SOFTWARE REQUIREMENTS SPECIFICATION

RECIPE ROULETTE

Software Engineering

Bachelor's Degree in Bioinformatics

Course 2023-24

(Version 1.2)
(23-05-24)

Group member:

Alessandra Bonilla Salon
Maria Lopez Moriana
Carmen Samedi

Supervisor:

Daniel Soto Alvarez

ESCI upf.
School of International Studies

I. REVISION HISTORY

Name	Date	Sections Changed	Version
María/Alessandra/Carmen	9/05/2024	2: Add description + format changes.	1.0
María/Alessandra/Carmen	16/05/2024	2.2 Implement changes of review. 3.2 User case diagram	1.1
María/Alessandra/Carmen	23/05/2024	3.2. Detailed use case diagram, 3.3. Use case diagrams 2.1. Use case list 2.2. Use case description	1.2

II. TABLE OF CONTENTS

1. PROJECT DOMAIN **3**
1.1. PURPOSE 3
1.2. TARGET MARKET 3
1.3. PRODUCT SCOPE 3
1.4. REFERENCES 3
2. SYSTEM REQUIREMENTS **4**
2.1. USE CASE LIST 4
2.2. USE CASE DESCRIPTIONS 4
2.3. DETAIL USE CASE DIAGRAM 6
2.4. FUNCTIONAL REQUIREMENTS 8
2.5. NON-FUNCTIONAL REQUIREMENTS 13
2.6. EXTERNAL INTERFACE REQUIREMENTS 16
2.6.1. USER INTERFACES 16
2.6.2. SOFTWARE INTERFACES 16
2.6.3. COMMUNICATIONS INTERFACES 16
2.7. ASSUMPTIONS AND DEPENDENCIES 16
3. SYSTEM ANALYSIS **16**
3.1. PRODUCT PERSPECTIVE 16
3.2. SYSTEM FEATURES 17
3.3. OPERATING ENVIRONMENT 18
3.4. DESIGN AND IMPLEMENTATION CONSTRAINTS 18
4. DESIGN **19**
4.1. ARCHITECTURAL OVERVIEW 19
4.2. INTERFACE DESIGN 19
4.3. DATABASE AND DATA FLOW DESIGN 20

4.4. SECURITY PROTOCOLS 20

1. PROJECT DOMAIN

1.1. PURPOSE

The purpose of the "Recipe Roulette" app is to provide a user-friendly platform that provides a personalized cooking assistant that helps users utilize their available ingredients to discover and prepare new recipes. By doing so, the app aims to reduce food waste and enhance the cooking experience for individuals of varying culinary skill levels, from beginner cook to culinary expert.

1.2. TARGET MARKET

This Software Requirements Specification (SRS) document outlines the functional and nonfunctional requirements for " Recipe Roulette ", it provides a detailed overview of the application's features, intended use, and constraints. It serves as a guideline for the development team to design and implement the software and as a contract between the stakeholder and the development team, ensuring that the final product meets the users' and stakeholders' expectations.
The intended audience would include:
Project Managers: Who organize and oversee the project progression.
Developers: Use this document as reference for feature implementation of the app.
Quality control team: Who ensure the app meets the outlined specifications.
Users

1.3. PRODUCT SCOPE

Nearly a third of all the food (around 1.3 billion tonnes of food) produced each year is squandered or lost before it can be consumed. All this food produced but never eaten would be sufficient to feed two billion people. That's more than twice the number of undernourished people across the globe!
"Recipe Roulette" is a mobile and web application that aims to minimize food wastage and promote cooking at home by recommending recipes based on ingredients that users already have. It provides a holistic approach to cooking by facilitating recipe sharing, community building, and culinary education. It allows users to enter ingredients, apply filters for dietary preferences, and receive recipes that they can prepare. Additionally, users can contribute recipes, earn rewards for engagement, and partake in a community of cooking enthusiasts. It includes a comprehensive notification system for engaging users with new recipes and app updates.

1.4. REFERENCES

- https://www.wfp.org/stories/5-facts-about-food-waste-and-hunger
- Links to video recipes informing tips or visualizing the way to cook.
- IEEE Recommended practice for SRS
- User Interface Design Guidelines (SRS Template)

2. SYSTEM REQUIREMENTS

In the following section, there are the detailed description and corresponding diagrams of the use cases and the requirements.

2.1. USE CASE LIST

The use case model serves as a key tool in understanding and documenting how users interact with the "Recipe Roulette" system. This model helps to identify and define the roles of various actors and the fundamental processes they perform within the application. It provides a framework for analyzing the functional requirements by showcasing the sequences of actions between the user and the system to achieve specific goals. Through use cases, we capture user interactions that are essential for the system to fulfill its functionalities, ensuring that all user expectations are met systematically.

- UC1: Register Account
- UC2: Search Recipes
- UC3: Upload Recipe
- UC4: Gain Points with Recipe Interactions
- UC5: Local Database Management
- UC6: Search In Local Database
- UC7: Recipe Approval

2.2. USE CASE DESCRIPTIONS

Each use case identified in the diagram above is described in detail below, outlining the system's response to user actions:

Requirement	Use Case	Test
	UC1	TP-01
	UC2	TP-02
	UC3	TP-03

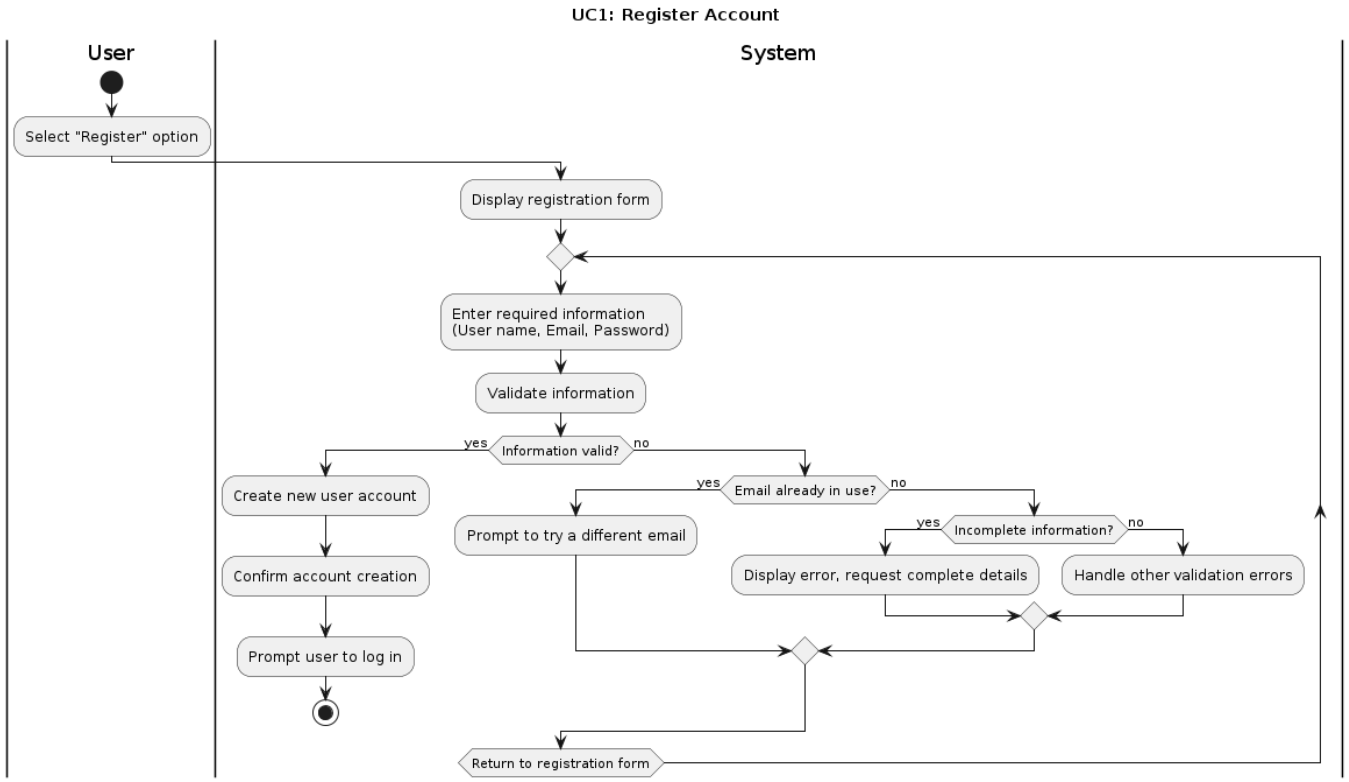
UC1: Register Account
Actor: Guest User
Description: Allows a guest user to register and create a personal account within the application.
Preconditions: The user must have valid personal information and access to the internet.
Postconditions: The user account is created and the user can log in to access personalized features.

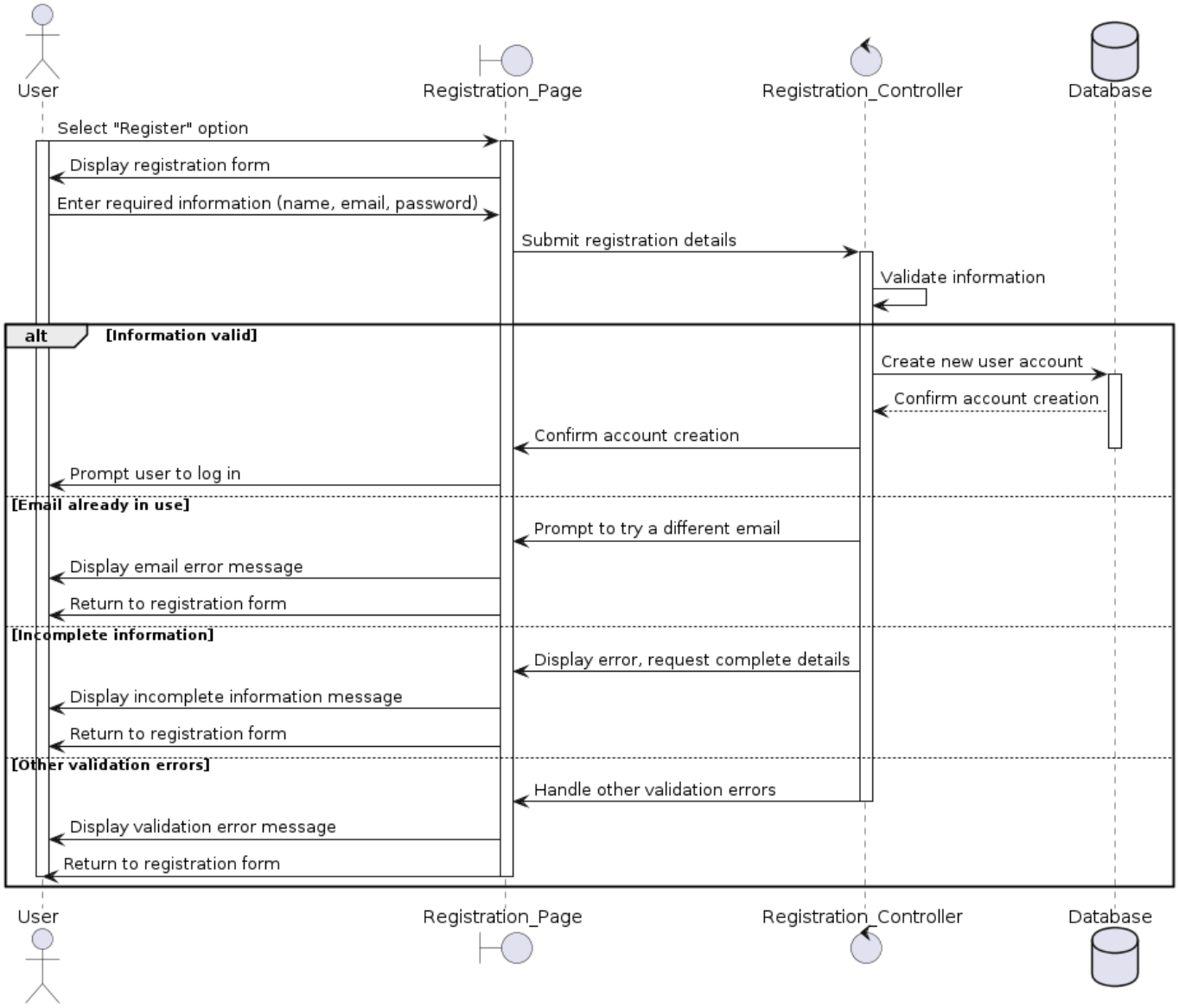
Basic Flow:

1. The user selects the "Register" option.
2. The user fills in the required information (e.g., name, email, password).
3. The system validates the information and creates a new user account.
4. The system confirms account creation and prompts the user to log in.

Alternative Flows:

- If the user enters an email that is already in use, the system prompts to try a different email.
- If the user provides incomplete information, the system displays an error and requests complete details.





UC2: Search Recipes

Actor: User (Registered)

Description: Allows users to search for recipes based on various criteria (ingredients, cuisine, dietary restrictions).

Preconditions: None.

Postconditions: Recipes matching the criteria are displayed.

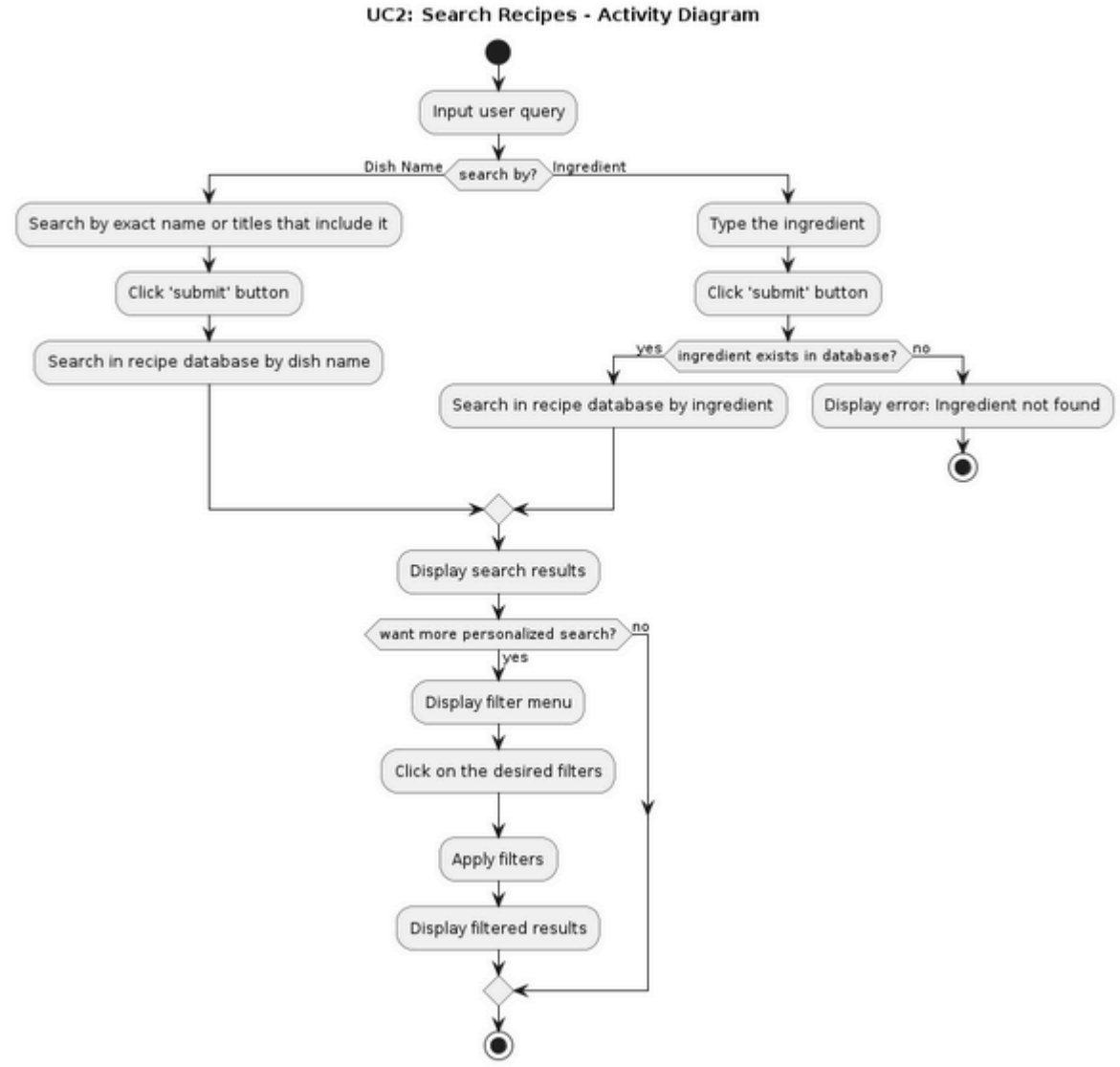
Basic Flow:

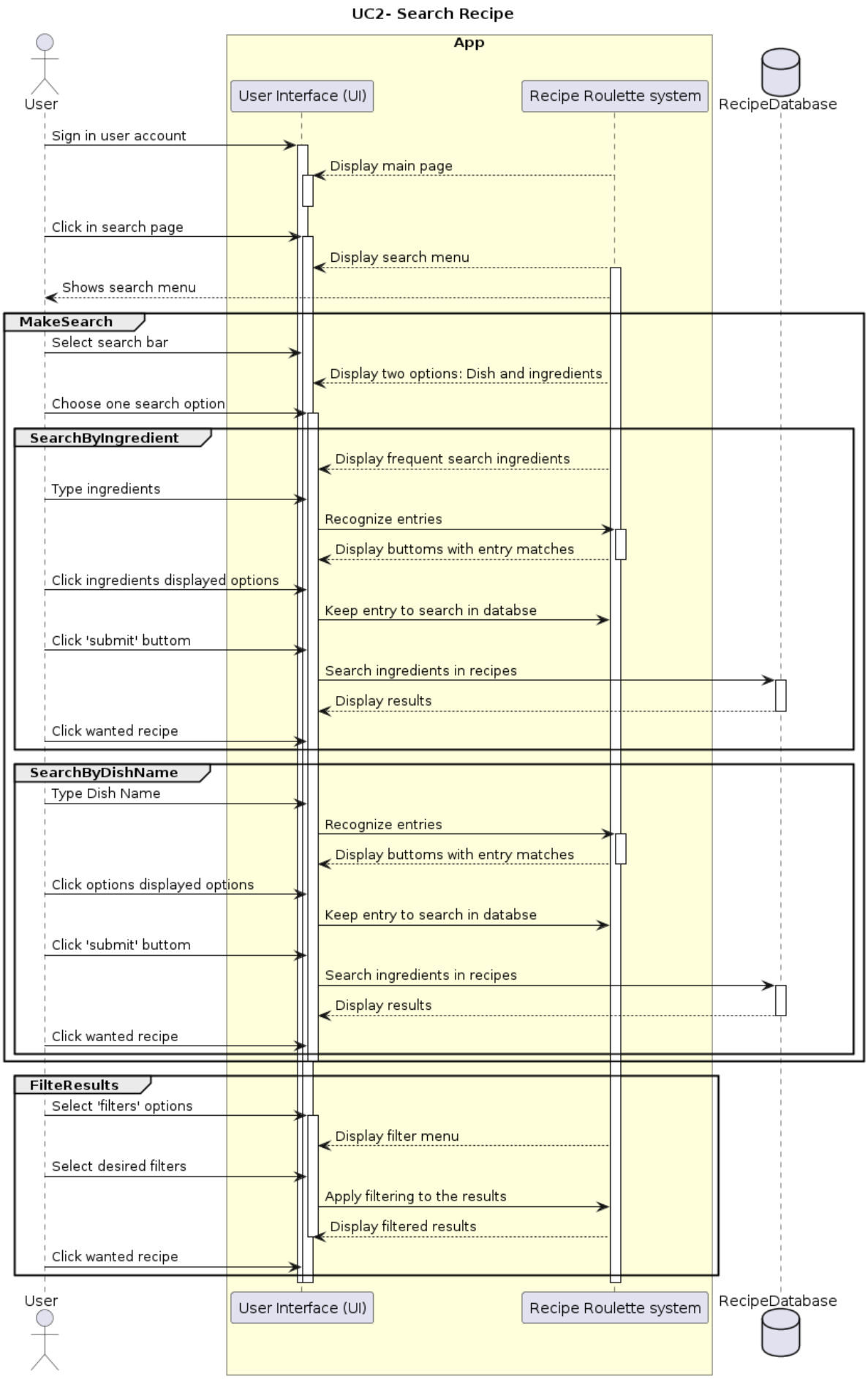
1. The user signs into their account.
2. The user goes to the search page.
3. The system displays the search page.
4. The user selects the search bar.
5. The user chooses to search by either dish name or ingredient.
6. The user types the dish name or ingredient and clicks the corresponding button.
7. The system checks if the ingredient exists in the database.
    - If yes, the system will search for exact matches in the ingredients of its database.
    - If no, the system will keep the entire word to find matches.
8. The user clicks the 'submit' button.

9. The system searches the recipe database and displays search results.
10. The user can choose to apply additional filters (e.g., time required, food type, dietary restrictions, difficulty level, allergies).
11. The system displays the filter menu.
12. The user selects and applies desired filters.
13. The system displays filtered results.

Alternative Flows:

- If no recipes match the criteria, the system displays a message indicating no results found and suggests altering search parameters.




UC2- Search Recipe
App
User
User Interface (UI)
Recipe Roulette system
RecipeDatabase
Sign in user account
Display main page
Click in search page
Display search menu
Shows search menu
MakeSearch
Select search bar
Display two options: Dish and ingredients
Choose one search option
SearchByIngredient
Display frequent search ingredients
Type ingredients
Recognize entries
Display buttoms with entry matches
Click ingredients displayed options
Keep entry to search in databse
Click 'submit' buttom
Search ingredients in recipes
Display results
Click wanted recipe
SearchByDishName
Type Dish Name
Recognize entries
Display buttoms with entry matches
Click options displayed options
Keep entry to search in databse
Click 'submit' buttom
Search ingredients in recipes
Display results
Click wanted recipe
FilteResults
Select 'filters' options
Display filter menu
Select desired filters
Apply filtering to the results
Display filtered results
Click wanted recipe
User
User Interface (UI)
Recipe Roulette system
RecipeDatabase

UC3: Upload Recipe
Actor: Registered User
Description: Allows registered users to upload their own recipes into the database.
Preconditions: User must be logged in.
Postconditions: The recipe is added to the database and available for others to view.

Basic Flow:

1. The user navigates to the "Upload Recipe" section.
2. The user enters the details of the recipe (name, ingredients, steps) and submits.
3. The system validates the entered data and adds the recipe to the database.
4. The system confirms the successful addition of the recipe.

Alternative Flows:

- If mandatory fields are missing, the system prompts the user to complete all required fields before submission.

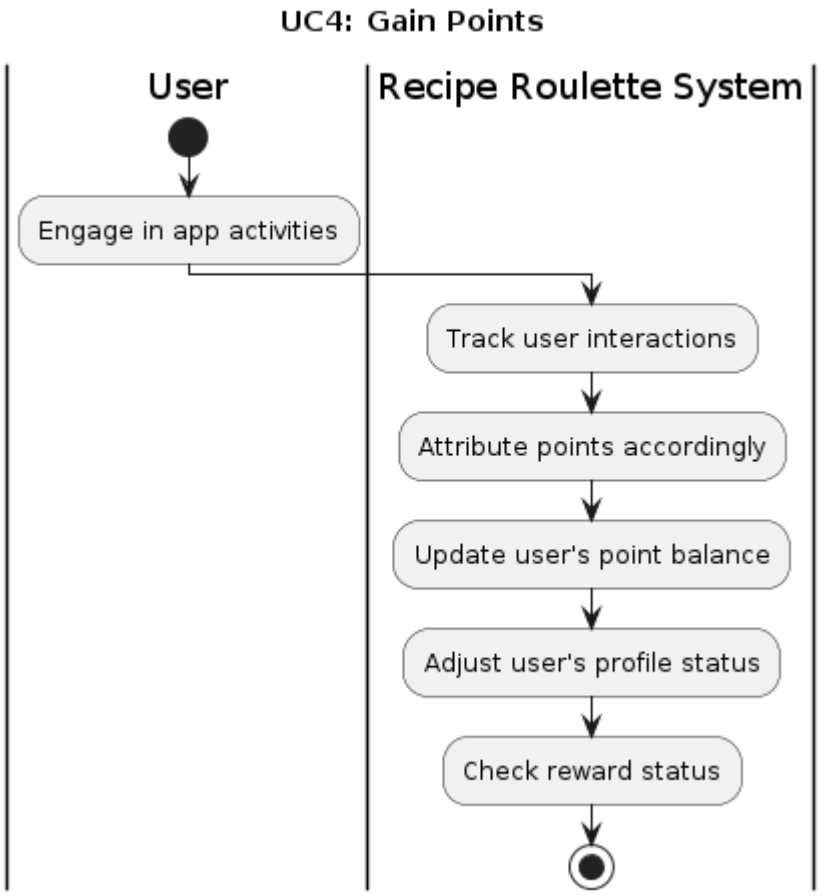
UC4: Gain Points with Recipe Interactions
Actor: Registered User
Description: Users earn points by engaging in app activities like uploading recipes, commenting, and interacting with posts (like and save recipes). These points increase the user's profile status, making them more visible to others. After gaining points, users can check their reward status to redeem earned points for rewards within the app.
Preconditions: User must be logged in, App is operational and tracks user interactions for point attribution.
Postconditions: User's point balance is updated, User's profile status is adjusted, User may check their reward status and redeem points for rewards.

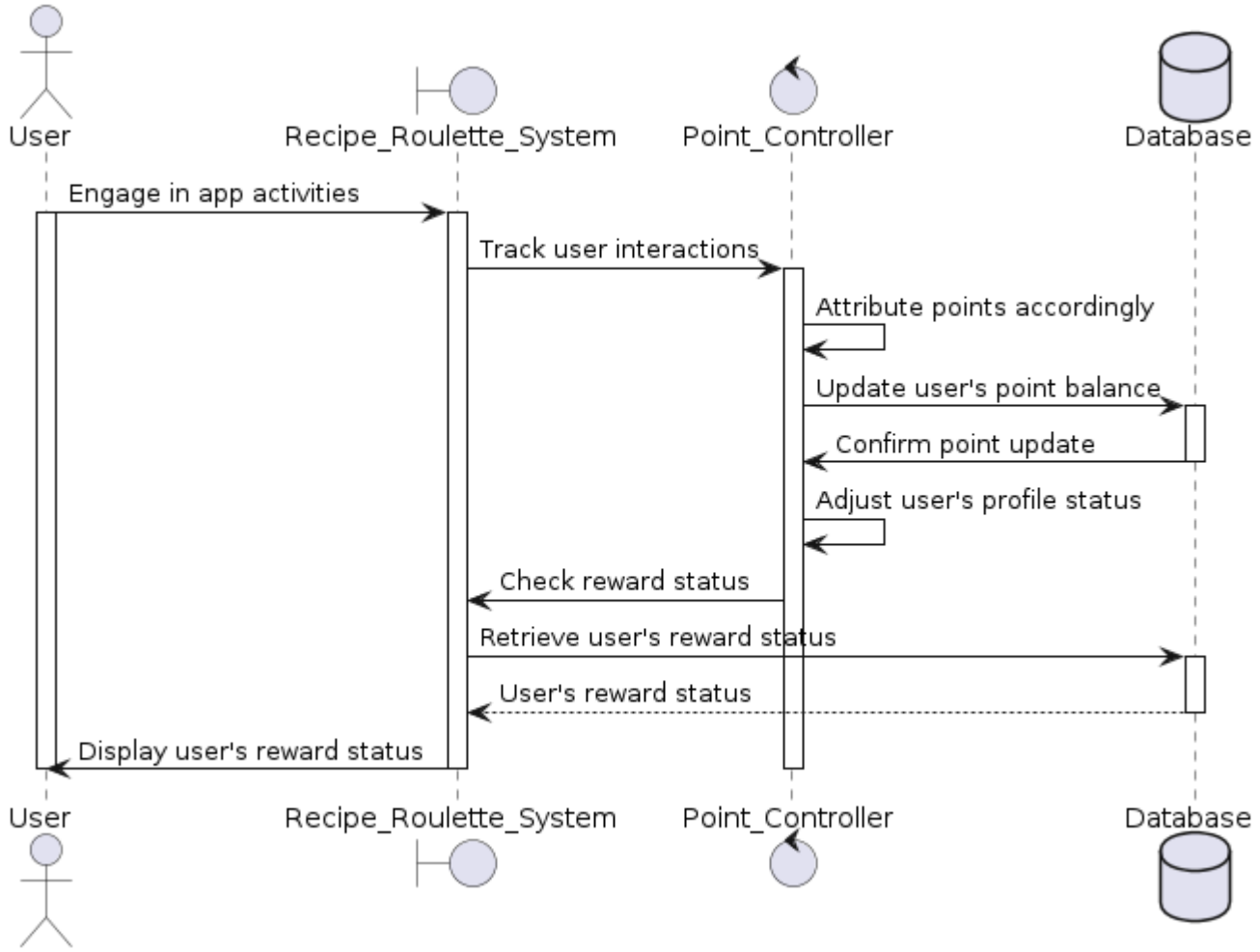
Basic Flow:

1. User engages in app activities such as:
    - ❖ Uploading recipes
    - ❖ Commenting on recipes or posts
    - ❖ Liking other recipes
    - ❖ Saving recipes
2. The app tracks the user's interactions and attributes points accordingly.
3. The user's point balance is updated in real-time within their profile.
4. The user's profile status is adjusted based on the accumulated points, increasing visibility to other users.

Alternative Flows:

- **Activity Limit Reached:** User reaches limit for certain activities within a specified time frame: daily upload limit for the recipes.
- **Technical Issues:** The tracking system can be interrupted, so points may not be updated real-time, once solved the issues then the user’s points should be updated accordingly.
- **Redemption of Points:** The gained points can be exchanged by some rewards; e.g. possibility of more daily uploads.


UC4: Gain Points
User
Recipe Roulette System
Engage in app activities
Track user interactions
Attribute points accordingly
Update user's point balance
Adjust user's profile status
Check reward status



UC5: Local Database Management
Actor:
Description:
Preconditions:
Postconditions:

Basic Flow:.

Alternative Flows:

UC6: Search In Local Database
Actor:
Description:
Preconditions:
Postconditions:

Basic Flow:.

Alternative Flows:

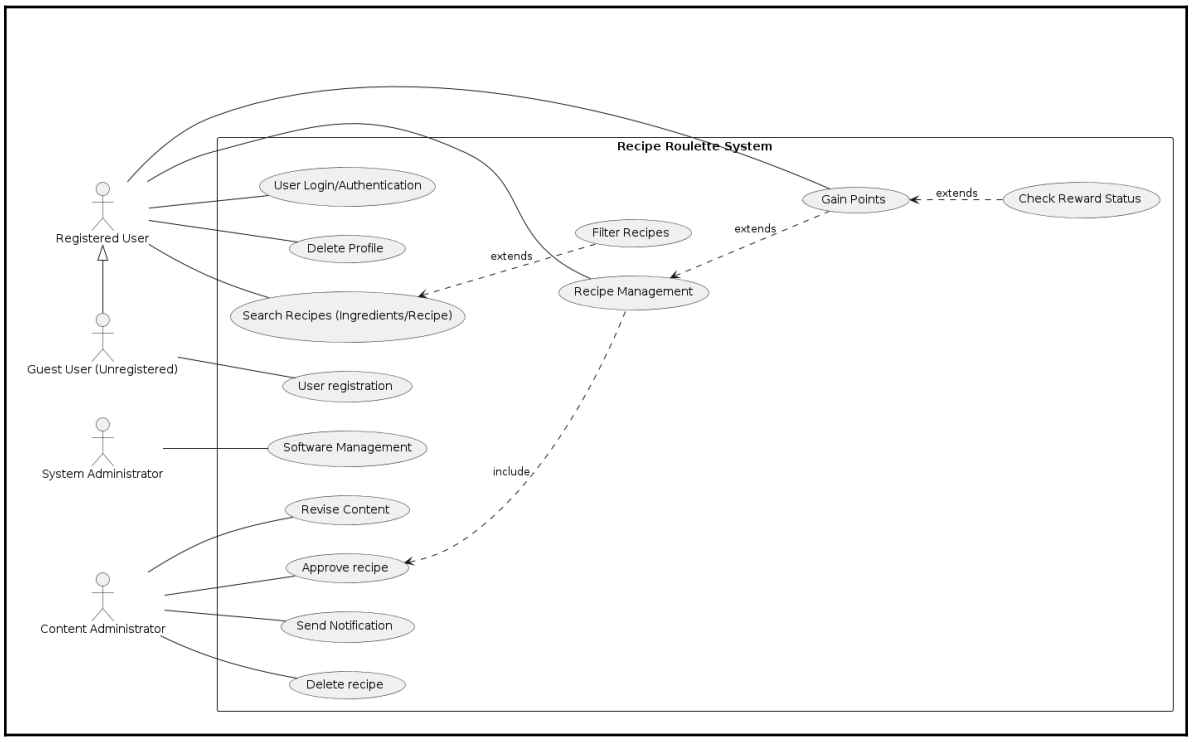
UC7: Recipe Approval
Actor:
Description:
Preconditions:
Postconditions:

Basic Flow:.

Alternative Flows:

2.3. DETAIL USE CASE DIAGRAM

Below is the detailed use case diagram for the "Recipe Roulette" application. This visual representation includes all user interactions, system responses, and the relationships between different use cases. The diagram shows how various features of the system interconnect and interact with the users:



The diagram includes several actors (Registered User, Guest User, System Administrator and content administrator) and use cases for each of the actors; e.g. A registered user can login the system, delete a profile, search for recipes (both searching by ingredients or by recipes) and filter them, they can also manage the recipe (upload, modify and delete them) by doing so they can gain points when providing content and therefore they should also be able to check the reward status in their profile.

In the following section, we provide a detailed description of all the requirements, from the functional, non functional and external. In the table below, we provide an overview of the app's requirements with each of its identifiers.

Functional Requirements	Non-functional requirements
UFR-1: User registration	NFR-001: Data Encryption
UFR-2: User Login/Authentication	NFR-002: Fast Response Time
UFR-3: Delete Profile	NFR-003: Easy Code Maintainability
UFR-4: Search Recipes by Ingredients	NFR-004: Large User Load Handling
UFR-5: Search Recipes by Dish Name	NFR-005: Intuitive Interface
UFR-6: Filter Recipes	NFR 006: Regular Backups
UFR-7: Recipe Management	NFR-007: Constant Availability
UFR-8: Gain Points	NFR-008: Cross Platform Compatibility
UFR-9: Check Reward Status	NFR-009: Performance Optimization
UFR-10: Local Database Management	
UFR-11: Search In Local Database	
MFR-12: Management of Software	
CFR-13: Approve recipe	
CFR-14: Send Notification	
CFR-15: Delete recipe	
CFR-16: Revise Content	

2.4. FUNCTIONAL REQUIREMENTS

The functional requirements of the Cooking App define the core functionalities and features that the system must provide to meet the needs of its users. The following tables provide comprehensive descriptions of each functional requirement, highlighting their specific purpose, expected behavior, and the relationships with other system components.

UFR-1	**User registration**	**Version: v1.1**
Description: Users access registration, enter personal info including name, email, phone, address, gender, age, and select role (Beginner, Intermediate, Chef). Role-specific info is then provided. Password creation and confirmation complete registration. First login verifies info for account creation		
Comments: If the person selects the role wrongly they can go back to select a different role. The password should follow the following requirements: - 8 to 10 characters - A combination of uppercase letters, lowercase letters, numbers, and symbols. If the information is invalid the user will return to the registration page where they have to enter their personal information to correct any errors. Some Role-Specific information are: Beginner, Intermediate, Chef		
Relationships: UFR-2, UFR-3, MFR-12, CFR-14, CFR-13		

UFR-2	**User Login/Authentication**	**Version: v1.0**
Description: Users log in using their Username / email and Password		
Comments: If user fails to login after multiple attempts there could be two-factor authentication There could be a security measure so as to prevent hacks by blocking accounts after multiple failed login attempts.		
Relationships: UFR-1		

UFR-3	**Delete Profile**	**Version: v1.1**
Description: Users can delete their profile and associated data by: Going to settings and Selecting 'Delete Profile' then confirming the decision twice to avoid mistakes After confirmation, the system deletes all data.		
Comments: To avoid permanently deleting all the information and progress of a user in the app by accident, we would add a warning as well as remind the user of the consequences of deleting the app accordingly.		
Relationships: UFR-2 (parent), UFR-1 (parent)		

UFR-4	Search Recipes by Ingredients	Version: v1.0
Description: Users search for recipes by entering available ingredients. They select ingredients from a list, press 'Search', and navigate through results that meet their requirements.		
Comments: Results should match with different names of ingredients (plural…) and not give back errors. If the ingredient is rare, the results should print a message that tell the user the that ingredient was not included in the results.App should be able to provide suggestions if the ingredient is rare. Users should be able to save recipes or download them to be able to access them off-line. We should monitor system performance to be able to search multiple ingredients at once. If no results are given the app could propose different related dishes or recipes.		
Relationships: UFR-1 (parent)		

UFR-5	Search Recipes by Dish Name	Version: v1.0
Description: Users search for recipes by entering the dish name. They then navigate results to find what they need.		
Comments: Users should be able to filter the recipes according to dietary restrictions or preferences. Users should be able to save recipes or download them to be able to access them off-line. As with recipes, results should be obtained based on different queries (different languages..) as well as take into account possible errors in the query. If no results are given the app could propose different related dishes.		
Relationships: URF-1 parent, UFR-6		

UFR-6	Filter Recipes	Version: v1.0
Description: Users can filter recipes based on → Time required, Food type (e.g., Indian, Mediterranean) ,Dietary restrictions (e.g., vegetarian, vegan), Difficulty level or Allergies (e.g., nuts, gluten)		
Comments: The user should be able to select from various different options.		
Relationships: URF-1 parent, UFR-6		

UFR-7	Recipe Management	Version: v1.0
Description: Users can upload, modify, and delete recipes on the platform. They input: **Recipe Upload:** Users upload recipes and key ingredients for easy searchability. **Dietary Restrictions:** Users specify dietary restrictions the recipe can or cannot follow.		
Comments: The user should be able to delete a recipe if wanted, and to avoid problems it would need confirmation. If user violates possible dangerous information the recipe should be deleted		
Relationships: UFR-1 (parent), UFR-8, CFR-13, CFR-14		

UFR-8	Gain Points	Version: v1.0
Description: Users earn points by engaging in app activities, which can be redeemed for rewards or discounts. Activities include: **Uploading Recipes:** Earn points for sharing recipes with the community. **Community Participation:** Get points for active engagement like commenting or liking. **Referrals:** Gain points by referring friends who sign up and use the app.		
Comments: The point system should be transparent, allowing users to understand how they earn points and what they can be redeemed for. The app should have a mechanism to prevent abuse or fraudulent behavior, such as spammy uploads or excessive referrals. Points should be trackable within the user profile, and users should have access to their point balance at all times. Points may have an expiration period to encourage continued engagement, with a notification system to alert users about upcoming expirations.		
Relationships: UFR-9, UFR-7, UFR-4, UFR-5, UFR-6, MFR-12		

UFR-9	Check Reward Status	Version: v0.1
Description: Users can track their points and view rewards in the app. The interface displays: **Point Balance:** Current points accumulated. **Rewards Catalog:** List of available rewards and their point costs. **Redemption:** Method to redeem points for rewards with clear instructions. **Point History:** Detailed transaction history, including earned, redeemed, and expired points.		
Comments: The rewards interface should be user-friendly and error-proof, with real-time balance updates and scalable features to accommodate growth and new rewards. Notifications can alert users about new rewards or expiring points.		
Relationships: UFR-8, UFR-7, MFR-12 , CFR-14		

UFR-10	Local Database Management	Version: v1.0
Description: This feature allows users to save their favorite recipes to their profile by clicking 'save'. A local copy of the recipe will be created on their device, enabling offline access and faster local searches. Users can initialize this feature when saving their first recipe, and can add or remove recipes as needed		
Comments: This functionality enhances user experience by providing offline access and quicker local searches. Users should ensure their local database is synced periodically to update any changes made while offline		
Relationships: UFR-8, UFR-11		

UFR-11	Search In Local Database	Version: v1.0
Description: This feature allows users to search for recipes stored in their local database. Since the recipes are stored locally, the search process is faster as it doesn't require network access or remote database queries. Users can quickly find and access their saved recipes offline.		
Comments: The local search is optimized for speed and efficiency, providing a seamless user experience even without internet connectivity. Ensure that the local database is regularly synced with the remote database to keep the saved recipes updated.		
Relationships: UFR-10		

MFR-12	Management of Software	Version: v1.0
Description: To ensure app performance and longevity, tasks include: Software Updates: Security and compliance Performance Monitoring: Identifying and fixing issues Error Handling: Swift response to errors Scalability: Efficient user load management Data Backups: Ensuring data integrity Security Measures: Protecting user data"		
Comments: The management of software should be proactive, focusing on preventing issues rather than just responding to them. The software management team should have clear roles and responsibilities, with processes in place for escalations and emergency response. Documentation should be maintained for all maintenance activities, including software updates, bug fixes, and performance reports.		
Relationships: NFR-001-PE, NFR-002-US, NFR-003-REL, NFR-005-SE, NFR-006-SC, NFR-007-SU		

CFR-13	Approve recipe	Version: v1.0
Description: The Content Administrator reviews submitted recipes to ensure they meet the app's quality standards, guidelines, and policies. This involves checking for complete and accurate information, compliance with community guidelines, and unique content. Approved recipes are marked for publication, while rejected ones receive feedback for improvement. Possible errors include incomplete information, guideline violations, duplicate content, and technical issues		
Comments: The approval process should be efficient to minimize delays in publishing user-submitted recipes. Content Administrators should have clear guidelines for recipe approval and consistent criteria for evaluating submissions		
Relationships: CFR-14, UFR-11, UFR-7		

CFR-14	Send Notification	Version: v1.0
Description: The Content Administrator sends notifications to users for different purposes, such as informing them about uploaded recipes that don't meet guidelines, requesting modifications, suggesting corrections, and notifying about rewards, points expiration, and app updates. This involves generating specific notifications and sending them to users via the app's messaging system.		
Comments: Notifications for rejected content should be clear, constructive, and personalized, with specific reasons for rejection and suggestions for improvement. Users should have a reasonable time frame to respond. The Content Administrator should have tools to track notification statuses and ensure follow-up actions.		
Relationships: CFR-13, UFR-1, MFR-12, CFR-15, UFR-9 , UFR-8		

CFR-15	Delete recipe	Version: v1.0
Description: The Content Administrator has the ability to delete recipes from the app, typically in response to violations of community guidelines, user requests, or inactivity. This function ensures the integrity and quality of content within the app. The process of deleting a recipe involves: - **Identify the recipe:** The Content Administrator identifies the recipe to be deleted, either from user reports, guideline violations, or system audits. - **Review and justify deletion:** Before deleting, the Content Administrator must review the reason for deletion to ensure it is justified and consistent with the app's policies. - **Delete the recipe:** Once the justification is confirmed, the recipe is deleted from the database. This removal is typically irreversible. - **Notify the user:** If the deletion is due to a guideline violation or user report, the Content Administrator sends a notification to the user explaining the reason for deletion.		
Comments: Deleting a recipe should be done with caution, ensuring that the reason for deletion is clear and justified. The Content Administrator should maintain a record of deleted recipes for auditing and accountability purposes. User notifications regarding deletions should be clear and provide guidance on next steps or appeal processes.		
Relationships: CFR-13, CFR-14, MFR-12, UFR-7		

CFR-16	Revise Content	Version: v1.0
Description: The Content Administrator is responsible for reviewing and editing existing content within the app to ensure accuracy and quality. This involves identifying content that needs revision based on user feedback, audits, or guideline changes. Once identified, the administrator revises and updates the content to correct errors, update information, or improve clarity. It's important to maintain a record of these revisions for auditing purposes. Additionally, if the revisions have a significant impact on users, the administrator communicates the changes, providing reasons and additional information for transparency.		
Comments: Revisions should be handled carefully to ensure they do not negatively impact user experience or introduce new errors.		
Relationships: CFR-14, CFR-15 , MFR-12, UFR-7		

2.5. NON-FUNCTIONAL REQUIREMENTS

The non-functional requirements of the Cooking App outline the essential criteria that determine the overall quality and performance of the system. These requirements address aspects such as performance, security, maintainability, and scalability, ensuring that the app operates reliably and efficiently under various conditions. The following sections detail each non-functional requirement, specifying the standards and guidelines that the system must adhere to in order to meet the expectations of both users and stakeholders.

NF-01	**Data encryption**	**Version: v1.1**
Type /Subtype: Security **Description:** All sensitive user data, including passwords, personal information, and recipe details, should be encrypted both during transit and at rest to prevent unauthorized access or data breaches.		
Comments: Utilize industry-standard encryption algorithms (e.g., AES) and protocols (e.g., TLS) to ensure the confidentiality and integrity of user data. Implement secure storage mechanisms and key management practices to protect encryption keys from unauthorized access. Regularly audit and update encryption protocols to address emerging security threats and vulnerabilities.		
Relationships:		

NF-02	**Fast response time**	**Version: v1.1**
Type /Subtype: Performance Optimization **Description:** Ensure quick system responses to user actions, optimizing database queries, image loading, and network requests for a seamless user experience.		
Comments: Prioritize efficiency to maintain user engagement and satisfaction. Regularly monitor and update performance to accommodate increasing user loads and evolving system demands		
Relationships: NFR-001-PE, NFR-004-MT, NFR-006-SC		

NF-03	**Easy Code Maintainability**	**Version: v1.1**
Type /Subtype: Maintainability **Description:** The codebase should be well-structured, documented, and adhere to coding best practices to facilitate easy maintenance by developers. Changes or updates to the code should be straightforward and require minimal effort.		
Comments: Implement modular design patterns and coding conventions to promote readability and ease of understanding. Document code comprehensively, including comments and documentation files, to aid future development efforts. Use version control systems (e.g., Git) and issue tracking tools to manage code changes and collaboration among development teams.		
Relationships: NF-01		

NF-04	Large User Load Handling	Version: v1.1
Type /Subtype: Performance **Description:** The application should be capable of efficiently handling a large number of concurrent users without experiencing performance degradation or downtime.		
Comments: Implement scalable architecture and optimize server resources to accommodate spikes in user traffic. Utilize load balancing techniques, caching mechanisms, and horizontal scaling to distribute the load evenly across multiple servers. Conduct stress testing and performance tuning to identify and address bottlenecks before they impact user experience.		
Relationships:		

NF-05	Intuitive Interface	Version: v1.1
Type /Subtype: Usability **Description:** The user interface (UI) of the application should be intuitive and easy to navigate, allowing users of varying levels of technical expertise to interact with the app effortlessly.		
Comments: Conduct user experience (UX) research and usability testing to identify user preferences and pain points. Design the UI with clear navigation paths, intuitive layout, and consistent design elements. Provide informative feedback and guidance to users to help them accomplish tasks efficiently. Continuously gather user feedback and iterate on the interface design to improve usability over time.		
Relationships:		

NF-06	Regular Backups	Version: v1.1
Type /Subtype: Reliability **Description:** The application should perform regular backups of all critical data to prevent data loss in the event of system failures, disasters, or other unforeseen incidents.		
Comments: Establish automated backup processes to regularly archive user data, configuration settings, and application state. Store backups in secure and redundant storage locations to ensure data integrity and availability. Implement backup monitoring and alerting mechanisms to promptly address any backup failures or issues.		
Relationships:		

NF-07	Constant Availability	Version: v1.1
Type /Subtype: Reliability **Description:**		

The app must maintain an uptime of 99.9% to ensure it is available to users at all times. This includes implementing redundant systems, regular backups, and robust disaster recovery plans to minimize downtime and ensure continuous operation.
Comments: Continuous monitoring and alert systems should be in place to detect and address any issues promptly. Scheduled maintenance should be communicated to users in advance to minimize disruption.
Relationships:

NF-08	**Cross-Platform Compatibility**	**Version: v1.1**
Type /Subtype: Design Constraints **Description:** The app must be compatible with both iOS and Android platforms. It should provide a consistent user experience across different devices and screen sizes. This ensures that users can access the app seamlessly regardless of the device they are using.		
Comments: The app must be compatible with both iOS and Android platforms. It should provide a consistent user experience across different devices and screen sizes. This ensures that users can access the app seamlessly regardless of the device they are using.		
Relationships: UFR-1 to UFR-9		

NF-09	**Performance Optimization**	**Version: v1.1**
Type /Subtype: Design Objectives **Description:** The app should be optimized for performance, including efficient database queries, optimized image loading, and minimized network requests. Caching mechanisms should be used where appropriate to enhance speed and reduce latency. This optimization ensures a fast and responsive user experience, even under heavy load conditions		
Comments: Performance optimization should be an ongoing process, with regular monitoring and updates to ensure continued efficiency as the app scales and user numbers increase.		
Relationships: UFR-4, UFR-5, UFR-6, etc		

2.6. EXTERNAL INTERFACE REQUIREMENTS

Description of the interfaces between the software and the world including user, ~~hardware,~~ software, and communication interfaces.

2.6.1. USER INTERFACES

App that should be compatible on both Android and IOS as well as all major web browsers for the web interface, so as to make the app available for all kinds of mobile devices.

The features should be able easy to read and use and to accommodate for all kinds of needs (different font or languages)

2.6.2. SOFTWARE INTERFACES

Our database would have all the input recipes users put as well as external recipes if possible.
We could integrate social medial so as to share recipes and attract more users

2.6.3. COMMUNICATIONS INTERFACES

App notifications to follow up on progress for each user and propose new recipes
Email for communication between app services and confirming accounts.

2.7. ASSUMPTIONS AND DEPENDENCIES

User Base Growth: The app's success is dependent on active user growth and engagement.
Ingredient Database: Access to a comprehensive and updatable ingredient and recipe database.
Technology Adoption: Assumption that users have access to mobile devices or computers with internet.

3. SYSTEM ANALYSIS

3.1. PRODUCT PERSPECTIVE

"Recipe Roulette" is introduced as a new application in the culinary space, distinct from existing products. Its main feature is to recommend recipes based on user-inputted ingredients, facilitating meal planning and cooking for individuals. The app is standalone and does not serve as a continuation of or replacement for any other product.

At its inception, "Recipe Roulette" will operate independently. Future updates may, however, extend its functionality to include integration with online grocery services and smart kitchen devices to further aid the cooking process. Additionally, features to share content on social media may be added to foster a sense of community among users.

The application will source recipe information from an assortment of databases and APIs. It may also offer insights into user preferences and behaviors to those conducting market research or analytics, pending the activation of such features.

3.2. SYSTEM FEATURES

3.2.1. SF 1: RECIPE DISCOVERY ENGINE

This discovery engine is the core feature of the app, since it allows users to input the ingredients and obtain recipe suggestions that would be feasible for the ingredients in their fridges or pantries.
Functional Requirements:
The engine must allow users to enter multiple ingredients as search criteria.
The engine must provide recipe suggestions that utilize the inputted ingredients.
The engine should offer the option to save favorite recipes for later reference.

3.2.2. SF 2: USER ACCOUNT MANAGEMENT

This management handles user registration, modifications and profile customization, and is in charge of data manipulation within the app.

Functional Requirements:

Users must be able to create a new account using an email address.
Users must be able to edit their profile information, preferences and allergies.
The system must provide a secure method for users to delete their accounts and related data.

3.2.3. SF 3: DIETARY AND CUISINE FILTERING

This filter feature lets users apply specific filters to the recipe searches, catering to various dietary needs, like allergies or low-fat recipes, and cuisine preferences.

Functional Requirements:

Users should be able to filter recipes by dietary restrictions such as vegetarian, vegan, gluten-free, etc.
Users should be able to filter recipes based on dietary needs related to medical conditions such as allergies to a certain type of food or other diagnosis.
Users should be able to filter recipes based on cuisine types; for example: Mediterranean, Asian, American, etc.
The system must update the recipe suggestions in real-time as filters are applied.

3.2.4. SF 4: COMMUNITY ENGAGEMENT PLATFORM

The Community Engagement Platform encourages interaction among users through forums, recipe sharing, and social features.

Functional Requirements:

Users must be able to post and share their own recipes with the community.
The platform should provide a forum where users can discuss cooking techniques and ingredients.
Integration with social media should be facilitated for sharing content outside the app, or sharing content from other media into the forum.

3.2.5. SF 5: REWARDS AND INCENTIVES

The Rewards and Incentives feature aims to enhance user engagement by offering rewards for various in-app activities.

Functional Requirements:

The system must track user activity points earned by sharing recipes or participating in community discussions.
Users should be able to redeem points for in-app features or content.
The system should display the user’s rewards status and history.

3.2.6. SF 6: REAL-TIME NOTIFICATIONS

Real-Time Notifications keep users informed about updates, new recipes, and community activity.

Functional Requirements:

Users must receive notifications for new recipes based on their preferences.
The system must alert users about updates to the app or their community interactions.
Users should have control over the types of notifications they receive.

3.3. OPERATING ENVIRONMENT

"Recipe Roulette" will be developed for iOS and Android platforms, as well as web browsers, ensuring wide accessibility.

3.4. DESIGN AND IMPLEMENTATION CONSTRAINTS

Device Compatibility: The app must be optimized for a wide range of device sizes and capabilities.
Internet Dependency: Real-time features and recipe updates require a stable internet connection.
Content Moderation: A system for moderating user-submitted recipes to maintain quality.
Scalability: The cloud infrastructure must handle an increasing number of users and data volume.
Regulatory Compliance: The app must comply with data protection and privacy laws.

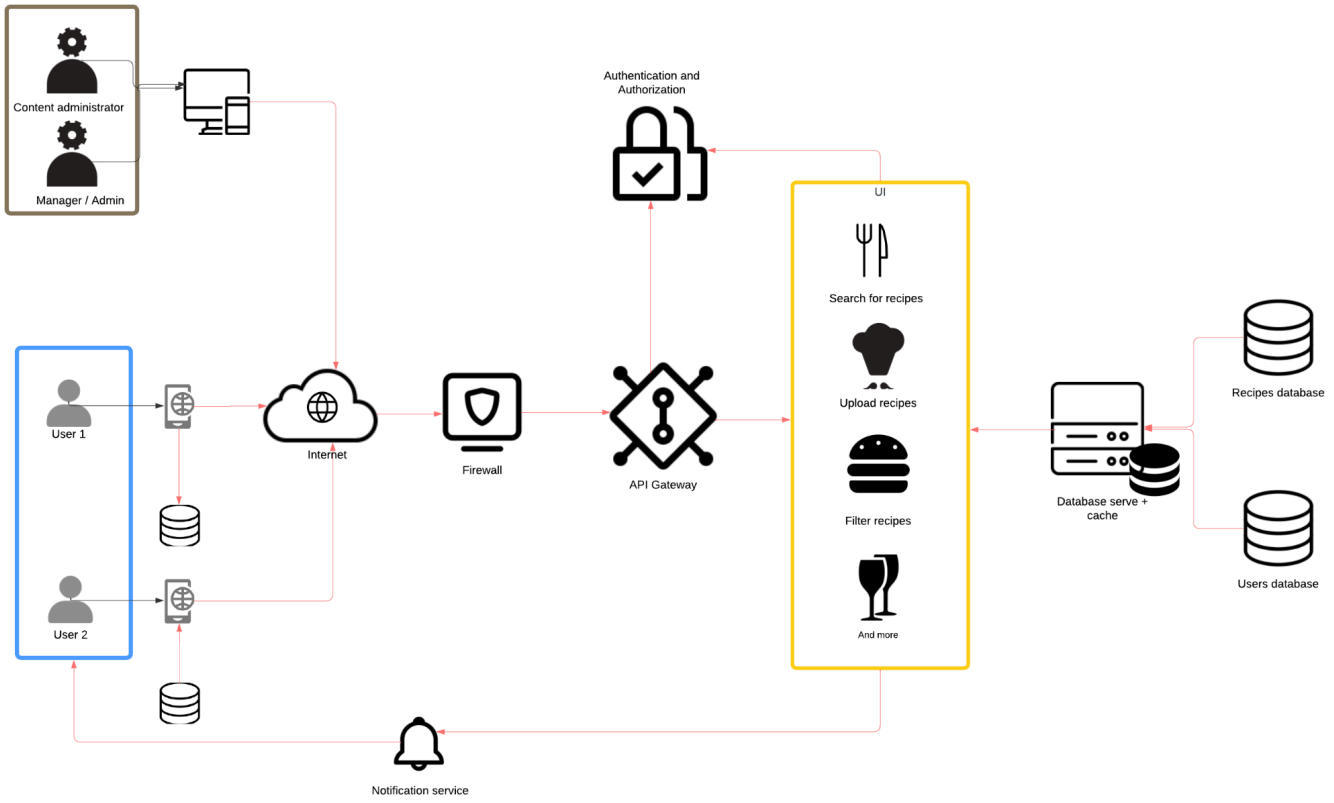
4. DESIGN

4.1. ARCHITECTURAL OVERVIEW

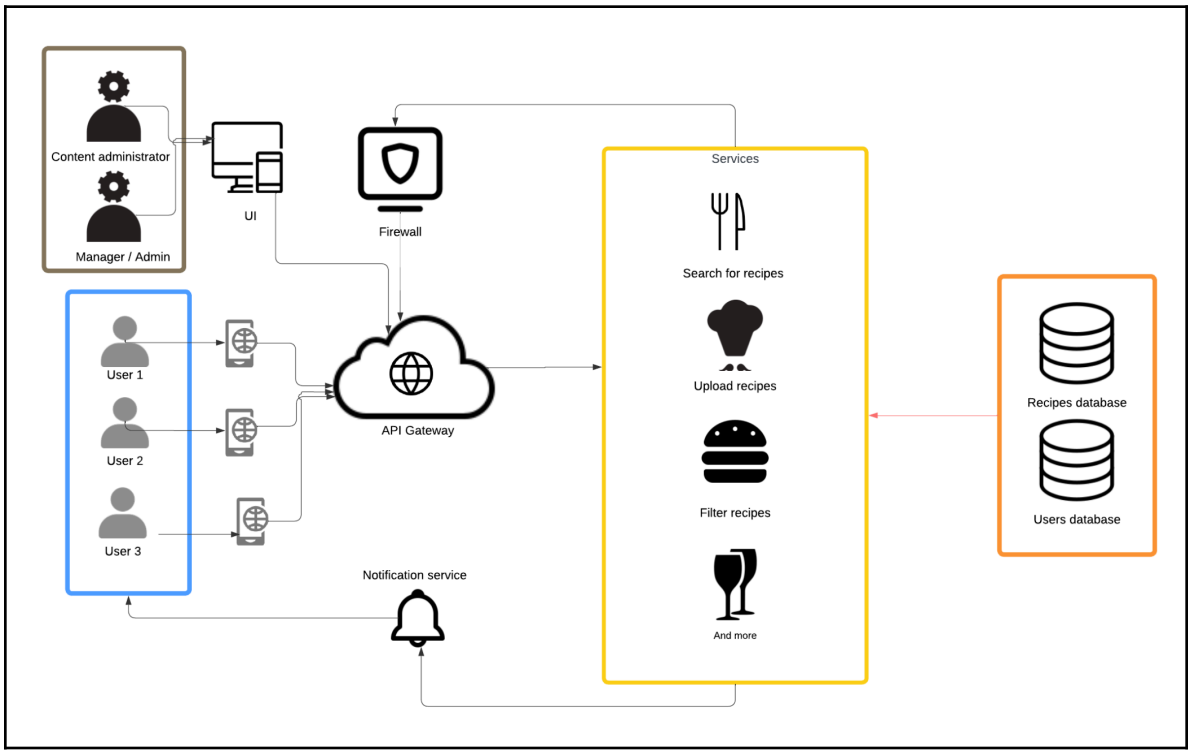
The architectural design of the "Recipe Roulette" application follows a client-server model with a three-tier architecture comprising presentation, application, and data layers.

1. **Presentation Layer:** This layer consists of the user interface components of the application, including the mobile app for iOS and Android platforms, as well as the web interface accessible via major web browsers. The presentation layer is responsible for rendering the user interface elements, handling user interactions, and displaying recipe information.
2. **Application Layer:** The application layer serves as the business logic and processing engine of the system. It includes the recipe discovery engine, user account management, dietary and cuisine filtering, community engagement platform, rewards and incentives system, and real-time notifications. This layer orchestrates the flow of data and logic between the presentation layer and the data layer, ensuring seamless interaction and functionality.
3. **Data Layer:** The data layer comprises the database and data storage components of the application. It stores user account information, recipe data, community forum posts, reward points, and other relevant data. The data layer is responsible for data retrieval, storage, and manipulation, providing the necessary data access interfaces for the application layer.

4.2. INTERFACE DESIGN



Architecture Diagram



Detailed designs for user interfaces, including mockups and navigation flows.

4.3. DATABASE AND DATA FLOW DESIGN

Description of database schemas and data flow diagrams.

4.4. SECURITY PROTOCOLS

Security measures and protocols to protect data and ensure privacy.