

# Milestone 5: Final Delivery and Design Reflection

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

<i>Introduction .....</i>	<b>3</b>
<i>Team Collaboration .....</i>	<b>3</b>
<i>Introduction to Project Repository.....</i>	<b>4</b>
<i>Summary of Changes.....</i>	<b>5</b>
<i>Updated Use-Case Diagram .....</i>	<b>11</b>
<i>Updated Activity Diagram .....</i>	<b>12</b>
<i>Updated Class Diagram.....</i>	<b>13</b>
<i>Challenges and Strengths .....</i>	<b>15</b>
<i>Design Reflection .....</i>	<b>16</b>

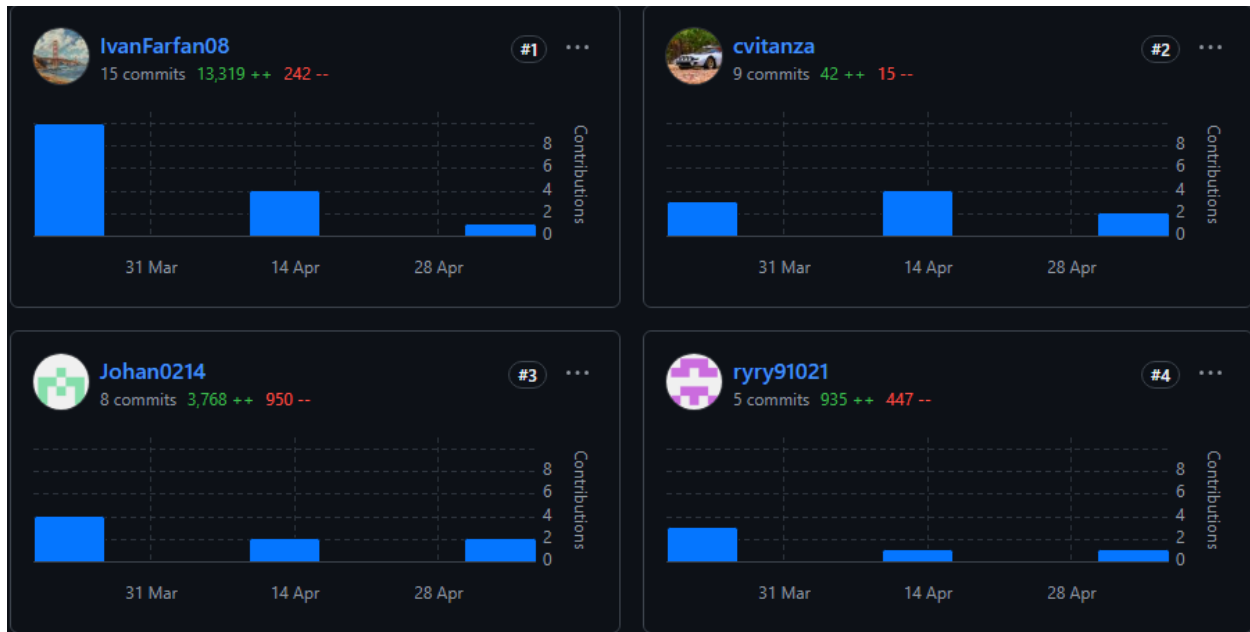
# Introduction

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 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 is much smoother and more efficient, as the group has developed a stronger understanding of the system architecture.

In this milestone, our group focused on enhancing interactivity and user engagement within the Meal Sharing Application. Key features implemented include guest-side meal browsing with filters for location, date, and meal details, as well as a request system for joining meals. Hosts now can view, accept, or reject incoming requests. We introduced meal rescheduling functionality, allowing hosts to change meal dates and automatically notify guests, who can then choose to accept the new date or opt out. Meal cancellation is now supported, ensuring flexibility for both parties. Additionally, the rating system has been refined with specific categories such as punctuality and friendliness for guests, and hospitality or organization for hosts to provide more meaningful feedback. These updates significantly improve user experience, reinforce dynamic host and guest interactions, and ensure a seamless and responsive application flow.

## Team Collaboration

- **Ryan Davis**
  - Focused on documentation. Implemented merge requests and bug fixes.
- **Ivan Farfan**
  - Manages Supabase DB. Implemented AI categorizations functionality.
- **Johan Jaramillo**
  - Implemented host meal cancellations and the notification center for meal status updates.
- **Cory Vitanza**
  - Focused on documentation. Performed bug testing/fixing.



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>

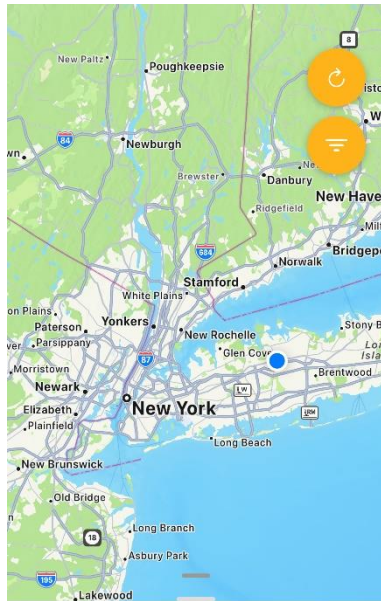
Since this project is deployed on GitHub, the team implemented GitHub tools to maintain proper version control. With different branches, the team was able to develop the application in different components, allowing each developer to write different sheets of the app. Furthermore, when a developer completed 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

Since milestones 2 and 3, milestone 4 has focused on expanding and refining the user experience around meal participation and management. Previously, joining a meal involved selecting a meal from a list of cards and receiving confirmation notifications. In milestone 4, meals are now displayed on an interactive map relative to the user's location, with added filters for easier browsing. Users can now filter meals by location, date, and other details before requesting to join. A new "My Meals" page has also been introduced, allowing users to view meals they are hosting, have requested to join, have been accepted to attend, or have completed, including pending reviews.

Additional features implemented in this milestone include the ability for hosts to reschedule meals and automatically notify guests of changes. Guests can then accept the new date or opt out if they are unavailable. Meal cancellation functionality was also added, allowing hosts or guests to cancel when necessary. Furthermore, the rating system has been upgraded to include specific categories for feedback, such as punctuality and friendliness for guests, and hospitality and organization for hosts. These improvements prompted several updates to the system design and user interface, particularly to support the new My Meals page and enhanced interaction between users.



Available Meals

1 meals found



11:21

3

Account

Username:  
ryry91021@gmail.com

Email:  
ryry91021@gmail.com

User ID:  
e1160d03-f873-4218-b849-17ac117dff1d

Average Rating:  
4.0 / 5.0

Reviews

★★★★★

No comment

4/16/2025

Log Out



### Meals You're Hosting

You haven't hosted any meals yet.

### Meals You've Joined

You haven't joined any meals yet.

### Meals You've Requested


You have no pending requests.


### Rate Previous Meals (as Guest)


You've reviewed all your past hosts.


### Rate Your Guests (as Host)


You've reviewed all your guests.

Find

My Meals


Host

Notificatio...

Account

11:19  
Huntington

The Paramount



## Pizza

**Location:**  
Location not specified

**Date & Time:**  
Thursday, April 24 at 7:49PM

**Price:**  
**\$2.99**


**Guests:**  
1/2


**Courses:**  
• 2


**Ingredients:**  
2: 2


Join Meal


Close

Find

My Meals

Host

Notificatio...

Account



Available Meals

1 meals found



Pizza

Thu, Apr 24 at 7:49 PM

\$2.99

Guests: 1/2

Join Meal



Find



My Meals



Host



Notificatio...



Account



Find



My Meals



Host



Notificatio...

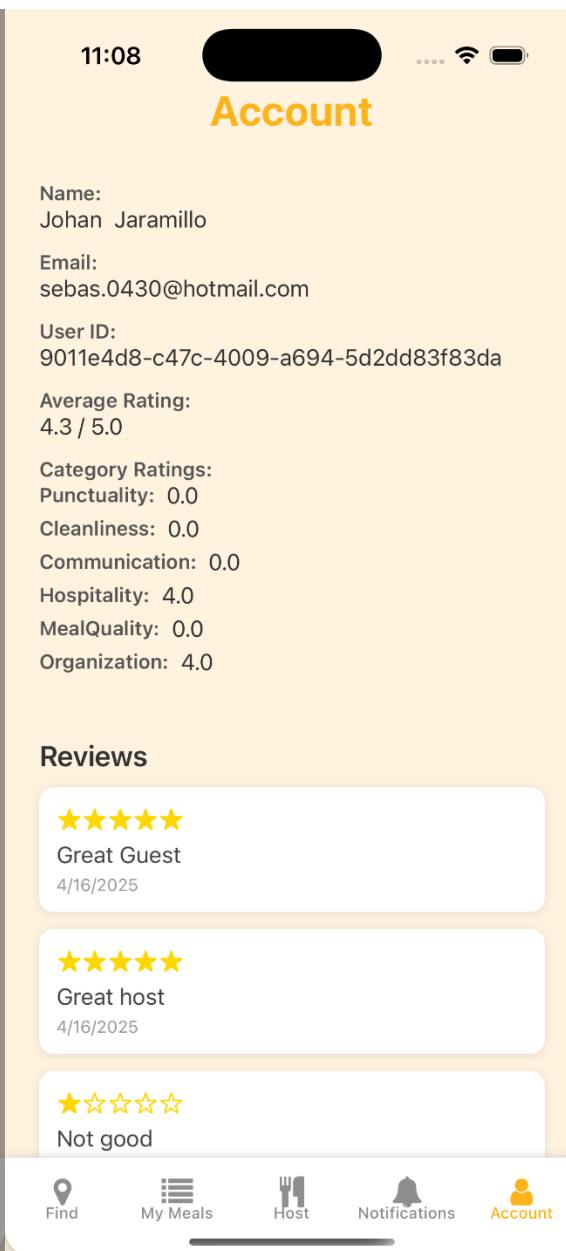
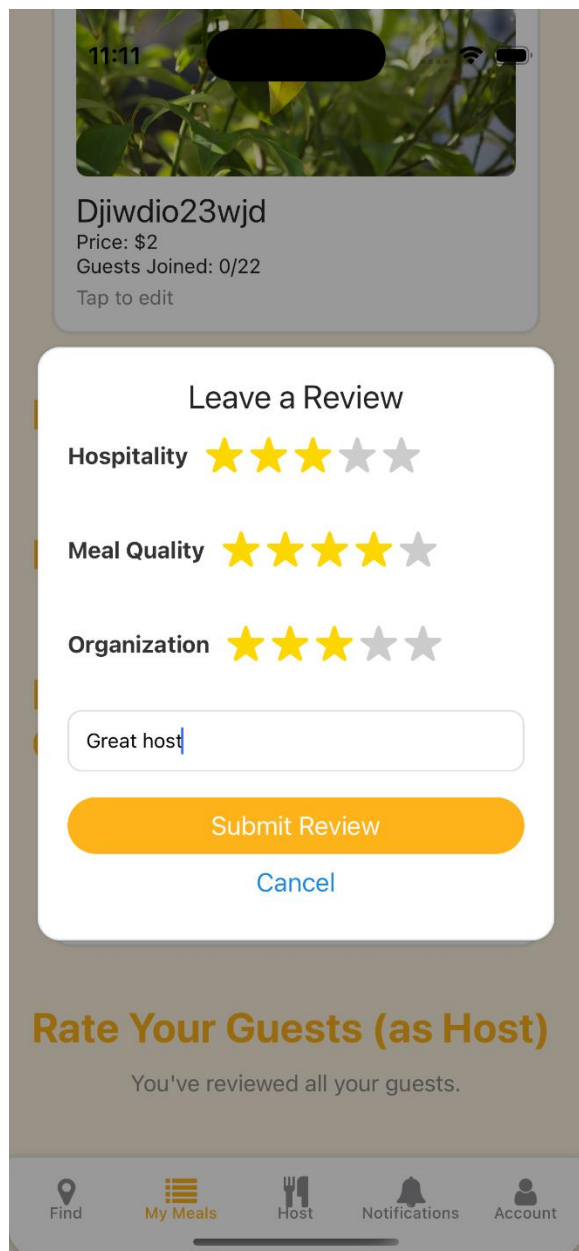


Account

## Notifications

You have no notifications





7:30

## Meals You're Hosting



### Pizza

Location: [object Object]

Price: \$2.99

Guests Joined: 0/2

Tap to edit

#### Incoming Join Requests:

##### Johan Jaramillo

Rating: 5.0/5.0



##### Recent Reviews:

5.0/5.0 - 4/16/2025

"Great Guest"

5.0/5.0 - 4/16/2025

"Great host"

Accept

Deny

## Meals You've Joined



Find



My Meals



Host

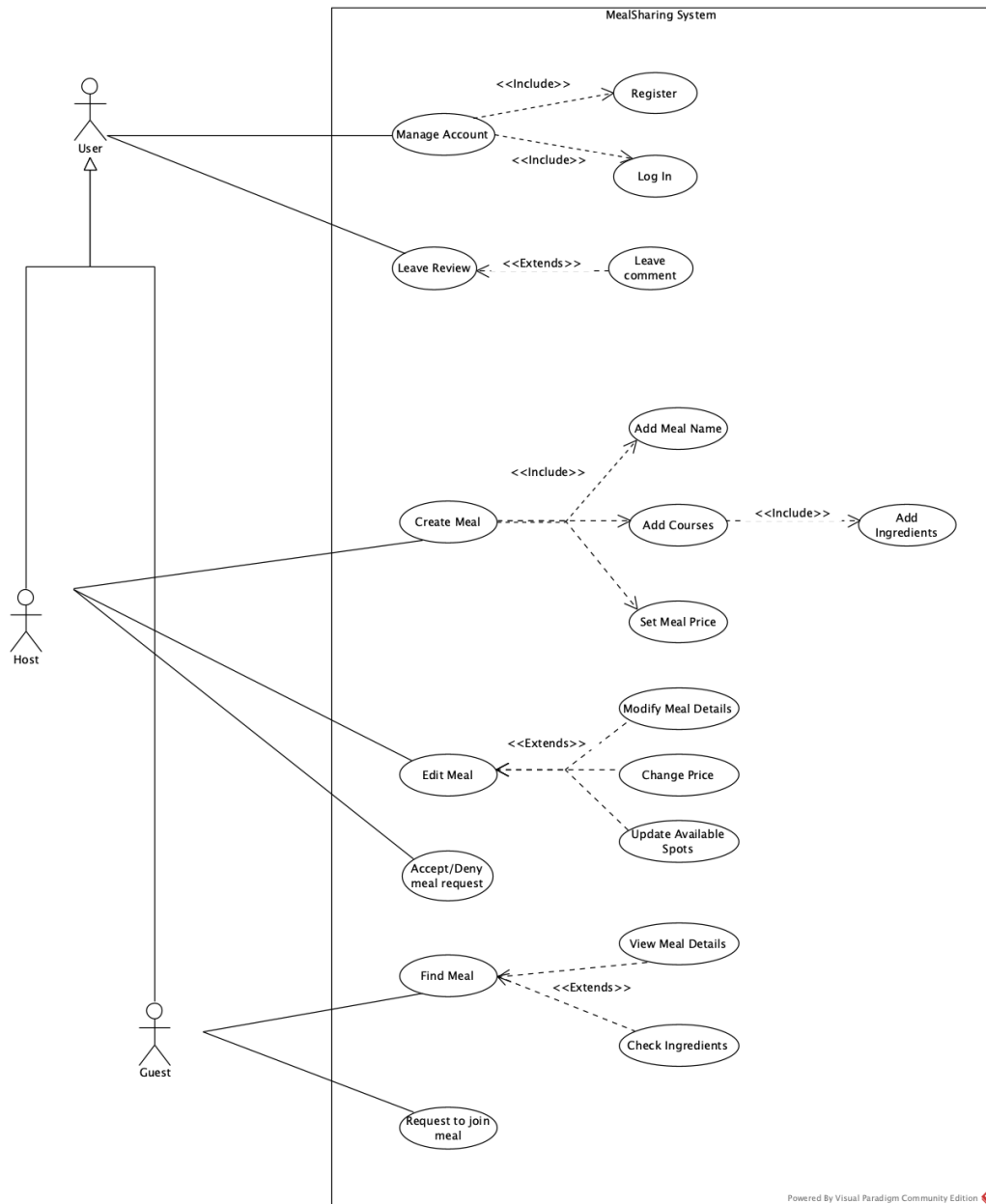


Notifications

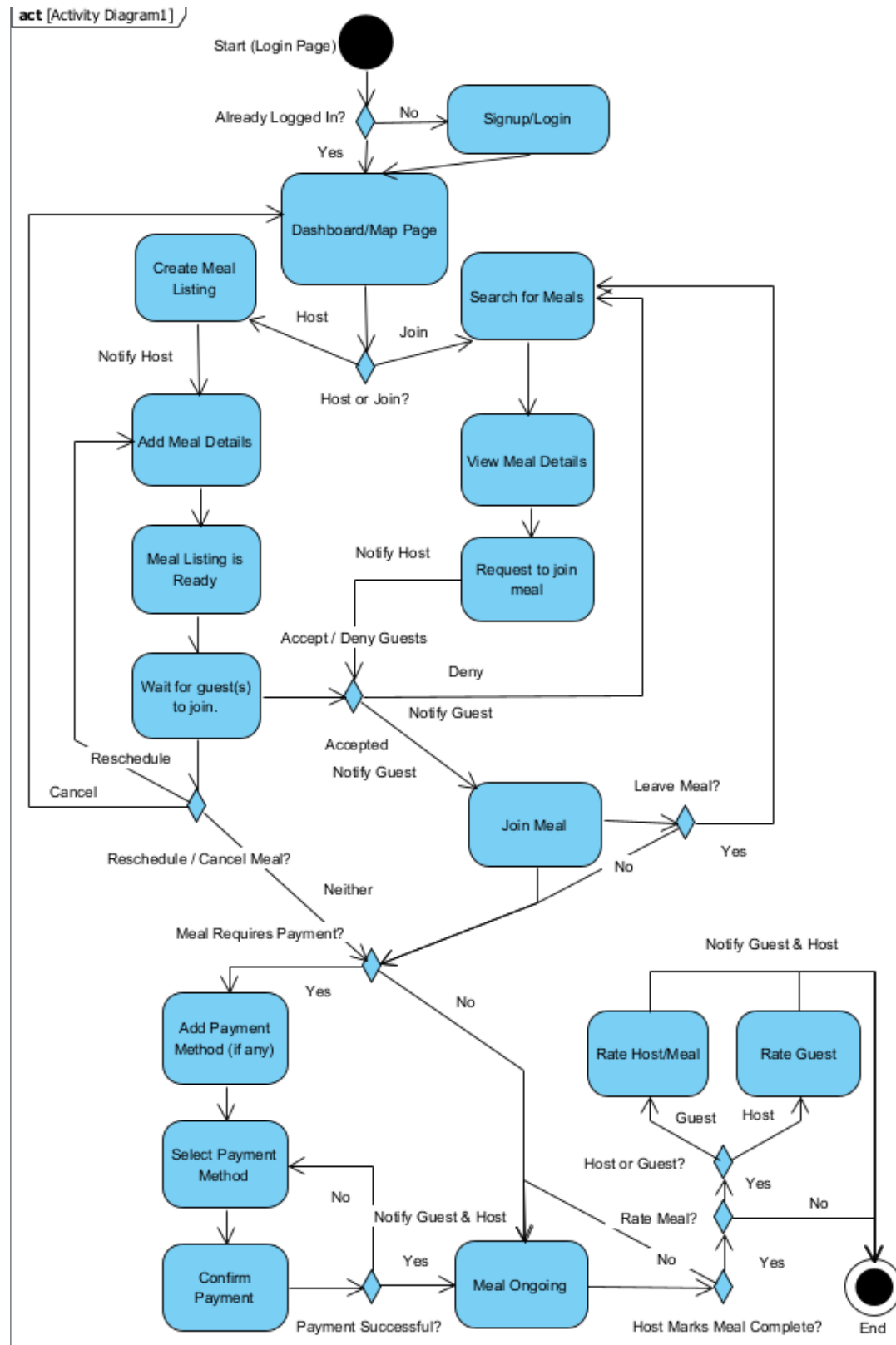


Account

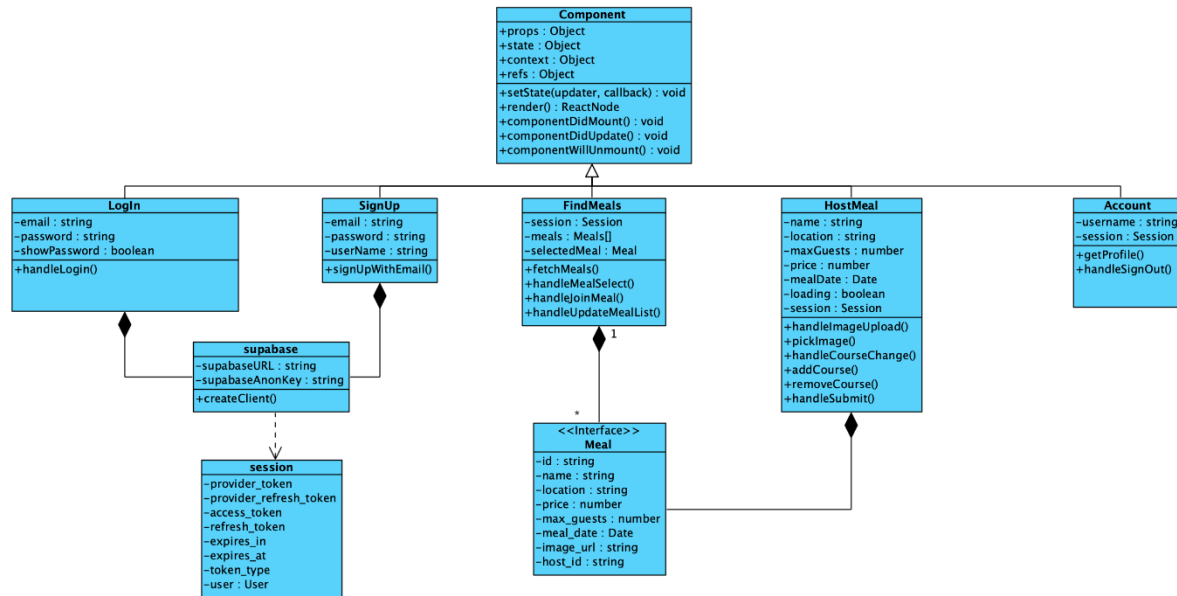
# Updated Use-Case Diagram



# Updated Activity Diagram



# Updated Class Diagram



## UPDATED WORKFLOW:

### 1. Start & Login Flow

The process begins at the Login Page:

- If the user is already logged in, they're directed to the Dashboard/Map Page.
- If not, they are sent to the Signup/Login screen to authenticate.

### 2. Choosing an Action: Hosting or Joining a Meal

Once on the Dashboard/Map Page, the user chooses to Host or Join a meal.

#### Hosting a Meal

- The host selects "Create Meal Listing."
- Adds meal details such as name, time, location, and an image.
- The meal listing becomes ready.
- The system waits for guest requests.
- Hosts can:
  - Accept or deny requests.
  - Reschedule the meal (date/time changes notify guests who can accept or opt out).
  - Cancel the meal, notifying all guests.

## **Joining a Meal**

- The guest searches for meals via the map.
- Views meal details.
- Send a request to join.
- Waits for host approval.
- If approved:
  - Guests join the meal.
  - If the meal is rescheduled:
    - Guest receives a notification with the new date/time.
    - Guests can choose to accept or opt out.
  - Guests may also cancel their participation before the meal.

## **3. Handling Payments (If required)**

- A decision checks if payment is needed.
- If yes:
  - User adds/selects a payment method.
  - Confirms payment.
  - If successful → meal proceeds.
  - If failed → notified and prompted to retry.
- If no → skip to meal ongoing.

## **4. Meal Ongoing and Completion**

- Meal enters the “ongoing” state.
- Once complete, the host marks the meal as done.
- The system then prompts for ratings based on roles:
  - Guests rate the host and the meal (categories: hospitality, organization, etc.)
  - Hosts rate guests (categories: punctuality, friendliness, etc.)
- Notifications are sent to remind users to submit feedback.

## 5. End of Flow

The flow ends after:

- Ratings are submitted or skipped.
- Users are taken back to the dashboard.

### Key Updates/Changes:

- Added meal rescheduling functionality for hosts
  - Hosts can update the date and time of a meal
  - Guests are automatically notified of changes and can choose to accept or opt out
- Implemented meal cancellation support for both hosts and guests
- Enhanced rating system with structured feedback categories
  - Guests are rated on aspects such as punctuality and friendliness
  - Hosts are rated on factors like hospitality and organization

## Challenges and Strengths

Through this milestone, the team encountered several technical and collaborative challenges while continuing to expand the application's core functionality. One of the primary challenges was resolving merge conflicts caused by multiple developers working on the same files simultaneously. In addition, updating the database schema to support new features such as meal requests, rescheduling, and cancellations required careful planning and coordination. The team also faced significant node package-related issues that disrupted development and introduced a variety of unexpected errors, which took time to resolve.

Despite these setbacks, the team remained focused and demonstrated strong problem-solving and collaboration throughout the milestone:

- Resolved complex merge conflicts through effective use of pull requests and communication
- Adapted to necessary database changes to support new request and scheduling features
- Overcame major package issues through debugging and dependency management

- Enhanced the application with new features including host rescheduling, guest opt-outs, and structured ratings
- Continued to build proficiency in React, contributing to improved development speed and component integration
- Leveraged task diversification to distribute work efficiently across different parts of the system

The team's consistent communication, flexibility, and growing familiarity with the tech stack were key to successfully integrating the new features and maintaining development momentum.

## Design Reflection

Over the course of this project, one of the most thoughtfully designed aspects of our system was the interaction model between hosts and guests. From meal creation and discovery, to request handling, rescheduling, and cancellation, we consistently worked to ensure that every flow felt intuitive, and user centered. A key example of this was the meal request system, which required clear communication between users, dynamic updates to the guest list, and real-time feedback. Our design allowed users to see their roles reflected in the interface with confirmations, and respond appropriately at each stage, creating a more engaging and functional experience.

As the project evolved, we saw our design mature from static lists and simple actions into a more event-driven, interactive platform. Early versions of the app were limited in scope and tightly coupled, but as we added features like real-time updates, role-based views, and category-based ratings, we began applying more modular and reusable components. We learned how important it is to design with change in mind, keeping our logic flexible and our UI adaptable. In terms of design patterns implemented within our code, we applied singleton and strategy patterns throughout our system. The instantiation of the SupaBase client throughout our code is singleton as the client creates an instance of the object upon entering the application's session. The strategy pattern is implemented throughout the system. When filtering meals in search, the meals shown are dynamically changed, and the algorithm is varied based on filtering inputs.