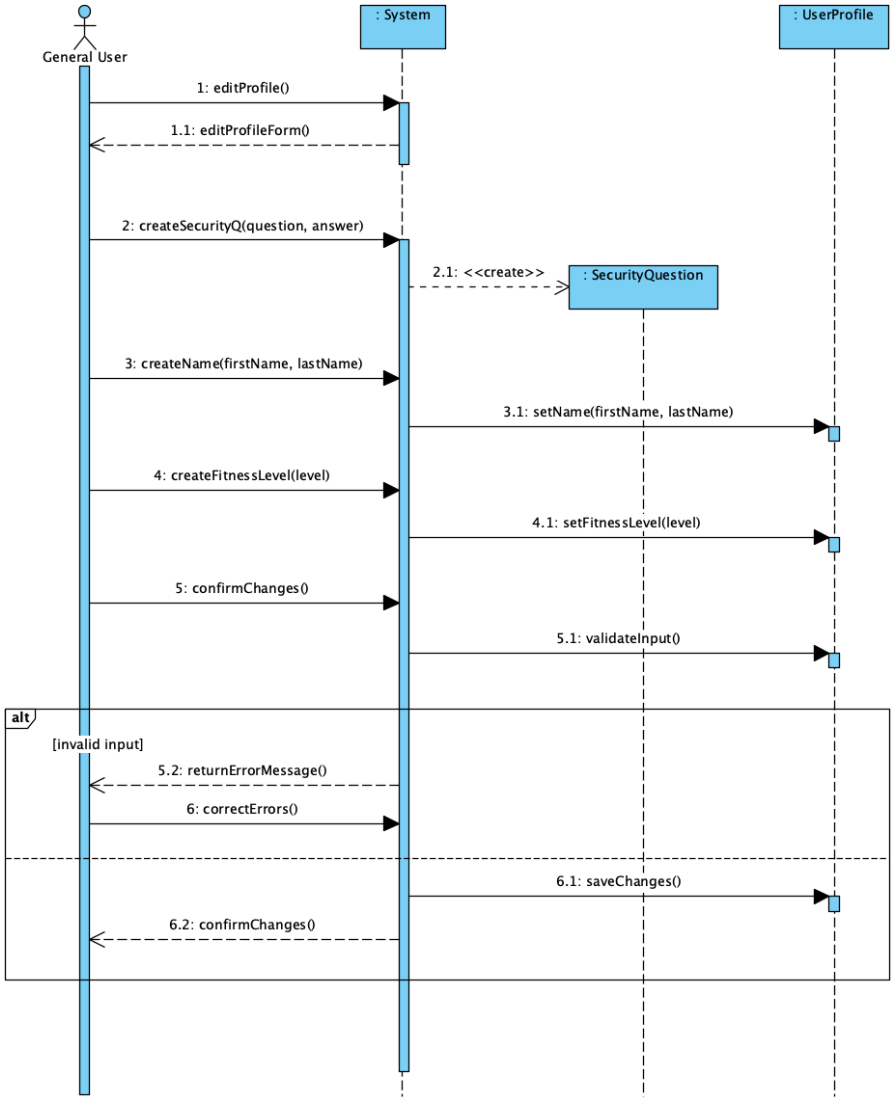
2b) The user writes a security question but does not provide a security answer.

- The system will ask the user to correct these errors or save everything else excluding them.

SSD:



SD:



UC15 – Logging in – Joshua Carroll

Scope: Health & Fitness Application

Level: User Goal

Stakeholders & Interests:

- User (general user, trainer, and admin) – a person who wants to log into the system and access their dashboard

Preconditions:

- The user has an existing account with a valid login and password
- The system is currently operational

Postcondition:

- The user is logged into their dashboard

Main Success Scenario:

1. The user navigates to the login screen
2. The user enters their username and password
3. The system validates their credentials
4. The user has access to their dashboard

Alternate Paths:

3a) An incorrect username or password was entered

- The system returns an error message and prompts again for the credentials
- The user will enter their credentials and try to log in

3b) The user attempts to log in too many times

- The system will lock the account
- The system notifies the user and provides instructions to unlock the account

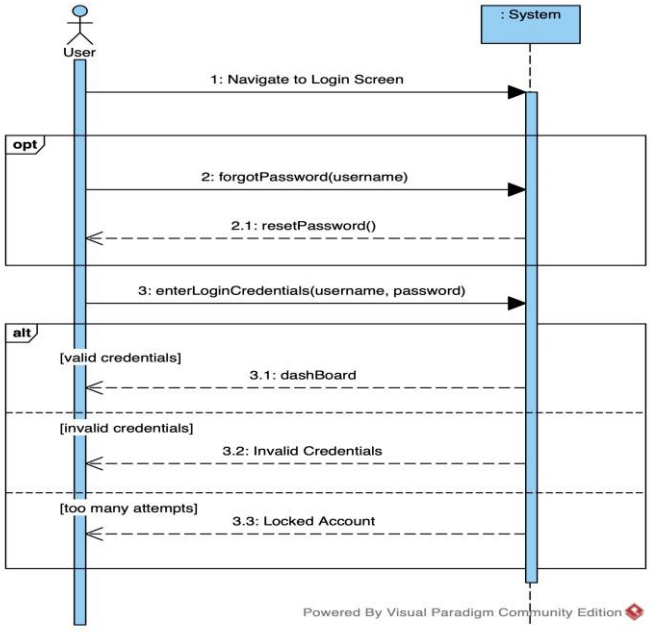
3c) The user forgot their password

- The user will select the forgot password option and go through those steps

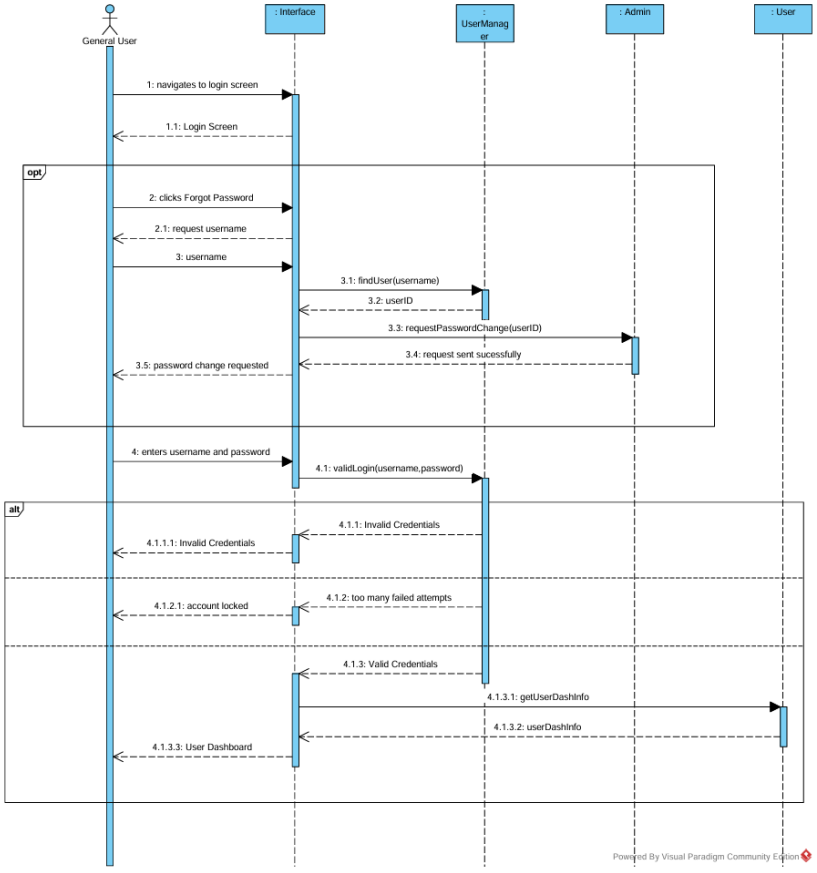
3d) The system is down

- The system will display an error message

SSD:



SD:



UC16 – Trainer Sends Message to Class – Joshua Carroll

Scope: Health & Fitness Application

Level: User Goal

Stakeholders & Interests:

- General User – person who is in the class and wants to receive important updates
- Trainer – person who manages the class and will send important messages regarding the class

Preconditions:

- The trainer is logged into the system
- The trainer has an active class with general users

Postcondition:

- The message is sent to all general users in the class
- Participants are notified through the app

Main Success Scenario:

1) The trainer navigates to the “My Classes” tab
2) The trainer selects the class that the message should be sent to
3) The trainer selects the send message button
4) The system displays an interface to make a message
5) The trainer enters the message
6) The trainer clicks send
7) The system sends the message to all of the general users in the class
8) The system confirms that the message was sent

Alternate Paths:

4a) The trainer tries to send a message to an empty class

- The system prevents the trainer from creating a message and returns an error message

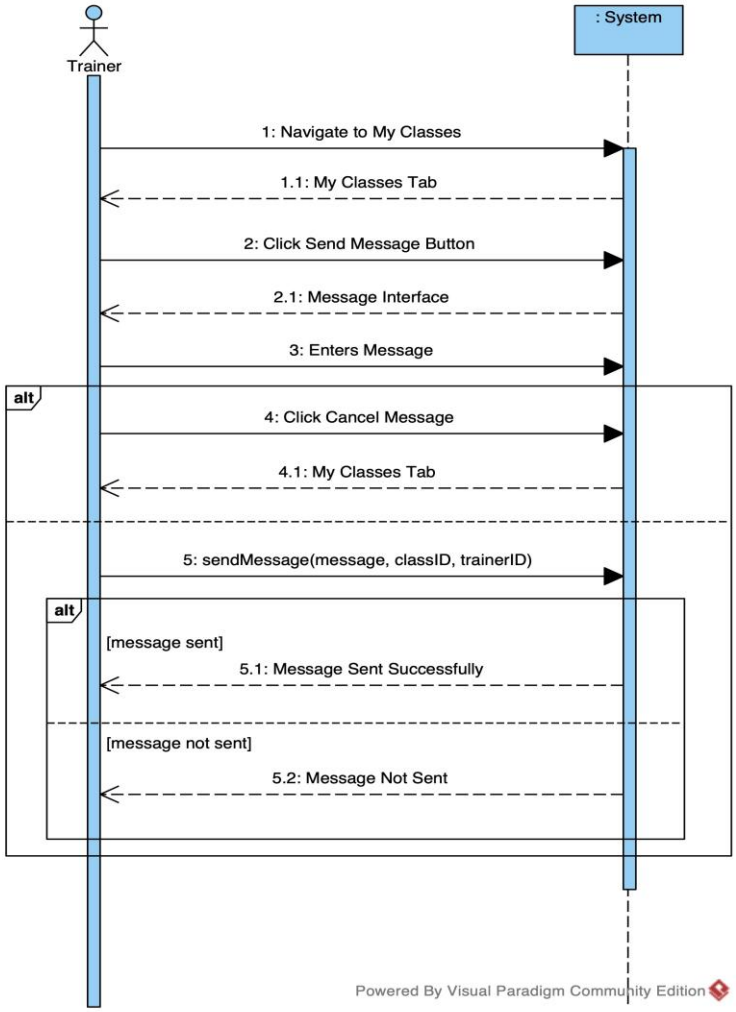
6a) The trainer decides to cancel the message

- The trainer selects exit on the message interface
- The system discards the message

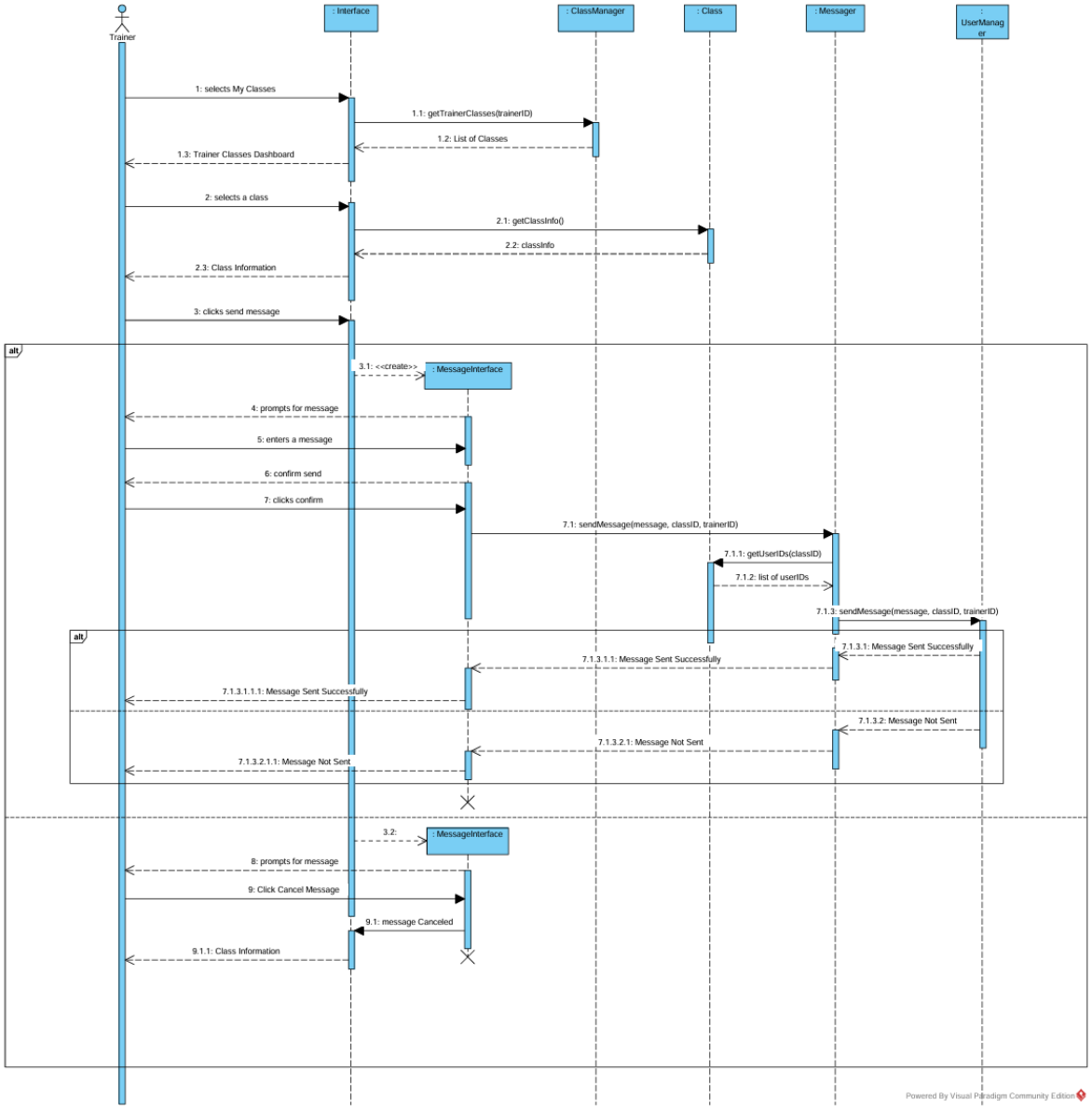
7a) The system fails to send the message due to an error

- The system notifies the trainer that the message could not send
- The trainer can try to send the message again later

SSD:


Trainer
: System
1: Navigate to My Classes
1.1: My Classes Tab
2: Click Send Message Button
2.1: Message Interface
3: Enters Message
alt
4: Click Cancel Message
4.1: My Classes Tab
5: sendMessage(message, classID, trainerID)
alt
[message sent]
5.1: Message Sent Successfully
[message not sent]
5.2: Message Not Sent
Powered By Visual Paradigm Community Edition

SD:



UC17 – Trainer Reviews User Progress – Dakota Hernandez

Scope: Health & Fitness Application

Level: User Goal

Stakeholders & Interests:

- **Trainer** – Wants to review a user's progress to provide personalized guidance.
- **General User** – Wants their trainer to analyze their progress and provide feedback.
- **System Administrator** – Ensures data security and access permissions.

Preconditions:

- Trainer is authenticated and logged into the system.
- The general user has authorized the trainer to view their progress.

Postconditions:

- The trainer successfully views and analyzes the user's progress data.
- The trainer provides feedback or recommendations based on the data.

Main Success Scenario:

1. Trainer logs into the system.
2. Trainer navigates to the "User Progress Review" section.
3. Trainer searches for and selects a user from their list of assigned clients.
4. System retrieves and displays the user's health and fitness data, including:
    a. Logged workouts
    b. Caloric intake
    c. Weight history
    d. Sleep data
    e. Goal progress
5. Trainer analyzes the data and provides feedback through comments or a structured report.
6. Trainer submits feedback, which is saved and sent to the user.
7. User receives a notification about new trainer feedback.

Alternate Paths:

- **3a. If the user has not granted access to their data:**
    - System notifies the trainer that access is restricted.
    - Trainer requests access from the user.
- **5a. If the trainer identifies inconsistencies in the data:**
    - Trainer can request clarification from the user.
    - User can update or edit their logged data.
- **6a. If the system encounters an error while submitting feedback:**
    - System notifies the trainer of the issue.
    - Trainer can retry submission.

Exceptions:

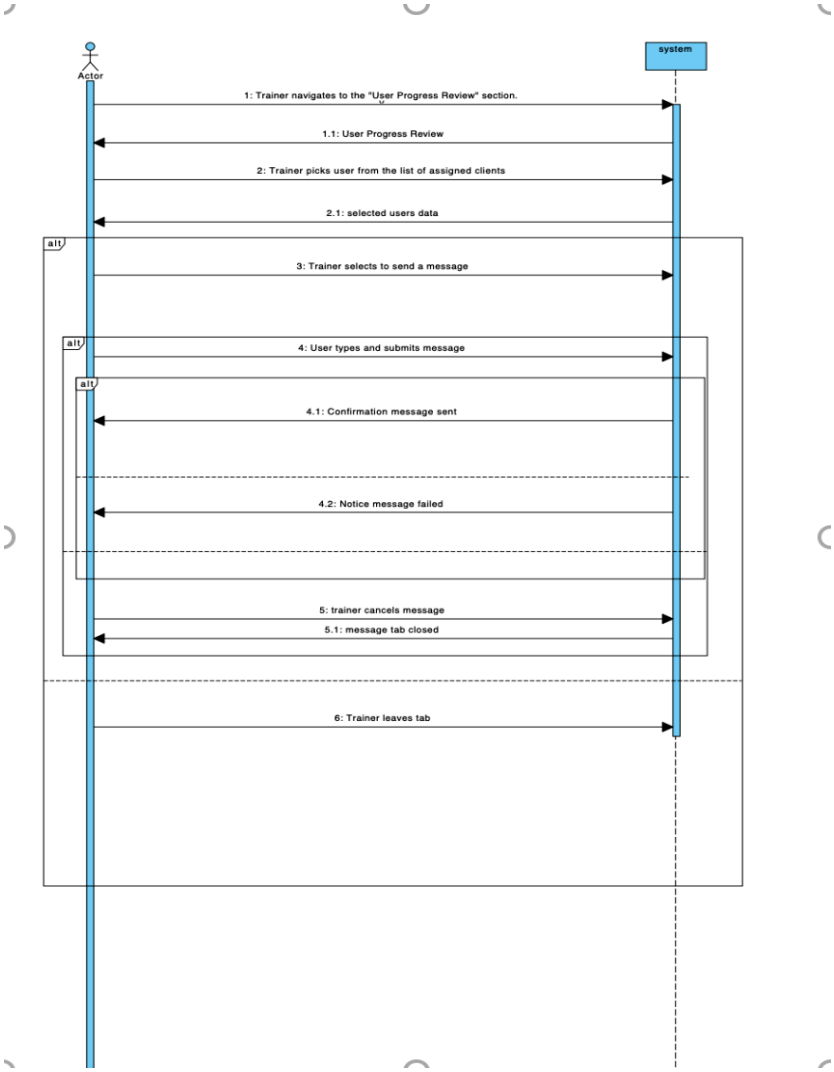
- If the system crashes, the trainer must wait until recovery to access user data.
- If the network connection is lost, the trainer must retry when connectivity is restored.

Business Rules:

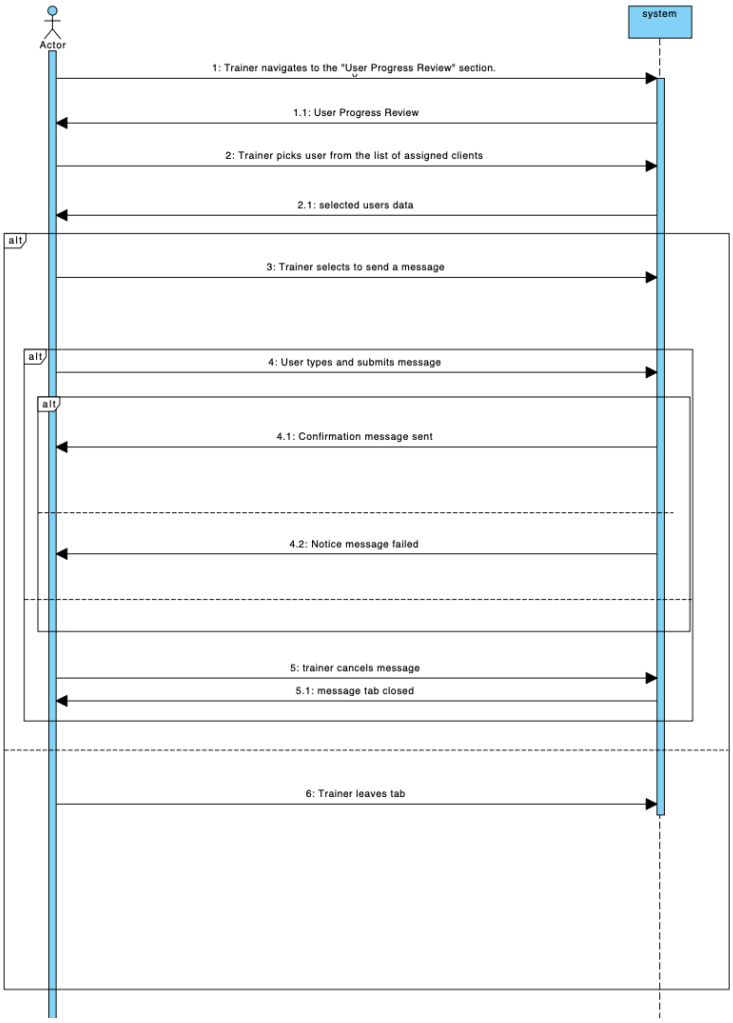
- Only authorized trainers can access a user's progress.
- User must grant explicit permission for trainers to view their progress.

- Trainers cannot edit user data, only provide feedback.

SSD:



SD:



UC18 – User Participates in a Fitness Challenge – Dakota Hernandez

Scope: Health & Fitness Application

Level: User Goal

Stakeholders & Interests:

- **User** – Wants to stay motivated and engage in a fitness challenge.
- **Friends/Community** – Competes and encourages each other.
- **System Administrator** – Ensures fair play and accurate tracking.

Preconditions:

- User is logged into the system.
- A fitness challenge is active or available to join.

- User has the necessary app permissions enabled (e.g., step tracking).

Postconditions:

- User successfully joins, participates in, and tracks progress in the challenge.
- System updates the leaderboard and awards badges/prizes accordingly.

Main Success Scenario:

1. **User logs into the system.**
2. **User navigates to the "Challenges" section.**
3. **User browses and selects a challenge, such as:**
    a. **Step Challenge** (e.g., 10,000 steps per day for a month)
    b. **Strength Challenge** (e.g., 100 push-ups per day)
    c. **Weight Loss Challenge** (e.g., lose 5 lbs in 30 days)
    d. **Streak Challenge** (e.g., workout every day for a week)
4. **User joins the challenge and optionally invites friends.**
5. **System tracks progress automatically (via device sync) or allows manual entry.**
6. **System updates leaderboard rankings and progress badges.**
7. **User receives motivational notifications and progress insights.**
8. **Upon completion, the system rewards the user with:**
    a. Achievement badges
    b. Possible app rewards (discounts, free coaching session, etc.)
    c. Social recognition (if enabled)

Alternate Paths:

- **4a. If no available challenges match the user's interest:**
    - User can create a custom challenge and invite others.
- **5a. If progress tracking fails:**
    - System prompts user for manual input or troubleshooting.
- **8a. If the user does not complete the challenge:**
    - System provides encouragement and suggests alternative goals.

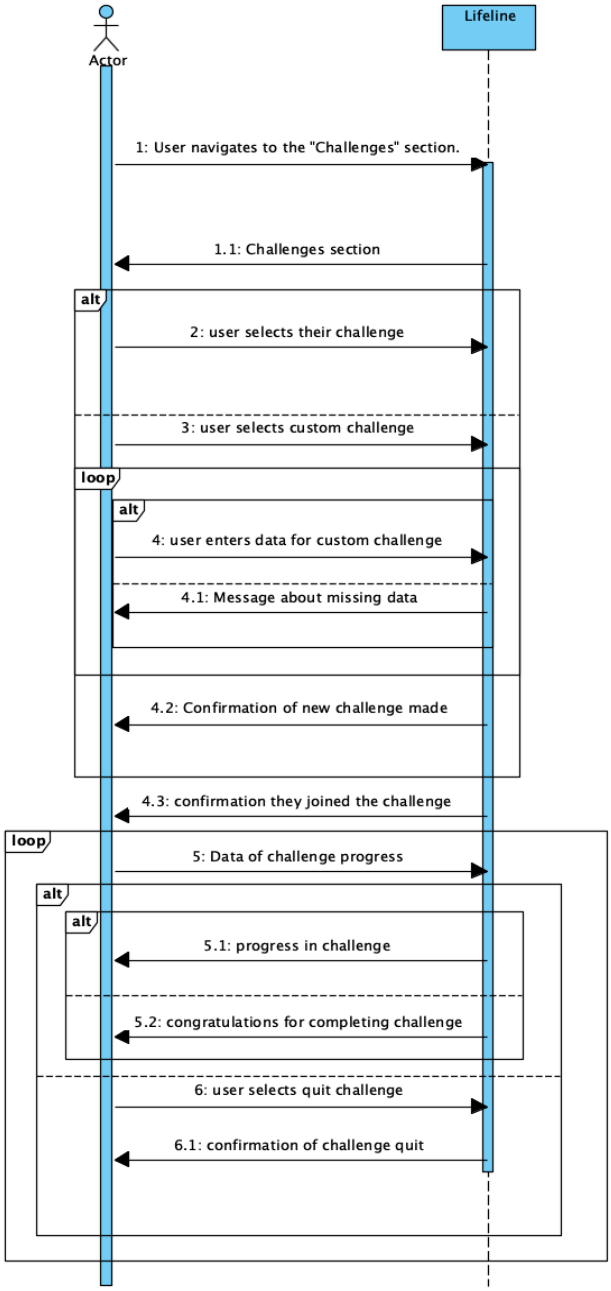
Exceptions:

- If a user is caught cheating (e.g., faking step counts), the system issues a warning or disqualifies them.
- If a challenge has limited spots, the user may be placed on a waitlist.

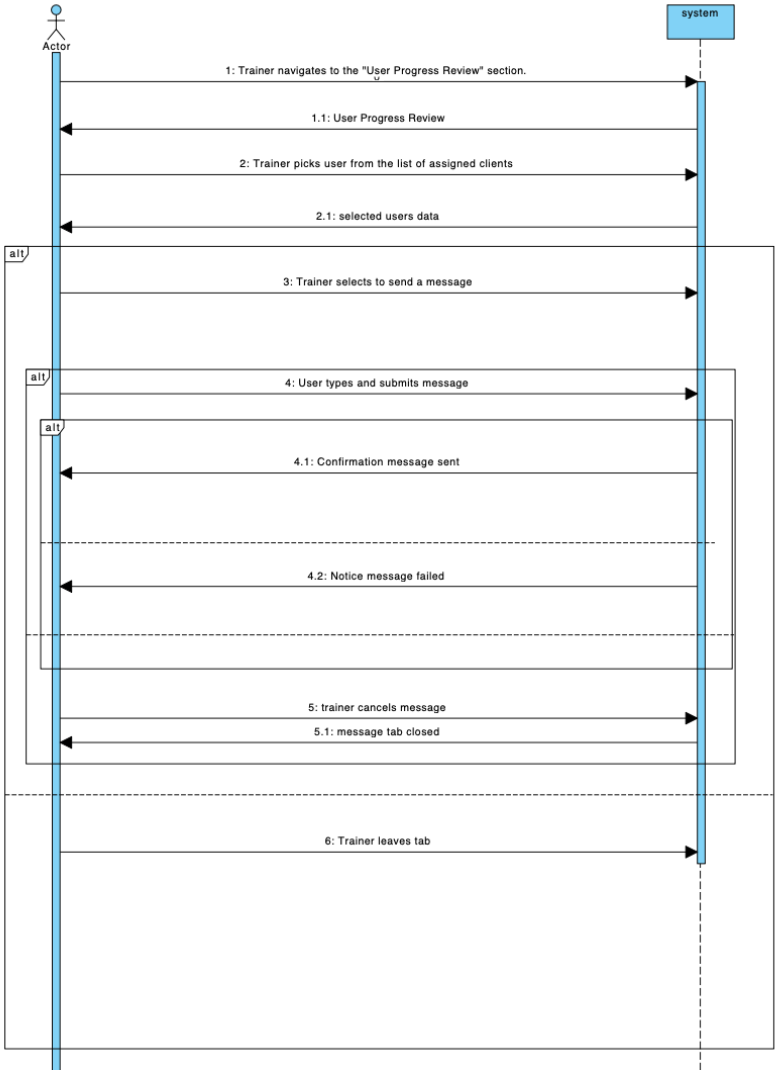
Business Rules:

- Users must consent to challenge tracking and leaderboard participation.
- Privacy settings allow users to hide their rankings if desired.
- System must ensure fair play using validation mechanisms.

SSD:


Actor
Lifeline
1: User navigates to the "Challenges" section.
1.1: Challenges section
alt
2: user selects their challenge
3: user selects custom challenge
loop
alt
4: user enters data for custom challenge
4.1: Message about missing data
4.2: Confirmation of new challenge made
4.3: confirmation they joined the challenge
loop
5: Data of challenge progress
alt
alt
5.1: progress in challenge
5.2: congratulations for completing challenge
6: user selects quit challenge
6.1: confirmation of challenge quit

SD:



10. Operation Contracts

UC1 – User wants to track a workout (Faith Ota)

Contract: trackWorkout

Operation: trackWorkout(date: Date, focus: String, duration: Time)
Cross References: Use Cases: login, Classes: Workout, WorkoutGoal
Pre-condition: The user is logged into the system with valid credentials
Post-conditions:

- A WorkoutItem instance *wi* is created (instance creation)
- *wi* is associated with the current workout (association formed)
- wi.date is set to date (attribute modification)
- wi.focus is set to focus (attribute modification)
- wi.duration is set to duration (attribute modification)
- wi is associated with the WorkoutGoal based on focus match (association formed)

- WorkoutGoal.progress is updated based on duration of wi (attribute modification)

UC2 – User wants to log down a meal (Faith Ota)

Contract: trackMeal(mealTime: Time, dishes: Dish, calories: Integer)
Cross References: Use Cases: login; Classes: Meal, MealGoal
Pre-condition: The user is logged into the system with valid credentials
Post-conditions:

- A MealItem instance *mi* is created (instance creation)
- *mi* is associated with the current meal (association formed)
- *mi.dishes* is set to dishes (attribute modification)
- Each *mi.dishes* is set to the number of calories per dish (attribute modification)
- *mi.calories* is subtracted from total calories of the day (attribute modification)
- MealGoal.progress is updated based on caloric intake (attribute modification)

UC3 – User Sets Reminder(s) for a Workout (Mac Johnson)

Contract CO3: setReminder

Operation: setReminder(name: String, workoutType: String, date: Date, isRepeating: Boolean)
Cross References: Use Cases: User Sets Reminder(s) for a Workout - UC3
Pre-Conditions: user clicks "Set Reminder" button
Post-Conditions:

- A Reminder object instance *reminder* is created. (instance creation)
- *reminder*.name is set to name; *reminder*.type is set to workoutType; *reminder*.date is set to date; *reminder*.isRepeating is set to isRepeating (attribute modification)
- *reminder* was associated with userProfile (association formed)
- *reminder* is added to userProfile.reminders (attribute modification)

UC4 – User Follows a Trainer/Other User (Mac Johnson)

Contract CO9: user.follow

Operation: user.follow(otherUser)
Cross References: Use Cases: User Follows a Trainer/Other User – UC9
Pre-Conditions: user clicks the "Follow" button on another user's profile.
Post-Conditions:

- A reference to *otherUser* is added to *user*.following (attribute modification)
- *otherUser* is associated with *user* (association formed)

UC5 – User Makes a Post on Their Profile (Mac Johnson)

Contract CO10: addNewPost

Operation: addNewPost(String msg, Image img)
Cross References: Use Cases: User Makes a Post on Their Profile – UC10
Pre-Conditions: User clicks "New Post" button on the "Feed" tab of their home page.
Post-Conditions:

- A new Post object instance p is created (instance creation)
- p.message is set to msg (attribute modification)
- p.image is set to reference img uploaded to the client (attribute modification)
- The new object p is returned and linked to the user profile (association formed)

UC6 – User wants to reset their account password (Sofia Amador)

Contract: resetPassword
Operation: **resetPassword(username: String, securityA: String)**
Cross References: Use Cases: login, verifySecurityQ. Classes: User, Account, SecurityQ.
Pre-conditions:

- Administrator is logged into the system with valid credentials.
- General user's account exists in the system.

Post-conditions:

- The Account instance associated with the general user is found (instance creation).
- SecurityQ is validated based on user's answer (attribute modification).
- If securityA matches the security question answer stored, the Account password is updated (attribute modification).
- The Account instance is updated with new password (attribute modification).
- General user can access their account with new password (attribute modification).
- If security question is answered incorrectly, error is logged and user's account is flagged (attribute modification).

UC7 – User tracks sleep hours (Eli Hall)

Contract: trackSleepHours
Operation: **trackSleepHours(sleepHours: Integer)**
Cross References: Use Cases: login, viewSleepStats. Classes: User, SleepRecord, SleepStats.
Pre-conditions:

- General user is logged into the system with valid username and password.
- The system initializes the default sleep hours to 0 each current day.

Post-conditions:

- A general user's SleepRecord instance is created or updated (instance creation).
- SleepRecord.sleepHours is updated from user's input (attribute modification).
- System generates SleepStats based on user's long term sleep recorded data (instance creation).
- General user can view graphs and stats from their tracked sleep hours (association formed).

UC8 – Registering for Scheduled Class (Joshua Carroll)

Contract: RegisterForClass

Operation: RegisterForClass(class ID: Integer, userID: Integer)

Cross References: Use Case: UC7 Registering for a Scheduled class

Pre-conditions:

- The general user is logged into the system with a valid username and password
- The requested class exists and has been created by a trainer
- The general user meets the prerequisites
- The general user has the required equipment
- The general user does not have a scheduling conflict
- The class is not full

Post-conditions:

- A ClassRegistraction instance registration was created
- registration was assocaited with the UserAccount and Class
- The TrainingSession.roster was updated to include the user
- The Trainer.schedule was updated to include the user
- The UserAccount.schedule was updated to include the registered class
- A confirmation message was sent to the UserAccount and the Trainer

UC9 – Contract: createExercisePlan – Dakota Hernandez

Operation: createExercisePlan(planType: String, name: String, description: String, equipment: String, fitnessLevel: String, sessionLength: Time, scheduleDetails: ScheduleDetails, prerequisites: String)

Cross References: Use Cases: UC1 - Trainer Creates an Exercise Plan; Classes: Trainer, ExercisePlan

Pre-conditions:

- Trainer is authenticated and logged into the system.

Post-conditions:

- An ExercisePlan instance ep is created (instance creation).
- ep is associated with the Trainer who created it (association formed).
- ep.planType is set to planType (attribute modification).
- ep.name is set to name (attribute modification).
- ep.description is set to description (attribute modification).

- ep.equipment is set to equipment (attribute modification).
- ep.fitnessLevel is set to fitnessLevel (attribute modification).
- ep.sessionLength is set to sessionLength (attribute modification).
- If ep.planType is 'Scheduled Class', scheduleDetails are set (attribute modification).
- ep.prerequisites are set to prerequisites (attribute modification).
- System validates entered details (validation process).
- System saves ep and makes it available for general users (persistence).
- Trainer receives confirmation of successful plan creation (notification).

Alternate Paths:

1. If the trainer is not authenticated:
    a. System prompts the trainer to log in.
2. If required details are missing:
    a. System highlights missing fields and prompts trainer to complete them.
3. If system validation fails:
    a. System displays an error message and the necessary corrections.
4. If the system encounters an error while saving the plan:
    a. System notifies the trainer of the issue.
    b. Trainer can retry saving the plan.
5. If the trainer wishes to modify the plan after creation:
    a. Trainer navigates to the plan management section.
    b. Trainer updates details and submits changes.
    c. System validates and saves the modifications.

Exceptions:

- If the system crashes during plan creation, trainer must reattempt after system recovery.
- If the network connection is lost, trainer must retry when connectivity is restored.

Business Rules:

- Only authenticated trainers can create exercise plans.
- Self-paced plans must contain at least session length and required equipment details.
- Scheduled classes must include date, time, and prerequisites.
- Trainers can update or delete their plans but not those created by other trainers.

UC 10 – User wants to set a goal – Faith Ota

Contract: setGoal
Operation: setGoal(Date: date, Integer: goalLift, Time: goalTime)
Cross References: UC 1 – tracking a workout
Pre-condition: The user is logged into the system with valid credentials
Post-conditions:

- **A goal instance, *gi,* is created (instance creation)**
- ***gi.date* is set to date (attribute modification)**
- ***gi.goalLift* is set to goalList if the category is "Weightlifting" (attribute modification)**
- ***gi.goalTime* is set to goalTime if the category is "Cardio" (attribute modification)**
- **A Workout collection instance, *wci,* is *created* (instance creation)**
- ***gi* is associated with *wci* (association formed)**
- ***gi.progress* is modified based on *wci's* general trend (attribute modification)**

UC 11 – User Views their Historical Health Data – Eli Hall

Contract: viewData

Operation: viewData(Date: startDate, Date: endDate)

Cross References: Use Cases: User Views their Historical Health Data - UC11

Pre-condition:

- The general user is logged into the system with valid credentials.
- The general user selects the health history tab.

Pos-conditions:

- A HealthReport instance *hr* was created (instance creation)
- *hr* was associated with user's HealthMetrics (association formed)
- *hr.start* is set to startDate (attribute modification)
- *hr.end* is set to endDate (attribute modification)
- *hr.caloriesConsumed* is set to the user's total caloric intake over the period (attribute modification)
- *hr.caloriesBurned* is set to the user's total burned calories over the period (attribute modification)
- *hr.totalWeight* is set to the user's total weight lost over the period (attribute modification)
- *hr.totalSleep* is set to the user's total sleep hours over the period (attribute modification)
- *hr.totalSteps* is set to the user's total step count over the period (attribute modification)

UC12 – Trainer Views Statistics for their Plan – Eli Hall

Contract: viewStatistics

Operation: viewStatistics(Plan: plan)

Cross References: Use Cases: Trainer Views Statistics for their Plan – UC12, Trainer Creates an Exercise Plan - UC1

Pre-condition:

- Trainer is authenticated and logged into the system

Post-conditions:

- A PlanData instance *pd* was created (instance creation)
- *pd* was associated with associated with plan (association formed)
- *pd.totalUsers* was set to the total number of users who have signed up (attribute modification)
- *pd.activeUsers* was set to the number of currently signed-up users (attribute modification)
- *pd.completedUsers* was set to the number of users who completed the plan (attribute modification)
- *pd.completionRate* was calculated as the percentage of signed-up users who completed the plan (attribute modification)

UC13 – Trainer Cancels a Class (Sofia Amador)

Contract: cancelClass

Operation: **cancelClass(trainerID: String, classID: String)**

Cross References: Use Cases: manage classes, notify users. Classes: Trainer, Classes, Notification.

Pre-conditions:

- Trainer is logged into the system with their valid username and password.
- The scheduled class is already created in the system and is associated with a trainer.

Post-conditions:

- Class instance associated with classID is found (instance retireval).
- Class status is updated to cancelled (attribute modification).
- Notification instance is created and sent to general users of the registered class (instance creation).
- If no general users were registered for the class, no notifications are sent out (conditional removal).

UC14 – General User Creates a Profile - (Sofia Amador)

Contract: createProfile

Operation: **createProfile(firstName: String, lastName: String, securityQ: String, securityA: String, fitnessLevel: String)**

Cross References: Use Cases: login, reset password, update profile. Classes: UserAccount, Profile.

Pre-conditions:

- User is logged into the system with their valid username and password.

Post-conditions:

- A new Profile instance is created and associated with the username (instance creation).
- User’s firstName and lastName are stored (attribute modification).

- The securityQ and securityA are saved for password reset (attribute modification).
- The fitnessLevel is recorded (attribute modification).

UC15 – Logging in (Joshua Carroll)

Contract: enterLoginCredentials

Operation: enterLoginCredentials(username: String, password: String)

Cross References: Use Case: UC15 Logging in

Pre-conditions:

- The user has a valid username and password
- The user's account is not locked
- The system is currently operational

Post-conditions:

- A UserSession instance session was created
- session was associated with the authenticated UserAccount
- The session.userRole was determined by generalUser, trainer, or admin
- The user is redirected to their dashboard
- If the login failed the failedAttempts for the UserAccount is incremented
- If failedAttempts exceeds the limit then the UserAccount is locked

UC16 – Logging in (Joshua Carroll)

Contract: sendMessage

Operation: sendMessage(message: String, classID: Integer, trainerID: Integer)

Cross References: Use Case: UC16 Trainer Sends Message to Class

Pre-conditions:

- The trainer is logged in with a valid username and password
- The trainer has an active class with participants
- The message is not empty
- The messaging system is functional

Post-conditions:

- A Message instance msg was created
- msg was associated with the class

- msg.content was set to the given message
- The system sent msg to all class participants
- The system logged msg for future reference

UC17- Trainer Reviews User Progress (Dakota Hernandez)

Contract: reviewUserProgress

Operation: reviewUserProgress(userID: String)

Cross References: Use Cases: UC8 - Trainer Reviews User Progress; Classes: Trainer, User, ProgressData

Pre-conditions:

- Trainer is authenticated and logged into the system.
- The user has authorized the trainer to view their progress.

Post-conditions:

- A ProgressData instance pd is retrieved for the specified user (data retrieval).
- pd is associated with the user (association formed).
- Trainer analyzes the progress data, including workouts, caloric intake, weight history, sleep data, and goal progress (data analysis).
- Trainer provides feedback, which is stored in the system (data persistence).
- User receives a notification regarding the new trainer feedback (notification).

Alternate Paths:

- If the user has not granted access, the system notifies the trainer and allows a request for access.
- If inconsistencies in the data are found, the trainer can request clarification from the user.
- If feedback submission fails, the system notifies the trainer and allows retry.

Exceptions:

- If the system crashes, the trainer must wait until recovery.
- If network connectivity is lost, the trainer must retry once restored.

Business Rules:

- Only authorized trainers can access user progress data.
- Trainers cannot modify user data, only provide feedback.

UC18 - User Participates in a Fitness Challenge (Dakota Hernandez)

Contract: participateInChallenge

Operation: participateInChallenge(challengeID: String, userID: String)

Cross References: Use Cases: UC9 - User Participates in a Fitness Challenge; Classes: User, Challenge, Leaderboard

Pre-conditions:

- User is logged into the system.
- A fitness challenge is active and available for participation.
- The user has necessary permissions enabled (e.g., step tracking).

Post-conditions:

- A ChallengeParticipant instance cp is created and associated with the user (instance creation, association formed).
- System tracks user progress in the challenge via device sync or manual input (progress tracking).
- System updates leaderboard rankings and progress badges (data update).
- User receives motivational notifications and progress insights (notification).
- Upon completion, system rewards user with achievement badges, possible app rewards, or social recognition (reward allocation).

Alternate Paths:

- If no available challenges match the user's interest, they can create a custom challenge.
- If tracking fails, the system prompts the user for manual input or troubleshooting.
- If the user does not complete the challenge, the system provides encouragement and suggests alternative goals.

Exceptions:

- If cheating is detected, the system issues a warning or disqualifies the user.
- If challenge slots are full, the user is placed on a waitlist.

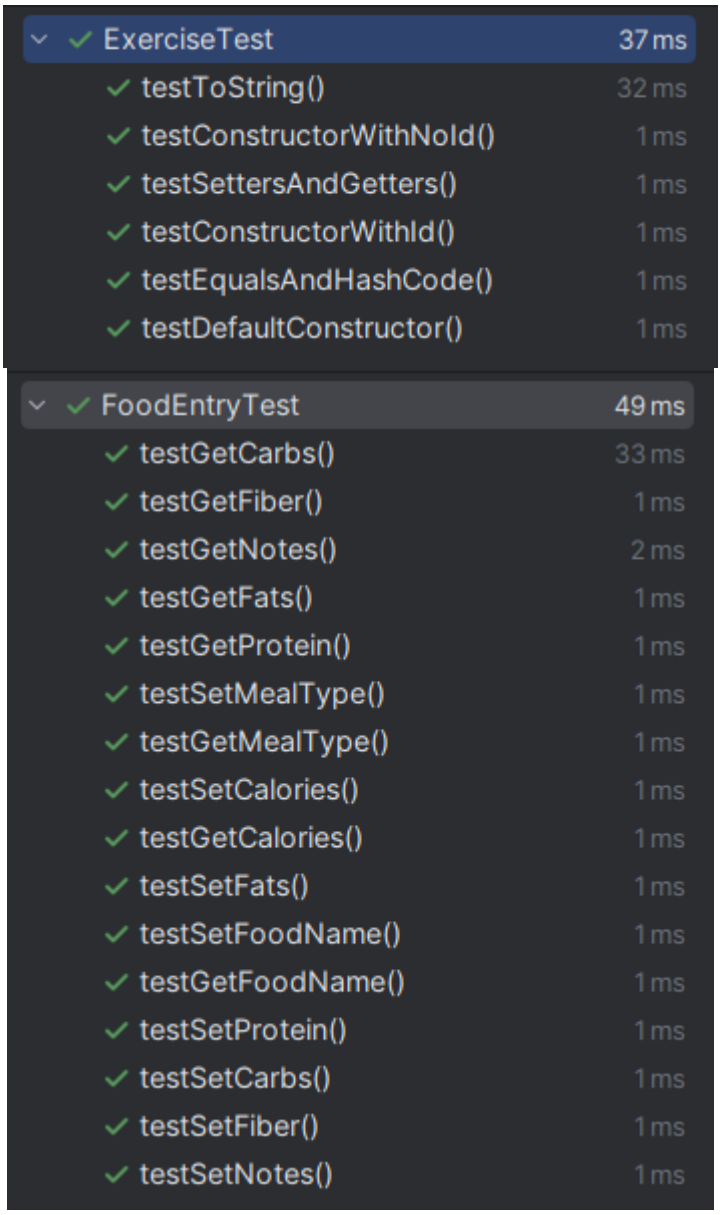
Business Rules:

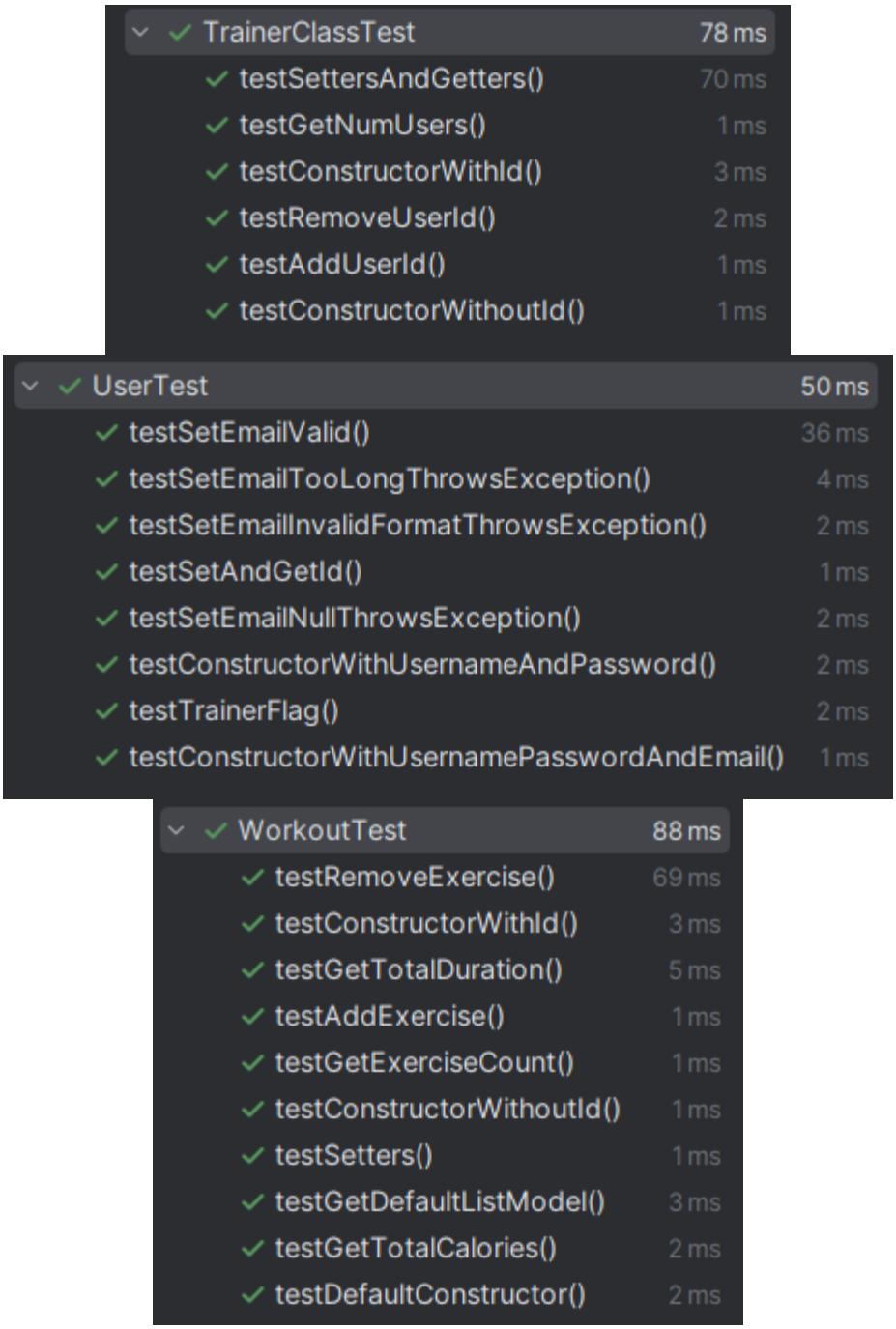
- Users must consent to challenge tracking and leaderboard participation.
- Privacy settings allow users to hide their rankings if desired.
- The system must ensure fair play using validation mechanisms.

11. Testing

JUnit Testing

We used JUnit to test our model classes that were responsible for holding and processing data. This included the User class itself, where we tested the different fields like username, password, and email with varying degrees of validity, as well as the trainer flag to determine if Trainer accounts are being created correctly. We tested all the variations of each constructor in each class as well as their getters and setters. In the TrainerClass class, we also tested exercise addition/removal and checked computed metrics like total duration, total calories burned, and exercise count.




TrainerClassTest 78 ms
testSettersAndGetters() 70 ms
testGetNumUsers() 1 ms
testConstructorWithId() 3 ms
testRemoveUserId() 2 ms
testAddUserId() 1 ms
testConstructorWithoutId() 1 ms
UserTest 50 ms
testSetEmailValid() 36 ms
testSetEmailTooLongThrowsException() 4 ms
testSetEmailInvalidFormatThrowsException() 2 ms
testSetAndGetId() 1 ms
testSetEmailNullThrowsException() 2 ms
testConstructorWithUsernameAndPassword() 2 ms
testTrainerFlag() 2 ms
testConstructorWithUsernamePasswordAndEmail() 1 ms
WorkoutTest 88 ms
testRemoveExercise() 69 ms
testConstructorWithId() 3 ms
testGetTotalDuration() 5 ms
testAddExercise() 1 ms
testGetExerciseCount() 1 ms
testConstructorWithoutId() 1 ms
testSetters() 1 ms
testGetDefaultListModel() 3 ms
testGetTotalCalories() 2 ms
testDefaultConstructor() 2 ms

Full Test Cases

Use Case: *CreateExercise*

- Assigned tester: Eli Hall

TC ID#	Scenario/Condition	Test Inputs	Expected Result	Actual Result
TC1	User logs a valid daily exercise	Name = " Stretch", Focus = "Flexibility", Description = "Exercises to loosen muscles"	"Exercise Created Successfully" message displayed; fields are cleared	"Exercise Created Successfully" message displayed; fields are cleared
TC2	User submits with missing Name field	Name = "", Focus = "Strength", Description = "Weight lifting"	System prompts user to fill required fields; exercise is not saved	“Exercise could not be added” message displayed; fields are cleared
TC3	User submits with missing Focus field	Name = "Upper Body ", Focus = "", Description = "Upper body strength workout"	System prompts user to fill required fields; exercise is not saved	“Exercise could not be added” message displayed; fields are cleared
TC4	User submits with missing Description field	Name = "Evening Jog", Focus = "Cardio", Description = ""	System prompts user to fill required fields; exercise is not saved	“Exercise could not be added” message displayed; fields are cleared
TC5	User tries to create an exercise that already exists	Name = "Stretch", Focus = "Flexibility", Description = " Exercises to loosen muscles " (already stored)	"Exercise Already Exists" message displayed; fields are cleared	“Exercise could not be added” message displayed; fields are cleared
TC6	User enters very long text in fields	Name = 300 characters long, Focus = 300 characters long, Description = 1000 characters long	System doesn't crash; either truncates or saves correctly	“Exercise could not be added” message displayed; fields are cleared
TC7	System fails to save exercise (internal error)	Name = "Power Yoga", Focus = "Flexibility", Description = "Advanced yoga flow" (simulate file write error)	"Exercise could not be added" message displayed; fields remain	"Exercise could not be added" message displayed; fields remain

Use Case: *TrackSleep*

- Assigned tester: Eli Hall

TC ID#	Scenario/Condition	Test Inputs	Expected Result	Actual Result
SP1	Launch SleepPage with valid user and default goal	Valid CURRENT_USER with no goal	Page loads with title “Sleep Tracker – [date]”, and goal set to 60	Page loads with title “Sleep Tracker – [date]”, and goal set to 25
SP2	Launch SleepPage with valid user and custom goal	Valid CURRENT_USER with goal = 25	Page loads with goal set to 25, progress bar max is 25	Page loads with goal set to 25, progress bar max is 25
SP3	Enter valid sleep hours	‘8’	Progress bar increases by 8, progress label updates, field clears	Progress bar increases by 8, progress label updates, field clears
SP4	Enter non-numeric input	‘abc’	Dialog shown: “Please enter a valid number of sleep hours.”	Dialog shown: “Please enter a valid number of sleep hours.”
SP5	Enter sleep hours below valid range	‘-1’	Dialog shown: “Sleep hours must be between 0 and 24.”	Dialog shown: “Sleep hours must be between 0 and 24.”
SP6	Enter sleep hours above valid range	‘25’	Dialog shown: “Sleep hours must be between 0 and 24.”	Dialog shown: “Sleep hours must be between 0 and 24.”
SP7	Exceed weekly sleep goal	Enter enough to surpass goal	Progress bar capped at goal, label shows full progress (25 /25)	Progress bar capped at goal, label shows full progress (25 /25)

Use Case: *TrackCalories*

- Assigned tester: Eli Hall

TC ID#	Scenario/Condition	Test Inputs	Expected Result	Actual Result
CMP1	Load UI with default values	Launch CalorieMacroPage	Three tabs displayed: "Breakfast", "Lunch", "Dinner"; tables are initially empty	Three tabs displayed: "Breakfast", "Lunch", "Dinner"; tables are initially empty
CMP2	Add a food entry to Breakfast tab	Select "Breakfast" tab → Click "Add	Row added to Breakfast table;	Row added to Breakfast table;

		Food Entry" → Enter valid food entry	progress bar updated based on entry calories	progress bar doesn't update but label does
CMP3	Add food entry to Lunch tab	Select "Lunch" tab → Click "Add Food Entry" → Enter valid food entry	Row added to Lunch table; progress bar updated	Row added to Lunch table; progress bar doesn't update but label does
CMP4	Add food entry with 0 calories	Enter food item with 0 calories	Entry added to table; progress bar remains unchanged	Entry added to table; progress bar remains unchanged
CMP5	Delete selected food entry	Select a row in Breakfast → Click "Delete Selected Entry"	Entry removed; progress bar decrements accordingly	Entry removed; progress bar is unchanged, label updates
CMP6	Edit selected food entry	Select a row in Lunch → Click "Edit Selected Entry" → Change calorie value	Row updated with new values; progress bar recalculated	Row updated with new values; progress bar unchanged but label updates
CMP7	Switch between tabs	Click between "Breakfast", "Lunch", and "Dinner" tabs	Correct table shown for each tab; previous table data preserved per tab	Correct table shown for each tab; previous table data preserved per tab
CMP8	Progress bar max reached	Add food entries until total calories ≥ 2000	Progress bar filled completely; does not exceed max	Progress bar doesn't update
CMP9	Add invalid entry in dialog	Open dialog → Leave required fields empty or invalid input → Press OK	Dialog remains open and shows error; no entry added	Dialog remains open and shows error; no entry added

Use Case: *RecordWeight*

- Assigned tester: Eli Hall

TC ID#	Scenario/Condition	Test Inputs	Expected Result	Actual Result
RW1	Load UI with existing weight entries and goal	User has entries in DB and a weight goal set	Table and chart reflect real weight data; second line appears for goal weight	Table and chart reflect real weight data; second line appears for goal weight
RW2	Enter valid weight for	Input "130" in	Entry is added to	Entry is added to

	today	weight field and click "Confirm"	chart and table; appropriate success message shown; curWeight is updated to 130	chart and table; appropriate success message shown; curWeight is updated to 130
RW3	Enter weight exactly equal to goal weight	Goal = 190, input = 190	"Congrats, you've met your weight goal!" message displayed	"Congrats, you've met your weight goal!" message displayed
RW4	Enter weight below goal weight	Goal = 190, input = 180	"Congrats, you've met your weight goal!" message displayed	"Congrats, you've met your weight goal!" message displayed
RW5	Enter weight above goal but lower than previous weight	Goal = 190, previous = 155, input = 150	"Great work, you're making progress towards your weight goal" message shown	"Great work, you're making progress towards your weight goal" message shown
RW6	Enter weight higher than goal and higher than previous	Goal = 190, previous = 145, input = 160	"Uh oh, double check your plan to stay on track" message displayed	"Uh oh, double check your plan to stay on track" message displayed
RW7	Enter invalid input (non-numeric)	Input "abc" and click "Confirm"	Error dialog shown with message: "Please enter a valid number."	Error dialog shown with message: "Please enter a valid number."
RW8	Confirm weight input with empty text field	Leave input field empty → Click "Confirm"	Error dialog shown; no update to chart or table	Error dialog shown with message: "Please enter a valid number."
RW9	UI resizes and components remain visible	Resize window	SplitPane adjusts correctly; chart and table still visible	SplitPane adjusts correctly; chart and table still visible
RW10	Confirm persistence of weight entry	Enter valid weight and relaunch app	Entry persists in database and reappears in chart/table	Entry persists in database and reappears in chart/table

Use Case: *LoginPage*

- Assigned tester: Eli Hall

TC ID#	Scenario/Condition	Test Inputs	Expected Result	Actual Result
LP1	Valid Login	Username: user, Password: user	A regular user successfully logs	A regular user successfully logs

			in.	in.
LP2	Invalid Login	Username: wrong Password: wrongpass	Error displayed, “Invalid username or password. Try again.”	Error displayed, “Invalid username or password. Try again.”
LP3	Empty Login Fields	Username: (empty) Password: (empty)	Error displayed for missing login input.	Error displayed for missing login input.
LP4	Valid Account Creation	Username: newuser Email: new@user.com Password: pass123 Confirm: pass123	Creates a new general user account.	Creates a new general user account.
LP5	Password Mismatch	Password: abc123 Confirm: xyz123	Displays an error that passwords do not match.	Displays an error that passwords do not match.
LP6	Trainer Account Creation	Account type: Trainer selected	Creates a new account with isTrainer set to true.	Creates a new account with isTrainer set to true.
LP7	General User Account Creation	Account type: General User selected	Creates a new account with isTrainer set to false.	Creates a new account with isTrainer set to false.
LP8	Empty Registration Fields	Username left empty	Displays error, “Username or password cannot be empty.”	Displays error, “Username or password cannot be empty.”
LP9	Duplicate Username	Username: user	Displays error, “A user with this username already exists.”	Displays error, “A user with this username already exists.”

12. Contributions

Contribution Percentiles (based on lines of code from Github): Dakota – 41.66%, Faith – 25.60 %, Mac – 13.52%, Joshua – 13.23%, Sofia – 3.35%, Eli – 2.64 %

SUMMARY OF CONTRIBUTIONS
Sofia Amador:

I implemented the “Create Profile”, “Reset Password”, and “Administrator Interface” pages, while keeping in mind how they interact with each other given what access general users and administrators should be able to perform and access. The create profile page has first and last name, fitness level, and two drop down menus for security question choices for the user to choose from to reset their password later. The reset password page accesses these questions so existing users can create a new password or reset a forgotten one. Finally the administrator interface is a large table with all of the users in the system’s database, keeping in mind that administrators should not be able to access private data such as passwords, but can aid in the resetting password verification process.

Joshua Carroll:
I created the TemplateFrame which has functions that every frame in the app needs, this contains things such as splitting the window into different frames, having a function to add the menu bar at the top, and the logout option. This also contains a function for a simple text field and label for getting input. I then implemented the create exercise using this template frame. Using the exercises I implemented the create workout page which allows the user to select from their exercises and add them to the workout, then they can create it after filling out the name and the date. Then I implemented the workout and exercise with the database to properly store it. Then I created the track workout where a user can view their workouts and the exercises that are in the workout, this also contains information about the workout as a whole and the individual components. I then created the trainer classes where trainers have the option to create a class. To do this I reused code from the workout and had trainerClass inherit from workout while adding to it. The trainer can create the class the exact same way they create a workout by clicking the create class button. The trainer can also manage their classes by viewing the manage class page and it will populate a chart with all of their classes. A user can register for a class by viewing the track classes page and click the register check box on the class they want to join. The track classes will be populated with all the classes that are made by trainers. This page also contains information about the number of people in the class and the username of the trainer.

Eli Hall:
I implemented several tracking pages, specifically the “Sleep” page and the foundations of the RecordWeight page. The sleep tracking page allows the user to record their sleep on a day-to-day basis, contributing to their weekly goal. A progress bar & label shows the user their progress towards their weekly goal, which can be updated via the SetGoals page. I included input validation so that users can only enter valid amounts of sleep, disregarding negative numbers and letters. For the RecordWeight page, I created the system where users input their current body weight and receive a report based on the progress they’ve made towards their weight goal. I was also in charge of the project testing, where I used a combination of JUnit and manual testing to affirm that every aspect of the project is working as it should. I used JUnit tests primarily for the model classes like Exercise and User that represent data and logic, confirming that their member methods work as intended. I ran 5 major manual tests to cover the user’s interaction with various pages from start (login) to finish (sign out), logging the results of different inputs during the process. The processes/use cases I choose to focus on were CreateExercise, RecordWeight, TrackCalories, RecordSleep, and LoginPage.

Dakota Hernandez:

I started initially by creating a basic calorie tracking page that uses variables to hold all data, this later evolved into a csv then finally a database connection for calorie tracking. I helped create and implement the template frame that helped ensure DRY principles. I implemented the template frame into multiple already made classes. I created a Themes class that holds a set of preset colors so the classes can all match and have the same UI theme. This was included with a UI overhaul that fixed the size of the pages that open along with matching old UI elements with the new ones. I used a date picker API to create a built-in calendar on the homepage so you can change the date. I implemented this later to update the data read from the database when you switch to other pages. I did a full overhaul on the weight tracking page as beforehand it was just a text box that received a weight. The weight class now has a table displaying previous weights along with a graph that displays weight over time using the JFreechart api. I helped connect. the goals, weight, calories to the database. I created a reminder system that uses Mailjet api to send email reminders to the user. I added a calorie calculator popup for the set goals page so the user can calculate their calorie goals rather than just guessing

Mac Johnson:
I was in charge of the User Account system, the Login Page, the Self-Paced Plans (User and Trainer interfaces) database integration for all of those mentioned. I created the database and login system to allow users to create accounts with a username, password, email, and specified preference for typical user or trainer account settings. All of these fields in the login page are input verified, including the email, where it must be a valid email address to be entered. I also created a system to integrate the current_user variable across interfaces, allowing the app to remember which user is currently logged in. I created the ability for Trainers to create self-paced plans, which include daily workouts, which are composed of individual exercises. Each of these have a corresponding table in our SQLite database which I was in charge of keeping track of. This allows users to search for and register for classes created by other trainers. Trainers can edit, delete, and view the corresponding information for each individual plan they create, all in the interface implemented only if the account is specified as a trainer. I also created the project website from scratch using HTML, CSS, and JavaScript.

Faith Ota:
I created the home page frame from an abstract class to have the other screens extend the template frame based on the page's purpose. The home page includes our logo, pages menu bar, and a logout menu bar. With the home page, I added a menu bar object to be implemented on every page so that the user can navigate back to the home page or to another feature. The menu bar required each button to create a new frame of that page's class while also disposing of the current frame to prevent having numerous windows open at the same time. In addition to the home page, I created a SetGoals page for the user to create goals and connect the other tracking pages with the input values. The SetGoals page includes text fields to receive input and dialog messages if the values are not possible (nonpositve weight, sleep hours, and caloric intake) or to confirm that the new goal values are saved. However, I also considered the case when the user does not want to use the goals feature, so I gave the calorie and sleep variables an initial value.

13. Setup Guide

1. Navigate to the website and download the zip file.

(https://macj2005.github.io/PawPlatesProjectSite/)

2. Extract the zip file and open the project in a Java IDE.
3. Run the Main class OR run the jar file. The app should open after build.
4. Create a new account.
5. Go back to the login screen and sign in with your new username and password.
6. Get that grind and have fun!

14. Group Meeting (05/01/2025)

This past Thursday, May 1st, we discussed and distributed the final tweaks that needed to be completed before submitting the final project. These aspects were: the SetGoals page (Faith), registering for a lesson (Joshua), a reminder system via email (Dakota), connecting the tracking attributes to a database system that is based on the user (Mac), implementing a reset password feature for the admin user (Sofia), and implementing the test package (Eli). We also utilized this time to work on our final responsibilities.