Sprint 4 Plan - Project Jimmy

Team roles:

Mrunmay Deshmukh - Scrum Master
Wei-Chien Cheng - Product Owner
Yash Phatak - Developer
Kushal Lahoti - Developer
ChuanHsin Wang - Developer
Kuan-Ru Huang - Developer
Barry Liu - Developer

Post sprint client meeting date/time/place:

Date: 30th October, 2024 and 6th December, 2024 (Every Wednesday)
Time: 10:30am - 11:00am
Place: Zoom Call

Client Meeting Summary:

In the recent client meeting, we provided a comprehensive demo showcasing the functionalities completed as part of the Sprint 3 plan. The client expressed satisfaction with the progress of the development, appreciating the pace and quality of the work delivered.

We discussed the upcoming beta user testing phase, which is scheduled to begin next Wednesday. This testing will cover the features developed up to that point and is aimed at gathering early user feedback to refine the application further.

Looking ahead to the current sprint, we outlined our primary objectives, including the development and full integration of a Chat feature to enable user communication within the application. Additionally, we informed the client about several bugs identified in the project. Resolving these issues is a key focus for this final sprint to ensure stability and quality in the overall product.

The client was receptive to the plan, and we will continue to update them on our progress and any feedback from beta testing.

Sprint goal:

The primary goals for the current sprint are as follows:

1. **Bug Fixes**: Resolve multiple bugs identified in the previous sprint to enhance the application's stability and ensure a smooth user experience. These bug fixes are essential for polishing the application as we approach the final testing phases.
2. **Chat Feature Implementation**: Develop and integrate a Chat feature that allows seamless communication between users. This feature is a core component to improve user interaction within the application, and we aim to complete its full integration by the end of this sprint.
3. **Initiation of Beta User Testing**: Begin the beta user testing phase next Wednesday to gather real-world feedback on features developed so far. This testing will provide valuable insights and allow us to make final adjustments based on user experiences and suggestions.

Each of these objectives contributes to the overall sprint goal of enhancing functionality, improving quality, and preparing the application for broader release.

User stories: A total of 11 user stories have been added to the sprint plan.

Feature: Chat Functionality Backend and Database Setup
As a user,
I want a reliable chat feature that allows me to communicate with workout partners,
So that I can easily coordinate and stay connected with my fitness community.

Scenario 1: Create Chat Feature Backend
Given: The development team is setting up the backend for the chat feature.
When: They implement the necessary API endpoints and service logic to support message sending, receiving, and real-time updates.
Then: The chat feature should facilitate smooth, secure, and responsive communication between users.

Scenario 2: Setup Database Tables for Chat Feature
Given: The development team is configuring the database for chat functionality.

When: They create tables for storing messages, user IDs, timestamps, and chat session details.
Then: The database should securely store chat data and allow for efficient retrieval and querying to support real-time messaging.

Scenario 3: Ensure Chat Feature Works Seamlessly
Given: The chat feature has been implemented and configured with database support.
When: The development team tests message sending, receiving, and real-time delivery between users.
Then: The chat functionality should work without errors, supporting consistent and instant communication.

Scenario 4: Write Cucumber Scenarios and RSpec Tests for Chat Feature Backend
Given: The chat backend and database setup are complete.
When: The development team writes Cucumber scenarios to test chat functionality from a user perspective and RSpec tests to verify API endpoints and database interactions.
Then: The tests should validate that messages can be sent, received, and stored correctly, and all aspects of the chat feature should function as expected.

Feature: Integration of Chat Backend with Frontend UI
As a user,
I want to use a fully functional chat feature with a smooth interface,
So that I can seamlessly communicate with my workout partners without any disruptions.

Scenario 1: Connect Backend with Chat UI
Given: The backend for chat functionality has been developed, and a chat UI exists on the frontend.
When: The development team integrates the backend endpoints with the chat UI to send and receive messages.
Then: The chat UI should display messages in real-time, reflecting the backend data accurately.

Scenario 2: Display Real-Time Messages in Chat UI
Given: The user is in a chat session with a workout partner.

When: Messages are sent or received through the chat backend.
Then: The messages should appear instantly in the chat UI, updating in real-time without the need for manual refresh.

Feature: User Matching and Interaction Options
As a user,
I want to view and interact with matched users,
So that I can connect with workout partners who align with my preferences and easily manage my interactions with them.

Scenario 1: View All Matched Users
Given: The user has set their fitness preferences.
When: The user navigates to the "Matched Users" page.
Then: A list of all users who match their preferences should be displayed, with relevant details such as name, shared interests, and location.

Scenario 2: Search for a Specific User
Given: The user wants to find a specific workout partner from their matched users.
When: The user types the name or a keyword related to the user's profile in the search bar on the "Matched Users" page.
Then: The matched user list should be filtered to show only users that match the search query.

Scenario 3: View Complete Profile of a Matched User
Given: The user is viewing the list of matched users.
When: The user clicks on a matched user's profile link.
Then: The complete profile of the selected user should be displayed, showing detailed information such as activity preferences, experience level, and availability.

Scenario 4: Block a Matched User
Given: The user wants to manage their interactions with a matched user.
When: The user clicks the "Block" button on a matched user's profile.
Then: The selected users should be blocked, and they should be removed from the matched users list to prevent future interaction.

Scenario 5: Open Chat with a Matched User

Given: The user wants to initiate a conversation with a matched workout partner.
When: The user clicks the "Chat" button on the matched user's profile.
Then: A chat window should open, allowing the user to start a conversation with the matched user in real-time.

Feature: Fitness Profile Management Bug Fixes
As a user,
I want a smooth experience on the fitness profile management page,
So that I can update my profile details without issues or interruptions.

Scenario 1: Fix Intermittent Loading Issue of Workout Days Input Box
Given: The user is on the fitness profile management page.
When: The page loads and the workout days input box appears.
Then: The input box should load consistently every time the page is accessed, without any intermittent failures.

Scenario 2: Close Dropdown Options When Clicking Outside
Given: The user has opened a dropdown on the fitness profile management page.
When: The user clicks anywhere outside the dropdown area.
Then: The dropdown should close automatically to provide a seamless and intuitive user experience.

Feature: Bug Fixes for Profile Matching Page
As a user,
I want a consistent and error-free experience on the profile matching page,
So that I can view and interact with matched profiles without interruptions or issues.

Scenario 1: Display “No Available Profiles” Message When No Matches Exist
Given: There are no profiles that match the user’s preferences.
When: The user opens the profile matching page.
Then: A message should appear saying, “No available profiles,” clearly indicating that no matches are found.

Scenario 2: Prevent Reappearance of the Last Matched Profile
Given: The user has matched with the last available profile.
When: The last profile is processed and there are no more matches.
Then: The last profile should not reappear, and the message “No available profiles” should be displayed instead.

Scenario 3: Restrict Profile Matching Before Completing Fitness Profile
Given: The user has not completed their fitness profile.
When: The user attempts to access the profile matching page.
Then: They should be redirected to complete their fitness profile first, preventing access to profile matching until it is finished.

Scenario 4: Add a Confirmation Modal Box for Blocking Users
Given: The user wants to block another user from their matches.
When: The user clicks the “Block” button on a matched profile.
Then: A custom modal box should appear, asking for confirmation before blocking, ensuring that users don’t accidentally block someone.

Scenario 5: Display User Photos in Profile Matching Cards
Given: The user is viewing matched profiles on the profile matching page.
When: The user photos should load for each matched profile card.
Then: The profile cards should display each user's photo, making the profiles more visually informative.

Scenario 6: Fix Top Navbar Padding for Mobile View
Given: The user is accessing the profile matching page on a mobile device.
When: The top navbar loads in mobile view.
Then: The navbar padding should be consistent with other screens, improving the visual consistency of the application on mobile.

Feature: Enhanced Dashboard Design for User Landing Page
As a user,
I want an aesthetically pleasing and functional dashboard,
So that I can easily access key features while enjoying a visually appealing interface.

Scenario 1: Implement a Visually Appealing Dashboard Layout
Given: The user logs into the application and lands on the dashboard.
When: The dashboard loads with an optimized layout.
Then: The dashboard should have a clean, organized design with distinct sections for recent activity, profile highlights, and navigation, enhancing readability and accessibility.

Scenario 2: Use Modern Color Scheme and Typography
Given: The user views the dashboard after login.
When: The design includes a new color scheme and typography.
Then: The colors should be visually appealing yet easy on the eyes, with a consistent font style that is legible on both desktop and mobile devices.

Scenario 3: Make the Dashboard Mobile-Responsive
Given: The user opens the dashboard on a mobile device.
When: The dashboard loads in a mobile-friendly format.
Then: The layout, buttons, and widgets should adapt to mobile screens, keeping functionality and aesthetics consistent across all devices.

Feature: Custom Alert Boxes for Action Confirmations
As a user,
I want custom alert boxes for confirmation actions,
So that I can have a clear, visually distinct confirmation prompt before taking important actions.

Scenario 1: Display Custom Confirmation Modal for Sensitive Actions
Given: The user is about to perform an action that requires confirmation, such as deleting a profile or blocking a user.
When: The user clicks the action button (e.g., “Delete” or “Block”).
Then: A custom alert box modal should appear, providing a clear and visually distinct confirmation prompt with “Confirm” and “Cancel”

options.

Scenario 2: Include Descriptive Text and Actionable Buttons
Given: The user opens a custom confirmation alert box.
When: The alert box is displayed for actions like deletion or blocking.
Then: The modal should display a descriptive message (e.g., “Are you sure you want to delete this item?”) and two clear buttons: “Confirm” (styled in a distinct color) and “Cancel” (styled in a neutral color), to guide the user’s choice.

Scenario 3: Prevent Background Interactions When Modal is Active
Given: The user opens a custom confirmation alert box.
When: The modal appears on the screen.
Then: The background should be dimmed, and interactions with background elements should be disabled until the user makes a choice (either confirming or canceling the action), focusing their attention on the alert box.

Scenario 4: Use Custom Styles for the Alert Box
Given: The user interacts with a confirmation modal.
When: The custom alert box appears on the screen.
Then: The alert box should use a visually consistent design with soft shadows, rounded corners, and a color scheme that matches the application’s theme, creating an aesthetically pleasing experience.

Feature: Fix Background Image Visibility for Tall Screens
As a user,
I want the background image to display consistently across all screens,
So that I have a seamless visual experience without empty areas on taller pages.

Scenario 1: Ensure Background Image Stretches to Fit Tall Screens
Given: The user is on a page with content that exceeds the screen height.
When: The page loads or is scrolled past the initial screen height.
Then: The background image should stretch or repeat seamlessly to cover the full height of the page, preventing any areas without the background.

Scenario 2: Set the Background Image to Cover the Entire Page

Height Dynamically
Given: The page content height exceeds the default screen size.
When: The background image is applied to the page.
Then: The CSS should dynamically adjust the background image to use background-size: cover or background-size: contain, with options like background-repeat: no-repeat or background-repeat: repeat (depending on the design), so the image covers the entire page consistently.

Feature: Hide Navbar Button on Login Page for Mobile View
As a user,
I want a simplified interface on the login page in mobile view,
So that unnecessary elements like the navbar button are hidden when they are not needed.

Scenario 1: Remove the Navbar Button on the Login Page for Mobile Devices
Given: The user is on the login page using a mobile device.
When: The page loads in mobile view.
Then: The navbar button should be hidden, as the login page does not have additional options, providing a cleaner and less distracting interface.

Scenario 2: Ensure the Navbar Button Reappears on Other Pages in Mobile View
Given: The user navigates to other pages in the mobile view.
When: The user leaves the login page and accesses other sections of the app.
Then: The navbar button should reappear to provide navigation options, maintaining usability across other pages.

Feature: Fix Notification Modal Box Bugs and Improve UI for Read/Unread Notifications
As a user,
I want a clear and organized notifications modal,
So that I can easily distinguish between read and unread notifications and avoid unnecessary options.

Scenario 1: Remove the "Mark as Unread" Button

Given: The user opens the notifications modal box.
When: The user views the available actions for notifications.
Then: The "Mark as Unread" button should be removed, leaving only relevant options to simplify the interface.

Scenario 2: Display Unread Notifications at the Top
Given: The user has both read and unread notifications.
When: The notifications modal box is opened.
Then: Unread notifications should appear at the top of the list, followed by read notifications at the bottom, making it easy to identify new information.

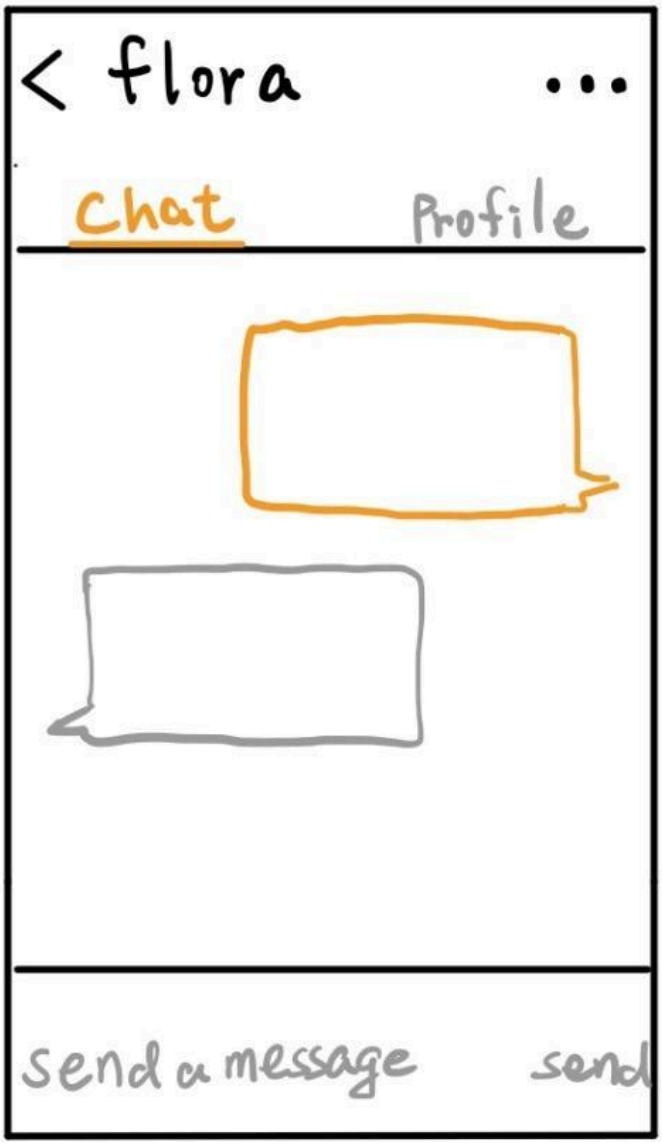
Scenario 3: Apply Bright Design to New (Unread) Notifications
Given: The user opens the notifications modal and views unread notifications.
When: Unread notifications are displayed.
Then: Unread notifications should have a bright design (e.g., bold text, highlighted background) to visually distinguish them as new, ensuring they capture the user's attention.

Feature: Fix User Profile Loading Issues Due to JavaScript and Cache Errors
As a user,
I want my profile to load reliably without interruptions,
So that I can access and update my information without facing loading errors.

Scenario 1: Fix JavaScript Errors Causing Profile Load Failures
Given: The user navigates to their profile page.
When: The profile page loads with JavaScript functions running.
Then: Any JavaScript errors that prevent the page from loading should be identified and resolved, ensuring the profile loads fully and reliably.

User interface: Lo-fi UI mockups and storyboards

Chat Room UI:



Which stories were pulled into the Sprint? To whom is each story assigned?

Stories	**Points**	**Assigned To**
Chat Functionality Backend and Database Setup	4	Barry Liu
Integration of Chat Backend with Frontend UI	4	Kuan-Ru Huang
User Matching and Interaction Options	3	Kushal Lahoti
Fitness Profile Management Bug Fixes	2	Mrunmay Deshmukh
Bug Fixes for Profile Matching Page	4	Yash Phatak
Enhanced Dashboard Design for User Landing Page	2	Mrunmay Deshmukh
Custom Alert Boxes for Action Confirmations	2	Wei-Chien Cheng
Fix Background Image Visibility for Tall Screens	1	Wei-Chien Cheng
Hide Navbar Button on Login Page for Mobile View	1	Wei-Chien Cheng
Fix Notification Modal Box Bugs and Improve UI for Read/Unread Notifications	4	ChuanHsin Wang
Fix User Profile Loading Issues Due to JavaScript and Cache Errors	1	Kushal Lahoti

What are the tasks and their time estimates?

A. Chat Functionality Backend and Database Setup - 1 Week
B. Integration of Chat Backend with Frontend UI - 1 Week
C. User Matching and Interaction Options - 4 Days
D. Fitness Profile Management Bug Fixes - 3 Days
E. Bug Fixes for Profile Matching Page - 1 Week
F. Enhanced Dashboard Design for User Landing Page - 3 Days
G. Custom Alert Boxes for Action Confirmations - 3 Days
H. Fix Background Image Visibility for Tall Screens - 2 Days
I. Hide Navbar Button on Login Page for Mobile View - 2 Days
J. Fix Notification Modal Box Bugs and Improve UI for Read/Unread Notifications - 1 Week
K. Fix User Profile Loading Issues Due to JavaScript and Cache Errors - 2 Days

Links to our GitHub Repo, Heroku Deployment, Pivotal Tracker, Slack workspace, and Code Climate:

- **Github Repo -** https://github.com/tamu-edu-students/jimmy-gym-buddy-finder
- **Deployment URL -** https://jimmy-buddy-finder-f97708d96ef8.herokuapp.com/
- **Pivotal Tracker -** https://www.pivotaltracker.com/n/projects/2721606
- **Slack Workspace -** https://app.slack.com/client/T07P2NT2ZM1/C07P00FFRGD
- **Code Climate -** https://codeclimate.com/github/tamu-edu-students/jimmy-gym-buddy-finder