Milestone 3: Project Implementation

By Ryan Davis, Ivan Farfan, Johan Jaramillo, and Cory Vitanza

We pledge our Honor that we have abided by the Stevens Honor System

Table of Contents

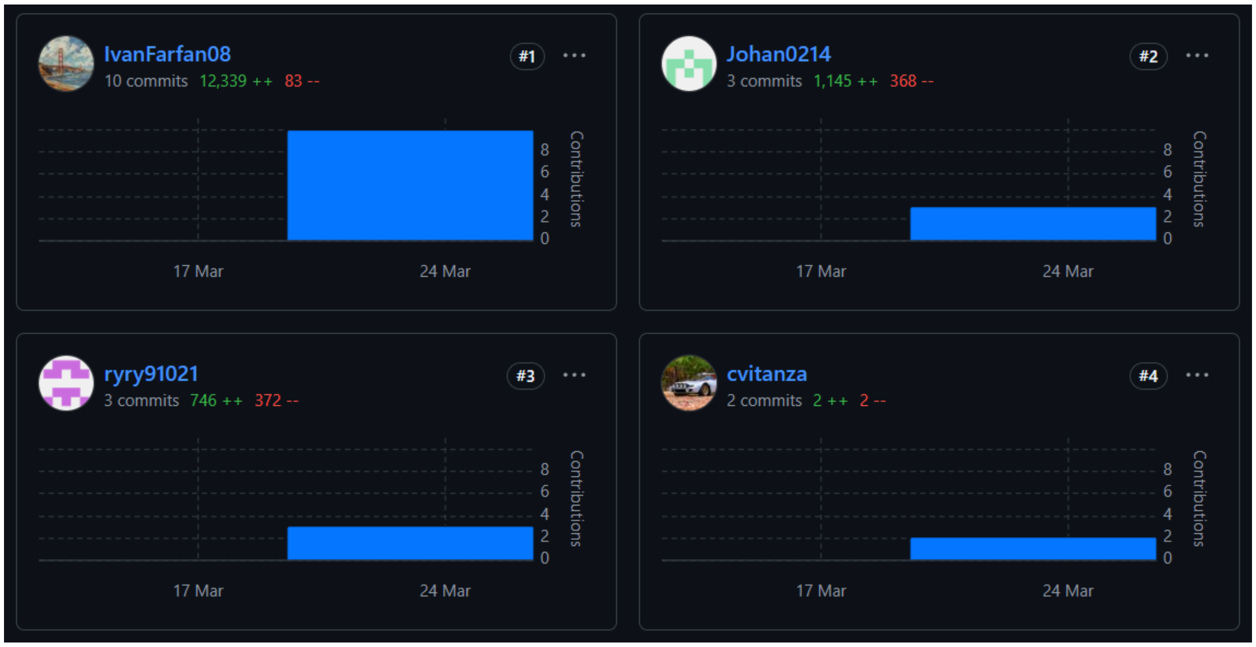
Introduction

Our group was tasked with creating a meal sharing application that allows users to host and join meals. Users can assume both roles under one account, and hosts are able to share information on a meal and invite others to join them while guests can view meals nearby and request to join the meal. The current milestone employs the following UML diagrams: use case diagram, activity diagram, class diagram, and sequence diagram. In doing so, the implementation of the design will be much smoother and more efficient, as the group will have an improved understanding of the system design.

In this milestone, our group focused on implementing key functionalities for the Meal Sharing Application. This included enabling users to log into the system, register new accounts, and allow hosts to create and share meal listings with detailed information such as photos and ingredients. Additionally, the team implemented a feature for guests to browse and view details of available meals. Throughout this sprint, we successfully developed and integrated these core features, ensuring the application met the essential requirements for user interaction, meal sharing, and basic payment processing.

Team Collaboration

- Ryan Davis- Stripe API implementation (CheckoutButton.tsx, App.tsx), documentation.
- Ivan Farfan- Manages React project. Implemented user account logic (Account.tsx, Auth.tsx, App.tsx), Manages GitHub and Pull Requests.
- Johan Jaramillo- Implemented Meal creation for user type “host” to provide meal cards (HostMeals.tsx).
- Cory Vitanza- Updated the "Find Meals" page layout and functionality (FindMeals.tsx). Implemented bugfixes for misc errors/bugs.



Note that these contributions / graphs do **not** include merge commits.
(Contributions per week to main, excluding merge commits)

Introduction to Project Repository

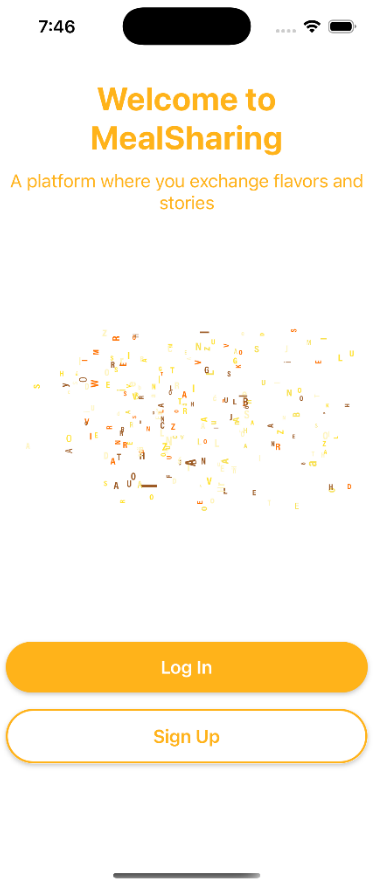
https://github.com/IvanFarfan08/MealSharing

Since this project is deployed on GitHub, the team implemented GitHub tools to maintain proper version control. With different branches, the team was able to develop the application in different components, allowing each developer to write different sheets of the app. Furthermore, when a developer completed a task on their branch, they were able to create a pull request, in which another developer can review the code before merging it into the main branch for deployment. Lastly, the team utilizes a features board, acting as an agile board, which manages the assignment and completion of tasks. With such measures, the team efficiently provided a functional first implementation of the Meal Sharing Application.

In the repository, a structured system of folders allows for the clear development of the application. The assets folder holds icons and images for the user interface. The components folder holds the typescript files, which hold the logic and the programming for the applications implementation. The lib folder holds configuration files for the Supabase database we are implementing. Other configuration files in the repository provide a set of rules and requirements for a machine to run the program. Such tools are essential for developers on different machines to have equivalent run time configurations. This repository's design includes simplicity and configurations allowing for simple and clear development.

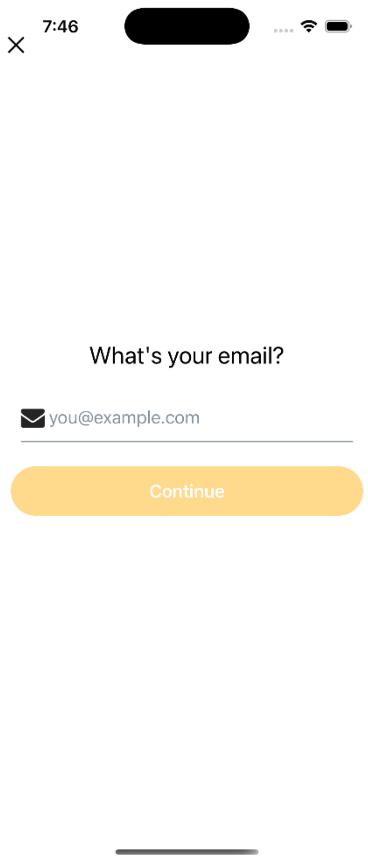
Summary of Changes

After this sprint’s development, the team was able to apply the UML diagrams created in the previous sprint to the functional requirements of this sprint. After signing in or creating a new account, users can host a new meal and provide information about such meals. When joining a meal, users can view the details of provided meals nearby.

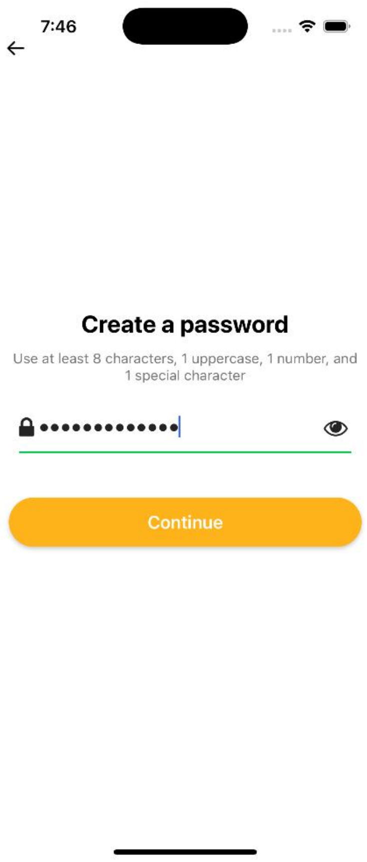


Welcome to MealSharing

A platform where you exchange flavors and stories

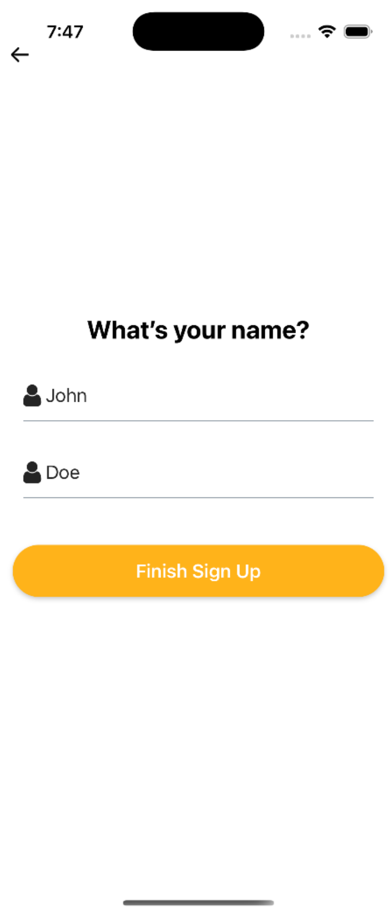


What's your email?



Create a password

Use at least 8 characters, 1 uppercase, 1 number, and 1 special character



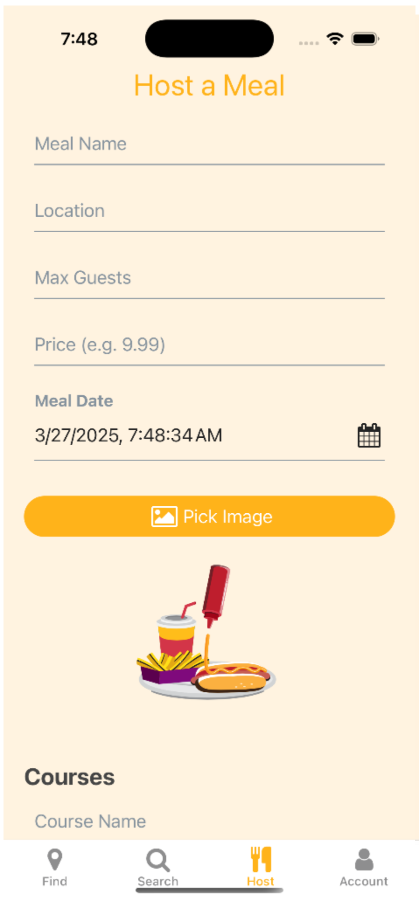
7:47

What's your name?

John

Doe

Finish Sign Up



7:48

Host a Meal

Meal Name

Location

Max Guests

Price (e.g. 9.99)

Meal Date

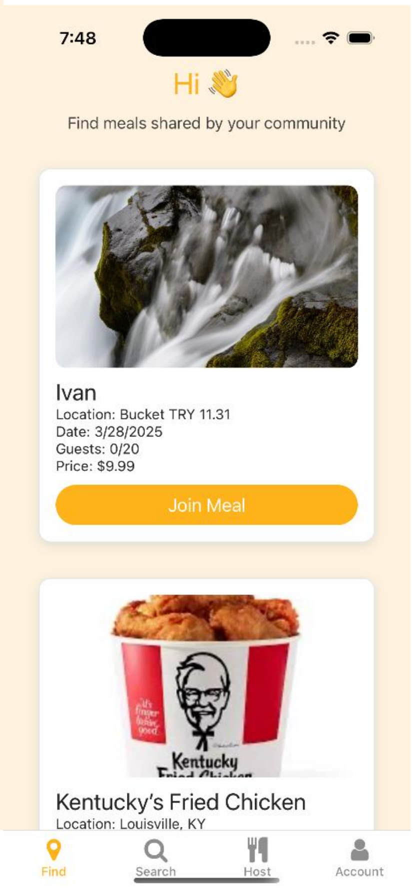
3/27/2025, 7:48:34 AM

Pick Image

Courses

Course Name

Find Search Host Account



7:48

Hi 👋

Find meals shared by your community

Ivan

Location: Bucket TRY 11.31

Date: 3/28/2025

Guests: 0/20

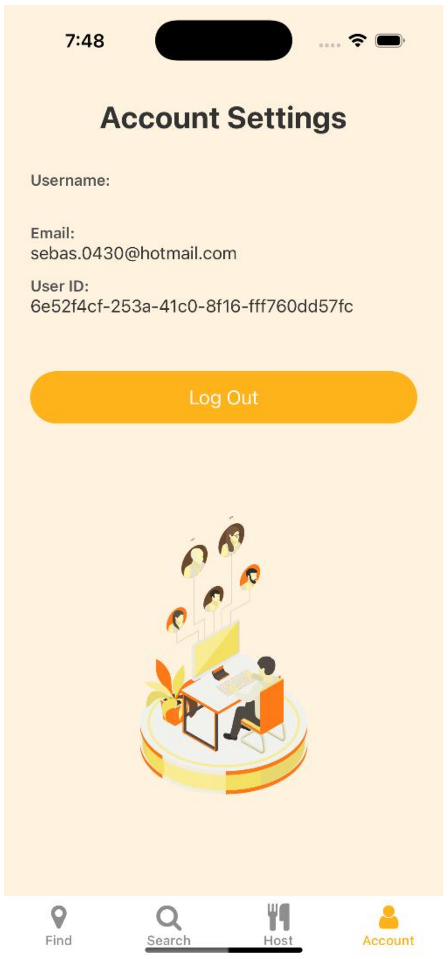
Price: $9.99

Join Meal

Kentucky's Fried Chicken

Location: Louisville, KY

Find Search Host Account



7:48

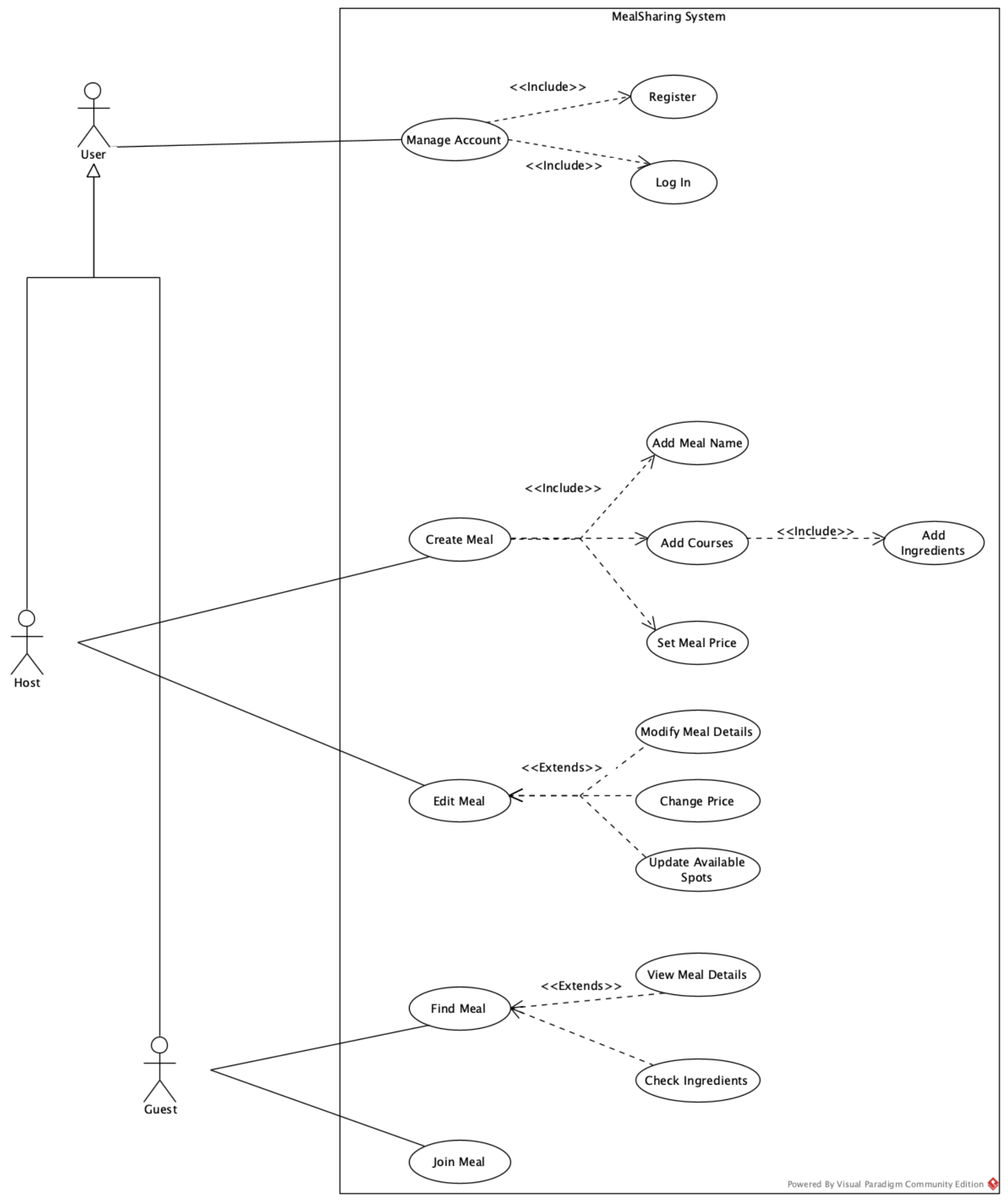
Account Settings

Username:

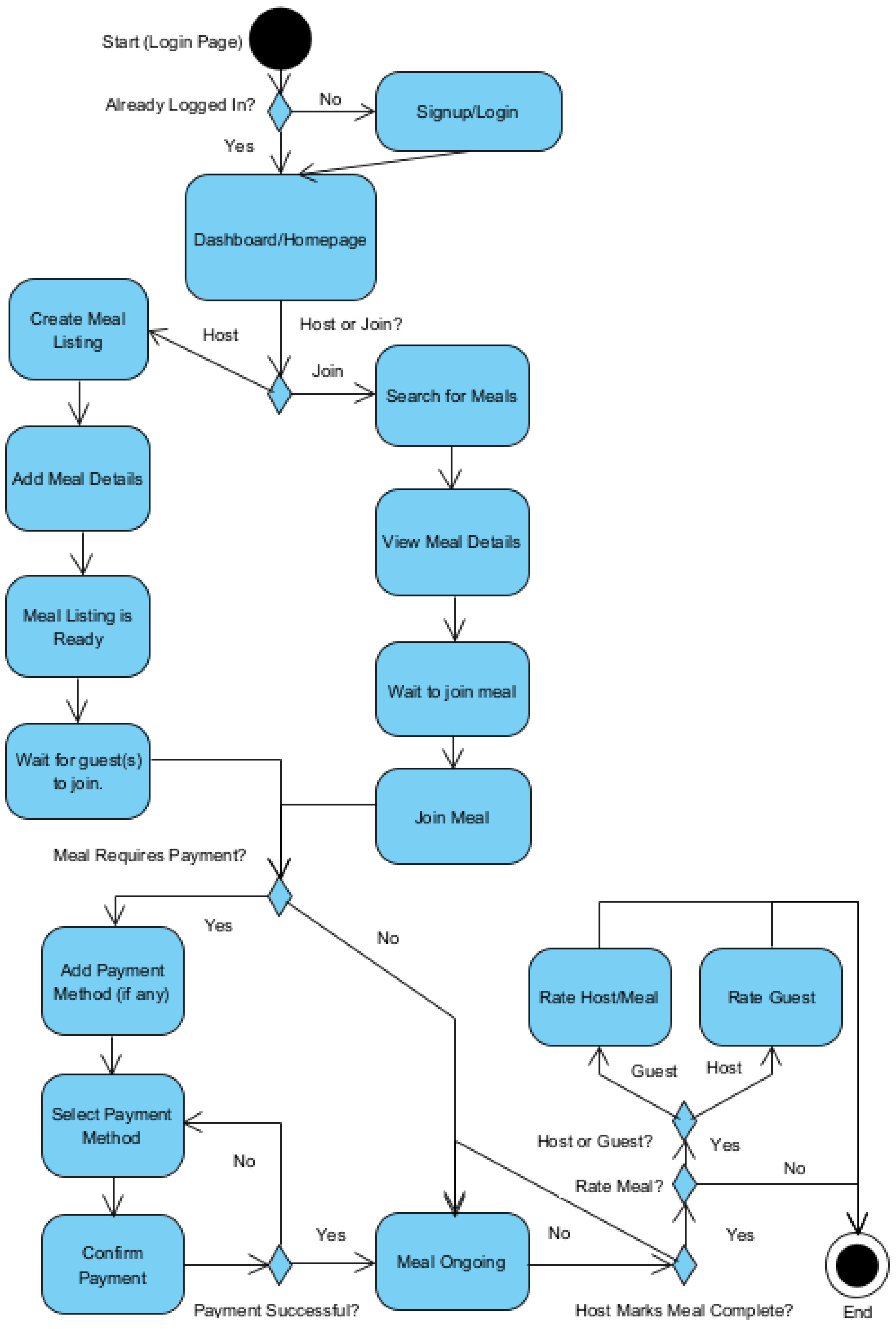
Email:
sebas.0430@hotmail.com

User ID:
6e52f4cf-253a-41c0-8f16-fff760dd57fc

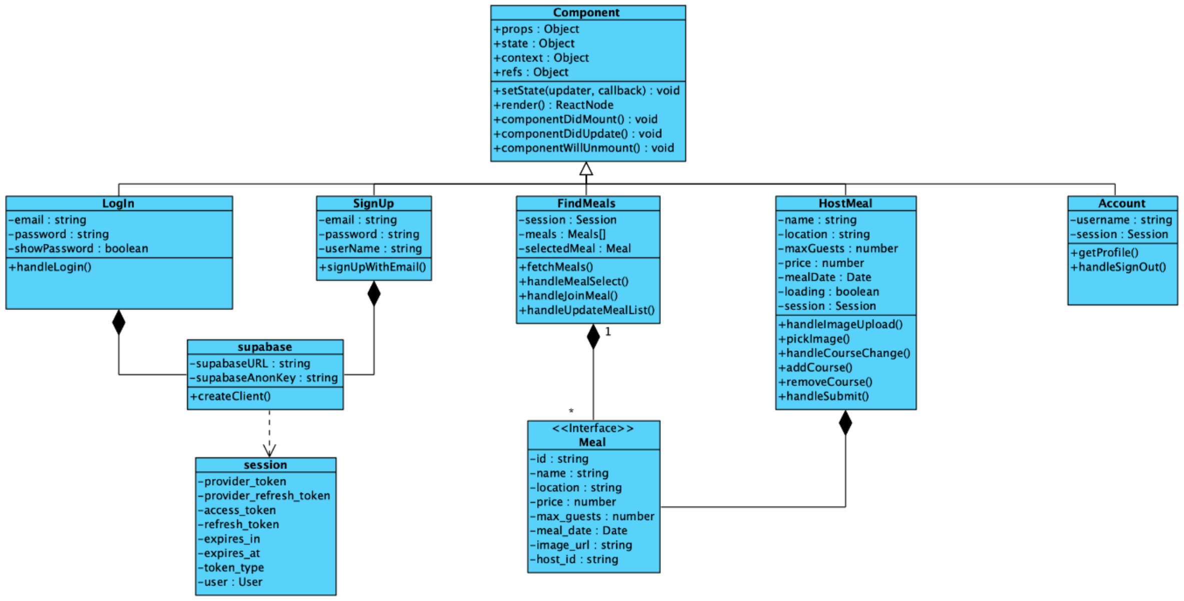
Updated Use-Case Diagram



Updated Activity Diagram



Updated Class Diagram



UPDATED WORKFLOW:

1. Start & Login Flow

The process begins at the Login Page where the system checks if the user is already logged in:

- Yes: The user is directed to the Dashboard/Homepage.
- No: The user is redirected to the Signup/Login screen to authenticate.

2. Choosing an Action: Hosting or Joining a Meal

Upon reaching the Dashboard/Homepage, the user decides whether to Host or Join a meal.

Hosting a Meal

- The host selects "Create Meal Listing."
- Adds meal details such as name, time, and location.
- The system confirms that the Meal Listing is Ready.
- The system waits for guests to join before proceeding.

Joining a Meal

- The guest selects "Search for Meals."

- Views the meal details.
- A Wait to Join Meal step is introduced
- The system then allows the user to join the meal.

3. Handling Payments (If required)

- A decision node checks if the meal requires payment.
- If payment is required:
    - The user is prompted to Add a Payment Method (if none exists).
    - The user selects the payment method and Confirms Payment.
    - A decision checks if the Payment is Successful:
        - Yes: The user proceeds to the Meal Ongoing step.
        - No: The user must retry payment.
- If no payment is required, the system moves directly to Meal Ongoing.

4. Meal Ongoing and Completion

- Once the meal is ongoing, the system waits for the Host to Mark the Meal as Complete.
- After the host marks the meal complete:
    - A decision checks if the user is a Host or Guest.
    - The system prompts for ratings:
        - Guest: Rates the host/meal.
        - Host: Rates the guest.
    - If the user skips the rating, the flow proceeds to the End.

5. End of Flow
After the meal is complete and any necessary ratings are submitted (or skipped), the workflow concludes, and the user's journey ends.

Key Updates/Changes:

- Added Wait to Join Meal step. (This is planned to be expanded upon in a later milestone)
- Introduced Host Marks Meal Complete as a required action.
- Refined Payment Process for greater clarity.

- Updated Rating Mechanism to distinguish between host and guest responsibilities.

Challenges and Strengths

Through this milestone, the team encountered several development and version control challenges. Since all developers were working on different machines, it was crucial to establish configurations that worked universally across environments. Additionally, multiple merge conflicts occurred due to branch-specific development and overlapping file changes. However, the team resolved these issues successfully, ensuring deployment remained functional.

Despite these challenges, the team demonstrated strong collaboration and technical execution:

- Resolved environment and configuration inconsistencies across different machines
- Handled and resolved multiple merge conflicts during development
- Successfully deployed a functional and responsive application
- Maintained consistent communication, enabling seamless component integration
- Leveraged strengths in React Native to complete tasks through clean pull requests

The most critical factor in our progress was effective communication, which allowed for smooth coordination and feature integration throughout this milestone.

Looking Forward

Moving forward, the team looks to further implementing the software, allowing for the full functionality of the program. We will continue in this iterative process to maintain proper version control through GitHub and will divide functional requirements into clear tasks. Given that the next milestone is more development, the team will continue using the same methodologies as the current sprint and will continue to meet the necessary requirements of each sprint.