

SW Engineering CSC648/848

UMAMe

Section 04 | Team 03

Member	Role
Khalid Mehtab Khan	Team Lead
Dat Vo	Backend Lead
Anish Khadka	Frontend Lead
Renee Sewak	Scrum Master
Jacob Perez	GitHub Master

Application Link	http://3.14.254.41/
Code Link	https://github.com/CSC-648-SFSU/csc648-04-fall23-csc648-04-fall23-team03/tree/master/Milestones/M2

Milestone 2

11th October 2023

Revision ID	Revision Date	Revised By	Revision Contents:
1	9 October 2023	Khalid Khan Dat Vo Renee Sewak Jacob Perez Anish Khadka	Incorporated feedback from Milestone 1, which includes data definitions and functional requirements.

1. Data Definitions:







Primary Data Name	Definition	Usage
User	The main data structure. A registered user who signed up on the application, e.g., Anshul, anshul214, <u>anshul@mail.com</u> , social influencer	A user's account information like name, username, recipe lists is stored here.
Recipe	A post created by a user, containing information on how to cook a recipe, e.g., Vegetarian Pizza with Sun Roasted Peppers, Cheap and Easy Udon Soup, Restaurant Style Spaghetti	A user can write recipes that other users can view and comment on. A recipe post contains things like instructions and ingredients.
Comment	A comment that can be left on recipes.	Comments allow users to interact with each other as they view recipes.

Primary Data Name	Sub-Data
User	<ul style="list-style-type: none">• firstName - user's first name• lastName - user's last name• profession - user's declared profession. Will be displayed on their profile.• username - user's unique username. Will be displayed on posts and comments.• email - user's unique email address required for registration.• password - user account's password required for login, hashed.• profilePicture - user's image. Will be displayed along with their username. Max 60 mb. JPEG, PNG.• followers - other users that are following the current user.• following - other users that the current user is following.• recipes - recipe posts that this user has created
Recipe	<ul style="list-style-type: none">• ingredients - ingredients required to cook the recipe.• instructions - step by step instructions to follow while cooking the recipe.• nutritionalInfo - a recipe's nutritional information• recipeOwner - the recipe's creator• comments - the comments posted on the recipe page.• hearts - number of hearts that other users gave the recipe. Each user can give one heart to any one recipe.

	<ul style="list-style-type: none"> tags - identifiers given to the recipe by the recipe author. Allows other users to easily find specific recipes.
Comment	<ul style="list-style-type: none"> commentOwner - the user who left the comment. hearts - number of hearts that other users gave the comment. Each user can give one heart to a comment. content - actual text content of the comment

1. Functional Requirements:

	Must-have		Desired		Opportunistic
---	------------------	---	----------------	---	----------------------

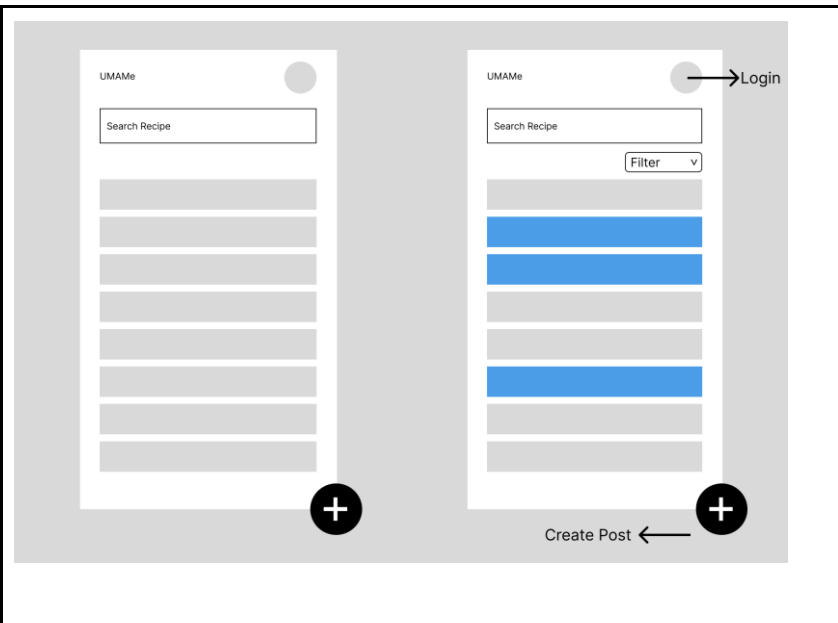
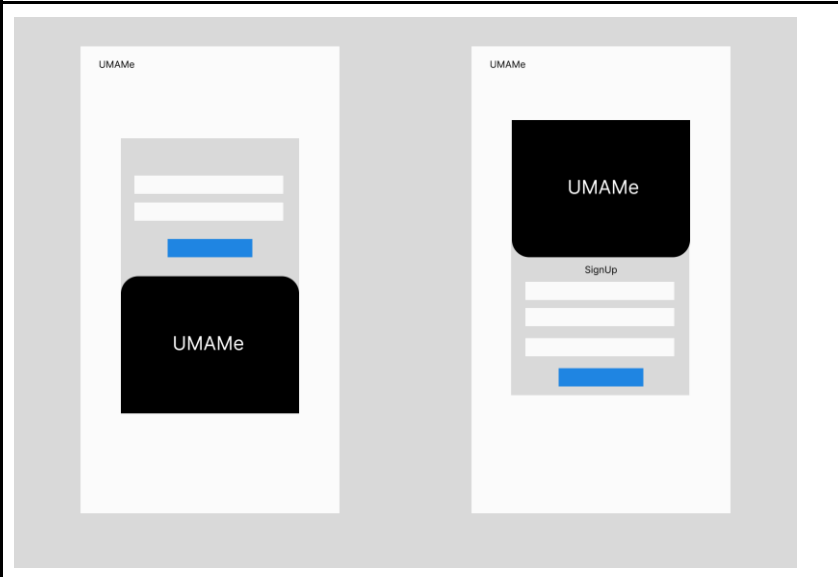
	ID	Description	Details
	01	Non-registered users can visit the website and view recipes and comments.	01.1) non-registered users cannot create recipes 01.2) Cannot leave comments 01.3) Cannot leave hearts
	02	Non-registered users can create an account.	02.1) Guest users can sign up for an account and log in for more privileges
	03	Registered users can create and post a recipe.	03.1) Registered users can write their recipe by giving it instructions, ingredients, and tags
	04	Registered users can share their recipes to others.	04.1) Creates a link to the user's recipe page
	05	Registered users can leave comments on recipes.	
	06	Registered users can leave hearts on recipes and comments.	06.1) Hearts are tallied up and displayed alongside a recipe's name 06.2) Comments have their own heart count

07	Registered users shall view statistics to find most popular recipes on the application and its distribution.	07.1) These statistics include most viewed, most commented, and most hearts
08	Any user can search for a recipe using a search bar.	08.1) Text search searches recipes by name and ingredients
09	Any user can filter for recipes using recipe tags for example ingredients, dishes, food type etc.	09.1) Filtering is done through recipe tags
10	All users can input a list of ingredients they have on hand, and the app will point to recipes they can cook with just those ingredients.	10.1) Recipes that require only the specified ingredients will be returned to the user
10	Users can see a complexity bar next to every recipe, indicating the skill required to create the recipe.	11.1) A complexity rating of 1 indicates a beginner friendly recipe. A max rating of 5 indicates a professional level recipe.
12	When viewing a recipe, users can enter cooking mode.	12.1) Step by step instructions are displayed in large fonts 12.2) Embedded cooking timer 12.3) Big, accessible button to move onto next instructions
13	Registered users can bookmark recipes and add them to a named collection.	13.1) Collections are shareable. 13.2) Can be made viewable by the public or private.

2. UI Mockup and UX Flows

a. UI Mockup

Mobile Device UI

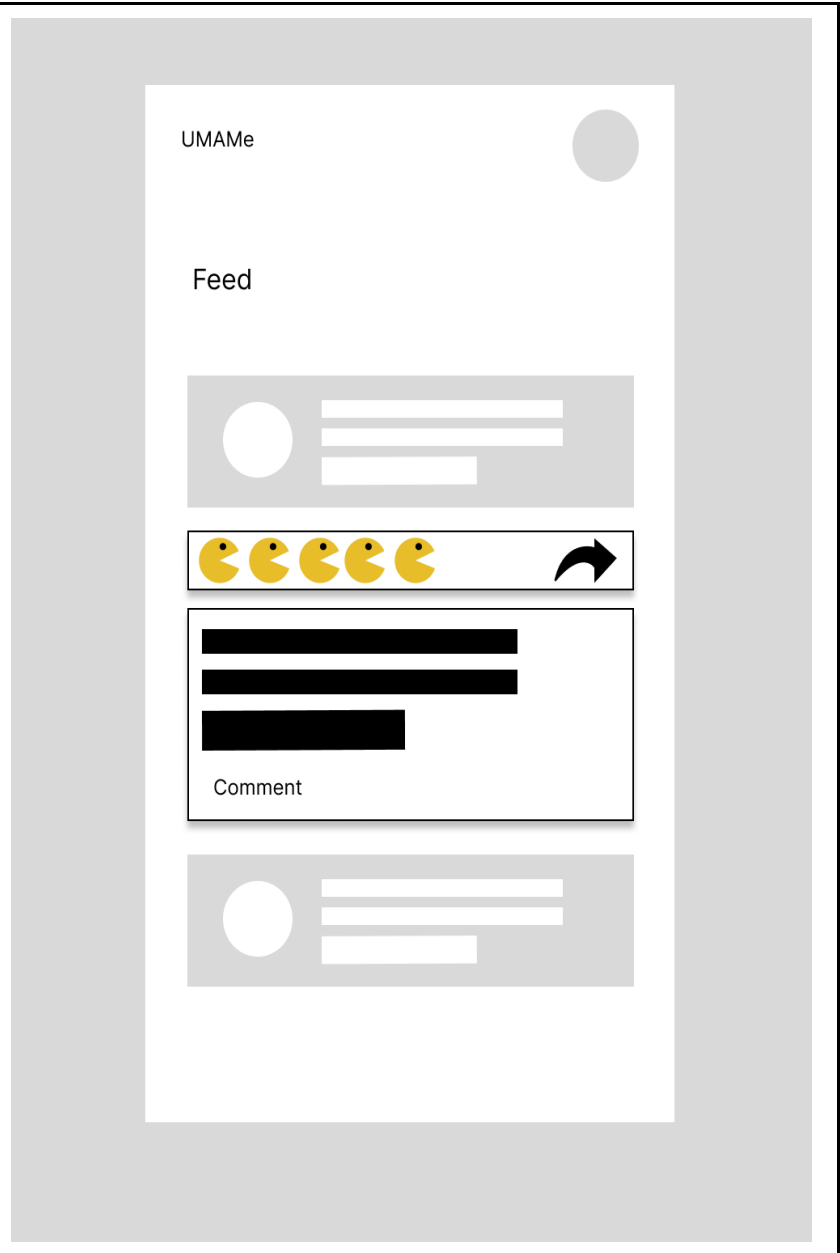
<p>Main Page:</p> <ul style="list-style-type: none">- Users can search and view recipes.- Users shall filter recipes according to the need.- Users can navigate to the login page. Users shall post new recipes.	 <p>The Main Page UI mockup shows two mobile device screens. The left screen displays a header with 'UMAMe' and a profile icon, a 'Search Recipe' input field, and a list of recipe cards. The right screen shows the same layout but with a 'Filter' dropdown menu open below the search field. A 'Login' button is indicated by an arrow pointing to the profile icon, and a 'Create Post' button is indicated by an arrow pointing to a plus icon at the bottom right. Both screens have a plus icon at the bottom center.</p>
<p>Login/ SignUp:</p> <ul style="list-style-type: none">- Users can login into existing accounts.- Users shall create new accounts using SignUp.	 <p>The Login/ SignUp UI mockup shows two mobile device screens. The left screen displays a login form with two input fields and a blue button, with a 'UMAMe' logo below. The right screen displays a sign-up form with a 'SignUp' label, two input fields, a blue button, and a 'UMAMe' logo above. Both screens have a white background with a gray border.</p>

Recipe Post:

- Users will be able to interact with posts.

- Users can

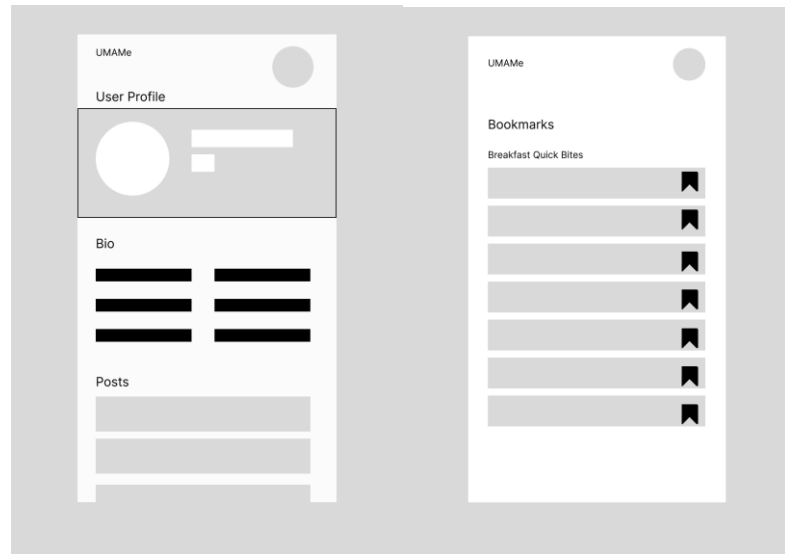
- Like
- Comment
- Share
- Bookmark



Profile:

- Users shall visit profiles of other users. Users can view and edit their profile.

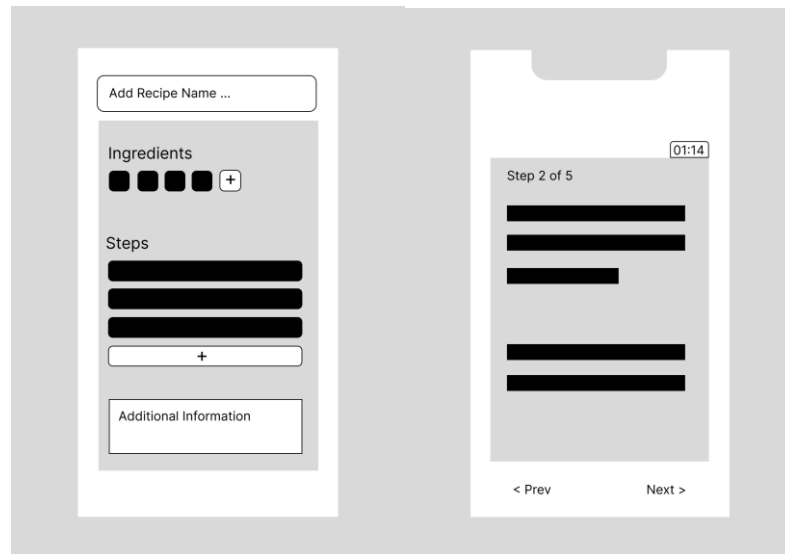
- Logged in users can see their posts and bookmarked recipes.



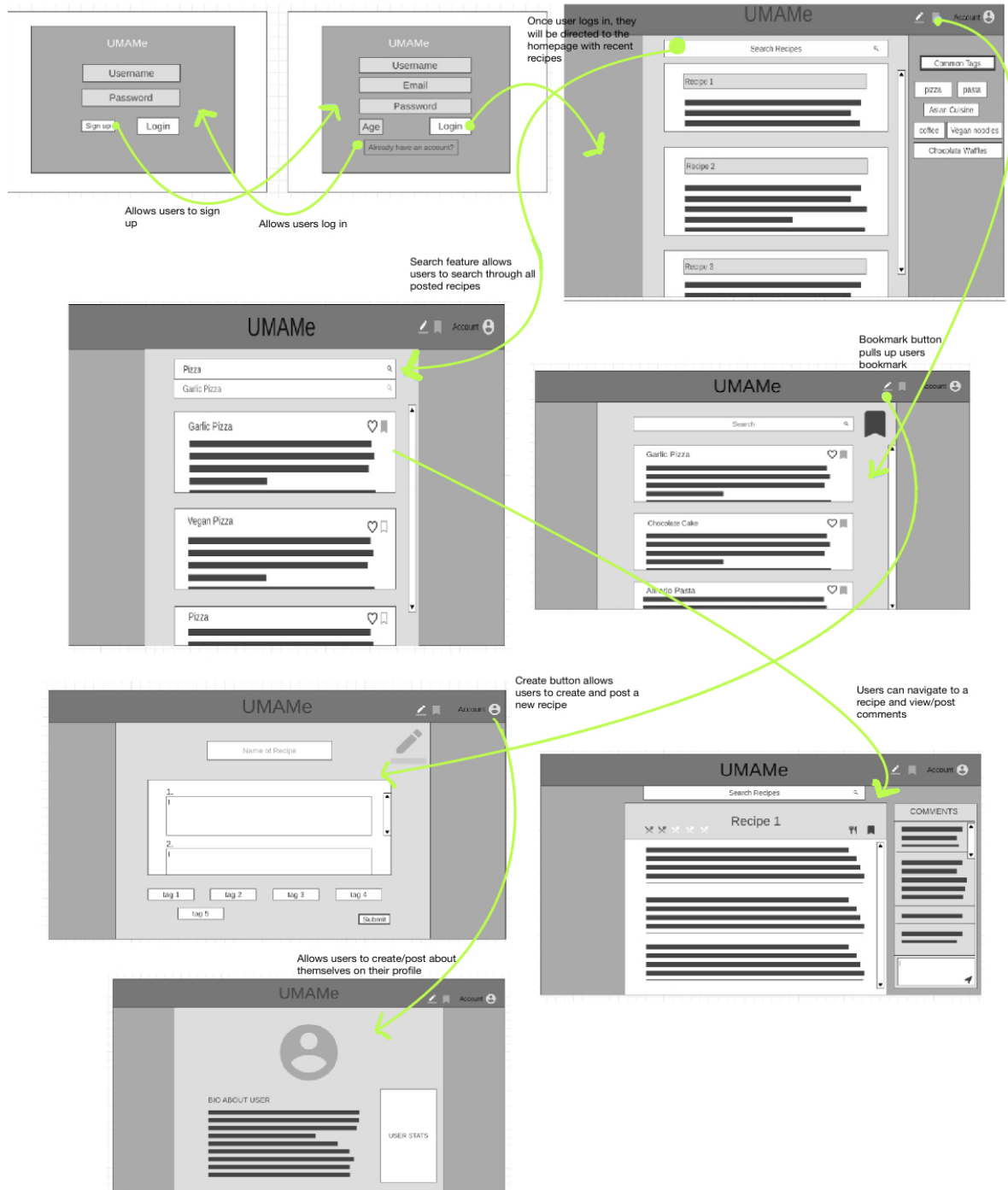
Cooking/ Recipe:

- Users shall create their own recipe and post it on the application.

- Users can enter the cooking mode of a recipe to follow instructions in a seamless manner to cook.



b. UX Flow



3. High Level Architecture, Database Organization

Databases	JSON-based Documents	Add/Search Architecture	APIs
User	userId: int32array firstName: String lastName: String username: String profession: String email: String password: String postedRecipes: Array (Recipe) savedRecipes: Array (Recipe)	Add/Delete/Search/Display Users can register and view recipes	MongoDB
Recipe	recipeId: String recipeName: String owner: String ingredients: Array (String) instructions: Array (String) nutritionalInfo: Array (String) hearts: int32array tags: Array (String) comments: Array (Comment)	Add/Search/Delete/Display Users can add more specific info and have comments	MongoDB
Comment	words: String hearts: int32Array owner: User	Add/Search/Delete/Display Users can leave comments	MongoDB

APIs:

Backend Endpoints	Description/Function
POST /users/	When a guest user attempts to create a new account on the signup page, The frontend will send user provided information to this endpoint. The backend will then attempt to add a new user into the database using the data received from the frontend.
POST /users/login/	When a registered user attempts to log into their account, the frontend will send the given credentials to this endpoint. The backend will then search the database to ensure the given credentials are valid for an existing user.
GET /home/	Every time the homepage is rendered, our frontend will send a get request to this endpoint. When it receives this request, the backends' job is to return a list of recipe names from the database. These names are used for the search bar's autofill feature.
POST /home/search/	When a user searches for a recipe on the homepage, the search term is sent to this endpoint. The backend directs the search to the database and returns any recipe that contains the term.

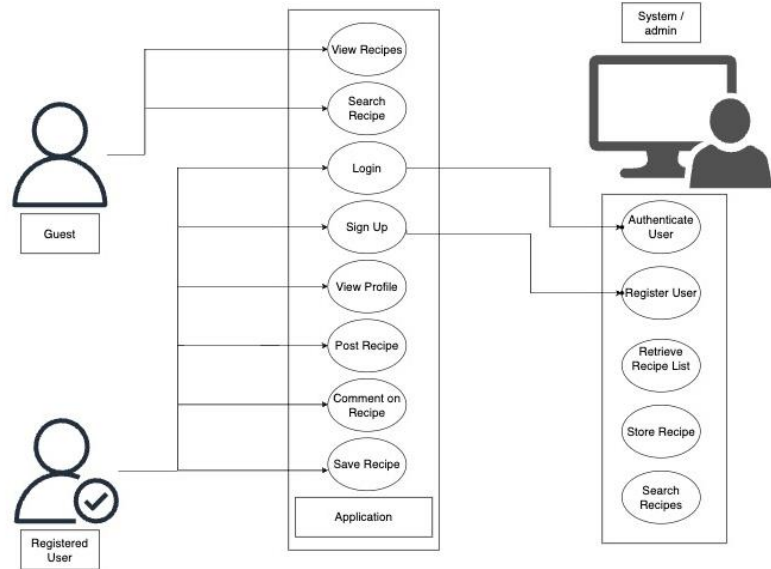
5. High Level UML Diagrams

Use Case Diagram:

Registered Users: Their abilities to create and post recipes, leave comments, save recipes, and search within the application.

Guest Users: They can view recipes but are limited in terms of interactions like commenting or posting recipes.

This distinction helps understand the user permissions and the journey a user might take from being a guest to becoming a registered member.

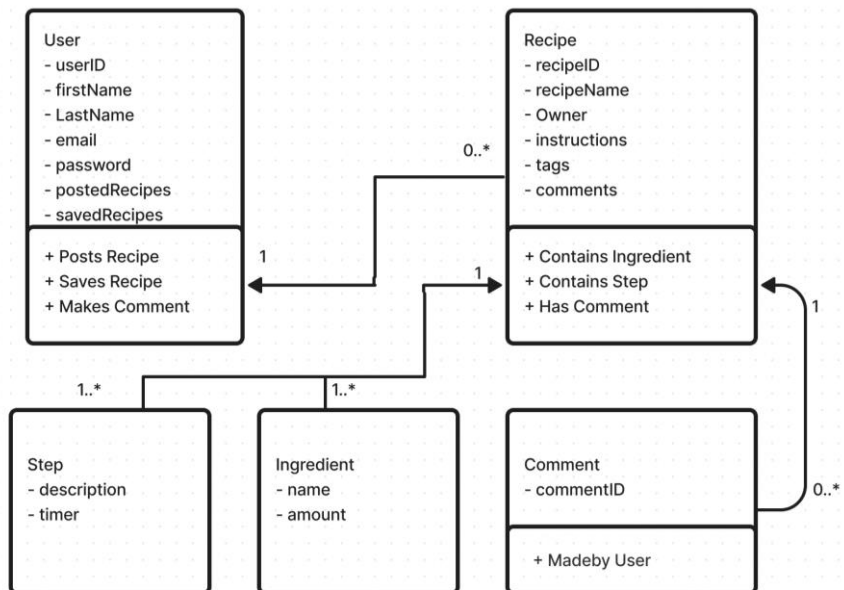


Class Diagram

User Class: With attributes like username, email, and lists of posted and saved recipes.

Recipe Class: Encompassing ingredients, instructions, and comments, this is central to the application. It shows how recipes relate to users (with owners) and comments.

Comment Class: Highlighting how users engage with recipes and indicating the relational aspect of how comments link back to both users (commenter) and recipes.

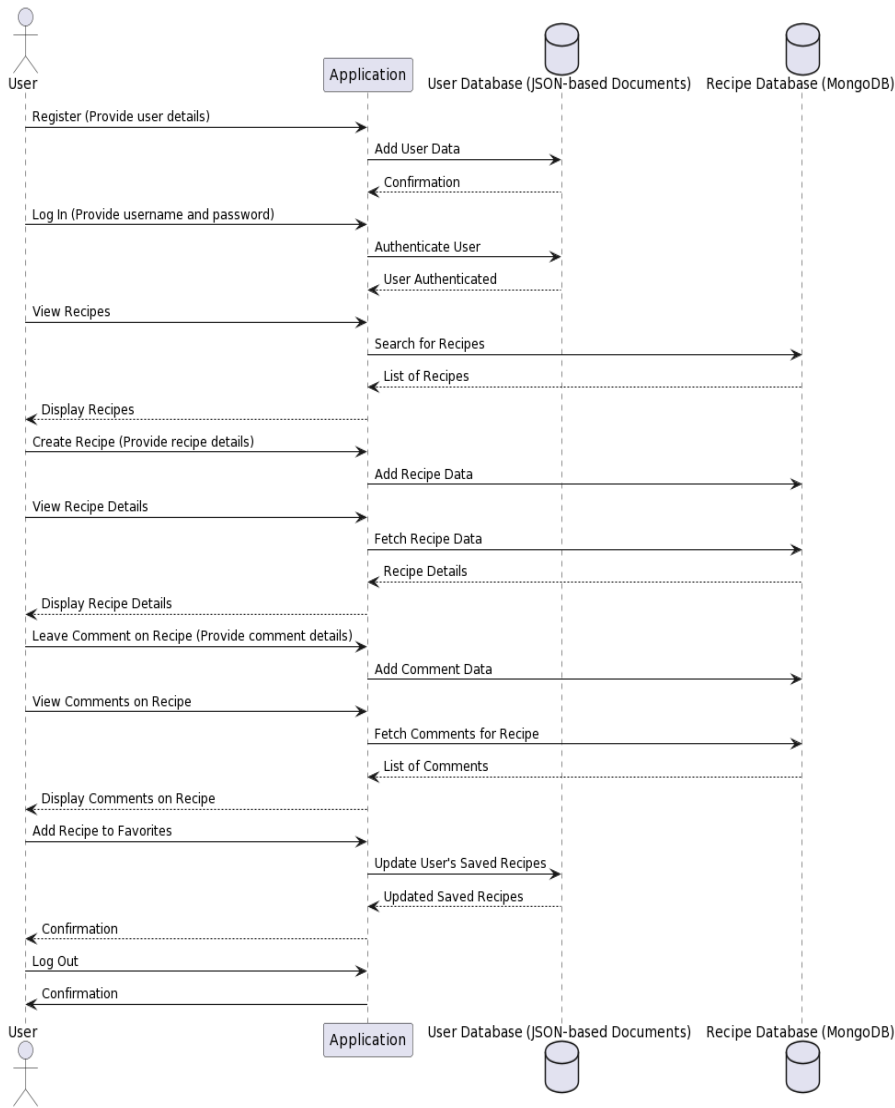


Sequence Diagram

Login Sequence: It depicts the steps a user undergoes from initiating a login request to receiving a confirmation from the system.

Guest Viewing: Demonstrates the streamlined process where guests view recipes without needing to engage deeper functionalities.

Recipe Posting: It showcases the user journey from drafting a recipe, adding ingredients and instructions, to finally posting it for others to see.



6. Key *Risks*

Scheduling risk

Due to our members' schedules, it's difficult for us to find a proper time for all of us to meet. With commuting, part-time jobs, and other classes we wanted to make sure we still prioritized this class. Our routine was to meet before class every Monday and Wednesday. However, we noticed that due to other classes, some members routinely could not make our meetings.

How we address the issue:

1. Introduced a new additional meeting time: M/W 2:00-3:30. This meeting has given the backend team the opportunity to meet on a consistent basis.
2. If a member cannot make a meeting, it is their responsibility to let the scrum master know so the scrum master can update them regarding meeting discussions and designated tasks.

Skill risk

It's common, and even normal, to encounter bugs/issues when working on an application of this size. We want to ensure that no individual feels as if they are working solo or don't have anyone to rely on.

How we address this issue:

1. Whenever we have a meeting, we go over any bugs or complications. A fresh set of eyes is always good, and it can do no harm consulting with other members when faced with a problem.
2. Consider different approaches instead of fixating on one approach!
3. Last resort: reach out to those outside of our group.

Other Skill risks

If you are assigned a task, you aren't completely sure how to handle, share with your teammates. Along the way, there will be lots of problems we've never dealt with. It is okay to ask for support or do extra research regarding the issue.

1. Before asking for help, understand what it is that your task is asking of you.
2. Create a rough draft/sketch of what it is you are tasked with.
3. Clarify with other teammates. If it something that they know how to do, ask for help.
4. If still iffy, reach out to the professor or T.A.

Teamwork risk

There is nothing wrong if two team members disagree or don't see eye to eye. In the real world, we will all encounter it. What is best practice is to consider both/all perspectives and handle the issue in a mature manner.

How we address this issue:

1. Hear both/all perspectives.
2. Talk to other teammates as to what the best route is.

3. Make sure to not take anything personal.
4. Individuals involved in the manner so act professionally.

7. Project Management

We've been able to gauge what works for us and what doesn't from previous meetings. From the professor's recommendation, our team follows a certain format to optimize each meeting. At the beginning of each meeting, each member answers three questions:

1. What have I worked on since the last meeting?
2. What do I plan to work on in this meeting?
3. What are some obstacles I am currently facing?

By doing so, each team member's progress is being shared. We are then able to gauge what still is high priority vs. what is low priority. For M2, we divided each task based on who did which tasks in M1 and their roles.

Our frontend team is composed of three individuals. So, we had our scrum master work on the UI Mockups/ UX flows since she worked on the wireframes for M1, as well as work on the Risks and Project Management since she's been keeping track of our previous meetings. Additionally, we had our Team Leader overseeing the entire Milestone since he has good exposure to frontend and backend. That way, even though he's on the frontend team, he would still be able to assist frontend or backend members.

Since our backend team is composed of two individuals, we want to make sure they have enough support. So, we have the two backend individuals work on data definitions, functionality requirements, and Database Organization since they worked on it previously in M1. Additionally, we had one of our frontend members work on the UML Diagrams so someone on our frontend team could get exposure to the backend while supplying our smaller backend team with support.

Our most common way of communicating is in-person meetings and Discord channels. Discord is something we are all highly familiar with, so we organize our server with multiple channels: backend, frontend, previous-meeting-topics, and general-questions.

To succeed, we must remain consistent and communicative. Additionally, we don't want to rely solely on certain individuals. Having secondary or even third strings just in case allows us to distribute the burden while still gaining and learning knowledge on certain aspects of the project we aren't as exposed to. Overall, our main motto is to share the burden - make sure everyone is being supported.