SOFTWARE REQUIREMENTS SPECIFICATION
RECIPE ROULETTE

Software Engineering

Bachelor's Degree in Bioinformatics

Course 2023-24

(Version 1.0)
(9-05-24)

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	Reason For Changes	Version
María/Alessandra/Carmen	16/04/2024	Creation of document	0.1
María/Alessandra/Carmen	26/04/2024	Implementation of revision	0.2
María/Alessandra/Carmen	30/04/2024	Improvement of structure and workflow	0.3
María/Alessandra/Carmen	9/05/2024	Changes in requirements section	1.0

II. TABLE OF CONTENTS

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

Functional Requirements	**Non-functional requirements**
UFR-1: User registration	NFR-001-PE-: Performance
UFR-2: User Login/Authentication	NFR-002-US: Usability
UFR-3: Delete Profile	NFR-003-REL: Reliability
UFR-4: Search Recipes by Ingredients	NFR-004-MT: Maintainability
UFR-5: Search Recipes by Dish Name	NFR-005-SE: Security
UFR-6: Filter Recipes	NFR 006-SC: Scalability
UFR-7: Upload, Modify, Delete Recipes	NFR-007-SU: Software Quality Attributes
UFR-8: Gain Points	
UFR-9: Check Reward Status	
MFR-10: Management of Software	
CFR-11: Approve recipe	
CFR-12: Send Notification	
CFR-13: Delete recipe	
CFR-14: Revise Content	

2.1. FUNCTIONAL REQUIREMENTS

UFR-1	**User registration**	**Version: v1.0**
Description: The user access to a registration page and enters their personal information and contact details such as: - Name - e-mail - Phone number - Address - Gender - Age Once the user has to select their role: Beginner, Intermediate and chef. Depending on the selected role the user will have to give the given Role-Specific Information. The user will have to create and confirm their password to complete the registration. After the registration they have to do an account verification by login for the first time to validate the information. If everything is valid and correct the account will be created.		

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:
- The user provides additional details about their current objective and dietary restrictions if any

Intermediate:
- The user should provide the difficulty they can work with and the dietary restrictions as well as the skills they want to work on.

Chef:
- The user provides details about their experience, qualifications, and availability.

Relationships: UFR-2, UFR-3, MFR-10, CFR-12, CFR-11, NFR-001-PE, NFR-005-SE, NFR-007-SU.

UFR-2	**User Login/Authentication**	**Version: v1.0**

Description:
The user should be able to log in to the system using their credentials to get access to their account

Input:
- Username or Email
- 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: v1.0**

Description:
Allow users to delete their profile and all the data associated with it from the system.

How-to:
- Go into the settings
- Select delete profile
- Confirm decision twice to avoid mistakes
- System will delete it all after confirmation and all data will be removed

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: Allows users to search for recipes based on the ingredients they have available. Input: - In the search bar, the user would be able to write the name of ingredients. Then, different matches will appear and when the user selects them, it will be added as a label. - Press 'Search' - User then navigates through results that fulfill the different requirements of the user		
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: Allows users to search for recipes by the name of the dish they want to prepare. Input: - Dish name entered by the user - User then navigates through the results according to his requirements		
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: Allows users to filter recipes based on various criteria to refine search results. Input: - Time required for the recipe - Type of food (Indian, Mediterranean…) - Dietary restriction (vegetarian, vegan…) - DIfficulty level - Allergies (nuts, gluten…)		
Comments: The user should be able to select from various different options.		
Relationships: URF-1 parent, UFR-6		

UFR-7	**Upload, Modify, Delete Recipes**	**Version: v1.0**
Description: Allows users to upload new recipes to the platform, modify existing recipes, and delete recipes they have previously uploaded. Input: - User uploads a recipes and key ingredients for it to be found - Users should also upload the dietary restrictions it can or not 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-11, CFR-12		

UFR-8	**Gain Points**	**Version: v1.0**
Description: Users can earn points by engaging in specific activities within the app. These points can be accumulated and later redeemed for rewards or discounts. The point system encourages user interaction and activity. - **Uploading recipes:** Users gain points for every recipe they upload and share with the community. - **Participation in the community**: Points are awarded for active participation, such as commenting or liking. . - **Referrals:** Users can 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-10

UFR-9	**Check Reward Status**	**Version: v0.1**

Description:
Users can check their current point balance and view available rewards. This feature allows users to track their progress toward redeeming points for rewards, such as discounts, special content, or other incentives. The system should provide a clear interface showing:

- **Point balance:** The current number of points the user has accumulated.
- **Rewards catalog:** A list of available rewards and the point cost for each.
- **Redeem points:** A method to redeem points for a chosen reward, with clear instructions on how to complete the process.
- **Point history:** A detailed history of point transactions, including points earned, redeemed, or expired.

Comments:
- The user interface for checking reward status should be intuitive and easy to navigate.
- The system should have mechanisms to prevent errors or misuse, such as double redemption or incorrect point calculations.
- Point balances should be updated in real-time to ensure accuracy.
- Notifications may be used to alert users about new rewards or points about to expire.
- The rewards system should be scalable to accommodate a growing user base and new reward types.

Relationships: UFR-8, UFR-7, MFR-10 , CFR-12

MFR-10	**Management of Software**	**Version: v1.0**

Description:
This requirement encompasses the administration and maintenance of the cooking app software, ensuring its smooth operation, stability, and continuous improvement. The management of software include

- **Software maintenance:** Regular updates to keep the software secure, up-to-date, and compliant with relevant regulations.
- **Performance monitoring:** Continuous monitoring of the app's performance to identify and resolve bottlenecks or degradation.
- **Error handling and bug fixing:** Rapid response to errors and bugs, with clear protocols for addressing issues.
- **Scalability and resource management:** Ensuring the software can scale to handle increasing user loads and efficiently use resources.
- **Data backups:** Regular backups to ensure data integrity and recovery in case of system failures or data loss.

- **Software security:** Implementation of robust security measures to protect the app and user data from unauthorized access or breaches

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-11	**Approve recipe**	**Version: v1.0**

Description:
The Content Administrator reviews submitted recipes to ensure they meet the application's quality standards, guidelines, and policies. The approval process includes:

- **Recipe content check:** Verify that the recipe contains complete and accurate information, including ingredients, preparation steps, difficulty and cooking time.
- **Compliance with community guidelines:** Ensure the recipe does not contain inappropriate content, such as offensive language or topics unrelated with the app objectives.
- **Quality assurance**: Check that the recipe is unique and does not duplicate existing content. Ensure it includes clear instructions and an appropriate level of detail for users to follow.
- **Approval process:** Once the Content Administrator confirms the recipe meets all requirements, they mark it as approved, allowing it to be published on the platform.
- **Rejection and feedback**: If the recipe does not meet the guidelines, the Content Administrator provides feedback to the user, indicating the reasons for rejection and suggesting improvements.

Possibles error that can happens
- **Incomplete information:** The recipe may lack critical details, such as missing ingredients or unclear preparation steps.
- **Violation of guidelines:** The recipe may contain content that violates community standards, leading to rejection.
- **Duplicate recipe:** If the submitted recipe is too similar to an existing one, it may be rejected to avoid redundancy.
- **Technical issues:** Problems with the content management system could prevent the approval process from completing, requiring troubleshooting or system maintenance

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-12, UFR-11, UFR-7

CFR-12	**Send Notification**	**Version: v1.0**
Description: The Content Administrator sends notifications to users for various purposes, including informing them about uploaded recipes that do not meet the app's guidelines, requesting modifications or suggesting corrections, as well as notifying users about gained rewards, points expiration and other relevant app updates. This process involves: - **Generate notification:** Creating notifications with specific feedback for rejected recipes, reward achievements, points expiration, engagements with the user, possible news and updates, etc. - **Send notification:** Sending notifications to users via the app's messaging system		
Comments: - The notification should be clear and constructive, providing specific reasons for rejection and suggesting how to correct the issues. - Notifications should be clear, constructive, and personalized to the recipient's activity or status within the app. - A reasonable time frame should be provided for users to take action based on the notification's content. - The Content Administrator should have tools to track notification statuses and ensure follow-up actions are taken as needed.		
Relationships: CFR-11, UFR-1, MFR-10, CFR-13, UFR-9 , UFR-8		

CFR-13	**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-11, CFR-12, MFR-10, UFR-7		

CFR-14	**Revise Content**	**Version: v1.0**
Description: The Content Administrator can revise or edit existing content in the app, typically to correct errors, update information, or align with community guidelines. This function helps maintain the accuracy and quality of content. - **Identify content to revise:** The Content Administrator identifies content that needs revision, either from user feedback, regular audits, or guideline changes. - **Revise and update content:** The Content Administrator makes the necessary changes to correct errors, update information, or improve clarity. - **Record changes:** A record of revisions should be maintained for auditing and transparency purposes, allowing for tracking of changes over time. - **Communicate changes:** If the revision affects users or has significant impact, the Content Administrator should communicate the changes to users, explaining the reasons and providing additional information.		
Comments: Revisions should be handled carefully to ensure they do not negatively impact user experience or introduce new errors.		
Relationships: CFR-12, CFR-13 , MFR-10, UFR-7		

2.2. NON-FUNCTIONAL REQUIREMENTS

NFR-001-PE	**Performance**	**Version: v1.0**
Description: - **Response time:** The app should respond to user requests (e.g., searching for recipes, uploading recipes) within 2 seconds. - **Load times:** The app's main screens (e.g., recipe search, user profile) should load in under 5 seconds. - **Concurrent users:** The app should support a large number of users concurrently without significant performance degradation. Aim for at least 5,000 to 10,000 concurrent users. - **Data processing:** The app should handle large datasets (e.g., thousands of recipes) efficiently and without performance issues		
Comments: Load testing should be conducted to ensure compliance with these performance metrics.		
Relationships: NFR-006-SC, NFR-003-REL		

NFR-002-US	**Usability**	**Version: v1.0**
Description: The system should be user-friendly and intuitive for all user classes. - **User interface**: Must adhere to established usability principles and guidelines for accessibility. - **Navigation:** Common tasks should be achievable in three taps to the screen or less.		

Comments: Usability testing should include diverse user groups to ensure the interface is intuitive and accessible to all users.
Relationships: NFR-007-SU

NFR-003-REL	Reliability	Version: v1.0
Description: The system must be reliable, with minimal downtime and errors. - **Uptime:** The system should have an uptime of at least 99.95%. - **Error rate:** System errors should not exceed 0.1% of all transactions		
Comments: Reliability testing should be performed periodically and after each major update.		
Relationships: NFR-004-MT		

NFR-004-MT	Maintainability	Version: v1.0
Description: The system should be easy to maintain, update, and troubleshoot. - **Modularity**: The system should be modular to simplify updates and maintenance. - **Documentation:** Provide comprehensive documentation for developers, including architecture diagrams, API documentation, and code comments. - **Code quality:** Follow coding standards and best practices to ensure maintainability. - **Automated testing:** Implement automated tests to ensure the system is testable and maintainable over time.		
Comments: Regular code reviews and refactoring sessions should be scheduled to ensure maintainability standards are met.		
Relationships: NFR-003-REL		

NFR-005-SE	Security	Version: v1.0
Description: The system must ensure the security and confidentiality of user data. - **Encryption**: All sensitive data (such as user information) should be encrypted during transmission and storage. Ensure user data (personal information) is protected against loss, corruption, or unauthorized access.. - **Compliance:** Must comply with GDPR, HIPAA, and other relevant data protection regulations.		

Comments: Security audits and penetration testing should be conducted annually or after significant changes
Relationships: NFR-001-PE, UFR-1

NFR 006-SC	**Scalability**	**Version: v1.0**
Description: The system must be able to scale seamlessly to meet growing user demand. - **Horizontal scaling:** The architecture should support horizontal scaling without downtime in order to accommodate increased traffic and usage. - **Vertical scaling:** The system should support vertical scaling to increase server capacity as needed. - **Load balancing:** Implement load balancers to distribute user load effectively as the user base grows.		
Comments: Scalability tests should be conducted to verify that the system can handle increased loads as expected.		
Relationships: NFR-001-PE		

NFR-007-SU	**Software Quality Attributes**	**Version: v1.0**
Description: The system should adhere to high software quality standards across various attributes. - **Attributes**: Focus on maintainability, testability, and usability. - **Standards**: Follow best practices and standards such as ISO/IEC 25010.		
Comments: Continuous integration and continuous deployment (CI/CD) practices should be used to maintain high quality in software updates.		
Relationships: NFR-002-US, NFR-004-MT, NFR-003-REL		

2.3. EXTERNAL INTERFACE REQUIREMENTS

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

2.3.1. 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)

2.3.2. 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

2.3.3. 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.

2.4. 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.

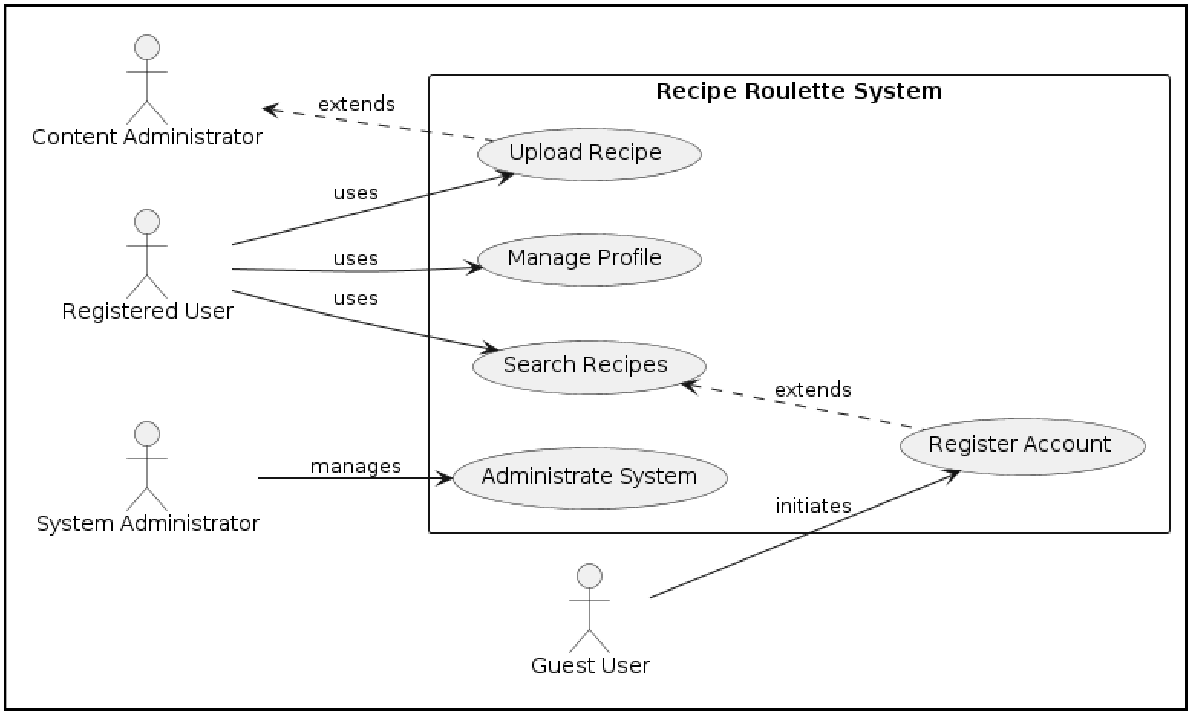
3. USE CASE DIAGRAM

3.1. OVERVIEW OF USE CASES

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.

3.2. 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 (e.g., Registered User, Guest User, System Administrator) and use cases (e.g., Register Account, Search Recipes, Upload Recipe, Manage Profile).

3.3. 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 or Guest)
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 accesses the search feature.
2. The user inputs search criteria and submits the search.
3. The system retrieves and displays recipes matching the criteria.

Alternative Flows:

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

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.

4. SYSTEM ANALYSIS

4.1. 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.

4.2. SYSTEM FEATURES

4.2.1. 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.

4.2.2. 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.

4.2.3. 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.

4.2.4. 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.

4.2.5. 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.

4.2.6. 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.

4.3. OPERATING ENVIRONMENT

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

4.4. 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.

5. DESIGN

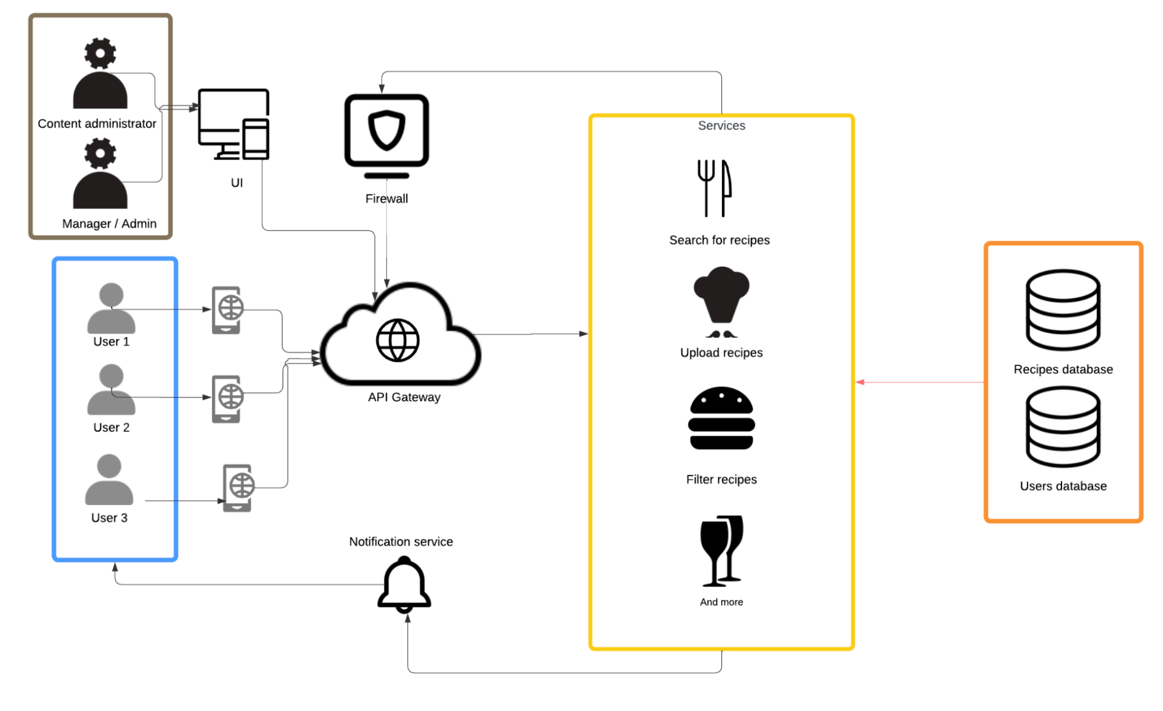
5.1. ARCHITECTURAL OVERVIEW

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.

5.2. INTERFACE DESIGN

Architecture Diagram



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

5.3. DATABASE AND DATA FLOW DESIGN

Description of database schemas and data flow diagrams.

5.4. SECURITY PROTOCOLS

Security measures and protocols to protect data and ensure privacy.