RECIPE NEST

A PROJECT REPORT

For
Project (KCA451)
Session (2023-24)

Submitted by

Pratush Singh

(2200290140115)

Submitted in partial fulfilment of the

Requirements for the Degree of

MASTER OF COMPUTER APPLICATION

Under the Supervision of

Ms.Komal Salgotra
Assistant Professor

Submitted to

DEPARTMENT OF COMPUTER APPLICATIONS

KIET Group of Institutions, Ghaziabad Uttar Pradesh-201206

(May-2024)

DECLARATION

I hereby declare that the work presented in this report entitled “Recipe Nest”, was carried out by me. I have not submitted the matter embodied in this report for the award of any other degree or diploma of any other University or Institute. I have given due credit to the original authors/sources for all the words, ideas, diagrams, graphics, computer programs, experiments, results, that are not my original contribution. I have used quotation marks to identify verbatim sentences and given credit to the original authors/sources. I affirm that no portion of my work is plagiarized, and the experiments and results reported in the report are not manipulated. In the event of a complaint of plagiarism and the manipulation of the experiments and results, I shall be fully responsible and answerable.

Name: Pratush Singh
(2200290140115)

(Candidate Signature)

CERTIFICATE

Certified that **Pratush Singh** have carried out the project work having “Recipe Nest" **(Major Project-KCA451)** for **Master of Computer Application** from Dr. A.P.J. Abdul Kalam Technical University (AKTU), Lucknow under my supervision. The project report embodies original work, and studies are carried out by the student himself / herself and the contents of the project report do not form the basis for the award of any other degree to the candidate or to anybody else from this or any other University/Institution.

Date:

Pratush Singh (2200290140115)

This is to certify that the above statement made by the candidate is correct to the best of my knowledge.

Date:

Ms.Komal Salgotra	**Dr. Arun Tripathi**
Assistant Professor	**Head**
Department of Computer Applications	**Department of Computer Applications**
KIET Group of Institutions, Ghaziabad	**KIET Group of Institutions, Ghaziabad**

ABSTRACT

Imagine a kitchen companion that transforms leftover ingredients into culinary masterpieces. Our recipe recommender does just that, empowering home cooks of all levels to discover new flavours and minimize food waste. Simply input your available ingredients, and our intelligent system scours a vast database to present the perfect recipe, tailored to your pantry and preferences.

But this isn't just a one-way street. Register and unleash your inner chef! Create, edit, and personalize your own recipes, adding secret ingredients and treasured family traditions. Share your culinary creations with the community, inspiring others and expanding your personal recipe library.

Our intuitive interface makes browsing and managing recipes a breeze. Visual recipe cards showcase key ingredients, preparation steps, and user ratings, while text search, voice commands, and even image recognition make adding ingredients to your list a snap.

More than just a recipe finder, we're fostering a collaborative community where food lovers can connect, share their passion, and learn from each other. Whether you're a kitchen novice seeking inspiration or a seasoned chef adding a new dish to your repertoire, join us and unlock a world of endless culinary possibilities.

So, ditch the meal-time rut and embrace the joy of culinary creativity. Let our ingredient-driven recommender be your guide, and turn your pantry into a playground of flavour!

ACKNOWLEDGEMENTS

Success in life is never attained single-handedly. My deepest gratitude goes to my thesis supervisor, **Ms.Komal Salgotra** for his guidance, help and encouragement throughout my research work. Their enlightening ideas, comments, and suggestions. Words are not enough to express my gratitude to **Dr. Arun Kumar Tripathi**, Professor and Head, Department of Computer Applications, for his insightful comments and administrative help at various occasions.

Fortunately, I have many understanding friends, who have helped me a lot on many critical conditions.

Finally, my sincere thanks go to my family members and all those who have directly and indirectly provided me with moral support and other kind of help. Without their support, completion of this work would not have been possible in time. They keep my life filled with enjoyment and happiness.

Pratush Singh

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

1.1 OVERVIEW

In the bustling realm of culinary exploration, the intersection of convenience and creativity has found a new ally in Pantry2Plate. As the pace of modern life quickens and the demand for home-cooked meals rises, the need for an intuitive and efficient recipe search engine becomes paramount. Enter Pantry2Plate, a culinary companion designed to simplify the cooking experience by leveraging the ingredients readily available in your home.

Pantry2Plate is not just another recipe search engine; it's a culinary aide with a clear mission—to empower individuals in their kitchens with an easy-to-use search tool that transforms their existing pantry staples into delightful, wholesome meals. This innovative platform is crafted for those who cherish the joy of home cooking, providing a seamless bridge between the ingredients at hand and the vast world of culinary possibilities.

At the heart of Pantry2Plate lies the My Pantry feature, a user-friendly input system that invites individuals to showcase the contents of their home kitchen. Users simply list the ingredients they have readily available, and with a few clicks, Pantry2Plate's powerful algorithms curate a diverse array of recipes tailored to match those ingredients. This intelligent matching system goes beyond basic searches, breaking down the recipes into categories to suit varied tastes and preferences.

The culinary landscape is diverse, and so is Pantry2Plate's repertoire. Whether you're a seasoned chef or a novice in the kitchen, our platform strives to make the art of cooking accessible to all. No longer will the question of "What's for dinner?" be met with uncertainty; Pantry2Plate ensures that the answer is just a few clicks away.

Join us on a journey where your pantry transforms into a canvas for culinary artistry. Pantry2Plate is not merely a recipe search engine—it's a gateway to a world of flavors, a companion in your culinary escapades, and a testament to the belief that home-cooked meals can be both delightful and effortlessly accessible. Welcome to Pantry2Plate, where your ingredients are the key, and the possibilities are endless.

The motivation to develop this “Recipe search engine using Yummy API” Web Application comes from my urge to learn technologies like ASP.NET with MVC, jQuery, AJAX and RESTful web services. A Web service is a service offered by an electronic device to another electronic device, communicating with each other via the World Wide Web. REST-compliant (Representation of State Transfer) Web services is one of the two identified major classes of Web services, in which the primary purpose of the service is to manipulate representations of Web resources using a uniform set of stateless operations.

Also, my interest to learn the implementation of OAuth to log users into this application. OAuth is an open standard for authorization, commonly used as a way for Internet users to log into third party websites using their Microsoft, Google, Facebook etc. accounts without exposing their password. In this web application web pages are created using ASP.NET 4.6 implemented in MVC 5 architecture, jQuery is used for client-side scripting and also for creating rich and interactive user interface.

Bootstrap 2.0 is used for creating responsive user interface and styling the web pages. This web application will communicate with other web API using RESTful web services to pull data from it. C# and JavaScript languages are used for building the business logic of the application. SQL server 2014 is used for database designing and Entity Framework 6 which is an object relational model (ORM) tool is used to connect the application with database.

Advanced technologies are used in building this web application thus making this application more effective. MVC 5 architecture is implemented i.e. model, view and controller layers are separated and independent of each other which increases the application performance in terms of speed, and also changes can be made in any layer without disturbing other layers.

Another motivation is to develop a powerful recipe search engine that will aptly search for the results the users are intending and display the obtained Also provide required filters to filter out the search results and sorting options to sort the results according to users’ interest. Thus, using the latest technologies to develop a powerful search engine and get familiar with the web services are the main motivation for the project.

1.2 OBJECTIVE

Eradicating Food Waste: Dish craft champions the fight against food waste. By understanding your pantry's potential, it crafts recipes that utilize every morsel, saving you money and reducing environmental impact. No more wilting vegetables or forgotten leftovers, just a celebration of culinary resourcefulness.

Unleashing Culinary Creativity: Step outside your comfort zone and embark on a global culinary adventure, all fuelled by what you already have. Dish craft's intelligent algorithms unlock a universe of diverse recipes, from Thai stir-fries to Italian pastas, all waiting to be explored with the ingredients at your fingertips. Break free from the shackles of predictable meals and embrace the thrill of culinary discovery.

Personalizing Your Kitchen Experience: Dish craft is more than just a recipe finder; it's a digital cookbook tailored to your palate. Register and build your own culinary haven, filled with cherished family recipes, personal modifications, and even your own innovative creations. Experiment with flavours, document your culinary journey, and create a personalized library that reflects your unique culinary identity.

Fostering a Vibrant Community: Connect with fellow foodies, share your culinary triumphs and disasters, and discover hidden gems through Dish craft’s intuitive social features. Leave reviews and comments on recipes, sparking conversations and enriching the user experience. Build a community where food unites, inspires, and fosters a culture of shared culinary passion.

The different functionalities available for the users in this application are as following. Users can search for their favourite dishes. The search results contain information about ingredients required, total time needed for cooking, user’s rating and cooking directions. Basic search filters are provided to filter out the search results like Breakfast, Lunch and Dinner recipes.

The order of displayed results can be sorted according to ratings, total time required to prepare the dish. User can create an account and build their own favourite recipe collection by liking the recipes displayed. The liked recipes are stored into user’s account and user can view, add and delete those recipes anytime from his recipe collection. Users can use their social networking platform Facebook account credentials to log into this application or create a new account in this application.

The primary objective of the Online Recipe Finder application is to provide users with a seamless and intuitive platform to discover, save, and share a wide variety of recipes. This application aims to enhance the culinary experience by offering features that cater to diverse dietary needs, preferences, and skill levels. Key objectives include:

1. **User-Friendly Interface:** Design a clean, intuitive, and easy-to-navigate interface that allows users of all ages and technical abilities to quickly find and use recipes.
2. **Comprehensive Recipe Database:** Build and maintain an extensive database of recipes covering various cuisines, dietary restrictions (e.g., vegan, gluten-free), and meal types (e.g., breakfast, lunch, dinner, snacks).

3. **Advanced Search and Filtering:** Implement robust search and filtering options to help users easily find recipes based on ingredients, cooking time, difficulty level, cuisine type, and dietary preferences.
4. **Personalization and Recommendations:** Develop personalized recommendation algorithms that suggest recipes based on user preferences, past searches, and saved recipes.
5. **Ingredient Management:** Offer features for ingredient management, such as creating shopping lists, tracking pantry items, and suggesting recipes based on available ingredients.
6. **User Contributions and Community Engagement:** Enable users to contribute their own recipes, rate and review recipes, and engage with other users through comments and forums, fostering a vibrant and supportive community.
7. **Step-by-Step Instructions and Multimedia:** Provide clear, step-by-step cooking instructions accompanied by photos, videos, and tips to assist users in preparing recipes successfully.
8. **Nutritional Information:** Include detailed nutritional information for each recipe to help users make informed choices about their meals.
9. **Cross-Platform Accessibility:** Ensure the application is accessible on multiple devices, including web browsers, smartphones, and tablets, to cater to users' varying needs and preferences.
10. **Regular Updates and Seasonal Content:** Keep the content fresh and relevant by regularly updating the recipe database and featuring seasonal recipes and trending culinary topics.

1.3 PROJECT FEATURE

Ingredient-driven Search: Tell Kitchen Maestro what you have, be it leftover vegetables, a can of beans, or a lonely chicken breast, and watch it whip up a symphony of flavors, prioritizing recipes that best utilize your existing ingredients. Minimize waste, maximize flavor, and discover hidden potential in your pantry.

Smart Recipe Recommendations: Forget wading through irrelevant options. Kitchen Maestro's intelligent algorithms consider your ingredients, dietary restrictions, preferences, and even cooking skills to present a curated selection of perfectly matched recipes. No more recipe ruts, just endless possibilities waiting to be explored.

Detailed Recipe Cards: Get a clear picture of each dish, with step-by-step instructions, cooking times, nutritional information, and user ratings all readily available. Stunning visuals, intuitive menus, and interactive features make following recipes a breeze, even for novice cooks.

Personalization Haven: Register and build your own culinary sanctuary. Save your favorite recipes, add personal notes and modifications, and even create your own dishes to share with the world. Kitchen Maestro becomes your digital cookbook, evolving with your culinary journey and reflecting your unique palate.

Social Features: Connect with fellow food enthusiasts, share your culinary creations, discover hidden gems through ratings and reviews, and engage in lively discussions about all thing's food. The Kitchen Maestro community fosters a vibrant atmosphere were inspiration and knowledge flow freely, enriching the user experience for everyone.

Beyond the Core: Imagine seamless integrations with grocery delivery services, where the app can suggest recipes based on both your pantry and available deliveries. Dietary filters cater to specific needs and preferences, making healthy cooking effortless. Voice recognition capabilities further streamline the experience, allowing users to simply speak their ingredients and receive instant recipe recommendations. Gamification elements, like points for trying new recipes or sharing creations, keep users engaged and motivated.

Project Features for Online Recipe Finder Application:

1. **User Registration and Profiles:**
    - Secure user registration and login.
    - Profile creation with personal preferences, dietary restrictions, and favorite cuisines.
    - Option to follow other users and see their activity.
2. **Recipe Database:**
    - Extensive collection of recipes with detailed descriptions, ingredients, and cooking instructions.
    - High-quality images and videos for each recipe.
    - Categorization by cuisine, meal type, dietary requirements, and difficulty level.
3. **Advanced Search and Filtering:**
    - Search recipes by ingredient, cuisine, cooking time, difficulty level, and dietary needs.
    - Filters for meal types (e.g., breakfast, lunch, dinner, snacks).
    - Keyword-based search functionality.
4. **Personalized Recommendations:**
    - Algorithm-driven personalized recipe suggestions based on user preferences, past searches, and saved recipes.
    - Weekly or monthly curated recipe collections.

5. **Ingredient Management:**
    - Feature to create and manage shopping lists.
    - Pantry tracker to keep track of available ingredients.
    - Recipe suggestions based on the ingredient’s users have on hand.
6. **Interactive Cooking Guides:**
    - Step-by-step cooking instructions with multimedia support (photos, videos, and audio guides).
    - Tips and tricks for cooking techniques and ingredient substitutions.
7. **Nutritional Information:**
    - Detailed nutritional information for each recipe (calories, macronutrients, vitamins, and minerals).
    - Filters to search for recipes based on nutritional content.
8. **User-Generated Content and Community Features:**
    - Allow users to submit their own recipes.
    - Rating and review system for recipes.
    - Comment sections for users to discuss recipes and share tips.
    - Forums and groups for specific dietary preferences and cooking interests.
9. **Social Sharing and Integration:**
    - Share recipes via social media platforms (Facebook, Instagram, Twitter, etc.).
    - Integration with social media for login and sharing.
    - Option to save and share shopping lists and meal plans.
10. **Cross-Platform Accessibility:**
    - Responsive web design for use on desktops, tablets, and smartphones.
    - Native mobile applications for iOS and Android.
    - Offline access to saved recipes and shopping lists.
11. **Notifications and Alerts:**
    - Push notifications for new recipes, updates from followed users, and personalized recommendations.
    - Alerts for expiring pantry items and reminders for meal planning.
12. **Seasonal and Trending Content:**
    - Regularly updated content with seasonal recipes and holiday-themed dishes.
    - Highlight trending recipes and popular cooking challenges.

13. **Monetization Features:**

- Subscription model for premium features (e.g., ad-free experience, exclusive recipes).
- In-app purchases for premium content or special recipe collections.
- Advertising space for relevant brands and products.

14. **Analytics and User Insights:**

- Analytics dashboard for tracking user engagement and popular recipes.
- User feedback mechanisms for continuous improvement.

1.3 Problems with existing systems

The purpose of creating this Web Application is to outcast the discrepancies in hundreds of such existing systems on the World Wide Web. The various existing systems for recipe search and their disadvantages are discussed below.

Internet search engines:

One of the existing systems is Internet search engines like google, bing etc. If user wants to search for their favourited recipe, user will go to web, say google, and search for his recipe in his mind. The search results may redirect user to any other food blog website or any video casting 2Zsite. But in that website, all the information required like list of ingredients, cooking instructions, nutritional data and several other information may or may not be available at one place. Moreover, you have to go through too many results to choose which result suits you best. This system involves more time and analysis which is a tedious task thus not ideal for many users.

Food Blogs:

Another existing system is Food blogs and channels on video sharing sites like YouTube. There are many food blogs which provide lot of information about recipes but the disadvantages with this kind of blogs is most of the times they don't support multi cuisine recipes. Most often they belong to one region which is a disadvantage most of the times. Personally, I follow few YouTube channels and one among those is 'VahChef' which contains interesting videos on how to cook various Indian dishes but if I want to cook some Mexican food I have to google it and follow some other blogs or video hosting sites.

In this case of existing system, food blogs are mostly confined to one or two types of cuisine which is a disadvantage for users looking for multi cuisine. For better understanding of the disadvantages of existing system consider a regular scenario, I went for shopping groceries and I'm planning to cook some new dish and I don't know what ingredients are required and of what quantity are required.

So I have to search on the web and I might be provided with numerous options throwing me into dilemma which one to rely and if it is video I have to manually jot down the ingredients or remember those list of ingredients which is not at all an ideal thing to do. These all are big drawbacks of existing system and this application is capable of addressing all these issues in an efficient way providing an ideal platform to search for any food recipe and get all the required information such as ingredients, nutritional data, cooking instructions. This application supports all multi cuisine thus serving wide range of people.

1.4 Intended Audience

The intended audiences for this website are the users who rely on web for cooking dishes and also who search for new recipe ideas on web. This web application acts as bridge connecting users with recipe ideas, ingredient lists, and cooking instructions.

CHAPTER 2

LITERATURE REVIEW

2.1 PANTRY2PLATE

Existing recipe search engines have paved the way for P2P. Platforms like Yummly and Allrecipes emphasize ingredient-based searches, but P2P aims to distinguish itself by prioritizing simplicity and accuracy in matching ingredients with recipes. Research suggests that user-friendly interfaces significantly impact user engagement in such platforms.

The surge in internet accessibility and digital transformation has revolutionized how people find and use recipes. Online recipe finders have become an integral part of cooking, allowing users to discover, save, and share recipes with ease. This literature review explores existing research and developments in the domain of online recipe finders, focusing on user needs, technological advancements, and current trends.

2.2 PERSONALIZED RECIPE RECOMMENDATIONS

Studies on personalized recommendation systems reveal the importance of algorithms in understanding user preferences. P2P's commitment to delivering the best recipes tailored to users' ingredients and dietary preferences aligns with the growing trend of personalization in online platforms, enhancing user satisfaction.

2.3 USER-GENERATED CONTENT IN CULINARY PLATFORMS

Examining successful platforms like Food Network's community and Epicurious, it becomes evident that user-generated content enhances the overall experience. By allowing users to contribute recipes, P2P taps into the collaborative spirit of cooking communities, fostering a sense of connection among users with shared culinary interests.

2.4 AUTHENTICATION AND SECURITY IN RECIPE PLATFORMS

Platforms that encourage user contributions necessitate robust authentication and security measures. Insights from research on secure user authentication and data protection underline the importance of P2P's decision to implement separate login credentials for recipe contributors, ensuring a secure and trustworthy environment.

2.5 TECHNOLOGICAL INNOVATIONS IN RECIPE DISCOVERY

Technological advancements, such as dynamic ingredient lists and real-time updates, have transformed the landscape of recipe discovery. P2P's utilization of a dynamic list for ingredient selection aligns with research advocating for intuitive and responsive interfaces in cooking-related applications.

2.6 CHALLENGES AND OPPORTUNITIES IN RECIPE PLATFORMS

By reviewing challenges faced by existing platforms, P2P can proactively address potential issues. The literature emphasizes the importance of responsiveness, scalability, and effective filtering mechanisms in recipe search engines, presenting opportunities for P2P to excel.

Introduction

The development of online recipe finder applications has transformed how people access and interact with culinary information. This literature review explores the existing research and developments in the field, focusing on user needs, technological advancements, personalization, community features, and health considerations.

User Needs and Behaviour

Research indicates that users seek convenience, variety, and personalization in online recipe platforms. A study by Verlegh et al. (2015) found that ease of use, clarity of instructions, and ingredient availability are critical factors influencing user satisfaction with recipe websites. Additionally, the integration of user-generated content, such as reviews and ratings, significantly impacts user engagement and trust (Moran & Muzellec, 2017).

Technological Advancements

Advancements in technology have significantly enhanced the functionality of online recipe finders. Natural language processing (NLP) and machine learning algorithms are increasingly used to improve search and recommendation systems. For instance, Trattner and Elsweiler (2017) highlighted how machine learning models can predict user preferences and suggest recipes that match their tastes and dietary restrictions. Furthermore, image recognition technology enables users to upload photos of ingredients to find corresponding recipes (Chen et al., 2018).

Personalization and Recommendations

Personalization is a critical feature of modern recipe finder applications. Collaborative filtering and content-based filtering are common techniques used to provide personalized recipe recommendations. A study by Said and Bellogín (2014) demonstrated that hybrid recommendation systems, which combine multiple filtering techniques, offer superior performance in suggesting relevant recipes. Additionally, context-aware recommendations, which consider factors such as time of day, season, and user activity, further enhance user experience (Hariri, Mobasher, & Burke, 2013).

Community and Social Features

The inclusion of community and social features in recipe applications fosters user engagement and knowledge sharing. Social media integration allows users to share their culinary creations and discover new recipes through their networks. According to Simons, Weinberger, and Brooks (2016), community-driven platforms benefit from network effects, where increased user participation leads to richer content and enhanced user satisfaction. User-generated content, such as recipe submissions and cooking tips, adds diversity and authenticity to the platform (Weber & Mitchell, 2019).

Health and Nutritional Information

Health-conscious users increasingly demand detailed nutritional information and support for dietary restrictions. Providing accurate and comprehensive nutritional data helps users make informed decisions about their meals. Research by Fallaize et al. (2019) emphasizes the importance of nutritional transparency and suggests that integrating tools for meal planning and calorie tracking can enhance the value of recipe applications. Additionally, features that cater to specific dietary needs, such as vegan or gluten-free options, broaden the application's appeal (Müller et al., 2017).

Conclusion

The literature underscores the importance of user-centric design, technological innovation, personalization, community engagement, and health considerations in the development of online recipe finder applications. By addressing these key areas, developers can create platforms that not only meet user needs but also promote culinary exploration and healthy eating habits. Future research should explore the integration of emerging technologies, such as augmented reality (AR) and virtual reality (VR), to further enrich the user experience.

CHEPTER 3

FEASIBILITY STUDY

After doing the project, study and analysing all the existing or required functionalities of the system, the next task is to do the feasibility study for the project. All projects are feasible-given unlimited resources and in finite time. Feasibility study includes consideration of all the possible ways to provide a solution to the given problem. The proposed solution should satisfy all the user requirements and should be flexible enough so that future changes can be easily done based on the future upcoming requirements. There are three parts in feasibility study

a) Technical Feasibility
b) Economic Feasibility
c) Operational Feasibility
d) Scheduling Feasibility
e) Security and Privacy Feasibility
f) User Engagement and Feedback

3.1 TECHNICAL FEASIBILITY

The technical feasibility of P2P is rooted in its ability to implement and maintain a sophisticated recipe search engine. The utilization of dynamic lists, real-time updates, and user authentication systems suggests a sound technical foundation. The study evaluates the availability of required technologies, the expertise of the development team, and the feasibility of incorporating innovative features.

The term "technical feasibility" refers to the assessment of the practicality and technical requirements of a proposed project or system. This analysis is crucial during the early stages of project planning to determine whether the project is viable from a technological standpoint. Here's a breakdown of the key aspects involved in evaluating technical feasibility:

1. Technology Assessment

- **Current Technology:** Evaluate whether the current technology is capable of meeting the project's requirements. This involves assessing existing hardware, software, and network capabilities.
- **Emerging Technology:** Consider the potential of new and emerging technologies to fulfil the project needs. This includes evaluating the maturity, stability, and adoption rates of these technologies.

2. Technical Skills and Expertise

- **Availability of Expertise:** Assess whether the organization has the necessary technical skills and expertise to implement and maintain the project. If not, determine whether these skills can be acquired through training or hiring.
- **Experience:** Evaluate the team's past experience with similar technologies or projects to ensure they can handle potential challenges.

3. System Requirements

- **Hardware Requirements:** Identify the hardware necessary to support the project, including servers, workstations, and other equipment.
- **Software Requirements:** Determine the software needs, including operating systems, applications, databases, and development tools.
- **Integration Requirements:** Assess how the new system will integrate with existing systems and whether any modifications are needed.

4. Scalability and Flexibility

- **Scalability:** Evaluate whether the proposed solution can scale to meet future demands. This includes considering the potential for increased data volumes, user numbers, and transaction rates.
- **Flexibility:** Assess the system's ability to adapt to changing requirements and technologies over time.

5. Security and Compliance

- **Security Requirements:** Identify the security measures necessary to protect the system from threats and vulnerabilities. This includes data encryption, access controls, and secure coding practices.
- **Regulatory Compliance:** Ensure the system complies with relevant laws, regulations, and industry standards. This may involve data privacy regulations, industry-specific standards, and organizational policies.

6. Cost and Time Estimates

- **Cost Estimation:** Estimate the costs associated with the technology, including hardware, software, development, and ongoing maintenance.
- **Time Estimation:** Assess the time required to develop, implement, and deploy the system. This includes considering potential delays and risks that may impact the timeline.

7. Risk Assessment

- **Identification of Risks:** Identify potential technical risks that could impact the project's success. This includes risks related to technology obsolescence, integration challenges, and dependency on third-party vendors.
- **Mitigation Strategies:** Develop strategies to mitigate identified risks, such as contingency planning, alternative solutions, and phased implementation.

Steps to Conduct a Technical Feasibility Study

1. **Define Objectives:** Clearly outline the objectives and scope of the project.
2. **Gather Requirements:** Collect detailed requirements from stakeholders.
3. **Analyse Existing Systems:** Review the current systems and technology in use.
4. **Identify Technology Options:** Explore various technology solutions that could meet the project requirements.
5. **Evaluate Options:** Assess the feasibility of each option based on the criteria mentioned above.
6. **Make Recommendations:** Provide a recommendation on the most feasible technology solution.
7. **Document Findings:** Compile the findings and recommendations into a comprehensive report.

3.2 ECONOMIC FEASIBILITY

Evaluating the economic viability of P2P involves assessing the costs associated with development, maintenance, and marketing against the anticipated benefits. The feasibility study considers budgetary constraints, potential revenue streams (such as advertisements or premium features), and a projected return on investment. This analysis ensures that P2P is economically sustainable and aligns with market expectations.

Economic feasibility is a crucial aspect of project evaluation that assesses the financial aspects of a proposed project to determine whether it is economically viable. This analysis helps stakeholders understand the cost-benefit ratio, potential financial risks, and overall economic impact of the project. Here’s an in-depth look at the components and steps involved in conducting an economic feasibility study:

Key Components of Economic Feasibility

1. **Cost Analysis**
    - **Initial Costs:** Estimate the upfront costs required to start the project. This includes costs for research and development, procurement of technology, infrastructure setup, and initial marketing.
    - **Operational Costs:** Calculate the ongoing costs associated with running the project. This includes labor, maintenance, utilities, supplies, and other recurring expenses.
    - **Indirect Costs:** Identify indirect costs such as administrative overhead, training, and potential opportunity costs.
2. **Revenue Projections**
    - **Revenue Streams:** Identify potential sources of revenue. This could include product sales, service fees, subscriptions, licensing, or other revenue-generating activities.
    - **Pricing Strategy:** Determine the pricing model and strategy for the product or service. This includes considering competitive pricing, value-based pricing, and market demand.
    - **Sales Forecast:** Project the expected sales volume over time. This involves market analysis, customer segmentation, and historical data if available.
3. **Benefit Analysis**
    - **Tangible Benefits:** Calculate the direct financial benefits, such as increased sales, cost savings, and improved productivity.
    - **Intangible Benefits:** Consider non-monetary benefits like improved customer satisfaction, brand value, market positioning, and competitive advantage.
4. **Break-Even Analysis**
    - **Break-Even Point:** Determine the point at which total revenues equal total costs, indicating when the project will start generating profit.
    - **Payback Period:** Calculate the time required to recoup the initial investment from net cash flows.
5. **Net Present Value (NPV) and Internal Rate of Return (IRR)**
    - **NPV:** Calculate the present value of net cash flows generated by the project, discounted at the project's cost of capital. A positive NPV indicates that the project is expected to generate more value than its cost.

- **IRR:** Determine the discount rate at which the NPV of the project is zero. A higher IRR compared to the cost of capital indicates a financially attractive project.

6. **Sensitivity Analysis**
    - **Variable Impact:** Assess how changes in key variables (e.g., costs, revenues, market conditions) impact the project's financial outcomes.
    - **Risk Assessment:** Identify potential financial risks and develop strategies to mitigate them.

Steps to Conduct an Economic Feasibility Study

1. **Define Objectives and Scope**
    - Clearly outline the financial goals and scope of the project. Define the parameters for the economic feasibility study.
2. **Gather Data**
    - Collect data on costs, revenue projections, market trends, and other relevant financial information.
3. **Estimate Costs**
    - Perform a detailed cost analysis, including initial, operational, and indirect costs.
4. **Forecast Revenues**
    - Project future revenues based on market analysis, historical data, and potential revenue streams.
5. **Perform Benefit Analysis**
    - Calculate both tangible and intangible benefits associated with the project.
6. **Conduct Break-Even Analysis**
    - Determine the break-even point and payback period for the project.
7. **Calculate NPV and IRR**
    - Use financial models to calculate the NPV and IRR, ensuring the project's potential profitability.
8. **Conduct Sensitivity Analysis**
    - Evaluate the impact of varying key assumptions and identify potential financial risks.
9. **Prepare the Report**
    - Compile the findings into a comprehensive report, including financial projections, risk assessments, and recommendations.

3.3 OPERATIONAL FEASIBILITY

The operational feasibility of P2P hinges on its seamless integration into users' daily cooking routines. User-friendliness, efficient recipe categorization, and a responsive interface are critical factors. The study explores how well P2P aligns with user behaviours and whether it can be easily adopted and embraced by the target audience. Operational feasibility also considers the scalability of the platform to accommodate a growing user base.

Operational feasibility is an essential aspect of project evaluation that assesses how well a proposed project or system will function within the existing operational environment of an organization. It examines the compatibility of the new system with the current operations, processes, personnel, and other systems to ensure smooth implementation and sustainable performance. Here's a detailed look at the components and steps involved in assessing operational feasibility:

Key Components of Operational Feasibility

1. **Organizational Compatibility**
    - **Alignment with Goals:** Determine if the project aligns with the organization's strategic goals and objectives. Evaluate how the project supports or enhances these goals.
    - **Cultural Fit:** Assess whether the project is compatible with the organizational culture and values. Consider potential resistance to change and how it can be managed.
2. **Process Integration**
    - **Current Processes:** Analyze existing workflows and processes to identify how the new system will integrate with or impact them. Evaluate any necessary modifications to these processes.
    - **Workflow Efficiency:** Assess how the new system will improve or hinder workflow efficiency. Consider potential bottlenecks or areas where the system may streamline operations.
3. **Resource Availability**
    - **Human Resources:** Evaluate the availability of skilled personnel to support the implementation and operation of the new system. Identify training needs and plan for staff development.
    - **Physical Resources:** Assess the availability of necessary physical resources, such as facilities, equipment, and materials required for the project.
4. **System Interoperability**

- **Compatibility with Existing Systems:** Determine how well the new system will integrate with current systems, databases, and technologies. Identify potential interoperability issues and solutions.
- **Data Migration:** Plan for the migration of data from existing systems to the new system, ensuring data integrity and minimizing disruption.

5. **Operational Support**
    - **Support Infrastructure:** Evaluate the existing support infrastructure, such as IT support, maintenance teams, and help desks. Determine if additional support structures are needed.
    - **Ongoing Maintenance:** Assess the requirements for ongoing maintenance and updates of the new system. Plan for regular updates and potential troubleshooting.
6. **User Acceptance**
    - **User Involvement:** Involve end-users in the planning and implementation process to gather feedback and ensure the system meets their needs.
    - **Training and Support:** Develop comprehensive training programs to ensure users are proficient with the new system. Provide ongoing support to address user issues and concerns.

Steps to Conduct an Operational Feasibility Study

1. **Define Objectives and Scope**
    - Clearly define the objectives of the project and the scope of the operational feasibility study. Identify key stakeholders and their roles.
2. **Gather Data**
    - Collect data on current operations, processes, systems, and resources. Engage with stakeholders to understand their needs and concerns.
3. **Analyze Current Operations**
    - Conduct a thorough analysis of existing workflows, processes, and systems. Identify areas of improvement and potential impacts of the new system.
4. **Assess Resource Requirements**
    - Evaluate the human and physical resources needed for the project. Identify gaps and plan for resource allocation and training.
5. **Evaluate System Integration**
    - Analyse how the new system will integrate with existing systems. Identify potential challenges and develop solutions for smooth integration.

6. **Develop Operational Plan**
    - Create a detailed operational plan outlining how the new system will be implemented, including timelines, resource allocation, and support structures.
7. **Conduct User Acceptance Testing**
    - Involve end-users in testing the new system to gather feedback and identify any usability issues. Adjust the system based on user feedback to ensure it meets operational needs.
8. **Prepare the Report**
    - Compile the findings into a comprehensive report, including analysis, recommendations, and an implementation plan. Ensure the report addresses potential operational challenges and solutions.

3.4 SCHEDULING FEASIBILITY

Meeting deadlines and launching the platform within a reasonable timeframe is a key consideration. The feasibility study outlines the project timeline, identifies potential bottlenecks, and assesses the realistic attainment of milestones. This ensures that the development and deployment of P2P align with market demands and competitor timelines.

Scheduling feasibility is an essential part of project feasibility studies that focuses on determining whether a proposed project can be completed within a specified timeframe. It involves assessing the project's timeline, identifying critical milestones, and ensuring that resources can be allocated efficiently to meet deadlines. Here’s an in-depth look at the components and steps involved in evaluating scheduling feasibility:

Key Components of Scheduling Feasibility

1. **Project Timeline**
    - **Milestones:** Identify major milestones and deliverables that must be achieved throughout the project. These serve as checkpoints to measure progress.
    - **Deadlines:** Establish realistic deadlines for each phase of the project, considering dependencies and potential delays.
2. **Task Breakdown**
    - **Work Breakdown Structure (WBS):** Develop a detailed WBS that outlines all the tasks and subtasks required to complete the project. This helps in understanding the scope and complexity of the project.
    - **Task Dependencies:** Identify dependencies between tasks to understand the sequence in which they must be completed. This is crucial for accurate scheduling.

3. **Resource Allocation**
    - **Human Resources:** Assess the availability of team members and their skill sets. Ensure that the right personnel are allocated to tasks based on their expertise and availability.
    - **Equipment and Materials:** Ensure that necessary equipment and materials will be available when needed. Plan for procurement and delivery schedules.
4. **Time Estimates**
    - **Task Duration:** Estimate the time required to complete each task. Use historical data and expert judgment to make realistic estimates.
    - **Buffer Time:** Include buffer time to account for uncertainties and unexpected delays. This helps in maintaining a realistic schedule.
5. **Critical Path Analysis**
    - **Critical Path Method (CPM):** Use CPM to identify the longest path of dependent tasks from start to finish. This path determines the shortest possible project duration and highlights critical tasks that cannot be delayed without affecting the overall schedule.
6. **Risk Management**
    - **Schedule Risks:** Identify potential risks that could impact the project schedule, such as resource shortages, technical challenges, or external factors.
    - **Mitigation Strategies:** Develop strategies to mitigate identified risks, such as contingency planning, alternative resource allocation, or adjusting timelines.

Steps to Conduct a Scheduling Feasibility Study

1. **Define Project Scope and Objectives**
    - Clearly define the project scope, objectives, and deliverables. Understand the overall goals and how they translate into specific tasks and milestones.
2. **Develop a Work Breakdown Structure (WBS)**
    - Break down the project into manageable tasks and subtasks. Create a detailed WBS to outline all activities required to complete the project.
3. **Estimate Task Durations and Resources**
    - Estimate the time and resources needed for each task. Consider past project data, expert input, and potential uncertainties to make realistic estimates.

4. **Identify Task Dependencies**
    - Determine the dependencies between tasks to understand their sequence. Use tools like Gantt charts or project management software to visualize these dependencies.
5. **Perform Critical Path Analysis**
    - Use CPM to identify the critical path and calculate the minimum project duration. Identify critical tasks that must be closely monitored to prevent schedule delays.
6. **Allocate Resources**
    - Assign human and physical resources to tasks based on availability and expertise. Ensure resources are optimally utilized to meet the project schedule.
7. **Develop the Project Schedule**
    - Create a detailed project schedule, including all tasks, milestones, deadlines, and buffer times. Use project management tools to develop and visualize the schedule.
8. **Review and Adjust**
    - Review the schedule with key stakeholders to ensure it is realistic and achievable. Make necessary adjustments based on feedback and potential risks.
9. **Monitor and Control**
    - Continuously monitor the project's progress against the schedule. Use tools and techniques to track progress, identify deviations, and take corrective actions as needed.
10. **Prepare the Report**
    - Compile the findings into a comprehensive report. Include the detailed schedule, critical path analysis, risk mitigation strategies, and recommendations.

3.5 SECURITY AND PRIVACY FEASIBILITY

Given the nature of user-contributed content and personal data involved in the login process, ensuring robust security and privacy measures is crucial. The feasibility study delves into the implementation of secure authentication practices, data encryption, and measures to protect user information. This address concerns related to user trust and compliance with data protection regulations.

Security and privacy feasibility assess the ability of a proposed project or system to protect sensitive information and maintain confidentiality, integrity, and availability. It involves identifying potential security and privacy risks, implementing appropriate safeguards, and ensuring compliance with relevant regulations and standards. Here's an overview of the key components and steps involved in evaluating security and privacy feasibility:

Key Components of Security and Privacy Feasibility

1. **Risk Assessment**
    - **Threat Identification:** Identify potential threats to the security and privacy of the system, such as unauthorized access, data breaches, malware attacks, or insider threats.
    - **Vulnerability Analysis:** Assess system vulnerabilities that could be exploited by threats. This includes weaknesses in software, hardware, network infrastructure, and human factors.
2. **Security Controls**
    - **Access Control:** Implement mechanisms to control access to the system and its resources, including user authentication, authorization, and encryption.
    - **Data Protection:** Employ encryption, data masking, and data loss prevention techniques to protect sensitive information from unauthorized disclosure or tampering.
    - **Security Monitoring:** Deploy intrusion detection systems, security information and event management (SIEM) tools, and logging mechanisms to monitor and detect security incidents in real-time.
    - **Physical Security:** Ensure physical security measures are in place to protect hardware, data centers, and other physical assets from theft, vandalism, or environmental hazards.
3. **Privacy Considerations**
    - **Data Privacy:** Assess the collection, use, storage, and sharing of personal data to ensure compliance with privacy regulations such as GDPR, CCPA, or HIPAA.
    - **Anonymization and Pseudonymization:** Implement techniques to anonymize or pseudonymize personal data to protect individual privacy while still allowing data analysis and processing.
    - **Consent Management:** Establish processes for obtaining and managing user consent for data processing activities. Ensure transparency and accountability in data handling practices.
4. **Compliance Requirements**
    - **Regulatory Compliance:** Ensure that the system complies with relevant laws, regulations, and industry standards related to security and privacy. This

may include data protection laws, industry-specific regulations, and international standards.

- **Certifications and Audits:** Obtain certifications such as ISO 27001 or SOC 2 to demonstrate compliance with industry best practices and standards. Conduct regular security audits and assessments to identify compliance gaps and areas for improvement.

Steps to Conduct a Security and Privacy Feasibility Study

1. **Define Security and Privacy Requirements**
    - Clearly define the security and privacy requirements of the project, considering factors such as data sensitivity, regulatory requirements, and stakeholder expectations.
2. **Conduct Risk Assessment**
    - Identify potential security and privacy risks associated with the project. Evaluate the likelihood and impact of each risk to prioritize mitigation efforts.
3. **Implement Security Controls**
    - Design and implement appropriate security controls to mitigate identified risks. This may involve the selection and deployment of security technologies, policies, and procedures.
4. **Address Privacy Concerns**
    - Implement measures to protect the privacy of individuals' personal data, including data minimization, consent management, and privacy-enhancing technologies.
5. **Ensure Regulatory Compliance**
    - Review applicable laws, regulations, and industry standards to ensure compliance with security and privacy requirements. Develop processes and documentation to demonstrate compliance.
6. **Test Security and Privacy Measures**
    - Conduct security testing, including penetration testing, vulnerability scanning, and security assessments, to validate the effectiveness of security controls.
    - Perform privacy impact assessments (PIAs) to evaluate the potential privacy risks and impacts of the project on individuals' personal data.
7. **Provide Training and Awareness**

- Train employees and stakeholders on security best practices, privacy policies, and procedures. Foster a culture of security and privacy awareness throughout the organization.

8. **Monitor and Maintain**
    - Continuously monitor the security and privacy posture of the system. Implement incident response procedures to address security incidents and privacy breaches promptly.
    - Regularly review and update security and privacy measures to adapt to evolving threats, technologies, and regulatory requirements.
9. **Prepare Documentation and Reporting**
    - Document the security and privacy measures implemented, including policies, procedures, technical controls, and compliance documentation.
    - Prepare reports summarizing the security and privacy posture of the project for stakeholders, regulators, and auditors.

3.6 USER ENGAGEMENT AND FEEDBACK

Anticipating user engagement and collecting feedback during the development and beta testing phases is integral. The feasibility study explores strategies for soliciting user input, addressing potential challenges, and adapting the platform based on user responses. This iterative approach ensures that P2P remains responsive to user needs and preferences.

In conclusion, the feasibility study for P2P is a holistic evaluation encompassing technical, economic, operational, scheduling, security, and user engagement considerations. By thoroughly assessing these factors, the study provides valuable insights that guide the development process, ultimately increasing the likelihood of P2P's success as a user-centric.

User engagement and feedback are crucial for the success of an online recipe application. Here's a strategy to enhance user engagement and gather valuable feedback:

User Engagement:

1. **Interactive User Interface:**
    - Design a visually appealing and intuitive interface that makes it easy for users to browse recipes, save favourites, and discover new content.
    - Incorporate interactive features such as filters, search bars, and personalized recommendations to enhance the user experience.

2. **Rich Content:**
    - Provide high-quality recipe content with detailed instructions, photos, videos, and nutritional information.
    - Include user-generated content such as reviews, ratings, and photos to add authenticity and engagement.
3. **Social Integration:**
    - Enable users to share recipes with friends and family on social media platforms.
    - Integrate social login options to facilitate seamless registration and encourage social interaction within the app.
4. **Community Building:**
    - Create a community platform where users can engage with each other, ask questions, and share cooking tips and experiences.
    - Host cooking challenges, recipe contests, and virtual cooking classes to foster a sense of belonging and participation.
5. **Personalization:**
    - Offer personalized recommendations based on users' dietary preferences, cooking habits, and past interactions with the app.
    - Allow users to create custom profiles and meal plans tailored to their needs and preferences.
6. **Gamification:**
    - Implement gamification elements such as badges, achievements, and progress tracking to incentivize user engagement and encourage repeat visits.
    - Reward users for completing tasks, trying new recipes, and sharing their achievements with others.

Feedback Collection:

1. **Feedback Forms:**
    - Include in-app feedback forms or surveys to gather input from users about their experience with the app, including likes, dislikes, and suggestions for improvement.
    - Keep the forms short and concise to encourage participation and ensure valuable insights.
2. **Rating and Review System:**
    - Allow users to rate recipes and leave reviews, comments, and feedback.

- Prompt users to leave feedback after trying a recipe, making it easy to share their thoughts directly within the app.

3. **Analytics and User Behaviour Tracking:**
    - Utilize analytics tools to track user behaviour, such as recipe views, search queries, and engagement metrics.
    - Analyse user data to identify trends, preferences, and areas for optimization.
4. **User Interviews and Focus Groups:**
    - Conduct user interviews and focus groups to gain deeper insights into users' needs, motivations, and pain points.
    - Use qualitative research methods to uncover valuable feedback and uncover areas for improvement.
5. **Continuous Iteration:**
    - Regularly review user feedback and prioritize feature enhancements, bug fixes, and content updates based on user input.
    - Communicate with users transparently about changes and updates, demonstrating responsiveness to their feedback.
6. **Incentivized Feedback:**
    - Offer incentives such as discounts, rewards, or entry into sweepstakes for users who provide feedback or participate in surveys.
    - Encourage users to share their feedback by highlighting the impact it has on improving the app and enhancing their experience.

CHEPTER 4

REQUIREMENT ANALYSIS

Requirement analysis for an online quiz application involves identifying and documenting the needs, expectations, and functionalities required for the successful development and deployment of the application. Here's a structured approach to conducting requirement analysis for an online quiz application:

1. **Identify Stakeholders:**
    - Determine who will be involved in the development, deployment, and usage of the application. This could include developers, administrators, quiz creators, and end-users (quiz takers).
2. **Gather Requirements:**
    - Conduct interviews, surveys, and workshops with stakeholders to gather their requirements and expectations from the quiz application.
    - Identify both functional requirements (what the system should do) and non-functional requirements (qualities the system should have).
3. **Define Functional Requirements:**
    - User Registration and Authentication: Users should be able to register and log in securely.
    - Quiz Creation: Administrators should be able to create quizzes, add questions, and set rules.
    - Quiz Taking: Users should be able to select and take quizzes. The system should present questions one at a time and provide feedback on answers.
    - Scoring and Results: The system should calculate scores and provide immediate feedback to users upon quiz completion.
    - Leaderboard: Optionally, include a feature where users can view their ranking compared to others who have taken the quiz.
    - Categories and Tags: Allow quiz creators to categorize quizzes and tag them appropriately for easy searchability.

4. **Define Non-functional Requirements:**
    - Performance: The application should be responsive, even during peak usage times.
    - Security: Ensure secure user authentication, data encryption, and protection against common web vulnerabilities like SQL injection and cross-site scripting (XSS).
    - Usability: The application should have an intuitive user interface, making it easy for users to navigate and interact with quizzes.
    - Compatibility: Ensure compatibility across different devices and web browsers.
    - Scalability: The application should be able to handle a growing number of users and quizzes without significant performance degradation.
    - Reliability: Minimize downtime and ensure the application is stable and reliable.
5. **Prioritize Requirements:**
    - Collaborate with stakeholders to prioritize requirements based on importance and feasibility. Use techniques like MoSCoW (Must have, Should have, Could have, Won't have) to categorize requirements.
6. **Document Requirements:**
    - Create a detailed requirement specification document that outlines each requirement, its priority, and any dependencies or constraints.
    - Use diagrams, wireframes, or mockups to visualize the application's structure and user interface.
7. **Review and Validation:**
    - Review the requirement specification document with stakeholders to ensure it accurately captures their needs and expectations.
    - Validate the requirements against real-world scenarios and use cases to identify any gaps or inconsistencies.
8. **Iterative Development:**
    - As the development progresses, continuously review and refine the requirements to accommodate any changes or new insights.

4.1 FUNCTIONAL REQUIREMENTS

Functional requirements specify what the system or software should do or the actions it should perform. They describe the intended functionality, features, and capabilities of the system. These requirements outline the system's behavior, inputs, outputs, and interactions with users or other systems. Functional requirements are typically specific, measurable, and verifiable. Examples include user authentication, data input validation, report generation, and system integration.

Functional requirements are essential specifications that outline what a system or software application should do to meet the needs of its users. They define the specific functionalities, features, and interactions within the system. In the context of an online quiz application, functional requirements describe the actions users can perform and the system's responses to those actions.

Functional requirements are essential specifications that outline what a system or software application should do to meet the needs of its users. They define the specific functionalities, features, and interactions within the system. In the context of an online quiz application, functional requirements describe the actions users can perform and the system's responses to those actions.

Functional requirements may involve calculations, technical details, data manipulation and processing, and other specific functionality that define what a system is supposed to accomplish. Behavioral requirements describe all the cases where the system uses the functional requirements, these are captured in use cases. Functional requirements are supported by non-functional requirements (also known as "quality requirements"), which impose constraints on the design or implementation (such as performance requirements, security, or reliability). Generally, functional requirements are expressed in the form "system must do <requirement>," while non-functional requirements take the form "system shall be <requirement>. The plan for implementing functional requirements is detailed in the system design, whereas *non-functional* requirements are detailed in the system architecture.[4][5]

As defined in requirements engineering, functional requirements specify particular results of a system. This should be contrasted with non-functional requirements, which specify overall characteristics such as cost and reliability. Functional requirements drive the application architecture of a system, while non-functional requirements drive the technical architecture of a system.

In some cases, a requirements analyst generates use cases after gathering and validating a set of functional requirements. The hierarchy of functional requirements collection and change, broadly speaking, is: user/stakeholder request → analyze → use case incorporate. Stakeholders make a request; systems engineers attempt to discuss, observe, and understand the aspects of the requirement; use cases, entity relationship diagrams, and other models are built to validate the requirement; and, if documented and approved, the requirement is implemented/incorporated. Each use case illustrates behavioral scenarios through one or more functional requirements. Often, though, an analyst will begin by eliciting a set of use cases, from which the analyst can derive the functional requirements that must be implemented to allow a user to perform each use case.

- **User Authentication**

Users must be able to create accounts with unique login credentials.

Secure authentication mechanisms should be implemented for user data protection.

- **Ingredient Input**

Users should have the option to input ingredients they have at home.

The system should support both dynamic list selection and manual entry.

- **Real-Time Recipe Updates**

Recipes should update instantly as users add or remove ingredients.

The system must provide a responsive and seamless user experience.

- **Recipe Categorization**

Recipes should be categorized into user-friendly sections for easy navigation.

Categories may include cuisine types, dietary restrictions, or meal types.

- **Personalized Recommendations**

The platform must analyse user preferences and dietary restrictions to offer personalized recipe recommendations.

Recommendation algorithms should be able to adapt to user feedback and evolving preferences.

- **Recipe Contribution**

Users interested in cooking can contribute recipes.

The contribution process should be straightforward, allowing users to share their culinary creations with the community.

- **Visibility Controls**

Contributors should be able to manage the visibility of their recipes.

Privacy settings should allow control over who can view and access the shared recipes.

- **Search Functionality**

The search tool should efficiently match input ingredients with available recipes.

Search results should be displayed in a user-friendly and organized manner.

4.2 NON-FUNCTIONAL REQUIREMENTS

Non-functional requirements, also known as quality attributes or constraints, define the characteristics and constraints of the system beyond its functionality. These requirements describe how the system should perform, rather than what it should do. Non-functional requirements are often related to performance, reliability, security, usability, and other aspects that contribute to the overall system quality. Examples include response time, system availability, data encryption, user interface design, and regulatory compliance.

Broadly, functional requirements define what a system is supposed to *do* and non-functional requirements define how a system is supposed to *be*. Functional requirements are usually in the form of "system shall do <requirement>", an individual action or part of the system, perhaps explicitly in the sense of a mathematical function, a black box description input, output, process and control functional model or IPO Model. In contrast, non-functional requirements are in the form of "system shall be <requirement>", an overall property of the system as a whole or of a particular aspect and not a specific function. The system's overall properties commonly mark the difference between whether the development project has succeeded or failed.

Non-functional requirements are often called the "quality attributes" of a system. Other terms for non-functional requirements are "qualities", "quality goals", "quality of service requirements", "constraints", "non-behavioral requirements", or "technical requirements". Informally these are sometimes called the "ilities", from attributes like stability and portability. Qualities—that is non-functional requirements—can be divided into two main categories:

1. Execution qualities, such as safety, security and usability, which are observable during operation (at run time).
2. Evolution qualities, such as testability, maintainability, extensibility and scalability, which are embodied in the static structure of the system. It is important to specify non-functional requirements in a specific and measurable .
3. Non-functional requirements, also known as quality attributes or constraints, define the characteristics and constraints of the system beyond its functionality. These requirements describe how the system should perform, rather than what it should do. Non-functional requirements are often related to performance, reliability, security, usability, and other aspects that contribute to the overall system quality. Examples include response time, system availability, data encryption, user interface design, and regulatory compliance.

Non-functional requirements in software engineering refer to the characteristics of a software system that are not related to specific functionality or behavior. They describe how the system should perform, rather than what it should do. Examples of non-functional requirements include:

1. Performance: This includes requirements related to the speed, scalability, and responsiveness of the system. For example, a requirement that the system should be able to handle a certain number of concurrent users or process a certain amount of data within a specific time frame.
2. Security: This includes requirements related to the protection of the system and its data from unauthorized access, as well as the ability to detect and recover from security breaches.
3. Usability: This includes requirements related to the ease of use and understandability of the system for the end-users.
4. Reliability: This includes requirements related to the system’s ability to function correctly and consistently under normal and abnormal conditions.

5. Maintainability: This includes requirements related to the ease of maintaining the system, including testing, debugging, and modifying the system.
6. Portability: This includes requirements related to the ability of the system to be easily transferred to different hardware or software environments.
7. Compliance: This includes requirements related to adherence to laws, regulations, industry standards, or company policies.

Non-Functional Requirements are the constraints or the requirements imposed on the system. They specify the quality attribute of the software. Non-Functional Requirements deal with issues like scalability, maintainability, performance, portability, security, reliability, and many more. Non-Functional Requirements address vital issues of quality for software systems. If NFRs not addressed properly, the results can include:

- **Performance**

The platform must provide quick response times, especially during real-time updates.

Load testing should be conducted to ensure optimal performance under varying user loads.

- **Security**

Robust security measures must be in place to protect user data and privacy.

Encryption protocols should be implemented for secure data transmission.

- **Scalability**

The system should be designed to handle a growing number of users and recipes.

Scalability testing should be performed to assess the platform's ability to expand.

- **Usability**

The user interface should be intuitive, accommodating users with varying levels of technical expertise.

User experience testing should be conducted to ensure ease of use.

- **Reliability**

The platform must be reliable, minimizing downtime and service interruptions.

Implementing backup and recovery mechanisms is essential for data integrity.

- **Regulatory Compliance**

The system must comply with relevant data protection and privacy regulations.

Transparency in terms of service and privacy policies is crucial for user trust.

- **Feedback Mechanism**

A user feedback mechanism should be implemented for continuous improvement.

Analytics tools must be in place to monitor user behaviour and preferences.

In conclusion, the functional requirements outline the specific features and functionalities of the P2P recipe search engine, while the non-functional requirements address aspects such as performance, security, scalability, usability, reliability, regulatory compliance, and user feedback mechanisms. Together, these requirements provide a comprehensive guide for the development and deployment of a successful and user-centric web page.

4.3 SOFTWARE REQUIREMENT

S. NO.	DESCRIPTION	TYPE
1	Operating System	Windows, MacOS
2	Language	HTML5, CSS3, JavaScript, Bootstrap, ExpressJS, NodeJS
3	IDE	VS Code
4	Database	MongoDB

Table 4.3 Software Requirement for Pantry2Plate

4.4 HARDWARE REQUIREMENTS

S. NO.	DESCRIPTION	TYPE
1	Hardware	I3 Processor
2	Clock Speed	3.0GHz
3	RAM	8GB
4	SSD	512GB

Table 4.4 Hardware Requirement for Pantry2Plate

CHEPTER 5

SYSTEM ARCHITECTURE AND DESIGN

5.1 ENTITY-RELATIONSHIP DIAGRAM

Entity-Relationship model stands for an ER model. It is a high-level data model. This model is used to define the data elements and relationship for a specified system. It develops a conceptual design for the database. It also develops a very simple and easy to design view of data. In ER modelling, the database structure is portrayed as a diagram called an entity relationship diagram.

Peter Chen developed the ER diagram in 1976 .The ER model was created to provide a simple and understandable model for representing the structure and logic of databases. It has sin ce evolved into variations such as the Enhanced ER Model and the Object Relationship Model

The Entity Relational Model is a model for identifying entities to be represented in the database and representation of how those entities are related. The ER data model specifies enterprise schema that represents the overall logical structure of a database graphically.

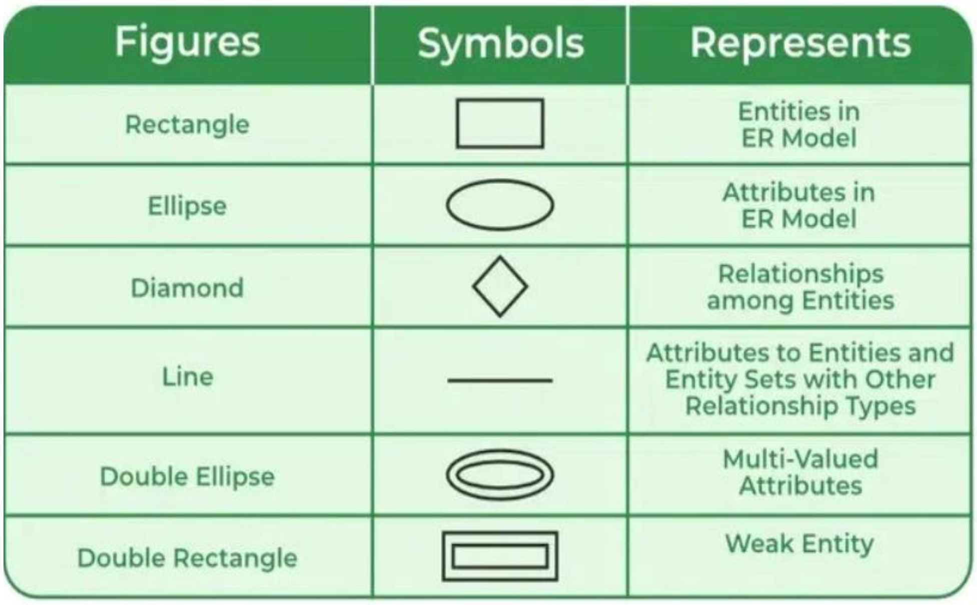
The Entity Relationship Diagram explains the relationship among the entities present in the database. ER models are used to model real-world objects like a person, a car, or a company and the relation between these real-world objects. In short, the ER Diagram is the structural format of the database.

Symbols Used in ER Model

ER Model is used to model the logical view of the system from a data perspective which consists of these symbols:

- **Rectangles:** Rectangles represent Entities in the ER Model.
- **Ellipses:** Ellipses represent Attributes in the ER Model.
- **Diamond:** Diamonds represent Relationships among Entities.
- **Lines:** Lines represent attributes to entities and entity sets with other relationship types.

- **Double Ellipse:** Double Ellipses represent Multi-Valued Attributes.
- **Double Rectangle:** Double Rectangle represents a Weak Entity.

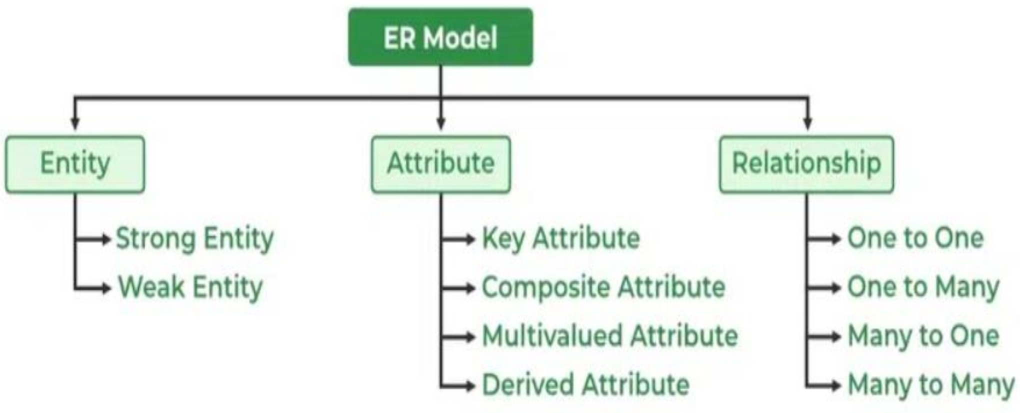


Figures	Symbols	Represents
Rectangle	▭	Entities in ER Model
Ellipse	⬭	Attributes in ER Model
Diamond	◇	Relationships among Entities
Line	—	Attributes to Entities and Entity Sets with Other Relationship Types
Double Ellipse	◎	Multi-Valued Attributes
Double Rectangle	▣	Weak Entity

Fig 5.1. ER Symbols

Components of ER Diagram

ER Model consists of Entities, Attributes, and Relationships among Entities in a Database System.



Entity

An Entity may be an object with a physical existence – a particular person, car, house, or employee – or it may be an object with a conceptual existence – a company, a job, or a university course.

Entity are of two types

1.Tangible Entity – Which can be touched like car , person etc.

2.Non – tangible Entity – Which can't be touched like air , bank account etc.

Entity Set: An Entity is an object of Entity Type and a set of all entities is called an entity set. For Example, E1 is an entity having Entity Type Student and the set of all students is called Entity Set. In ER diagram, Entity Type is represented as We can represent the entity set in ER Diagram but can't represent entity in ER Diagram because entity is row and column in the relation and ER Diagram is graphical representation of data.

1. Strong Entity

A Strong Entity is a type of entity that has a key Attribute. Strong Entity does not depend on other Entity in the Schema. It has a primary key, that helps in identifying it uniquely, and it is represented by a rectangle. These are called Strong Entity Types.

2. Weak Entity

An Entity type has a key attribute that uniquely identifies each entity in the entity set. But some entity type exists for which key attributes can't be defined. These are called Weak Entity types

Attributes

Attributes are the properties that define the entity type. For example, Roll_No, Name, DOB, Age, Address, and Mobile_No are the attributes that define entity type Student. In ER diagram, the attribute is represented by an oval.

1. Key Attribute

The attribute which **uniquely identifies each entity** in the entity set is called the key attribute. For example, Roll No will be unique for each student. In ER diagram, the key attribute is represented by an oval with underlying lines

2. Composite Attribute

An attribute **composed of many other attributes** is called a composite attribute. For example, the Address attribute of the student Entity type consists of Street, City, State, and Country. In ER diagram, the composite attribute is represented by an oval comprising of ovals.

3. Multivalued Attribute

An attribute consisting of more than one value for a given entity. For example, Phone_No (can be more than one for a given student). In ER diagram, a multivalued attribute is represented by a double oval.

4. Derived Attribute

An attribute that can be derived from other attributes of the entity type is known as a derived attribute. e.g.; Age (can be derived from DOB). In ER diagram, the derived attribute is represented by a dashed oval.

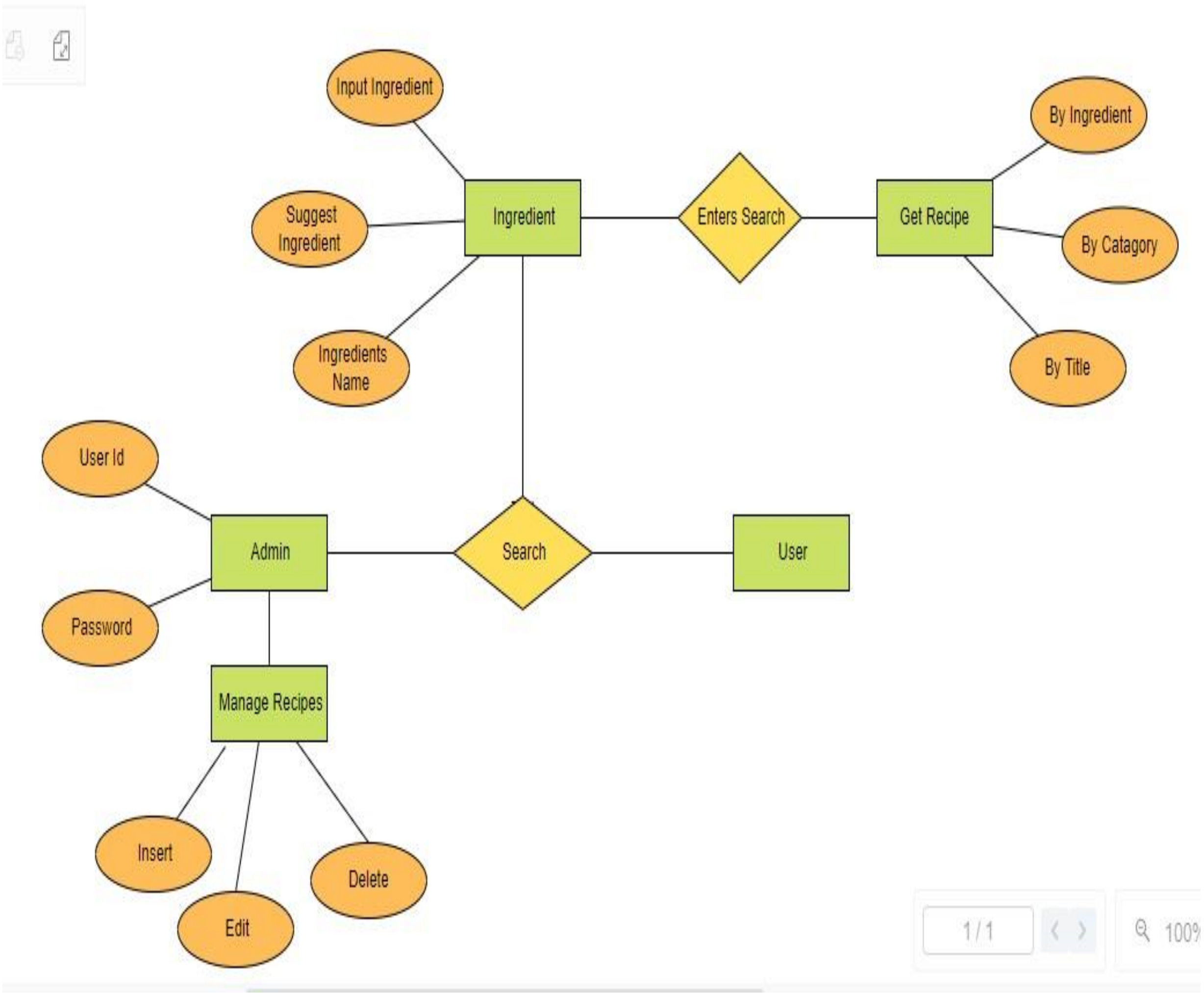


Fig 5.1 Entity-Relationship Diagram

5.2 USE CASE DIAGRAM

A Use Case Diagram is a vital tool in system design, it provides a visual representation of how users interact with a system. It serves as a blueprint for understanding the functional requirements of a system from a user's perspective, aiding in the communication between stakeholders and guiding the development process.

A case study in software engineering is an empirical research method that uses multiple sources of evidence to investigate a contemporary software engineering phenomenon. It's an in-depth examination of a specific real-world situation, project, problem, or success story within the field.

A case study can be a document, such as a video, white paper, or blog post, that outlines how a customer used a product to overcome a problem. It's real-world proof that a product works and gets results.

Case studies are commonly used in social, educational, clinical, and business research. In computer programming courses, case studies involve learners in activities of expert programmers such as identifying decisions, justifying choices among alternatives, and evaluating the consequences of these choices.

5.2.1 Use Case Diagram Notations

UML notations provide a visual language that enables software developers, designers, and other stakeholders to communicate and document system designs, architectures, and behaviours in a consistent and understandable manner.

1.1. Actors

Actors are external entities that interact with the system. These can include users, other systems, or hardware devices. In the context of a Use Case Diagram, actors initiate use cases and receive the outcomes. Proper identification and understanding of actors are crucial for accurately modelling system behaviour.

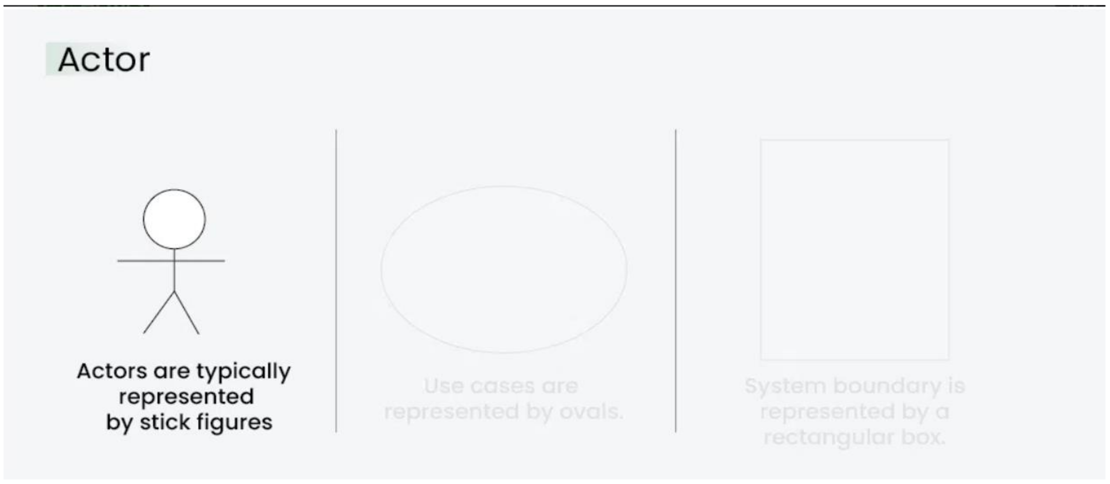

Actor
Actors are typically represented by stick figures
Use cases are represented by ovals.
System boundary is represented by a rectangular box.

Fig 5.1.1 Showing Actor

1.2. Use Cases

Use cases are like scenes in the play. They represent specific things your system can do. In the online shopping system, examples of use cases could be “Place Order,” “Track Delivery,” or “Update Product Information”. Use cases are represented by ovals.

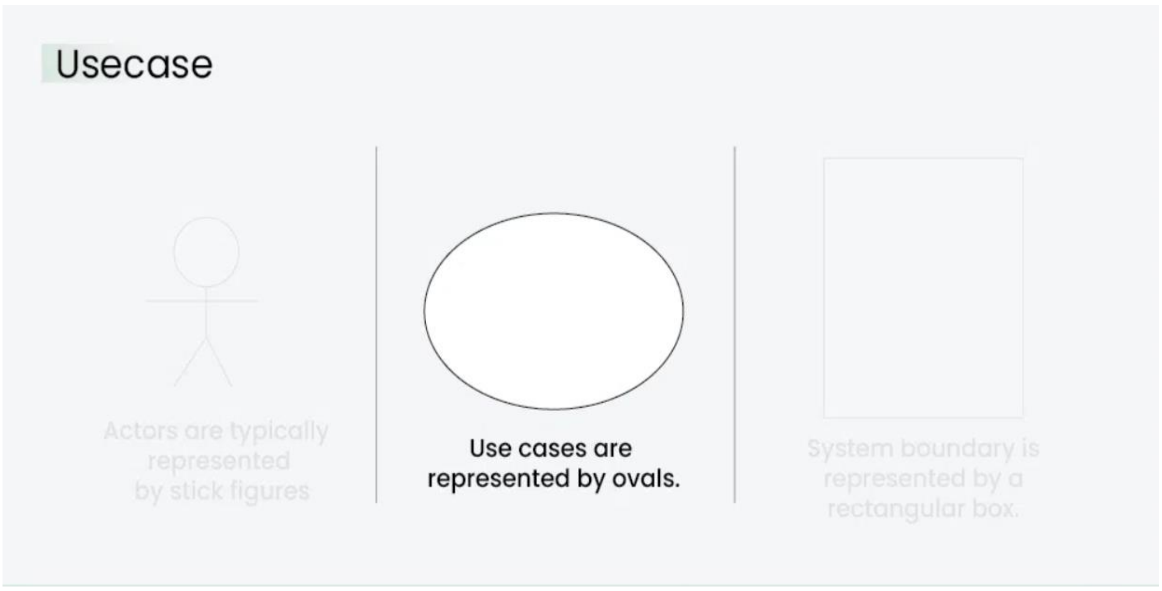

Usecase
Actors are typically represented by stick figures
Use cases are represented by ovals.
System boundary is represented by a rectangular box.

Fig 5.1.2 Showing Use Case

1.3. System Boundary

The system boundary is a visual representation of the scope or limits of the system you are modelling. It defines what is inside the system and what is outside. The boundary helps to establish a clear distinction between the elements that are part of the system and those that are external to it. The system boundary is typically represented by a rectangular box that surrounds all the use cases of the system.

Purpose of System Boundary:

- **Scope Definition:** It clearly outlines the boundaries of the system, indicating which components are internal to the system and which are external actors or entities interacting with the system.
- **Focus on Relevance:** By delineating the system’s scope, the diagram can focus on illustrating the essential functionalities provided by the system without unnecessary details about external entities.

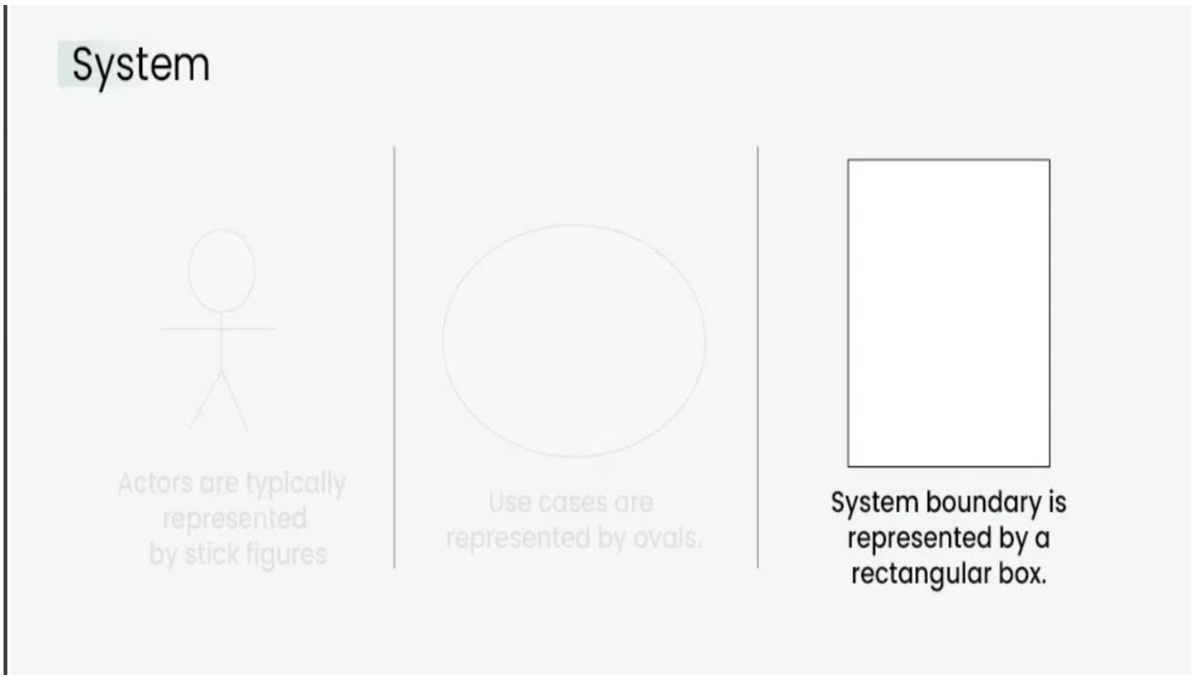

System
Actors are typically represented by stick figures
Use cases are represented by ovals.
System boundary is represented by a rectangular box.

Fig 5.1.3 Showing System

3. Use Case Diagram Relationships

In a Use Case Diagram, relationships play a crucial role in depicting the interactions between actors and use cases. These relationships provide a comprehensive view of the system's functionality and its various scenarios. Let's delve into the key types of relationships and explore examples to illustrate their usage.

3.1. Association Relationship

The Association Relationship represents a communication or interaction between an actor and a use case. It is depicted by a line connecting the actor to the use case. This relationship signifies that the actor is involved in the functionality described by the use case.

Example: Online Banking System

- **Actor:** Customer
- **Use Case:** Transfer Funds
- **Association:** A line connecting the "Customer" actor to the "Transfer Funds" use case, indicating the customer's involvement in the funds transfer process.

3.2. Include Relationship

The Include Relationship indicates that a use case includes the functionality of another use case. It is denoted by a dashed arrow pointing from the including use case to the included use case. This relationship promotes modular and reusable design.

Example: Social Media Posting

- **Use Cases:** Compose Post, Add Image
- **Include Relationship:** The “Compose Post” use case includes the functionality of “Add Image.” Therefore, composing a post includes the action of adding an image.

3.2. Include Relationship

The Include Relationship indicates that a use case includes the functionality of another use case. It is denoted by a dashed arrow pointing from the including use case to the included use case. This relationship promotes modular and reusable design.

Example: Social Media Posting

- **Use Cases:** Compose Post, Add Image
- **Include Relationship:** The “Compose Post” use case includes the functionality of “Add Image.” Therefore, composing a post includes the action of adding an image.

3.3. Extend Relationship

The Extend Relationship illustrates that a use case can be extended by another use case under specific conditions. It is represented by a dashed arrow with the keyword “extend.” This relationship is useful for handling optional or exceptional behavior.

Example: Flight Booking System

- **Use Cases:** Book Flight, Select Seat
- **Extend Relationship:** The “Select Seat” use case may extend the “Book Flight” use case when the user wants to choose a specific seat, but it is an optional step.

3.4. Generalization Relationship

The Generalization Relationship establishes an “is-a” connection between two use cases, indicating that one use case is a specialized version of another. It is represented by an arrow pointing from the specialized use case to the general use case.

Example: Vehicle Rental System

- **Use Cases:** Rent Car, Rent Bike
- **Generalization Relationship:** Both “Rent Car” and “Rent Bike” are specialized versions of the general use case “Rent Vehicle.”

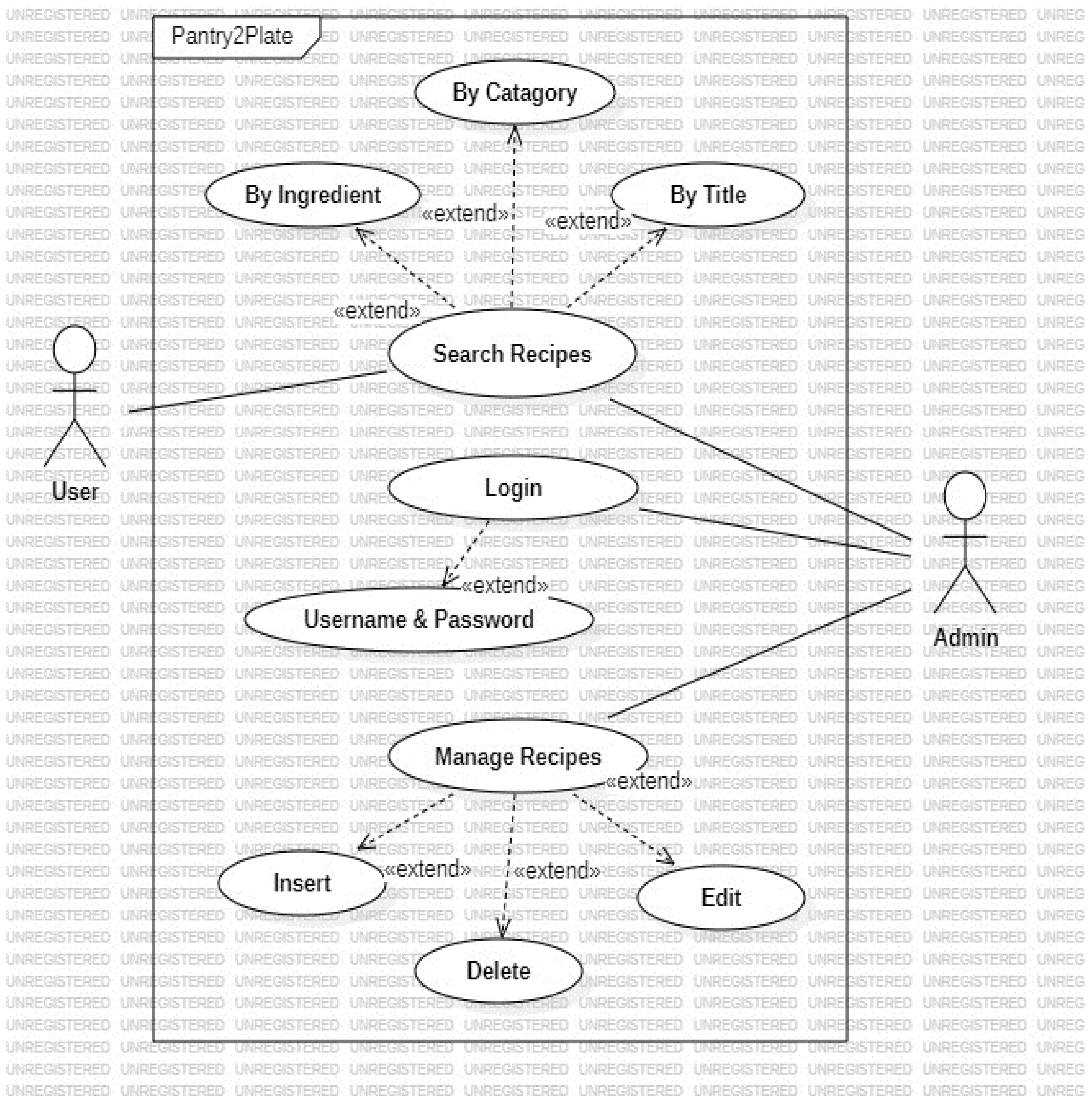


Fig 5.2 Case Study Diagram

5.3 DATA FLOW DIAGRAM

DFD is the abbreviation for **Data Flow Diagram**. The flow of data in a system or process is represented by a Data Flow Diagram (DFD). It also gives insight into the inputs and outputs of each entity and the process itself. Data Flow Diagram (DFD) does not have a control flow and no loops or decision rules are present. Specific operations, depending on the type of data, can be explained by a flowchart. It is a graphical tool, useful for communicating with users, managers and other personnel. it is useful for analysing existing as well as proposed systems.Data Flow Diagram can be represented in several ways. The Data Flow Diagram (DFD) belongs to structured-analysis modelling tools. Data Flow diagrams are very popular because they help us to visualize the major steps and data involved in software-system processes.

Characteristics of Data Flow Diagram (DFD)

Below are some characteristics of Data Flow Diagram (DFD):

- **Graphical Representation**: Data Flow Diagram (DFD) use different symbols and notation to represent data flow within system. That simplify the complex model.
- **Problem Analysis:** Data Flow Diagram **(**DFDs) are very useful in understanding a system and can be effectively used during analysis. Data Flow Diagram (DFDs) are quite general and are not limited to problem analysis for software requirements specification.
- **Abstraction**: Data Flow Diagram (DFD) provides a abstraction to complex model i.e. DFD hides unnecessary implementation details and show only the flow of data and processes within information system.
- **Hierarchy**: Data Flow Diagram (DFD) provides a hierarchy of a system. High- level diagram i.e. 0-level diagram provides an overview of entire system while lower-level diagram like 1-level DFD and beyond provides a detailed data flow of individual process.
- **Data Flow**: The primary objective of Data Flow Diagram (DFD) is to visualize the data flow between external entity, processes and data store. Data Flow is represented by an arrow Symbol.
- **Ease of Understanding**: Data Flow Diagram (DFD) can be easily understand by both technical and non-technical stakeholders.
- **Modularity**: Modularity can be achieved using Data Flow Diagram (DFD) as it breaks the complex system into smaller module or processes. This provides easily analysis and design of a system.

There are two types of Data Flow Diagram (DFD)

1. Logical Data Flow Diagram
2. Physical Data Flow Diagram

Logical Data Flow Diagram (DFD)

Logical data flow diagram mainly focuses on the system process. It illustrates how data flows in the system. Logical Data Flow Diagram (DFD) mainly focuses on high level processes and data flow without diving deep into technical implementation details. Logical DFD is used in various organizations for the smooth running of system. Like in a Banking software system, it is used to describe how data is moved from one entity to another.

Physical Data Flow Diagram

Physical data flow diagram shows how the data flow is actually implemented in the system. In the Physical Data Flow Diagram (DFD), we include additional details such as data storage, data transmission, and specific technology or system components. Physical DFD is more specific and closer to implementation.

Levels of Data Flow Diagram (DFD)

Data Flow Diagram (DFD) uses hierarchy to maintain transparency thus multilevel Data Flow Diagram (DFD's) can be created. Levels of Data Flow Diagram (DFD) are as follows:

0-level DFD

It is also known as a context diagram. It's designed to be an abstraction view, showing the system as a single process with its relationship to external entities. It represents the entire system as a single bubble with input and output data indicated by incoming/outgoing arrows.

1-Level DFD

This level provides a more detailed view of the system by breaking down the major processes identified in the level 0 DFD into sub-processes. Each sub-process is depicted as a separate process on the level 1 DFD. The data flows and data stores associated with each sub-process are also shown. In 1-level DFD, the context diagram is decomposed into multiple bubbles/processes. In this level, we highlight the main functions of the system and breakdown the high-level process of 0-level DFD into subprocesses

2-level DFD

This level provides an even more detailed view of the system by breaking down the sub-processes identified in the level 1 DFD into further sub-processes. Each sub-process is depicted as a separate process on the level 2 DFD. The data flows and data stores associated with each sub-process are also shown.

Rules for Data Flow Diagram (DFD)

Following are the rules of DFD:

- Data can flow from:
    - Terminator or External Entity to Process
    - Process to Terminator or External Entity
    - Process to Data Store
    - Data Store to Process
    - Process to Process
- Data Cannot Flow From
    - Terminator or External Entity to Terminator or External Entity
    - Terminator or External Entity to Data Store
    - Data Store to Terminator or External Entity
    - Data Store to Data Store

Components of Data Flow Diagrams (DFD)

The Data Flow Diagram has 4 components:

- **Process:** Input to output transformation in a system takes place because of process function. The symbols of a process are rectangular with rounded corners, oval, rectangle or a circle. The process is named a short sentence, in one word or a phrase to express its essence
- **Data Flow:** Data flow describes the information transferring between different parts of the systems. The arrow symbol is the symbol of data flow. A relatable name should be given to the flow to determine the information which is being moved. Data flow also represents material along with information that is being moved. Material shifts are modeled in systems that are not merely informative. A given flow should only transfer a single type of information. The direction of flow is represented by the arrow which can also be bi-directional.
- **Warehouse (Data Store) :** The data is stored in the warehouse for later use. Two horizontal lines represent the symbol of the store. The warehouse is simply not restricted to being a data file rather it can be anything like a folder with documents, an optical disc, a filing cabinet. The data warehouse can be viewed independent of its implementation. When the data flow from the warehouse it is considered as data reading and when data flows to the warehouse it is called data entry or data updating.
- **Terminator (External Entity):** The Terminator is an external entity that stands outside of the system and communicates with the system. It can be, for example, organizations

like banks, groups of people like customers or different departments of the same organization, which is not a part of the model system and is an external entity.

A data flow diagram (DFD) is a graphical representation of how data moves through a system or process. DFDs are a key tool in software development and business analysis.

DFDs have no control flow, meaning they don't have decision rules or loops. Instead, specific operations based on the data can be represented by a flowchart.

DFDs have four main components: Processes, Data flows, Data stores, External entities.

DFDs have two levels:

0-level DFD

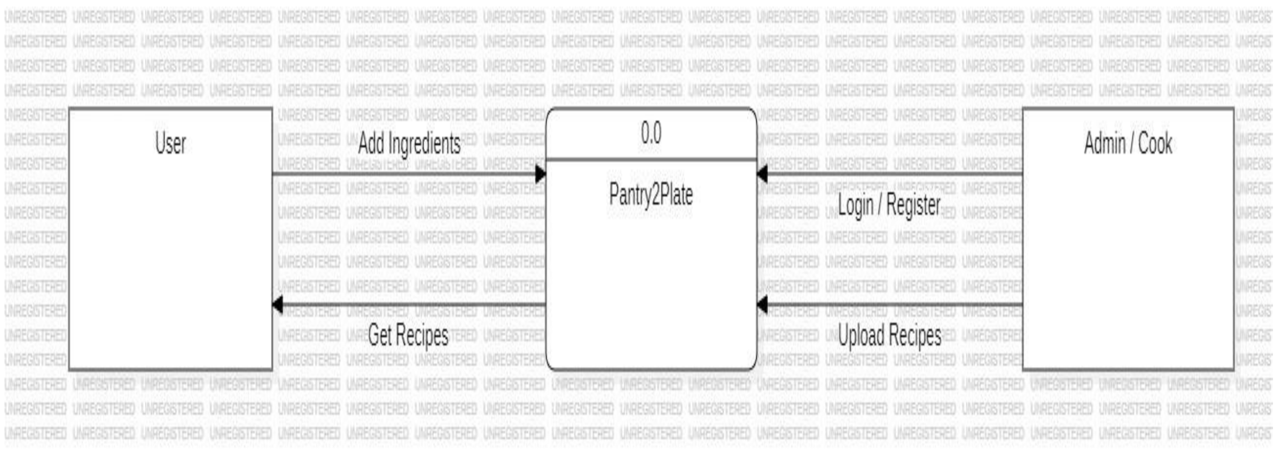


Fig 5.3.1 Data Flow Diagram 0-level

1-level DFD

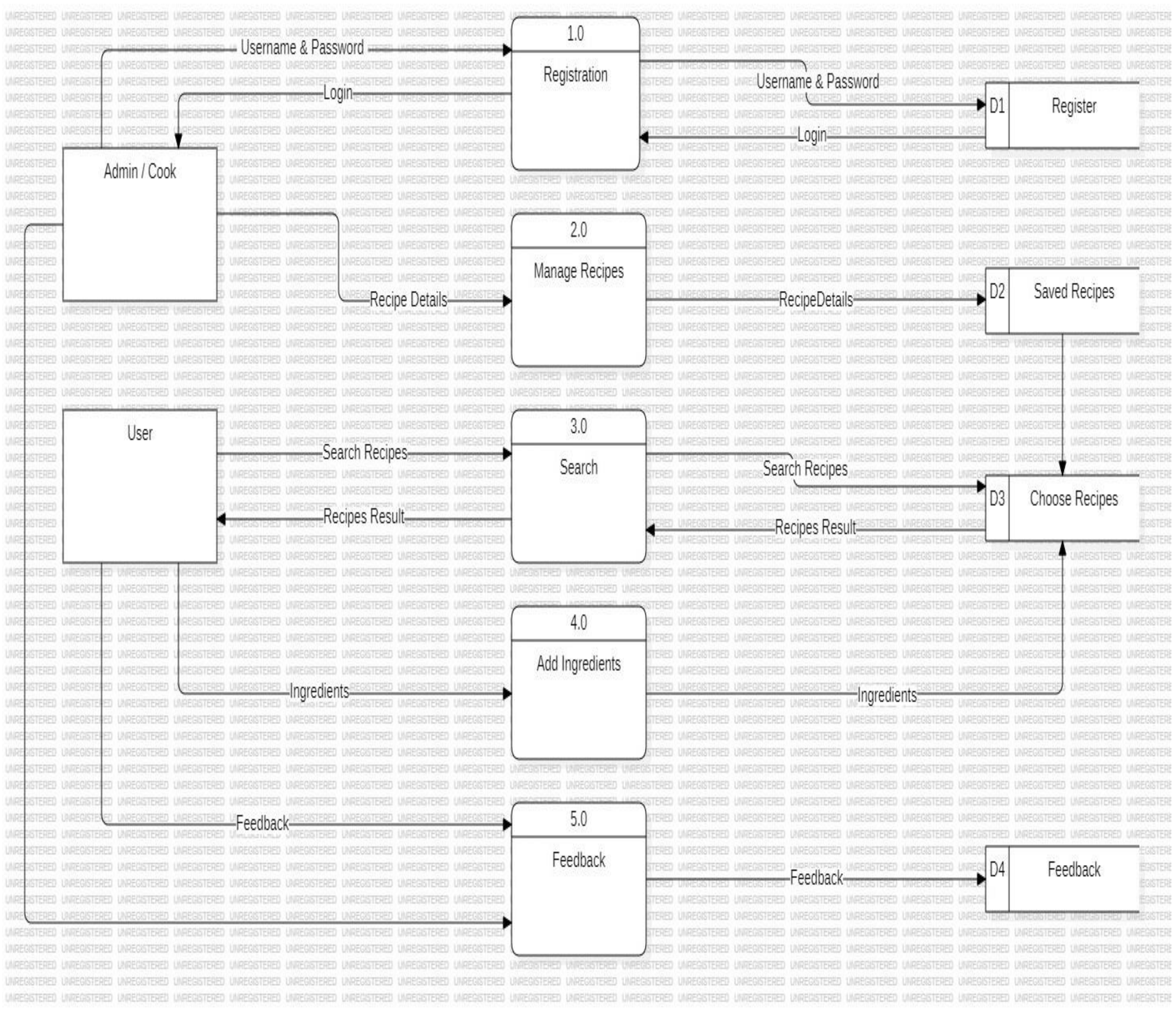


Fig 5.3.2 Data Fow Diagram 1-level

CHEPTER 6

DATABASE DESIGN

6.1 INTRODUCTION

MongoDB (link resides outside IBM) is an open source, nonrelational database management system (DBMS) that uses flexible documents instead of tables and rows to process and store various forms of data. As a NoSQL database solution, MongoDB does not require a relational database management system (RDBMS), so it provides an elastic data storage model that enables users to store and query multivariate data types with ease. This not only simplifies database management for developers but also creates a highly scalable environment for cross-platform applications and services.

MongoDB documents or collections of documents are the basic units of data. Formatted as Binary JSON (Java Script Object Notation), these documents can store various types of data and be distributed across multiple systems. Since MongoDB employs a dynamic schema design, users have unparalleled flexibility when creating data records, querying document collections through MongoDB aggregation and analysing large amounts of information.

6.2 DATA MODELING

Data modelling refers to the organization of data within a database and the links between related entities. Data in MongoDB has a **flexible schema model**, which means:

- Documents within a single collection are not required to have the same set of fields.
- A field's data type can differ between documents within a collection.

Generally, documents in a collection share a similar structure. To ensure consistency in your data model, you can create schema validation rules.

6.3 USE CASES

The flexible data model lets you organize your data to match your application's needs. MongoDB is a document database, meaning you can embed related data in object and array fields.

A flexible schema is useful in the following scenarios:

- Your company tracks which department each employee works in. You can embed department information inside of the employee collection to return relevant information in a single query.
- Your e-commerce application shows the five most recent reviews when displaying a product. You can store the recent reviews in the same collection as the product data, and store older reviews in a separate collection because the older reviews are not accessed as frequently.
- Your clothing store needs to create a single-page application for a product catalog. Different products have different attributes, and therefore use different document fields. However, you can store all of the products in the same collection.

6.4 SCHEMA DESIGN

When you design a schema for a document database like MongoDB, there are a couple of important differences from relational databases to consider.

Relational Database Behaviour	**Document Database Behaviour**
You must determine a table's schema before you insert data.	Your schema can change over time as the needs of your application change.
You often need to join data from several different tables to return the data needed by your application.	The flexible data model lets you store data to match the way your application returns data, and avoid joins. Avoiding joins across multiple collections improves performance and reduces your deployment's workload.

6.5 LINK RELATED DATA

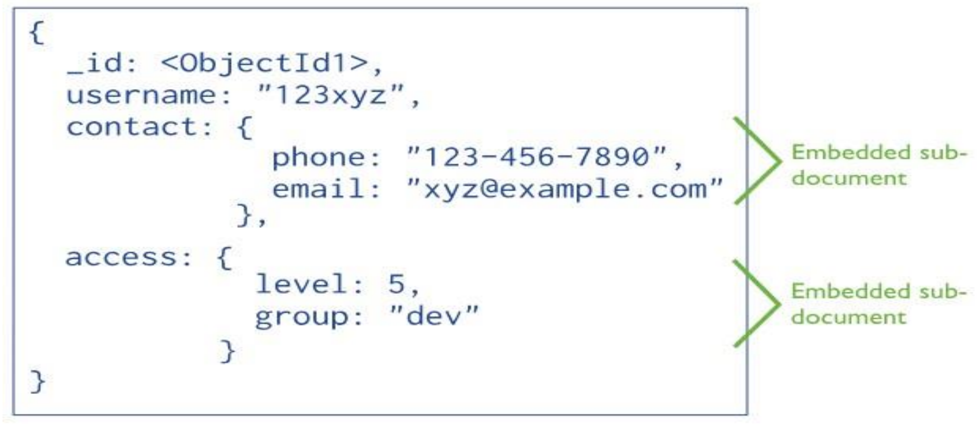
When you design your data model in MongoDB, consider the structure of your documents and the ways your application uses data from related entities.

To link related data, you can either:

- Embed related data within a single document.
- Store related data in a separate collection and access it with a reference.

6.5.1 Embedded Data

Embedded documents store related data in a single document structure. A document can contain arrays and sub-documents with related data. These **denormalized** data models allow applications to retrieve related data in a single database operation.



```
{
   _id: <ObjectId1>,
   username: "123xyz",
   contact: {
                phone: "123-456-7890",
                email: "xyz@example.com"
            },
   access: {
               level: 5,
               group: "dev"
           }
}
```

Fig 6.5.1 Embedded Data

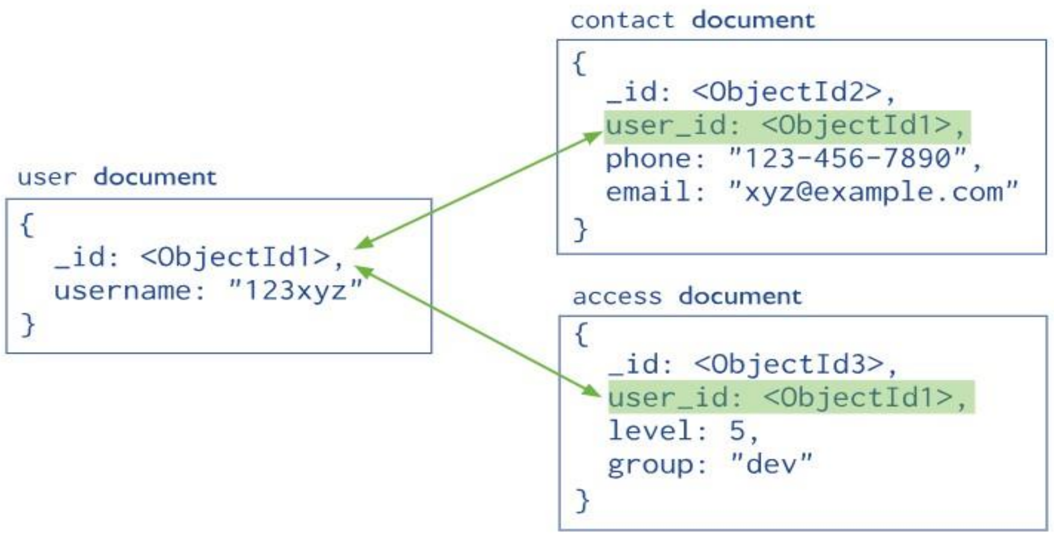
For many use cases in MongoDB, the denormalized data model is optimal.

To learn about the strengths and weaknesses of embedding documents, see Embedded Data Models.

6.5.2 References

References store relationships between data by including links, called **references**, from one document to another. For example, a customerId field in an orders collection indicates a reference to a document in a customer’s collection.

Applications can resolve these references to access the related data. Broadly, these are normalized data models.



```
user document
{
  _id: <ObjectId1>,
  username: "123xyz"
}

contact document
{
  _id: <ObjectId2>,
  user_id: <ObjectId1>,
  phone: "123-456-7890",
  email: "xyz@example.com"
}

access document
{
  _id: <ObjectId3>,
  user_id: <ObjectId1>,
  level: 5,
  group: "dev"
}
```

Fig 6.5.2 References

6.6 DATA STORAGE AND RETRIEVAL

The storage engine is the component of the database that is responsible for managing how data is stored, both in memory and on disk. MongoDB supports multiple storage engines, as different engines perform better for specific workloads. Choosing the appropriate storage engine for your use case can significantly impact the performance of your applications.

You can perform find operations to retrieve data from your MongoDB database. You can perform a find operation to match documents on a set of criteria by calling the find () or find One () method.

CHEPTER 7

FORM DESIGN

7.1 LANDING PAGE

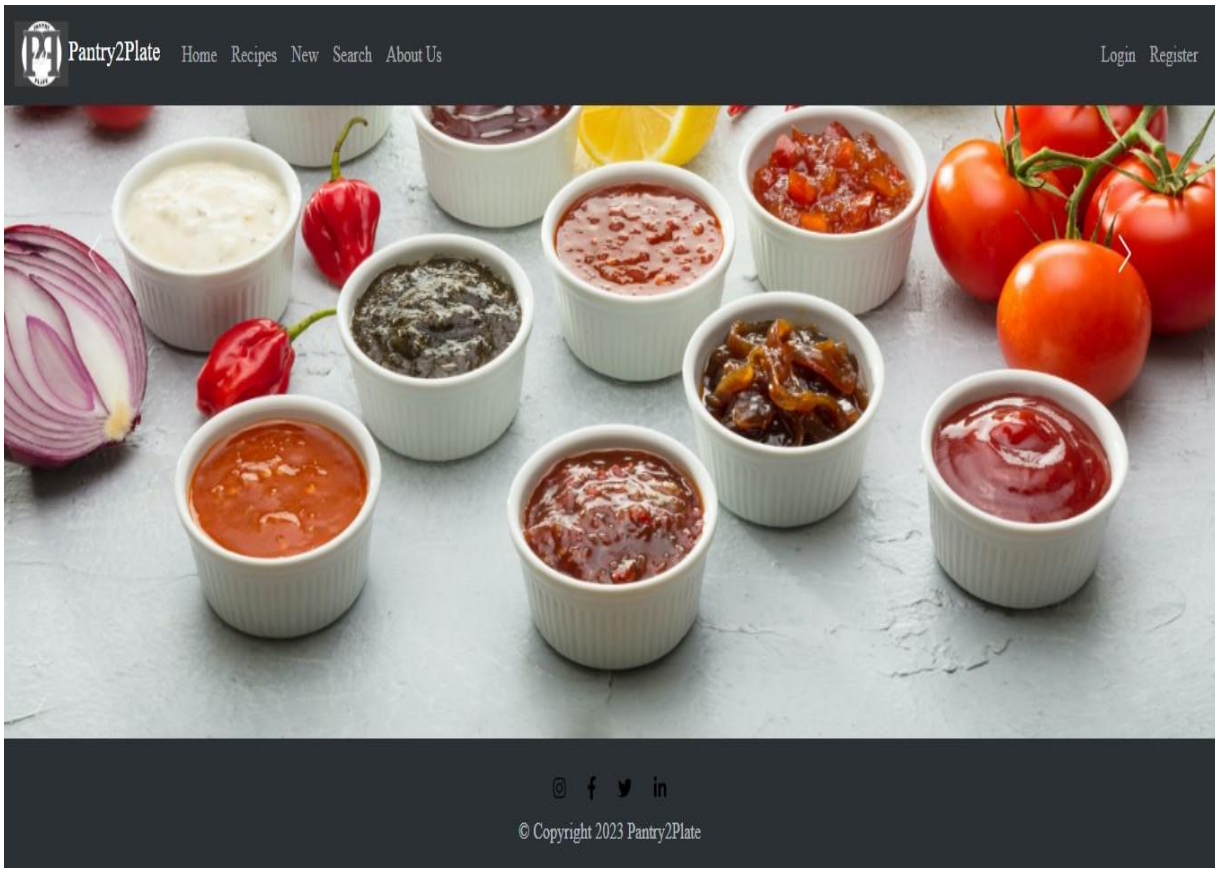
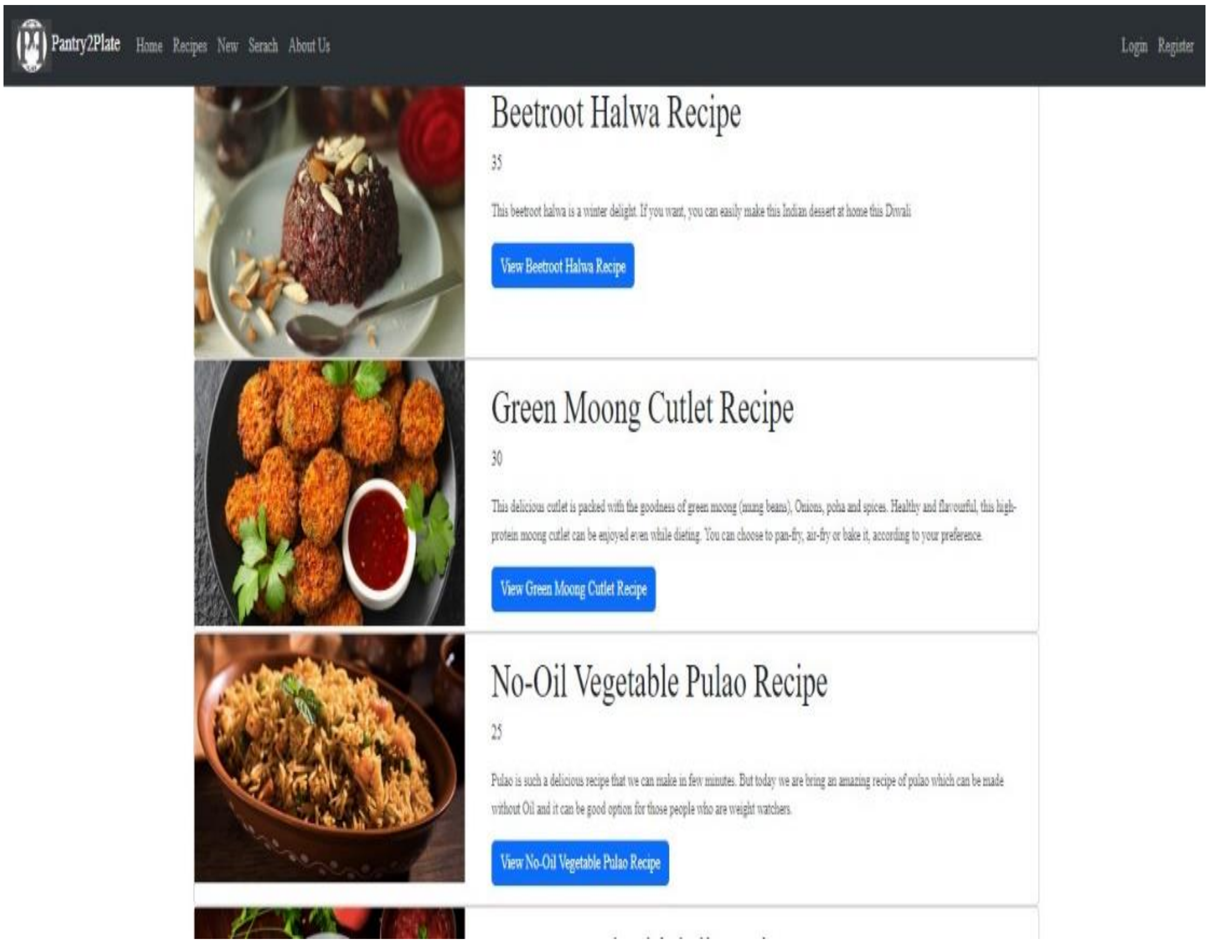


Fig 7.1 Landing Page

7.2 LIST OF RECIPES



Beetroot Halwa Recipe

35

This beetroot halwa is a winter delight. If you want, you can easily make this Indian dessert at home this Diwali

Green Moong Cutlet Recipe

30

This delicious cutlet is packed with the goodness of green moong (mung beans), Onions, poha and spices. Healthy and flavourful, this high-protein moong cutlet can be enjoyed even while dieting. You can choose to pan-fry, air-fry or bake it, according to your preference.

No-Oil Vegetable Pulao Recipe

25

Pulao is such a delicious recipe that we can make in few minutes. But today we are bring an amazing recipe of pulao which can be made without Oil and it can be good option for those people who are weight watchers.

Fig 7.2 List of Recipes

7.3 LOGIN PAGE

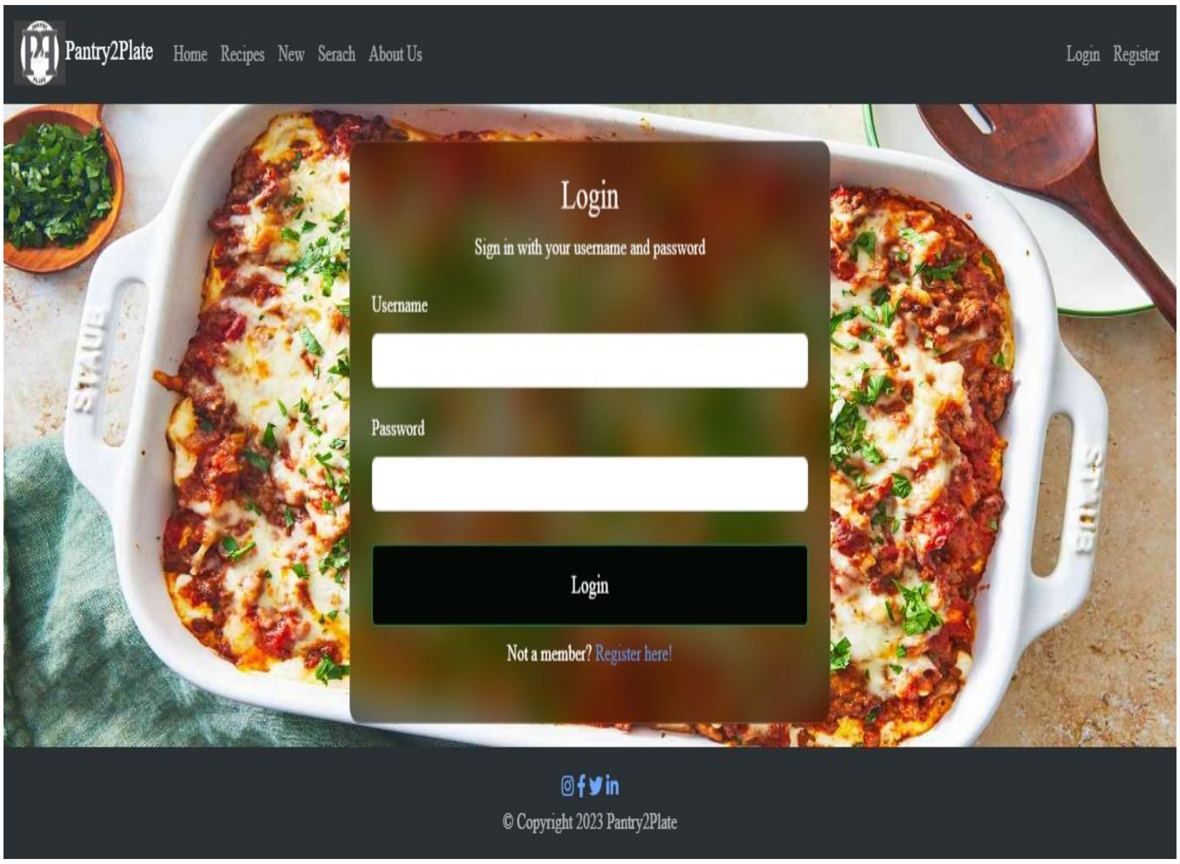

Pantry2Plate
Home
Recipes
New
Serach
About Us
Login
Register
Login
Sign in with your username and password
Username
Password
Login
Not a member? Register here!
© Copyright 2023 Pantry2Plate

Fig 7.3 Login Page

7.4 REGISTER PAGE

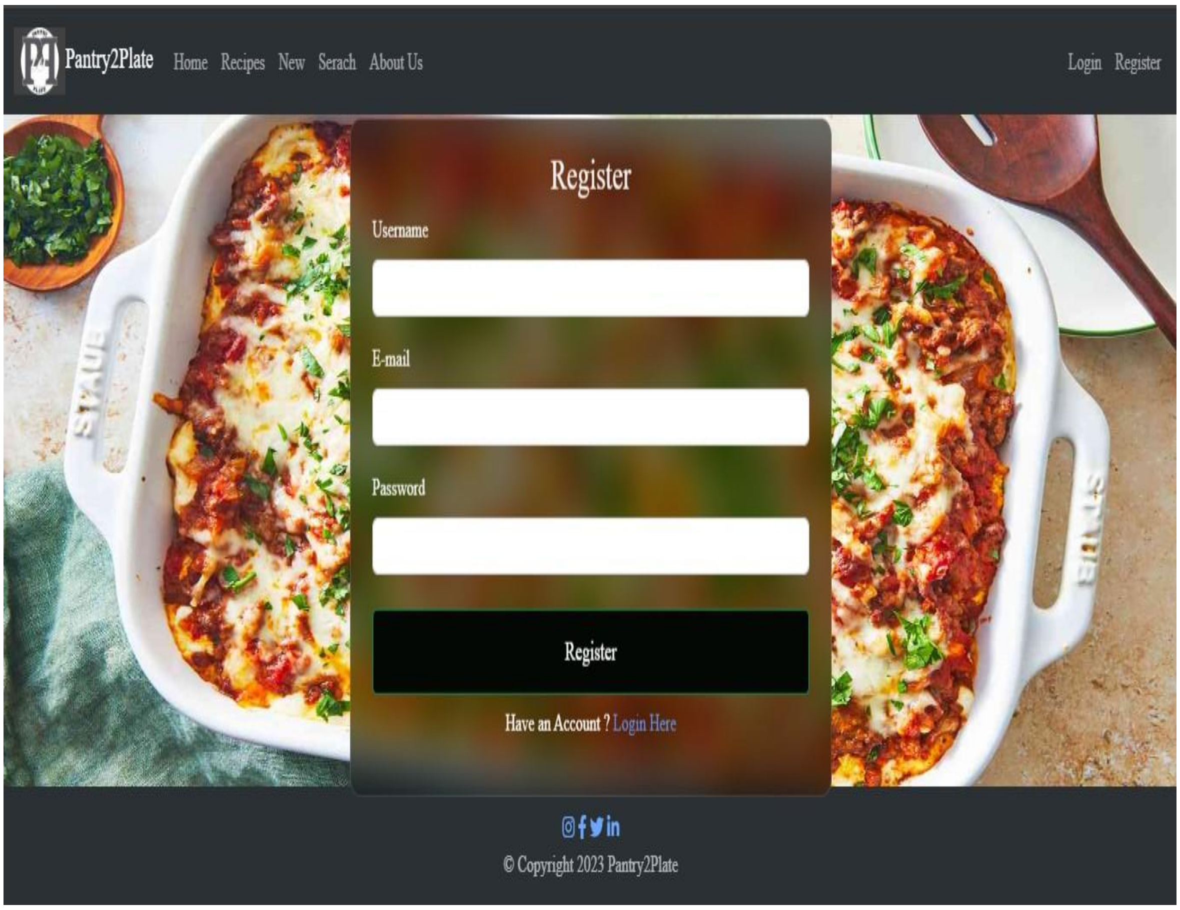


Fig 7.4 Register Page

7.5 ADDING RECIPE PAGE

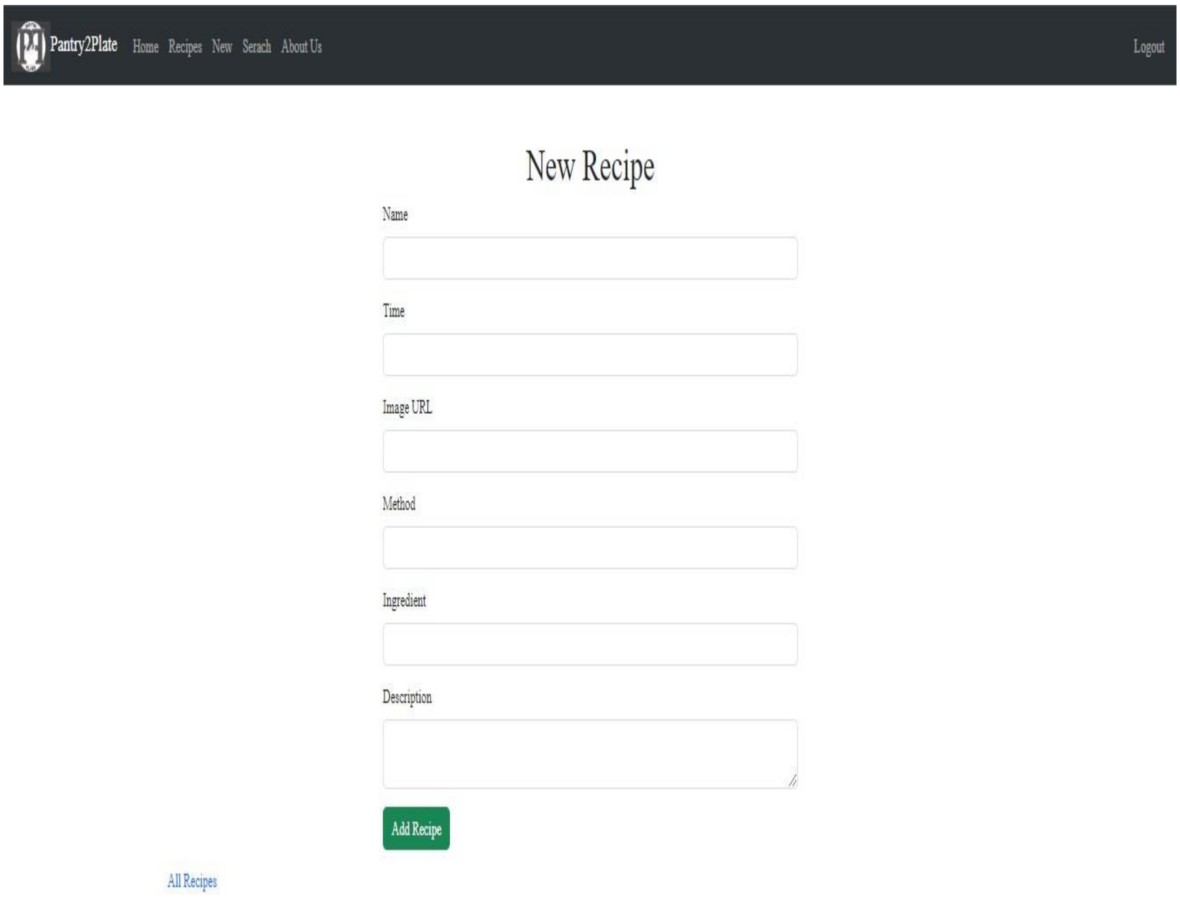
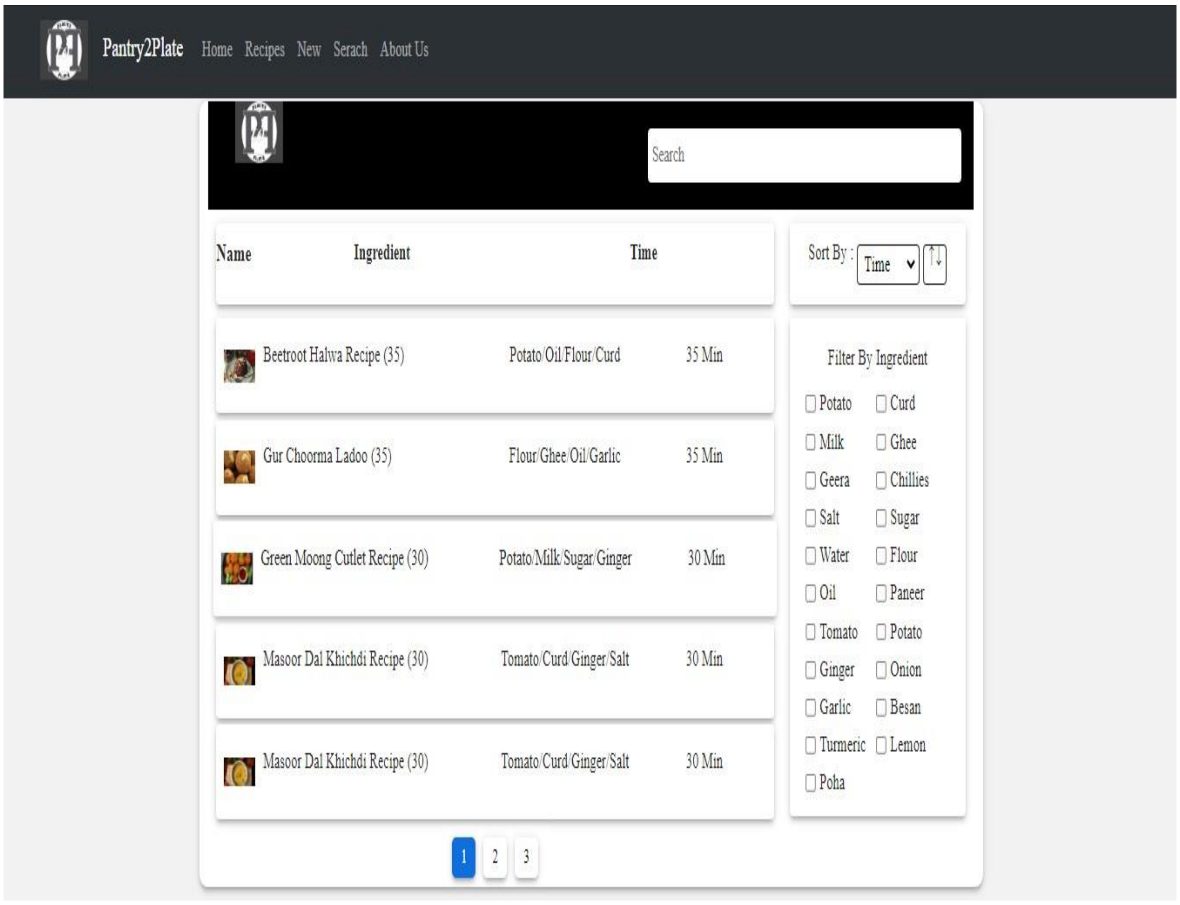


Fig 7.5 Adding Recipe Page

7.6 SEARCH RECIPE PAGE


Pantry2Plate Home Recipes New Serach About Us

Search

Sort By : Time

Name	Ingredient	Time
Beetroot Halwa Recipe (35)	Potato/Oil/Flour/Curd	35 Min
Gur Choorma Ladoo (35)	Flour/Ghee/Oil/Garlic	35 Min
Green Moong Cutlet Recipe (30)	Potato/Milk/Sugar/Ginger	30 Min
Masoor Dal Khichdi Recipe (30)	Tomato/Curd/Ginger/Salt	30 Min
Masoor Dal Khichdi Recipe (30)	Tomato/Curd/Ginger/Salt	30 Min

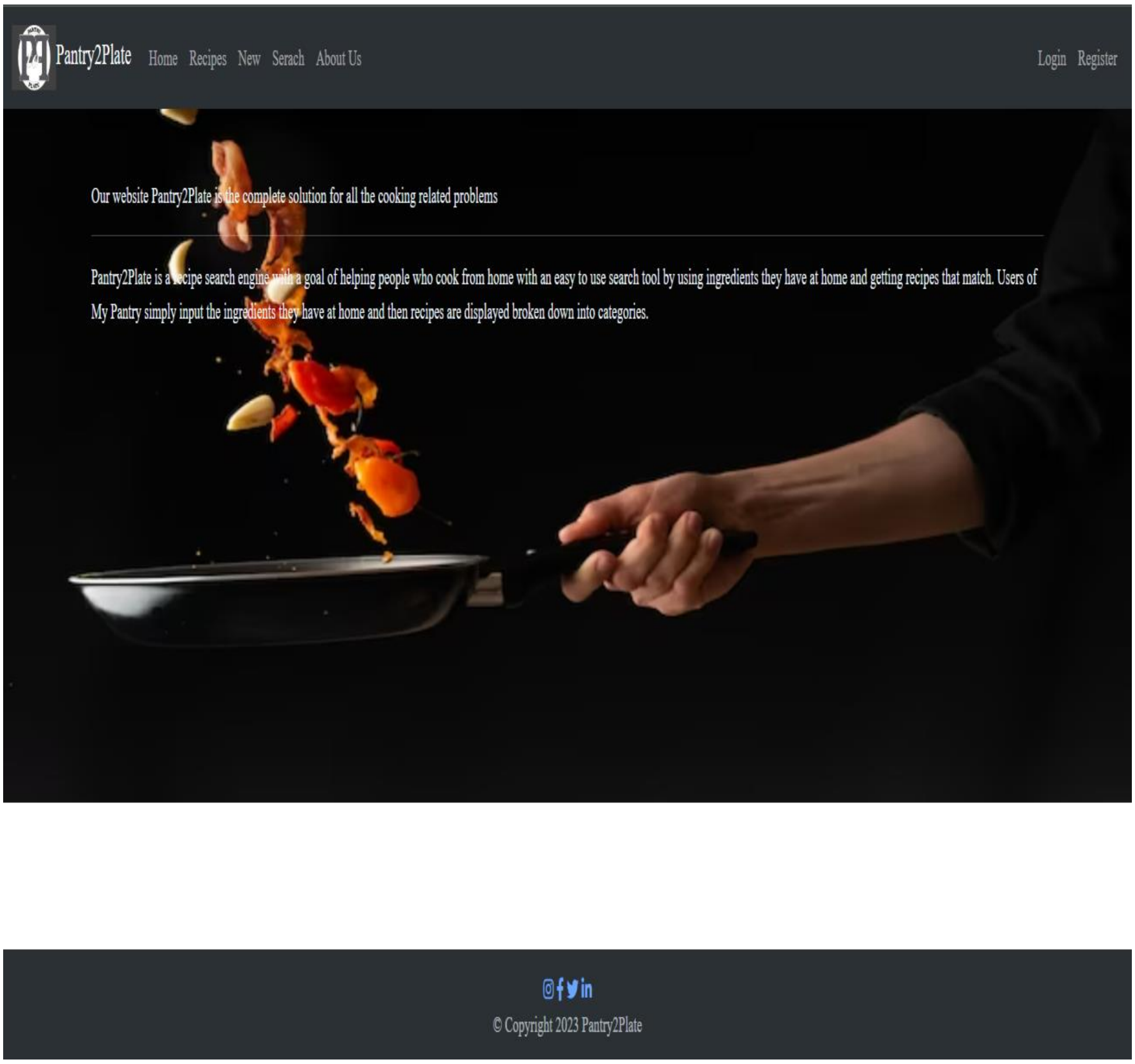
Filter By Ingredient

☐ Potato ☐ Curd
☐ Milk ☐ Ghee
☐ Geera ☐ Chillies
☐ Salt ☐ Sugar
☐ Water ☐ Flour
☐ Oil ☐ Paneer
☐ Tomato ☐ Potato
☐ Ginger ☐ Onion
☐ Garlic ☐ Besan
☐ Turmeric ☐ Lemon
☐ Poha

1 2 3

Fig 7.6 Search Recipe Page

7.7 ABOUT RECIPE PAGE



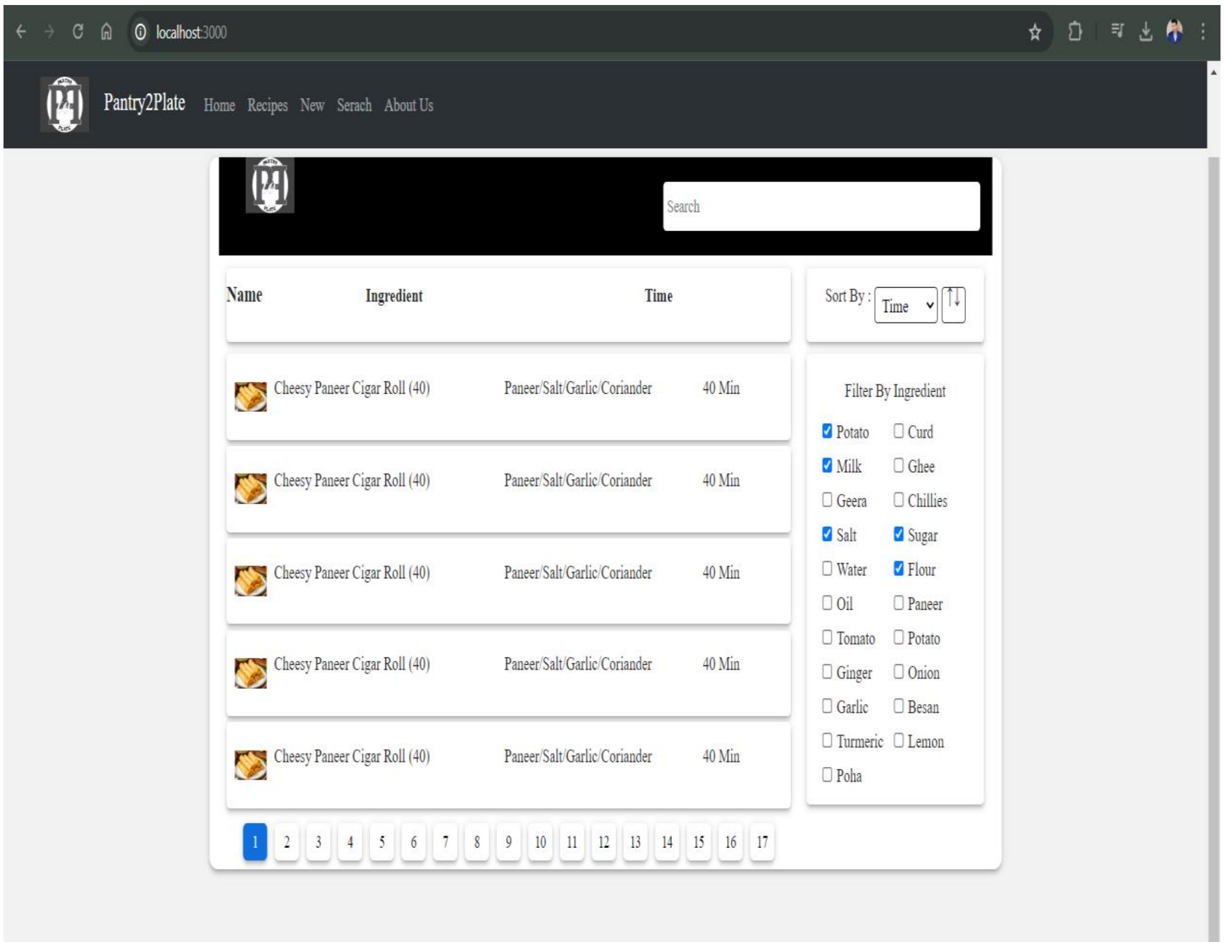
Our website Pantry2Plate is the complete solution for all the cooking related problems

Pantry2Plate is a recipe search engine with a goal of helping people who cook from home with an easy to use search tool by using ingredients they have at home and getting recipes that match. Users of My Pantry simply input the ingredients they have at home and then recipes are displayed broken down into categories.

© Copyright 2023 Pantry2Plate

Fig 7.7 About Recipe Page

7.8 SEARCH USING FILTER



localhost:3000

Pantry2Plate Home Recipes New Serach About Us

Search

Name	Ingredient	Time
Cheesy Paneer Cigar Roll (40)	Paneer/Salt/Garlic/Coriander	40 Min
Cheesy Paneer Cigar Roll (40)	Paneer/Salt/Garlic/Coriander	40 Min
Cheesy Paneer Cigar Roll (40)	Paneer/Salt/Garlic/Coriander	40 Min
Cheesy Paneer Cigar Roll (40)	Paneer/Salt/Garlic/Coriander	40 Min
Cheesy Paneer Cigar Roll (40)	Paneer/Salt/Garlic/Coriander	40 Min

Sort By : Time

Filter By Ingredient

[✓ Potato]	Curd
[✓ Milk]	Ghee
Geera	Chillies
[✓ Salt]	[✓ Sugar]
Water	[✓ Flour]
Oil	Paneer
Tomato	Potato
Ginger	Onion
Garlic	Besan
Turmeric	Lemon
Poha	

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17

Fig 7.8 Search Recipe Page

CHEPTER 8

TESTING

Testing is the process of executing a program to find errors. To make our software perform well it should be error-free. If testing is done successfully it will remove all the errors from the software. In this article, we will discuss first the principles of testing and then we will discuss, the different types of testing.

Principles of Testing

- All the tests should meet the customer's requirements.
- To make our software testing should be performed by a third party.
- Exhaustive testing is not possible. As we need the optimal amount of testing based on the risk assessment of the application.
- All the tests to be conducted should be planned before implementing it
- It follows the Pareto rule(80/20 rule) which states that 80% of errors come from 20% of program components.
- Start testing with small parts and extend it to large parts.
- Types of Testing

Different Types of Software Testing

1. **Manual Testing**
2. **Automation Testing**

1. Manual Testing

Manual testing is a technique to test the software that is carried out using the functions and features of an application. In manual software testing, a tester carries out tests on the software by following a set of predefined test cases. In this testing, testers make test cases for the codes, test the software, and give the final report about that software. Manual testing is time-consuming because it is done by humans, and there is a chance of human errors.

Advantages of Manual Testing:

- **Fast and accurate visual feedback:** It detects almost every bug in the software application and is used to test the dynamically changing GUI designs like layout, text, etc.
- **Less expensive:** It is less expensive as it does not require any high-level skill or a specific type of tool.
- **No coding is required:** No programming knowledge is required while using the black box testing method. It is easy to learn for the new testers.
- **Efficient for unplanned changes:** Manual testing is suitable in case of unplanned changes to the application, as it can be adopted easily.

2. Automation Testing

Automated Testing is a technique where the Tester writes scripts on their own and uses suitable Software or Automation Tool to test the software. It is an Automation Process of a Manual Process. It allows for executing repetitive tasks without the intervention of a Manual Tester.

Advantages of Automation Testing:

- **Simplifies Test Case Execution:** Automation testing can be left virtually unattended and thus it allows monitoring of the results at the end of the process. Thus, simplifying the overall test execution and increasing the efficiency of the application.
- **Improves Reliability of Tests:** Automation testing ensures that there is equal focus on all the areas of the testing, thus ensuring the best quality end product.
- **Increases amount of test coverage:** Using automation testing, more test cases can be created and executed for the application under test. Thus, resulting in higher test coverage and the detection of more bugs. This allows for the testing of more complex applications and more features can be tested.
- **Minimizing Human Interaction:** In automation testing, everything is automated from test case creation to execution thus there are no changes for human error due to neglect. This reduces the necessity for fixing glitches in the post-release phase.

Types of Manual Testing

1. **White Box Testing**
2. **Black Box Testing**
3. **Gray Box Testing**

1. White Box Testing

White box testing is a software testing technique that involves testing the internal structure and workings of a software application. The tester has access to the source code and uses this knowledge to design test cases that can verify the correctness of the software at the code level.

Advantages of Whitebox Testing:

- **Thorough Testing**: White box testing is thorough as the entire code and structures are tested.
- **Code Optimization:** It results in the optimization of code removing errors and helps in removing extra lines of code.
- **Early Detection of Defects:** It can start at an earlier stage as it doesn't require any interface as in the case of black box testing.
- **Integration with SDLC:** White box testing can be easily started in the Software Development Life Cycle.
- **Detection of Complex Defects:** Testers can identify defects that cannot be detected through other testing techniques.

2. Black Box Testing

Black-box testing is a type of software testing in which the tester is not concerned with the internal knowledge or implementation details of the software but rather focuses on validating the functionality based on the provided specifications or requirements.

Advantages of Black Box Testing:

- The tester does not need to have more functional knowledge or programming skills to implement the Black Box Testing.
- It is efficient for implementing the tests in the larger system.
- Tests are executed from the user's or client's point of view.
- Test cases are easily reproducible.
- It is used to find the ambiguity and contradictions in the functional specifications.

3. Gray Box Testing

Gray Box Testing is a software testing technique that is a combination of the Black Box Testing technique and the White Box Testing technique.

1. In the Black Box Testing technique, the tester is unaware of the internal structure of the item being tested and in White Box Testing the internal structure is known to the tester.

2. The internal structure is partially known in Gray Box Testing.
3. This includes access to internal data structures and algorithms to design the test cases.

Advantages of Gray Box Testing:

1. Clarity of goals: Users and developers have clear goals while doing testing.
2. Done from a user perspective: Gray box testing is mostly done from the user perspective.
3. High programming skills not required: Testers are not required to have high programming skills for this testing.
4. Non-intrusive: Gray box testing is non-intrusive.
5. Improved product quality: Overall quality of the product is improved**.**

Types of Black Box Testing

1. **Functional Testing**
2. **Non-Functional Testing**

1. Functional Testing

Functional Testing is a type of Software Testing in which the system is tested against the functional requirements and specifications. Functional testing ensures that the requirements or specifications are properly satisfied by the application. This type of testing is particularly concerned with the result of processing. It focuses on the simulation of actual system usage but does not develop any system structure assumptions. The article focuses on discussing function testing.

Benefits of Functional Testing:

- **Bug-free product:** Functional testing ensures the delivery of a bug-free and high-quality product.
- **Customer satisfaction:** It ensures that all requirements are met and ensures that the customer is satisfied.
- **Testing focussed on specifications:** Functional testing is focussed on specifications as per customer usage.
- **Proper working of application:** This ensures that the application works as expected and ensures proper working of all the functionality of the application.
- **Improves quality of the product:** Functional testing ensures the security and safety of the product and improves the quality of the product.

2. Non-Functional Testing

Non-functional Testing is a type of Software Testing that is performed to verify the non- functional requirements of the application. It verifies whether the behavior of the system is as per the requirement or not. It tests all the aspects that are not tested in functional testing. Non-functional testing is a software testing technique that checks the non-functional attributes of the system. Non-functional testing is defined as a type of software testing to check non-functional aspects of a software application. It is designed to test the readiness of a system as per nonfunctional parameters which are never addressed by functional testing. Non-functional testing is as important as functional testing.

Benefits of Non-functional Testing

- **Improved performance:** Non-functional testing checks the performance of the system and determines the performance bottlenecks that can affect the performance.
- **Less time-consuming:** Non-functional testing is overall less time-consuming than the other testing process.
- **Improves user experience:** Non-functional testing like Usability testing checks how easily usable and user-friendly the software is for the users. Thus, focus on improving the overall user experience for the application.
- **More secure product:** As non-functional testing specifically includes security testing that checks the security bottlenecks of the application and how secure is the application against attacks from internal and external sources.

8.1 TESTING CASE 1 (Login)

8.1.1 Functional Test Cases-

- Verify if a user will be able to login with a valid username and valid password.
- Verify if a user cannot login with a valid username and an invalid password.
- Verify the login page for both, when the field is blank and Submit button isclicked.
- Verify the messages for invalid login.

8.1.2 Non-Functional Security Test Cases-

- Verify the time out functionality of the login session.
- Verify the login page by pressing ‘Back button’ of the browser. It should not allow you to enter the system once you log out.
- Verify if a user should not be allowed to log in with different credentials from the same browser at the same time.

8.2 TESTING CASE 2 (Adding Recipes)

8.2.1 Functional Test Cases-

- Verify that the all required fields are filled.
- Verify that the ingredients are properly field with commas.
- Verify that the adding recipe will update to the list.
- Verify that the adding recipe will be available for the only registered user or not.
- Verify that the only registered user can edit or delete the save recipe.

8.2.2 Non-Functional Security Test Cases-

- Verify that the all fields are visible to the user.
- Verify that the "nav bar" should be visible for direct reach to the "All recipes" tab for show updated recipe.

8.3 TESTING CASE 3 (Logout)

8.3.1 Functional Test Cases-

- Verify After successful login in Pantry2Plate, click on the profile icon to check logout button is visible or not.
- Verify by Clicking on the sign-out button without an internet connection and reconnecting to the internet to check if it's properly logout or not.
- Verify by clicking on the logout button, after successful logout on the login screen press the back button.
- Verify, login into more than two browser or mobiles and log out from anyone them and check all other account is properly working or all get logout.
- Verify after logout tries to re-login with the same or different account it'sallowing or not.

8.3.2 Non-Functional Security Test Cases-

- Verify the logs for the login and logout sessions.
- Verify if the logs contain multiple IPs for a single ID at the same time.
- Verify if the logs contain a denial-of-service attack for the login or logout.
- Verify if the log has suspicious activity.

CHEPTER 9

IMPILENTATION

9.1 Home Page

```
<nav class="navbar navbar-expand-lg sticky-top bg-body-tertiary" data-bs-theme="dark" >
  <div class="container-fluid">
    <a class="navbar-brand nav-link" href="/"><img class="logo"
      src="/images/logo.png"
      alt="P2P">Pantry2Plate</a>
    <button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-
target="#navbarNavAltMarkup"
      aria-controls="navbarNavAltMarkup" aria-expanded="false" aria-label="Toggle
navigation">
      <span class="navbar-toggler-icon"></span>
    </button>
    <div class="collapse navbar-collapse" id="navbarNavAltMarkup">
      <div class="navbar-nav">
        <a class="nav-link" aria-current="page" href="/">Home</a>
        <a class="nav-link" href="/recipes">Recipes</a>
        <a class="nav-link" href="/recipes/new">New</a>
        <a class="nav-link" href="http://localhost:3000">Serach</a>
        <a class="nav-link" href="/aboutUs">About Us</a>
      </div>
      <div class="navbar-nav ms-auto">
        <% if(!currentUser) {%>
        <a class="nav-link" href="/login">Login</a>
        <a class="nav-link" href="/register">Register</a>
        <% }  else { %>
```

```
                <a class="nav-link" href="/logout">Logout</a>
                <% } %>
            </div>
        </div>
    </div>
</nav>
```

9.2 Login Page

```
<% layout('layouts/boilerplate') %>
    <div class="login_bg">
        <img src="/images/2.jpg" alt="">
    </div>
    <div class="col-6 offset-3">
        <form action="/login" method="POST" class="validated-form login_from"
novalidate>

            <div class="mainContainer_login">
                <div class="login_headings">
                    <h3>Login</h3>
                    <p class="text">Sign in with your username and password</p>
                </div>

                <div class="mb-3">
                    <label class="form-label" for="username">Username</label>
                    <input class="form-control" type="text" id="username" name="username"
required>
                    <div class="valid-feedback">
                        Looks good!
                    </div>
                </div>
                <div class="mb-3">
                    <label class="form-label" for="password">Password</label>
                    <input class="form-control" type="password" id="password"
name="password" required>
                    <div class="valid-feedback">
                        Looks good!
                    </div>
                </div>
                <button class="btn btn-success login_btn">Login</button>
                <p class="logins_register">Not a member? <a class="linkhover"
href="/register">Register here!</a></p>
            </div>
        </form>
    </div>
```

9.3 Register Page

```
<% layout('layouts/boilerplate') %>


  <div class="register_bg">
    <img src="/images/2.jpg" alt="">
  </div>


  <div class="col-6 offset-3">
    <form action="/register" method="POST" class="validated-form register_from"
novalidate>
        <div class="mainContainer_register">


            <div class="register_headings">
                <h3>Register</h3>
            </div>


            <div class="mb-3">
                <label class="form-label" for="username">Username</label>
                <input class="form-control" type="text" id="username" name="username"
required>
```

```
                <div class="valid-feedback">
                    Looks good!
                </div>
            </div>
            <div class="mb-3">
                <label class="form-label" for="email">E-mail</label>
                <input class="form-control" type="email" id="email" name="email"
required>
                <div class="valid-feedback">
                    Looks good!
                </div>
            </div>
            <div class="mb-3">
                <label class="form-label" for="password">Password</label>
                <input class="form-control" type="password" id="password"
name="password" required>
                <div class="valid-feedback">
                    Looks good!
                </div>
            </div>
            <button class="register_btn btn btn-success">Register</button>


            <p class="registers_login">Have an Account ? <a class="linkhover"
href="/login">Login Here</a></p>
        </div>
    </form>
</div>
```

9.4 Adding New Recipe

```
<% layout('layouts/boilerplate')%>

  <div class="row">
    <h1 class="text-center">New Recipe</h1>
    <div class="col-6 offset-3">
      <form action="/recipes" method="POST" class="validated-form" novalidate>

        <!-- NAME -->
        <div class="mb-3">
          <label class="form-label" for="name">Name</label>
          <input class="form-control" type="text" id="name" name="recipe[name]"
required>
          <div class="valid-feedback">
            Looks good!
          </div>
        </div>

        <!-- TIME -->
        <div class="mb-3">
          <label class="form-label" for="time">Time</label>
          <input class="form-control" type="text" id="time" name="recipe[time]"
required>
          <div class="valid-feedback">
            Looks good!
          </div>
        </div>

        <!-- IMAGE URL -->
        <div class="mb-3">
          <label class="form-label" for="img">Image URL</label>
          <input class="form-control" type="text" id="img" name="recipe[img]"
required>
```

```
            <div class="valid-feedback">
                Looks good!
            </div>
        </div>


        <!-- METHOD -->
        <div class="mb-3">
            <label class="form-label" for="method">Method</label>
            <input class="form-control" type="text" id="method" name="recipe[method]"
required>
            <div class="valid-feedback">
                Looks good!
            </div>
        </div>


        <!-- INGREDIENT -->
        <div class="mb-3">
            <label class="form-label" for="ingredient">Ingredient</label>
            <input class="form-control" type="text" id="ingredient"
name="recipe[ingredient]" required>
            <div class="valid-feedback">
                Looks good!
            </div>
        </div>
        <!-- ------------------------------------------------------------------------ -->
        <!-- <div class="mb-3">
            <label class="form-label" for="ingredient1">Ingredient1</label>
            <input class="form-control" type="text" id="ingredient1"
name="recipe[ingredient1]" required>
            <div class="valid-feedback">
                Looks good!
            </div>
        </div>
```

```
            <div class="mb-3">
              <label class="form-label" for="ingredient2">Ingredient2</label>
              <input class="form-control" type="text" id="ingredient2"
name="recipe[ingredient2]" required>
              <div class="valid-feedback">
                Looks good!
              </div>
            </div>
            <div class="mb-3">
              <label class="form-label" for="ingredient3">Ingredient3</label>
              <input class="form-control" type="text" id="ingredient3"
name="recipe[ingredient3]" required>
              <div class="valid-feedback">
                Looks good!
              </div>
            </div>
            <div class="mb-3">
              <label class="form-label" for="ingredient4">Ingredient4</label>
              <input class="form-control" type="text" id="ingredient4"
name="recipe[ingredient4]" >
              <div class="valid-feedback">
                Looks good!
              </div>
            </div>
            <div class="mb-3">
              <label class="form-label" for="ingredient5">Ingredient5</label>
              <input class="form-control" type="text" id="ingredient5"
name="recipe[ingredient5]" >
              <div class="valid-feedback">
                Looks good!
              </div>
            </div> -->
            <!-- DESCRIPTION -->
            <div class="mb-3">
                <label class="form-label" for="description">Description</label>
                <textarea class="form-control" type="text" id="description"
name="recipe[description]" required>
                </textarea>
            </div>


            <!-- BUTTON -->
            <div class="mb-3">
                <button class="btn btn-success">Add Recipe</button>
            </div>
        </form>
    </div>
    <a href="/recipes" class="text-decoration-none">All Recipes</a>
</div>
```

9.5 Nave bar

```
<nav class="navbar navbar-expand-lg sticky-top bg-body-tertiary" data-bs-theme="dark" >
  <div class="container-fluid">
    <a class="navbar-brand nav-link" href="/"><img class="logo"
        src="/images/logo.png"
        alt="P2P">Pantry2Plate</a>
    <button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-
target="#navbarNavAltMarkup"
        aria-controls="navbarNavAltMarkup" aria-expanded="false" aria-label="Toggle
navigation">
        <span class="navbar-toggler-icon"></span>
    </button>
    <div class="collapse navbar-collapse" id="navbarNavAltMarkup">
        <div class="navbar-nav">
            <a class="nav-link" aria-current="page" href="/">Home</a>
            <a class="nav-link" href="/recipes">Recipes</a>
            <a class="nav-link" href="/recipes/new">New</a>
```

```
          <a class="nav-link" href="http://localhost:3000">Serach</a>
          <a class="nav-link" href="/aboutUs">About Us</a>
        </div>
        <div class="navbar-nav ms-auto">
          <% if(!currentUser) {%>
          <a class="nav-link" href="/login">Login</a>
          <a class="nav-link" href="/register">Register</a>
          <% }  else { %>
          <a class="nav-link" href="/logout">Logout</a>
          <% } %>
        </div>
      </div>
  </div>
</nav>
```

9.6 Data Base Connection

```
 const mongoose = require('mongoose');

const details = require('./all_recipes.json');

const Recipe = require('../models/recipe');

mongoose.connect('mongodb://localhost:27017/p2p-recipe')
mongoose.connect('mongodb+srv://pratyushsingh15393:<reciperest>@reciperest.n1l0iu1.
mongodb.net/')
const db = mongoose.connection;

db.on("connected", () => {
  console.log("Connected to database sucessfully");
});
db.on("error", (err) => {
  console.log("Error while connecting to database :" + err);
});
db.on("disconnected", () => {
   console.log("Mongodb connection disconnected");
});
db.once("open", () => {
   console.log("Database connected!");
});

const recipeDB = async() => {
   await Recipe.deleteMany({});
   for(let i=0; i< 13; i++){
      const reci = new Recipe({
         name: `${details[i].name}`,
         time: `${details[i].time}`,
         img: `${details[i].img}`,
         method: `${details[i].method}`,
         ingredient: details[i].ingredient,
         description: `${details[i].description}`
      })
      await reci.save();
   }
}

recipeDB().then(() => {
   mongoose.connection.close();
})
```

CHEPTER 10

BIBLIOGRAPHY

- Node.js v20.9.0 and express 4.18.2 – Modern Cross-Platform Development: Build applications with express JS, node JS web development framework, MongoDB 7.0.2 and using VS Code 1.85.1.
- C. J. Date, A. Kannan, and S. Swaminathan, An Introduction to DatabaseSystems, Pearson Education, Eighth Edition, 2009.
- Abraham Silber Schatz, Henry F. Korte and S. Sudarshan, Database SystemConcepts, McGraw-Hill Education (Asia), Fifth Edition, 2006.
- Pressmen, Somerville, Software Engineering: Design, Implementation, andManagement.

APPLICATIONS

- www.google.com
- www.stackoverflow.com
- www.w3school.com
- www.udemy.com
- www.gitHub.com
- www.youtube.com
- https://unsplash.com/
- https://hoppscotch.io/