(Screenshot of word pdf of template)

OsoFit Vision Document

Version 1.0 | Date: 14 September 2025

1. Introduction

The purpose of this Vision document is to define the high-level needs, features, and objectives of OsoFit, a health and fitness tracking application designed to help users monitor daily wellness, set goals, and engage with trainers and peers. The document captures the motivation behind the product and outlines its major capabilities.

References

- Group Project Deliverable 7, OsoFit (Sept 2025)
- Visual Paradigm Diagrams (Domain Model, SSDs)
- GitHub Repository: https://github.com/jaackfont05/OsoFit

2. Positioning

Problem Statement

The problem of inconsistent health tracking and structured goal monitoring affects both individuals who want a better tool to manage their lifestyle and trainers who need easier access to clients. The impact of this problem is unstructured routines that lack guidance and accountability. A successful solution would provide integrated health, nutrition, and exercise tracking, personalized goal setting and progress reporting, and trainer-led classes with social accountability.

Product Position Statement

For health-focused individuals and trainers who need a unified health and fitness management platform, the OsoFit App is a mobile application that allows users to track health metrics, manage workouts, and set personalized goals. Unlike other fitness trackers that isolate these features, our product integrates tracking, social interaction, and trainer functionality into one cohesive system.

3. Stakeholder and User Descriptions

Stakeholder Summary

Name	Description	Responsibilities
General Users	Individuals seeking to improve fitness	Log health data, view progress, follow plans
Trainers	Certified professionals	Create exercise plans, lead classes, monitor clients
Admins	System administrators	Manage accounts, ensure security, handle support

User Environment

Accessible via mobile and desktop; designed for quick daily interaction and clear dashboards. Requires internet connectivity for data syncing.

Summary of Key Stakeholder or User Needs

Need	Priority	Current Solution	Proposed Solution
Track all health metrics	High	Multiple separate trackers	Unified OsoFit platform
Social accountability	Medium	Social media or none	Friend/group systems
Trainer-client integration	High	Manual messaging	Built-in trainer tools

4. Product Overview

Product Perspective

OsoFit is a standalone mobile and web application designed to synchronize user data through secure cloud storage. It interacts with standard APIs for authentication and optional wearable device integration.

Assumptions and Dependencies

- Users have access to smartphones or computers.
- Requires stable internet connection.
- Depends on secure authentication and database infrastructure.

5. Product Features

- User Management: Create, log in, reset passwords.
- Health Tracking: Record calories, weight, sleep, and water intake.
- Workout Logging: Save workout sessions and monitor progress.
- Goals and Reporting: Set weight/sleep goals, view analytics.
- Trainer Tools: Create classes, self-paced or scheduled plans.
- Class Management: Register, track attendance, manage sessions.
- Social Features: Add friends, create groups, send requests.
- Injury Logging: Record injuries and adjust plans accordingly.

6. Other Product Requirements

- Platform: Mobile (Android/iOS) and Web.
- Performance: Real-time syncing between users and trainers.
- Security: Encrypted user data, password recovery.
- Scalability: Supports growing user base and integration with wearable APIs.
- Usability: Intuitive interface suitable for all experience levels.
- Documentation: Online help and tutorials integrated into the app.

Use Cases:

Jack Fontenot
1. Fully-dressed

ID UC Change User/Trainer Password
Scope Login system
Level User goal
Stakeholders and interests
User
- Person that forgot or needs to change their password to track their fitness goals.

Trainer
- Person that forgot or needs to change their password to assist users.

System admin
- Person responsible for login system run

Precondition: Password change is requested.

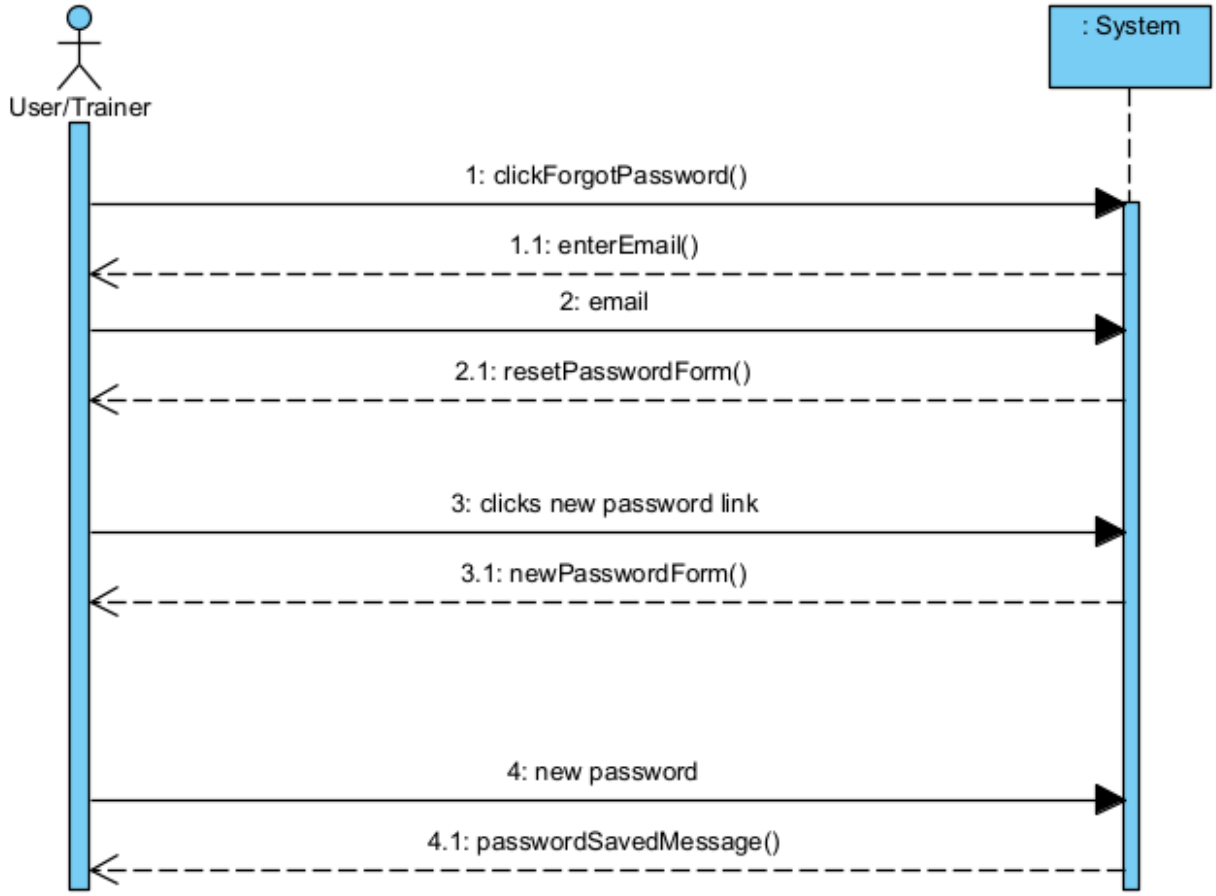
Postcondition: System updates the account's password.

Main success scenario:

1. User/trainer forgets their password
2. User/trainer clicks "forgot password?"
3. User/trainer enters their email and clicks "Send password reset link"
4. User/trainer clicks the link in their email
5. User/trainer enters the updated password into the "new password" and "confirm new password" boxes
6. User/trainer clicks "update password"
7. System saves the account's new password
8. User/trainer logs in with new password

Alternate paths:
a.* anytime the process does not work
1. User/trainer emails user support
2. Admin reads user/trainer's email and manually resets password

5.a "new password" and "confirm new password" boxes do not match
1. User retypes in both boxes



Contract CO2: Change User/Trainer password
Cross-references: none
Pre-Condition: User/Trainer has an account and has forgotten their password
Post-Conditions:

- uPass was changed (attribute modification)

2. Fully-dressed

ID UC Logging Calories and Nutritional Macros
Scope Nutrition tracking system
Level user goal
Stakeholders and interests
User

- Person that is interested in logging their calories and macros

Trainer

- Person that is interested in checking the diet of their clients

System admin

- Person responsible for nutrition tracking system run

Precondition: User has eaten a meal

Postcondition: Meal is saved and nutritional info updated

Main success scenario:

1. User eats a meal
2. User logs in to the app
3. User clicks “log meal”
4. User enters the name of the meal
5. User enters the number of calories and amount of fat, carbs, and protein in grams of the meal
6. User clicks “confirm meal”
7. System saves all the info of the meal
8. System adds meal to the user’s previous meals list
9. System adds the amount of calories and macros towards their daily goals of each
10. Trainer can view the user’s meal

Alternate paths:

a.* anytime the app does not respond

1. User/trainer restarts the app

2.a If user cannot login to the app

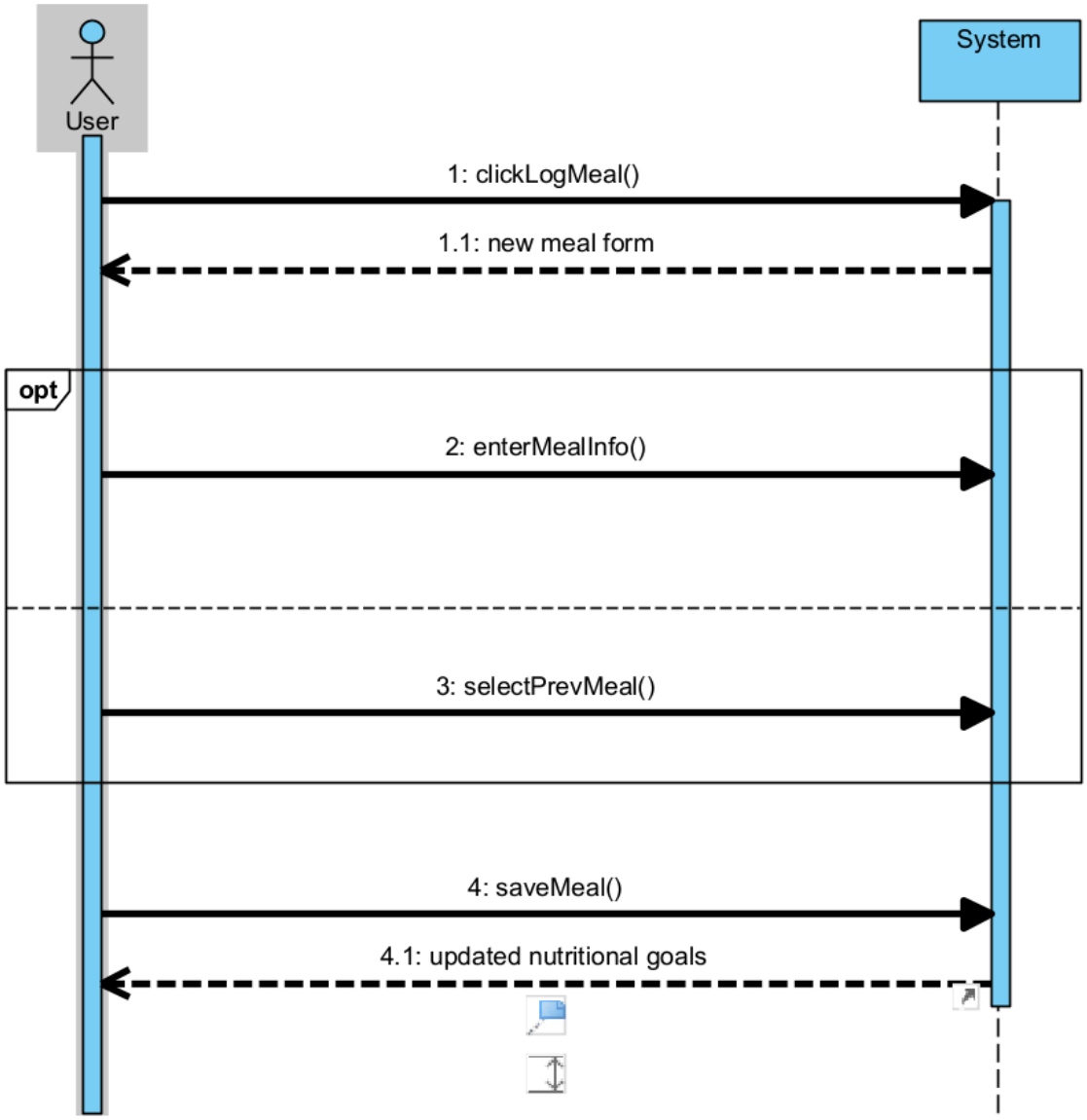
1. User resets their password

4.a If user ate a meal that was previously logged

1. User finds the meal in their previous meals list
2. User selects the meal
3. System saves all the info of the meal

5.a if user does not enter calories, fat, carbs, or protein,

1. The system updates according to the fields entered



Contract CO2: enterMeal
Operation: enterMeal(calories: integer, fat: integer, protein: integer, carbs: integer)
Pre-conditions: User has eaten a meal
Post-conditions:

- A Meal instance uMeal was created (instance creation)
- uMeal was associated with the User. (association formed)
- uMeal.calories was set to calories. (attribute modification)
- uMeal.fat was set to fat. (attribute modification)
- uMeal.protein was set to protein. (attribute modification)

- uMeal.carbs was set to carbs. (attribute modification)

3. Fully-dressed

ID UC User Logs Injury

Scope User Injury System

Stakeholders and interests

User
- Person interested in recording their injury
- Person interested in having their workouts tailored to their injury

Trainer
- Person interested in planning workouts according to their client's ability

System admin
- Person responsible for user injury system run

Precondition: User is injured

Postcondition: Injury is recorded

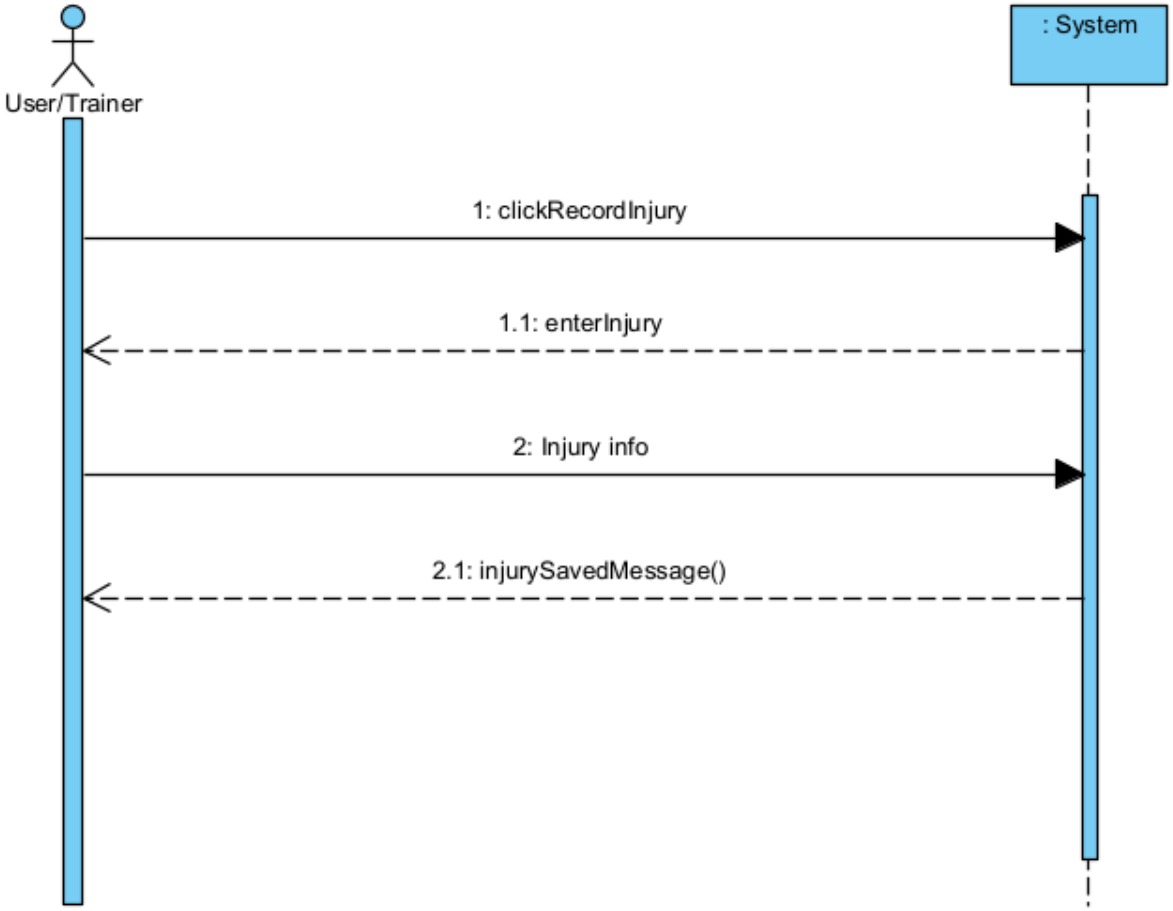
Main success scenario:
1. User is injured
2. User logs into app
3. User clicks "Record Injury"
4. User enters injury location, type, pain level, activities to avoid, and the origin.
5. System saves the user's injury
6. Trainer can view the user's injury(s)

Alternate paths:

a.* anytime the app does not respond
1. User restarts application

1.a User forgot their password
1. User resets password

4.a User does not enter all of the injury info
1. System saves only the entered info



Contract CO2: User Logs Injury
Operation: enterInury(injury : string, location : string, cause : string, painLevel : integer)
Pre-conditions: user is injured
Post-conditions:

- An injury instance uInjury was created (instance creation)
- uInury was associated with user (association formed)
- uInjury.location was set to location (attribute modification)
- uInjury.injury was set to injury (attribute modification)
- uInjury.painLevel was set to painLevel (attribute modification)

- **Savannah Johnson:**

1. Fully Dressed: Logging weight

ID UC Tracking weight
Scope Health & Fitness System
Level User goal
Stakeholders and Interests
General User (User) - A user wants to track and log their weight over time
Trainer - Creates and manages weight logs and fitness plans, monitors progress
Admin - Ensures content follows app rules, policies, and maintains privacy
System Manager/Maintainer - Ensures the application runs smoothly, stores and updates data efficiently.
Precondition: User profile is created and the user is logged in
Postcondition: Weight progress is logged and saved. The user's weight history log is updated with the newly created weight entry data. The user's weight goals are updated accordingly.
Main Success Scenario:
1. User wants to log their current weight
2. User clicks "Record Weight"
3. System prompts the user to input their weight and specified unit of measurement
4. User enters date for the weight input
5. User enters weight
6. User clicks "Save Weight"
7. System validates fields, ensuring correct units and valid weight and date ranges
8. System stores the new weight entry in the user's history
9. User views their updated progress and weight history

Alternate Paths:
a.* System does not respond
    1. An error message is displayed to the user asking them to restart the entry process
3a. A new form can't be created or loaded
    1. System shows an error message and asks the user to refresh the page or try again later.
7a. The user entered an invalid unit measurement
    1. System shows an error message and asks user to change the unit
7b. The user entered an invalid range for weight
    1. System shows an error message and asks user to enter a value greater than 0
7c. The user entered an invalid date
    1. System shows an error message and asks user to enter a correct date where months are between 0 and 12, days are between 1 and 31, and year is 4 digits.
7d. The user left a required field blank
    1. System shows an error message and asks user to enter a value in the fields
8a. System is unable to save entry
    1. System shows an error message and asks user to try again
9a. System is unable to update the user's weight
    1. System sends error report to system manager

2. System warns the user that their progress may not have saved

2. System shows error message to user, informs them that goal progress might not be accurate until a system manager can correct the issue

Operation Contracts:

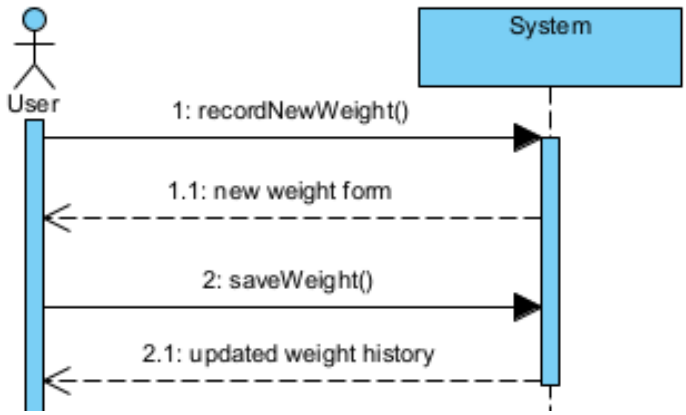
recordNewWeight()

Operation: recordNewWeight()
Cross References: None
Precondition: User profile has been created. The user is logged in.
Postcondition: A new weight entry form is created and displayed for the user to fill out

saveWeight()

Operation: saveSleep(Date: date, Weight: double, Unit: string)
Cross References: None
Precondition: User has filled out the form with correct values for unit, date, and weight.
Postcondition:

- A weightRecord is created for the user
- User's weight goals have their progress updated
- The weight entry is stored in user's weight history



2. Fully Dressed: Set weight goal

ID UC Setting Weight Goal
Scope Health & Fitness System
Level User Goal
Stakeholders and Interests
General User (User) - Wants to track and set weight goals over time
Trainer - Creates and manages exercise plans
Admin - Ensures content follows app rules and policies
System Manager/Maintainer - Person responsible for ensuring the application runs smoothly
Precondition: User profile has successfully been created.
Postcondition: New weight goal is saved in the system. This goal's progress is updated. The user can view their weight goal and monitor progress of the weight goal.

Main Success Scenario:
1. User wants to create or view a weight goal
2. User clicks “Weight Goals”
3. System displays the user’s current weight goal(s)
4. User clicks “Create Weight Goal”
5. System displays weight goal form
6. User inputs a weight and unit of measurement for the goal
7. User enters a target date for the goal (optional)
8. User clicks “Save New Goal”
9. System validates weight range, valid date, and valid unit of measurement
10. If fields are correct, the system stores the weight goal and updates the user’s weight goal history.
11. The user can view weight goal history and progress toward the current weight goal.

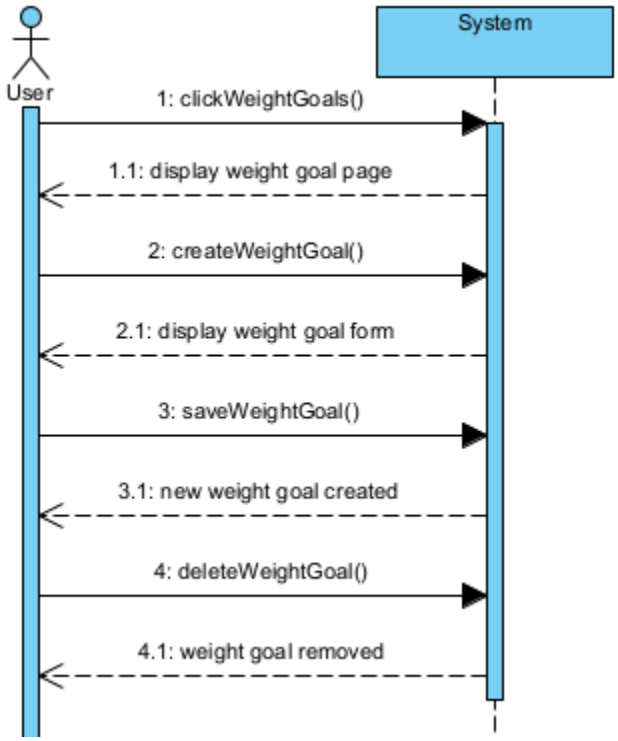
Alternate Paths:
2a. User’s sleep goals page can’t be loaded or displayed properly
    1. An error message is displayed to the user notifying them that there has been an error and to refresh the page or to try again later.
    2. System sends error message to the system manager
6a. The user entered an invalid unit
    1. System displays an error message and asks user to enter a valid unit
6b. The user entered an invalid weight range
    1. System shows an error message and asks user to enter a positive weight goal
6c. The user entered an invalid date
    1. System shows an error message and asks user to enter a correct date that is in the future, where the values are within correct range: months are between 0 and 12, days are between 1 and 31, and year is 4 digits
10a. The system is unable to store the weight goal
    1. System sends error report to system manager
    2. System warns the user that their progress may not have saved

Operation Contracts:
<u>clickWeightGoals:</u>
Operation: clickWeightGoals()
Cross References: None
Precondition: User account is created and the user is logged in.
Postcondition: The user is able to view the weight goals page

<u>createWeightGoal:</u>
Operation: createWeightGoal()
Cross References: None
Precondition: User is currently viewing the weight goals page
Postcondition: The user is able to view the weight goal form.

saveWeightGoal:
Operation: saveWeightGoal(TargetWeight: double, Unit: string, TargetDate: date)
Cross References: None
Precondition: User has entered correct values for target weight, unit of measurement, and target date (optional).
Postcondition: A new weight goal is created for the user and updated. The user can view the weight goal and history.

deleteWeightGoal:
Operation: deleteWeightGoal(ID: int)
Cross References: None
Precondition: The specified goal already exists
Postcondition: The specified weight goal is removed from the user's history, and the user's weight goals are updated.



3. Fully Dressed: Tracking hydration
ID UC Logging water intake
Scope Health & Fitness System
Level User goal
Stakeholders and Interests
General User (User) - A user wants to track and log their water intake over time
Trainer - Creates and manages hydration logs, monitors progress

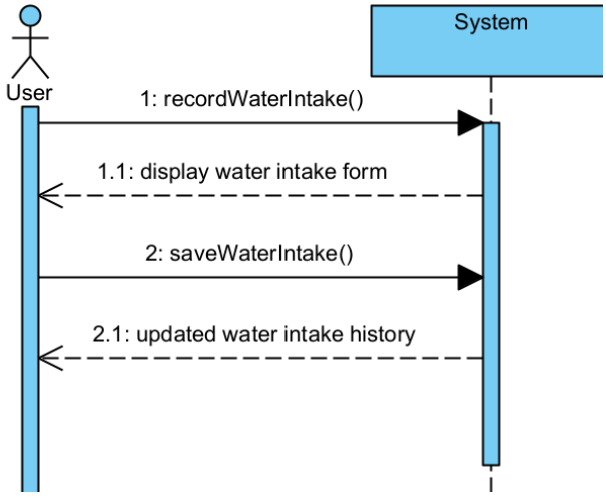
Admin - Ensures content follows app rules, policies, and maintains privacy
System Manager/Maintainer - Ensures the application runs smoothly, stores and updates data efficiently.
Precondition: User profile is created and the user is logged in
Postcondition: Hydration progress is logged and saved. The user's hydration log is updated with the newly created entry data. The user's hydration goals are updated accordingly.
Main Success Scenario:
1. User wants to log water intake
2. User clicks "Record Water"
3. System prompts the user to input the amount of water and units (oz, ml, etc.)
4. User enters date for the water input
5. User clicks "Save"
6. System validates fields, ensuring valid units, quantity, and date range
7. System stores and updates the new water intake entry in the user's hydration log
9. User views their updated hydration progress and history
Alternate Paths:
a.* System does not respond
    1. An error message is displayed to the user asking them to restart the entry process
3a. A new form can't be created or loaded
    1. System shows an error message and asks the user to refresh the page or try again later.
6a. The user entered an invalid unit of measurement
    1. System shows an error message and asks user to change the unit
6b. The user entered an invalid range for water intake
    1. System shows an error message and asks user to enter a value greater than 0
6c. The user entered an invalid date
    1. System shows an error message and asks user to enter a correct date where months are between 0 and 12, days are between 1 and 31, and year is 4 digits.
6d. The user left a required field blank
    1. System shows an error message and asks user to enter a value in the fields
7a. System is unable to save entry
    1. System shows an error message and asks user to try again
7a. System is unable to update the user's hydration
    1. System sends error report to system manager
    2. System warns the user that their progress may not have saved
    2. System shows error message to user, informs them that goal progress might not be accurate until a system manager can correct the issue

Operation Contracts:
recordWaterIntake()
Operation: recordWaterIntake()
Cross References: None
Precondition: User profile has been created. The user is logged in.

Postcondition: A new water intake entry form is created and displayed for the user to fill out

saveWaterIntake()
Operation: saveWaterIntake(Date: date, Weight: double, Unit: string)
Cross References: None
Precondition: User has filled out the form with correct values for unit, date, and water intake.
Postcondition:

- A waterRecord is created for the user
- User's hydration goals have their progress updated
- The water intake entry is stored in user's water intake history



Mason Baxter:

1.Fully Dressed: Tracking Sleep

ID UC Tracking Sleep

Scope Health & Fitness System

Level User Goal

Stakeholders and Interests

General User (User) - Person who is interested in tracking their sleep

Trainer - Person who creates and manages exercise plans/classes

Admin - Person who is responsible for ensuring content follows app rules and policies.

System Manager/Maintainer - Person responsible for ensuring the application runs smoothly

Precondition: User profile has been created.

Postcondition: Sleep goal progress has been updated and saved. Entry data for sleep is created and saved in the user's sleep history.

Main Success Scenario:

1. User wants to log an amount of sleep they had.
2. User clicks "Record Sleep"
3. System prompts user to input the amount of sleep and a rating of the quality of sleep on a scale from one to ten
4. User enters the date(s) when the sleep occurred
5. User enters the amount of sleep
6. (optional) User enters the quality of the sleep from one to ten
7. User clicks "Save Sleep"
8. System validates all fields
9. System creates a new sleep entry in the user's history
10. System updates any of the user's sleep goals with this new information

Alternate Paths:

a.* Anytime system does not respond

    1. An error message is displayed to the user asking them to restart the entry process

3a. A new form can't be created or loaded

    1. System shows an error message, asks the user to refresh the page or try again later.

8a. The user entered a value that can't be validated

    1. System shows an error message, asks the user to change the entry amount or try again

8b. The user entered a value that is out of the given range for quality of sleep

    1. System shows an error message, asks the user to re-enter a value between one and ten

8c. The user left a required field blank

    1. System shows an error message, asks the user to enter a value in the required fields

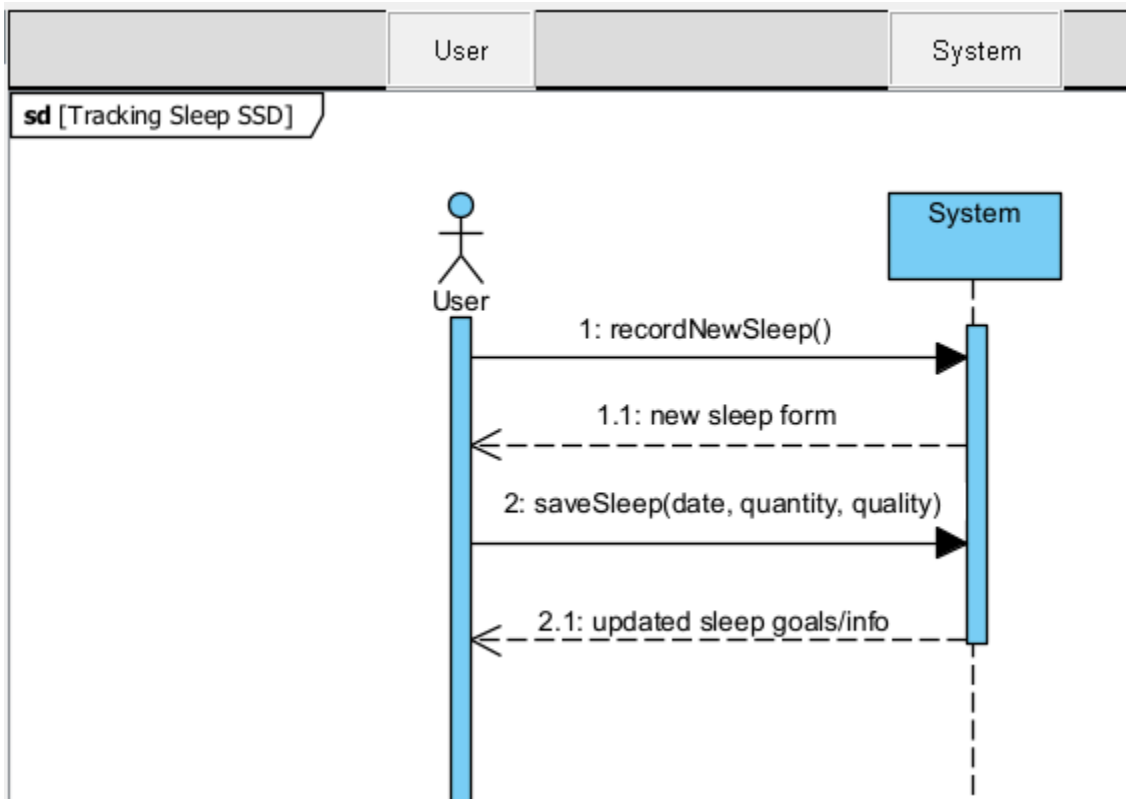
9a. System is unable to save or create a new entry

    1. System shows an error message, asks user to try again

10a. System is unable to update the user's sleep goals

    1. System sends error message to system manager
    2. System shows error message to user, informs them that goal progress might not be accurate until a system manager can correct the issue

Screenshot of SSD:



Operation Contracts:
recordNewSleep()
Operation: recordNewSleep()
Cross References: None
Precondition: User profile has been created. The user is on their dashboard.
Postcondition: A new sleep form is created and displayed for the user to fill out

saveSleep()
Operation: saveSleep(Date: date, Quantity: integer, Quality: integer)
Cross References: None
Precondition: User has filled out the form with correct values for date, and quantity.
Postcondition:

- A sleepRecord has been created for the user
- User's sleep goals have their progress updated if this sleepRecord's date, quality, or quantity is within the range

2. Fully Dressed: Setting Sleep Goals

ID UC Setting Sleep Goals

Scope Health & Fitness System

Level User Goal

Stakeholders and Interests

General User (User) - Person who is interested in tracking and setting goals for their sleep.

Trainer - Person who creates and manages exercise plans/classes

Admin - Person who is responsible for ensuring content follows app rules and policies

System Manager/Maintainer - Person responsible for ensuring the application runs smoothly

Precondition: User profile has successfully been created.

Postcondition: New sleep goal is saved in the system. This goal's progress is updated based upon the user's previous sleep entries that fit within the goal's ranges.

Main Success Scenario:

1. User wants to create a new sleep goal
2. User clicks "Sleep Goals"
3. System displays the user's created sleep goals
4. User clicks "Create New Sleep Goal"
5. System shows sleep goal form
6. User inputs a name for the goal
7. User selects if the goal is a daily, weekly, or monthly goal.
8. (optional) User selects a start date for the goal.
9. User inputs the amount of sleep they want to reach in the given timeframe
10. (optional) User inputs the minimum quality of sleep that will count towards this goal
11. User clicks "Confirm New Goal"
12. System validates all the fields
13. System creates a new sleep goal for the user with the given information and constraints
14. System updates the newly created sleep goal with the user's previous sleep records

Alternate Paths:

a.* Anytime system does not respond

    1. An error message is displayed to the user asking them to restart the goal creation process

3a. User's sleep goals page can't be loaded or displayed properly

    1. An error message is displayed to the user notifying them that there has been an error and to refresh the page or to try again later.
    2. System sends error message to the system manager

5a. A new form can't be created or loaded

    1. System shows an error message, asks the user to refresh the page or try again later

8a. The user did not set a start date for the goal

    1. System sets the start date for the goal to the default start date (the date the user created the goal).

10a. The user does not set a minimum quality of sleep that will count

    1. System sets the minimum quality of sleep for the goal to the default value of one.

12a. The user did not complete all required fields

    1. System shows an error message, asks the user to enter a value in the required fields

12b. The user entered a value that can't be validated

1. System shows an error message, asks the user to either change the entry or try again

12c. The user entered a minimum quality of sleep value outside the given range for quality of sleep

1. System shows an error message, asks the user to re-enter a value between one and ten

12d. Policy Keywords Found in Name

1. System shows an error message, asks user to input a different name

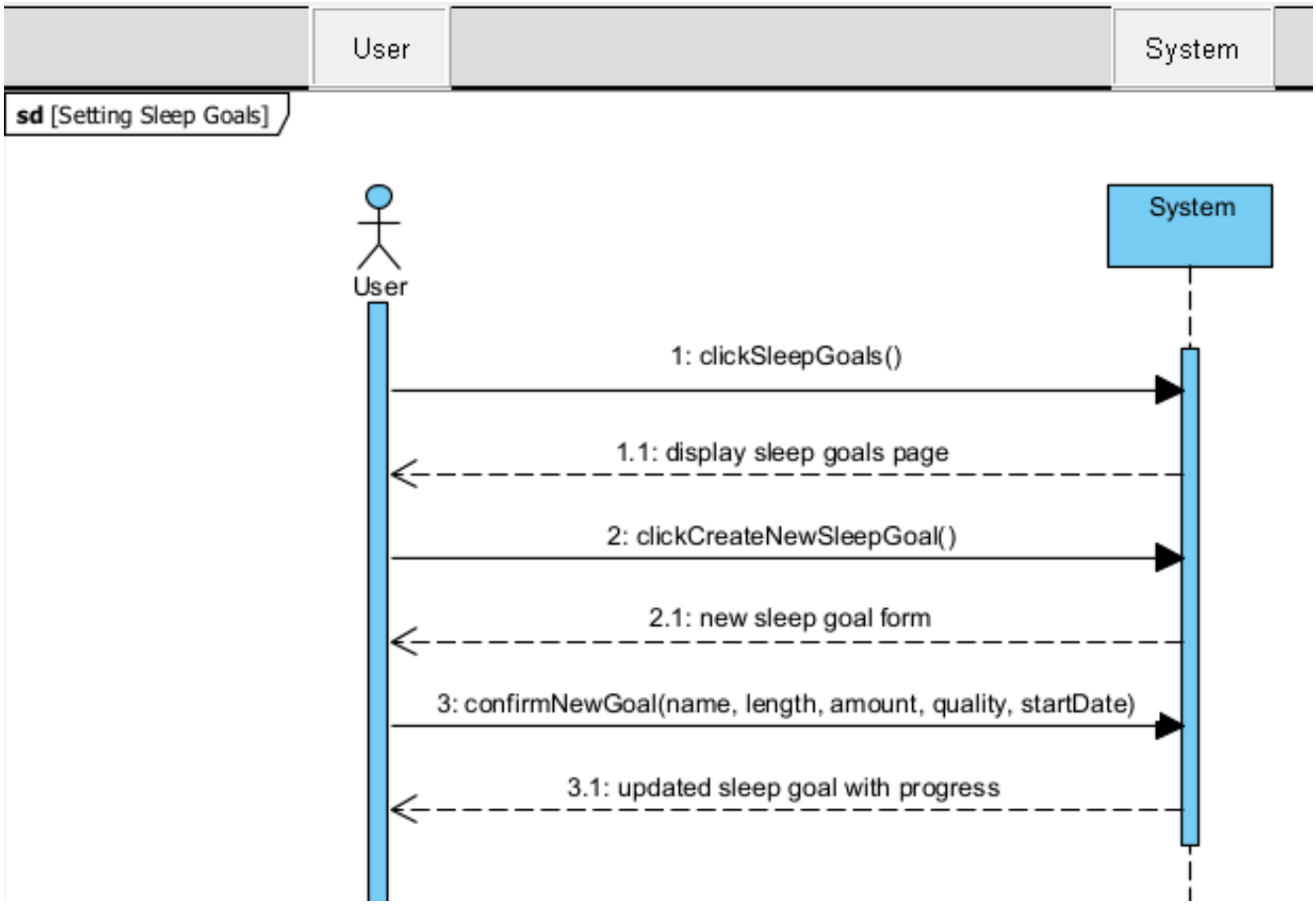
13a. System is unable to save or create a new entry

1. System shows an error message, asks user to try again

14a. System is unable to update the user's sleep goal

1. System sends error message to system manager
2. System shows an error message to the user, informing them that goal progress might not be accurate until a system manager can correct the issue.

Screenshot of SSD:



Operation Contracts:

clickSleepGoals:

Operation: clickSleepGoals()
Cross References: None
Precondition: User profile has already been created. The user is on their dashboard.
Postcondition: None

clickCreateNewSleepGoal:

Operation: clickCreateNewSleepGoal()
Cross References: None
Precondition: User is on the sleep goals page
Postcondition: A new sleep goal form is created for the user to fill out

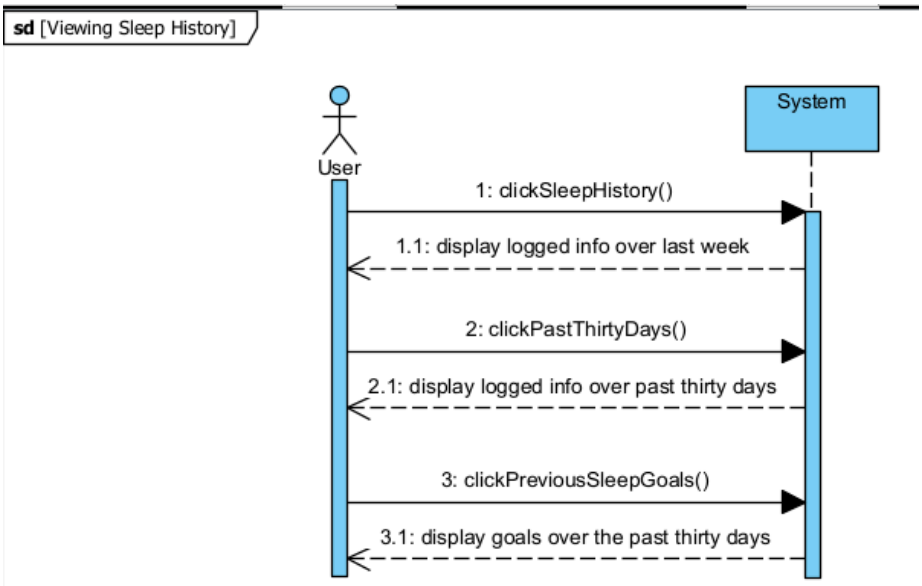
confirmNewGoal:

Operation: confirmNewGoal(name: string, length: integer, amount: integer, quality: integer, startDate: date)
Cross References: None
Precondition: User has filled out a sleep goals form with correct values for name, length, and amount
Postcondition: A sleepGoal has been created for the user and updated according to the user's sleepRecords

3. Fully Dressed: Viewing Sleep History

ID UC Viewing Sleep History and Goals Over the Past Month
Scope Health & Fitness System
Level User Goal
Stakeholders and Interests
General User (User) - Person who is interested in tracking and setting goals for their sleep
Trainer - Person who creates and manages exercise plans/classes
Admin - Person who is responsible for ensuring content follows app rules and policies
System Manager/Maintainer - Person responsible for ensuring the application runs smoothly
Precondition: User profile has successfully been created.
Postcondition: None
Main Success Scenario:
1. User wants to view their logged sleep history
2. User clicks “Sleep History”
3. System displays the user’s previously logged sleep information over the past week
4. User clicks “Past 30 Days”
5. System displays the user’s previously logged sleep information over the past 30 days
6. User clicks “Previous Sleep Goals”
7. System displays the user’s sleep goals that had occurred during the past 30 days
Alternate Paths:
a.* Anytime system does not respond
    1. An error message is displayed to the user asking them to refresh the page.
3a. The user's sleep history can’t be loaded or displayed properly for the past week.
    1. An error message is displayed to the user notifying them that there has been an error and to refresh the page or to try again later.
5a. The user’s sleep history can’t be loaded or displayed properly for the previous thirty days
    1. An error message is displayed to the user notifying them that there has been an error and to refresh the page or to try again later.
7a. The user’s sleep goals for the past thirty days can’t be loaded or displayed
    1. An error message is displayed to the user notifying them that there has been an error and to refresh the page or to try again later.

Screenshot of SSD:



Operation Contracts:

clickSleepHistory:

Operation: clickSleepHistory()
Cross References: None
Precondition: User profile has been created. The user is on their dashboard.
Postcondition: None

clickPastThirtyDays:

Operation: clickPastThirtyDays()
Cross References: None
Precondition: User is on their sleep history page.
Postcondition: viewSleepTimeline is set to thirty days.

clickPreviousSleepGoals:

Operation: clickPreviousSleepGoals()
Cross References: None
Precondition: User is on their sleep history page.
Post Condition: None

Ethan Rogers:

1. Fully Dressed: Sending a Friend Request

ID UC Send Friend Request

Scope Social

Level Friends

Stakeholders and Interests

General User (user): someone who wants to connect with other people virtually

System Maintainer: person responsible for making sure that the social menus work

Precondition: User profile has already been created

Postcondition: none

Main Success Scenario:

1. User wants to send friend request
2. User clicks on “Social” menu
3. User navigates to “Friends” section
4. User clicks on “Add Friend” button
5. User submits the username of the person they want to friend
6. System notifies the other user of the friend request

Alternate Paths:

a. System does not respond

1. An error message is displayed to the user asking them to refresh the app

3a. The user’s friends list could not be loaded

1. An error message is displayed that there was an error loading their friends and to reload the app to try again

5a. The submitted username is invalid

1. An error message is displayed saying that the specified user doesn’t exist and to try another name

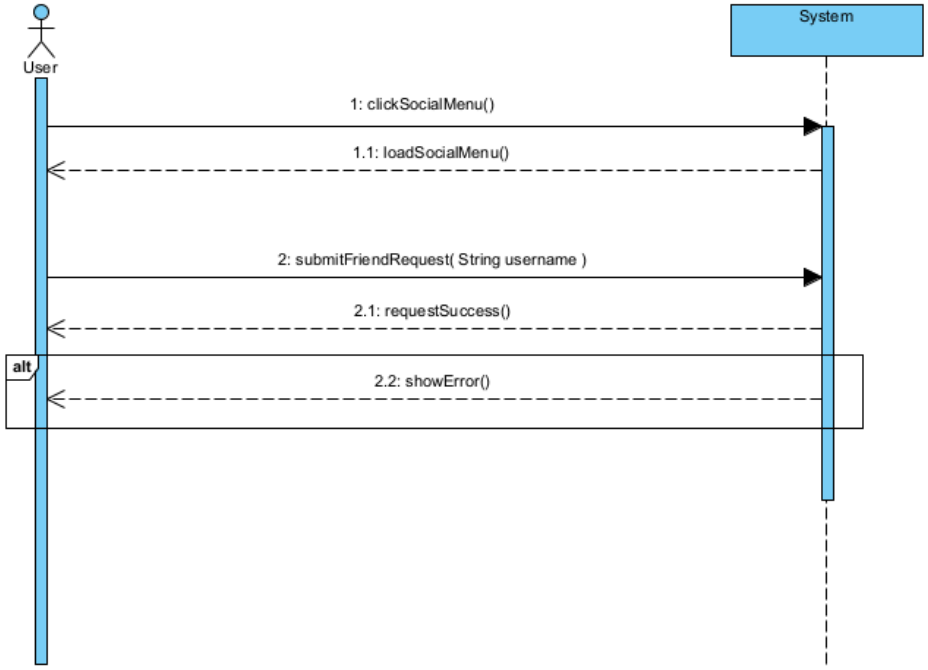
6a. The system does not receive the request

1. An error message is displayed telling the user to try again

6b. The system does not notify the other user

1. The other user will not be aware of the friend request until they open the app

SSD:



Operation Contracts:

clickSocialMenu:

Operation: clickSocialMenu()
CrossReferences: None
Precondition: The profile has been created and the user is on the dashboard
Postcondition: none

clickAddFriend:

Operation: clickAddFriend()
CrossReferences: none
Precondition: user is already in the social menu
Postcondition: none

submitFriendRequest:

Operation: submitFriendRequest(String username)
CrossReferences: none
Precondition: user has clicked add a friend button
Postcondition: the system will notify the other user of the friend request

2. Fully Dressed: Accepting a Friend Request

ID UC Accept Friend Request

Scope Social

Level Friends

Stakeholders and Interests

General User (user): someone who wants to connect with other people virtually

System Maintainer: person responsible for making sure that the social menus work

Precondition: User profile has already been created. User has received a friend request.

Postcondition: none

Main Success Scenario:

1. User wants to accept friend request
2. User clicks on “Social” menu
3. User navigates to “Friends” section
4. User clicks the “Incoming Friend Request” button
5. User accepts the request of the person they want to be friends with
6. System records the friendship between the two users. System notifies the other user that the friend request has been accepted.

Alternate Paths:

a. System does not respond

    1. An error message is displayed to the user asking them to refresh the app

3a. The user’s friends list could not be loaded

    1. An error message is displayed that there was an error loading their friends and to reload the app to try again

4a. The user’s incoming friend request failed to load

    1. An error message is displayed saying that their friend requests could not be displayed and to try restarting the app.

5a. The user rejects the friend request

    1. The system deletes the request

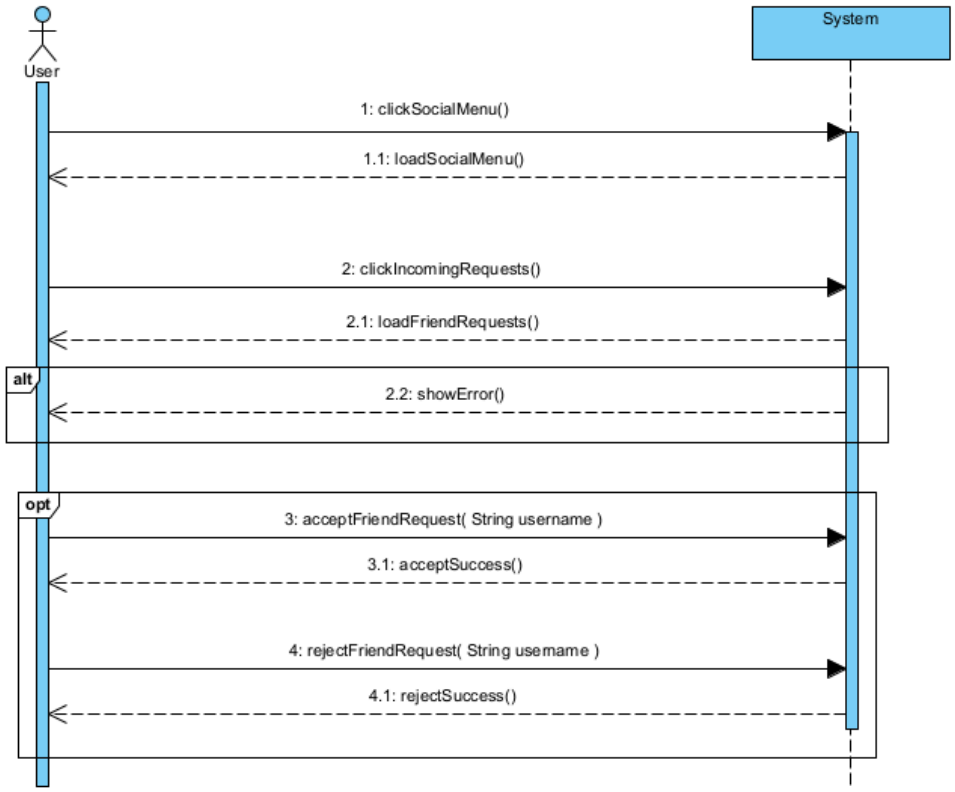
6a. The system does not receive the accept request

    1. An error message is displayed telling the user to try again

6b. The system does not notify the other user

    1. The other user will not be aware of the accepted friend request until they open the app

SSD:



Operation Contracts:

clickSocialMenu:

Operation: clickSocialMenu()
CrossReferences: None
Precondition: The profile has been created and the user is on the dashboard
Postcondition: none

clickIncomingRequests:

Operation: clickIncomingRequests()
CrossReferences: none
Precondition: user is already in the social menu, the user must have at least 1 incoming friend request
Postcondition: the app must load all of the incoming friend requests

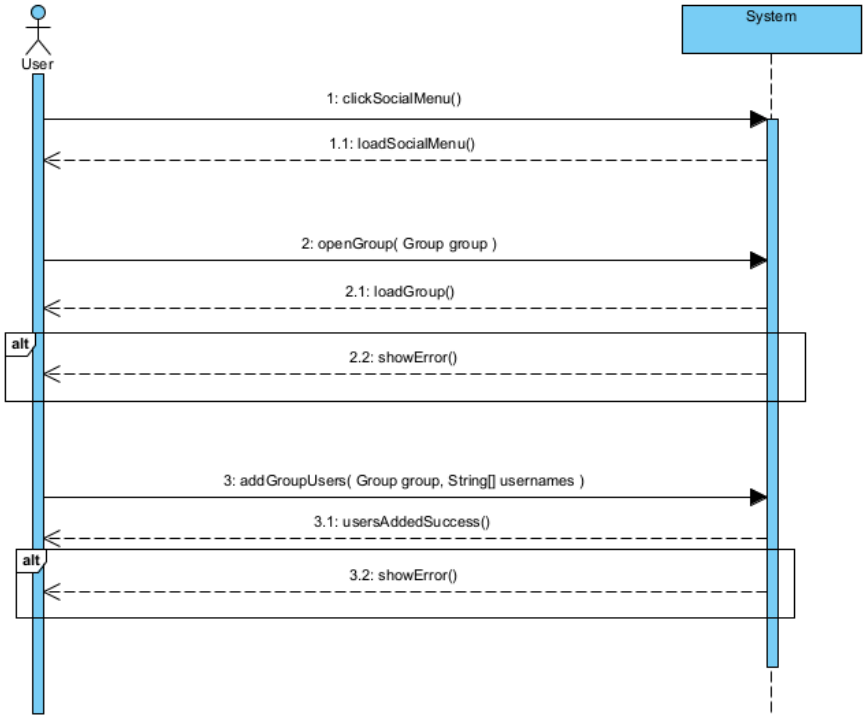
acceptFriendRequest:

Operation: submitFriendRequest(String username)
CrossReferences: none
Precondition: user has clicked show incoming friend requests button
Postcondition: the system will record the friendship between the two users

rejectFriendRequest:
Operation: rejectFriendRequest(String username)
CrossReferences: none
Precondition: user has clicked show incoming friend requests button
Postcondition: none

3. Fully Dressed: Adding Users to Group
ID UC Add Group Users
Scope Social
Level Groups
Stakeholders and Interests
General User (user): someone who wants to connect with other people virtually
System Maintainer: person responsible for making sure that the social menus work
Precondition: User profile has already been created. User has received a friend request.
Postcondition: none
Main Success Scenario:
1. User wants to accept friend request
2. User clicks on “Social” menu
3. User navigates to “Groups” section
4. User clicks on the Group they want to add users to
5. User clicks on “Add members”
6. User selects users from their friends list to add to the group and clicks submit
7. System adds those users to the group in the data base
Alternate Paths:
a. System does not respond
    1. An error message is displayed to the user asking them to refresh the app
3a. The user’s groups list could not be loaded
    1. An error message is displayed that there was an error loading their groups and to reload the app to try again
5a. The user rejects the friend request
    1. The system deletes the request
7a. The system does not receive the request
    1. An error message is displayed telling the user to try again

SSD:



Operation Contracts:

clickSocialMenu:
Operation: clickSocialMenu()
CrossReferences: None
Precondition: The profile has been created and the user is on the dashboard
Postcondition: none

openGroup:
Operation: openGroup(Group group)
CrossReferences: none
Precondition: user is already in the social menu
Postcondition: none

clickAddUsers:
Operation: clickAddUsers()
CrossReferences: None
Precondition: the user is in the group menu
Postcondition: none

addGroupUsers:
Operation: addGroupUsers(Group group, String[] usernames)
CrossReferences: none

Precondition: user is already in the group menu and has input as many of their friends that are not in group as they want
Postcondition: it will be recorded that each of the users in usernames will be added to the group

Dannis Wu:
1. Full-dressed: Create a Self-Paced Exercise Plan

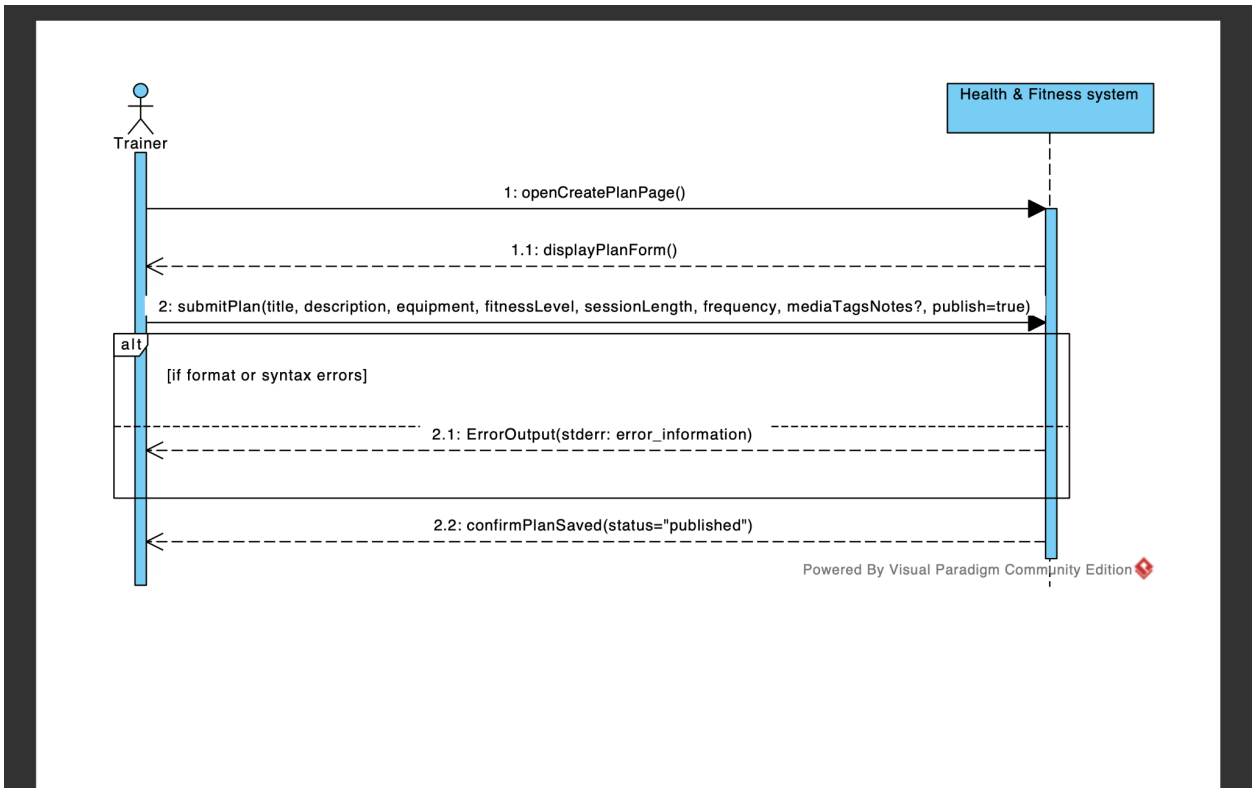
ID UC Create a Self-paced Plan
Scope Health & Fitness system
Level user goal
Stakeholders and interests
Customer (General user) — person that uses plans to exercise safely.
Trainer — person who creates and manages exercise plans/classes.
Admin — person checking content follows policy.
System maintainer — person responsible for application run.

Precondition: Trainer is authenticated.
Postcondition: Plan is saved (draft or published).

Main success scenario:

1. trainer wants to create a self-paced plan.
2. trainer opens **Create plan** page.
3. system shows plan form.
4. trainer fills basic info: title and description.
5. trainer selects required equipment.
6. trainer selects recommended fitness level.
7. trainer enters average session length.
8. trainer enters suggested frequency (e.g., 3 times per week).
9. (optional) trainer adds media/tags/safety notes.
10. trainer chooses **Publish**.
11. system validates all fields.
12. system saves plan and confirms success.

Alternate paths:
a.* anytime system does not respond — system autosaves draft; trainer will try again.
3.a form cannot load — system shows error; trainer will refresh or come back later.
11.a some required field is empty or invalid — system highlights fields; trainer fixes and goes to step 10.
10.a trainer chooses **Save as draft** — system saves draft; plan is not visible to users.
11.b duplicate title detected — system suggests rename; trainer picks a new title and goes to step 10.
11.c policy keywords found — system marks **Pending review**; trainer is notified.
12.a network failure on save — system keeps local draft and shows warning.



Contract CO1: createSelfPacedPlan

Operation:
createSelfPacedPlan

Pre-condition:
Trainer is logged in and has opened the create plan page.

Post-conditions:

- A new Plan was created.

- The Plan was linked to the Trainer.
- The Plan title was set.
- The Plan description was set.
- The Plan equipment was set.
- The Plan fitness level was set.
- The Plan session length was set.
- The Plan frequency was set.
- The Plan notes or media were set if provided.
- The Plan status was set to Published if valid, or Draft if saved as draft.
- If there are errors, the system shows a message and does not save.

2. Fully-dressed: Create and Register a Scheduled Class

ID UC Create and Register a Scheduled Class
Scope Health & Fitness system
Level user goal
Stakeholders and interests
Customer (General user) — person that wants clear time, level, prerequisites, and seats.
Trainer — person who schedules, hosts, and manages the class.
Admin — person ensuring limits and policies are respected.
System maintainer — person responsible for application run.

Precondition: Trainer is authenticated.
Postcondition: Class is saved (draft or published) with schedule and capacity.

Main success scenario:

1. trainer wants to create a scheduled class.
2. trainer opens **Create class** page.
3. system shows class form.
4. (optional) trainer links an existing plan as curriculum.
5. trainer fills title and description.
6. trainer sets date/time, days of week, session length, and number of weeks.

7. trainer selects delivery mode (in-person location or online link).

8. trainer sets enrollment limit (max participants).

9. trainer sets recommended fitness level and prerequisites (recommended or required).

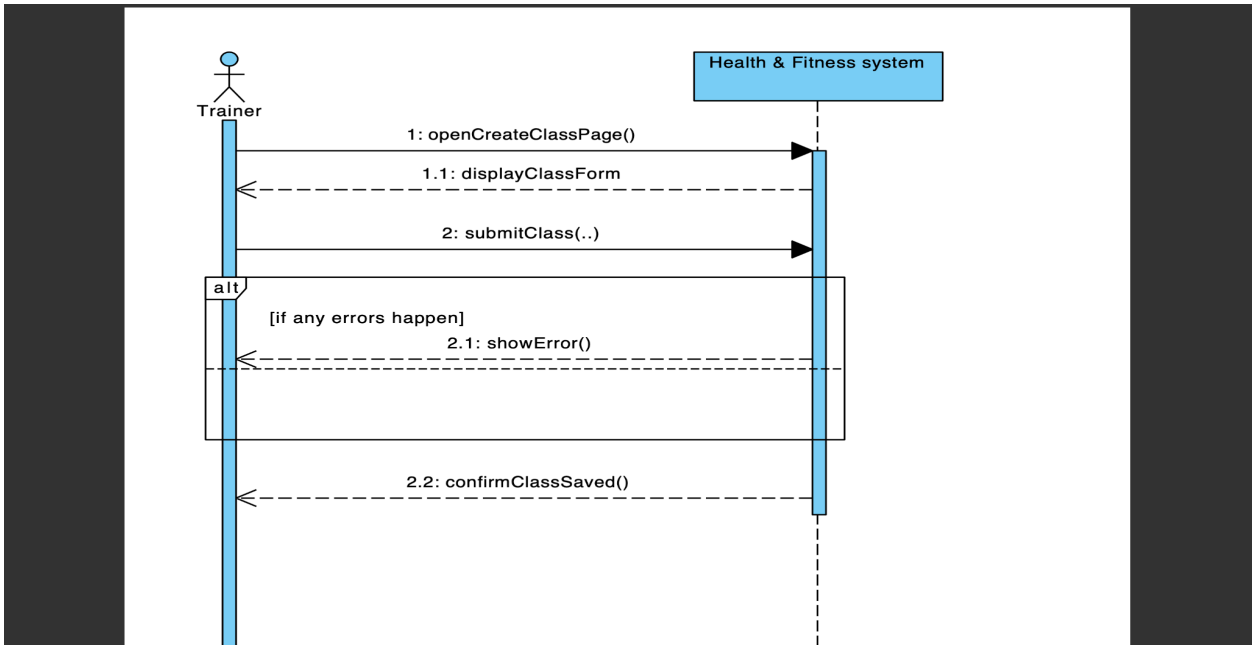
10. trainer lists required equipment.

11. trainer chooses **Publish**.

12. system validates fields and checks for schedule conflict with trainer's other classes.

13. system saves class (and recurrence), opens registration, and confirms success.

Alternate paths:
a.* anytime system does not respond — system autosaves draft; trainer will try again.
7.a online link cannot be generated — system asks trainer to paste a link; continue to step 11.
8.a enrollment limit missing — system applies default; trainer may edit before step 11.
9.a prerequisites marked "required" but empty — system blocks publish; trainer fills and goes to step 11.
12.a conflict with another class — system shows conflict list; trainer edits schedule and returns to step 11.
11.a trainer chooses **Save as draft** — class saved as draft; not visible for registration.
12.b policy issue detected — class set to **Pending review**; trainer notified.
13.a storage/network error — keep local draft and show warning.



Contract CO2: submitClass

Operation:
submitClass

Pre-condition:
Trainer is logged in and has opened the Create Class page.

Post-conditions:

- A new Class was created.
- The Class was linked to the Trainer.
- The Class title and description were set.
- The schedule (date, time, days, weeks) was set.
- The delivery mode (in-person or online) was set.
- The enrollment limit was set.
- The fitness level and prerequisites were set.
- The required equipment was stored.
- If publish was chosen and validation passed, Class status was Published.
- If trainer chose Save as Draft, Class status was Draft.
- If errors happen, the system shows an error and does not save.

3. Full-dressed: Host a Scheduled Class Session

ID UC Host a Scheduled Class Session
Scope Health & Fitness system
Level user goal

Stakeholders and interests
Customer (General user) — person attending a class, wants fair admission and accurate attendance tracking.
Trainer — person hosting the class, wants to run session smoothly, admit participants, and track attendance.
Admin — person monitoring policy compliance and participant disputes.
System maintainer — person responsible for system uptime and correct roster/attendance functions.

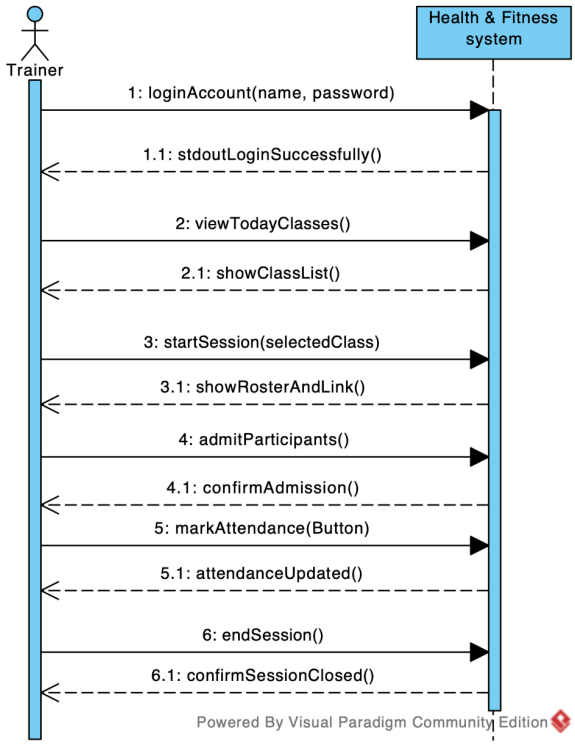
Precondition: Trainer is authenticated and has an active scheduled class today.
Postcondition: Session is closed; attendance, duration, and notes are stored; statistics are updated.

Main success scenario:

1. trainer logs in to the system.
2. trainer opens Today's Classes page.

3. system displays the trainer's scheduled classes for the day.
4. trainer selects the desired class.
5. trainer presses Start Session.
6. system opens the roster, attendance controls, and if online, shows the meeting link.
7. participants join the class.
8. trainer checks registration and prerequisites for each participant.
9. system admits eligible participants and shows confirmation.
10. trainer marks attendance as participants join.
11. late participants may join if seats remain; trainer admits them.
12. trainer runs the class session.
13. when the workout ends, trainer presses End Session.
14. system saves attendance, duration, and optional notes.
15. system updates class statistics and closes the session.

Alternate paths:
a.* anytime system does not respond — trainer may retry; if failure persists, session may be canceled.
6.a meeting link fails to generate — trainer pastes external link manually; continue to step 7.
9.a participant not registered or missing required prerequisites — system denies admission and shows message.
11.a class is full — late participant placed on waitlist for this session.
12.a no participants join within grace period — trainer cancels session; system notifies registered users.
13.a trainer cancels mid-session due to technical issues — system notifies users and logs partial session.
14.a storage/network failure — attendance and notes kept locally until sync succeeds.



Contract CO3: hostScheduledClassSession

Operation:
hostScheduledClassSession

Pre-condition:
Trainer is logged in and has selected a scheduled class for today.

Post-conditions:

- The session was opened for that class.
- The roster and, if online, the meeting link were shown.
- Eligible participants were admitted; ineligible were not (or waitlisted).
- Attendance marks were saved.
- The session was ended and marked closed.
- Duration and optional notes were stored.
- Class statistics were updated.

- **Connor Griffin**

  1. Full-Dressed: Log Workout Session

  ID: UC Log a Workout Session

  Scope: Health and Fitness system

  Level: user goal

  Stakeholders and Interests:

  User: wants quick, accurate logging and immediate dashboard updates.

  Admin: needs data integrity and auditability

  System maintain: needs reliable storage and aggregation

  Precondition: User is authenticated and has access to the workout page

  Postcondition: Workout is saved, dashboard(weekly minutes, calories, goal progress) is updated

  Main success scenario:

  1. user opens the Workout page.
  2. system displays the workout form (type, duration, calories, optional notes, default date/time).
  3. user selects workout type (cardio, strength, yoga, or custom).
  4. user enters duration and calories; optionally notes.
  5. user clicks Save Workout.
  6. system validates required fields and numeric ranges.
  7. system stores the workout and updates dashboard aggregates.
  8. system shows confirmation and refreshed dashboard summary.

  Alternate paths:

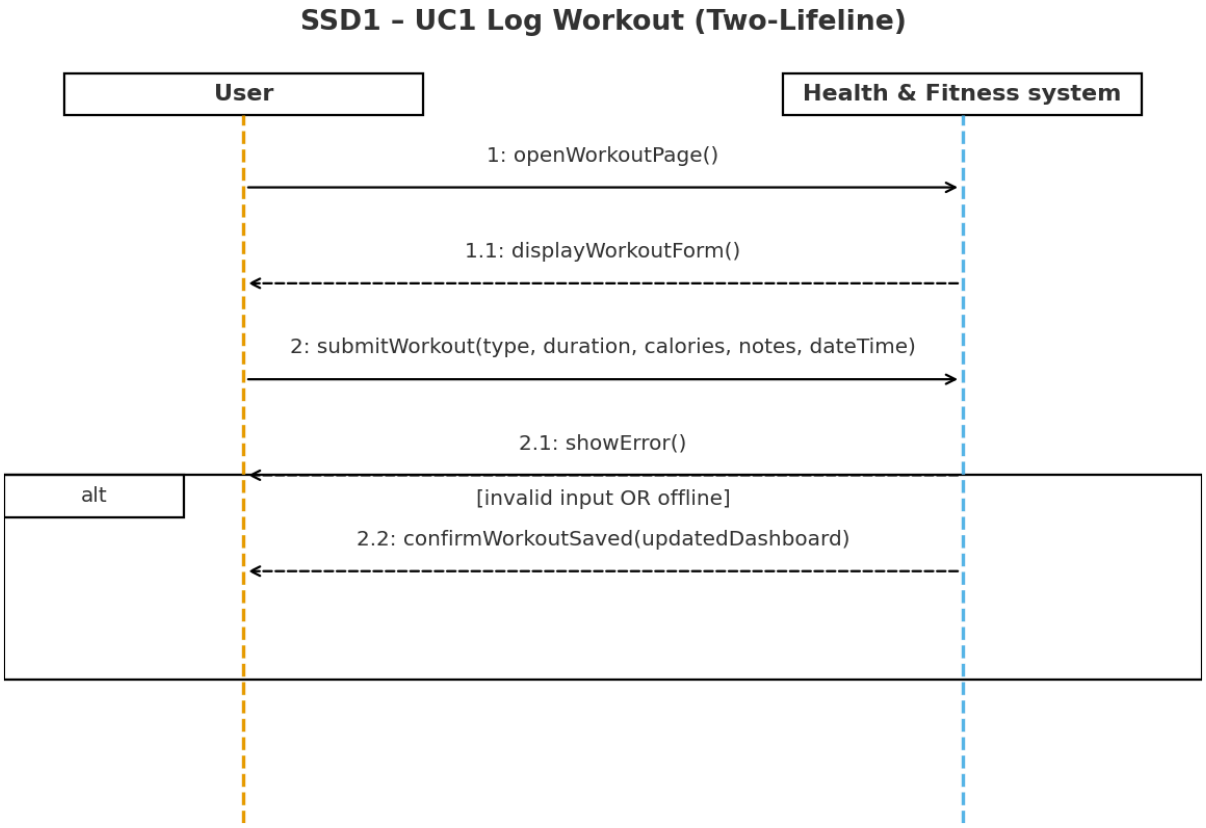
  a.* any time the system is unreachable — show connection error; allow retry or offline queue.

  3.a user leaves workout type empty — system highlights missing field; user fills it and continues.

  6.a invalid values (negative/too large) — system shows validation error; user corrects.

  7.a possible duplicate within short window — system warns; user confirms or cancels

.



Contract OC1: createWorkout
Operation:
createWorkout(userId, type, durationMin, calories, notes, dateTime)
Pre-conditions:
• userId corresponds to an authenticated user.
• type ∈ {cardio, strength, yoga, custom}.
• durationMin ≥ 0, calories ≥ 0. Post-conditions (Success):
• A new Workout is created and persisted.
• Dashboard aggregates for the user are recalculated (or flagged for recalculation).
• Confirmation is returned with workoutId.
Exceptions:
• ValidationError for missing/invalid fields.
• DuplicateWarning if similar workout exists within a brief threshold window (optional).

-

2. Full-Dressed: Join a Training Plan

ID: UC Join Plan

Scope: Health & Fitness system

Level: user goal

Stakeholders and interests:

- User: wants to follow a structured plan that matches goals, equipment, and level.
- Trainer: wants appropriate participants and plan adoption.
- Admin: ensures only approved (Active) plans are joinable.

Precondition: User is logged in; the selected plan exists in the catalog.

Postcondition: Enrollment is created (if plan is Active); plan appears on dashboard; progress can begin.

Main success scenario:

1. user browses the plan catalog.
2. system shows plans with key details (difficulty, equipment, recommended level, average session length).
3. user opens a plan and reviews details.
4. user clicks Join Plan.
5. system verifies authentication and confirms plan status is Active.
6. system creates an enrollment and confirms.
7. system updates the dashboard to include the plan card and next session.

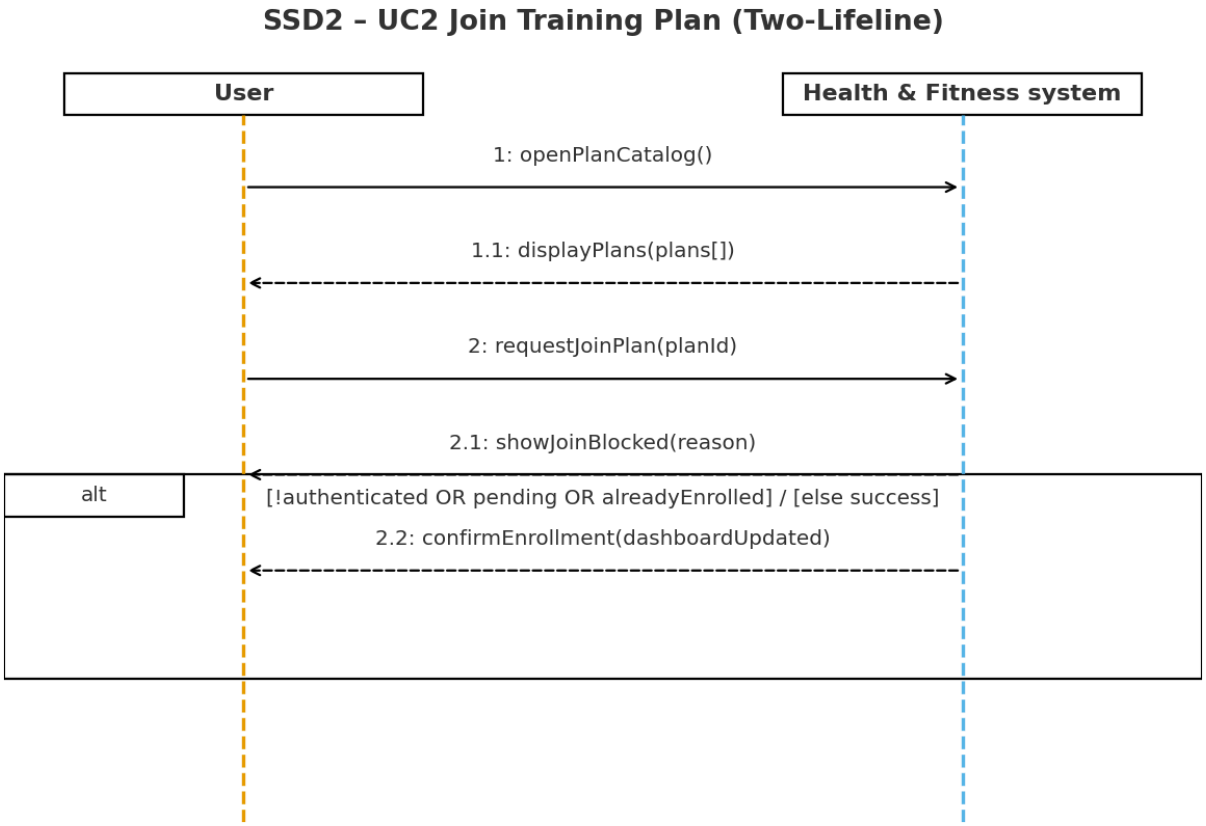
Alternate paths:

a.* user not logged in — system requests login; on success return to step 4.

5.a plan is Pending Review — system shows “not available yet”; user can choose another plan or wait.

6.a user already enrolled and active — system offers “Go to plan” instead of duplicating enrollment.

a.* system/catalog unavailable — system shows error; user may retry later..



Contract OC2: enrollInPlan
Operation:
enrollInPlan(userId,planId)
Pre-conditions:
• userId is authenticated.
• planId exists; plan status is Active.
Post-conditions (Success):
• A new Enrollment is created (isActive=true, progress=0%).
• The user's dashboard is updated to show the plan and next session.
• Confirmation is returned with enrollmentId.
Exceptions:
• NotAuthenticated if the user is not logged in.
• PlanUnavailable if plan status ≠ Active.
• EnrollmentExists if an active enrollment already exists.

3. Full-Dressed: Edit/Delete workout entry

ID: UC Edit/Delete Workout Entry
Scope: Health & Fitness system
Level: User goal
Stakeholders and interests:

User: wants to correct mistakes or remove accidental entries while keeping totals accurate.

Admin:needs auditability for changes/deletions.
System maintainer: needs consistent recalculation of aggregates after edits/deletes.

Precondition: User is authenticated and owns the selected workout entry.
Postcondition: Workout updated or removed; dashboard and weekly aggregates recalculated.
Main success scenario:
1. user opens Workout History.
2. system displays a list of workouts with filters.
3. user selects a workout to open details.
4. user chooses Edit and modifies fields (type, duration, calories, notes).
5. user clicks Save Changes.
6. system validates and saves the update.
7. system recalculates aggregates and shows confirmation.
Alternate paths:
3.a user chooses Delete instead of Edit — system asks to confirm.
3.a.1 user confirms delete — system soft-deletes the workout; recalculates aggregates; shows undo option.
3.a.2 user cancels — no change.
6.a invalid values — system shows field errors; user corrects and resubmits.
a.* server unavailable — system shows error; retry later; changes are not applied until saved.

OC: editOrDeleteWorkoutEntry
Operation: editOrDeleteWorkoutEntry()

Cross References:
UC- edit/delete WorkoutEntry

Preconditions:
-user is authenticated
- the user owns the selected workout entry

Postconditions:
- Exiting workout instances with given workout id is retrieved
- Values are updated
- Conformation message appears

System Sequence Diagram: Edit/Delete Workout Entry

Actors:

- **User** (initiates edit or delete actions)
- **System** (OsoFit application)

Messages (System Operations):

1. **openWorkoutHistory()** — User requests to view past workouts.
2. **selectWorkout(workoutID)** — User selects a specific workout entry.
3. **chooseAction(actionType)** — User chooses whether to *Edit* or *Delete* the entry.
4. **editWorkoutEntry(workoutID, newType, newDuration, newCalories, newNotes)** — If editing, system validates and updates the record.
5. **deleteWorkoutEntry(workoutID)** — If deleting, system performs soft delete and recalculates aggregates.
6. **recalculateDashboard()** — System updates weekly totals and progress charts.
7. **displayConfirmation()** — System returns success or error message to the user.

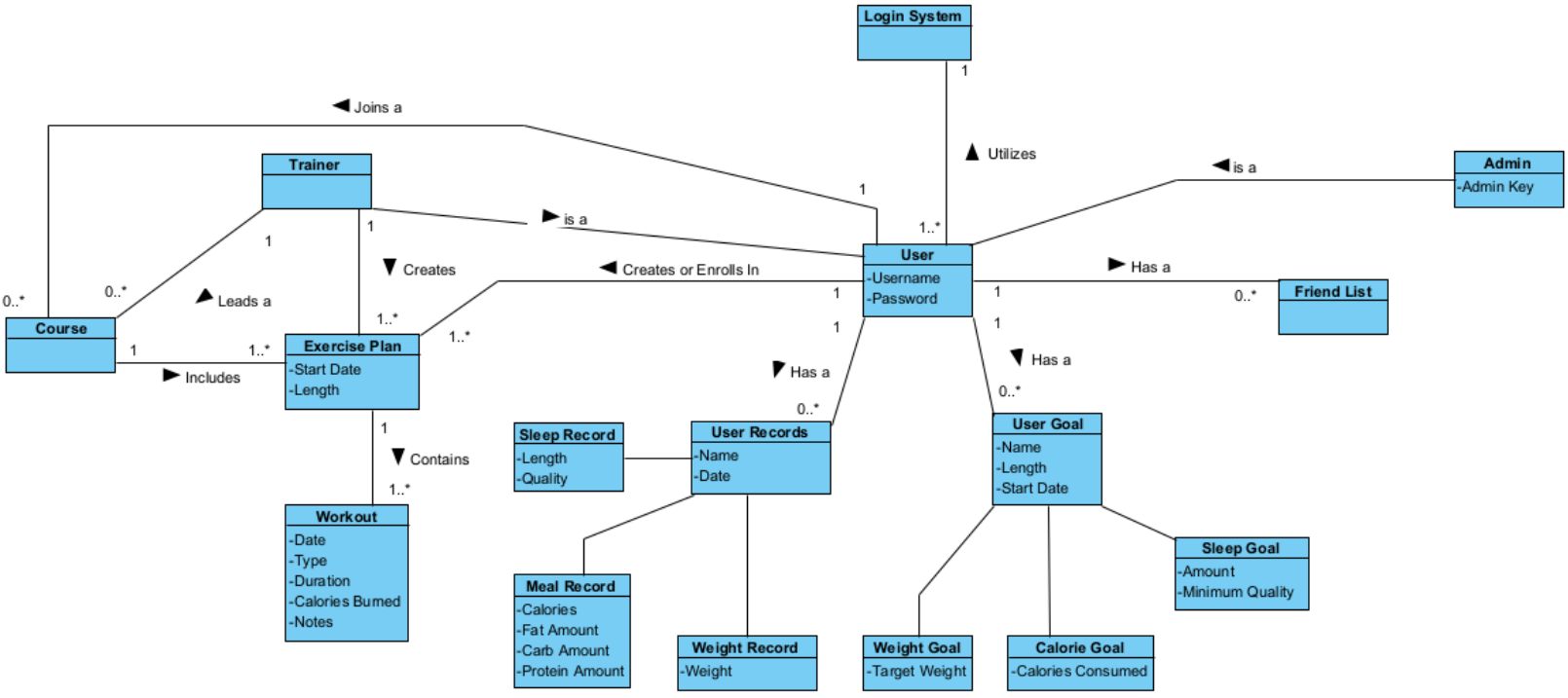
Alternative Flows:

- *3a.* If invalid values → showValidationError()
- *3b.* If user cancels delete → showNoChange()
- *a** If connection/server unavailable → showConnectionError()

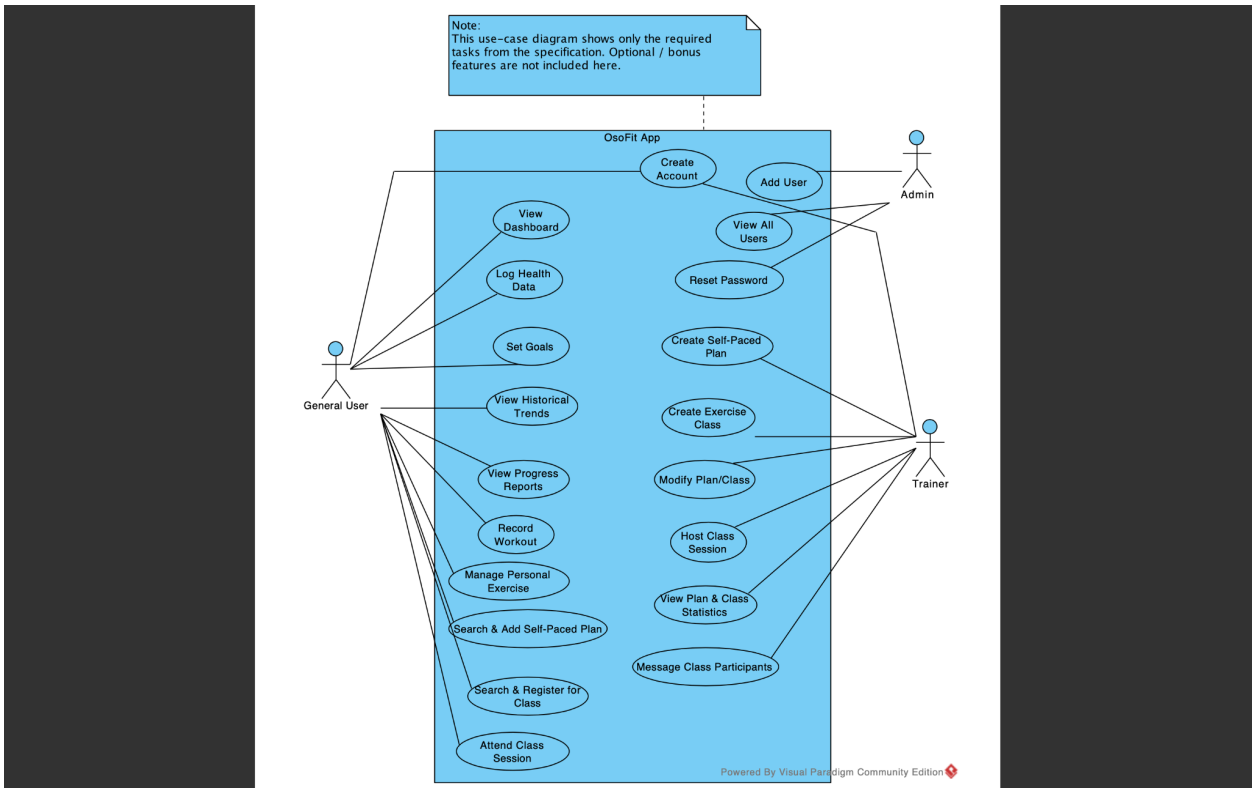
SSD:

Hours Worked:
Jack - 12 hours
Ethan - 9 hours
Dannis - 9 hours
Savannah - 11 hours
Mason - 11 hours
Connor - 12 hours

DOMAIN MODEL



USE-CASE DIAGRAM



Website: https://github.com/jaackfont05/OsoFit

Wireframes Drafts:

Home Screen

User Info

Past Sleep

Past workouts

Past meals

Contact Trainer

Request Trainer

User Support

Settings

Nutritional goals

Calories:

fat:

protein:

Carbs:

Today's Workout

Type:

Place:

length:

Plan: Link to workout plan

Injured?

Log Meal

Log Workout

Create Exercise

Log Sleep

Log Weight

OsoFit

Login

ATHLETIC PERFORMANCE

Enter email:

Enter password:

Forgot password?

Create Account

Account Creation

Create Account

Email:

Password:

Confirm Password:

Age:

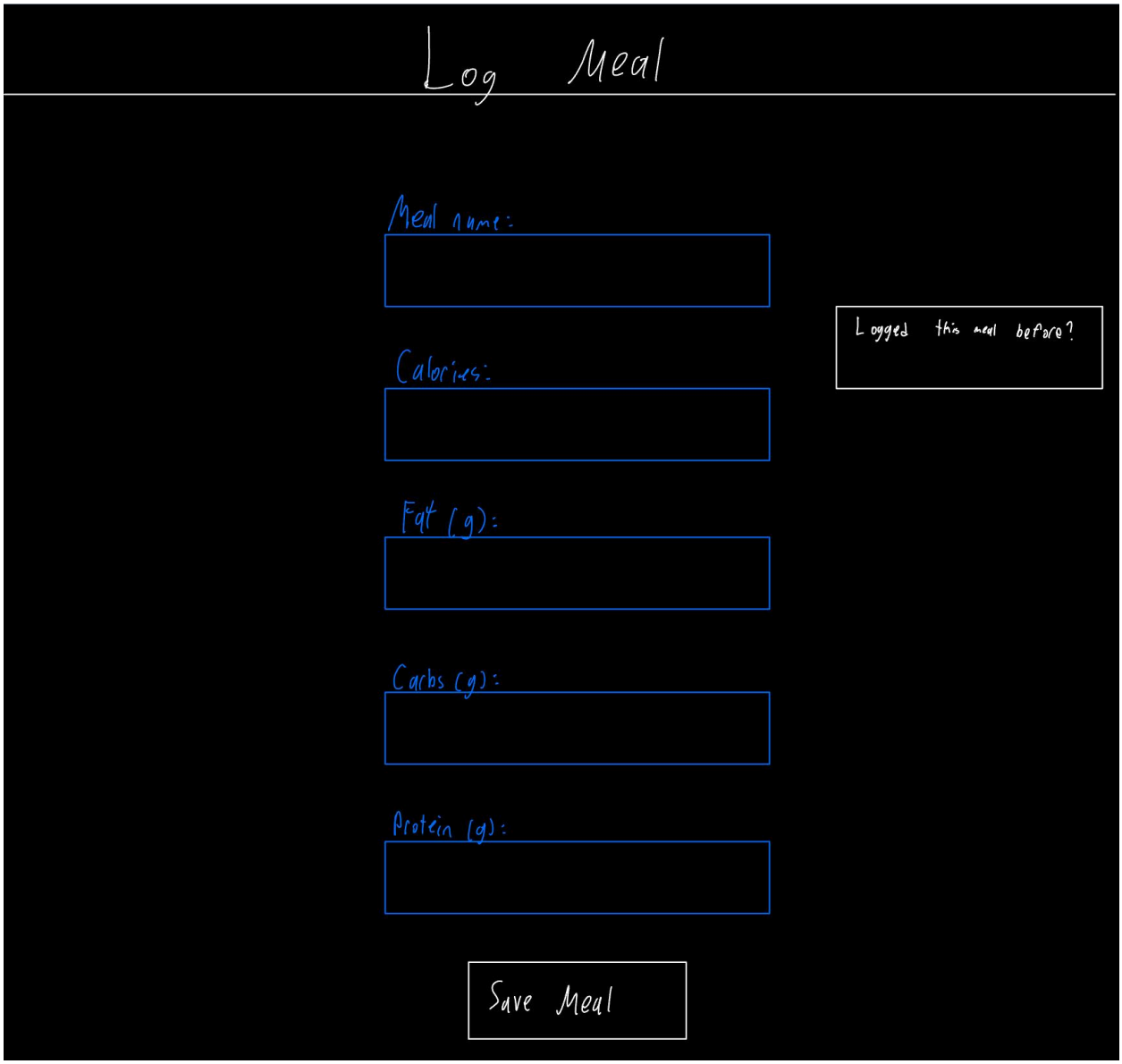
Weight:

Gender:

Experience:

upload profile pi

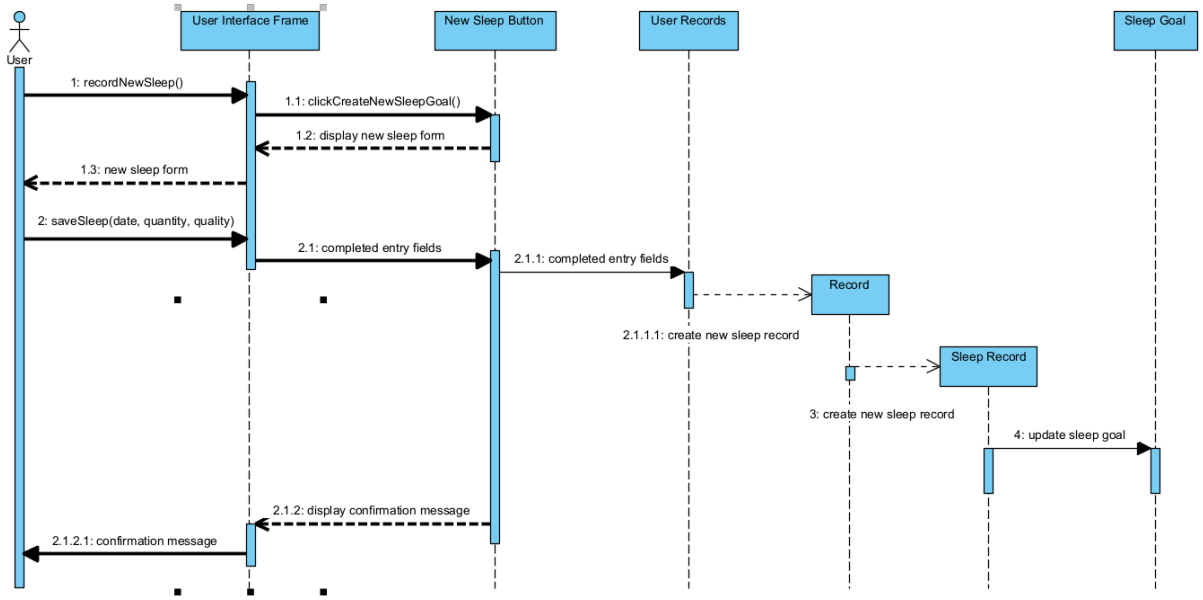
Confirm


Log Meal
Meal name:
Logged this meal before?
Calories:
Fat (g):
Carbs (g):
Protein (g):
Save Meal

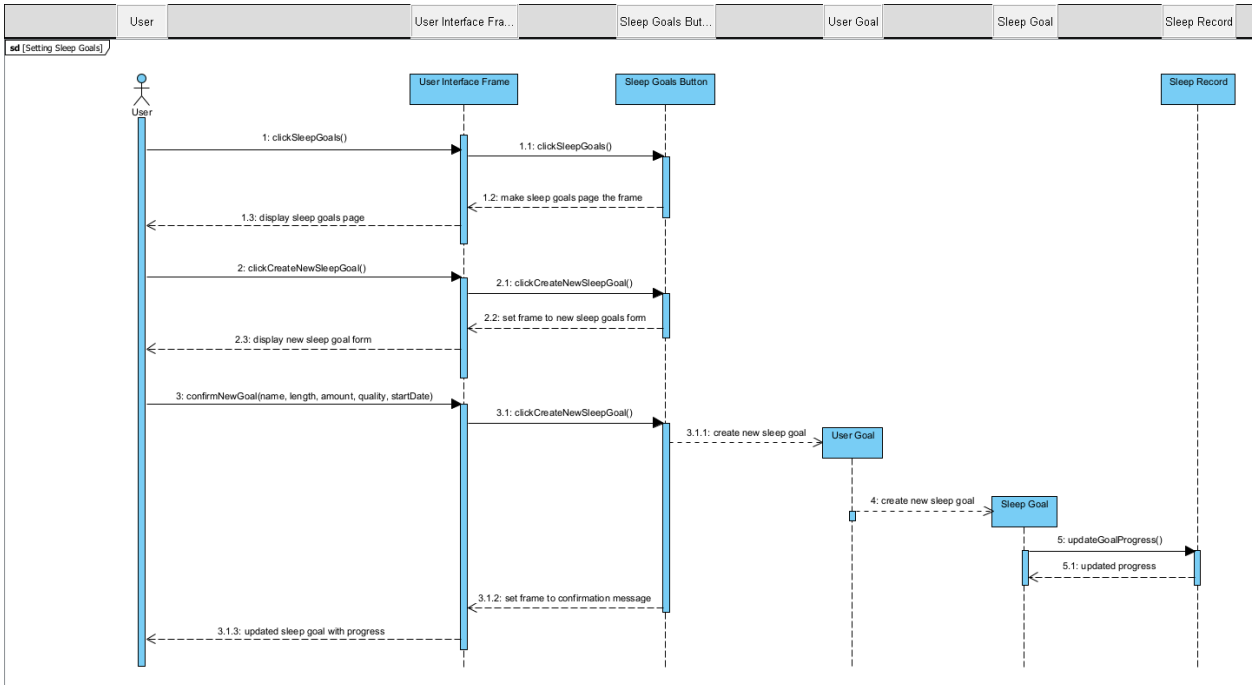
Project Deliverable 7:

SD Screenshots:
Mason:

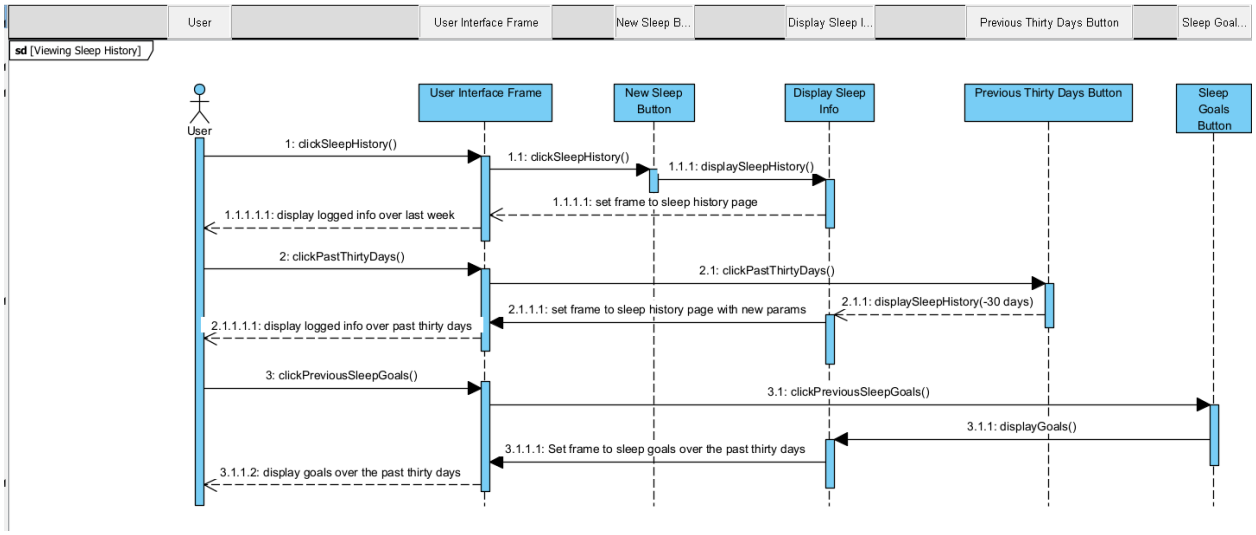
Use Case 1: Tracking Sleep



Use Case 2: Setting new sleep goal

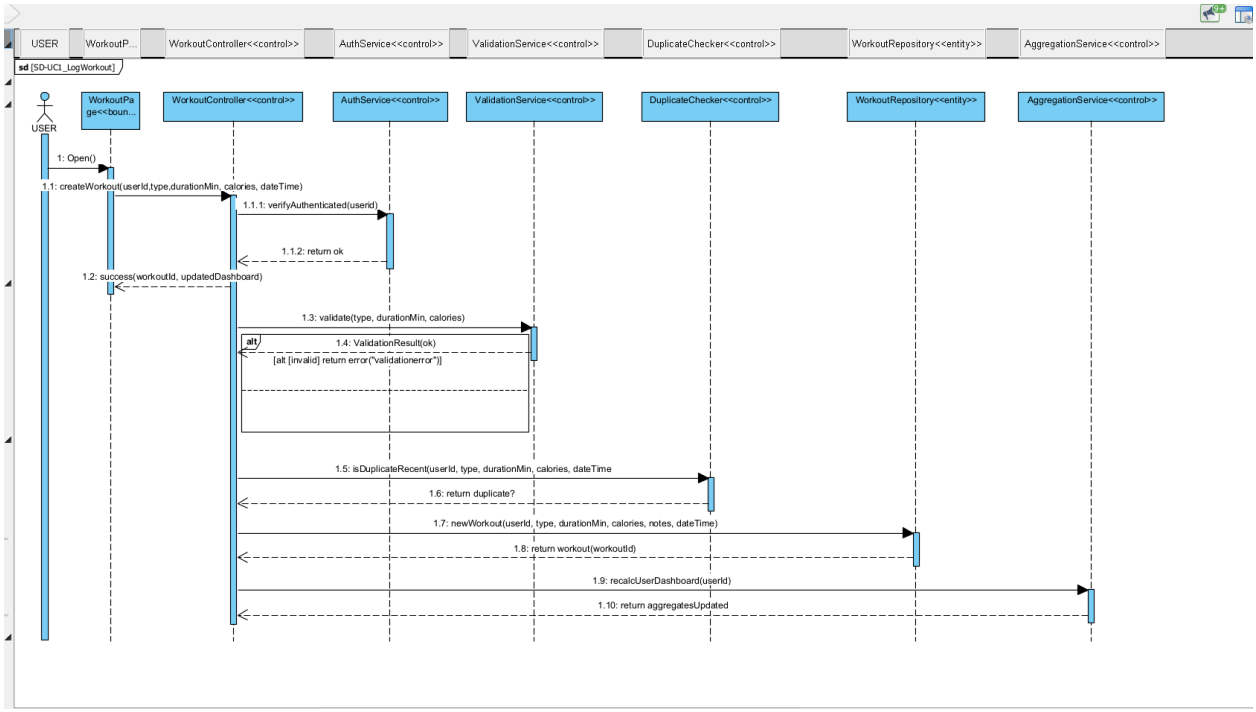


Use Case 3: Viewing Sleep history

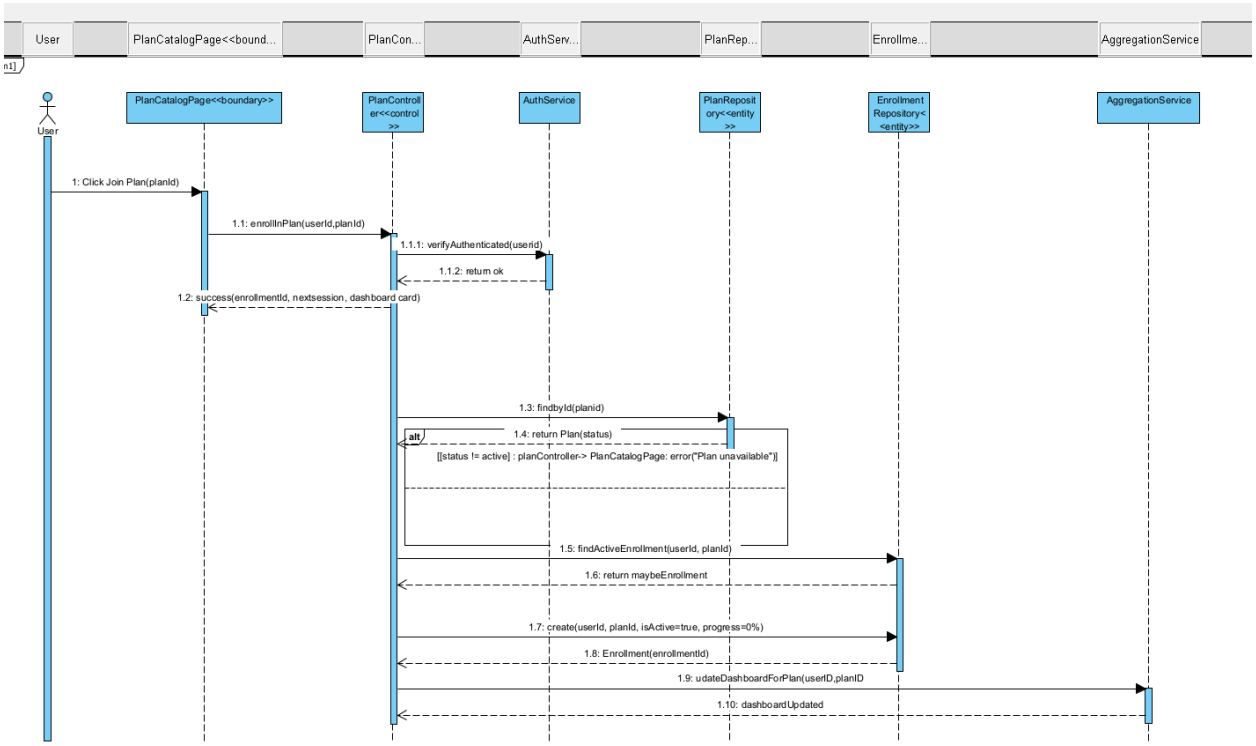


Connor:

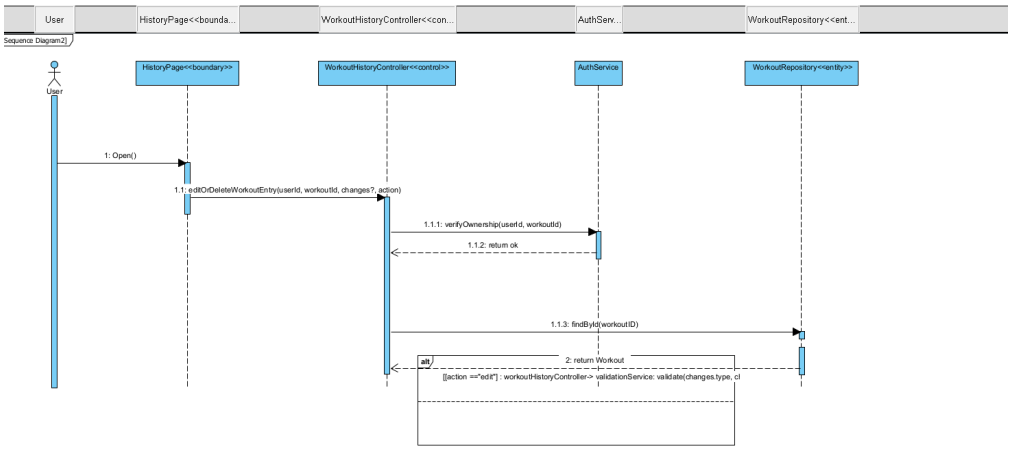
UC1: Log workout session



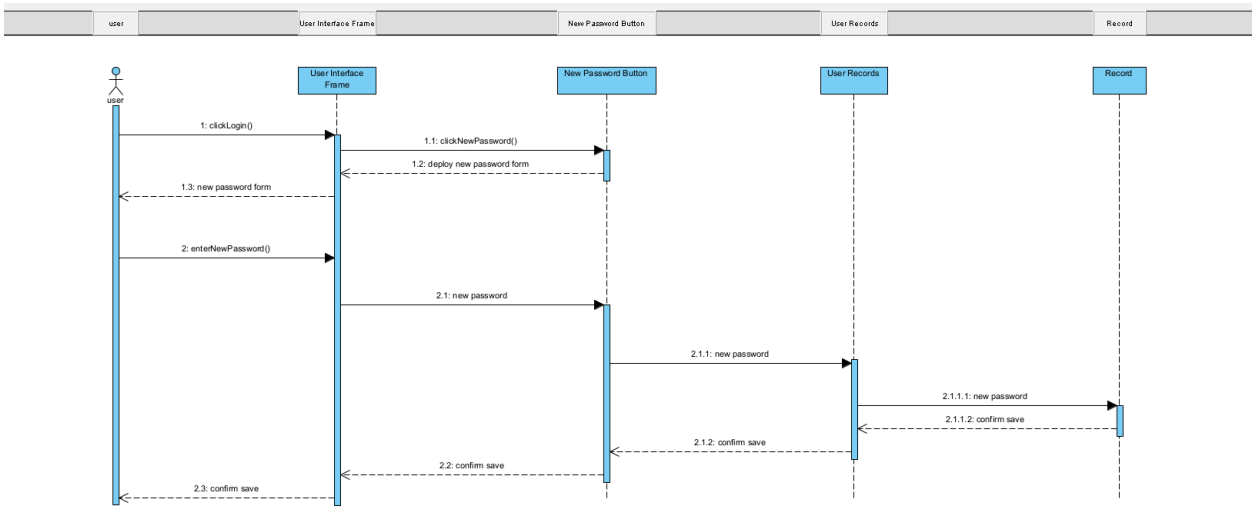
UC2: Join a training plan

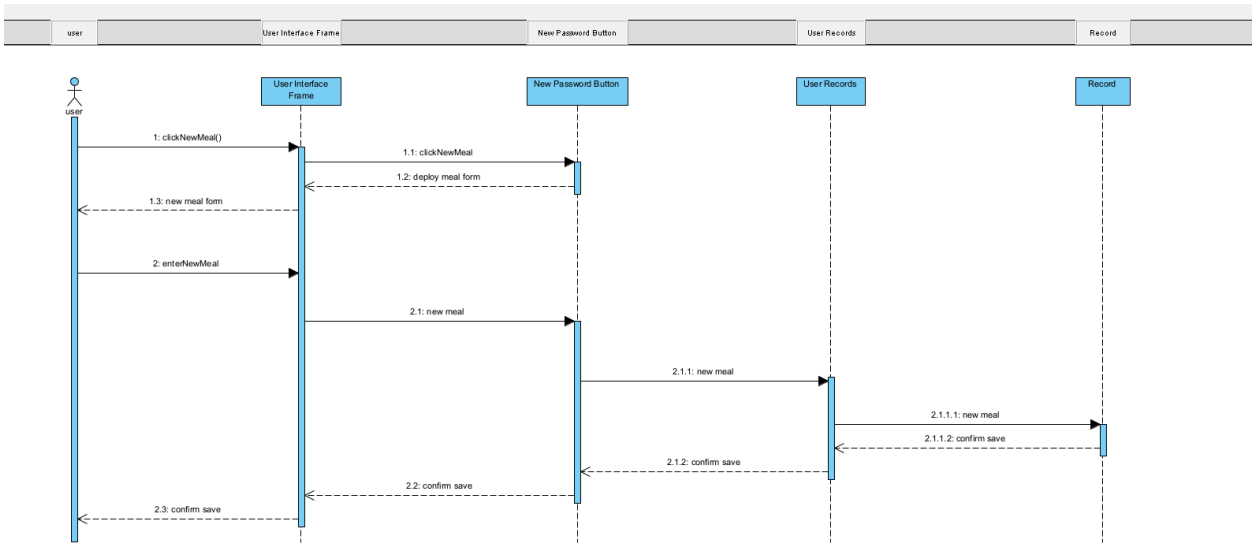


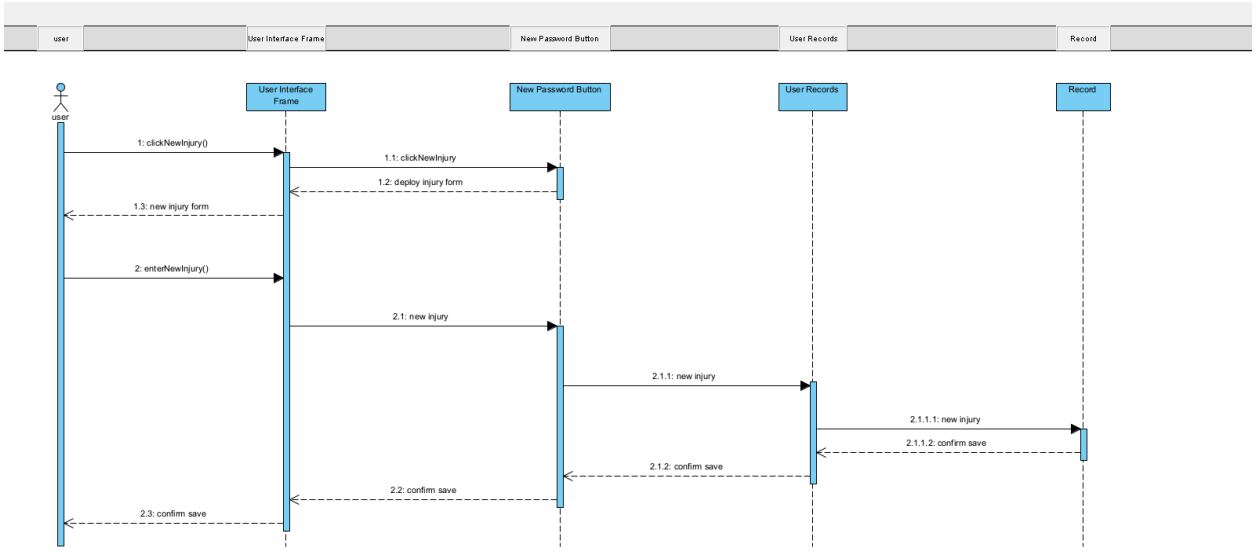
UC3: Edit/Delete workout



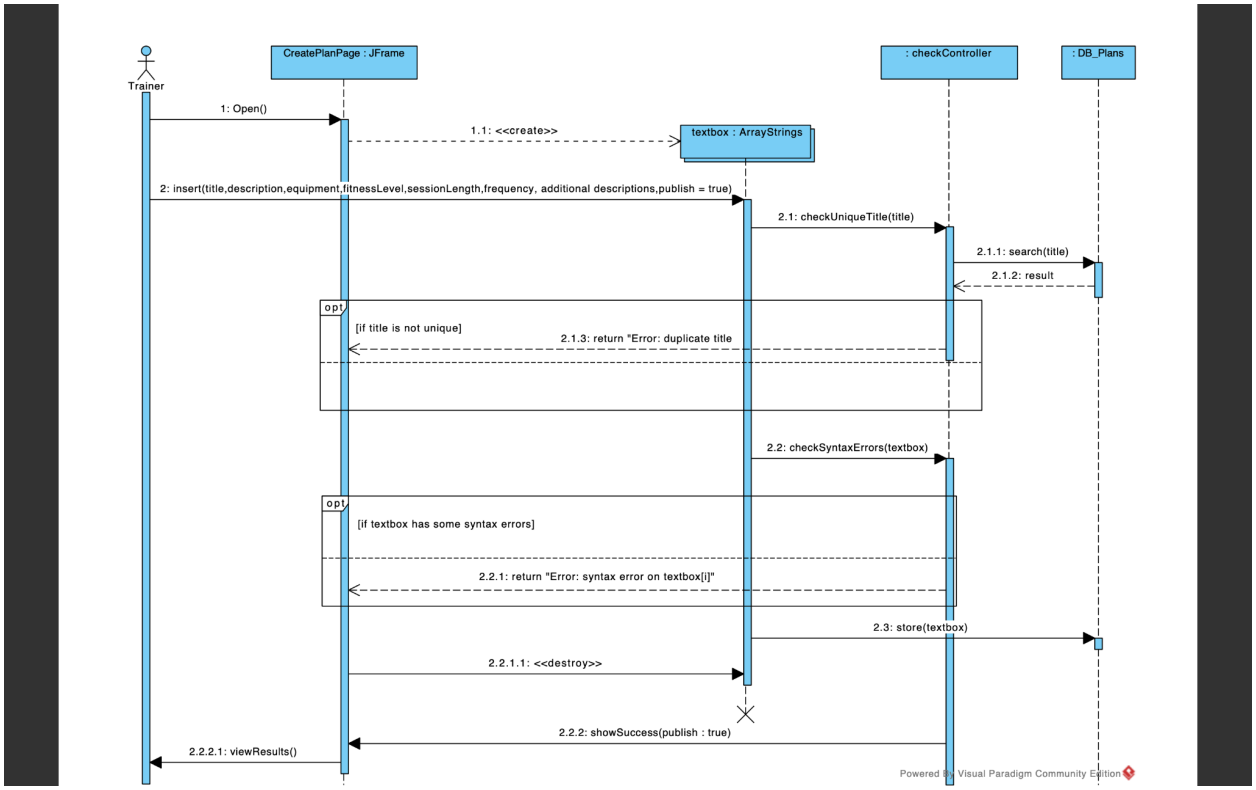
Jack Fontenot:



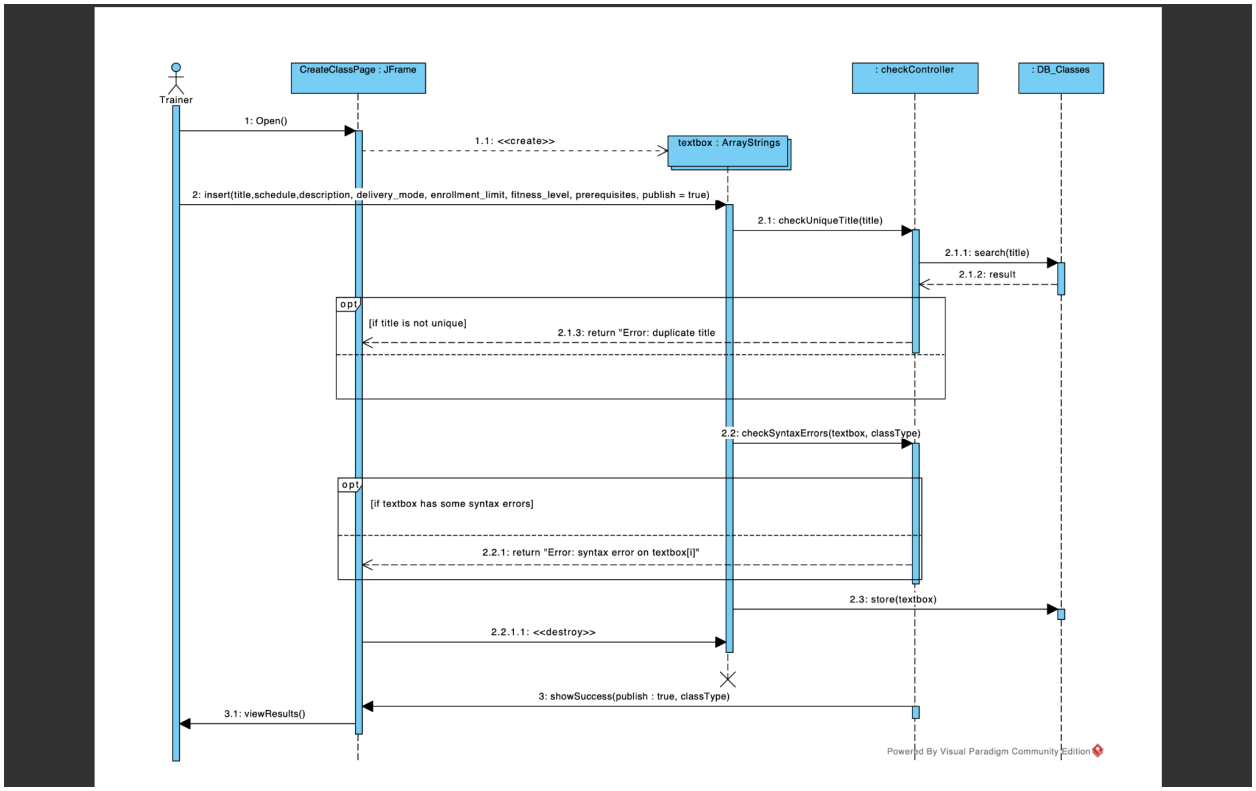




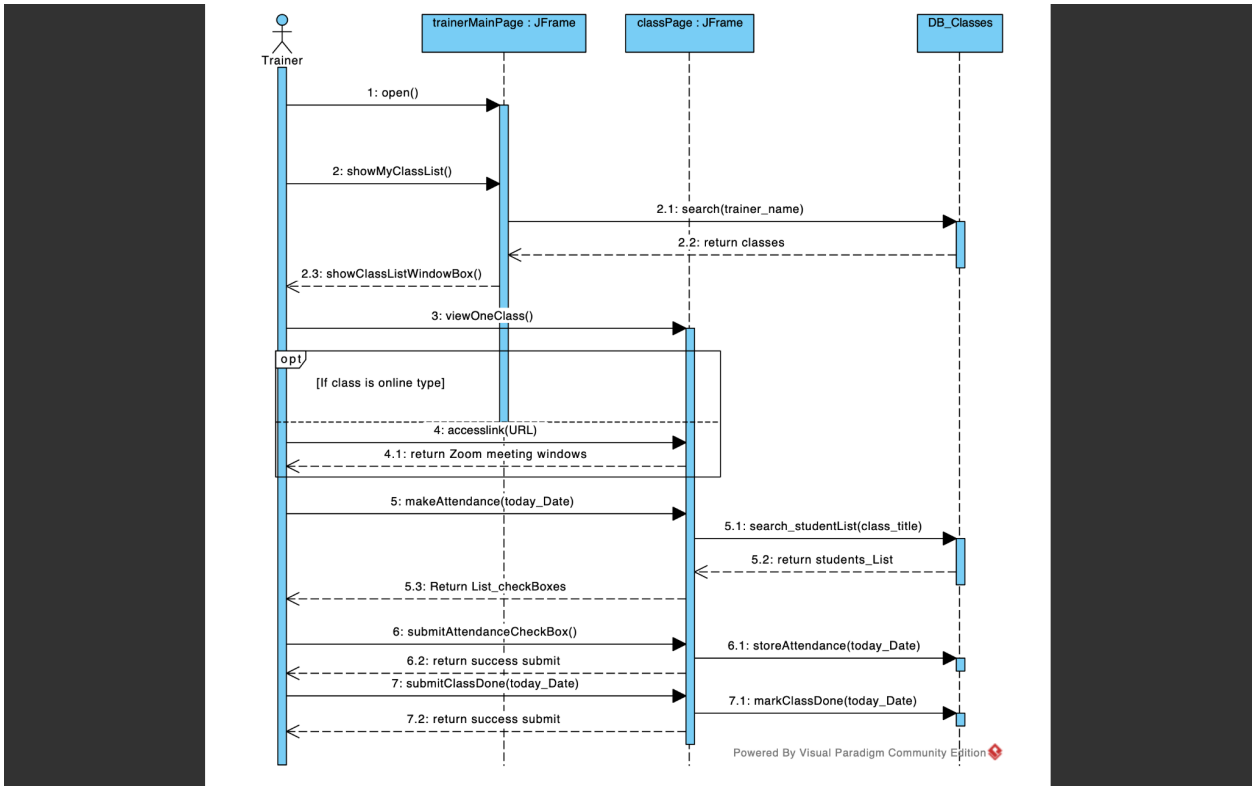
Dannis Wu:
UC1: Create a Self-Paced Exercise Plan



UC2: Create a Scheduled Class

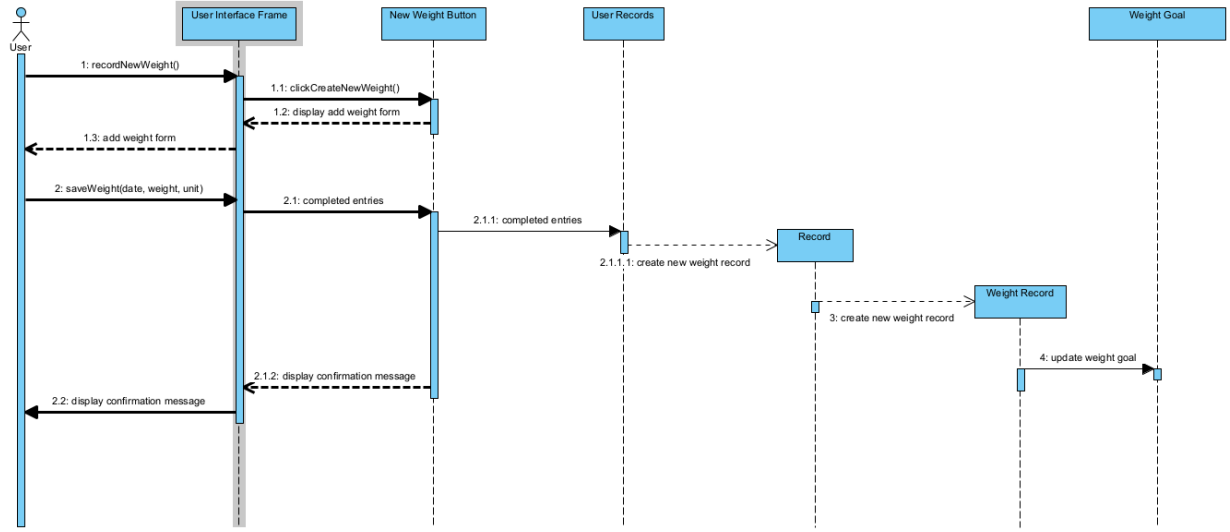


UC3: Host a Scheduled Class Session

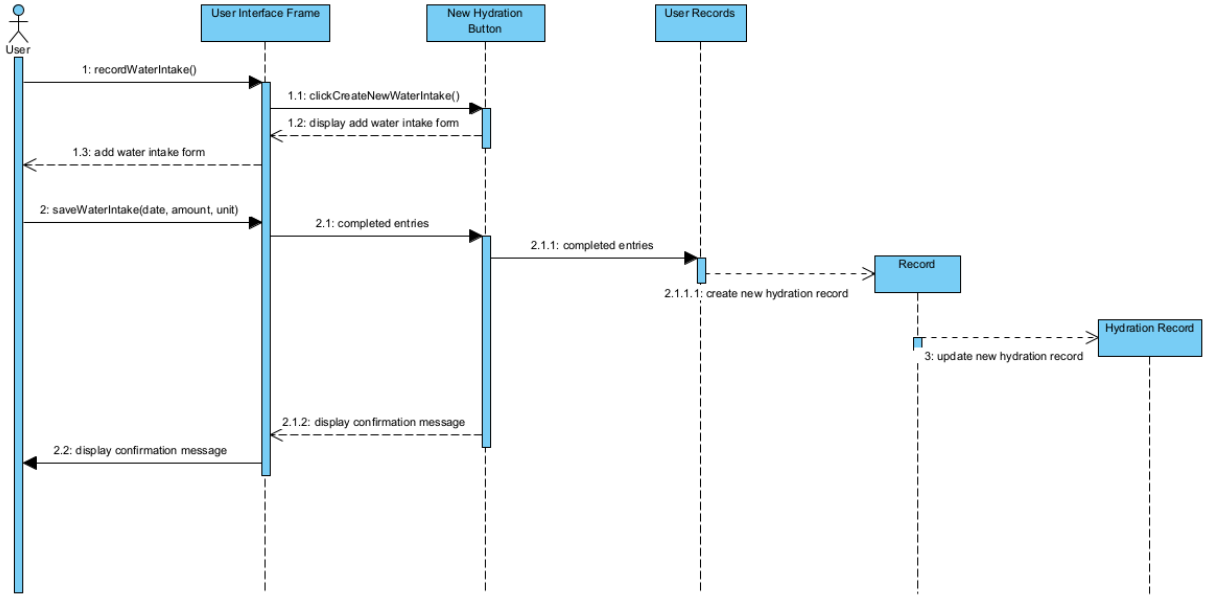


Savannah Johnson:

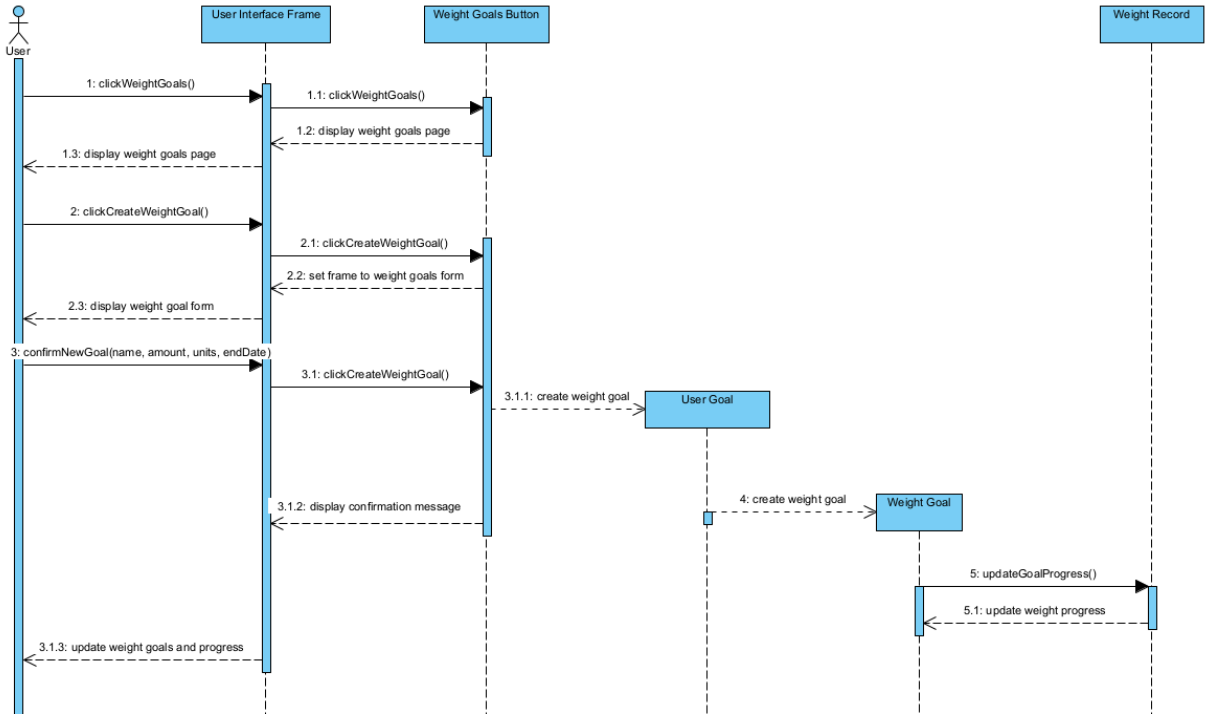
Use Case 1: Record Weight



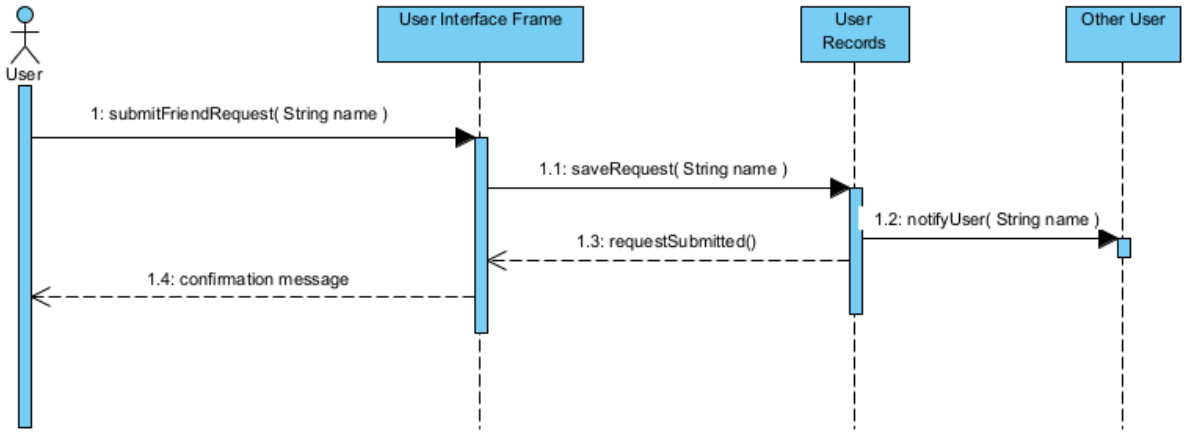
Use case 2: Record hydration



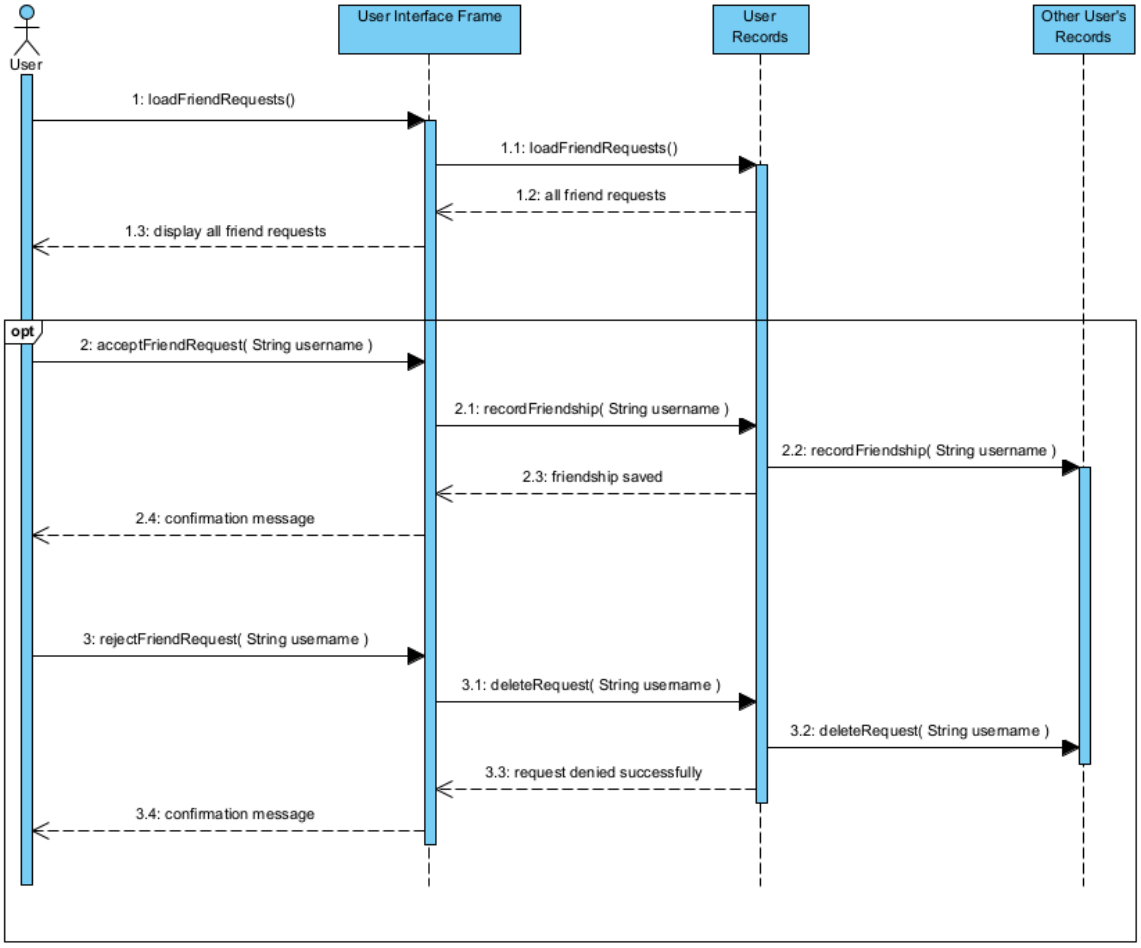
Use case 3: Add new weight goal



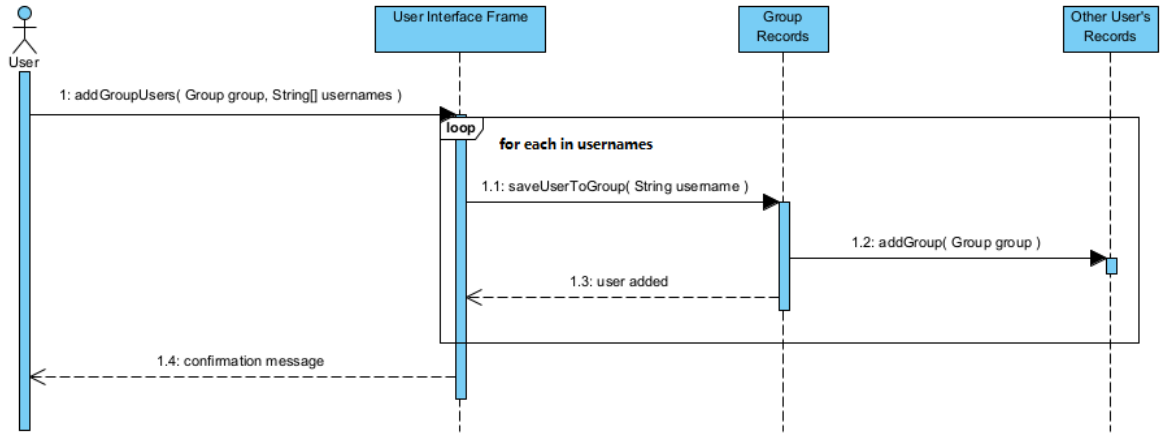
Ethan Rogers:
UC 1



UC 2



UC 3


User
User Interface Frame
Group Records
Other User's Records
1: addGroupUsers(Group group, String[] usernames)
loop
for each in usernames
1.1: saveUserToGroup(String username)
1.2: addGroup(Group group)
1.3: user added
1.4: confirmation message