

CSE3310 Fundamentals of Software Engineering Fall 2024

Final Report

Legacy: Personal Health and Fitness Tracker

Emma Slonaker, egs2749@mavs.uta.edu

Austin Mitchell, axm6256@mavs.uta.edu

Thien Nguyen, tpn3461@mavs.uta.edu

Huy Nguyen, hnn5648@mavs.uta.edu

Raza Inthanongsack, rxl8991@mavs.uta.edu

Date: 12/03/2024

Table of Contents

1. README	2
2. Introduction and Project Overview	3
3. Objectives	3
4. Project Context Diagram	6
5. System Requirements	7
6. Software Processes and Infrastructure	16
a. UML Diagrams	16
7. Conceptual Data Model	28
8. Test Cases	29
9. User Manual	42
10. Assumptions and Constraints	58
11. Delivery and Schedule	58
12. Stakeholder Approval Form	59

README

Emma Slonaker, egs2749@mavs.uta.edu
Austin Mitchell, axm6256@mavs.uta.edu
Thien Nguyen, tpn3461@mavs.uta.edu
Huy Nguyen, hnn5648@mavs.uta.edu
Raza Inthanongsack, rxl8991@mavs.uta.edu

The chat of the program uses the firebase tool in order to keep track of users and their chats between each other. In order to use firebase you will need to connect it using the tool provided within android studio and add the sdk there underneath authentication. To connect to the firebase database in the google-services.json file, put this url between the “” under firebase_url `https://healthchat-833ce-default-rtdb.firebaseio.com` . Then there is another database that is used to store other information, mainly information for the user specifically that is within the class Data.

The functionality to lookup a specific user within the chat instead of having a list of users that are eligible to chat too was not completed in time. The functionality to cumulatively update a user's metrics was not completed in time. When viewing the chart under Progress Monitoring, data will not be accurately reflected until it is imported first. Data is only reflected for steps metric.

The selected code for the report is just the main code that is more important but a lot of code was left out in order to fit the 15 page limit.

Introduction and Project Overview

The purpose of this document on the Personal Health and Fitness Tracker app is to outline the development of our mobile fitness application. It is designed to assist users to record and monitor the user's fitness activities and provide a platform to audit their workouts and goals.

This application will be available only to the Android platform, providing a user-friendly interface for the user. The application will provide fitness activities in which the user can participate, and track the user's progress and social features to showcase their progress to their peers.

Objectives

→ Business Objectives:

- ◆ **Objective 1:** User Registration: General User and Personal Trainer members must provide the following information prior to using the system:
 - First Name, Middle Name {Optional}, Last Name
 - General User/ Guest User {Optional} / Personal Trainer
 - Home and mailing addresses
 - E-mail address

- ◆ **Objective 2:** Login functionality: General User and Personal Trainer members must log in to the system with a username and password established during the Member registration stage. Guest users can skip the login step with limited features.

- ◆ **Objective 3:** Activity Categories: The following Service categories will be supported initially:
 - Running
 - Walking
 - Cycling
 - Yoga
 - High-intensity interval training (HIIT)
 - Weightlifting

- ◆ **Objective 4:** Goal Setting: Users must be able to set personal fitness goals. Examples include:
 - Running 5 miles a week
 - Burning 2000 calories a week
 - Lose/gain 10 lbs. a month

- ◆ **Objective 5:** Tracking: Users must be able to track metrics depending on the exercise. Imported data from wearable devices will be supported. The following metrics will be measured for each activity type:
 - Running metrics: calories burned, heart rate, distance, steps
 - Walking: calories burned, distance, steps
 - Cycling: calories burned, heart rate, distance
 - Yoga: duration, heart rate
 - High-intensity interval training: calories burned, heart rate
 - Weightlifting: calories burned, heart rate
- ◆ **Objective 6:** Progress Monitoring: Users must be able to view their fitness progress. Fitness progress will be visually displayed using graphs and/or charts. Graphical representations will be categorized by activity type. These graphical representations will be formed based on the data collected in accordance with Objective 5.
- ◆ **Objective 7:** Exercise Plans: Users must be able to follow pre-set workout routines or custom workout routines.
- ◆ **Objective 8:** Review of Trainer: A review feature must be available to the user to provide feedback to the personal trainer.
- ◆ **Objective 9:** Subscription: Users must be registered to pay for a subscription, see Objective 1. App subscription must include zero ads, special training videos, and personal trainer instructions. Subscription terms can be monthly or annually with a discount.
- ◆ **Objective 10:** Payment Integration: Users must pay with debit/credit card or PayPal. In-app purchases include:
 - Subscriptions
 - Personalized workout plans
 - Virtual trainers
- ◆ **Objective 11:** Social Feature: Users must be allowed to share fitness achievements, fitness challenges, and updates on social media via a one-button-share function
- ◆ **Objective 12:** Communication: Users may receive notifications for activity reminders, progress updates, and social interactions via email and/or text. Users

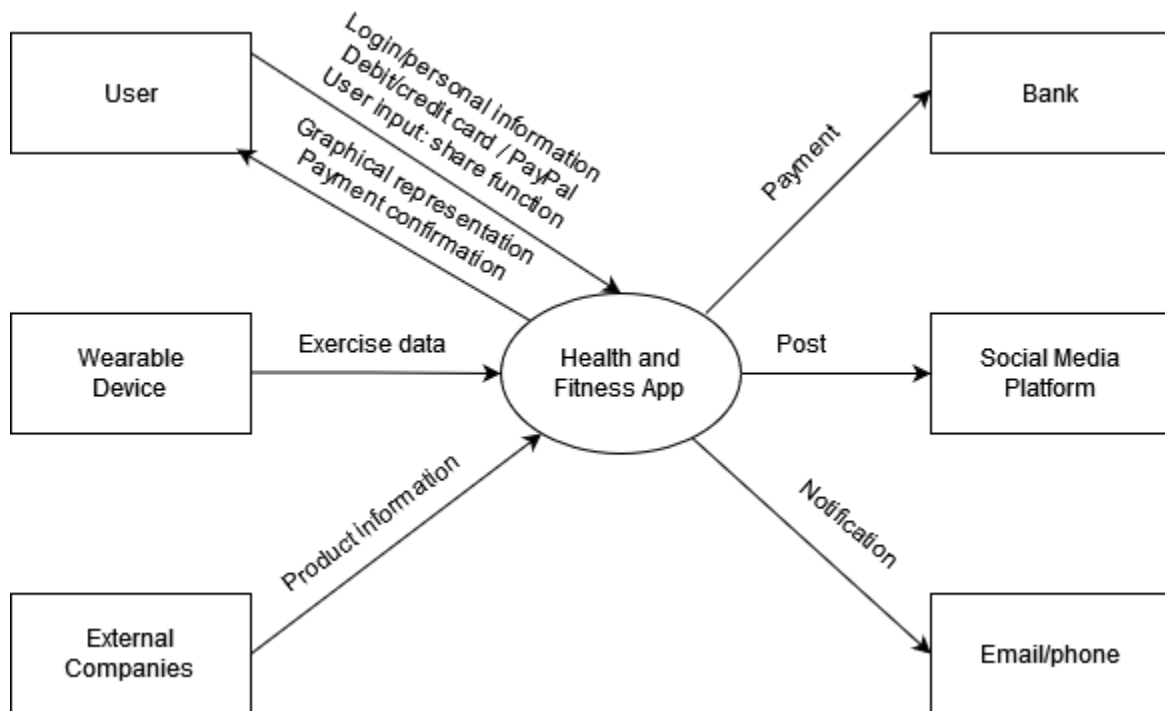
may select this preference under their profile.

- ◆ **Objective 13:** Advertisement: The application must advertise workout plans, personal trainer programs, and fitness equipment within the app. The application must also allow health-related businesses to advertise within the app.

→ **System Objectives:**

- ◆ **Objective 1:** System will be an Android based Mobile system
- ◆ **Objective 2:** Wearable device API will be utilized
- ◆ **Objective 3:** SQLite will be used to store data
- ◆ **Objective 4:** Application will be built with Java
- ◆ **Objective 5:** In-payment methods will support debit/credit card and PayPal
- ◆ **Objective 6:** System will have a user-friendly interface, graphical interfaces will be used, such as Java Graphics
- ◆ **Objective 7:** One-button-share function will support multiple social media platforms (ie. X, Facebook, Instagram, messaging, email)

Project Context Diagram



System Requirements

“User Registration” Requirements

Requirement Title:	Registration
Sequence No:	001
Short description:	Register New User
Description	<p>New users must register before accessing the application. The following information will be collected:</p> <ul style="list-style-type: none">- Name {First, Last name}- Email address- Phone Number- Select an ID {at least 8 characters long, alphanumeric, special character not allowed}- Select a password {Must include letters, at least one number, at least one capital letter, one special character}- Select a security question/answer (to be use for password reset) <p>User can press:</p> <ul style="list-style-type: none">- Submit- Cancel (i.e discard changes)- Exit (or close)
Pre-Conditions:	<ul style="list-style-type: none">- Application must be loaded already- Duplicate registration not allowed
Post Conditions:	<ul style="list-style-type: none">- All “Saved” changes will be kept permanently
Other attributes:	<ul style="list-style-type: none">- None

“Login” Requirements

Requirement Title:	Login
Sequence No:	002
Short description:	Login Existing User
Description:	<p>Enter User Id and Password (already established during registration) and press Submit</p> <p>User can press:</p>

	<ul style="list-style-type: none"> - Submit - Forgot ID or Password - Cancel(i.e discard changes) - Exit Screen (or close)
Pre-Conditions:	<ul style="list-style-type: none"> - User must have already registered
Post Conditions:	<ul style="list-style-type: none"> - Keep a log of the date and time of login
Other attributes:	<ul style="list-style-type: none"> - Allow maximum of 3 tries, then recommend using “Forgot Password”

Requirement Title:	Forgot Password / ID
Sequence No:	003
Short description:	Account Recovery for Existing User
Description:	<p>Ask for the user's email address and the answer to the security question selected during registration. If both are correct, the system will email the user a temporary password to reset their login credentials</p> <p>User can press:</p> <ul style="list-style-type: none"> - Submit - Cancel (i.e discard changes) - Exit Screen (or close)
Pre-Conditions:	<ul style="list-style-type: none"> - The user must have already registered - The system validates both the email address and security question
Post Conditions:	<ul style="list-style-type: none"> - The system sends a temporary password, and the user is recommended to change it after logged in
Other attributes:	<ul style="list-style-type: none"> - None

“Tracking” Requirements

Requirement Title:	Tracking
Sequence No:	004

Short description:	Tracking and Progress Monitoring
Description:	The system allows users to track their fitness and health progress through a dashboard interface. Data is stored in a database and may be imported from external devices, such as Samsung Galaxy watch or a fitness tracker.
Pre-Conditions:	<ul style="list-style-type: none"> - Users must be logged in to view and track their progress.
Post Conditions:	<ul style="list-style-type: none"> - Progress is continuously monitored and updated based on the user's activities, and the information is stored in the database. - Users can view their history of workouts and other activities.
Other attributes:	<ul style="list-style-type: none"> - None.

“Activity Categories” Requirements

Requirement Title:	Activity Categories
Sequence No:	005
Short description:	Activity Categories
Description:	<p>Users can access and choose from predefined Activity Categories such as workout plans, cardio exercises, strength training, and flexibility exercise. The system allows users to select a category, log their activity, and track their progress.</p> <p>User can press:</p> <ul style="list-style-type: none"> - Select Activity Category - Start Activity - Log Activity Completion
Pre-Conditions:	<ul style="list-style-type: none"> - The user must be logged in.
Post Conditions:	<ul style="list-style-type: none"> - The system tracks the user’s activity under the selected category and updates progress toward their goals.
Other attributes:	<ul style="list-style-type: none"> - Users can view statistics and history of each activity category. - New categories may be added over time with system updates.

“Goal Setting Request” Requirements

Requirement Title:	Goal Setting
Sequence No:	006
Short description:	Goal Setting
Description:	<p>Users can set goals related to their health and fitness within the app. The system will track their activity and update progress toward the goal in real-time. Once the goal is completed, it will be marked and sent out a notification.</p> <p>User can press:</p> <ul style="list-style-type: none"> - Create new goal - View/Edit existing goal - Delete goal
Pre-Conditions:	<ul style="list-style-type: none"> - The user must be logged in to access the goal-setting function.
Post Conditions:	<ul style="list-style-type: none"> - The system continuously updates the user's progress toward goals based on completed activities. - When all tasks under a goal are finished, the goal is marked complete.
Other attributes:	<ul style="list-style-type: none"> - None.

“Progress Monitoring Request” Requirements

Requirement Title:	Progress Monitoring Request Reminders.
Sequence No:	007
Short description:	Progress Monitoring Request Reminders.
Description:	<p>The system tracks the user's progress toward their goals and sends reminders if goals are not completed by a set time each day. Users can request reminders on specific goals they want to complete. For example, if a user has a daily workout goal, the system will check progress at the specific time, and if incomplete, send a reminder to encourage users to finish their workout.</p> <p>User can press:</p> <ul style="list-style-type: none"> - Set reminder time - Dismiss reminder - Snooze reminder

Pre-Conditions:	<ul style="list-style-type: none"> - Users must have set a goal in the app. - The user must enable notifications and provide a reminder time.
Post Conditions:	<ul style="list-style-type: none"> - The system will check goal completion status daily at the reminder time. - If the goal is incomplete, the system sends a reminder to the user to finish the activity.
Other attributes:	<ul style="list-style-type: none"> - Users can adjust reminder settings, including frequency and time of day. - Reminders are sent via push notification, email, or SMS, depending on user preference.

“Exercise Plans” Requirements

Requirement Title:	Exercise Plans
Sequence No:	008
Short description:	Exercise Plans
Description:	<p>The app will provide different workout plans for the users to choose and customize. Users can make their own workout plans and track their progress.</p> <p>User can press:</p> <ul style="list-style-type: none"> - Select a workout plan - Create a workout plan - Edit workout plan - Start plan
Pre-Conditions:	<ul style="list-style-type: none"> - The user must be logged in to access exercise plans. - The user must have already bought a subscription to access exercise plans. - The user must select a plan or create a custom plan.
Post Conditions:	<ul style="list-style-type: none"> - The system will track the completion of each exercise within the selected plan. - Users can view their progress and modify plans based on their preference.
Other attributes:	<ul style="list-style-type: none"> - Users receive reminders to complete exercise if progress is incomplete.

	- Exercise plans can be shared with friends or on social media.
--	---

“Review of Trainer” Requirements

Requirement Title:	Review of Trainer
Sequence No:	009
Short description:	Making a review of a trainer
Description:	<p>The system asks/displays reviews of the trainer to the user. Reviews are made up into 2 parts, the rating and the comment that came from other users that had been under the trainer prior.</p> <p>What user can press:</p> <ul style="list-style-type: none"> - Select the rating between 1-5 stars (1-10) - Comment box to type in up to [enter amount of char] characters - Next button - Cancel
Pre-Conditions:	<ul style="list-style-type: none"> - User must be logged in - The user must have been under the trainer prior. - The user must be on the trainer's page.
Post Conditions:	<ul style="list-style-type: none"> - The system will add the user review to a database. - The review will be displayed under the trainer reviews.
Other attributes:	- None

“Subscription” Requirements

Requirement Title:	Subscription
Sequence No:	010
Short description:	Paying/enabling or disabling subscription plan
Description:	<p>A subscription model that allows the user to pay a set amount per month (or annual) to the subscription plan. The subscription plan will disable advertisements for the user while it's active and enable premium trainer support for the user. If the user is in a subscription plan there is a prompt to cancel the plan if needed.</p> <p>User can press:</p> <ul style="list-style-type: none"> - Selection tab to either the monthly or annual plan displaying the price. (when not in a subscription plan)

	<ul style="list-style-type: none"> - Confirm button (when not in a subscription plan) - Cancel button (when in a subscription plan) - Back button
Pre-Conditions:	No subscription plan <ul style="list-style-type: none"> - User must be logged in and online in the application - When the user isn't in a plan Cancellation <ul style="list-style-type: none"> - To cancel user must be in a subscription plan already
Post Conditions:	No subscription plan <ul style="list-style-type: none"> - System will open the the payment screen (payment integration) Cancellation <ul style="list-style-type: none"> - System will send a cancellation request to the database to disable the subscription till the next payment.
Other attributes:	<ul style="list-style-type: none"> - The pricing will change spending on the user's country and their regional currency

“Payment Integration” Requirements

Requirement Title:	Payment Integration.
Sequence No:	011
Short description:	Payment Integration.
Description:	Users can purchase personalized workout plans or hire virtual trainers via in-app payment. Payments methods should support debit/credit cards and Paypal.
Pre-Conditions:	<ul style="list-style-type: none"> - The user must have an active account and add a payment method.
Post Conditions:	<ul style="list-style-type: none"> - Payment details are securely stored, and services are unlocked after successful payment.
Other attributes:	<ul style="list-style-type: none"> - None.

“Social Media Sharing” Requirements

Requirement Title:	Social Feature
Sequence No:	012

Short description:	Social Media Sharing.
Description:	<p>Users can share their fitness achievements, compete in challenges with friends, and post updates on social media platforms through a share button function that connects apps like Snapchat, Instagram, Facebook, Thread, X, etc.</p> <p>User can press:</p> <ul style="list-style-type: none"> - Share button - Icons of social medias that is supported
Pre-Conditions:	<ul style="list-style-type: none"> - Users must link their social media account within the app.
Post Conditions:	<ul style="list-style-type: none"> - Activity updates are posted on the selected social media platform.
Other attributes:	<ul style="list-style-type: none"> - None.

“Communication” Requirements

Requirement Title:	Notification
Sequence No:	013
Short description:	Creating/Sending Notification
Description:	<p>Users can enable notifications within the settings, the system will create and send notifications to the user, notifying the user any updates and reminders from trainers or user goals.</p> <p>What user can press:</p> <ul style="list-style-type: none"> - The notification pop up outside the application - Enabling Notification (in the setting)
Pre-Conditions:	<ul style="list-style-type: none"> - The ‘Allow notification permissions’ in the user’s systems. - System sending the message notification
Post Conditions:	<ul style="list-style-type: none"> - Open the application.
Other attributes:	<ul style="list-style-type: none"> - Users must enable notification settings in their application.

“Advertisement” Requirements

Requirement Title:	Advertisement
Sequence No:	014

Short description:	Displaying advertisements to the user
Description:	<p>At a certain point of time or at a specific menu within the application it will display internal advertisements (directly points to the in app monetization) and external advertisements (leading towards health products that are outside the application).</p> <p>User can press:</p> <ul style="list-style-type: none"> - The popup advertisement - The exit button of the advertisement
Pre-Conditions:	<ul style="list-style-type: none"> - Advertisements must be active on the screen. - The user must not be in the subscription plan. - User must be in the application
Post Conditions:	<ul style="list-style-type: none"> - Open the towards the in app monetization for internal advertisements - Open the browser leading to the outside product for external advertisements - Close advertisements when closing the advertisement
Other attributes:	<ul style="list-style-type: none"> - None

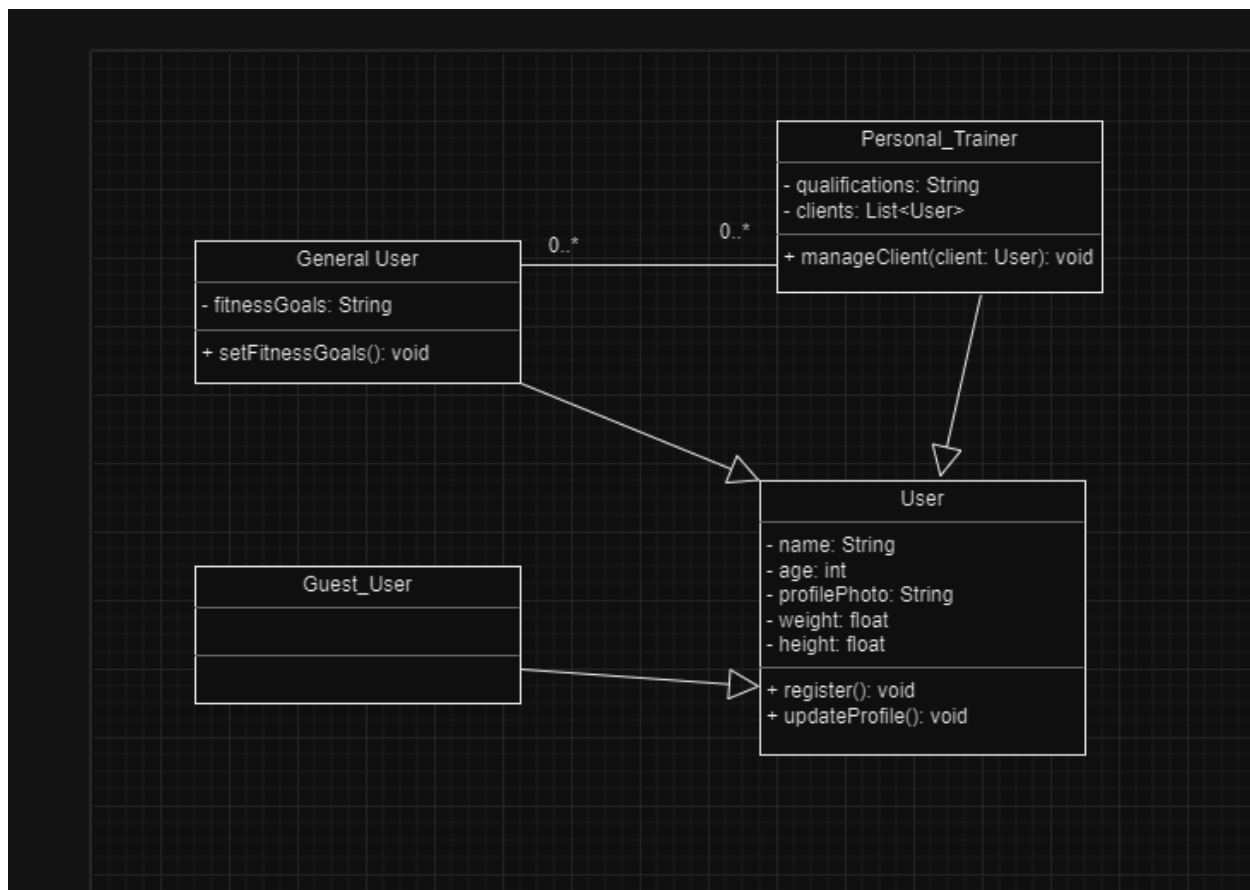
Software Processes and Infrastructure

Hardware and Infrastructure

Android versions might not be compatible with our program or our program will not be up to date with new releases.

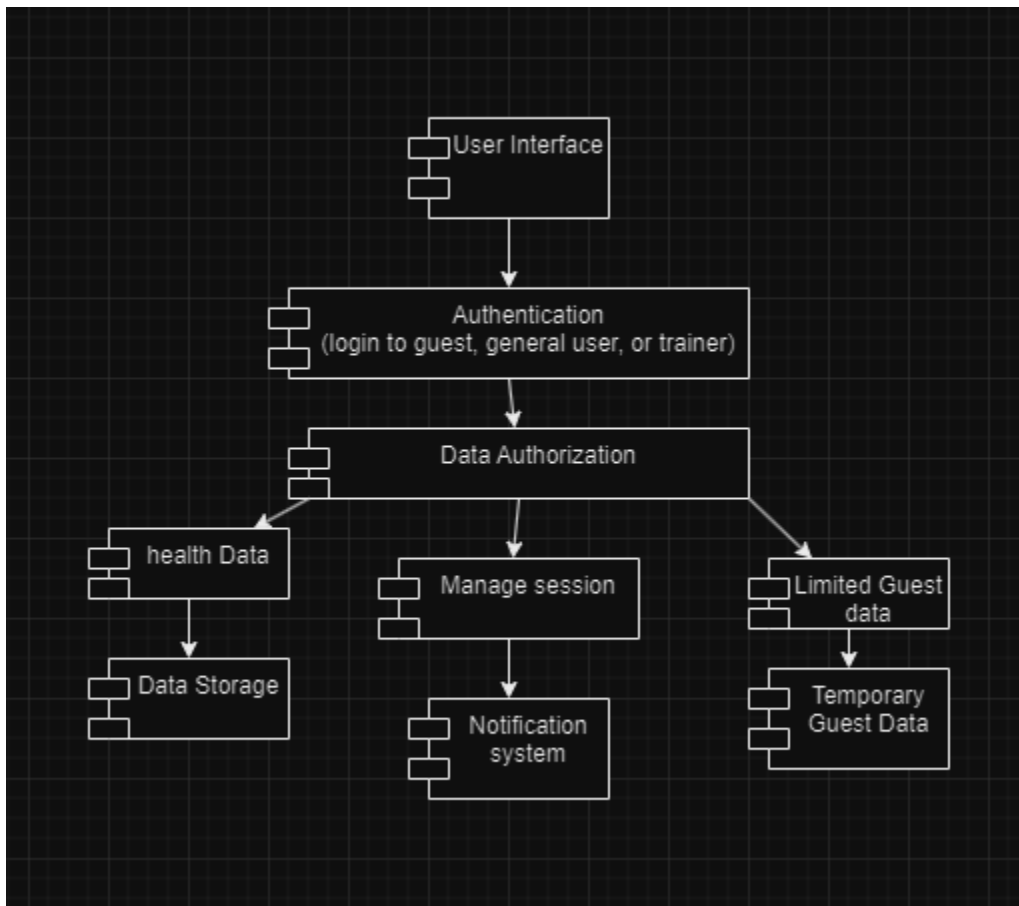
UML Diagrams

App Registration: Class Diagram (Austin Mitchell):



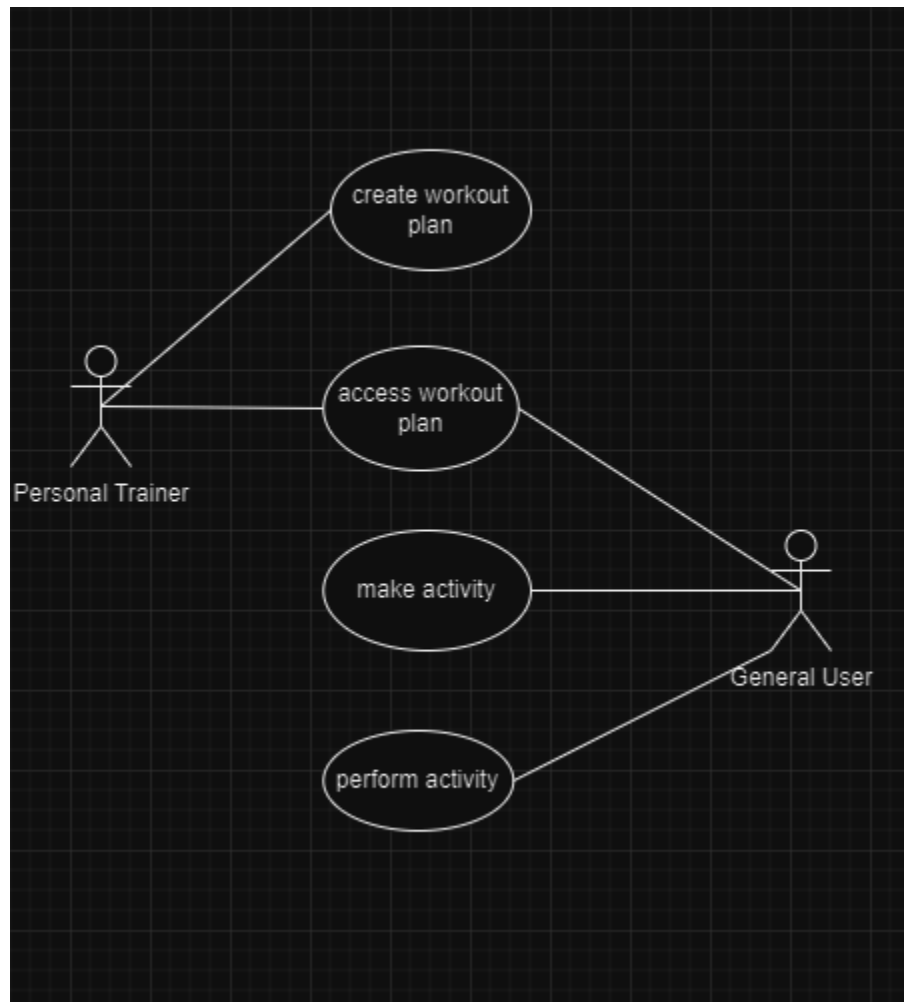
In this class diagram the User class functions as a base case for the other 3 subclasses that inherit the attributes of the User class. The General User and Personal trainer have their own methods and attributes and have an association that general users can have none to many personal trainers and personal trainers can have none to many Clients (General Users).

Login & Authentication: Component Diagram (Austin Mitchell):



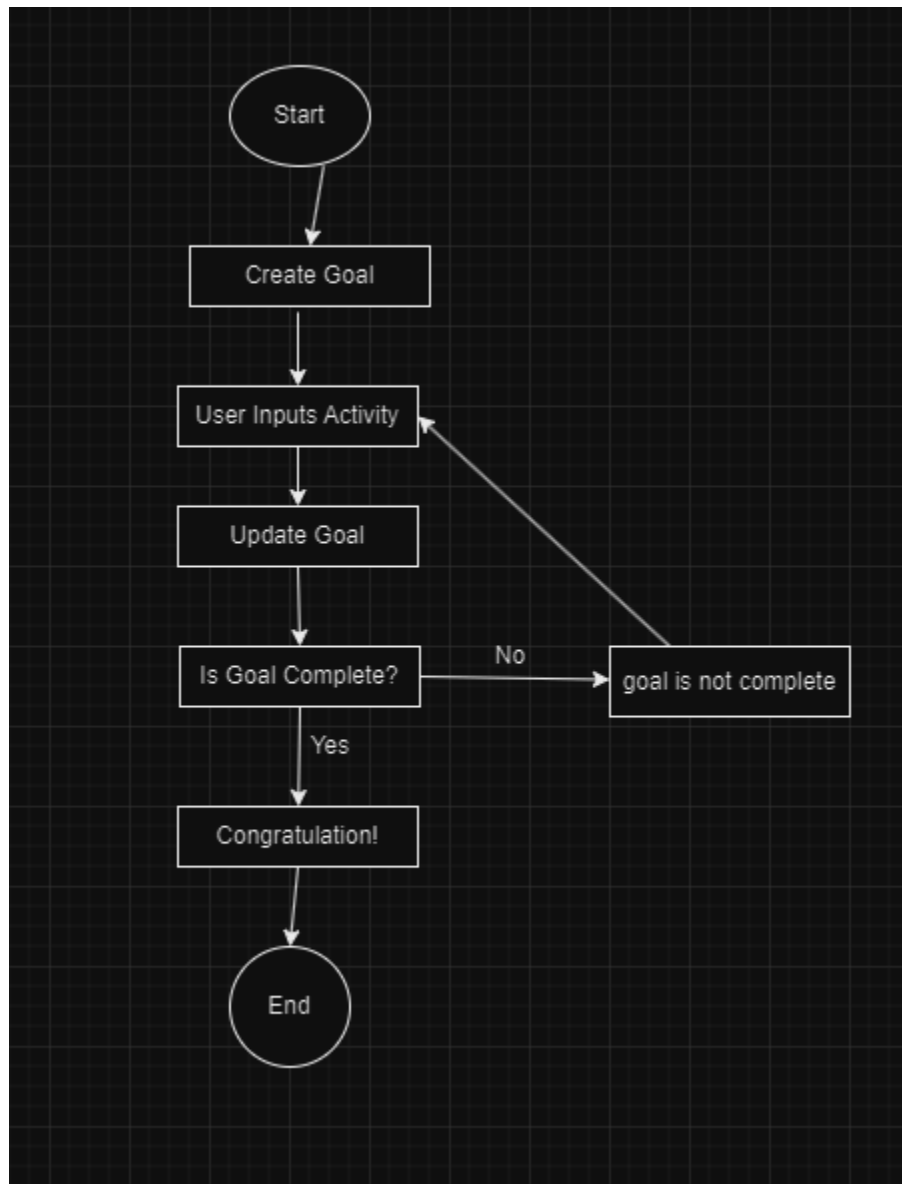
In the component diagram the user interface interacts with the authentication in order to login either as a Guest, General User , or trainer. Then Authentication will interact based on what the user has logged into that being a personal trainer (manage session), Health Data (General Users accessing their data and possible personal trainers), and limited Guest data (for guest to access their temporary data).

Activity Categories: Case Diagram (Austin Mitchell):



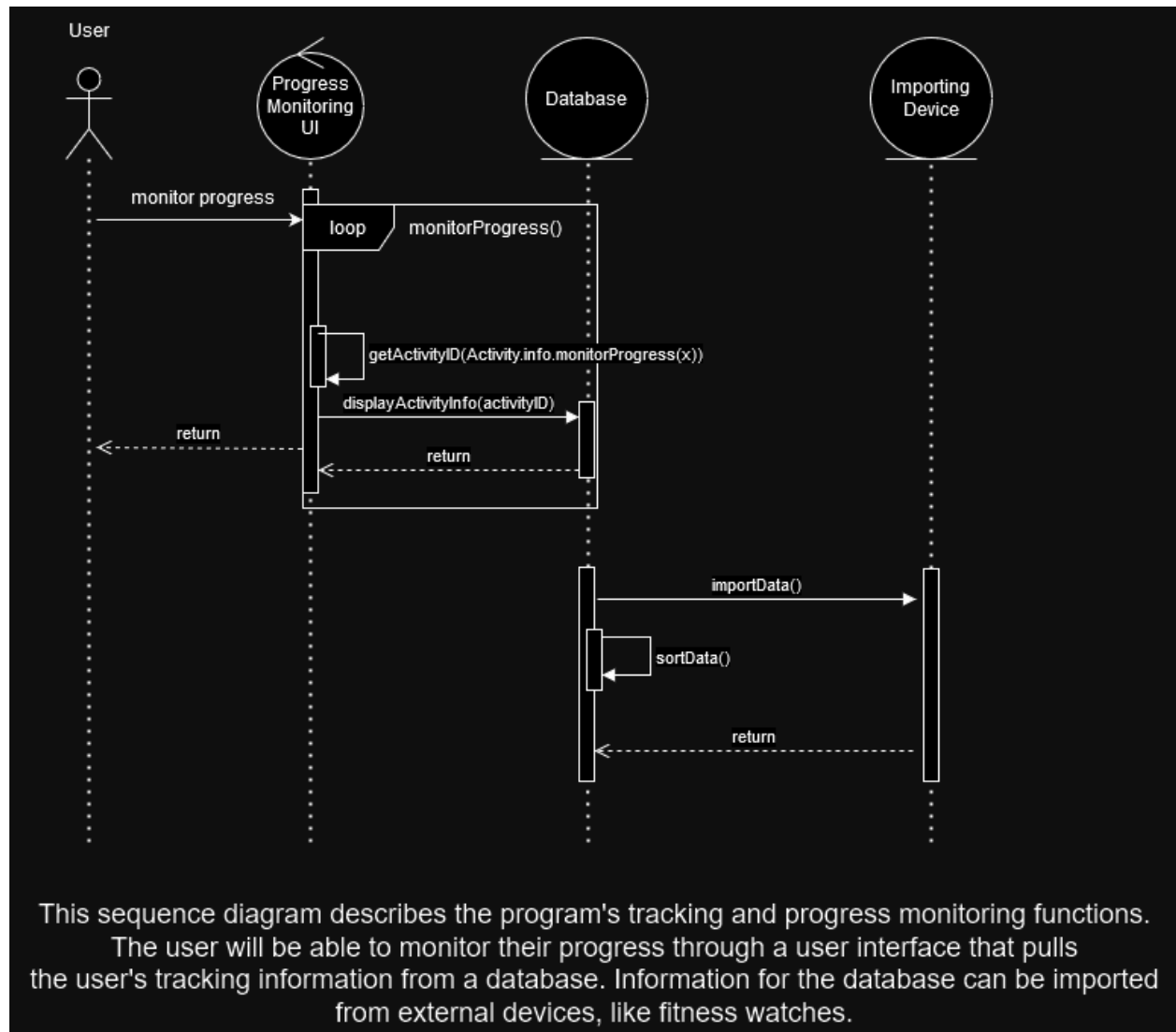
In this diagram personal trainer can access workout plans and create workout plans while General Users can also access workout plans but get to perform and make activities.

Goal Setting: Activity Diagram (Austin Mitchell):

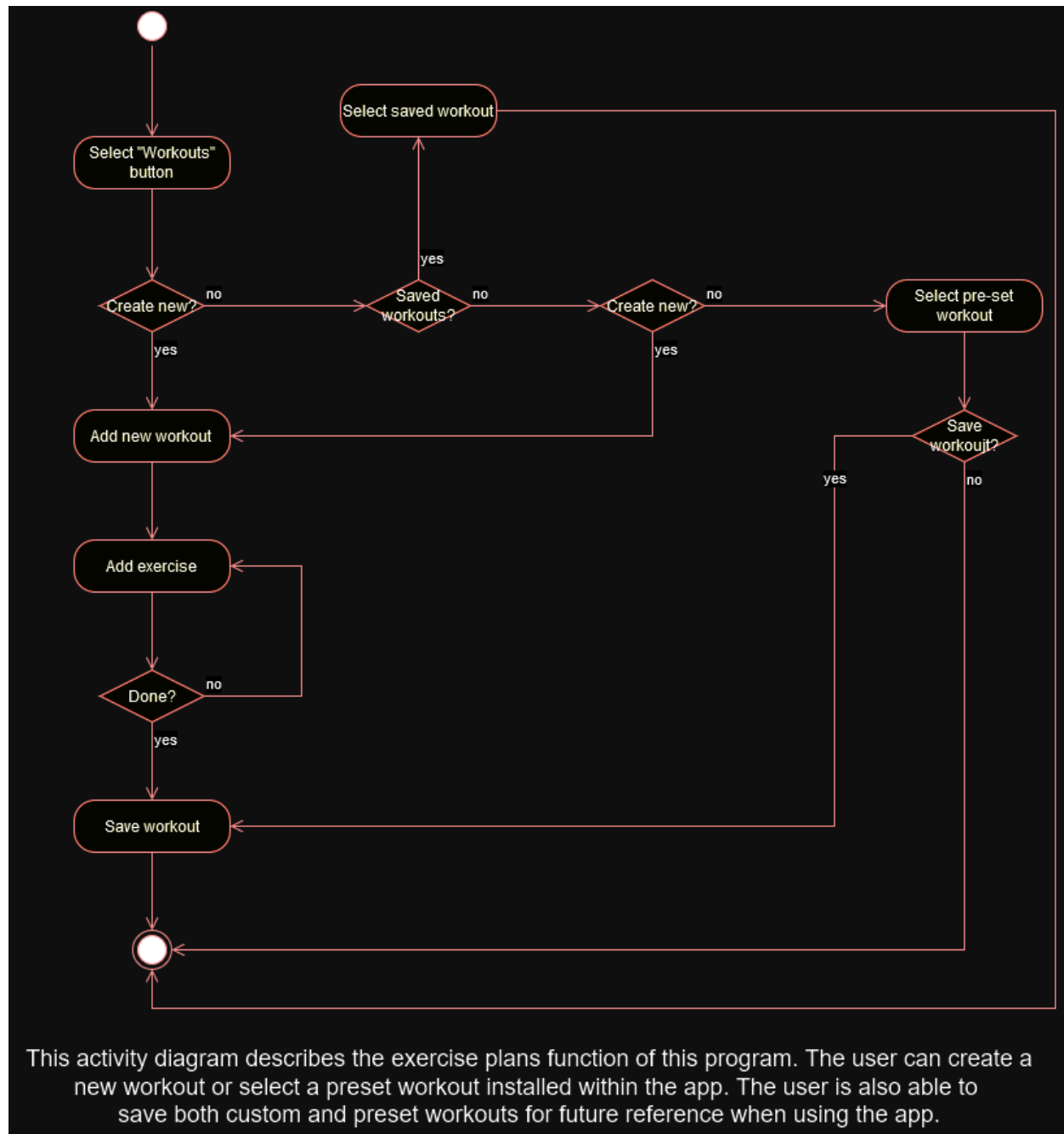


In this activity diagram the user first makes a Goal that is then continuously updated when they input their activities and is checked whether it was completed or not. If it is not completed it will wait for the next update and will continue the process until the goal is reached and ends.

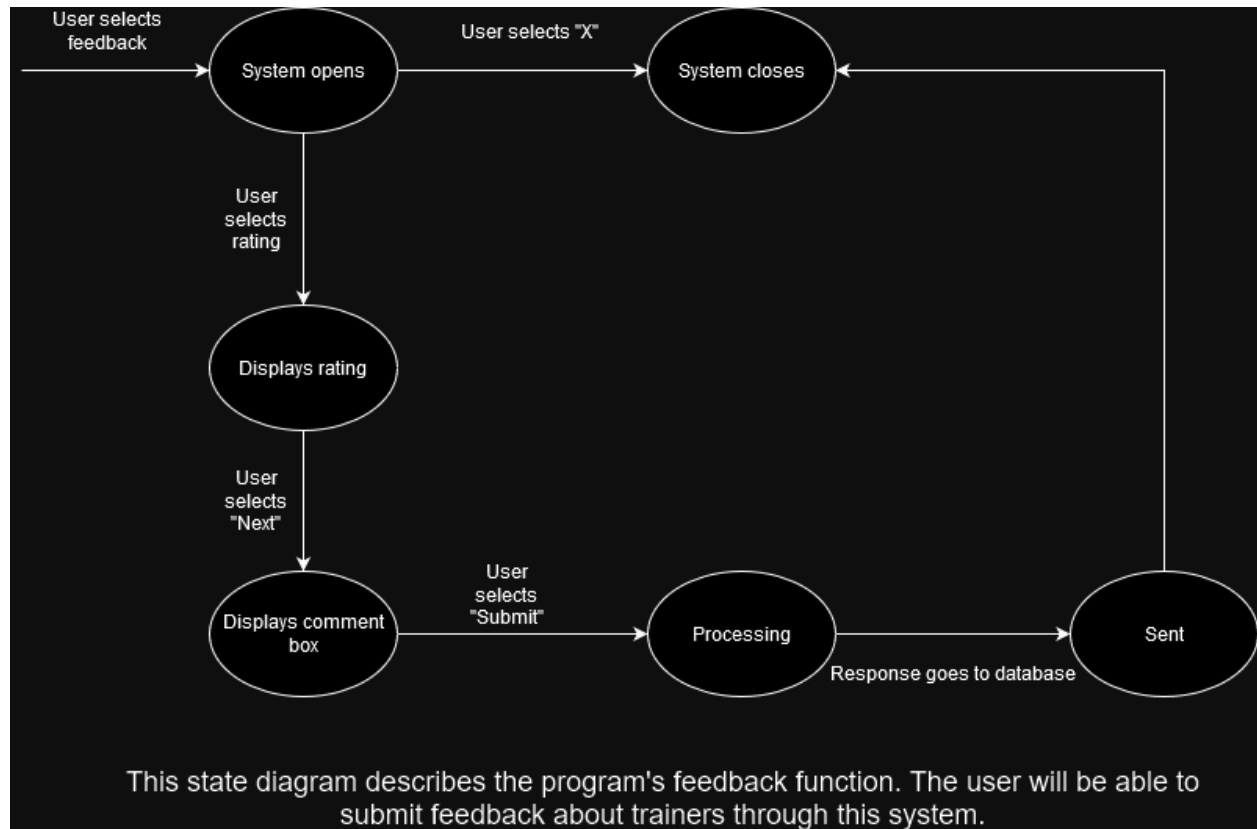
Tracking and Progress Monitoring: Sequence Diagram:



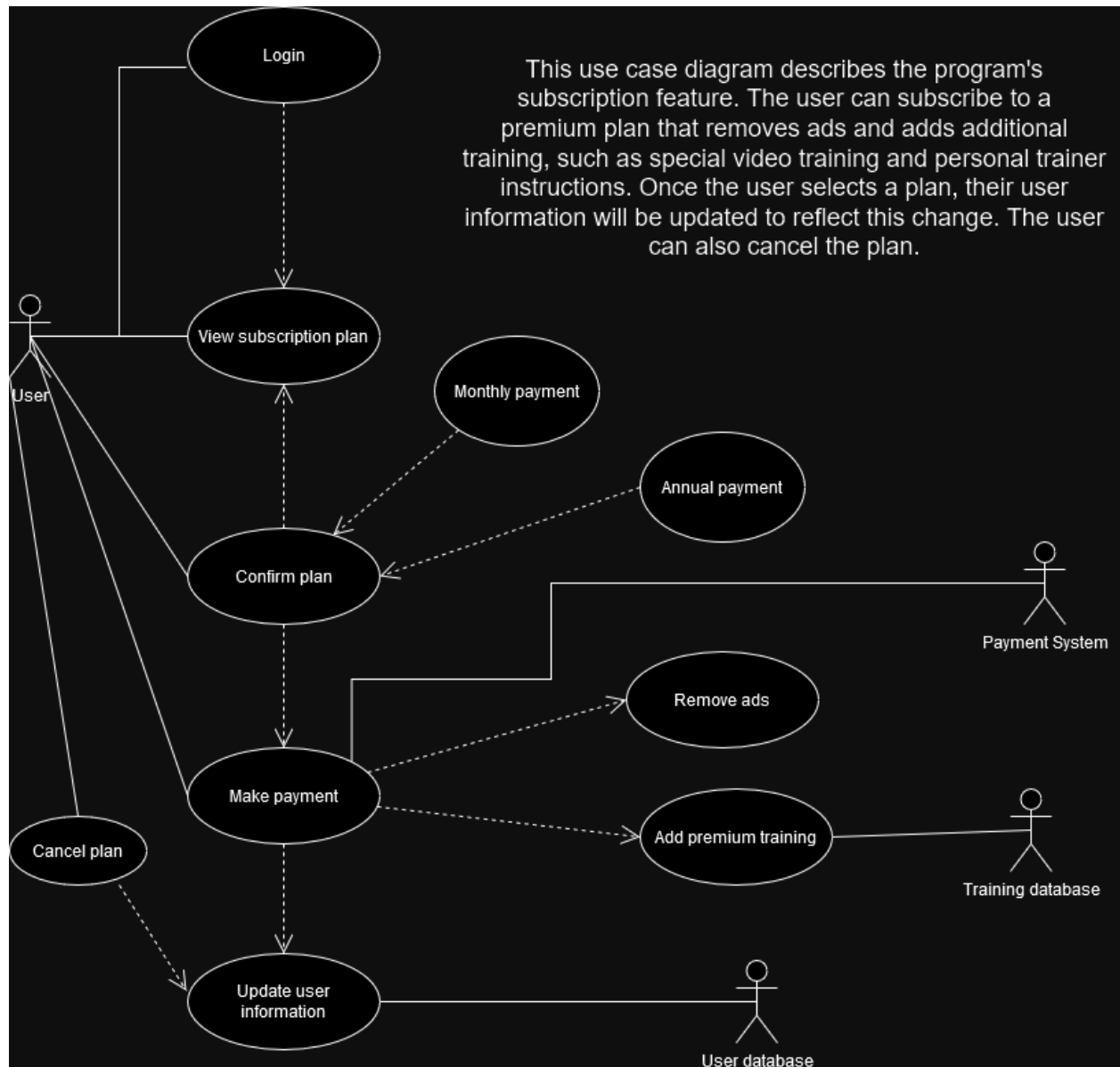
Exercise Plans: Activity Diagram:



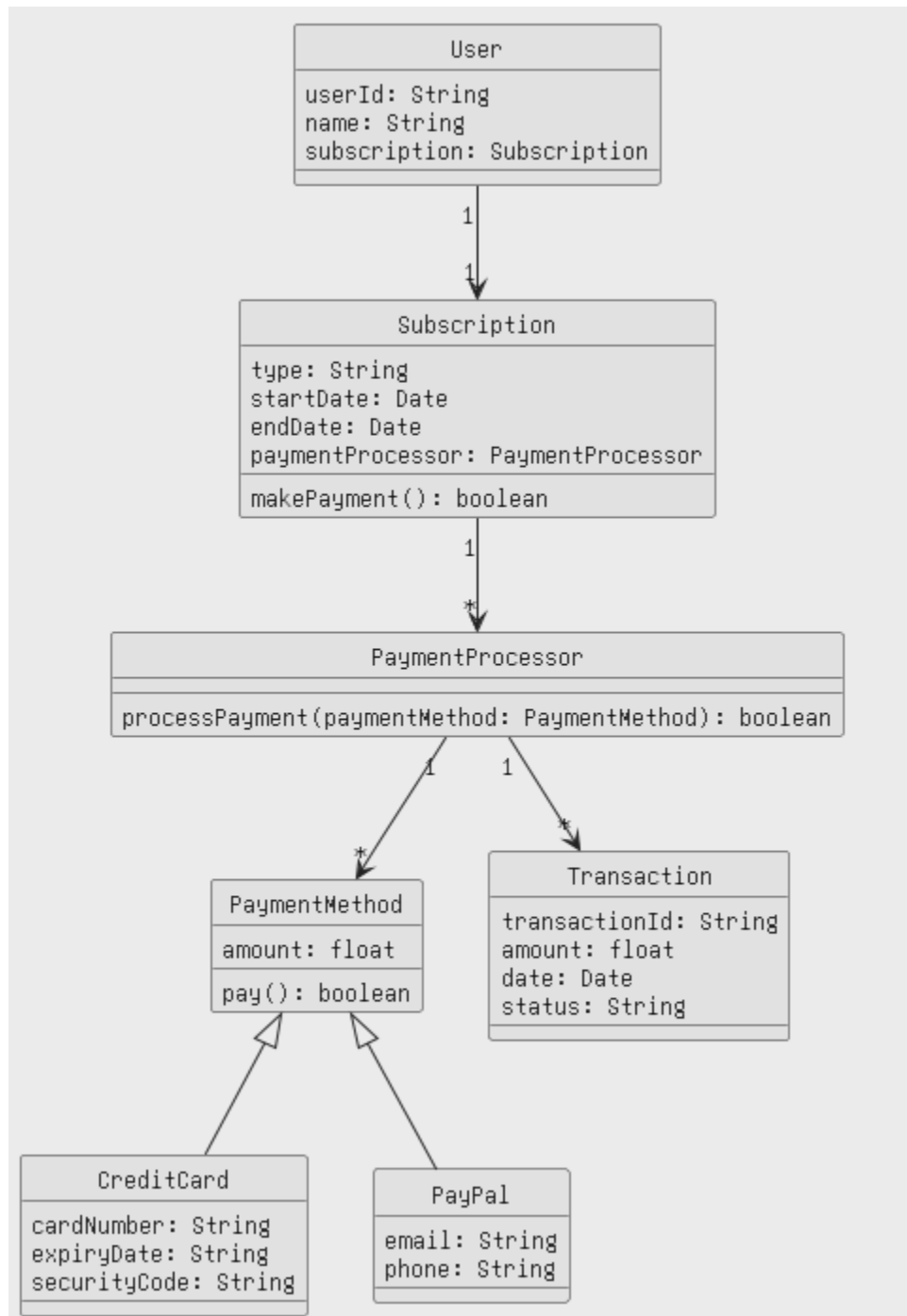
Review of Trainer: State Diagram:



Subscription: Use Case Diagram:

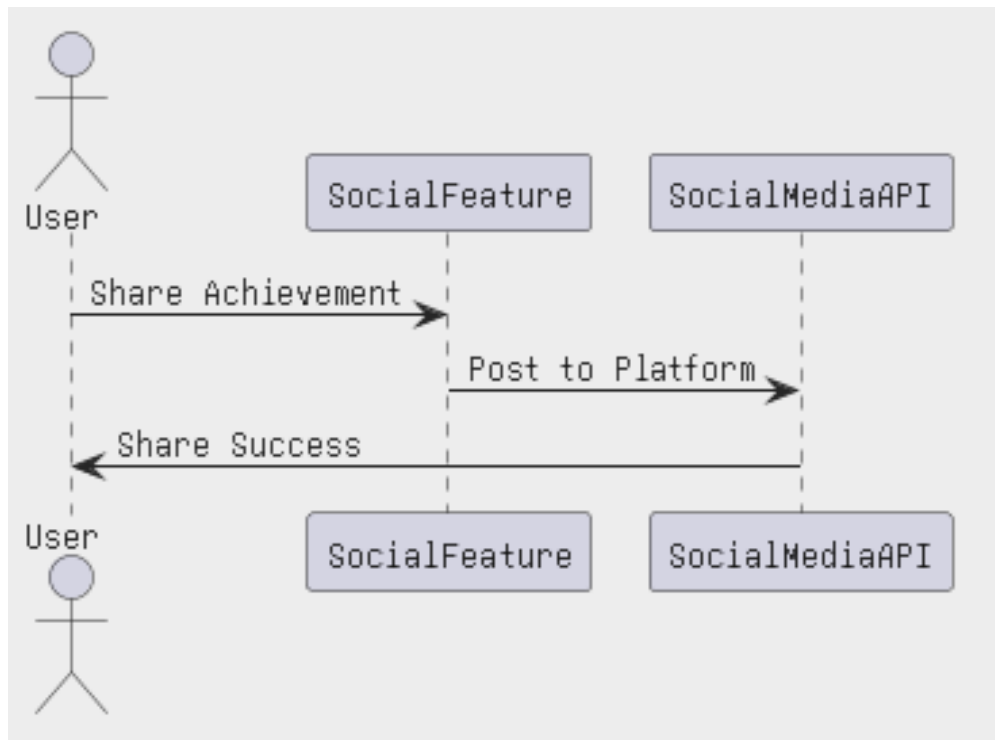


Payment Integration: Class Diagram (Thien Nguyen):



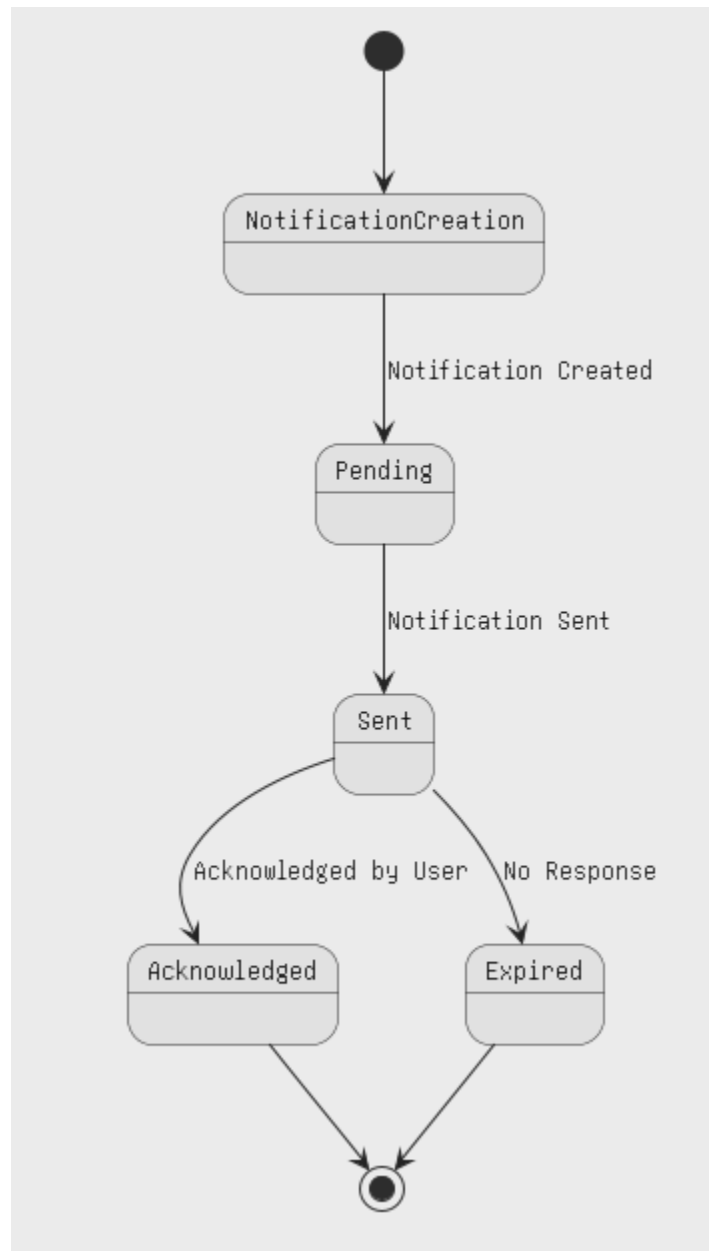
This class diagram describes payment integration that allow users to purchase personalized workout plans or hire virtual trainers via in-app payments. Payment method should support at least: debit/credit card, and PayPal.

Social Feature: Sequence Diagram (Thien Nguyen):



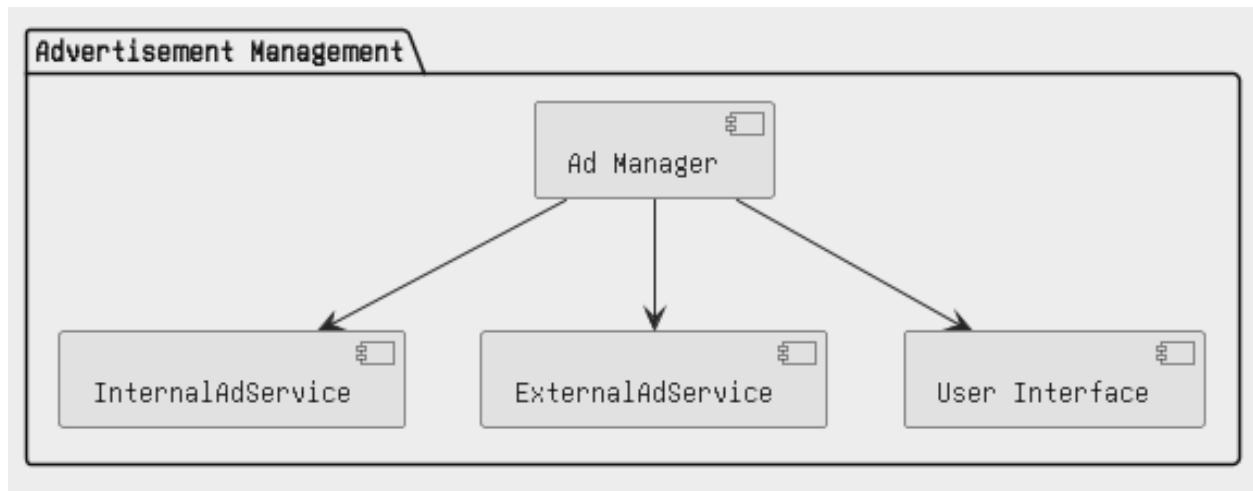
This is social feature function using Sequence Diagram. Users can share their fitness achievements, compete in challenges with friends, and post updates on social media platforms via the one-button-share function.

Communication: State Diagram (Thien Nguyen):



This state diagram describes the communication. Requires the user to provide email and text notifications for activity reminders, progress updates, and social interactions.

Advertisement: Component Diagram (Thien Nguyen):

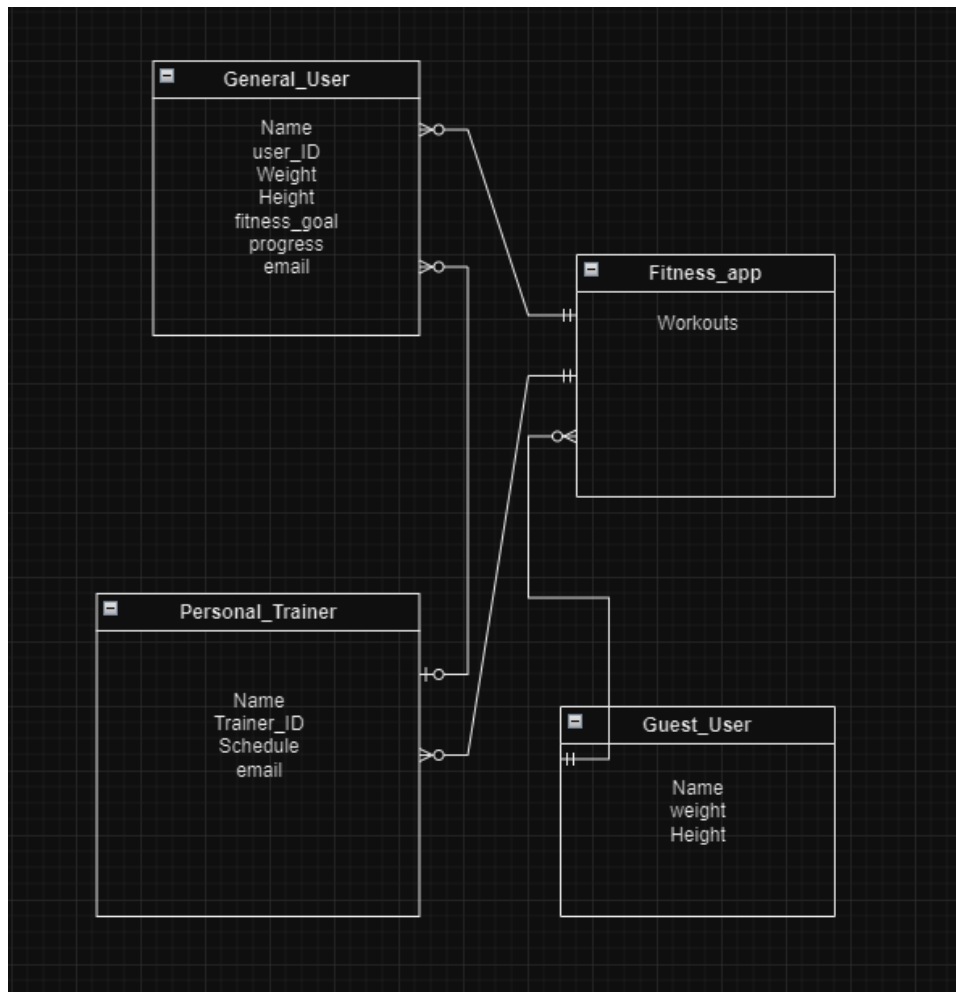


This is an advertisement using a component diagram to describe. In this function, we have internal ads and external ads along with the user interface. Ad Manager connects to both internal and external ad services to fetch and display ads.

- Internal: Advertise workout plans, personal trainer program, and fitness equipment within the app.
- External: Allow health-related businesses to advertise (e.g., local gyms, health food brands).

Conceptual Data Model

There will be a database that we will use SQLite to access and store data. Android Studio will be used to create and test the program.



Test Cases

Components to be Tested

The following components will be tested to ensure the quality and functionality of our application:

- Registration
- Login
- Tracking
- Activity categories
- Goal setting
- Progress monitoring
- Exercise plans
- Review of trainer
- Subscription
- Payment integration
- Social media sharing
- Communication
- Advertisement

Assumptions and Anomalies

The following includes a list of assumptions that are made when performing test cases for each component:

- All users are over the age of 18
- International shipping is ignored
- Cross-platform compatibility issues for wearable devices are ignored
- Legal issues are ignored
- All personal trainers are presumed to have passed their background check
- Debit/credit card validation is assumed

The following includes a list of anomalies in our test plan:

-

Test Cases: “Login and Registration”

Project Name: Personal Fitness and Health Tracker
Test Case Name: Registration
Test Case ID: CSE 3310/Team3/Registration

Test Case	Test Case Description	Expected Results	Outcome (Pass/Fail)
TC1	Enter the login screen and press “Register new user”	The system should display and collect the information below and establish a new user ID and password: <ul style="list-style-type: none">- First and last name- Email address- Phone number- User ID (at least 8 characters, may not include any special characters)- Password (at least 8 characters, must include a number and a special character)- Answer to security question for password reset	pass
TC2	Check for duplicate registration	If a user is already registered, the system should prevent the user from registering and bring them to the login screen.	pass
TC3	Give subscription options	After registering, offer the user to subscribe for more features or to choose no subscription.	pass
TC4	Display option to register	Guest Users should be taken to a registering screen after trying to select options that require being logged into for or that require a subscription.	pass
TC5	Give option to log out of account	Users should have an option to log out of the account that is currently logged in.	pass

Project Name: Personal Fitness and Health Tracker
Test Case Name: Login
Test Case ID: CSE 3310/Team3/Login

Test Case	Test Case Description	Expected Results	Outcome (Pass/Fail)
TC1	Enter the login screen and successfully login with correct user ID and password	The system should bring the user to the application's home screen.	pass
TC2	Enter the login screen and unsuccessfully login with incorrect user ID and/or password	The system should display the message "Incorrect login, please try again." to the user and allow them to re enter their login.	pass
TC3	Forgot ID or Password	The system should bring the user to a screen that asks them to select which part of their login they forgot. The system should also collect the user's email and the answer to the security question so that a temporary password can be sent to allow the user to reset their login.	pass
TC4	Create account	When at the login screen the system should allow the option to create an account and upload their credentials to be stored as a new account that can be logged into.	pass
TC5	Guest User	When at the login screen allow the user the option to continue as guest where they will be restricted to guest material or activities and no data will be stored for them.	pass

Test Cases: “Tracking”

Project Name: Personal Fitness and Health Tracker

Test Case Name: Tracking

Test Case ID: CSE 3310/Team3/Tracking

Test Case	Test Case Description	Expected Results	Outcome (Pass/Fail)
TC1	Import data from wearable device	Data from a user’s wearable device, such as an Apple watch or a Samsung Galaxy watch must be reflected in the user’s fitness metrics.	Pass
TC2	Data update in database	Whenever a user completes a workout or imports data from a wearable device, the data for the related metrics must be updated automatically in the database.	Fail
TC3	Display what devices are compatible	Display all devices that are implemented into the system to track and store data.	Fail
TC4	Add new device	Allow the user to add devices that will be tracked and to also manage them.	Fail
TC5	Delete device	When a user needs to stop updating the database off their device, allow the option to stop tracking the device and remove it from the database.	Fail

Test Cases: “Activity Categories”

Project Name: Personal Fitness and Health Tracker

Test Case Name: Activity Categories

Test Case ID: CSE 3310/Team3/ActivityCategories

Test Case	Test Case Description	Expected Results	Outcome (Pass/Fail)
TC1	Select an activity category	When the user selects an activity category, the system should display a list of activities that fall under that category for the user to participate in	Fail
TC2	Start an activity	When the user selects the activity they would like to participate in, the system should display	Fail

		the activity page, along with instructions and a timer to record how long the user is doing this activity.	
TC3	Log activity completion	If the user wants to log their activity, the system should update the data in the database with the related metrics and display this information to the user. This information can also be displayed in relation to the user's goals.	Fail
TC4	View history	The system should display a list of activities the user has logged in the past, along with the metrics the user had at the time of the activity.	Fail
TC5	Subscription restrictions	System should recognize whether the user is subscribed to access Activities or not.	Fail

Test Cases: "Goal Setting"

Project Name: Personal Fitness and Health Tracker

Test Case Name: Goal Setting

Test Case ID: CSE 3310/Team3/GoalSetting

Test Case	Test Case Description	Expected Results	Outcome (Pass/Fail)
TC1	Create a new goal	The system should bring the user to a screen that will allow the user to set goals in the following: <ul style="list-style-type: none"> - # of steps - # of pounds lost/gained - # of miles - PR lifted (pounds) - # of minutes doing yoga - # of intervals doing HIIT 	Pass
TC2	View a goal	The system should bring the user to a screen that shows an existing goal the user has created. If there are no goals created, the system should display a message that says "You have no goals yet. Go and create one!"	Pass
TC3	Edit a goal	The system should allow the user to change the metric of the goal (the number of steps, the number of miles, etc.) and/or the type of goal (steps instead of miles, etc.).	Pass

TC4	Goal has been reached	The system should mark the goal complete and prompt the user if they would like to start a new goal. If the user is not in the app, the system should also send a notification to the user that their goal has been completed.	Fail
TC5	Checking subscription	The system should check whether the user is allowed to access Goal setting based on their subscription or if they are a general user or guest user.	Pass

Test Cases: “Progress Monitoring”

Project Name: Personal Fitness and Health Tracker
Test Case Name: Progress Monitoring
Test Case ID: CSE 3310/Team3/Progress Monitoring

Test Case	Test Case Description	Expected Results	Outcome (Pass/Fail)
TC1	Display progress toward goals	The system should display a screen that shows a graph or chart measuring the user’s progress toward the goal(s) they’ve set in the application.	Pass
TC2	A goal remains incomplete	The system should send a notification to the user showing that their goal is incomplete. The system should allow the user to set a new reminder time to receive another notification, dismiss the reminder, or snooze the reminder.	Fail
TC3	Request a reminder	The system should send a notification at the time specified by the user.	Fail
TC4	No goals have been set	The system should display a message that says “You have no goals yet. Go and create one!”	Fail
TC5	Option to delete a goal	Give the user the option to delete a goal when the user desires to and update the database when chosen.	Pass

Test Cases: “Exercise Plans”

Project Name: Personal Fitness and Health Tracker
Test Case Name: Exercise Plans
Test Case ID: CSE 3310/Team3/ExercisePlans

Test Case	Test Case Description	Expected Results	Outcome (Pass/Fail)
TC1	View workout plans available	The system should display a screen that shows a list of workout plans that the user can choose from.	pass
TC2	Select a workout plan	The system should bring the user to a screen that allows them to view each exercise within the workout plan. The user should also be able to select a button that allows them to start the plan.	pass

TC3	Create a workout plan	The system should bring the user to a screen that prompts the following information: <ul style="list-style-type: none"> - Number of exercises - Type of exercises - # of reps, miles, intervals, etc. for each exercise - Name for workout plan 	TBD
TC4	Start plan	The system should bring the user to the first exercise in the plan with instructions on how to complete the exercise. The user will be able to mark the exercise as complete once done and the system should update its database accordingly with the user's metrics to enable progress monitoring.	TBD
TC5	User does not have a subscription	The system should display a message that says "This is a subscription feature. You must have a subscription plan to access this feature."	Fail
TC6	Edit a plan	The system should bring the user to a screen that allows them to change the following information: <ul style="list-style-type: none"> - Number of exercises - Type of exercises - # of reps, miles, intervals, etc. for each exercise - Name of workout plan 	Pass
TC7	User starts a plan, does not complete	The system should send a notification reminding the user that their plan remains incomplete.	Fail
TC8	Share a plan	The system will display a screen that allows the user to directly post to Instagram, X, Facebook, or copy a link to send to friends/family via messaging.	pass

Test Cases: "Subscriptions"

Project Name: Personal Fitness and Health Tracker
Test Case Name: Subscriptions
Test Case ID: CSE 3310/Team3/Subscriptions

Test	Test Case	Expected Results	Outcome
------	-----------	------------------	---------

Case	Description		(Pass/Fail)
TC1	Display the subscription details(or multiple subscription details)	The system should bring the user to a screen that will allow the user to view each of the subscription options or just one that displays the prices and benefits.	pass
TC2	Choosing a subscription	After the user has chosen the subscription they want, the system should take them to a screen to enter their payment information to be stored and charged when the subscription is renewed.	pass
TC3	Charging for the subscription	The system should find which subscription was chosen by the user and when the subscription will be renewed or if the subscription has been canceled. Then charge the amount needed to the card when renewed or singed up.	Pass
TC4	Canceling subscription	There should be an option within account setting to manage your subscription displaying your current subscription and when it will be renewed. There will also be an option to cancel to subscription on the screen when selected will ask if the user is sure they wanna cancel. After a subscription is canceled the data stored should state that there is no subscription and no more charges will be made to the card for the subscription.	Fail
TC5	Changing payment method	The system should have an option for the user to change the payment type and once entered will update the stored payment type to the new card.	Fail

Test Cases: “Review of Trainer”

Project Name: Personal Fitness and Health Tracker
Test Case Name: Review of Trainer
Test Case ID: CSE 3310/Team3/Subscriptions

Test Case	Test Case Description	Expected Results	Outcome (Pass/Fail)
TC1	Users submit	The system should save the review with the	Pass

	a rating and comment for a trainer	specified rating (e.g., 1-5 stars) and user comment in the trainer's profile section. The new review should be immediately visible to all users viewing that trainer's profile and any cached versions of the trainer's reviews are updated.	
TC2	Users update their previous review	The system should allow the user to retrieve and edit their existing review details (rating and comment). Upon resubmission, the previous review is replaced with the updated version in the database, and all visible instances of the review reflect the changes. The system should confirm successful updating, and the edit should be recorded in the user's activity log.	fail
TC3	Users deleted their submitted review	The review should be permanently removed from the trainer's profile and from any user-generated content section. The system should display a confirmation prompt before deletion and a success message once the review is deleted.	fail
TC4	Users view all reviews for a specific trainer	The app should retrieve and display a list of all reviews associated with the selected trainer. Reviews should load in order of most recent first, and each review should display the rating, comment, and username. The display should allow for sorting by rating and filtering by date or type of review, providing a user-friendly view of the trainer's feedback.	fail
TC5	Users submit a rating without comments	The system should accept and store the rating in the trainer's profile without requiring a text comment. The rating should immediately contribute to the trainer's overall average rating, and the trainer's profile should refresh to reflect the new rating. The system should confirm the rating submission was successful.	Pass

Test Cases: "Social Media Sharing"

Project Name: Personal Fitness and Health Tracker
Test Case Name: Social Media Sharing
Test Case ID: CSE 3310/Team3/Subscriptions

Test Case	Test Case Description	Expected Results	Outcome (Pass/Fail)
TC1	Users share a fitness achievement on social media	The system provides sharing options for various social media platforms. After the user selects a platform, the app automatically formats the achievement data, including text and image content. Upon completion, the app confirms the successful share and returns to the app's main screen	Fail
TC2	Users share their workout plans with friends via social media	The app allows the user to select from their workout plans and prepares a formatted link or image. The system transfers this data to the chosen social media platform, including an optional message. If successfully posted, the app confirms and logs the sharing event in the user's history	Fail
TC3	Users view shared achievements by friends	The app displays a social feed of friends' recent achievements. Achievements include details such as activity type, duration, and goals met. The user can like, comment, or share these posts within the app or on external platforms.	Fail
TC4	The user previews an achievement or workout plan post before sharing on social media.	The system generates a preview of the post, showing text, images, and any links or tags. The preview allows the user to make adjustments before posting. After confirming, the user can share directly on the selected platform, with the post appearing exactly as previewed. The app confirms a successful share or allows the user to return and edit.	Fail
TC5	The user disconnects their social media accounts from the app.	The app removes all social media connections, revoking permissions. A success message confirms disconnection, and social media sharing buttons are removed from the app. All previous posts shared on social media remain unaffected.	Pass

Test Cases: “Communication”

Project Name: Personal Fitness and Health Tracker
Test Case Name: Social Media Sharing
Test Case ID: CSE 3310/Team3/Subscriptions

Test Case	Test Case Description	Expected Results	Outcome (Pass/Fail)
TC1	System sends a reminder for the user's scheduled activity	The user receives a notification based on their chosen method (email/SMS/app alert) at the designated time. The notification displays a friendly message reminding the user to complete their activity. If tapped, the notification opens the app's activity section for immediate tracking	fail
TC2	System sends a weekly progress summary	At the end of the week, a notification or email provides a breakdown of the user's progress, highlighting key metrics like calories burned, steps taken, and goals achieved. This summary includes a link to view more details within the app, giving users an overall progress snapshot.	fail
TC3	User enables or disables notifications in settings	Changes are saved immediately, and the user receives a confirmation prompt. If notifications are enabled, reminders and progress updates are sent as scheduled. If disabled, the user receives no further alerts until they choose to re-enable notifications.	Pass
TC4	System sends notifications about friend activities	The user receives notifications for friends' milestones or shared activities. Each notification provides quick access to view or comment on the achievement within the app.	fail
TC5	User sets a custom reminder for a specific goal	The system allows the user to specify the day and time for reminders. At the chosen time, the app sends a notification, helping the user stay on track with their goal. The reminder is logged, and the user can update or delete it as needed.	fail

Test Cases: "Advertisement"

Project Name: Personal Fitness and Health Tracker
Test Case Name: Social Media Sharing
Test Case ID: CSE 3310/Team3/Subscriptions

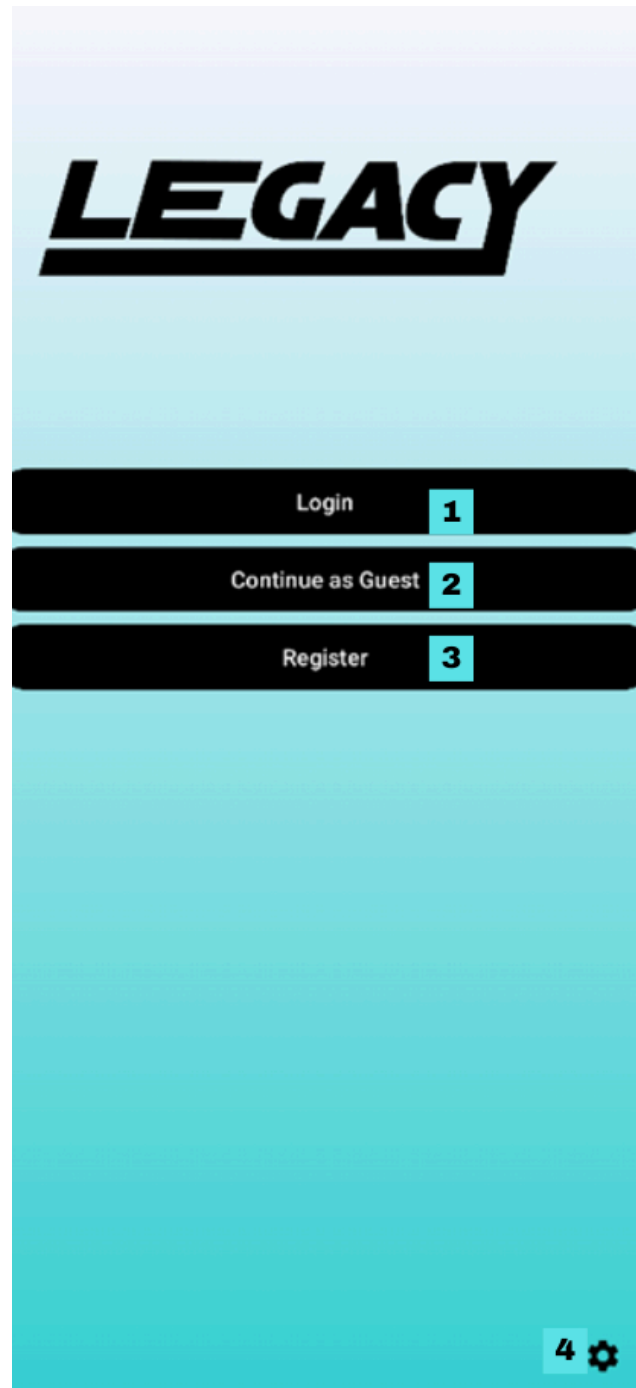
Test Case	Test Case Description	Expected Results	Outcome (Pass/Fail)
------------------	------------------------------	-------------------------	----------------------------

TC1	Ad is shown to users without an active subscription	The ad appears in a designated area, such as between workout sessions or on the home screen. Ads are relevant to health and fitness products, and users can dismiss or click the ad to learn more. Premium users do not see these ads.	Pass
TC2	User clicks on an ad leading to an external product	The ad opens a browser window with the external product's webpage. Users are shown an alert before exiting the app. The ad interaction is recorded for analytics, and the user can return seamlessly to the app.	Pass
TC3	Ad for a premium workout plan is displayed	The ad appears in the app, showcasing premium workout options. If the user clicks, they're directed to the in-app purchase page. If uninterested, the user can dismiss the ad and continue using the app.	Pass
TC4	App tracks user interaction with ads	Each interaction (click, dismiss) is recorded in the analytics database, providing insights on ad effectiveness. Summary data is displayed in the admin section for review, supporting targeted advertising.	Fail
TC5	User closes an ad without interacting	The ad disappears without affecting the user's activity flow. The system registers the dismissal action, tracking engagement without disrupting the user experience.	Fail

User Manual

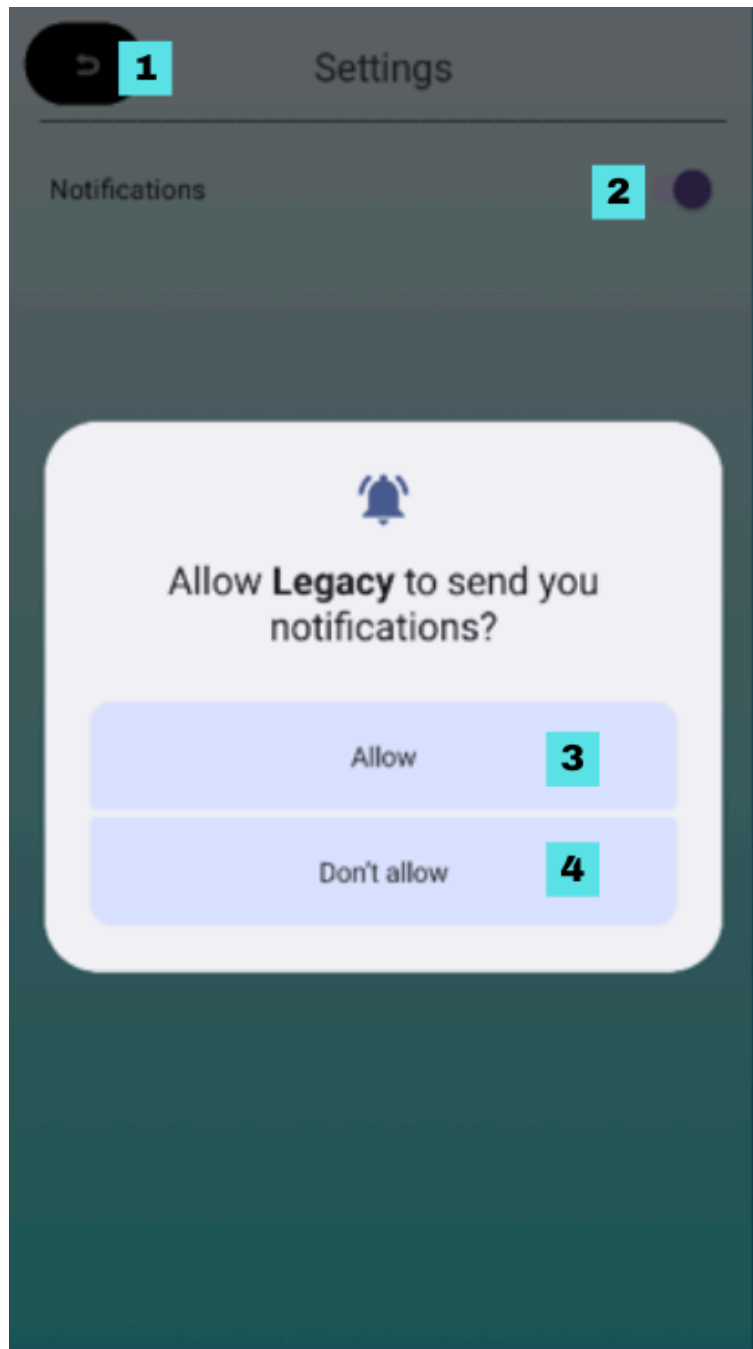
Main Page

1. Login button will navigate user to login page.
2. Continue as guest will navigate user to a home page that design only for user without subscription.
3. Register will take user to registration page.
4. Setting icon will take users to setting page.



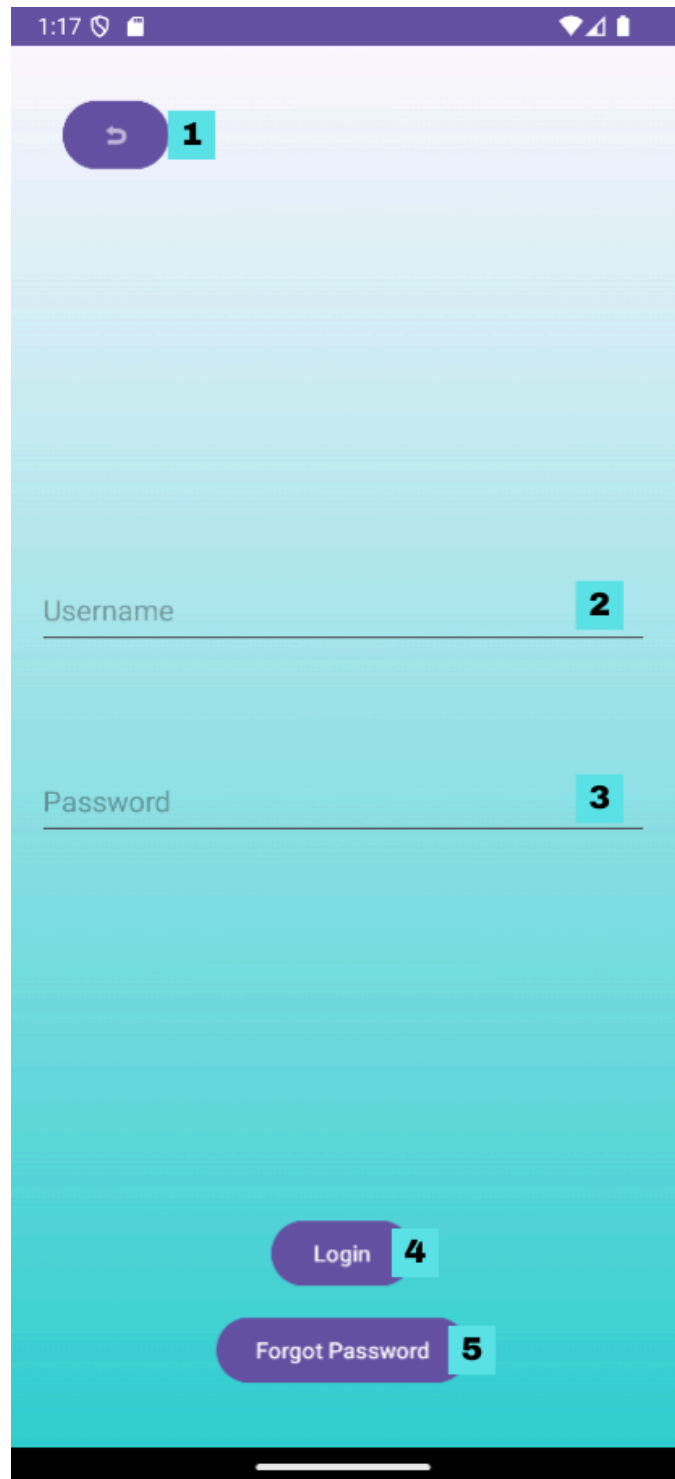
Settings (Notifications)

1. Back button will take the user back to the main page.
2. Notification switch. If turned on, the system will prompt and ask for permission for the application to send notification.
3. Enable permissions for notifications in the system.
4. Don't enable permissions for notifications in the system.



Login Page

1. Back button will navigate the user to the previous page.
2. Users enter their username.
3. Users enter their password.
4. When the user presses the login button, the system will check if the username and password match in the local database. If yes, it will navigate the user to the home page, if not then it will prompt “Account does not exist”.
5. Forgot password will navigate users to forgot password page.



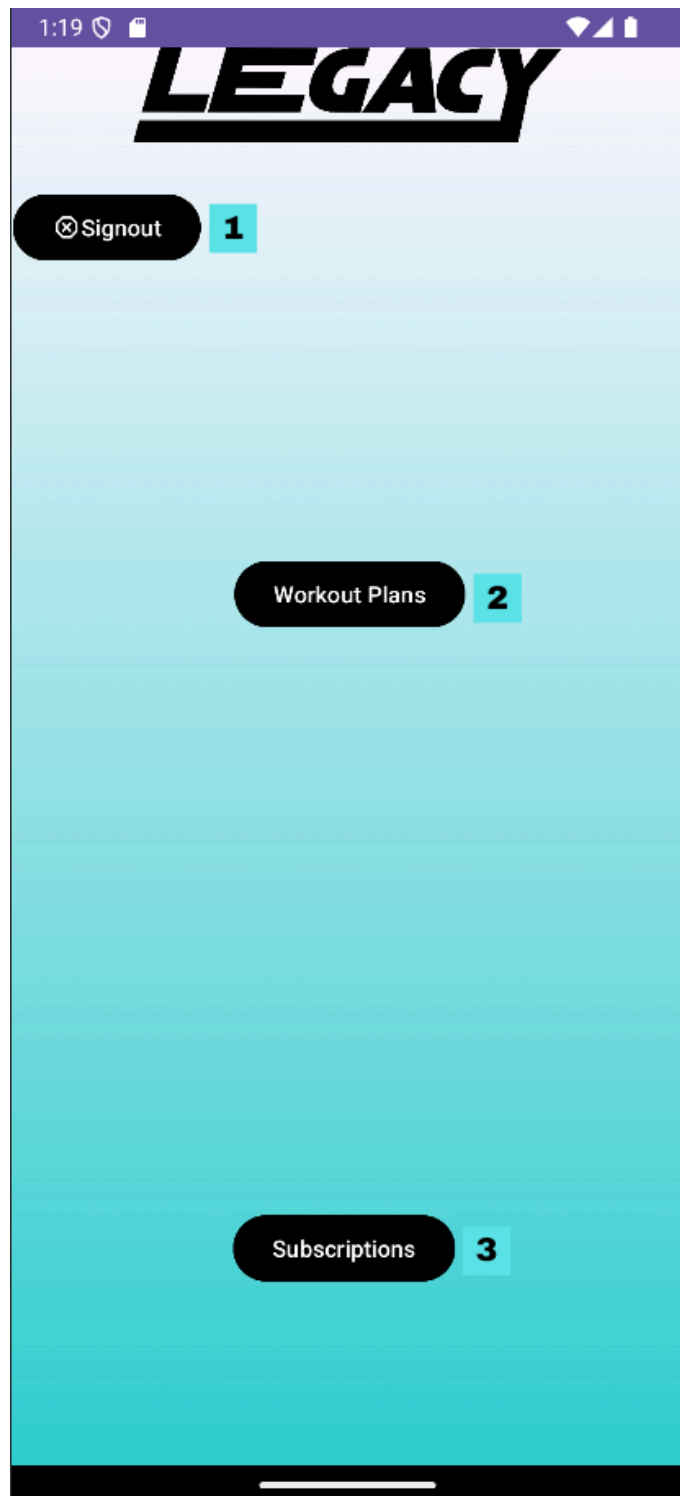
Forgot Password Page

1. Back button will navigate users to previous page.
2. Users will enter their username.
3. Users will enter their password.
4. Users will select their security question.
5. Users will enter their security answer.
6. If the user press submit button, then the system will navigate users back to login page and validate the information. If the information match with local database, the system will send an email with reset password link. (this is just the concept, not actual email sent out)

The screenshot shows a mobile application interface for a 'Forgot Password' page. The page has a light purple background. At the top, there is a dark purple header bar with the time '2:12' and status icons. Below the header, there is a back button (a dark purple circle with a white arrow) labeled with a red '1'. The main form area contains four input fields, each with a red number in a yellow box to its right: 'Enter username' (2), 'Enter your email' (3), 'Select Security Question' (4), and 'Answer security question' (5). At the bottom of the form is a large, rounded dark purple button labeled 'Submit' with a red '6' in a yellow box to its right. The bottom of the screen shows a black home indicator bar.

Continue as Guest Page

1. Sign out button will take users back to main page.
2. Workout Plans will take users to work out plans page.
3. Subscriptions will take users to subscriptions page.



Workout Plans Page

1. Back button will take users to the previous page.
2. Lists of available pre-set workout plans for users to choose.
3. Make plans buttons will navigate users to make plans page where users can make their custom plan.



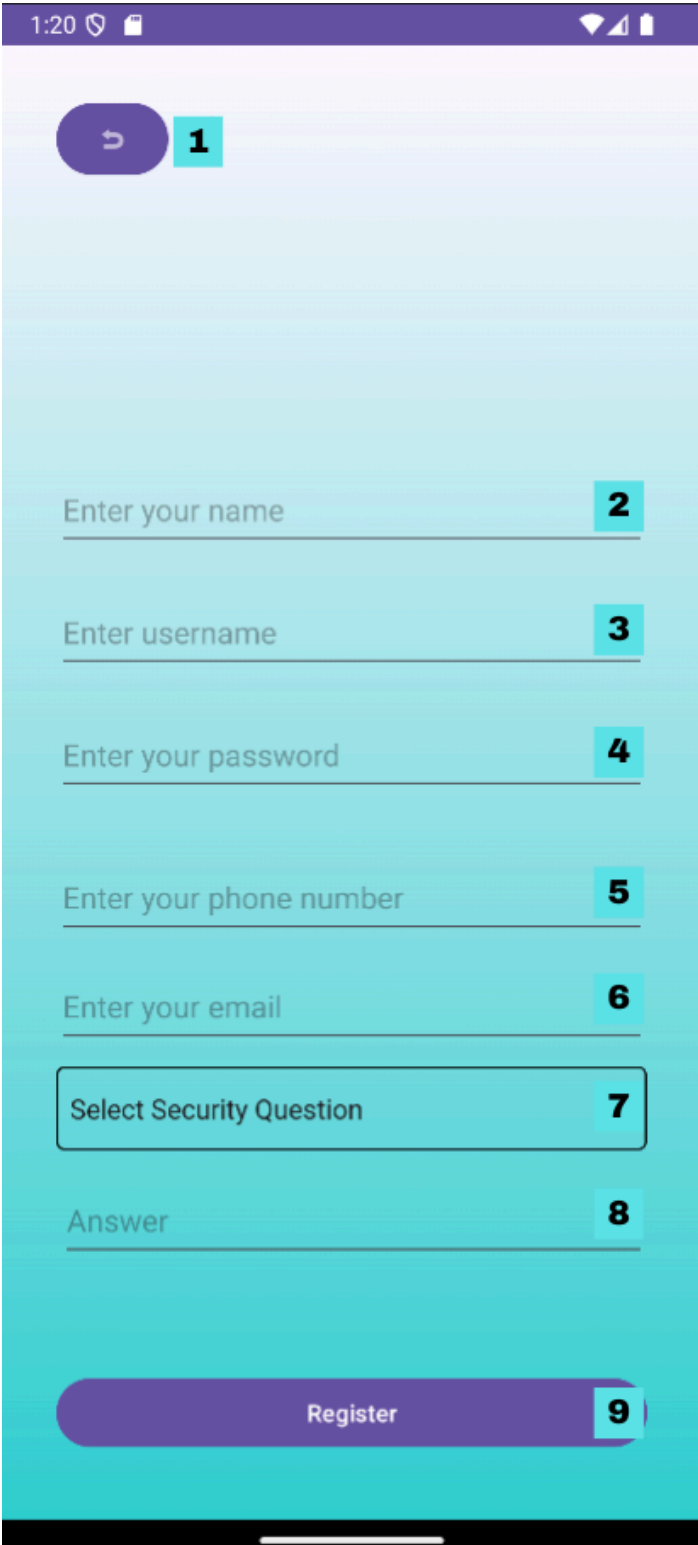
Make Plans Page

1. Allows users to put their plan title.
2. Descriptions of the custom plan.
3. Cancel button will cancel the action and navigate users to previous page at the same time.
4. Complete button will save the plans to local database and appear in Workout Plans page.

The screenshot shows a mobile application interface titled "Make Your Plan". At the top, there is a status bar with the time 1:26 and various icons. Below the title, there is a text input field with the placeholder text "type your title for your plan here". A red square with the number "1" is placed at the end of this input field. Below the input field is a large rectangular area with a light blue gradient background, intended for the plan description. A red square with the number "2" is centered within this area. At the bottom of the screen, there are two buttons: "Cancel" and "Complete". A red square with the number "3" is placed next to the "Cancel" button, and a red square with the number "4" is placed next to the "Complete" button.

Registration Page

1. Back button will take users to previous page.
2. Enter user's name.
3. Enter user's username.
4. Enter user's password.
5. Enter user's phone number.
6. Enter user's email address.
7. Users choose their security question.
8. Enter user's answer for security question.
9. When user press register button, the system will validate all the input such as password needs to have at least 8 letters, 1 special letters, email needs to have @, etc. If all the requirements are met, then the system will save the credentials in local database and navigate users to login page.

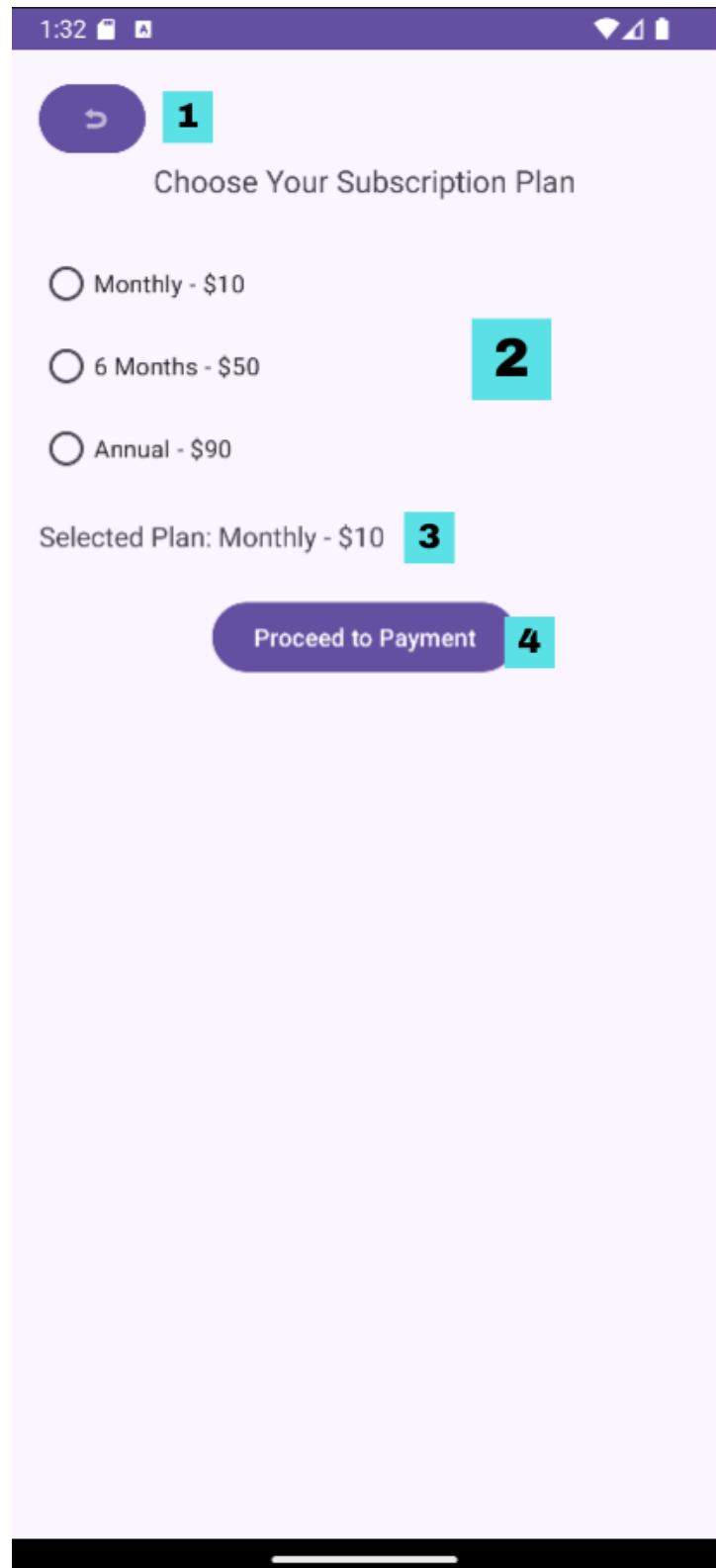


A mobile application registration page with a teal background. At the top is a purple header bar with a back arrow icon and a red square with the number 1. Below the header are nine input fields, each with a red square containing a number from 2 to 9. The fields are: 'Enter your name', 'Enter username', 'Enter your password', 'Enter your phone number', 'Enter your email', 'Select Security Question', 'Answer', and a large purple 'Register' button. The status bar at the top shows the time 1:20 and various icons.

Number	Field Label
1	Back button
2	Enter your name
3	Enter username
4	Enter your password
5	Enter your phone number
6	Enter your email
7	Select Security Question
8	Answer
9	Register

Subscriptions Page

1. Back button will take users to previous page.
2. List of available subscription plans and price
3. The selected plans will change according to user's choice of plans.
4. Proceed to payment will take users to payment page.



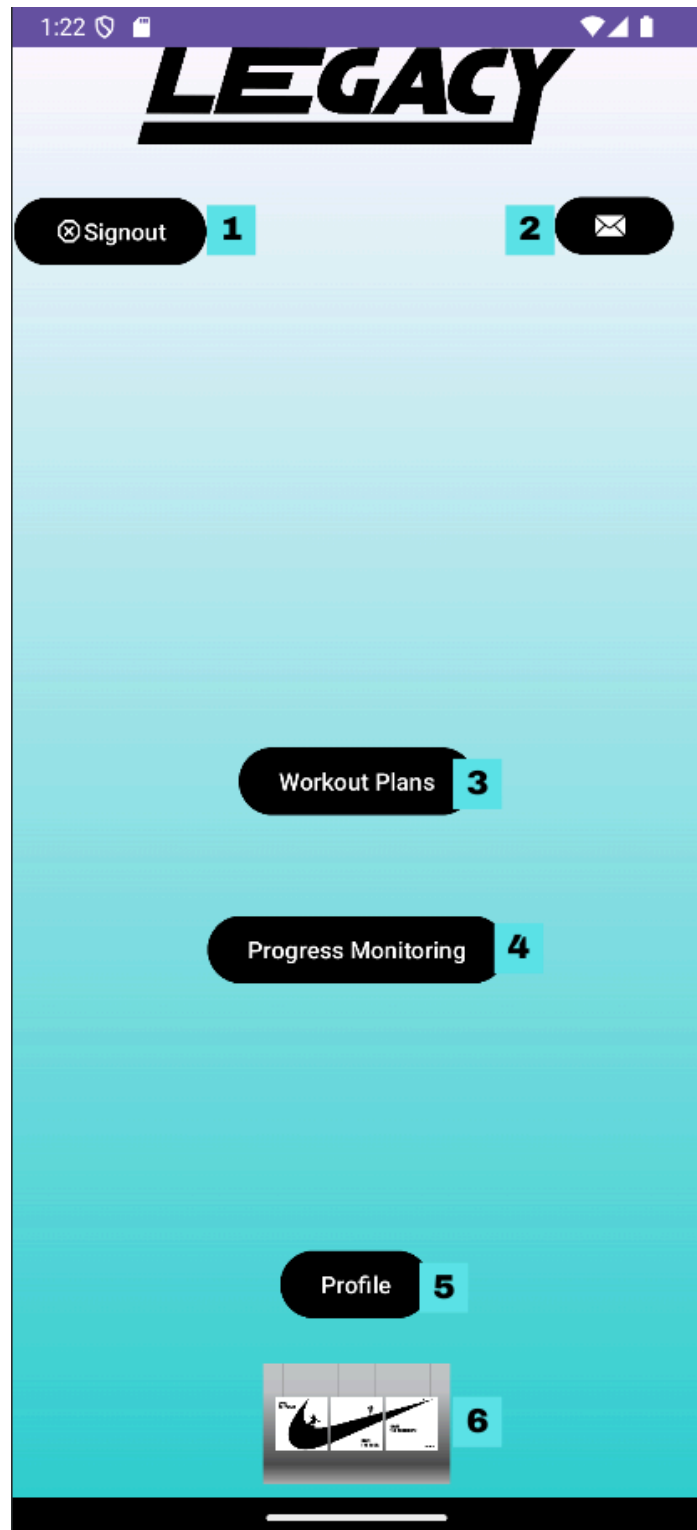
Payment Page

1. Back button will take users to previous page.
2. Users enter card number.
3. Users enter expiration date.
4. Users enter CVV.
5. The price of user's choice of plan will appear here.
6. When user press submit payment, the system will validate all the input, card number needs to have 16 digits, expiration date needs to have 4 digits and a "/", CVV needs to have 3 digits. After all the requirements are met, the system will navigate users to home page.

The image shows a mobile application interface for a payment page. At the top, there is a status bar with the time 1:33 and icons for signal, Wi-Fi, and battery. Below the status bar is a purple header bar containing a back arrow icon and a red square with the number 1. The main content area has a light purple background and the title "Enter Payment Details" in bold black text. Below the title are three input fields, each with a red square containing a number: "Enter card number" (2), "Expiration Date (MM/YY)" (3), and "CVV" (4). Below these fields is the text "Selected Plan: Annual - \$90" next to a red square with the number 5. At the bottom is a large purple button labeled "Submit Payment" next to a red square with the number 6. The bottom of the screen shows a black home indicator bar.

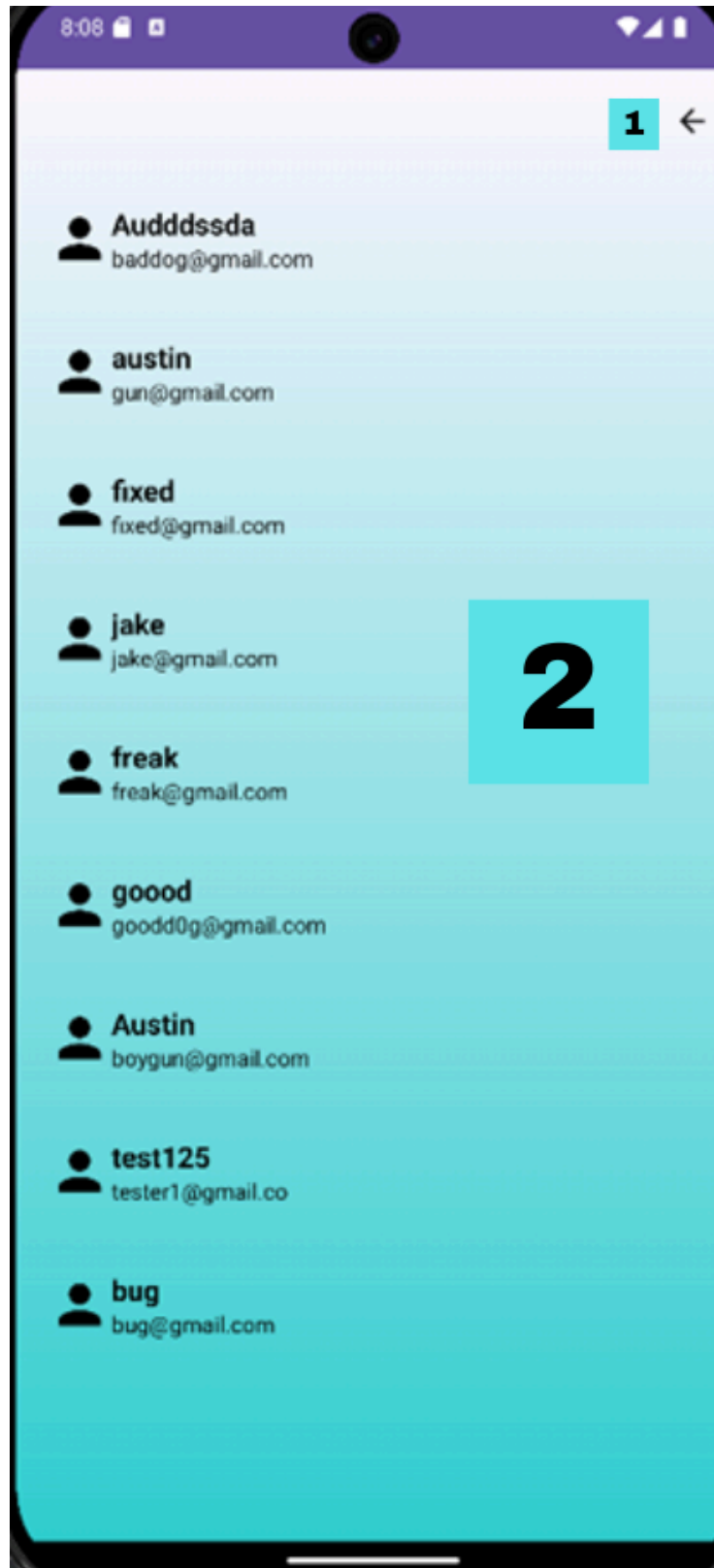
Home page

1. Sign out will take users back to the main page.
2. Mail icon button will take users to chat.
3. Workout plans button will take users to the workout plans page.
4. Progress Monitoring will navigate users to the Progress Monitoring page.
5. Profile button will take users to the profile page.
6. Picture of advertisement, the advertisement will randomly rotate every time users access the home page. If users press the image, it will take users to the system browser with the link associated with the picture.



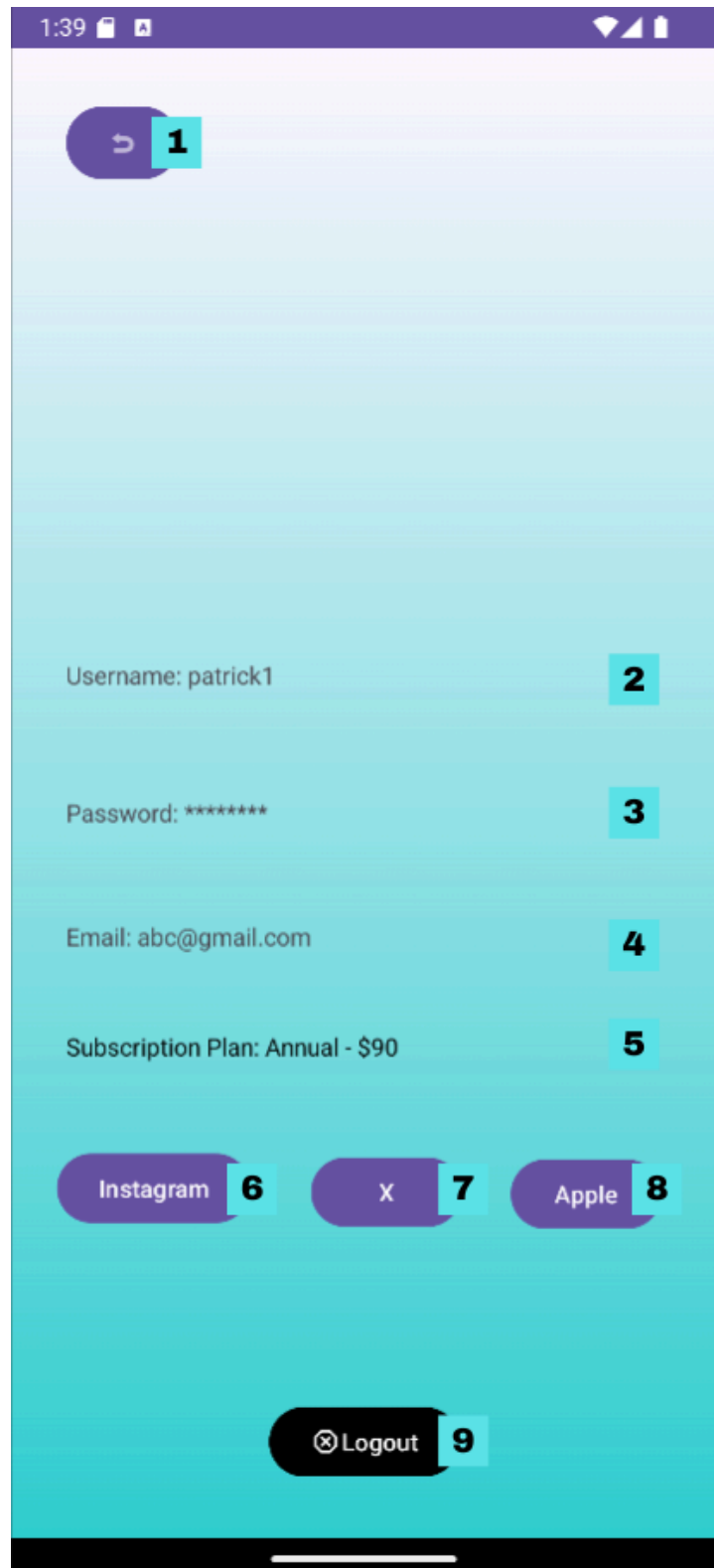
Chat Page

1. Back icon button will take users back to the previous page.
2. List of users you can chat with.



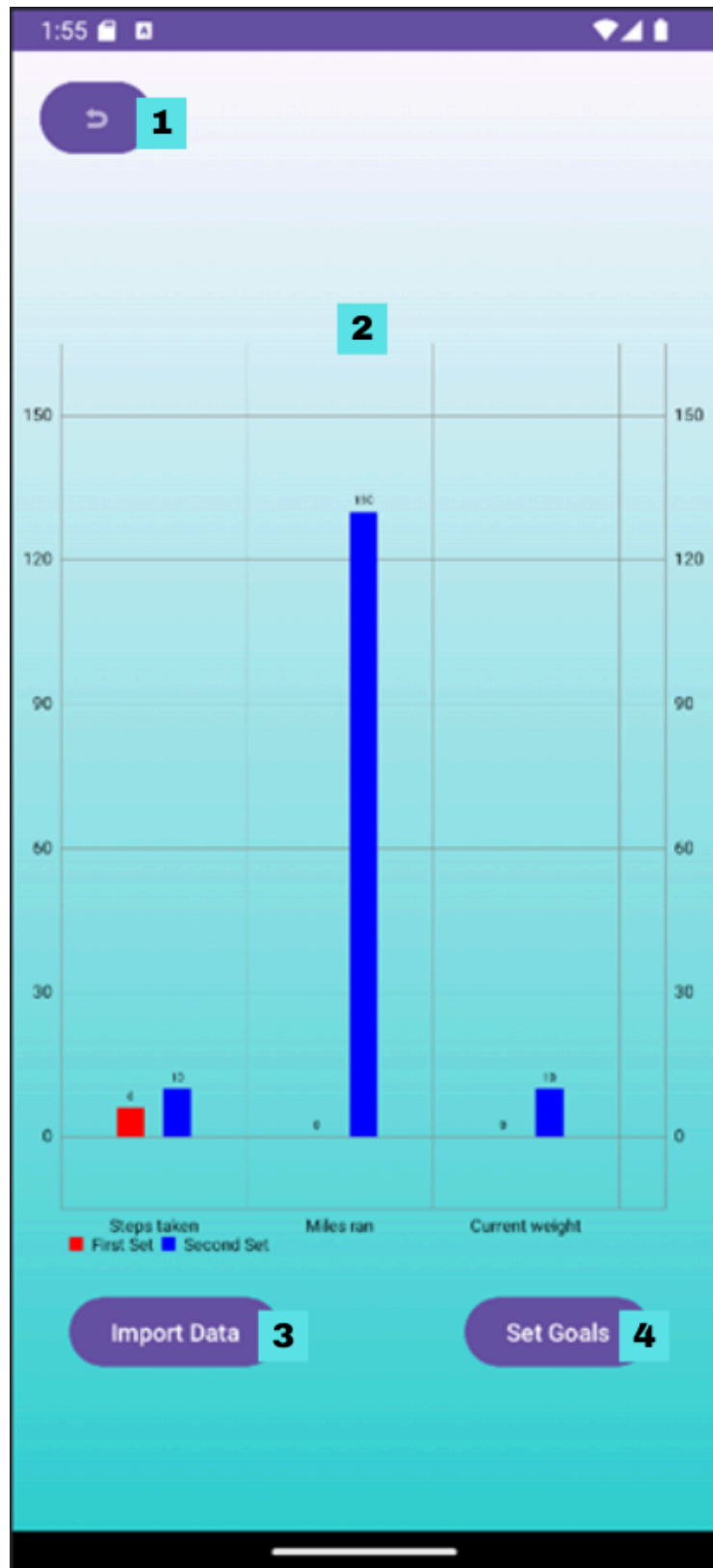
Profile Page

1. Back button will navigate users to the previous page.
2. Username will appear here.
3. Password will appear here with an asterisk symbol for security reasons.
4. Email will appear here.
5. Subscription plan will appear here.
6. Instagram link account. If users press this button, it will take users to the browser for Instagram to link their Instagram account.
7. X (Twitter) link account. If users press this button, it will take users to the browser for X (Twitter) to link their account.
8. Apple Fitness link account. If users press this button, it will take user to browser for Apple Fitness to link their Apple Fitness account
9. Sign out button will log out and take users to the home page.



Progress Monitoring Page.

1. Back button will take user to previous page.
2. Diagram reflects data of progress and goals that user put in.
3. Import Data button will take user to import page where users can input their data of progress.
4. Set Goals button will take users to import page where users can input their data of goals.



Import Data Page.

1. Back button will take users to previous page.
2. Submit button will record all the data and take users back to Progress Monitor page and the diagram will reflect the data of Progress.

1:57

1

Steps taken 10

Miles ran 150

Minutes of yoga 30

Intervals of HIIT 5

New PR 200

Weight 150

Submit 2

Set Goals Page.

1. Back button will take users to the previous page.
2. Submit button will record all the data and take users back to Progress Monitoring page and the diagram will reflect data of the Goals column.

1:41

1

Steps taken 6

Miles ran 5

Minutes of yoga 4

Intervals of HIIT 3

New PR 130

Weight 120

Submit 2

Assumptions and Constraints

Assumptions

The following is a list of assumptions:

- Assume all users over the age of 18 (for under 18 clients, an adult guardian is needed).
- Ignore any international shipping for users.
- Ignore any cross-platform wearable devices compatible issues.
- Ignore any legal issues, assume all personal trainers pass background checks.
- No need to do real debit/credit card validation, just make sure the Credit card number is 16 digits, Card Expiration date has the format “mmyy” and the Security code is 3 digits.
- Ignore email validation.
- Registration is for general users at the moment.

Constraints

The following is a list of constraints:

- Team lacks Android development skills
- Limited engagement of some team members
- Team lacks Android device to test implementation of system

Out of Scope Material

The following is a list of “out of scope” material:

- Post Project maintenance is not covered
- Payment verification is not covered
- Social media implementation is not covered
- Importation of data from external systems is not covered

Delivery and Schedule

{List all tasks/milestones from start of the project to the end with specific dates for both Anticipated Start & End Dates. Status includes Complete, In Progress, and To Be Completed (TBC)}

Task Description	Anticipated Start Date	Anticipated End Date	Status	Comments
Prepare UML diagrams	09/20/2024	10/01/2024	Complete	Deliverable UML document
SRA document (Includes project objectives, Requirements and UML diagrams)	10/02/2024	10/22/2024	Complete	Deliverable will be the SRA document. All stakeholders agree on the content of the SRA by signing in section 8.

Application Homepage design	11/01/2024	11/16/2024	Complete	Design of the homepage will be complete
Login and Registration	11/01/2024	12/03/2024	Complete	User can register as well as login
Displaying progress on a goal	11/01/2024	12/03/2024	Complete	Users should be able to see their progress on their goals
Workout Plans with creating plans	11/01/2024	12/03/2024	Complete	Users should be able to see and add to their workout plans
Implementing Graphs for the user	11/01/2024	12/03/2024	Complete	Should display users progress in graph format
Setting up notifications	11/01/2024	12/03/2024	Complete	User should be able to receive notifications as well as disable them
Display Advertising	11/01/2024	12/03/2024	Complete	Allow for advertising on the home page
Test Plan Delivery	10/20/2024	10/29/2024	Complete	Deliverable will be the Test plan document.
Final Report Delivery	11/26/2024	12/03/2024	In-Progress	Deliverable will be the final report plus Team Presentation

Stakeholder Approval Form

Stakeholder Name	Stakeholder Role	Stakeholder Comments	Approval Signature	Signature Date
Chenxi Wang	Client			
Pujan Budhathoki	Client Project Manager			
Austin Mitchell	Developer		<i>Austin Mitchell</i>	10/21/2024
Thien Nguyen	Developer		<i>Thien Nguyen</i>	10/21/2024
Emma Slonaker	Developer		<i>Emma Slonaker</i>	10/21/2024
Huy Nguyen	Developer		<i>Huy Nguyen</i>	10/21/2024
Raza Inthanongsack	Developer		<i>Raza Inthanongsack</i>	10/21/2024

Source Code (Selected/Main)

Database / part of data class

```
public void onCreate(SQLiteDatabase db) {  
  
    // Existing users table creation  
  
    String createUsersTable = "CREATE TABLE " + TABLE_NAME + " (" +  
        COL_USERID + " TEXT PRIMARY KEY, " +  
        COL_NAME + " TEXT, " +  
        COL_EMAIL + " TEXT, " +  
        COL_PHONE + " TEXT, " +  
        COL_PASSWORD + " TEXT, " +  
        COLUMN_SECURITY_QUESTION + " TEXT," +  
        COLUMN_SECURITY_ANSWER + " TEXT," +  
        COL_SUBSCRIPTION + " INTEGER DEFAULT 0," +  
        COLUMN_SUBSCRIPTION_PLAN + " TEXT," +  
        COL_STEPS + " INTEGER DEFAULT 0," +  
        COL_WEIGHT + " DOUBLE DEFAULT 0," +  
        COL_MILES + " DOUBLE DEFAULT 0," +  
        COL_LIFTED + " INTEGER DEFAULT 0," +  
        COL_MINUTES + " INTEGER DEFAULT 0," +  
        COL_INTERVALS + " INTEGER DEFAULT 0," +  
        COL_STEPS_GOAL + " INTEGER DEFAULT 0," +  
        COL_WEIGHT_GOAL + " DOUBLE DEFAULT 0," +  
        COL_MILES_GOAL + " DOUBLE DEFAULT 0," +  
        COL_LIFTED_GOAL + " INTEGER DEFAULT 0," +  
        COL_MINUTES_GOAL + " INTEGER DEFAULT 0," +  
        COL_INTERVALS_GOAL + " INTEGER DEFAULT 0)";
```

```

String createWorkoutPlansTable = "CREATE TABLE workout_plans (" +
    "id INTEGER PRIMARY KEY AUTOINCREMENT, " +
    "userID TEXT, " +
    "plans TEXT, " +
    "FOREIGN KEY (userID) REFERENCES " + TABLE_NAME + "(" + COL_USERID + ")";

db.execSQL(createWorkoutPlansTable);

db.execSQL(createUsersTable);
}

@Override

public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {

    db.execSQL("DROP TABLE IF EXISTS " + TABLE_NAME);

    onCreate(db);

}

public User getUser(String userID) {

    SQLiteDatabase db = this.getReadableDatabase();

    Cursor cursor = db.rawQuery("SELECT * FROM " + TABLE_NAME + " WHERE " + COL_USERID + " = ?", new String[]{userID});

    User user = null;

    if (cursor.moveToFirst()) {

        String username = cursor.getString(cursor.getColumnIndexOrThrow(COL_USERID));

```

```

        String email = cursor.getString(cursor.getColumnIndexOrThrow(COL_EMAIL));

        String subscriptionPlan =
cursor.getString(cursor.getColumnIndexOrThrow(COLUMN_SUBSCRIPTION_PLAN));

        user = new User(username, subscriptionPlan, email);
    }

    cursor.close();

    return user;
}

// Put account into database

public boolean insertUser(String name, String email, String phone, String userID, String password, String
securityQuestion, String securityAnswer) {

    SQLiteDatabase db = this.getWritableDatabase();

    ContentValues contentValues = new ContentValues();

    contentValues.put(COL_NAME, name);

    contentValues.put(COL_EMAIL, email);

    contentValues.put(COL_PHONE, phone);

    contentValues.put(COL_USERID, userID);

    contentValues.put(COL_PASSWORD, password);

    contentValues.put(COLUMN_SECURITY_QUESTION, securityQuestion);

    contentValues.put(COLUMN_SECURITY_ANSWER, securityAnswer);

    contentValues.put(COL_SUBSCRIPTION, 0);

    contentValues.put(COL_STEPS, 0);

    contentValues.put(COL_WEIGHT, 0);

    contentValues.put(COL_MILES, 0);

    contentValues.put(COL_LIFTED, 0);

```



```

contentValues.put(COL_MINUTES, 0);

contentValues.put(COL_INTERVALS, 0);


contentValues.put(COL_STEPS_GOAL, 0);
contentValues.put(COL_WEIGHT_GOAL, 0);
contentValues.put(COL_MILES_GOAL, 0);
contentValues.put(COL_LIFTED_GOAL, 0);
contentValues.put(COL_MINUTES_GOAL, 0);
contentValues.put(COL_INTERVALS_GOAL, 0);


long result = db.insert(TABLE_NAME, null, contentValues);

return result != -1;
}

```

Workout Plans

```

public class WorkoutPlans extends AppCompatActivity {

    private ArrayAdapter<String> planAdapter;
    private ArrayList<String> planList;
    private Data database;
    private String userID;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.workout_plans);
    }
}

```

```

userID = getIntent().getStringExtra("userID");

// Initialize the database object
database = new Data(this);

ListView listPlans = findViewById(R.id.list_view);

// Load plans from database
planList = loadPlansFromDatabase();

planAdapter = new ArrayAdapter<>(this, android.R.layout.simple_list_item_1, planList);
listPlans.setAdapter(planAdapter);

// Check if a new plan was passed from makePlan activity
Intent intent = getIntent();
if (intent.hasExtra("newPlan")) {
    String newPlan = intent.getStringExtra("newPlan");
    planList.add(newPlan); // Add new plan to list
    planAdapter.notifyDataSetChanged(); // Update ListView
}

listPlans.setOnItemClickListener((parent, view, position, id) -> {
    String planToDelete = planList.get(position); // Get the selected plan

    new AlertDialog.Builder(this)
        .setTitle("Delete Plan")
        .setMessage("Are you sure you want to delete this plan?")

```

```

        .setPositiveButton("Yes", (dialog, which) -> deletePlan(planToDelete))

        .setNegativeButton("No", null)

        .show();

    });
}

```

```

private ArrayList<String> loadPlansFromDatabase() {

    ArrayList<String> plans = new ArrayList<>();

    SQLiteDatabase db = database.getReadableDatabase();

```

```

    if (userID == null) {

        Log.e("WorkoutPlans", "userID is null. Cannot load plans.");

        return plans;

    }

```

```

    Cursor cursor = db.rawQuery("SELECT plans FROM workout_plans WHERE userID = ?", new
String[]{userID});

```

```

    if (cursor != null) {

        while (cursor.moveToNext()) {

            plans.add(cursor.getString(0));

        }

        cursor.close();

    }

    db.close();

    return plans;

}

```

```

public void backView(View v) {

```

```

        Intent intent = new Intent(this, Home.class);

        intent.putExtra("userID", userID);

        startActivity(intent);
    }

    public void makePlan(View v) { // Navigate to makePlan activity

        Intent intent = new Intent(this, makePlan.class);

        intent.putExtra("userID", userID);

        startActivity(intent);
    }

    public void deletePlan(String planToDelete) {

        boolean isDeleted = database.deleteWorkoutPlan(userID, planToDelete);

        if (isDeleted) {

            planList.remove(planToDelete);

            planAdapter.notifyDataSetChanged();

            Toast.makeText(this, "Plan deleted successfully!", Toast.LENGTH_SHORT).show();

        } else {

            Toast.makeText(this, "Error deleting plan", Toast.LENGTH_SHORT).show();

        }

    }
}

```

Registration

```

public class Registration extends AppCompatActivity {

    private EditText name, editTextUsername, editTextPassword, editTextPhone, editTextEmail;

    private Button buttonRegister, back;

```

```

private Data dbHelper;

private TextView passwordFeedback;


private Spinner SecurityQuestion;

private EditText editTextSecurityAnswer;

DatabaseReference databaseReference;


@SuppressLint("MissingInflatedId")
@Override
protected void onCreate(Bundle savedInstanceState) {

    super.onCreate(savedInstanceState);

    setContentView(R.layout.registration);


    // Database

    dbHelper =new Data(this);


    SecurityQuestion = findViewById(R.id.SecurityQuestion);

    editTextSecurityAnswer = findViewById(R.id.editTextSecurityAnswer);

    name = findViewById(R.id.name);

    editTextUsername = findViewById(R.id.editTextUsername);

    editTextPassword = findViewById(R.id.password);

    editTextPhone = findViewById(R.id.Phone);

    editTextEmail = findViewById(R.id.emailField);

    buttonRegister = findViewById(R.id.buttonRegister);

    passwordFeedback = findViewById(R.id.passwordFeedback);

    back = findViewById(R.id.back1);


    editTextPassword.addTextChangedListener(new TextWatcher() {

```

```

@Override

public void beforeTextChanged(CharSequence s, int start, int count, int after) {}

@Override

public void onTextChanged(CharSequence s, int start, int before, int count) {

    validatePasswordRealTime(s.toString());

}

@Override

public void afterTextChanged(Editable s) {}

});

//back button

back.setOnClickListener(new View.OnClickListener() {

    @Override

    public void onClick(View view) {

        finish();

    }

});

//Register button

buttonRegister.setOnClickListener(new View.OnClickListener() {

    @Override

    public void onClick(View v) {

        if (validateInputs()) {

            String Uname = name.getText().toString().trim();

            String username = editTextUsername.getText().toString().trim();

            String email = editTextEmail.getText().toString().trim();

```

```

String phone = editTextPhone.getText().toString().trim();

String password = editTextPassword.getText().toString().trim();

String securityQuestion = SecurityQuestion.getSelectedItem().toString();

String securityAnswer = editTextSecurityAnswer.getText().toString().trim();


// Check duplicate

if (dbHelper.isUserDuplicate(username, phone, email)) {

    Toast.makeText(Registration.this, "Account with this username, phone number, or email already
exists.", Toast.LENGTH_SHORT).show();

} else {

    boolean isInserted = dbHelper.insertUser(

        Uname,email,phone,username,password, securityQuestion, securityAnswer

    );

    if (isInserted) {

        Toast.makeText(Registration.this, "Registration Successful", Toast.LENGTH_SHORT).show();

        databaseReference = FirebaseDatabase.getInstance().getReference("users");


        FirebaseAuth.getInstance().createUserWithEmailAndPassword(email.trim(), password)
//following is all firebase in order to chat

        .addOnSuccessListener(new OnSuccessListener<AuthResult>() {

            @Override

            public void onSuccess(AuthResult authResult) {

                UserProfileChangeRequest userProfileChangeRequest = new
UserProfileChangeRequest.Builder().setDisplayName(username).build();

                FirebaseUser firebaseUser = FirebaseAuth.getInstance().getCurrentUser();

                firebaseUser.updateProfile(userProfileChangeRequest);


                UserModel userModel = new UserModel(FirebaseAuth.getInstance().getUid(), Uname,
email, password);

```

```

        databaseReference.child(FirebaseAuth.getInstance().getUid()).setValue(usermodel);

        Intent intent = new Intent(Registration.this, MainActivity.class);

        intent.putExtra("Uname", Uname);

        startActivity(intent);
    }
})

.addOnFailureListener(new OnFailureListener() {

    @Override

    public void onFailure(@NonNull Exception e) {

        Toast.makeText(Registration.this, "FireBase Registration Failed. Try Again.",
Toast.LENGTH_SHORT).show();

    }

});

```

```

        Intent intent = new Intent(Registration.this, Login.class);

        startActivity(intent);

        finish();

    } else {

        Toast.makeText(Registration.this, "Registration Failed. Try Again.",
Toast.LENGTH_SHORT).show();

    }

}

}

}

});

```

```

String[] securityQuestions = {

    "Select Security Question",

    "What was the name of your first pet?",

```



```

        "What was the name of your first school?",
        "What is your favorite book?",
        "What city were you born in?"
    };

```

```

    ArrayAdapter<String> adapter = new ArrayAdapter<>(this, android.R.layout.simple_spinner_item,
securityQuestions);

```

```

    adapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item);

    SecurityQuestion.setAdapter(adapter);
}

```

```

// Pass validate

```

```

private void validatePasswordRealTime(String password) {
    if (TextUtils.isEmpty(password)) {
        passwordFeedback.setText("");
        return;
    }
}

```

```

boolean isLongEnough = password.length() >= 8;
boolean hasNumber = password.matches(".*\\d.*");
boolean hasSpecialChar = password.matches(".*[!@#$%^&*].*");

```

```

if (isLongEnough && hasNumber && hasSpecialChar) {
    passwordFeedback.setText("Password meets requirements.");
    passwordFeedback.setTextColor(Color.GREEN);
} else {

```

```

    passwordFeedback.setText("Password must be at least 8 characters, contain a number, and a special
character.");
}

```

```

        passwordFeedback.setTextColor(Color.RED);
    }
}

// User input validate
private boolean validateInputs() {
    String Uname = name.getText().toString().trim();
    String username = editTextUsername.getText().toString().trim();
    String password = editTextPassword.getText().toString().trim();
    String phone = editTextPhone.getText().toString().trim();
    String email = editTextEmail.getText().toString().trim();
    String selectedQuestion = SecurityQuestion.getSelectedItemAt().toString();

    boolean isValid = true;

    if (TextUtils.isEmpty(Uname)) {
        name.setError("Name is required");
        isValid = false;
    }

    if (TextUtils.isEmpty(username)) {
        editTextUsername.setError("Username is required");
        isValid = false;
    }

    if (TextUtils.isEmpty(password) || !checkPassword(password)) {

```

```
        editTextPassword.setError("Password must be at least 8 characters, contain a number, and a special character");
```

```
        isValid = false;
    }
}
```

```
if (TextUtils.isEmpty(phone) || !checkPhone(phone)) {
    editTextPhone.setError("Phone number is required and must be exactly 10 digits");
    isValid = false;
}
```

```
if (TextUtils.isEmpty(email) || !checkEmail(email)) {
    editTextEmail.setError("Invalid email format");
    isValid = false;
}
```

```
if (selectedQuestion.equals("Select Security Question")) {
    Toast.makeText(this, "Please select a valid security question", Toast.LENGTH_SHORT).show();
    isValid = false;
}
```

```
String securityAnswer = editTextSecurityAnswer.getText().toString().trim();
if (TextUtils.isEmpty(securityAnswer)) {
    editTextSecurityAnswer.setError("Answer to security question is required");
    isValid = false;
}
```

```
return isValid;
```

```

    }

    // 1 number 1 special

    private boolean checkPassword(String password) {

        return password.length() >= 8 &&

            password.matches(".*\\d.*") &&

            password.matches(".*[!@#$%^&*].*");

    }


    // Number

    private boolean checkPhone(String phone) {

        return phone.matches("\\d{10}");

    }


    // email @

    private boolean checkEmail(String email) {

        return email.contains("@") && android.util.Patterns.EMAIL_ADDRESS.matcher(email).matches();

    }

}

```