**Flavornaut**

**Recipe Sharing Platform**

*Main Project Report*

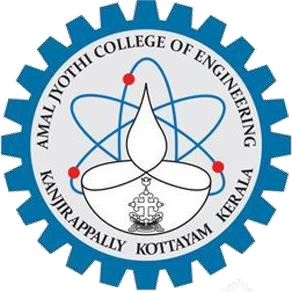
*Submitted by*

**SANDRA MOHAN**

**Reg. No.: AJC23MCA-2056**

*In Partial fulfillment for the Award of the Degree of*

**MASTER OF COMPUTER APPLICATIONS (MCA)**



**AMAL JYOTHI COLLEGE OF ENGINEERING AUTONOMOUS**

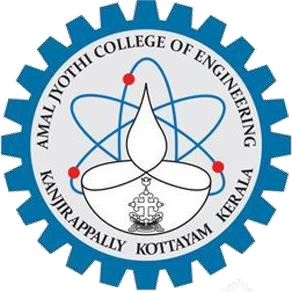
**KANJIRAPPALLY**

[Approved by AICTE, Accredited by NAAC, Accredited by NBA. Koovappally, Kanjirappally, Kottayam, Kerala – 686518]

**2024-2025**

**DEPARTMENT OF COMPUTER APPLICATIONS**

# AMAL JYOTHI COLLEGE OF ENGINEERING AUTONOMOUS KANJIRAPPALLY



**CERTIFICATE**

This is to certify that the Project report, “**FLAVORNAUT -RECIPE SHARING PLATFORM”** is the bonafide work of **SANDRA MOHAN (Regno:**

**AJC23MCA-2056)** in partial fulfillment of the requirements for the award of the Degree of Master of Computer Applications under Amal Jyothi College of Engineering Autonomous, Kanjirappally during the year 2024-25.

**Ms. Gloriya Mathew Ms. Meera Rose Mathew**

**Internal Guide Coordinator**

**Rev. Fr. Dr. Rubin Thottupurathu Jose**  **Head of the Department**

# DECLARATION

I hereby declare that the project report **“FLAVORNAUT-RECIPE SHARING PLATFORM”** is a bonafide work done at Amal Jyothi College of Engineering, towards the partial fulfilment of the requirements for the award of the Master of Computer Applications (MCA) from Amal Jyothi College of Engineering Autonomous during the academic year 2024-2025.

**Date: SANDRA MOHAN**

**KANJIRAPPALLY Reg: AJC23MCA-2056**

# ACKNOWLEDGEMENT

First and foremost, I thank God almighty for his eternal love and protection throughout the project. I take this opportunity to express my gratitude to all who helped me in completing this project successfully. It has been said that gratitude is the memory of the heart. I wish to express my sincere gratitude to our Director (Administration) **Rev. Fr. Dr. Roy Abraham Pazhayaparampil** and Principal **Dr. Lillykutty Jacob** for providing good faculty for guidance.

I owe a great depth of gratitude towards our Head of the Department **Rev. Fr. Dr. Rubin Thottupurathu Jose** for helping us. I extend my whole hearted thanks to the project coordinator **Ms. Meera Rose Mathew** for her valuable suggestions and for overwhelming concern and guidance from the beginning to the end of the project. I would also express sincere gratitude to my guide **Ms. Gloriya Mathew** for her inspiration and helping hand.

I thank our beloved teachers for their cooperation and suggestions that helped me throughout the project. I express my thanks to all my friends and classmates for their interest, dedication, and encouragement shown towards the project. I convey my hearty thanks to my family for the moral support, suggestions, and encouragement to make this venture a success.

SANDRA MOHAN

# 

# 

# ABSTRACT

The Recipe Sharing Platform is a powerful digital tool that enhances the way users store, organize, and access recipes, making everyday cooking more streamlined and enjoyable. With features like recipe input, categorization, ingredient scaling, nutritional analysis, and meal planning, the platform allows users to make informed choices in the kitchen. Recipes can be effortlessly organized by type—such as appetizers, snacks, and desserts—making it easy to access and categorize them for any occasion. Ingredient scaling automatically adjusts quantities based on serving needs, eliminating the hassle of manual calculations. Each recipe includes detailed nutritional information, allowing users to track calories, macronutrients, and more for healthier choices. With intuitive search and filter tools, users can quickly find recipes by keywords, ingredients, or dietary preferences, helping them save time and focus on what they love. Multimedia support brings recipes to life with images, making it easy to visualize each step or the final dish before cooking. This platform transforms meal preparation into a wellorganized and creative process, empowering users to save time, explore new recipes, and unlock their culinary potential.

# CONTENT

|  |  |  |
| --- | --- | --- |
| **SL. NO** | **TOPIC** | **PAGE NO** |
| **1** | **INTRODUCTION** | **1** |
| **1.1** | **PROJECT OVERVIEW** | **2** |
| **1.2** | **PROJECT SPECIFICATION** | **2** |
| **2** | **SYSTEM STUDY** | **4** |
| **2.1** | **INTRODUCTION** | **5** |
| **2.2** | **EXISTING SYSTEM** | **5** |
| **2.3** | **DRAWBACKS OF EXISTING SYSTEM** | **5** |
| **2.4** | **PROPOSED SYSTEM** | **6** |
| **2.5** | **ADVANTAGES OF PROPOSED SYSTEM** |  |
| **3** | **REQUIREMENT ANALYSIS** |  |
| **3.1** | **FEASIBILITY STUDY** |  |
| **3.1.1** | **ECONOMICAL FEASIBILITY** |  |
| **3.1.2** | **TECHNICAL FEASIBILITY** |  |
| **3.1.3** | **BEHAVIORAL FEASIBILITY** |  |
| **3.1.4** | **FEASIBILITY STUDY QUESTIONNAIRE** |  |
| **3.2** | **SYSTEM SPECIFICATION** |  |
| **3.2.1** | **HARDWARE SPECIFICATION** |  |
| **3.2.2** | **SOFTWARE SPECIFICATION** |  |
| **3.3** | **SOFTWARE DESCRIPTION** |  |
| **3.3.1** | **DJANGO** |  |
| **3.3.2** | **MYSQL** |  |
| **4** | **SYSTEM DESIGN** |  |
| **4.1** | **INTRODUCTION** |  |
| **4.2** | **UML DIAGRAM** |  |
| **4.2.1** | **USE CASE DIAGRAM** |  |
| **4.2.2** | **SEQUENCE DIAGRAM** |  |
| **4.2.3** | **STATE CHART DIAGRAM** |  |
| **4.2.4** | **ACTIVITY DIAGRAM** |  |
| **4.2.5** | **CLASS DIAGRAM** |  |
| **4.2.6** | **OBJECT DIAGRAM** |  |
| **4.2.7** | **COMPONENT DIAGRAM** |  |

|  |  |  |
| --- | --- | --- |
| **4.2.8** | **DEPLOYMENT DIAGRAM** |  |
| **4.3** | **USER INTERFACE DESIGN USING FIGMA** |  |
| **4.4** | **DATABASE DESIGN** |  |
| **5** | **SYSTEM TESTING** |  |
| **5.1** | **INTRODUCTION** |  |
| **5.2** | **TEST PLAN** |  |
| **5.2.1** | **UNIT TESTING** |  |
| **5.2.2** | **INTEGRATION TESTING** |  |
| **5.2.3** | **VALIDATION TESTING** |  |
| **5.2.4** | **USER ACCEPTANCE TESTING** |  |
| **5.2.5** | **AUTOMATION TESTING** |  |
| **5.2.6** | **SELENIUM TESTING** |  |
| **6** | **IMPLEMENTATION** |  |
| **6.1** | **INTRODUCTION** |  |
| **6.2** | **IMPLEMENTATION PROCEDURE** |  |
| **6.2.1** | **USER TRAINING** |  |
| **6.2.2** | **TRAINING ON APPLICATION SOFTWARE** |  |
| **6.2.3** | **SYSTEM MAINTENANCE** |  |
| **7** | **CONCLUSION & FUTURE SCOPE** |  |
| **7.1** | **CONCLUSION** |  |
| **7.2** | **FUTURE SCOPE** |  |
| **8** | **BIBLIOGRAPHY** |  |
| **9** | **APPENDIX** |  |
| **9.1** | **SAMPLE CODE** |  |
| **9.2** | **SCREEN SHOTS** |  |

## List of Abbreviations

* **UML -** Unified Modelling Language
* **ORM -** Object-Relational Mapping
* **MVT -** Model-View-Template
* **MVC -** Model-View-Controller
* **RDBMS -** Relational Database Management System
* **1NF -** First Normal Form
* **2NF** - Second Normal Form
* **3NF -** Third Normal Form
* **IDE -** Integrated Development Environment
* **HTML -** HyperText Markup Language
* **JS -** JavaScript
* **CSS -** Cascading Style Sheets
* **AJAX -** Asynchronous JavaScript and XML
* **JSON -** JavaScript Object Notation
* **API -** Application Programming Interface
* **UI -** User Interface
* **PK -** Primary Key
* **FK -** Foreign Key
* **SQL -** Structured Query Language
* **CRUD -** Create, Read, Update, Delete

**CHAPTER 1**

**INTRODUCTION**

* 1. **PROJECT OVERVIEW**

The “Recipe Sharing Platform” is an innovative online community where users can discover, share, and save recipes, catering to both casual cooks and food enthusiasts. Designed to enhance user interaction, recipe management, and community engagement, the platform offers a personalized and immersive culinary experience. Users can create and manage profiles, allowing for a tailored experience as they navigate the platform. The core feature is recipe management, enabling users to create, edit, and delete recipes, complete with ingredients, instructions, and images. An extensive library of recipes is available for browsing, with tools to categorize and tag recipes, making it easy to find and organize dishes. Community engagement is fostered through a rating and review system, encouraging users to share feedback and interact with others. The platform’s powerful search and discovery features allow users to find recipes by name, ingredient, category, tags, and ratings, ensuring quick access to preferred dishes. Advanced functionalities include ingredient substitution suggestions, meal planning tools, and the ability to share recipes via social media or email. Nutritional information is provided for each recipe, and users can save favourites and contribute cooking tips, enriching the collaborative environment. Unique features like storytelling allow users to share the personal stories behind their recipes, while pairing suggestions help create well-rounded meals. The platform also offers an ingredient knowledge base and advanced dietary filtering. To add fun and engagement, the platform hosts recipe competitions with clear guidelines, showcasing winners on the homepage and offering high-rated recipes for sale. Overall, the Recipe Sharing Platform is a vibrant community that blends practical features with social interaction, making it an ideal destination for culinary exploration.

* 1. **PROJECT SPECIFICATION**

### Administrators

* **Manage Users:** Manage user accounts, including registration approvals, account suspensions, and handling user roles (e.g., promoting users to Recipe Managers).
* **Manage Nutritional information:** Manages responsible for overseeing the accurate management and display of nutritional information.
* **Manage Events:** Admins set up and manage recipe contests, including creating guidelines, selecting judges, and announcing winners.
* **Manage Recipes:** Admins are responsible for creating, editing, viewing, and deleting recipes.
* **Manage Ingredients:** They provide in-depth ingredient knowledge, ensuring that substitution suggestions are accurate and useful.
* **Manage Categories:** Admins can create new categories like "Vegetarian," "Non-Vegetarian," "Desserts," "Appetizers," etc.
* **Manage Sub-Categories:** Admins can create sub-categories under specific main categories, such as "Pickles" under "Vegetarian" or "Snacks" under "Non-Vegetarian."

### Users

* **Authentication:** Users can sign up, log in, and manage their profiles.
* **Recipe Management:** Users can create, edit, categorize, and delete their own recipes.
* **Search & Discovery:** Users can search for recipes using various filters, including dietary needs and ingredient substitutions.
* **Meal Planning:** Users can plan meals by selecting recipes and generating shopping lists.

**CHAPTER 2**

**SYSTEM STUDY**

* 1. **INTRODUCTION**

The “Recipe Sharing Platform” is an online community where users can discover, share, and save recipes, catering to both casual cooks and food enthusiasts. It offers personalized profiles, allowing users to create, edit, and manage recipes complete with ingredients, instructions, and images. An extensive recipe library enables easy search and organization by category, tags, or ingredients.

Users can rate and review recipes, share tips, and explore dishes using dietary filters and nutritional information. Unique features include ingredient substitution suggestions, meal planning tools, and recipe storytelling. The platform also hosts recipe competitions, showcasing winners and highrated recipes for sale, creating a vibrant and interactive culinary community.

* 1. **EXISTING SYSTEM**

The existing system of recipe-sharing-platforms generally offers basic features like recipe discovery , user submissions, and simple search functions. However, these platforms often fall short in terms of depth and personalization. Search capabilities are limited to broad categories or keywords, making it difficult for users to find recipes tailored to specific dietary needs or preferences. Personalization is minimal, lacking tailored recommendations based on user behaviour. Social engagement features are also underdeveloped, with few opportunities for users to build connections or participate in community activities. Additionally, comprehensive ingredient knowledge, substitution options, and detailed nutritional information are often missing or inconsistent. Overall, while functional, these platforms do not fully meet users’ expectations for a rich, interactive, and personalized culinary experience.

* 1. **DRAWBACKS OF EXISTING SYSTEM**
* Limited search and discovery: Finding specific recipes or exploring new cuisines can be challenging due to poor search functionality or limited categorization.
* Inconsistent recipe quality: Variations in recipe format, clarity, and accuracy can lead to frustration for users.
* Lack of personalization: Most systems fail to tailor recommendations based on individual preferences, dietary restrictions, or cooking experience.
* Insufficient nutritional information: Many recipes lack detailed nutritional data, making it difficult for users to make informed dietary choices.
* Paid content or advertisements: Some platforms rely on subscription fees or intrusive ads, which can diminish the user experience.

**2.4 PROPOSED SYSTEM**

The proposed recipe-sharing platform goes beyond conventional cooking sites by focusing on a holistic, community-driven culinary experience. Every element has been thoughtfully designed to engage users in ways that blend functionality, inspiration, and social interaction, making it a one-stop destination for food lovers, from home cooks to culinary enthusiasts. This platform empowers users not only to discover and save recipes but also to develop a sense of ownership and pride in their cooking journeys. With features such as user profiles, followers, and recipe collections, users can showcase their culinary accomplishments, build personalized recipe libraries, and follow their favorite creators for ongoing inspiration.

A highlight of the platform is its smart ingredient and dietary management capabilities, catering to modern dietary needs and preferences with unmatched ease. Ingredient substitution suggestions help users adapt recipes to their taste or dietary restrictions, whether by swapping dairy with plant-based alternatives or finding gluten-free ingredient options. With meal planning tools, users can organize their weekly menus, ensuring balanced meals and minimizing food waste. The platform’s detailed nutritional analysis, readily available for each recipe, empowers users to make health-conscious decisions in line with their dietary goals, tracking everything from calories and macronutrients to vitamin intake.

Beyond its core features, the platform encourages community engagement and social sharing in creative and dynamic ways. Through storytelling, users can add personal reflections, family anecdotes, or cultural context to recipes, transforming each dish into a shared experience that extends beyond the kitchen. This feature builds deeper connections, creating a sense of community and encouraging conversations around food traditions, memories, and regional flavors. An advanced rating and review system adds an extra layer of trust, allowing users to share detailed feedback on recipes, helping others discover top-rated dishes or avoid common pitfalls in complex recipes.

Search and discovery tools on the platform are enhanced by a powerful tagging system, which enables users to filter recipes not only by type or cuisine but also by unique attributes like cooking methods, preparation times, or seasonal ingredients. Users can explore everything from quick weeknight dinners to elaborate holiday feasts with ease, guided by smart search filters and visually appealing recipe thumbnails. Multimedia support further enriches the platform, with high-quality images and video tutorials for each recipe step, providing users with a visually engaging cooking guide that boosts confidence and eases the learning curve for challenging dishes.

In fostering a collaborative spirit, the platform regularly hosts recipe contests, seasonal challenges, and themed cooking events, inviting users to showcase their skills and explore new techniques. This gamified approach keeps users motivated, bringing an element of fun and excitement to cooking while also offering the opportunity to earn recognition and rewards. Furthermore, administrators and recipe managers actively curate and promote high-quality content, highlighting exceptional recipes and elevating standout creators, ensuring that users are constantly exposed to fresh ideas and trending flavors.

With its seamless blend of social features, practical tools, and a vibrant community focus, this platform doesn’t just cater to everyday cooking needs—it aims to inspire culinary exploration, encourage knowledge sharing, and celebrate the universal love of food. The platform aspires to become a culinary hub, where users can learn, teach, and connect with others, creating a dynamic ecosystem that transforms cooking from a solitary task into a shared, interactive, and enriching experience.

**2.4 ADVANTAGES OF PROPOSED SYSTEM**

* **Comprehensive Recipe Management:** Users can easily create, manage, and categorize their recipes, providing a well-organized and personalized collection. Advanced search functionalities and filters ensure users can quickly find recipes that meet their specific needs, such as dietary restrictions or ingredient availability.
* **Ingredient Substitution Suggestions:** The platform offers intelligent substitution options, allowing users to adapt recipes based on available ingredients or dietary preferences. This feature adds flexibility and usability to the recipe collection.
* **Meal Planning Tools**: Users can plan their meals efficiently by selecting recipes and generating shopping lists, which simplifies meal preparation and grocery shopping.
* **Detailed Nutritional Information**: Users have access to comprehensive nutritional data for recipes, enabling them to make informed dietary choices and track their nutritional intake more effectively.
* **Recipe Contests**: The introduction of recipe contests with clear guidelines and judging criteria motivates users to showcase their culinary skills and creativity. This feature not only engages users but also adds a fun and competitive element to the platform.

## CHAPTER 3

**REQUIREMENT ANALYSIS**

### 3.1 FEASIBILITY STUDY

A feasibility study is a comprehensive evaluation of a proposed project to determine its practicality and potential for success. This process begins with analysing the technical aspects of the project to ensure that the necessary technology, skills, and infrastructure are available or can be acquired. It involves evaluating whether the proposed system can be integrated with existing technologies and if the team possesses the requisite technical expertise. The study also considers operational aspects such as system usability and integration with existing workflows, as well as market demand and competitive landscape. By identifying potential risks and challenges early, the feasibility study helps stakeholders make informed decisions, optimize resource allocation, and reduce the likelihood of costly errors and project failures. This comprehensive analysis provides a solid foundation for successful project planning and execution.

**3.1.1 Economical Feasibility**

Conducting an economic feasibility analysis is essential for assessing the financial viability and investment requirements of the recipe-sharing platform. This analysis involves a detailed examination of the costs associated with developing the platform, including technology infrastructure, content management, and operational expenses. It also evaluates potential revenue streams, such as subscription fees The platform’s alignment with growing consumer interest in culinary exploration and personalized food experiences further enhances its economic feasibility. Core features like user profiles, recipe management, and community interactions can be built with web development frameworks, databases, and APIs. Advanced functionalities such as ingredient substitution, meal planning, and dietary filtering are achievable with search engines and nutritional databases. Social sharing, security, and scalability can be managed using social media APIs, cloud infrastructure, and secure authentication protocols. Overall, the platform can be effectively built, scaled, and maintained with current technology and best practices.

**3.1.2 Technical Feasibility**

The technical feasibility study for the Recipe Sharing Platform demonstrates a strong potential for successful implementation due to several key factors. The development team possesses the expertise and resources needed to leverage modern web technologies and best practices. Advanced features like robust search functionalities, ingredient substitution suggestions, and personalized recommendations are within reach, thanks to the platform's use of Django and its extensive libraries. Scalability is a critical aspect of the platform’s feasibility. Efficient coding practices and a solid infrastructure ensure that the system can handle increasing user traffic and an expanding recipe database. Technical assessments confirm that the platform adheres to industry standards, which guarantees it will meet performance requirements and remain adaptable as the platform grows. Overall, the study verifies that the project is not only technically viable but also wellpositioned to deliver a comprehensive and user-friendly experience.

**3.1.3 Behavioral Feasibility**

Behavioral feasibility is a crucial aspect of evaluating the recipe-sharing platform, focusing on user and stakeholder acceptance and engagement.

**User Acceptance**: For the platform to succeed, it must enhance the culinary experience by providing valuable features such as meal planning tools, and nutritional information. The success of the project hinges on users' willingness to adopt and actively engage with the platform for discovering, sharing, and managing recipes.

**Recipe Contributor Engagement**: The platform relies on recipe contributors to upload and manage their recipes. Their readiness to embrace the digital environment, share detailed recipes, and interact with the community is essential. Behavioral feasibility assesses their enthusiasm and readiness to engage with the platform’s features.

**Administrative Adaptation**: Administrators are crucial for the effective operation of the platform, handling tasks such as user management, content moderation, and system maintenance. Their acceptance of the platform’s functionalities and their ability to manage these responsibilities efficiently are vital for smooth operation.

**3.1.4 Feasibility Study Questionnaire**

**1.Project Overview ?**

The primary objective of the project is to develop a comprehensive recipe-sharing platform that enables users to create, share, and discover a wide range of culinary delights. The platform aims to enhance user experience through advanced features like ingredient substitution suggestions, meal planning tools, and detailed nutritional information. It seeks to foster a vibrant community by encouraging recipe sharing, storytelling, and user-generated content. By incorporating social interaction features, engaging contests, and a premium content system, the platform aims to build a connected and dynamic online community centered around food, inspiration, and shared culinary experiences.

1. **To What Extent the System Is Proposed For?**

The “Recipe Sharing Platform” is proposed to offer a comprehensive and user-centric recipe sharing experience. It includes core functionalities such as user authentication,recipe management, and advanced search features. The platform will provide additional tools like ingredient substitution suggestions, meal planning, and nutritional information. It aims to foster community engagement through recipe sharing, storytelling, and user-generated content. Social interaction features, recipe contests will further enhance user involvement. The system is designed with a focus on robust security and scalability to support a growing user base and ensure a reliable platform.

1. **Specify the Viewers/Public which is to be involved in the System?**

General Visitors, Regular Users, Recipe Managers

1. **List the Modules included in your System?**

Admin, Guest Users, Recipe Managers

1. **Identify the users in your project?**

Guest Users, Recipe Managers

1. **Who owns the system?**

Administrator

1. **System is related to which firm/industry/organization?**

Food and Culinary

1. **Details of person that you have contacted for data collection?**

Nakul B Narayanan (Kitchen alchemist)

1. **Questionnaire to collect details about the project?**
2. **What are the primary types of recipes featured on your website (e.g., traditional, contemporary, regional)?**

Our website features a wide range of recipes, including traditional, contemporary, and regional dishes. We cater to diverse tastes, offering everything from classic comfort foods to modern, health-conscious meals.

1. **Are there any unique features on your website that set it apart from other recipe platforms?**

One of our standout features is the "Ingredient Swap" tool, which suggests alternatives for ingredients users may not have on hand or want to avoid. We also offer interactive cooking classes and live Q&A sessions with professional chefs.

1. **What tools or features do you provide for users to interact with the content (e.g., commenting, rating, sharing)?**

User can interact with our content by commenting on recipes, rating them, and sharing their own tips or modifications. They can also share recipes with friends and family via email or social media.

1. **Can users create and save personal recipe collections?**

Yes, users can create and save personal recipe collections. They can categorize these collections based on themes like "Holiday Favourite’s," "Quick Weeknight Dinners," or "Vegan Options."

1. **Do you have a feature for users to share recipes on social media?**

We have built-in features that allow users to easily share their favourite recipes on social media platforms like Facebook, Instagram.

1. **How do you handle negative feedback or reviews on recipes?**

We value all feedback, including negative reviews, as it helps us improve our content. Our moderation team reviews negative comments to ensure they are constructive, and we often respond with tips or suggestions for better results.

1. **What’s the most recent feature you’ve added to your platform, and what was the motivation behind it?**

The most recent feature we've added is a "Meal Planner" tool that allows users to plan their meals for the week, complete with a shopping list. This was introduced in response to user feedback requesting more tools for meal preparation and organization**.**

1. **Have you partnered with any influencers or chefs to create exclusive content?**

We have partnered with several renowned chefs and food influencers to create exclusive content. These collaborations include video tutorials, special recipe collections, and live cooking demonstrations**.**

1. **How do you ensure that your website is accessible and easy to use for all users?**

We prioritize accessibility by ensuring our website is navigable with screen readers, providing text alternatives for images, and using a clear, easy-to-read font. We also offer step-by-step guides with video and audio instructions for those who prefer multimedia content.

1. **Do you offer personalized recipe recommendations? If so, how do they work?**

Yes, we offer personalized recipe recommendations based on users' preferences, past activity, and dietary restrictions. The more a user interacts with our platform, the more tailored the recommendations become.

### 3.1 SYSTEM SPECIFICATION

**3.2.1 Hardware Specification**

Processor - Intel core i3

RAM - 8 G B

Hard disk - 2 3 8 G B

**3.2.2 Software Specification**

Front End - HTML5, Bootstrap, CSS

Back End - DJANGO, PYTHON

Database - MYSQL

Client on PC - Windows 7 and above.

Technologies used - DJANGO, HTML5, JS, CSS, Bootstrap

### 3.3 SOFTWARE DESCRIPTION

#### 3.3.1 Django Framework

The Django Framework stands as a popular and robust web framework specifically designed for Python developers. Its reputation is built on its attributes of simplicity, clean code, and rapid development capabilities. Django follows the Model-Template-Views (MTV) architectural pattern, which bears resemblance to the Model-View-Controller (MVC) pattern seen in other frameworks. Notably, it incorporates an Object-Relational Mapping (ORM) system, which simplifies database interactions by representing database tables as Python objects. This abstraction effectively eliminates the need for writing raw SQL queries, significantly simplifying database operations. Django's offerings include a built-in administrative interface that streamlines content management. Its URL routing system empowers developers to establish clean and user-friendly URLs for web applications. Moreover, Django comprehensively supports form handling, data validation, and user authentication, reducing the intricacies of common web development tasks. Emphasizing security, Django comes equipped with built-in defences against common web vulnerabilities such as Cross-Site Scripting (XSS) and Cross-Site Request Forgery (CSRF). The framework's modular architecture encourages extensibility, enabling developers to seamlessly integrate third-party packages or create custom components. Thanks to a thriving community and a wide array of reusable packages, Django proves to be a versatile choice suitable for web development projects of varying sizes and complexities.

**3.3.2 MYSQL:**

MySQL is a widely-used open-source relational database management system (RDBMS) developed by Oracle Corporation. It utilizes Structured Query Language (SQL) for managing and manipulating data, providing a reliable and flexible platform for storing and retrieving information. MySQL organizes data into structured tables that can be linked based on common attributes, ensuring efficient data management and integrity. Known for its performance and scalability,

MySQL can handle both small-scale and large, complex applications with high volumes of data.

It operates across various operating systems, including Windows, Linux, and macOS, and includes robust security features like user authentication and data encryption. As an open-source platform, MySQL benefits from a large community of developers and users who contribute to its ongoing improvement and provide extensive support. Its combination of speed, reliability, and versatility makes MySQL a popular choice for diverse database needs.

## CHAPTER 4 SYSTEM DESIGN

### 4.1 INTRODUCTION

The development of any system or product begins with the design phase is crucial to developing the "Recipe Sharing Platform," ensuring functionality, performance, and user engagement. It focuses on features like personalized profiles, recipe management, advanced search, rating systems, and nutritional information display. Unique elements such as storytelling and recipe competitions were also integrated to enhance the user experience. This design blueprint provides a clear path for implementation, ensuring a smooth and efficient platform development

### 4.2 UML DIAGRAM

A standardized dialect known as Unified Modeling Language (UML) serves as a vital tool for conceptualizing, defining, designing, and describing software systems. The Object Management Group (OMG) was responsible for the development of UML, and the initial UML 1.0 draft was introduced in January 1997. It's important to note that UML is distinct from programming languages like Java, C++, and COBOL. UML is a generic visual modeling language employed for computer program systems and a pictorial language used for program designs. Although UML is widely employed in representing software systems, its utility extends beyond software and encompasses various applications, including manufacturing processes.

UML encompasses a range of diagrams, each tailored to specific purposes:

* Class diagram
* Object diagram
* Use case diagram
* Sequence diagram
* Collaboration diagram
* Activity diagram
* State chart diagram
* Deployment diagram
* Component diagram

#### 4.2.1 USE CASE DIAGRAM

A use case diagram is a visual representation that illustrates how users and other external entities interact with the internal components of a system. Its primary function is to identify, outline, and organize a system's functional requirements from the perspective of its users. Typically constructed using the Unified Modeling Language (UML), which serves as a standard language for modeling physical entities and systems. It is composed of system boundaries, actors, use cases, and their interconnections. The system boundary delineates the system's limits concerning its environment. Actors are characterized based on their roles and represent individuals or systems that interact with the system.

Use case diagrams are visual tools that capture a system's functional requirements. When creating a use case diagram, it is important to adhere to these guidelines for an efficient and effective representation:

* Select descriptive names for use cases that accurately convey their functionalities.
* Assign appropriate names to actors to clarify their roles in the system.
* Ensure that relationships and dependencies are clearly depicted in the diagram.
* Avoid overcomplicating the diagram with unnecessary relationships, focusing on identifying the essential requirements.
* Use notes when necessary to provide additional context or explanations. By following these guidelines, a clear and concise use case diagram can be created, accurately depicting the system's functional requirements.

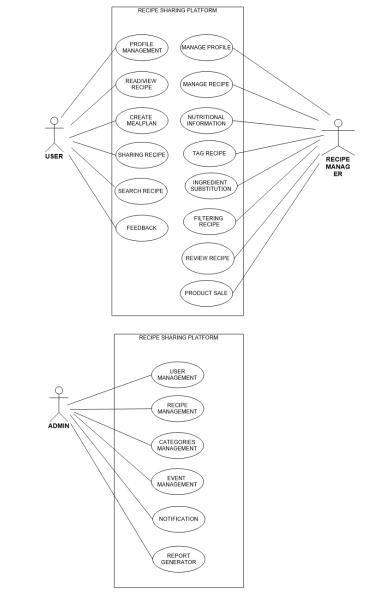


Figure 4.2.1 Use case diagram

#### 4.2.2 SEQUENCE DIAGRAM

A sequence diagram, categorized as an interaction diagram, is a graphical representation of how various system components interact with one another over a series of messages or actions. These diagrams are also referred to as event scenarios or event scenario diagrams. Sequence diagrams are commonly used in software engineering to comprehend the requirements of both new and existing systems, aiding in the visualization of object control relationships and the identification of systemic issues.

Applications of Sequence Diagrams:

* Modeling and visualizing complex functions, operations, or procedures.
* Detailing UML use case diagrams.
* Understanding the intricate functionality of current or future systems.
* Visualizing the flow of messages and tasks between system objects or components.
* Overall, sequence diagrams are valuable for representing the interaction flow between objects in a system, aiding both business stakeholders and software engineers in comprehending and communicating system requirements and behaviors.

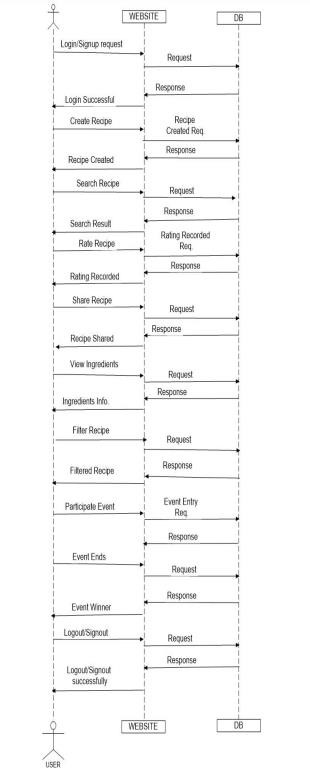


Figure 4.2.2 Sequence Diagram

#### 4.2.3 STATE CHART DIAGRAM

A state diagram, often created using the Unified Modeling Language (UML), is a visual representation that illustrates the different states an object can assume and the transitions between these states. It is alternatively known as a state machine diagram or a state chart diagram. A State Chart Diagram, a type of behavioral diagram in UML, provides insight into the behavior of a system or an object over time. It comprises various key elements:

* Initial State: This state marks the system or object's starting point and is depicted by a solid black circle.
* State: These elements describe the system or object's current condition at a specific moment and are symbolized by rectangles with rounded corners.
* Transition: Represented by arrows, transitions illustrate the movement of the system or object from one state to another.
* Event and Action: An event acts as a trigger that initiates a transition, while an action denotes the behavior or consequence of that transition.
* Signal: Signals, triggered by events, are messages sent to a state, prompting a transition.
* Final State: The State Chart Diagram concludes with a Final State element, recognizable by a solid black circle with a dot inside. It signifies the completion of the system or object's behavior.

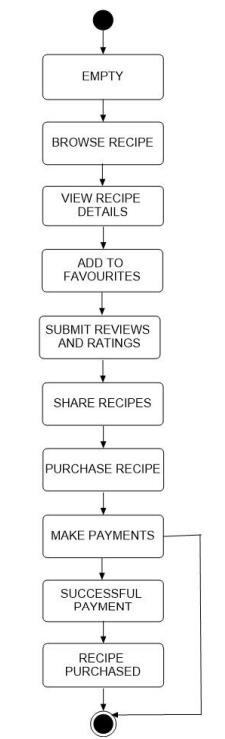


Figure 4.2.3 State chart Diagram

#### 4.2.4 ACTIVITY DIAGRAM

An activity diagram serves as a visual representation of a workflow, illustrating the sequential progression from one activity to another. Each activity, which is a system operation, leads to another in the control flow. These flows can take various forms, such as parallel, concurrent, or branched, and activity diagrams employ functions like branching and joining to manage these different types of flow control. Activity diagrams fall under the category of behavior diagrams and depict a system's behavior. They reveal the control flow from the starting point to the endpoint, highlighting the diverse decision paths encountered during activity execution.

Key elements include:

1. **Action/Activity**: Represents a specific task or step in the process.
2. **Initial Node**: Marks the starting point of the workflow.
3. **Final Node**: Represents the end of the workflow.
4. **Decision Node**: Shows a point where a decision must be made, with branches for different outcomes.
5. **Merge Node**: Combines multiple decision branches back into a single flow.
6. **Fork Node**: Splits the flow into parallel activities.
7. **Join Node**: Merges parallel activities back into a single flow.
8. **Transitions/Arrows**: Indicate the flow of control between actions.
9. **Swimlanes**: Divide the diagram into sections to show different roles or responsibilities.

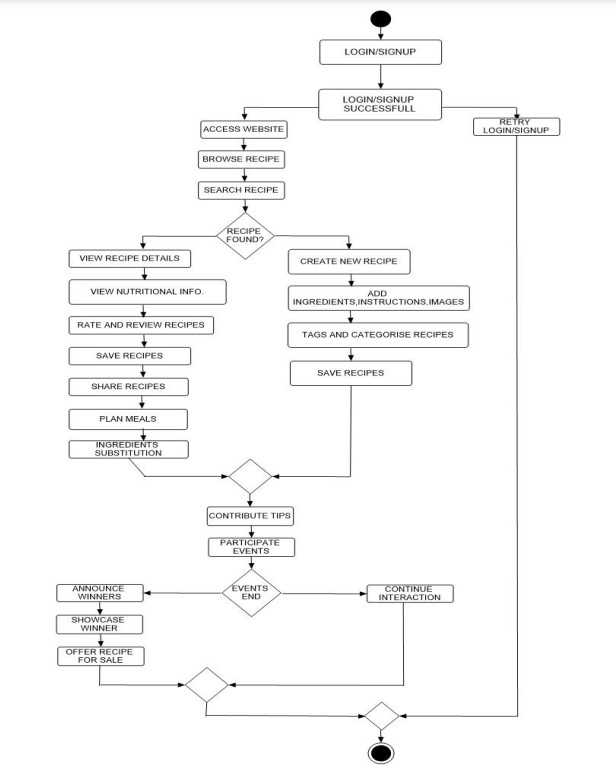


Figure 4.2.4: Activity Diagram

#### 4.2.5 CLASS DIAGRAM

Class diagrams are foundational elements in object-oriented modeling and serve as the primary tool for conceptually visualizing the structure of an application. Additionally, class diagrams are valuable for detailed modeling that can be translated into actual programming code. They also find application in data modeling.

Key aspects include:

1. **Classes**: Represent entities or objects, with the class name at the top.
2. **Attributes**: Define the properties or characteristics of a class.
3. **Methods**: Represent the operations or functions that a class can perform.
4. **Visibility**: Shows the access level of attributes and methods (e.g., + for public, - for private, # for protected).
5. **Relationships**:
   * **Association**: A basic connection between two classes.
   * **Aggregation**: Represents a "whole-part" relationship, where one class is part of another but can exist independently.
   * **Composition**: A stronger form of aggregation, where the part cannot exist independently of the whole.
   * **Inheritance (Generalization)**: Indicates that one class (subclass) inherits from another (superclass).
   * **Dependency**: A weak relationship where one class depends on another.
6. **Multiplicity**: Specifies the number of instances involved in a relationship (e.g., one-toone, one-to-many).
7. **Abstract Classes**: Indicate classes that cannot be instantiated directly, often represented with italics or <abstract>.

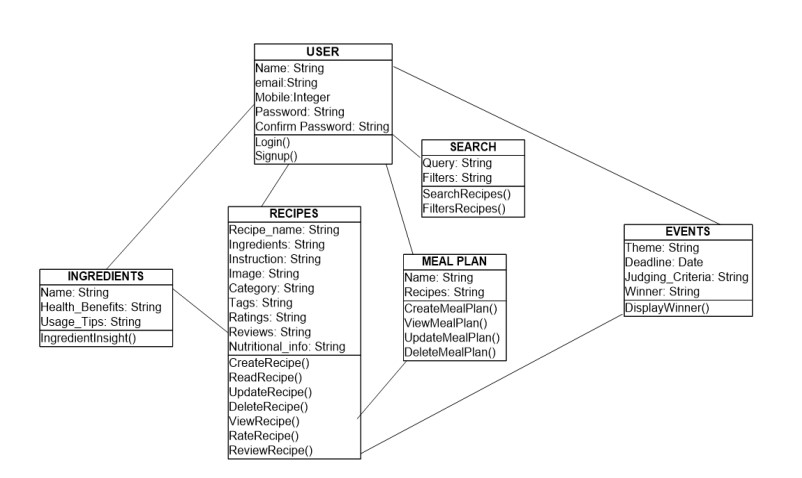


Figure 4.2.5: Class Diagram

#### 4.2.6 OBJECT DIAGRAM

Class diagrams and object diagrams are closely intertwined in the realm of object-oriented modeling. Object diagrams can be considered as real-world snapshots derived from class diagrams, capturing a system's state at a particular moment. Both types of diagrams employ identical concepts and notations to delineate a system's structure. While class diagrams focus on modeling the system's structure, encompassing classes, attributes, and methods, object diagrams illustrate a collection of objects and their interconnections at a precise instant.

Key aspects include:

1. **Objects**: Instances of classes, shown with object names (e.g., objectName : ClassName).
2. **Attributes (Instance Values)**: Specific values assigned to the object's attributes at the time of the snapshot.
3. **Links/Relationships**: Connections between objects, representing associations or dependencies as seen in the class diagram.
4. **Multiplicity**: Shows how many objects participate in a relationship, indicating one-to-one, one-to-many, or many-to-many connections.
5. **State of Objects**: Reflects the specific state of objects and their attributes at the given moment.
6. **Snapshot in Time**: Unlike class diagrams, which show possible structures, object diagrams represent the actual instance relationships and values during runtime.

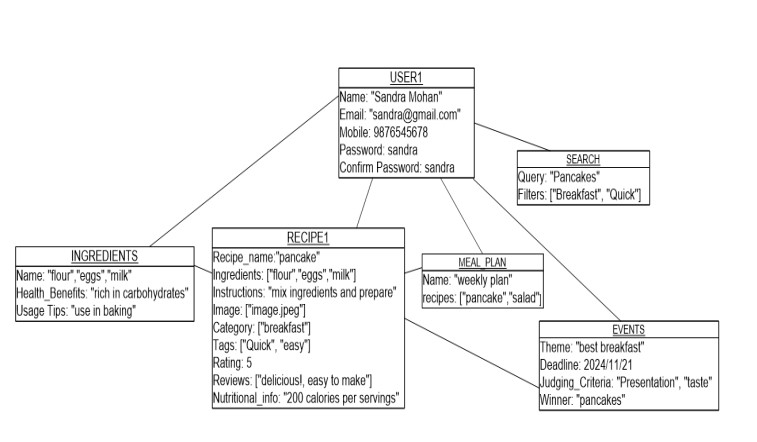


Figure 4.2.6: Object Diagram

### 4.3 USER INTERFACE DESIGN USING FIGMA

**4.3.1**

**Form Name: Registration Page**

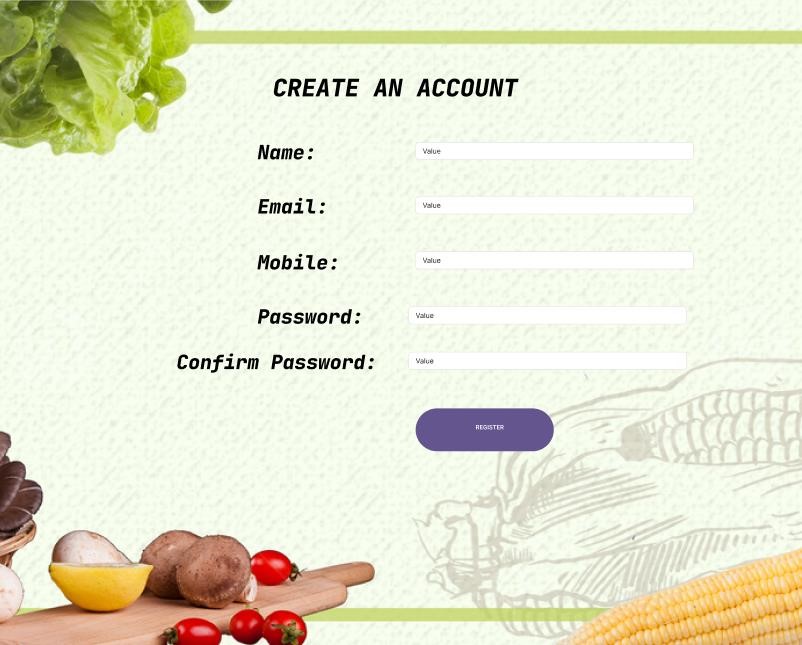


Figure 4.3.1: Registration Form

**4.3.2. Form Name: Login Page**

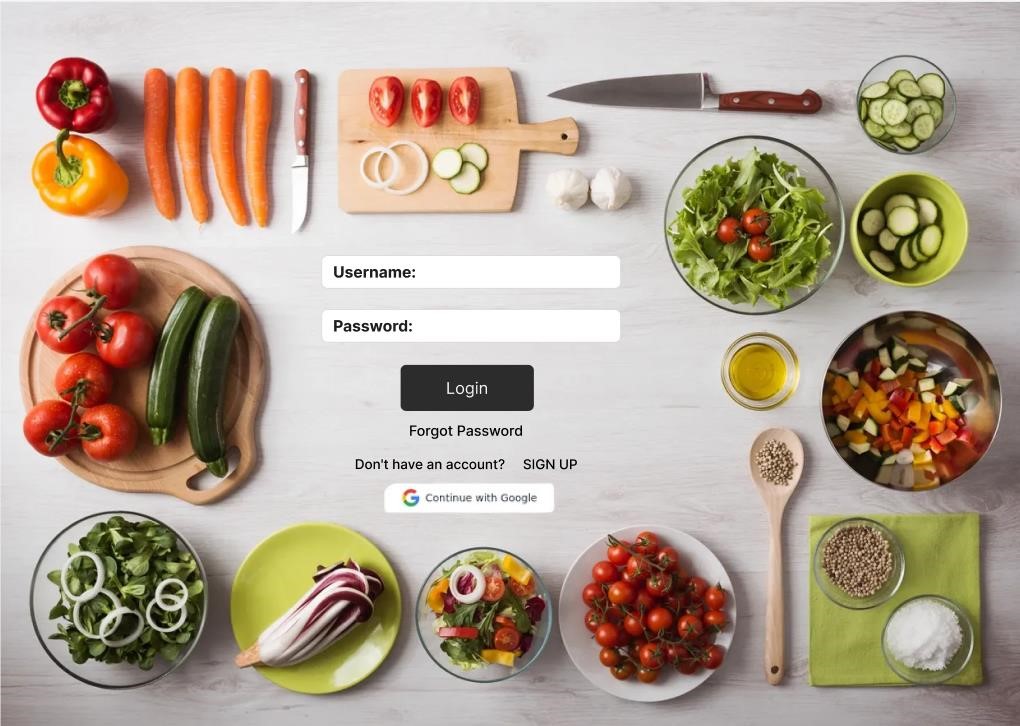
Figure

4.

3.2

:

Login Page



**4.3.**

**3**

**. Form Name: Home Page**

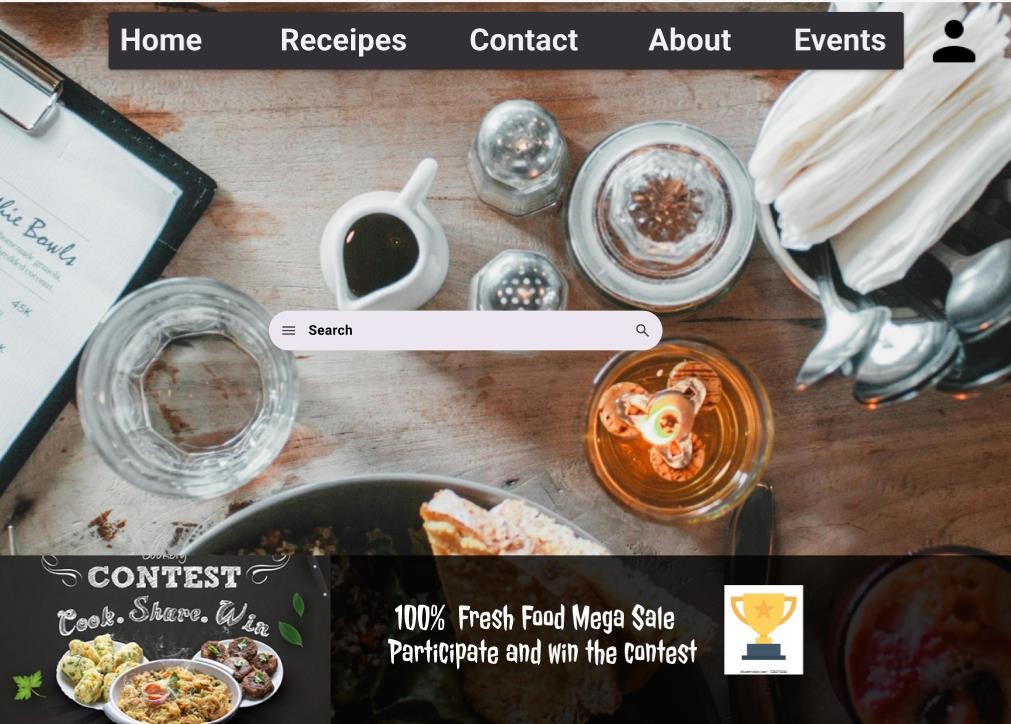


Figure 4.3.3: Homepage

**4**

**.3.4**

**. Form Name:**

**Recipe**

**View Page**

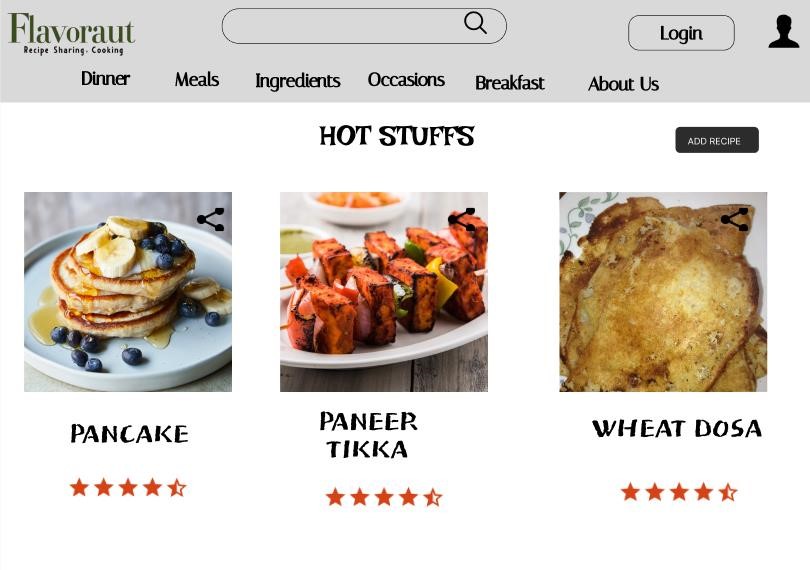


Figure 4.3.4: Recipe Viewing Page

**4.3.5. Form Name: Recipe Details Page**

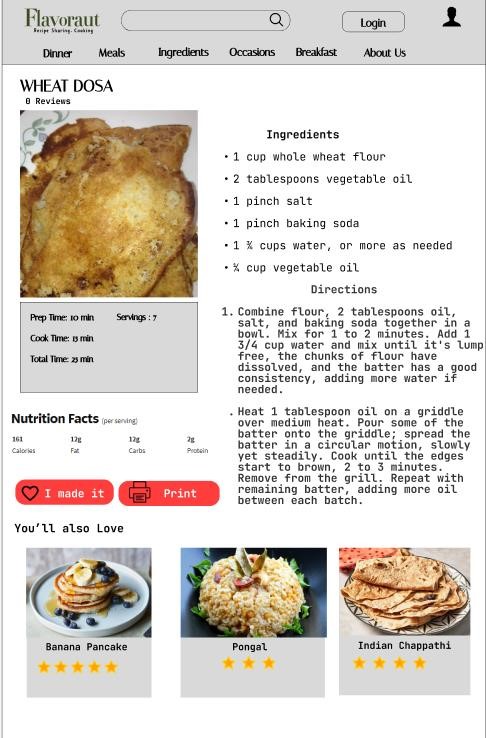


Figure 4.3.5: Recipe Details Page

### 4.4 DATABASE DESIGN

Database design is the process of defining the structure, organization, and relationships of data within a database system. It involves creating a blueprint that outlines how data will be stored, accessed, and managed. Key steps include:

1. Requirements Analysis: Understanding user needs and data requirements.
2. Conceptual Design: Creating a high-level data model (often using Entity-Relationship diagrams) that outlines entities, attributes, and relationships.
3. Logical Design: Translating the conceptual model into a logical structure, defining tables, columns, data types, and relationships.
4. Normalization: Organizing data to reduce redundancy and improve integrity through various normal forms.
5. Physical Design: Implementing the database structure in a specific database management system, optimizing for performance, security, and storage.

Effective database design ensures efficient data retrieval, scalability, and maintenance, ultimately supporting the overall functionality of applications that rely on the database.

**4.4.1 Relational Database Management System (RDBMS)**

A Relational Database Management System (RDBMS) is a sophisticated software framework designed to efficiently store, manage, and manipulate data in a structured format. The foundational concept behind an RDBMS is the relational model, which was introduced by E.F. Codd in the 1970s. In this model, data is organized into tables, which consist of rows and columns. Each table represents a distinct entity, such as customers, products, or orders, allowing for a clear and organized way to manage related information.

One of the key features of an RDBMS is the use of relationships between tables. These relationships are established through the use of primary and foreign keys. A primary key is a unique identifier for each record in a table, ensuring that no two rows have the same value in that key column. A foreign key, on the other hand, is a field in one table that uniquely identifies a row in another table, establishing a connection between the two tables.

**4.4.2 Normalization**

Normalization is a systematic approach in database design that organizes data to minimize redundancy and dependency by dividing large tables into smaller, related tables. It helps to improve data integrity and reduce anomalies during data operations like insertion, deletion, and updating.

The normalization process involves several levels, known as normal forms. Here’s a brief overview of the different types of normalization:

1. **First Normal Form (1NF)**:
   * Ensures that each column in a table contains atomic (indivisible) values and each entry is unique. It eliminates duplicate columns from the same table and creates unique identifiers for rows.
2. **Second Normal Form (2NF)**:
   * Builds on 1NF by ensuring that all non-key attributes are fully functionally dependent on the primary key. It eliminates partial dependencies, meaning no nonkey attribute should depend on a part of a composite key.
3. **Third Normal Form (3NF)**:
   * Requires that a table is in 2NF and that all the attributes are functionally dependent only on the primary key. It eliminates transitive dependencies, where non-key attributes depend on other non-key attributes.
4. **Boyce-Codd Normal Form (BCNF)**:
   * A stricter version of 3NF, BCNF ensures that for every functional dependency (A → B), A should be a superkey. This addresses certain anomalies that 3NF does not cover.

**4.4.3 Sanitization**

Sanitization is the process of cleansing input data to ensure it is safe and valid before processing or storing it in a system. It is a critical security measure that helps prevent various vulnerabilities, especially in web applications, where user input can be exploited by attackers. **Importance of Sanitization**

1. **Prevention of SQL Injection**: Sanitization is essential in preventing SQL injection attacks, where malicious users inject harmful SQL statements into input fields. By sanitizing input, you can remove or escape potentially harmful characters or commands.
2. **Cross-Site Scripting (XSS) Prevention**: Sanitizing user input helps prevent XSS attacks, where attackers inject malicious scripts into web pages viewed by other users. This is done by stripping out or encoding HTML tags and JavaScript code that could be executed in a user's browser.
3. **Data Integrity**: Sanitization ensures that the data stored in databases is clean and adheres to expected formats. This helps maintain the integrity of the data, reducing the risk of errors and inconsistencies.
4. **Improved User Experience**: By validating and sanitizing input, developers can provide clearer feedback to users, improving the overall experience by ensuring that only valid and expected data is processed.

**4.4.4 Indexing**

Indexing is a data structure technique used in databases to improve the speed and efficiency of data retrieval operations. By creating an index on one or more columns of a table, the database can quickly locate and access the relevant rows without having to scan the entire table. Here’s a detailed look at indexing, its importance, types, and how it works: **Importance of Indexing**

1. **Performance Enhancement**: Indexes dramatically speed up query performance, particularly for large datasets. Without indexing, a database must perform a full table scan to locate the required data, which can be time-consuming.
2. **Efficient Searching**: Indexes allow for faster searching, sorting, and filtering of records based on indexed columns. This is especially beneficial for operations involving WHERE clauses, JOINs, and ORDER BY statements.
3. **Reduced I/O Operations**: By minimizing the number of data pages the database engine needs to read, indexing can reduce disk I/O operations, which are often the slowest part of database performance.
4. **Support for Unique Constraints**: Indexes can enforce uniqueness on a column or a set of columns, preventing duplicate entries in a table, which is crucial for maintaining data integrity.

**4.4 TABLE DESIGN**

**1.tbl\_user\_login**

Eg.Primary key: **login\_id**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No:** | **Field name** | **Datatype (Size)** | **Key Constraints** | **Description of the field** |
| 1 | login\_id | Int | Primary key | Primary key for user login |
| 2 | Username | Varchar (100) | Not null | Name of user |
| 3 | Password | Varchar (100) | Not null | Password of user |
| 4 | Role | Varchar (100) | Not null | Role of |

**2.tbl\_customuser**

Primary Key**: user\_id**

Foreign Key**: login\_id references to tbl\_login**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No:** | **Field name** | **Datatype (Size)** | **Key Constraints** | **Description of the field** |
| 1 | Login\_id | Int | Foreign key | foreign key for user login |
| 2 | User\_id | Varchar (100) | Primary key | primary key for user |
| 3 | Name | Varchar (100) | Not Null | password of user |
| 4 | Email | Varchar (100) | Unique | email of the user |
| 5 | Password | Varchar(100) | Not Null | password of the user |
| 6 | Status | Varchar(100) | Not Null | active or inative |

**3.tbl\_recipe**

Primary Key: **recipe\_id**

Foreign Key**: user\_id references tbl\_login**

**subcategory\_id references tbl\_subcategory category\_id references tbl\_category**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No:** | **Field name** | **Datatype (Size)** | **Key**  **Constraints** | **Description of the field** |
| 1 | Recipe\_id | Int | Primary Key | Recipe id of recipe |
| 2 | User\_id | Int | Foreign Key | User\_id of recipe |
| 3 | Category\_id | Int | Foreign Key | Category id of recipe |
| 4 | Subcategory\_id | Int | Foreign Key | Subcategory\_id of recipe |
| 5 | Recipe\_name | Varchar(100) | Not Null | Name of the recipe |
| 6 | Instructions | Varchar(100) | Not Null | Instructions to prepare recipe |
| 7 | Image | Varchar(100) | Url | Image of recipe |
| 8 | Tags | Varchar(100) | Not Null | Which tags that it belongs |
| 9 | Description | Varchar(100) | Not Null | Short description of recipe |
| 10 | Nutritional\_info\_id | Int | Not Null | Id of nutritional information |

**4.tbl\_category**

Primary Key**: category\_id**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No:** | **Field name** | **Datatype (Size)** | **Key Constraints** | **Description of the field** |
| 1 | Category\_id | Int | Primary key | Primary key for category |
| 2 | Name | Varchar (100) | Not null | Name of the category |

**5.tbl\_subcategory**

Primary Key: **Subcategory\_id**

ForeignKey**:Category\_id**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No:** | **Field name** | **Datatype (Size)** | **Key Constraints** | **Description of the field** |
| 1 | Subcategory\_id | Int | Primary key | Primary key for subcategory |
| 2 | Category\_id | Int | Foreign key | Foreign key for category |
| 3 | Name | Varchar (100) | Not null | Name of the subcategory |

**6.tbl\_ingredients**

Primary Key: **Ingredients\_id**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No:** | **Field name** | **Datatype (Size)** | **Key Constraints** | **Description of the field** |
|  | Ingredients\_id | int | Primary key | Id of Ingredients |
| 1 |  |  |  |  |
| 2 | Category\_id | varchar(100) | Not null | Id of category |
| 3 | Substitution | Varchar(100) | Not Null | Substitution of the ingredients |
| 4 | Name | Varchar(100) | Not null | Ingredient Name |

**7.tbl\_nutritional\_information**

Primary Key**:** **Nutritionalinfo\_id**

Foreign Key**:** **Recipe\_id, Calories**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No:** | **Field name** | **Datatype (Size)** | **Key Constraints** | **Description of the field** |
| 1 | Nutritionalinfo\_id | Int | Primary key | Primary key for nutritional information |
| 2 | Recipe\_id | Int | Foreign key | Foreign key of recipe\_ |
| 3 | Calories | Int | Foreign key | Calories of recipe |
| 4 | Protein | Int | Not null | Protein of recipe |
| 5 | Fat | Int | Not null | Fat of recipe |
|  |  |  |  | Carbohydrates of recipe |
| 6 | Carbohydrates | Int | Not null |  |
| 7 | Sugar | Int | Not null | Sugar content of recipe |
|  |  |  |  | Fiber amount of recipe |
| 8 | Fiber | Int | Not null |  |

**8.tbl\_rating**

Primary Key**: Rating\_id**

ForeignKey: **Recipe\_id, User\_id**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No:** | **Field name** | **Datatype (Size)** | **Key Constraints** | **Description of the field** |
| 1 | Rating\_id | Int | Primary key | Primary key for rating |
| 2 | Recipe\_id | Int | Foreign key | Foreign key recipe\_id |
| 3 | User\_id | Int | Foreign key | Foreign key of user\_id |
| 4 | Rating | Int | Not null | Rating of recipe |
|  |  | DateTime | Not null | Date and time of rating |
| 5 | Created\_at |  |  |  |

**9.tbl\_recipeingredient**

Foreign Key:**Recipe\_id, Ingredient\_id**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No:** | **Field name** | **Datatype (Size)** | **Key Constraints** | **Description of the field** |
| 1 | Recipe\_id | Int | Foreign Key | Foreign key for recipe |
| 2 | Ingredient\_id | Varchar (100) | Foreign Key | Foreign key for ingredient |
| 3 | Quantity | Decimal(10) | Not null | Quantity of recipe |
| 4 | Measurement | Char (10) | Not null | Measurement of recipe |

**CHAPTER 5**

**SYSTEM TESTING**

### 5.1 INTRODUCTION

Software testing is the process of systematically evaluating a software program to determine if it operates as intended. It typically involves using methods for verification and validation. Validation ensures that the product aligns with its specifications, while verification may encompass reviews, analyses, inspections, and walkthroughs. Static analysis focuses on examining the software's source code to detect issues, whereas dynamic analysis observes its behavior during runtime, collecting data like execution traces, timing profiles, and test coverage details.

Testing comprises a series of organized activities that commence with individual modules and extend to the integration of the entire computer-based system. The primary goals of testing include the identification of errors and bugs within the software, confirmation that the software adheres to its specifications, and validation of its performance against requirements. Testing can encompass assessments of correctness, implementation efficiency, and computational complexity.

A successful test is one that uncovers previously undetected errors, and a valuable test case holds a high likelihood of revealing such errors. Testing is a critical element in achieving system testing objectives and encompasses various techniques like functional testing, performance testing, and security testing.

### 5.2 TEST PLAN

A test plan is a comprehensive document that delineates the necessary steps for executing diverse testing methodologies. It serves as a roadmap for the activities that must be carried out during the testing process. Software developers are tasked with crafting computer programs, documentation, and the associated data structures. Their responsibility entails scrutinizing each component of the program to ensure that it aligns with its intended purpose. To mitigate self-assessment biases, it is common practice to establish an independent test group (ITG).

When setting testing objectives, it is essential to express them in measurable terms. This may involve metrics such as mean time to failure, the cost associated with identifying and rectifying defects, the remaining defect density or frequency of occurrences, and the number of test work-hours required for regression tests.

The various tiers of testing encompass:

* Unit testing
* Integration testing
* Data validation testing
* Output testing

**5.2.1 Unit Testing**

Unit testing is a software testing approach that concentrates on the validation of individual software components or modules. Its primary aim is to assess the functionality of the smallest units within the software design, ensuring they operate as intended. Typically, unit testing takes a white-box perspective, and it's possible to test multiple components simultaneously. The testing process relies on the component-level design description as a reference, allowing the identification of critical control paths and potential faults within the module's scope. In the course of unit testing, the focus lies on testing the module's interface to confirm that data effectively enters and exits the software unit under examination. The local data structure is meticulously examined to guarantee that temporarily stored data maintains its integrity throughout the execution of algorithms. Boundary conditions are examined to ensure that all statements within a module have been executed at least once. Additionally, error handling paths are scrutinized to verify the software's capability to handle errors correctly.

Prior to conducting any other testing, it is imperative to ensure the proper flow of data through a module's interface. If data cannot flow in and out of the system correctly, all other tests become inconsequential. Another crucial aspect of unit testing involves a selective evaluation of execution paths to anticipate potential errors and ensure that error-handling mechanisms are in place to redirect or halt processes in the event of an error. Finally, boundary testing is executed to confirm that the software operates flawlessly even under extreme conditions.

**5.2.2 Integration Testing**

Integration testing is a methodical process that involves not only building the program structure but also conducting tests to detect interface-related issues. The primary goal is to construct program structure based on the design, incorporating components that have undergone unit testing. Subsequently, the entire program is subjected to testing. Rectifying errors during integration testing can be a formidable task, mainly due to the scale of the overall program, which makes it challenging to pinpoint the sources of errors. It can often be a dynamic process were fixing one set of errors may lead to the emergence of new ones, creating what seems like an unending cycle. Upon completing unit testing for all modules within the system, these modules are integrated to identify and resolve any inconsistencies in their interfaces. Any disparities in program structures are addressed, ultimately resulting in the creation of a cohesive and unified program structure.

**5.2.3 Validation Testing or System Testing**

The concluding phase of the testing process entails evaluating the complete software system in its entirety, encompassing all forms, code, modules, and class modules. This phase is typically known as system testing or black box testing. Black box testing places its emphasis on assessing the software's functional requirements. In this approach, a software engineer can establish input conditions that comprehensively examine each program requirement. The key categories of errors that black box testing aims to identify encompass incorrect or absent functions, interface discrepancies, issues in data structures or external data access, performance-related problems, initialization errors, and termination errors.

**5.2.4 Output Testing or User Acceptance Testing**

User acceptance testing is conducted to verify that the system aligns with the business requirements and fulfills the user's expectations. It's crucial to engage end users throughout the development process to ensure that the software caters to their needs and anticipations. In the course of user acceptance testing, the input and output screen designs are scrutinized using various types of test data. The meticulous preparation of test data plays a vital role in ensuring a thorough assessment of the system. Any discrepancies uncovered during testing are duly addressed and rectified, and these corrections are duly documented for future reference.

**5.2.5 Automation Testing**

Automation testing is a software testing approach that leverages specialized automated testingsoftware tools to run a suite of test cases. Its primary objective is to validate that the software or equipment functions precisely as intended. Automation testing serves to identify defects, bugs, and other issues that may surface during the course of product development. While certain types of testing, like functional or regression testing, can be carried out manually, automating the process offers several advantages. Automation testing can be conducted at any time, employing scripted sequences to assess the software. The outcomes are documented, and these results can be compared to previous test runs. Automation developers commonly write code in programming languages such as C#, JavaScript, and Ruby.

**5.2.6 Selenium Testing**

Selenium is an open-source automated testing framework designed for validating web applications across various browsers and platforms. Selenium facilitates the creation of test scripts in different programming languages, such as Java, C#, and Python. The framework was initially developed by Jason Huggins, an engineer at Thought Works, in 2004, while he was working on a web application that required frequent testing. He introduced a JavaScript program called "JavaScriptTestRunner" to automate browser actions and enhance testing efficiency. Since its inception, Selenium has evolved and is continually improved by a community of contributors. In addition to Selenium, another widely used tool for automated testing is Cucumber. Cucumber is an open-source software testing framework that supports behavior-driven development (BDD). It enables the creation of executable specifications in a human-readable format called Gherkin. One of the significant advantages of Cucumber is its capability to bridge the communication gap between business stakeholders and technical teams. By using a common language, Cucumber enhances effective communication and collaboration during the testing process. It promotes a shared understanding of requirements and ensures that the developed software aligns with the intended business objectives. Cucumber can be seamlessly integrated with Selenium to harness the benefits of both tools. Selenium is employed for interacting with web browsers and automating browser actions, while Cucumber provides a structured framework for organizing and executing tests. This combination empowers the creation of end-to-end tests that validate the behavior of web applications across diverse browsers and platforms. It does so using a format that is both business-readable and easily maintainable.

**Test Case 1-Login Code**

package Definition; import org.openqa.selenium.By; import org.openqa.selenium.WebDriver; import org.openqa.selenium.firefox.FirefoxDriver; import io.cucumber.java.en.And; import io.cucumber.java.en.Given; import io.cucumber.java.en.Then; import io.cucumber.java.en.When; public class stepdefinition {

WebDriver driver=null; @Given("browser is open") public void browser\_is\_open() {

System.out.println("Inside step-Browser is open");

System.setProperty("webdriver.gecko.marionette","C:\\Users\\sandr\\eclipseworkspace\\1234\\src\\test\\resources\\drivers\\geckodriver.exe");

driver=new FirefoxDriver(); driver.manage().window().maximize();

}

@And("user is on login page") public void user\_is\_on\_login\_page() throws Exception { driver.navigate().to("http://127.0.0.1:8000/");

Thread.sleep(2000);

}

@When("user enters username and password") public void user\_enters\_username\_and\_password() throws Throwable{ driver.findElement(By.id("Email")).sendKeys("sandra@gmail.com"); driver.findElement(By.id("password")).sendKeys("Sandra@02");

}

@When("User clicks on login") public void user\_clicks\_on\_login() { driver.findElement(By.id("loginbutton")).click(); }

@Then("user is navigated to the home page")

public void user\_is\_navigated\_to\_the\_home\_page() throws Exception { driver.findElement(By.id("button")).isDisplayed();

Boolean isLogoutDisplayed = driver.findElement(By.id("logout")).isDisplayed();

if (isLogoutDisplayed) {

System.out.println("Login successful and user is on the home page");

} else {

System.out.println("Login failed or not navigated to the home page");

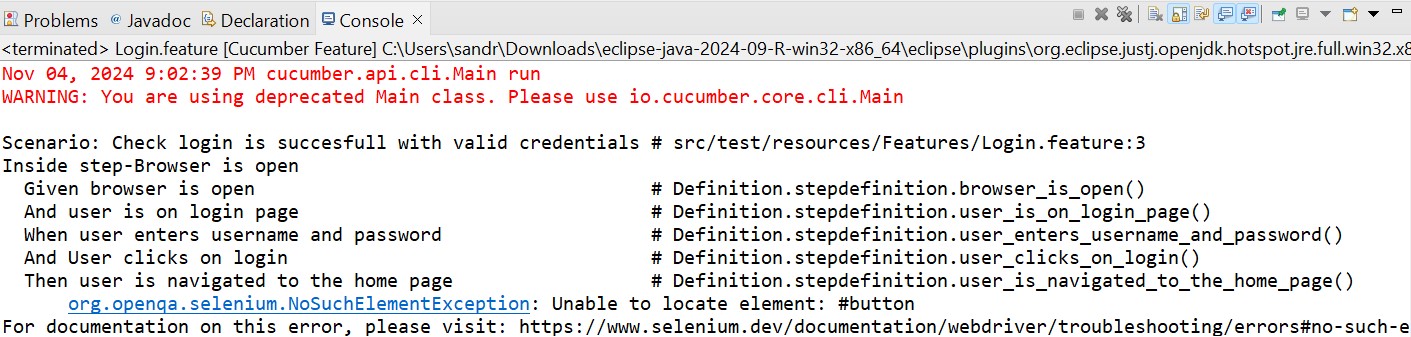
}

Thread.sleep(2000); driver.close(); driver.quit();

}

}

**Eg.Screenshot**



**Eg.Test Report**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test Case 1** | | | | | |
| **Project Name: Recipe Sharing Platform** | | | | | |
| **Login Test Case** | | | | | |
| **Test Case ID: Test\_1** | | | **Test Designed By:** Sandra Mohan | | |
| **Test Priority**  **(** **Low/Medium/High):** Medium | | | **Test Designed Date:** 15/10/2024 | | |
| **Module Name**: Login Page | | | **Test Executed By :** Ms. Gloriya Mathew | | |
| **Test Title :** User Login | | | **Test Execution Date:** 15/10/2024 | | |
| **Description:** Verify login with valid email and password | | |  | | |
| **Pre-Condition :**User has valid username and password | | | | | |
| **Step** | **Test Step** | **Test Data** | **Expected Result** | **Actual Result** | **Status(Pass/**  **Fai l)** |
| 1 | Navigation to Login Page |  | Homepage should be displayed | Login page displayed | Pass |
| 2 | Provide valid email | Email:  sandra@gmail .com | User should be able to login | User logged in and navigated to user home page | Pass |
| 3 | Provide valid password | Password: sandra@22 |
| 4 | Click on login button |  |
| **Post-Condition:** User is validated with database and successfully login into account. The account session details are logged in database. | | | | | |

**Test Case 2- Add Recipe Code**

package Definition; import org.openqa.selenium.By; import org.openqa.selenium.WebDriver; import org.openqa.selenium.firefox.FirefoxDriver; import io.cucumber.java.en.And; import io.cucumber.java.en.Given; import io.cucumber.java.en.Then;

import io.cucumber.java.en.When; public class addrecipe {

WebDriver driver=null;

@Given("browser is open for recipe adding page") public void browser\_is\_open() {

System.***out***.println("Inside step-browser is open for recipe adding page"); System.*setProperty*("webdriver.gecko.marionette","C:\\Users\\sandr\\eclipseworkspace\\1234\\src\\test\\resources\\drivers\\geckodriver.exe");

driver=new FirefoxDriver(); driver.manage().window().maximize();

}

@And("user is on addrecipe page") public void user\_is\_on\_login\_page() throws Exception

{

driver.navigate().to("http://127.0.0.1:8000/addrecipe/");

Thread.*sleep*(2000);

}

@When("user enters Recipe Name,Category, Subcategory, Servings, \r\n"+ " Prep

Time (minutes), Cook Time (minutes), Add Ingredient, Step 1,Tags, Recipe Image")

public void user\_enters\_username\_and\_password() throws Throwable

{ driver.findElement(By.*id*("recipename")).sendKeys("burger"); driver.findElement(By.*id*("category")).sendKeys("Non-Vegetarian"); driver.findElement(By.*id*("subcategory")).sendKeys("Snacks and Appetizers"); driver.findElement(By.*id*("servings")).sendKeys("2"); driver.findElement(By.*id*("prep\_time")).sendKeys("5"); driver.findElement(By.*id*("cook\_time")).sendKeys("5"); driver.findElement(By.*id*("ingredients")).sendKeys("Bun,Egg,Chicken"); driver.findElement(By.*id*("instructions")).sendKeys("prepare 2 slice of bun and place the ingredients"); driver.findElement(By.*id*("tags")).sendKeys("#snacks,#appetizers"); driver.findElement(By.*id*("image")).sendKeys("D:\\MINI\_PROJECT\\recipe\_sharing \_platform\\media\\recipes\\burger.jpeg");

}

@And("User clicks on Add Recipe") public void user\_clicks\_on\_Add\_Recipe() {

driver.findElement(By.*xpath*("//button[contains(text(),'Add Recipe')]")).click();

}

@Then("user is navigated to the recipe page") public void user\_is\_navigated\_to\_the\_home\_page() throws Exception { driver.findElement(By.*id*("button")).isDisplayed();

BooleanisLogoutDisplayed=driver.findElement(By.*id*("logout")).isDisplayed();

if (isLogoutDisplayed) {

System.*out*.println("Recipe added successful and user is on the recipe page");

}

else {

System.*out*.println("Recipe failed or not navigated to the recipe page");

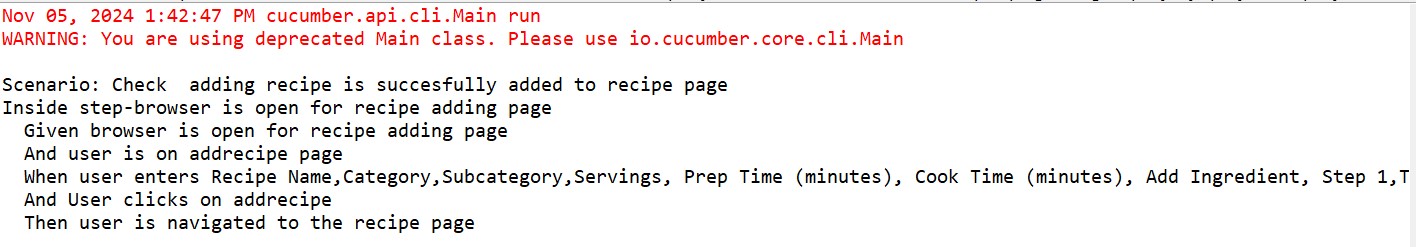
}

Thread.*sleep*(2000); driver.close(); driver.quit();

}}

**Eg.S**

**creenshot**



|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Test Case 2** | | | | | | | | | | | |
| **Project Name: Recipe Sharing Platform** | | | | | | | | | | | |
| **Recipe Adding Test Case** | | | | | | | | | | | |
| **Test Case ID: Test\_2** | | | | | | **Test Designed By:** Sandra Mohan | | | | | |
| **Test Priority**  **(** **Low/Medium/High):** High | | | | | | **Test Designed Date:** 05/11/2024 | | | | | |
| **Module Name**: Recipe Add Page | | | | | | **Test Executed By :** Ms. Gloriya Mathew | | | | | |
| **Test Title :** User Adding Recipe | | | | | | **Test Execution Date:** 05/11/2024 | | | | | |
| **Description:** Adding recipe and its details | | | | | |  | | | | | |
| **Pre-Condition :**User adding recipename, ingredients, tags etc | | | | | | | | | | | |
| **Step** | | **Test Step** | | **Test Data** | | **Expected Result** | | **Actual Result** | | **Status(Pass/**  **Fai l)** | |
| 1 | | Navigation to addrecipe Page | |  | | Recipe adding page should be displayed | | Recipe adding page displayed | | Pass | |
| 2 | | Provide recipe name | | Recipename:  burger | | User should be able to add recipes | | User added recipes | | Pass | |
| 3 | | Provide category | | Category: Non-  vegetarian | |
| 4 | | Provide subcategor y | | Subcategory:  Snacks and  Appetizers | |
| 5 | | servings | | Servings:2 | |  | |  | |  | |
| 6 | | Prep\_time | | Prep\_time:5 | |  | |  | |  | |
| 7 | | Cook\_time | | Cook\_time:5 | |  | |  | |  | |
| 8 | | Ingredients | | Ingredients:  Bun, Egg,  Chicken | |  | |  | |  | |
| 9 | | Instruction | | Instructions: prepare 2 slice of bun and place the ingredients | |  | |  | |  | |
| 10 | | Tags | | Tags:  #snacks | |  | |  | |  | |
| 11. | | Image | | Image:  D:\\MINI\_PR OJECT\\recip e\_sharing\_plat form\\media\\r ecipes\\burger.  jpeg" | |  | |  | |  | |
| **Post-Condition:** User can add recipes into database and confirmation message is shown | | | | | | | | | | | |

**Test Case 3-Edit Recipe**

#### Code

package Definition; import org.openqa.selenium.By; import org.openqa.selenium.WebDriver; import org.openqa.selenium.WebElement; import org.openqa.selenium.firefox.FirefoxDriver; import io.cucumber.java.en.And; import io.cucumber.java.en.Given; import io.cucumber.java.en.Then; import io.cucumber.java.en.When;

public class edit { WebDriver driver = null;

@Given("browser is open for editing recipe") public void browser\_is\_open\_for\_editing\_recipe() {

System.*out*.println("Inside step-browser is open for editing recipe");

System.*setProperty*("webdriver.gecko.marionette","C:\\Users\\sandr\\eclipseworkspace\\1234\\src\\test\\resources\\drivers\\geckodriver.exe"); driver=new FirefoxDriver(); driver.manage().window().maximize();

}

@And("user is on editrecipe page") public void user\_is\_on\_editrecipe\_page() throws InterruptedException { driver.navigate().to("http://127.0.0.1:8000/recipe/20/edit/");

Thread.*sleep*(2000); // Wait for page to load

}

@When("user enters Recipe Name, Category, Subcategory, Servings, Prep Time (minutes), Cook Time (minutes), Add Ingredient and all steps") public void user\_enters\_recipe\_name\_category\_subcategory\_servings\_prep\_time\_minutes\_cook\_time\_minut es\_add\_ingredient\_and\_all\_steps() {

// Example actions to fill the form

WebElement recipeNameField = driver.findElement(By.*id*("recipeName")); recipeNameField.sendKeys("Delicious Pancakes");

WebElement categoryField = driver.findElement(By.*id*("category")); categoryField.sendKeys("Breakfast");

WebElement subcategoryField = driver.findElement(By.*id*("subcategory")); subcategoryField.sendKeys("Pancakes");

WebElement servingsField = driver.findElement(By.*id*("servings")); servingsField.sendKeys("4");

WebElement prepTimeField = driver.findElement(By.*id*("prepTime")); prepTimeField.sendKeys("10");

WebElement cookTimeField = driver.findElement(By.*id*("cookTime")); cookTimeField.sendKeys("20");

WebElement ingredientField = driver.findElement(By.*id*("ingredients")); ingredientField.sendKeys("Flour, Eggs, Milk, Sugar");

// Assuming you have steps to fill out instructions as well

WebElement instructionsField = driver.findElement(By.*id*("instructions")); instructionsField.sendKeys("Mix all ingredients and cook on a skillet.");

}

@And("User clicks on update") public void user\_clicks\_on\_update\_button() { driver.findElement(By.*xpath*("//button[text()='Update Recipe']")).click();

}

@Then(" Success Message is shown")

public void shown\_Success\_Message() throws InterruptedException { Thread.*sleep*(2000); // Wait for success message to appear

// Check for success message

Boolean isSuccessDisplayed = driver.findElements(By.*xpath*("//\*[contains(text(),'recipe updated successfully!')]")).size() > 0;

if (isSuccessDisplayed) {

System.*out*.println("recipe updated successfully.");

} else {

System.*out*.println("Success message not found.");

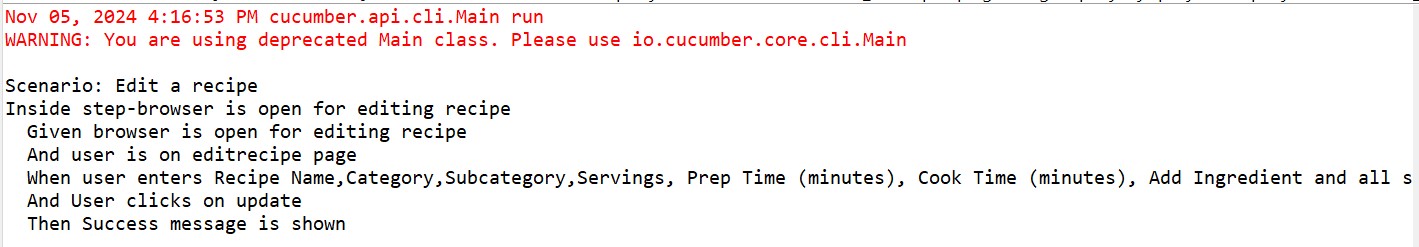
}

driver.close(); driver.quit();

}

}

**Eg: Screenshot**



|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test Case 3** | | | | | |
| **Project Name: Recipe Sharing Platform** | | | | | |
| **Edit Recipe Test Case** | | | | | |
| **Test Case ID: Test\_3** | | | **Test Designed By:** Sandra Mohan | | |
| **Test Priority**  **(** **Low/Medium/High):** High | | | **Test Designed Date:** 05/11/2024 | | |
| **Module Name**: Edit Recipe Page | | | **Test Executed By :** Ms. Gloriya Mathew | | |
| **Test Title :** User Edit Recipe Page | | | **Test Execution Date:** 05/11/2024 | | |
| **Description:** Updating the recipes | | |  | | |
| **Pre-Condition :**User Updating the recipe details | | | | | |
| **Step** | **Test Step** | **Test Data** | **Expected Result** | **Actu**  **al**  **Resu**  **lt** | **Status(Pass/**  **Fai l)** |
| 1 | Navigation to editrecipe Page |  | Recipe editing should be displayed | Recipe editing page displayed | Pass |
| 2 | Provide recipename | Recipename:  burger | User should be able to login | User logged in and navigated to user home page | Pass |
| 3 | Provide category | Category: nonvegetarian |
| 4 | Provide subcategory | Subcategory:  snacks |
| 5 | servings | Servings:2 |
| 6 | Prep\_time | Prep\_time: 5 |
| 7 | Cook\_time | Cook-time: 4 |
| 8 | ingredients | Ingredients: Bun,egg, chicken |
| 9 | instructions | Instructions: Mix all ingredients and cook |
| 10 | Click on update button |  |
| **Post-Condition:** User can update recipes into database and confirmation message is shown | | | | | |

**CHAPTER 6**

**IMPLEMENTATION**

### 6.1 INTRODUCTION

The implementation phase of a project marks the transformation of the design into a fully functional system. It stands as a critical juncture in ensuring the success of the new system, as it necessitates building user confidence in its effective and accurate operation. Of great importance during this phase are user training and the creation of comprehensive documentation. The conversion process may happen alongside user training or at a later stage. Implementation encompasses the process of turning a freshly revised system design into an operational and functional system. In this stage, the user department shoulders the primary workload, experiences the most substantial changes, and wields the greatest influence over the existing system. If implementation is poorly planned or lacks control, it can lead to confusion and disorder. Whether the new system is entirely novel, replaces an existing manual or automated system, or modifies an existing one, a well-executed implementation is essential to align with the organization's needs. System implementation encompasses all the activities necessary to transition from the old system to the new one. The new system can only be implemented after thorough testing confirms that it operates according to the specified requirements. System personnel assess the feasibility of the new system. Implementation demands substantial effort in three key areas: education and training, system testing, and changeover. The implementation phase calls for meticulous planning, exploration of system and constraints, and the development of methods to achieve a smooth transition.

### 6.2 IMPLEMENTATION PROCEDURES

Software implementation involves the process of installing software in its intended environment and verifying that it meets its intended purpose while functioning correctly. In certain organizations, the software development project may be initiated by someone who won't be using the software personally. At the initial stages, doubts about the software may arise, but it's crucial to prevent resistance from building up.

This can be achieved by:

* Ensuring that active users are well-informed about the advantages of the new system, thus building their confidence in the software.
* Offering proper guidance to users to ensure they feel at ease when using the application. It's important for users to understand that the server program needs to be operational on the server before they can access the system. Without the server object running, the desired processes will not occur.

**6.2.1 User Training**

User training is a crucial phase intended to get users ready for testing and transitioning to a new system. To realize the anticipated advantages of a computer-based system, it's vital for individuals involved to have confidence in their roles within the new system. As systems become increasingly complex, the necessity for training becomes more pronounced. Through user training, individuals gain knowledge on tasks such as data entry, handling error messages, querying databases, and utilizing routines for generating reports and executing other essential functions. This ensures that users are well-prepared and capable of effectively utilizing the system.

**6.2.2 Training on the Application Software**

Following the provision of fundamental computer awareness training, it becomes crucial to deliver training on the new application software to the users. This training should encompass the fundamental principles of using the new system, including screen navigation, screen design, available on-screen assistance, potential data entry errors, corresponding validation checks for each entry, and methods to rectify entered data. Furthermore, the training should encompass user or group specific information that is essential for effective system or module utilization. It's worth noting that this training may vary among distinct user groups and hierarchical levels.

**6.2.3 System Maintenance**

The maintenance phase is a crucial aspect of the software development cycle, as it is the time when the software is actually put to use and performs its intended functions. Proper maintenance is essential to ensure that the system remains functional, reliable, and adaptable to changes in the system environment. Maintenance activities go beyond simply identifying and fixing errors or bugs in the system. It may involve updates to the software, modifications to its functionalities, and enhancements to its performance, among other things. In essence, software maintenance is an ongoing process that requires continuous monitoring, evaluation, and improvement of the system to meet changing user needs and requirements.

**6.2.4 Hosting**

**Eg.000Webhost**

Explanation

**Procedure for hosting a website on 000Webhost:**

Step 1: explanation

Step 2: explanation

Step 3: explanation

.

.

**Hosted Website:**

**Hosted Link: https://abc.000webhostapp.com**

**Screenshot**

**CHAPTER 7**

**CONCLUSION AND FUTURE SCOPE**

### 7.1 CONCLUSION

The "Recipe Sharing Platform" is a dynamic online community designed for culinary enthusiasts to discover, share, and manage recipes with ease. The platform prioritizes user engagement through personalized profiles, an extensive recipe library, rating and review systems, and fostering an interactive and supportive culinary space. With advanced functionalities like ingredient substitutions, meal planning, nutritional insights, it offers a comprehensive experience for users. Administrators and Recipe Managers ensure quality content, oversee user support, and facilitate community events like recipe contests. This platform is a go-to destination for anyone looking to explore, connect, and thrive in the culinary world.

### 7.2 FUTURE SCOPE

**Enhanced Social and Community Engagement:** Building on the rating, review, and competition features, the platform could incorporate live community events, collaborative recipe projects, or user groups based on interests (e.g., vegan cooking, international cuisines).

**Expanded Ingredient Knowledge Base:** The ingredient knowledge base can be developed further by adding videos, tips from chefs, and detailed substitution guides to make it an authoritative resource.

**Personalized Nutrition and Dietary Tracking:** Enhance the meal planning and nutritional information features by enabling personalized dietary tracking, helping users manage their dietary goals and preferences over time.

**Collaborative Cooking Features:** Add options for users to plan, cook, and share recipes together in real-time, making the platform more interactive.

**Expanded Recipe Contests and Rewards:** Develop the competition feature to include seasonal contests, collaborations with brands, or live voting events, creating more excitement and incentives for user participation.

**CHAPTER 8**

**BIBLIOGRAPHY**

**REFERENCES:**

* Sanjeev Kapoor, “How to Cook Indian: More Than 500 Classic Recipes for the Modern Kitchen,” Stewart, Tabori & Chang, 2011.
* Sashi Kumar, “Kerala Cook Book,” DC Books, 2000.
* Mallika Basu, “Masala: Indian Cooking for Modern Living,” Bloomsbury Publishing, 2018.
* K.M. Mathew, “Flavours of the Spice Coast,” Penguin Books, 1999.

**WEBSITES:**

* https://www.allrecipes.com/

* https://chatgpt.com/

• https://getbootstrap.com/

* https://www.w3schools.com/django/

## CHAPTER 9 APPENDIX

### 9.1 Sample Code

#### 1. Login

{% load static %}

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Login Page</title>

<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css" rel="stylesheet"> <style> body { margin: 0; display: flex; height: 100vh; font-family: Arial, sans-serif;

}

.left-panel { width: 50%; background-image: url("{% static 'images/login.jpg' %}"); background-size: cover; background-position: center;

}

.right-panel { width: 50%; display: flex; justify-content: center; align-items: center;

}

.login-form { width: 80%; max-width: 400px;

}

.login-form h1 { color: #ff6f00; font-size: 36px; font-weight: bold; margin-bottom: 30px;

}

.login-form label { font-weight: 500; margin-bottom: 5px; display: block;

}

.login-form input { width: 100%; padding: 12px; margin-bottom: 15px; border: 1px solid #ccc; border-radius: 4px;

}

.login-form button { width: 100%; padding: 12px; background-color: #ff6f00; color: white; border: none; border-radius: 4px; font-size: 16px; font-weight: bold; cursor: pointer;

}

.login-form button:hover { background-color: green;

}

.login-form .google-button { background-color: #4285F4; color: white; display: flex; align-items: center; justify-content: center; padding: 10px; border: none; border-radius: 4px; font-size: 16px; font-weight: bold; margin-top: 10px; cursor: pointer;

}

.login-form .google-button img { width: 20px; margin-right: 10px;

}

.login-form .signup-link, .login-form .forgot-password { text-align: center; margin-top: 20px;

}

.login-form .signup-link a, .login-form .forgot-password a { color: green; text-decoration: none;

}

.login-form .signup-link a:hover, .login-form .forgot-password a:hover { text-decoration: underline;

}

</style>

</head>

<body>

<div class="left-panel"></div>

<div class="right-panel">

<form class="login-form" action="{% url 'login' %}" method="POST">

{% csrf\_token %}

<h1>Flavoraut</h1>

{% if error %}

<div class="error-message">{{ error }}</div>

{% endif %}

<div>

<label for="Email">Email</label>

<input type="text" id="Email" name="Email" required>

</div>

<div>

<label for="password">Password</label>

<input type="password" id="password" name="password" required> </div>

<button id="loginbutton" type="submit">Log in</button>

<div class="forgot-password">

<a href="{% url 'forgot\_password' %}">Forgot Password?</a> </div>

<!-- Google Sign In -->

<div class="google-button">

<a href="{% url 'social:begin' 'google-oauth2' %}">

<img src="{% static 'images/google-logo.png' %}" alt="Google Logo"> Sign in with Google

</a>

</div>

<div class="signup-link">

Don't have an account? <a href="{% url 'signup' %}">Join now</a>

</div>

</form>

</div>

</body>

</html>

#### 2. Addrecipe

{% load static %}

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Add Recipe</title>

<link rel="stylesheet" href="{% static 'styles.css' %}">

<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.css" rel="stylesheet">

</head>

<body>

{% include 'navbar.html' %}

<div class="form-container">

<h1>Add New Recipe</h1>

<form method="post" action="{% url 'addrecipe' %}" enctype="multipart/form-data"> {% csrf\_token %}

<!-- Recipe Name -->

<div class="form-group">

<label for="recipename">Recipe Name:</label>

<input type="text" id="recipename" name="recipename" required>

<p class="error">{{ form.recipename.errors }}</p>

</div>

<!-- Category -->

<div class="form-group">

<label for="category">Category:</label>

<select id="category" name="category" required onchange="loadSubcategories()">

<option value="">Select a category</option>

{% for category in categories %}

<option value="{{ category.category\_id }}">{{ category.name }}</option>

{% endfor %}

</select>

<p class="error">{{ form.category.errors }}</p>

</div>

<!-- Subcategory -->

<div class="form-group">

<label for="subcategory">Subcategory:</label>

<select id="subcategory" name="subcategory">

<option value="">Select a subcategory</option>

</select>

<p class="error">{{ form.subcategory.errors }}</p>

</div>

<!-- New fields for servings and timings -->

<div class="form-row">

<div class="form-col">

<label for="servings">Servings:</label>

<input type="number" id="servings" name="servings" min="1" required>

</div>

<div class="form-col">

<label for="prep\_time">Prep Time (minutes):</label>

<input type="number" id="prep\_time" name="prep\_time" min="0" required>

</div>

<div class="form-col">

<label for="cook\_time">Cook Time (minutes):</label>

<input type="number" id="cook\_time" name="cook\_time" min="0" required>

</div>

</div>

<!-- Ingredients Selection -->

<div class="form-group">

<label for="ingredients">Add Ingredient:</label>

<div class="d-flex">

<select id="ingredients" class="flex-grow-1 me-2"> <option value="">Select an ingredient</option>

{% for ingredient in ingredients %}

<option value="{{ ingredient.ingredient\_id }}">{{ ingredient.name }}</option>

{% endfor %}

</select>

<input type="text" id="newIngredient" class="flex-grow-1 me-2" placeholder="Or

type a new ingredient">

<button type="button" class="btn btn-primary" onclick="addIngredientRow()">Add</button> <button type="button" class="btn btn-secondary ms-2" data-bs-toggle="modal" databs-target="#addIngredientModal">New Ingredient</button>

</div>

</div>

<!-- Ingredients Table -->

<div id="ingredientDetailsContainer">

<table id="ingredientsTable">

<thead>

<tr>

<th>Ingredient</th>

<th>Quantity</th>

<th>Measurement</th>

<th>Action</th>

</tr>

</thead>

<tbody id="ingredientRows">

<!-- Rows will be added dynamically -->

</tbody>

</table>

</div>

<!-- Instructions -->

<div class="form-group">

<label for="instructions" class="step-label">Step 1:</label>

<div id="instructionsContainer">

<textarea name="instructions\_1" id="instructions" rows="3" placeholder="Step 1" required></textarea>

</div>

<div class="add-step-btn" onclick="addStep()">Add Step</div>

<p class="error">{{ form.instructions.errors }}</p>

</div

<!-- Tags -->

<div class="form-group">

<label for="tags">Tags:</label>

<input type="text" id="tags" name="tags" required>

<p class="error">{{ form.tags.errors }}</p> </div>

<!-- Image Upload -->

<div class="form-group">

<label for="image">Recipe Image:</label>

<input type="file" id="image" name="image" accept="image/\*">

<p class="error">{{ form.image.errors }}</p>

</div>

<!-- Submit Button -->

<button type="submit">Add Recipe</button>

</form>

</div>

<!-- Add Ingredient Modal -->

<div class="modal fade" id="addIngredientModal" tabindex="-1" arialabelledby="addIngredientModalLabel" aria-hidden="true">

<div class="modal-dialog">

<div class="modal-content">

<div class="modal-header">

<h5 class="modal-title" id="addIngredientModalLabel">Add New Ingredient</h5> <button type="button" class="btn-close" data-bs-dismiss="modal" arialabel="Close"></button>

</div>

<div class="modal-body">

<form id="addIngredientForm">

{% csrf\_token %}

<div class="mb-3">

<label for="ingredientName" class="form-label">Name</label>

<input type="text" class="form-control" id="ingredientName" name="name" required>

</div>

<div class="mb-3">

<label for="ingredientSubstitutions" class="form-label">Substitutions</label>

<input type="text" class="form-control" id="ingredientSubstitutions" name="substitutions"> </div>

<div class="mb-3">

<label for="ingredientCategory" class="form-label">Category</label>

<select class="form-select" id="ingredientCategory" name="category\_id" required>

{% for category in categories %}

<option value="{{ category.category\_id }}">{{ category.name }}</option>

{% endfor %}

</select>

</div>

<button type="submit" class="btn btn-primary">Add Ingredient</button>

</form>

</div>

</div>

</div>

</div>

<scriptsrc="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/js/bootstrap.bundle.min.js"></script

<script> let stepCount = 1; let ingredientRowCount = 0; let isUpdatingIngredients = false; function addStep() { stepCount++;

const newStep = document.createElement('textarea'); newStep.name = 'instructions\_' + stepCount; newStep.rows = 3; newStep.placeholder = 'Step ' + stepCount; newStep.required = true;

const newLabel = document.createElement('label'); newLabel.className = 'step-label'; newLabel.textContent = 'Step ' + stepCount + ':'; const container = document.getElementById('instructionsContainer'); container.appendChild(newLabel); container.appendChild(newStep);

}

document.addEventListener('DOMContentLoaded', function() { const categorySelect = document.getElementById('category'); const ingredientsSelect = document.getElementById('ingredients'); if (categorySelect) {

categorySelect.addEventListener('change', function() { if (!isUpdatingIngredients) { updateIngredients();

}

});

} if (ingredientsSelect) { ingredientsSelect.addEventListener('change', function() { document.getElementById('newIngredient').value = '';

});

}

if (categorySelect && categorySelect.value) { updateIngredients();

}

});

function addIngredientRow() { const selectedIngredient = document.getElementById('ingredients'); const newIngredientInput = document.getElementById('newIngredient'); let ingredientId, ingredientName; if (selectedIngredient.value) { ingredientId = selectedIngredient.value; ingredientName = selectedIngredient.options[selectedIngredient.selectedIndex].text;

} else if (newIngredientInput.value.trim()) { ingredientId = 'new\_' + Date.now();

ingredientName = newIngredientInput.value.trim();

} else { console.log('No ingredient selected or entered'); return;

}

constexistingIngredient=document.querySelector(`input[name^="ingredient\_"][value="${ingredie ntId}"]`); if (existingIngredient) { alert('This ingredient is already in the recipe.'); selectedIngredient.value = ''; newIngredientInput.value = ''; return;

}

ingredientRowCount++; const newRow = document.createElement('tr'); newRow.innerHTML = `

<td>

<input type="hidden" name="ingredient\_${ingredientRowCount}" value="${ingredientId}">

<input type="hidden" name="ingredient\_name\_${ingredientRowCount}"value="${ingredientName}">

${ingredientName}

</td>

<td><input type="number" name="quantity\_${ingredientRowCount}" step="0.01" required></td>

<td>

<select name="measurement\_${ingredientRowCount}" required>

<option value="g">Grams (g)</option>

<option value="kg">Kilograms (kg)</option>

<option value="ml">Milliliters (ml)</option>

<option value="l">Liters (l)</option>

<option value="tsp">Teaspoon (tsp)</option>

<option value="tbsp">Tablespoon (tbsp)</option>

<option value="cup">Cup</option>

<option value="piece">Piece</option>

<option value="">As required</option>

</select>

</td>

<td><button type="button" onclick="removeIngredientRow(this)">Remove</button></td>

`;

document.getElementById('ingredientRows').appendChild(newRow);

selectedIngredient.value = ''; newIngredientInput.value = '';

} function removeIngredientRow(button) {

button.closest('tr').remove();

}

function updateIngredients() { if (isUpdatingIngredients) { console.log('Already updating ingredients, skipping'); return;

}

isUpdatingIngredients = true; console.log('Updating ingredients'); const categoryId = document.getElementById('category').value; const ingredientsDropdown = document.getElementById('ingredients');

ingredientsDropdown.innerHTML = '<option value="">Select an ingredient</option>'; if (categoryId) { console.log('Fetching ingredients for category:', categoryId); fetch(`/get-ingredients/${categoryId}/`)

.then(response => response.json())

.then(data => {

console.log('Ingredients received:', data.ingredients); data.ingredients.forEach(ingredient => { const option = document.createElement('option'); option.value = ingredient.id; option.textContent = ingredient.name; ingredientsDropdown.appendChild(option);

});

const newIngredientId = localStorage.getItem('newIngredientId'); if (newIngredientId) { console.log('Setting newly added ingredient:', newIngredientId); ingredientsDropdown.value = newIngredientId; localStorage.removeItem('newIngredientId');

}

})

.catch(error => console.error('Error fetching ingredients:', error))

.finally(() => { isUpdatingIngredients = false; console.log('Finished updating ingredients');

});

} else { console.log('No category selected, skipping ingredient update'); isUpdatingIngredients = false;

}

}

document.getElementById('addIngredientForm').addEventListener('submit', function(e) { e.preventDefault(); console.log('Submitting add ingredient form'); fetch('{% url "add\_ingredient" %}', { method: 'POST', body: new FormData(this), headers: {

'X-Requested-With': 'XMLHttpRequest',

'X-CSRFToken': this.querySelector('[name=csrfmiddlewaretoken]').value,

},

})

.then(response => { console.log('Response status:', response.status); if (!response.ok) { return response.json().then(data => { throw new Error(JSON.stringify(data));

});

}

return response.json();

})

.then(data => { console.log('Response data:', data); if (data.success) { alert('Ingredient added successfully!');

var modal = bootstrap.Modal.getInstance(document.getElementById('addIngredientModal')); modal.hide();

localStorage.setItem('newIngredientId', data.id); updateIngredients();

} else { alert('Error: ' + JSON.stringify(data.errors));

}

})

.catch(error => { console.error('Error:', error); alert('An error occurred: ' + error.message);

});

});

function logFormData(formData) { for (let [key, value] of formData.entries()) { console.log(key, value);

}}

// Add validation for the new fields document.querySelector('form').addEventListener('submit', function(event) { const servings = document.getElementById('servings'); const prepTime = document.getElementById('prep\_time'); const cookTime = document.getElementById('cook\_time'); if (servings.value < 1) { alert('Servings must be at least 1'); event.preventDefault(); return;}

if (prepTime.value < 0 || cookTime.value < 0) { alert('Prep time and cook time cannot be negative'); event.preventDefault(); return;

} });

function loadSubcategories() {

var categoryId = document.getElementById('category').value; var subcategorySelect = document.getElementById('subcategory');

// Clear existing options subcategorySelect.innerHTML = '<option value="">Select a subcategory</option>'; if (categoryId) { fetch(`/get-subcategories/${categoryId}/`)

.then(response => response.json())

.then(data => { data.subcategories.forEach(subcategory => { var option = document.createElement('option'); option.value = subcategory.subcategory\_id; option.textContent = subcategory.name; subcategorySelect.appendChild(option);

});

});

}}

</script>

</body>

</html>

#### 3. Editrecipe

{% load static %}

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Edit Recipe</title>

<link rel="stylesheet" href="{% static 'styles.css' %}">

<link rel="stylesheet"

href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css">

</head>

<body>

{% include 'navbar.html' %}

<div class="container">

<div class="form-container">

<div class="d-flex justify-content-between align-items-center mb-4">

<button onclick="history.back()" class="btn btn-secondary">Back</button>

<h1>Edit Recipe</h1>

<div style="width: 70px;"></div>

</div>

<form method="post" action="{% url 'edit\_recipe' recipe.recipe\_id %}" enctype="multipart/form-data">

{% csrf\_token %}

<div class="form-group">

<label for="recipename">Recipe Name:</label>

<input type="text" id="recipename" name="recipename" value="{{ recipe.recipename }}" required class="form-control">

</div>

<div class="form-group">

<label for="category">Category:</label>

<select id="category" name="category" required onchange="updateIngredients()" class="form-control">

{% for category in categories %}

<option value="{{ category.category\_id }}" {% if category.category\_id == recipe.category\_id %}selected{% endif %}>

{{ category.name }}

</option>

{% endfor %}

</select>

</div>

<div class="form-group">

<label for="ingredients">Select Ingredient:</label>

<select id="ingredients" onchange="addIngredientRow()" class="form-control">

<option value="">Select an ingredient</option>

</select>

</div>

<div id="ingredientDetailsContainer">

<table id="ingredientsTable" class="table table-striped">

<thead>

<tr>

<th>Ingredient</th>

<th>Quantity</th>

<th>Measurement</th>

<th>Action</th>

</tr>

</thead>

<tbody id="ingredientRows">

{% for recipe\_ingredient in recipe\_ingredients %}

<tr>

<td>

<input type="hidden" name="ingredient[]" value="{{ recipe\_ingredient.ingredient.ingredient\_id }}">

{{ recipe\_ingredient.ingredient.name }}

</td>

<td><input type="number" name="quantity[]" value="{{ recipe\_ingredient.quantity }}" step="0.01" required class="form-control"></td>

<td>

<select name="measurement[]" required class="form-control">

<option value="g" {% if recipe\_ingredient.measurement == 'g'

%}selected{% endif %}>Grams (g)</option>

<option value="kg" {% if recipe\_ingredient.measurement == 'kg' %}selected{% endif %}>Kilograms (kg)</option>

<option value="ml" {% if recipe\_ingredient.measurement == 'ml' %}selected{% endif %}>Milliliters (ml)</option>

<option value="l" {% if recipe\_ingredient.measurement == 'l' %}selected{% endif %}>Liters (l)</option>

<option value="tsp" {% if recipe\_ingredient.measurement == 'tsp' %}selected{% endif %}>Teaspoon (tsp)</option>

<option value="tbsp" {% if recipe\_ingredient.measurement == 'tbsp' %}selected{% endif %}>Tablespoon (tbsp)</option>

<option value="cup" {% if recipe\_ingredient.measurement == 'cup' %}selected{% endif %}>Cup</option>

<option value="piece" {% if recipe\_ingredient.measurement == 'piece' %}selected{% endif %}>Piece</option>

<option value="" {% if recipe\_ingredient.measurement == '' %}selected{% endif %}>As required</option>

</select>

</td>

<td><button type="button" onclick="removeIngredientRow(this)" class="btn btn-danger btnsm">Remove</button></td>

</tr>

{% endfor %}

</tbody>

</table>

</div>

<div class="form-group">

<label>Instructions:</label>

<div id="instructionsContainer" data-step-count="{{ instructions|length }}">

{% for instruction in instructions %}

<div class="instruction-item">

<label class="step-label">Step {{ forloop.counter }}:</label>

<textarea name="instructions[]" rows="3" required class="form-control">{{ instruction }}</textarea>

<button type="button" onclick="removeInstructionField(this)" class="btn btn-danger btn-sm mt-

2">Remove</button>

</div>

{% endfor %}

</div>

<div class="add-step-btn" onclick="addInstructionField()">Add Step</div> </div>

<div class="form-group">

<label for="tags">Tags:</label>

<input type="text" id="tags" name="tags" value="{{ recipe.tags }}" required class="formcontrol">

</div>

<div class="form-group">

<label for="image">Recipe Image:</label>

{% if recipe.image %}

<img src="{{ recipe.image.url }}" alt="{{ recipe.recipename }}" style="max-width: 200px;" class="img-thumbnail mb-2">

<br>

<div class="form-check">

<input type="checkbox" id="clear\_image" name="clear\_image" class="form-check-input">

<label for="clear\_image" class="form-check-label">Clear current image</label>

</div>

{% endif %}

<input type="file" id="image" name="image" accept="image/\*" class="form-controlfile mt-2">

</div>

<button type="submit" class="btn btn-primary btn-block">Update Recipe</button>

</form>

</div>

</div> <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/bootstrap.bundle.min.js"></script>

<script> function updateIngredients() {

const categorySelect = document.getElementById('category'); const ingredientSelect = document.getElementById('ingredients'); const selectedCategoryId = categorySelect.value;

// Clear existing options ingredientSelect.innerHTML = '<option value="">Select an ingredient</option>';

// Fetch ingredients based on selected category fetch(`/get-ingredients/${selectedCategoryId}/`)

.then(response => response.json())

.then(data => { data.ingredients.forEach(function(ingredient) { const option = document.createElement('option'); option.value = ingredient.id;

option.textContent = ingredient.name; ingredientSelect.appendChild(option);

});

})

.catch(error => console.error('Error:', error));

}

function addIngredientRow() { const selectedIngredient = document.getElementById('ingredients'); if (selectedIngredient.value) { const ingredientId = selectedIngredient.value; const ingredientName = selectedIngredient.options[selectedIngredient.selectedIndex].text;

constexistingIngredient=document.querySelector(`input[name^="ingredient[]"][value="${ingredi entId}"]`); if (existingIngredient) { alert('This ingredient is already in the recipe.'); selectedIngredient.value = ''; return;}

const newRow = document.createElement('tr'); newRow.innerHTML = `

<td>

<input type="hidden" name="ingredient[]" value="${ingredientId}">

${ingredientName}

</td>

<td><input type="number" name="quantity[]" step="0.01" required class="formcontrol"></td>

<td>

<select name="measurement[]" required class="form-control"> <option value="g">Grams (g)</option>

<option value="kg">Kilograms (kg)</option>

<option value="ml">Milliliters (ml)</option>

<option value="l">Liters (l)</option>

<option value="tsp">Teaspoon (tsp)</option>

<option value="tbsp">Tablespoon (tbsp)</option>

<option value="cup">Cup</option>

<option value="piece">Piece</option>

<option value="">As required</option>

</select>

</td>

<td><button type="button" onclick="removeIngredientRow(this)" class="btn btndanger btn-sm">Remove</button></td>

`;

document.getElementById('ingredientRows').appendChild(newRow); selectedIngredient.value = '';

}}

function removeIngredientRow(button) { button.closest('tr').remove();

}

function addInstructionField() { const container = document.getElementById('instructionsContainer'); const newInstruction = document.createElement('div'); newInstruction.classList.add('instruction-item'); newInstruction.innerHTML = `

<label class="step-label">Step ${container.children.length + 1}:</label>

<textarea name="instructions[]" rows="3" required class="form-control"></textarea> <button type="button" onclick="removeInstructionField(this)" class="btn btn-danger btnsm mt-2">Remove</button>

`;

container.appendChild(newInstruction);

}

function removeInstructionField(button) { button.closest('.instruction-item').remove();

// Update step numbers const container = document.getElementById('instructionsContainer'); container.querySelectorAll('.instruction-item').forEach((item, index) => { item.querySelector('.step-label').textContent = `Step ${index + 1}:`; });

}

// Initial call to populate ingredients based on the default selected category updateIngredients();

</script>

</body>

</html>

**Views.py**

#### #login

def login(request): if request.method == 'POST':

email = request.POST.get('Email') password = request.POST.get('password') if not email or not password:

return render(request, 'login.html', {'error': 'Email and password are required'}) try:

user = CustomUser.objects.get(email=email)

# Check if it's the admin user if user.email == "admin@gmail.com" and user.password == "Admin@123": request.session['id'] = user.id request.session['is\_admin'] = True request.session['username'] = user.name request.session['email'] = user.email return redirect('admin\_dashboard') else:

# For regular users, directly compare plain text passwords if user.password == password: request.session['id'] = user.id

request.session['is\_admin'] = False # Assuming regular users are not admins request.session['username'] = user.name request.session['email'] = user.email return redirect('homepage') else:

return render(request, 'login.html', {'error': 'Incorrect password'}) except CustomUser.DoesNotExist:

return render(request, 'login.html', {'error': 'Email does not exist'})

return render(request, 'login.html')

user\_pins = {}

#### #addrecipe

@transaction.atomic def addrecipe(request): if request.method == 'POST': try:

# Extract basic recipe information recipename = request.POST.get('recipename') category\_id = request.POST.get('category') subcategory\_id = request.POST.get('subcategory') tags = request.POST.get('tags') image = request.FILES.get('image') servings = request.POST.get('servings') prep\_time = request.POST.get('prep\_time') cook\_time = request.POST.get('cook\_time')

# Collect all instructions instructions = [] for key, value in request.POST.items(): if key.startswith('instructions\_') and value.strip():

instructions.append(value.strip()) instructions = '\n'.join(instructions)

# Validate fields if not all([recipename, category\_id, tags, instructions, servings, prep\_time, cook\_time]):

raise ValueError('All fields except image are required!')

# Create Recipe

recipe = Recipe.objects.create( recipename=recipename, category\_id=category\_id, subcategory\_id=subcategory\_id, tags=tags, instructions=instructions, image=image,

user\_id=request.session.get('id'), # Assuming user ID is stored in session servings=servings,

prep\_time=prep\_time, cook\_time=cook\_time

)

# Process ingredients ingredient\_count = 1 while True:

ingredient\_id = request.POST.get(f'ingredient\_{ingredient\_count}') ingredient\_name = request.POST.get(f'ingredient\_name\_{ingredient\_count}') quantity = request.POST.get(f'quantity\_{ingredient\_count}') measurement = request.POST.get(f'measurement\_{ingredient\_count}') if not all([ingredient\_id, quantity, measurement]): break if ingredient\_id.startswith('new\_'): # Create a new ingredient new\_ingredient = Ingredient.objects.create(name=ingredient\_name) ingredient\_id = new\_ingredient.ingredient\_id RecipeIngredient.objects.create( recipe=recipe, ingredient\_id=ingredient\_id, quantity=quantity, measurement=measurement

)

ingredient\_count += 1 messages.success(request, 'Recipe added successfully!') return redirect('recipe') except ValueError as ve: messages.error(request, str(ve)) # Catch specific validation error except Exception as e:

messages.error(request, f'Error adding recipe: {str(e)}') # General error

# Fetch categories and ingredients for the form categories = Category.objects.filter(status=True) ingredients = Ingredient.objects.all()

return render(request, 'addrecipe.html', {'categories': categories, 'ingredients': ingredients})

#### #usereditrecipe

def usereditrecipe(request, recipe\_id):

recipe = get\_object\_or\_404(Recipe, recipe\_id=recipe\_id) if request.method == 'POST': try: with transaction.atomic():

# Update basic recipe information recipe.recipename = request.POST.get('recipename') recipe.category\_id = request.POST.get('category') recipe.tags = request.POST.get('tags') if 'image' in request.FILES:

recipe.image = request.FILES['image'] elif 'clear\_image' in request.POST:

recipe.image = None

# Handle ingredients

RecipeIngredient.objects.filter(recipe=recipe).delete() ingredient\_ids = request.POST.getlist('ingredient[]') quantities = request.POST.getlist('quantity[]') measurements = request.POST.getlist('measurement[]') for i in range(len(ingredient\_ids)): ingredient = get\_object\_or\_404(Ingredient, ingredient\_id=ingredient\_ids[i]) RecipeIngredient.objects.create( recipe=recipe, ingredient=ingredient, quantity=quantities[i], measurement=measurements[i]

)

# Handle instructions instructions = request.POST.getlist('instructions[]') recipe.instructions = '\n'.join(filter(None, instructions)) # Join non-empty instructions recipe.save() messages.success(request, "Recipe updated successfully!")

return redirect('recipe\_detail', recipe\_id=recipe.recipe\_id) except Exception as e:

logger.error(f"Error updating recipe {recipe\_id}: {str(e)}")

messages.error(request, f"An error occurred while updating the recipe: {str(e)}") else:

form = RecipeForm(instance=recipe) context = {

'recipe': recipe,

'form': form,

'recipe\_ingredients': recipe.recipe\_ingredients.all(),

'all\_ingredients': Ingredient.objects.all(),

'categories': Category.objects.all(),

'instructions': recipe.instructions.split('\n') if recipe.instructions else [],

}

return render(request, 'usereditrecipe.html', context)

### 9.2 Screen Shots

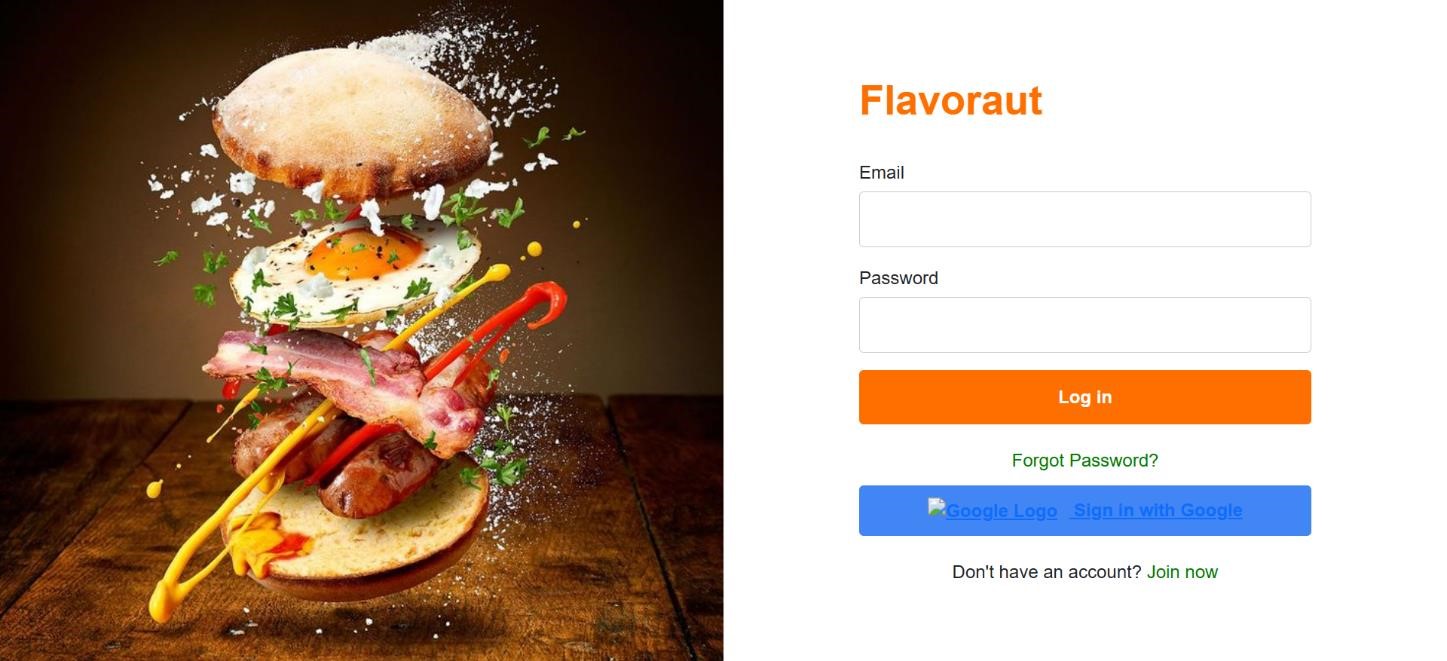
#### 9.2.1 Login Page

Figure

9.2.1

:

LoginPage



**9.2.2**

**Signup Page**

Figure

9.2.2

:

Signup Page



#### 9.2.3 Homepage

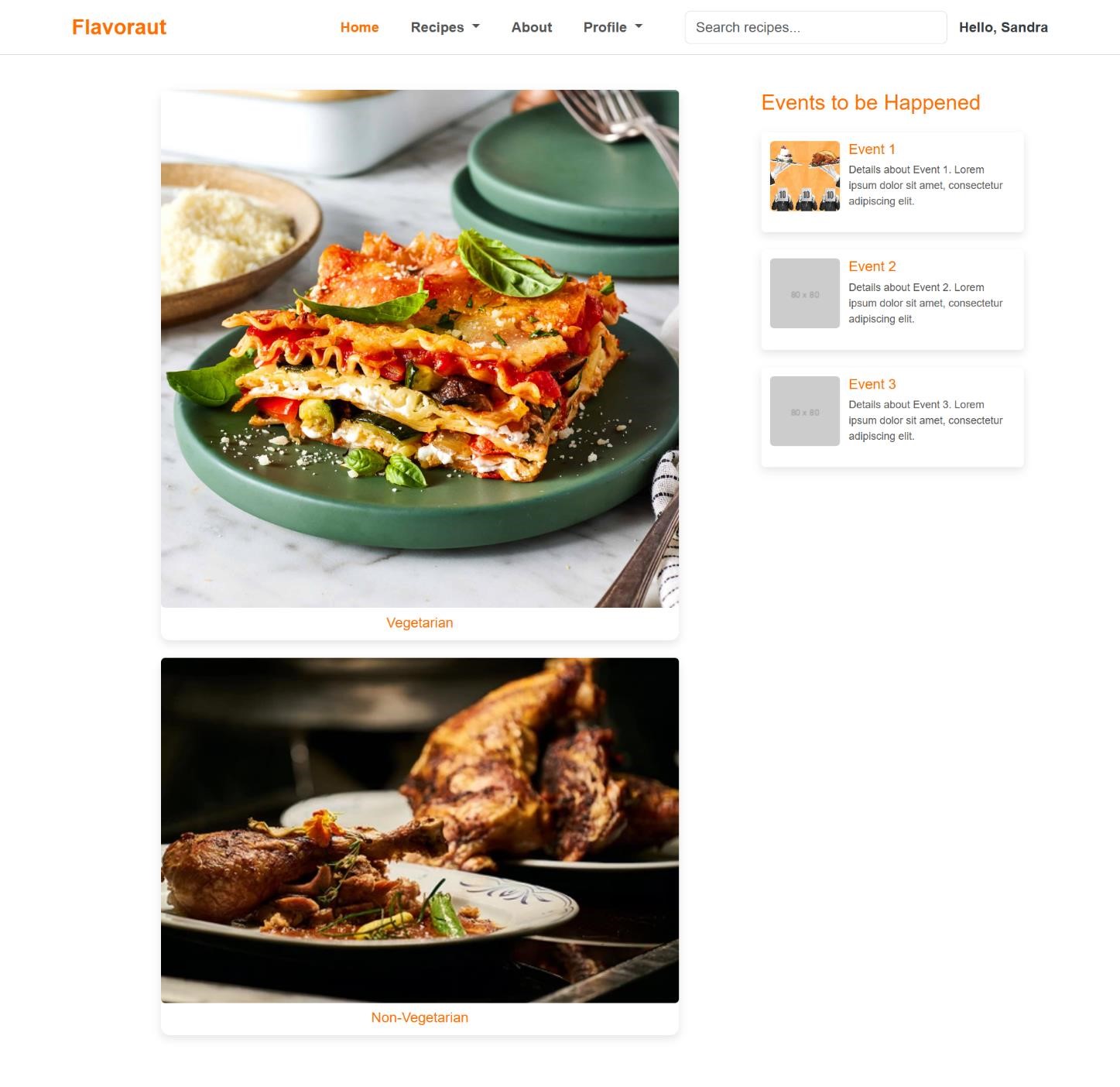
Figure

9.2.3

:

Home

Page



**9.2.4 ViewRecipe Page:**

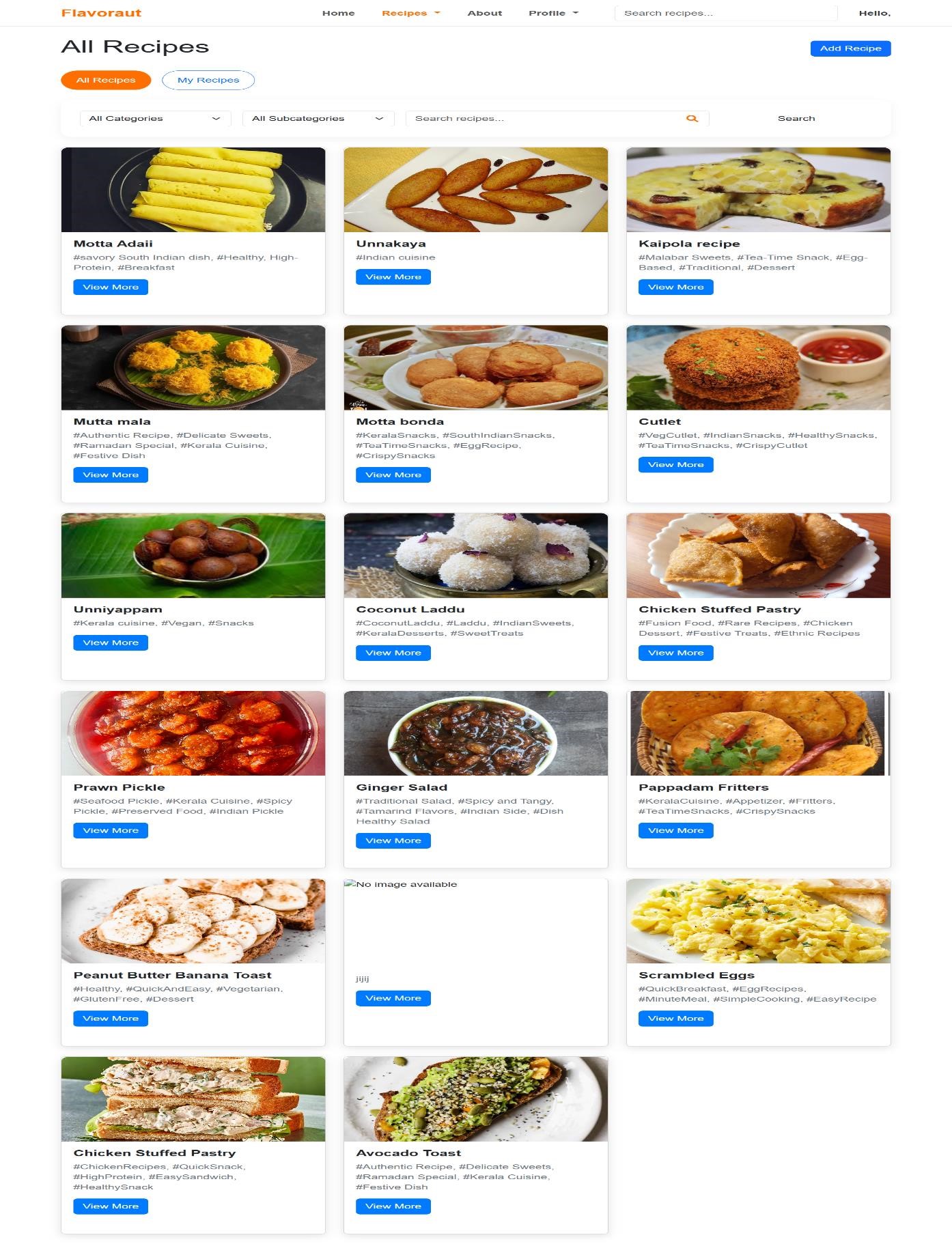


Figure 9.2.4: View Recipe Page

**9.2.5 AddRecipe Page:**

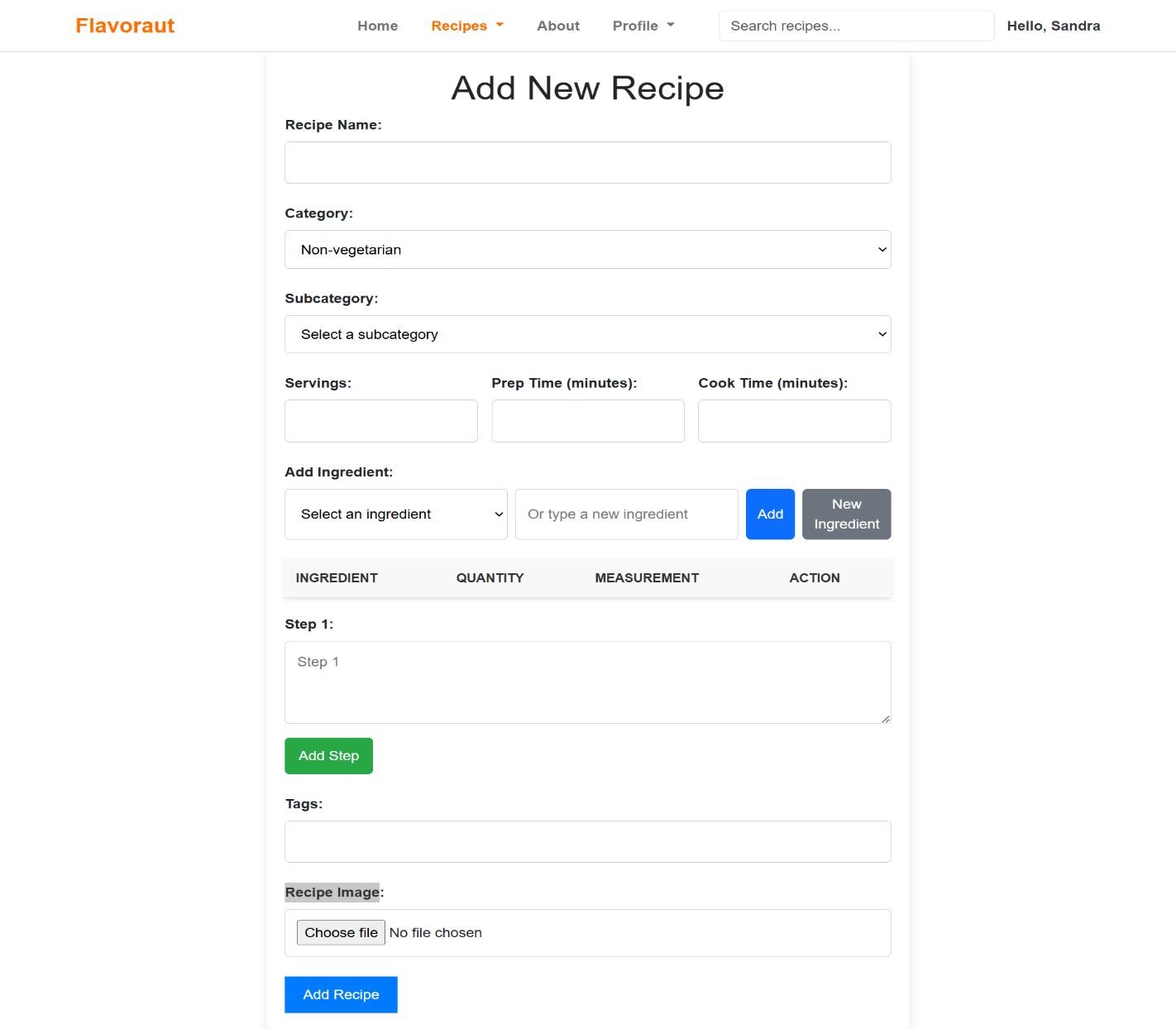


Figure 9.2.5: Recipe Adding Page

**9.2.6 RecipeDetails**

Figure

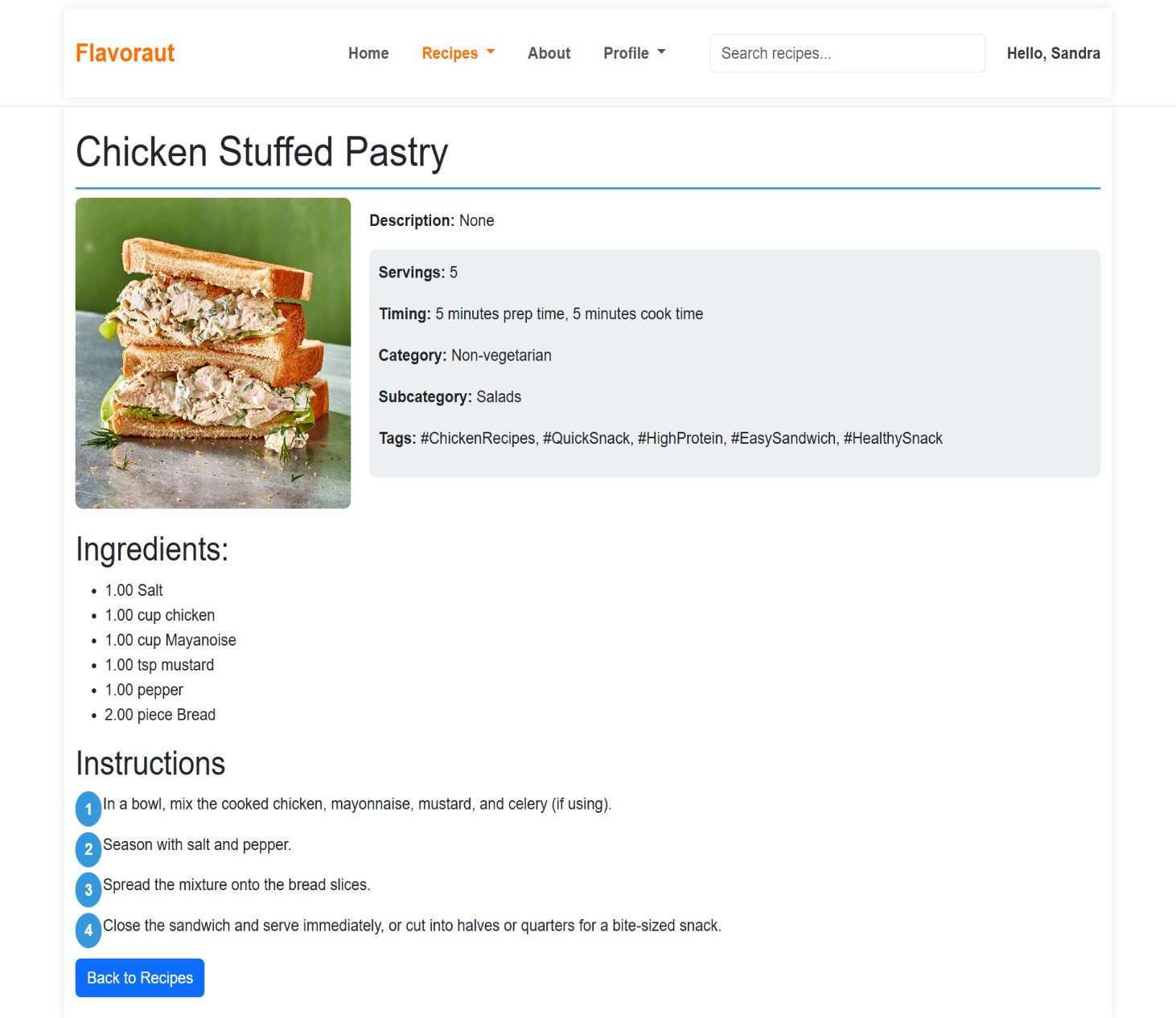
9.2.6

:

Recipe Adde

d

Page



**Amal Jyothi College of Engineering, Kanjirappally Department of Computer Applications**