

**NIRMALA MEMORIAL FOUNDATION COLLEGE OF
COMMERCE AND SCIENCE**

Kandivali (East), Mumbai-400 101.

(Affiliated to University of Mumbai)



Food Recipe Finder

A Project Report

Submitted in partial fulfillment of the
Requirements for the award of the Degree of (TYBSC-IT)

By

Choudhary Nirmeet Ramdhani

Seat no.:3029044

Semester: VI

Under the esteemed guidance of

Mrs. Vandana Singh

Designation: Assistant Professor

BACHELOR OF SCIENCE (INFORMATION TECHNOLOGY)

Academic Year 2024-2025

NIRMALA MEMORIAL FOUNDATION COLLEGE OF COMMERCE AND SCIENCE

Kandivali (East), Mumbai-400 101.

(Affiliated to University of Mumbai)

MUMBAI, 400 101

MAHARASHTRA

DEPARTMENT OF INFORMATION TECHNOLOGY



CERTIFICATE

This is to certify that the project entitled, "**Food Recipe Finder**", is bonafied work of
“**Choudhary Nirmeet Ramdhani**” bearing Roll.No: “A/18” submitted in partial fulfillment of
the requirements for the award of degree of BACHELOR OF SCIENCE in INFORMATION
TECHNOLOGY from University of Mumbai.

Internal Guide

Coordinator

External Examiner

Date:

College Seal

PROFORMA FOR THE APPROVAL PROJECT PROPOSAL

PNR No.: 202201640000634

Roll no: 18

1. Name of the Student

Choudhary Nirmeet Ramdhani

2. Title of the Project

Food Recipe Finder

3. Name of the Guide

Mrs Vanadana Singh

4. Teaching experience of the Guide 10 years

5. Is this your first submission?

Yes

☐

No

☐

Signature of the Student

Signature of the Guide

Date:

Date:

Signature of the coordinator

Date:

Abstract

The “Food Recipe Finder” is a web application designed to provide a modern and comprehensive platform for discovering and exploring a wide variety of recipes. It caters to individuals seeking quick and convenient access to recipes that meet their culinary needs and dietary preferences. The application enables users to search for recipes based on specific ingredients or dish names and provides detailed information, including step-by-step cooking instructions, ingredient lists, and nutritional details.

The primary objective of this project is to create a user-friendly platform that enhances the experience of meal preparation by leveraging modern web technologies and robust APIs. Users will have the ability to save their favorite recipes, create custom collections, and filter recipes based on dietary restrictions. The application also promotes community engagement through recipe ratings, reviews, and sharing options.

By providing a mobile-compatible and visually appealing interface, the “Food Recipe Finder” aims to become a one-stop destination for cooking enthusiasts, offering a diverse range of recipes accessible on any device. The project expects to deliver a robust web application that simplifies the cooking experience and encourages users to explore new culinary options.

ACKNOWLEDGEMENT

This is a sincere effort from me to express my heartfelt gratitude in creating the “Food Recipes Finder” which is a desktop application.

I would like to thank Prof. Vandana Singh for her help, motivation towards the project. Her guidance has helped me at the time of research and writing of thesis, continuous support, enthusiasm and immense knowledge has given a boost for the successful fulfilment of this project. I take this opportunity to express my profound gratitude to management of Nirmala Memorial Foundation College of Commerce & Science and the entire Information Technology Department.

I would like to thank all the teaching and non teaching staff who have directly or indirectly helped in the successful creation of this project. I would like to thank my friends and family for helping me, encouraging me which led to the accomplishment of the project within the time frame.

This project has helped me gain a lot of practical experience which will be beneficial to me in the future.

Choudhary Nirmeet Ramdhani

TYIT A

DECLARATION BY LEARNER

I, the undersigned Mr. Choudhary Nirmeet Ramdhani hereby declare that the work embodied in this project work titled “Smart Parking System” forms from my own contribution to the research work carried out under the guidance of Ms. Vandana Singh is a result of my own research work and has not been submitted to any other University for any other Degree/Diploma to this or to any other University.

Wherever reference has been made to previous work of others, it has been clearly indicated as such and included in the Bibliography.

Hereby further declare that all the information of this document has been obtained and presented in accordance with academic rules and ethical conduct.

(Signature of the Candidate)

Choudhary Nirmeet Ramdhani

Index

CHAPTER 1: INTRODUCTION	1
1.1 BACKGROUND	1
1.2 OBJECTIVE.....	1
1.3 PURPOSE	2
1.4 SCOPE.....	2
1.5 ADVANTAGE.....	3
1.6 FRONT END	3
1.7 BACK END	5
CHAPTER 2: SURVEY OF TECHNOLOGIES	6
2.1 SURVEY OF TECHNOLOGIES	6
2.1.1 Existing System	6
2.1.2 Limitations of Existing Systems	6
2.1.3 Advantages Over Previous Projects	7
2.2 Feasibility Study	8
2.3 LIST OF TECHNOLOGIES	9
2.4 Selected Technologies.....	12
CHAPTER 3: REQUIREMENTS AND ANALYSIS.....	13
3.1 Problem Definition	13
3.2 Requirements Specification	13
3.3 GANTT CHART	16
3.4 Software and Hardware Requirements	17
3.5 Preliminary Product Description	17
CHAPTER 4: SYSTEM DESIGN	18
4.1 UML.....	18

4.2 Use Case Diagram.....	19
4.2.1 Activity diagram	23
4.2.1 Sequence Design	25
CHAPTER 5: IMPLEMENTATION AND TESTING	27
5.1 Implementation Approach	27
5.2 Coding	30
5.3 Testing	36
5.3.1 UNIT TESTING	37
5.3.2 INTEGRATION TESTING	37
5.3.3 Black box testing.....	38
5.3.4 WHITE BOX TESTING.....	41
5.3.5 TOP DOWN APPROACH.....	43
5.3.6 BOTTOM UP APPROACH	44
5.4 SYSTEM TESTING	45
5.5 FUNCTIONAL TESTING	45
5.6 COMPATIBILITY TESTING	46
5.7 TEST CASES.....	48
CHAPTER 6: USER DOCUMENTATION.....	49
CHAPTER 7: CONCLUSION AND FUTURE SCOPE.....	52
7.1 Conclusion.....	52
7.2 Limitations of the System.....	52
7.3 Future Scope.....	52

CHAPTER 1: INTRODUCTION

1.1 BACKGROUND

The food industry has rapidly evolved due to technological advancements. In this era of digitization, food recipe applications have become increasingly popular. People no longer rely solely on cookbooks but instead, seek digital solutions to explore, create, and experiment with new recipes. Such applications cater to diverse user preferences, providing customized recipe suggestions based on ingredients, dietary preferences, and nutrition information.

This project is based on developing a Food Recipe Web Application that utilizes the Spoonacular API. The application serves as a repository of various recipes, where users can search for food recipes based on their ingredients, view detailed recipe information, and save their favorite recipes. Additionally, the web app provides a user-friendly interface with modals and search capabilities. The Spoonacular API enhances the application's functionality by providing access to a vast database of recipes, nutritional details, and cooking instructions.

The application targets food enthusiasts who wish to discover new recipes, monitor their nutritional intake, and conveniently store their favorite meals. Users can interact with the system by inputting recipe queries and receiving relevant search results displayed with images, ingredients, and step-by-step instructions. It simplifies recipe discovery, saving users time and effort by consolidating diverse food choices in one place.

1.2 OBJECTIVE

The objectives of the Food Recipe Web Application are as follows:

Ease of Recipe Discovery: Provide users with a seamless experience of finding recipes by ingredients or dish names, utilizing an extensive recipe database.

Nutritional Awareness: Offer detailed nutritional information for each recipe to help users make informed choices based on their dietary preferences and needs.

Favorite Recipe Storage: Enable users to mark and store recipes as favorites for future reference.

User-Friendly Interface: Develop an easy-to-navigate user interface with clear recipe instructions, images, and modals for viewing recipe details.

Responsive Design: Ensure the application is responsive across different devices, including desktops, tablets, and smartphones.

1.3 PURPOSE

The primary purpose of this project is to build a web-based platform where users can search for food recipes and view essential details, such as ingredients, nutrition facts, and preparation steps. By using the Spoonacular API, the application leverages a large database of recipes and related information to offer users an enriching experience. This project aims to simplify meal planning for users while considering their dietary restrictions and preferences.

1.4 SCOPE

The scope of the project includes:

Recipe Search Functionality: Users can search for recipes by entering keywords (e.g., "chicken") to find dishes based on specific ingredients or names.

Recipe Details: Each recipe result will include ingredients, instructions, a food image, and nutritional information.

Favorite Functionality: Users can mark recipes as favorites for easy access later.

Responsive Design: The web application will be optimized for various devices to ensure usability across platforms.

Security: Users' data, especially favorites, will be securely stored to maintain privacy and provide a personalized experience.

1.5 ADVANTAGE

The application is relevant in various contexts, including:

Home Cooking: Home chefs who wish to experiment with new dishes can easily find and prepare meals based on ingredients they already have.

Health-Conscious Users: For users concerned about their nutritional intake, the app will offer detailed nutritional breakdowns of each recipe.

Meal Planning: Users who want to plan their meals for the week can explore different recipes, save them to favorites, and have easy access to them when needed.

1.6 FRONT END

HTML5



HTML5 (Hypertext Markup Language 5) is a markup language used for structuring and presenting hypertext documents on the World wide web. It was the fifth and final major HTML version that is now a retired World wide web Consortium (W3C) recommendation. The current specification is known as the HTML Living Standard

CSS



Cascading Style Sheets (CSS) is a style sheet language used for specifying the presentation and styling of a document written in a markup language such as HTML or XML (including XML dialects such as SVG, MathML or XHTML). CSS is a cornerstone technology of the World Wide Web, alongside HTML and JavaScript.

Javascript



JavaScript is a high-level programming language that follows the ECMAScript standard. It was originally designed as a scripting language for websites but became widely adopted as a general-purpose programming language, and is currently the most popular programming language in use.[1] JavaScript is usually found running in a web browser as interactive or automated content, ranging from popup messages and live clocks to large web applications. JavaScript is also commonly used in server-side programming through platforms like Node.js

1.7 BACK END

PHP



PHP (Hypertext Preprocessor):

PHP is a powerful server-side scripting language designed for web development. It enables developers to create dynamic and interactive web pages quickly. PHP code runs on the server, generating HTML content that is then sent to the user's browser. It seamlessly integrates with databases (such as MySQL) to handle data storage and retrieval.

MySQL



MySQL is a widely-used relational database management system. It stores and manages structured data efficiently. MySQL is ideal for web applications, supporting both small and large-scale projects. It uses standard SQL for querying and manipulation. MySQL is fast, reliable, and free to use.

CHAPTER 2:

SYSTEM ANALYSIS

2.1 SURVEY OF TECHNOLOGIES

2.1.1 Existing System

There are numerous food recipe applications available in the market today, both in the form of web and mobile applications. These existing systems allow users to search for recipes, view cooking instructions, ingredients, and sometimes nutritional information. Popular existing platforms include **AllRecipes**, **Yummly**, and **BBC Good Food**. Many of these platforms leverage large databases of recipes to provide a diverse set of meal options to users.

Most of these existing systems offer basic features such as:

- Searching for recipes based on ingredients or meal types.
- Displaying cooking instructions and ingredients.
- Allowing users to save recipes as favorites or bookmarks.
- Offering a mobile-friendly or responsive web design.

While these systems are popular, they often lack personalized or interactive elements, such as nutritional breakdowns, the ability to tailor recommendations based on dietary restrictions, and a modern, user-friendly interface with enhanced search functionality.

2.1.2 Limitations of Existing Systems

Despite the popularity of existing recipe platforms, they come with certain limitations:

- **Scattered and Inconsistent Information:** Many of the existing systems are not consistent in providing nutritional details for each recipe, leaving users uninformed about the dietary implications of the meals they prepare.
- **Limited Integration with Other Platforms:** Many applications do not integrate seamlessly with external APIs to fetch updated, diverse recipes from multiple sources.
- **Complex User Interfaces:** Some platforms have cluttered interfaces with too many features that overwhelm the user, making recipe discovery less intuitive.

- **Lack of Personalization:** Existing systems do not provide enough customization options, such as filtering recipes by dietary preferences (e.g., vegan, gluten-free) or tracking previously made meals.
- **Difficulty in Saving Recipes:** In some platforms, users can only bookmark recipes on their devices, which becomes inconvenient if they want to access them across multiple devices.
- **Poor Search Functionality:** Users are often limited to searching for recipes by name and not by specific ingredients or dietary requirements, which can limit the usefulness of the application.

2.1.3 Advantages Over Previous Projects

The **Food Recipe Web Application** developed as part of this project introduces several advantages over existing recipe systems:

- **API Integration:** By integrating the **Spoonacular API**, the application provides a vast database of recipes, ingredients, and nutritional information. This ensures that users have access to a diverse set of recipes.
- **Enhanced Search Features:** Users can search for recipes not just by meal names but also by ingredients, dietary restrictions, and nutritional preferences, offering more flexibility.
- **Detailed Nutritional Information:** The application provides users with accurate nutritional data for each recipe, including macronutrients, vitamins, and caloric information. This feature is crucial for health-conscious users.
- **Favorites List:** Users can securely save their favorite recipes to their profiles, which can be accessed across devices, thanks to the system's login functionality.
- **Clean, User-Friendly Interface:** The application focuses on a minimalistic, intuitive design, ensuring that users can quickly find and access the information they need without overwhelming them with unnecessary features.
- **Responsive Design:** The application is fully responsive, making it easy to access and use on both desktop and mobile devices.

2.2 Feasibility Study

A feasibility study evaluates whether the project is viable in terms of technology, operations, and cost. The **Food Recipe Web Application** passed through the following feasibility analyses:

Types of Feasibility Study:

- **Technical Feasibility:** The project's technical feasibility revolves around the use of widely supported technologies like **HTML**, **CSS**, **JavaScript**, **PHP**, and **MySQL**. The chosen **Spoonacular API** is reliable and well-documented, ensuring smooth integration with the application. Furthermore, the tools required for development, such as **XAMPP** for local hosting, are readily available and easy to use, making the project technically feasible.
- **Operational Feasibility:** The application addresses the problems identified in the existing systems and provides improved functionality. Users can easily search for recipes, view details, and save their favorite recipes. The user-friendly interface ensures that both tech-savvy and less experienced users can efficiently navigate the system. As the application is hosted on the web, it can be accessed from anywhere, making it operationally viable.
- **Economic Feasibility:** From a cost perspective, the project is highly feasible. The primary costs involve development tools (which are largely free), domain hosting, and potential API usage fees. However, most of the tools used in this project are open-source or come with free plans that support the initial development phase. As the project grows, the cost of hosting and API usage may increase, but these costs can be managed through subscription models or ads, making the project economically sustainable.

2.3 LIST OF TECHNOLOGIES

HTML5



HTML5 (Hypertext Markup Language 5) is a markup language used for structuring and presenting hypertext documents on the World wide web. It was the fifth and final major HTML version that is now a retired World wide web Consortium (W3C) recommendation. The current specification is known as the HTML Living Standard

CSS



Cascading Style Sheets (CSS) is a style sheet language used for specifying the presentation and styling of a document written in a markup language such as HTML or XML (including XML dialects such as SVG, MathML or XHTML). CSS is a cornerstone technology of the World Wide Web, alongside HTML and JavaScript.

Javascript



JavaScript is a high-level programming language that follows the ECMAScript standard. It was originally designed as a scripting language for websites but became widely adopted as a general-purpose programming language, and is currently the most popular programming language in use.[1] JavaScript is usually found running in a web browser as interactive or automated content, ranging from popup messages and live clocks to large web applications. JavaScript is also commonly used in server-side programming through platforms like Node.js

BACK END

PHP



PHP (Hypertext Preprocessor):

PHP is a powerful server-side scripting language designed for web development. It enables developers to create dynamic and interactive web pages quickly. PHP code runs on the server, generating HTML content that is then sent to the user's browser. It seamlessly integrates with databases (such as MySQL) to handle data storage and retrieval.

MySQL



MySQL is a widely-used relational database management system. It stores and manages structured data efficiently. MySQL is ideal for web applications, supporting both small and large-scale projects. It uses standard SQL for querying and manipulation. MySQL is fast, reliable, and free to use.

Technology	Purpose	Role in the project
HTML5	Frontend	Structure the content of the webpages
CSS3	Frontend	Style the webpages and ensure responsive design
Javascript	Frontend	Handle interactivity and dynamic content
PHP	Backend	Process and handle user requests, communicate with the database
MySQL	Database	Store user data such as favorite recipes and login credentials
Spoonacular API	External API	Provide access to recipe data, ingredients, nutrition information
XAMPP	Development Environment	Local web server for testing and deployment of the web application

2.4 Selected Technologies

After evaluating various technology options, the following technologies were selected for the project due to their compatibility, performance, and ease of use:

- **Frontend:** The combination of **HTML5**, **CSS3**, and **JavaScript** was chosen for building the web pages because these technologies provide flexibility and allow for the creation of responsive, interactive interfaces. Additionally, **Bootstrap** was selected as the CSS framework to speed up development and ensure a responsive design that works well across different screen sizes.
- **Backend:** **PHP** was chosen for server-side programming because of its simplicity, open-source nature, and compatibility with **MySQL**, the selected database for storing user data. **PHP** also integrates well with **XAMPP**, which provides a local environment for testing the application before deployment.
- **Database:** **MySQL** was selected for storing user data because it is an open-source, relational database management system that is widely supported and efficient for handling structured data.
- **External API:** The **Spoonacular API** was selected for fetching recipe data, including ingredients, instructions, and nutritional information. This API offers a vast amount of structured data, ensuring that users have access to diverse recipes and dietary details.

CHAPTER 3: REQUIREMENTS AND ANALYSIS

3.1 Problem Definition

In today's fast-paced world, cooking has become a challenge for many, especially for those with limited time or cooking experience. Most individuals rely on web searches, apps, or social media to find recipes, which are often scattered across various platforms. As a result, users face challenges such as:

- Difficulty in finding the right recipes that fit their preferences or available ingredients.
- Lack of access to detailed nutritional information to make informed dietary decisions.
- Time-consuming process of bookmarking or saving recipes on multiple platforms.
- Inconsistent user interfaces that hinder a smooth user experience.

The Food Recipe Web Application aims to address these challenges by providing users with a unified platform to search for, view, and save recipes, all while offering nutritional insights. By integrating the Spoonacular API, the application provides a broad variety of recipes, ensuring that users find meals that match their dietary needs and preferences.

3.2 Requirements Specification

HTML5



HTML5 (Hypertext Markup Language 5) is a markup language used for structuring and presenting hypertext documents on the World wide web. It was the fifth and final major HTML version that is now a retired World wide web Consortium (W3C) recommendation. The current specification is known as the HTML Living Standard

CSS



Cascading Style Sheets (CSS) is a style sheet language used for specifying the presentation and styling of a document written in a markup language such as HTML or XML (including XML dialects such as SVG, MathML or XHTML). CSS is a cornerstone technology of the World Wide Web, alongside HTML and JavaScript.

Javascript



JavaScript is a high-level programming language that follows the ECMAScript standard. It was originally designed as a scripting language for websites but became widely adopted as a general-purpose programming language, and is currently the most popular programming language in use.[1] JavaScript is usually found running in a web browser as interactive or automated content, ranging from popup messages and live clocks to large web applications. JavaScript is also commonly used in server-side programming through platforms like Node.js

BACKED

PHP



PHP (Hypertext Preprocessor):

PHP is a powerful server-side scripting language designed for web development. It enables developers to create dynamic and interactive web pages quickly. PHP code runs on the server, generating HTML content that is then sent to the user's browser. It seamlessly integrates with databases (such as MySQL) to handle data storage and retrieval.

MySQL



MySQL is a widely-used relational database management system. It stores and manages structured data efficiently. MySQL is ideal for web applications, supporting both small and large-scale projects. It uses standard SQL for querying and manipulation. MySQL is fast, reliable, and free to use.

3.3 GANTT CHART

A Gantt chart is a project management tool that displays work done over time in connection to the time scheduled for the work. It usually consists of two sections: a left-side job list and a right-side timeline with plan bars that show work. The start and finish times of tasks, milestones, task dependencies, and assignees can all be included in the Gantt chart.

Today, plan tools are a common term used to describe Gantt chart tools. Roadmap tools like Jira Software have features like a flexible task structure and resource management sections to keep up with the demands of contemporary software development. In spite of the continuous character of the software development process, this aids teams in maintaining a consistent project strategy.

Task/Activity	June	July	August	September	October	November	December	January	February	March
Planning	X	X								
Requirements Analysis		X	X							
Design				X	X	X				
Coding/Implementation						X	X	X		
Testing									X	X

Timeline Details:

1. Planning: June 12 - July 15, 2024
2. Requirements Analysis: July 16 - August 20, 2024
3. Design: August 21 - November 30, 2024
4. Coding/Implementation: December 1, 2024 - February 15, 2025
5. Testing: February 16 - March 15, 2025

3.4 Software and Hardware Requirements

- Software Requirements:

- Operating System: Windows 10 or higher (for development), Android/iOS (for mobile testing).
- Development Tools: XAMPP for local hosting, Visual Studio Code for code editing.
- Technologies Used: HTML5, CSS3, JavaScript, PHP, MySQL.
- API: Spoonacular API for fetching recipe data.

- Hardware Requirements:

- Processor: Intel i5 or higher.
- RAM: Minimum 8GB.
- Storage: Minimum 256GB SSD (for local development).

3.5 Preliminary Product Description

The Food Recipe Web Application is an interactive platform designed to simplify the process of discovering new recipes based on users' preferences or available ingredients. It allows users to:

- Search for recipes and view detailed information including ingredients, instructions, and nutrition data.
- Mark recipes as favorites for easy access later.
- View results in a clean, scrollable modal interface for easy reading and navigation.

The application's core functionality is powered by the Spoonacular API, which supplies recipe data. The application includes login functionality, ensuring that users can store their favorite recipes securely.

CHAPTER 4:

SYSTEM

DESIGN

4. UML

UML is an acronym that stands for Unified Modeling Language. Simply put, UML is a modern approach to modeling and documenting software. In fact, it's one of the most popular business process modeling techniques.

It is based on diagrammatic representations of software components. As the old proverb says: "a picture is worth a thousand words". By using visual representations, we are able to better understand possible flaws or errors in software or business processes.

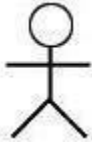
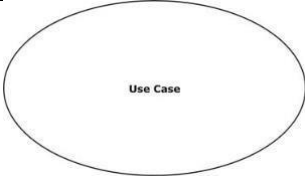


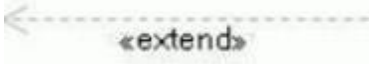
UML was created as a result of the chaos revolving around software development and documentation. In the 1990s, there were several different ways to represent and document software systems. The need arose for a more unified way to visually represent those systems and as a result, in 1994-1996, the UML was developed by three software engineers working at Rational Software. It was later adopted as the standard in 1997 and has remained the standard ever since, receiving only a few updates

4.1 Use Case Diagram

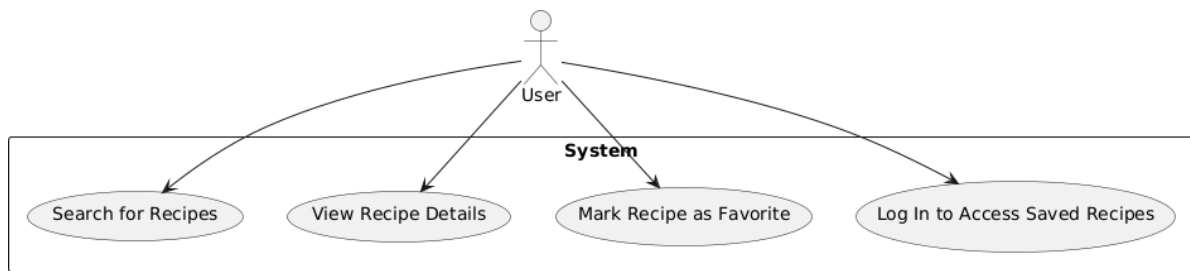
AUML use case diagram is the primary form of system/software requirements for a new software program underdeveloped. Use cases specify the expected behavior (what), and not the exact method of making it happen (how). Use cases once specified can be denoted both textual and visual representation (i.e. use case diagram). A key concept of use case modeling is that it helps us design a system from the end user's perspective. It is an effective technique for communicating system behavior in the user's terms by specifying all externally visible system behavior.

A use case diagram is usually simple. It does not show the detail of the use cases:

- It only summarizes some of the relationships between use cases, actors, and systems.
- It does not show the order in which steps are performed to achieve the goals of each use case.

Symbol	Name	Description
	Actor	An actor is external to a system, interacts with the system, may be a human user or another system, and has goals and responsibilities to satisfy in interacting with the system. Actors address the question of who and what interacts with a system.
	Use case	It provides a visual representation of how users interact with a system.
	Association	An association is a connection between an actor and a use case. An association indicates that an actor can carry out a use case. Several actors at one use case mean that each actor can carry out the use case on his or her own and not that the actors carry out the use case together.
	Include Relationships	An include relationship is a relationship between two use cases. It indicates that the use case to which the arrow points is included in the use case on the other side of the arrow.
	Extend Relationship	Extend relationship is to specify that one use case (extension) extends the behavior of another use case (base). This type of relationship reveals details about a system or application that are typically hidden in a use case.

Use Case Diagram with Respect to Project:



Activity Diagram

An activity diagram is a behavioral diagram i.e. it depicts the behavior of a system. An activity diagram portrays the controlflow from a start point to a finish point showing the various decision paths that exist while the activity is being executed.

➤ Symbols and Notations

❖ Action States

- **Definition:** An action state represents the execution of an action or a process within the system. It shows what activity is happening at a particular moment.
- **Representation:** In the diagram, action states are represented as rectangles with rounded corners.
 - Examples from the diagram:
 - "Display the list of recipes"
 - "User selects a recipe"
 - "Show ingredients"
 - "User logs out"

❖ Action Flow (Control Flow)

- **Definition:** Action flows or controlflows show the transition from one action to another, representing the sequence in which activities occur.
- **Representation:** Action flows are shown using arrows that connect the different action states.
 - Example from the diagram:

- The arrow between "User searches for a recipe" and "Recipe found?"

❖ Initial State

- **Definition:** The initial state shows where the activity begins in the diagram.
- **Representation:** This is represented by a small filled black circle, typically at the top of the diagram.
 - Example from the diagram:
 - The filled circle at the very top, indicating the start of the process: "User searches for a recipe".

❖ Final State

- **Definition:** The final state represents the completion of the activity.
- **Representation:** This is shown as a filled black circle nested inside another circle (like a bullseye).
 - Example from the diagram:
 - The final circle at the bottom of the diagram, which indicates the end of the process when the user logs out.

❖ Branching (Decision Points)

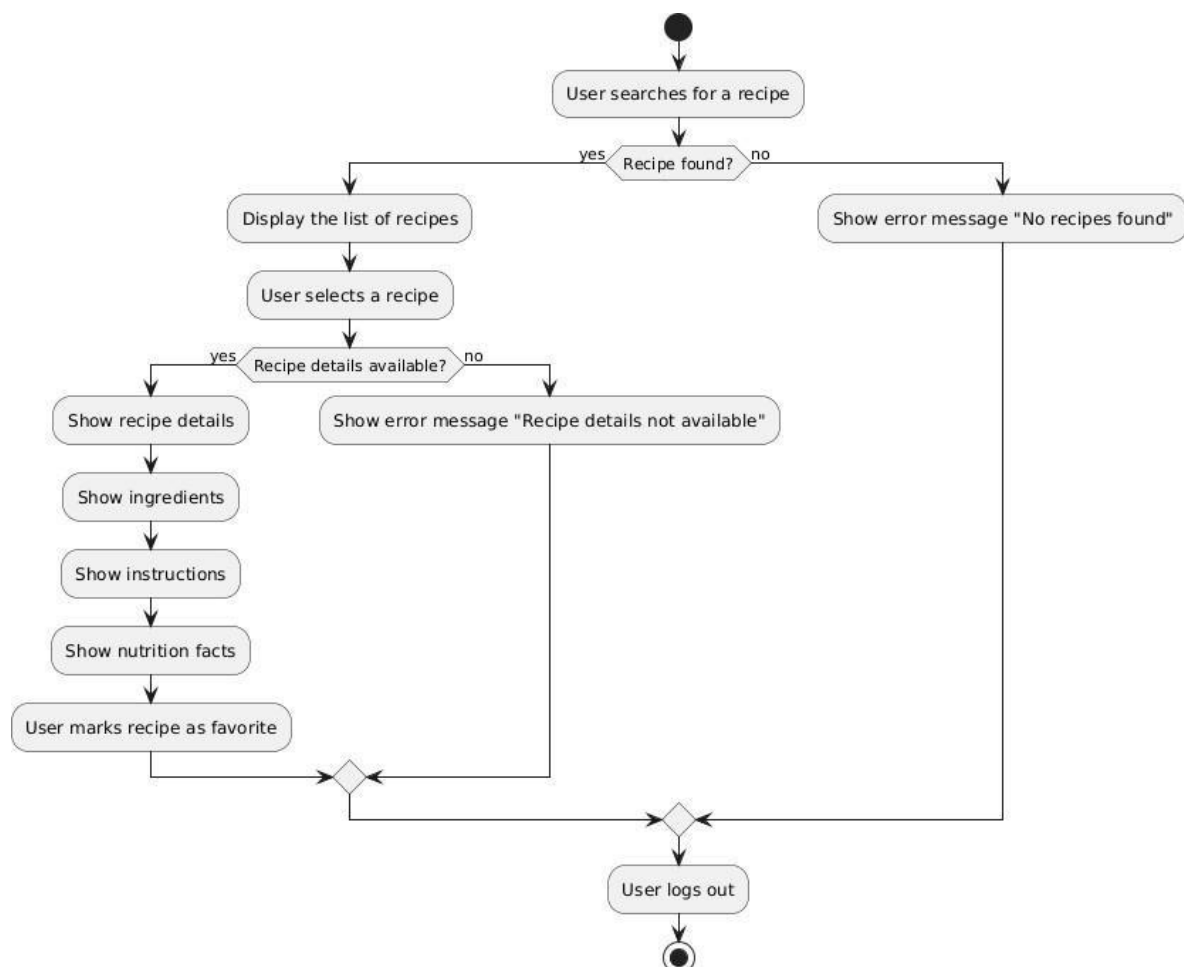
- **Definition:** A decision node (branch) allows for alternative paths in the activity flow, depending on a condition.
- **Representation:** This is shown using a diamond symbol, where the flow splits based on different conditions (e.g., yes/no, true/false).
 - Examples from the diagram:
 - The diamond for "Recipe found?" with two outgoing paths: "yes" and "no".
 - The diamond for "Recipe details available?" with paths for "yes" and "no".

❖ Synchronization (Fork and Join)

- **Fork Node:** A fork allows a single action flow to split into multiple concurrent flows.
 - **Representation:** It is shown as a horizontal or vertical thick line from which multiple flows branch out.

- **Join Node:** A join node combines multiple concurrent flows back into a single action.
 - **Representation:** It is also represented by a horizontal or vertical thick line.
 - Example from the diagram:
 - This would be represented if there were multiple concurrent processes occurring in the application, though your diagram does not appear to use a synchronization element.

Activity Diagram with respect to project



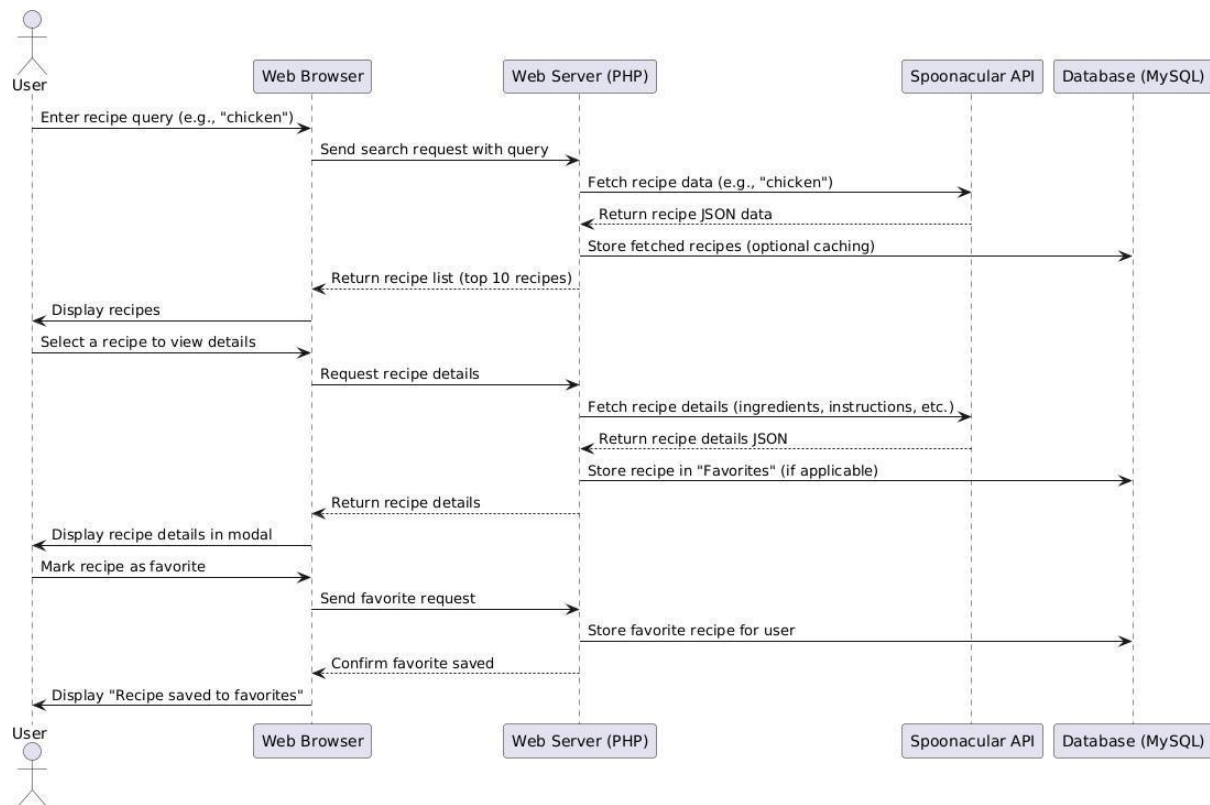
Flowchart: Recipe Search and Details

This image is a flowchart that illustrates the user interaction flow for searching and viewing recipes in a recipe application.

- Start: The user searches for a recipe.
- Decision: Recipe Found?
- If Yes, the list of recipes is displayed.
- If No, an error message “No recipes found” is shown.
- User Selects a Recipe: The user selects a recipe from the list.
- Decision: Recipe Details Available?
- If Yes, the app shows recipe details, ingredients, instructions, and nutrition facts.
- If No, an error message “Recipe details not available” is shown.
- User Marks Recipe as Favorite: If desired, the user can mark the recipe as a favorite.
- End: The user logs out of the application.

This flowchart provides a clear overview of the decision points and outcomes in the user interaction process.

4.1.1 Sequence Design



Sequence Diagram: Recipe Request and Response

The second image is a sequence diagram that shows the detailed interaction between the user, web browser, web server (PHP), Spoonacular API, and the database (MySQL) for recipe search and details retrieval.

User Actions:

- The user enters a query (e.g., "chicken") and sends a search request.
- The user selects a recipe from the displayed list to view details.
- The user can also mark a recipe as a favorite.

Web Browser:

- Sends search and request details to the web server (PHP).
- Displays recipe list and details received from the server.

Web Server (PHP):

- Sends a request to the Spoonacular API to fetch recipe data.
- Returns the list of top 10 recipes and displays the details in response to user actions.
- Handles storing of favorite recipes in the database if marked by the user.

Spoonacular API:

- Returns recipe JSON data and details upon request from the webserver.

Database (MySQL):

- Stores fetched recipe details for caching (optional).
- Stores favorite recipe details for users if applicable.

This sequence diagram demonstrates the communication flow and data exchanges between the user and various components of the recipe application.

Chapter 5

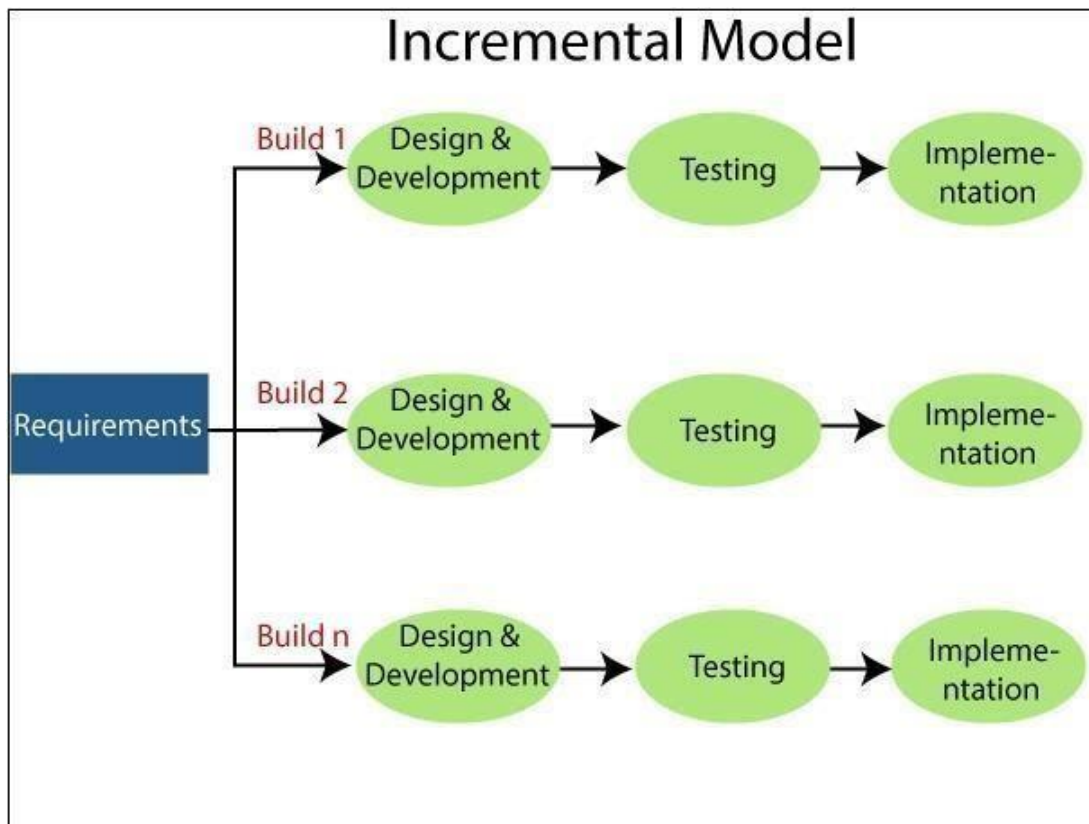
IMPLEMENTATION AND TESTING

5.1 Implementation Approach

The Incremental Model is a software development approach where the system is developed and delivered in small parts (increments). Each increment adds functionality to the system, and new builds are developed in multiple cycles until the final product is completed.

Incremental Model

The Incremental Model follows a structured approach where the software is built progressively in multiple iterations. Instead of delivering the entire system at once, it is divided into smaller, manageable modules. Each module goes through the following phases:



1. Requirement Analysis

- The requirements are gathered and analyzed at the start. However, instead of defining all requirements at once, only the requirements needed for the first increment are considered.
- The remaining requirements are analyzed progressively with each iteration.

2. Design & Development

- Each increment undergoes design and development, where new features or modules are coded and integrated into the system.
- The design process ensures that newly developed components can be easily merged with the existing system.

3. Testing

- Each increment is tested before integration to identify and fix any defects.
- Testing ensures that the newly developed functionality does not break existing features.
- Regression testing is performed to verify the stability of the overall system.

4. Implementation

- Once an increment passes testing, it is deployed for use.
- Implementation continues for multiple iterations until the final system is fully functional.

Advantages of the Incremental Model

- Early Delivery of Functional Software: The core functionalities can be delivered early, providing users with a working version of the software.
- Flexibility in Requirement Changes: Since the development is iterative, changes in requirements can be incorporated in later increments.

- Easier Debugging & Testing: Each increment is tested separately, making it easier to identify and fix defects.
- User Feedback Incorporation: Users can test early builds and provide feedback, which helps improve future increments.
- Lower Risk: Problems can be detected early, reducing the risk of major failures at later stages.

Disadvantages of the Incremental Model

- Requires Proper Planning: Each increment needs proper planning to ensure seamless integration.
- Higher Cost: Since testing is done in every increment, it may increase overall costs.
- Dependency on Previous Increments: If an earlier increment has issues, it may affect subsequent increments.
- Not Suitable for Complex Systems: Large-scale projects with interdependent components may struggle with incremental development.

5.2 coding

index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Food Race in India</title>
  <!-- Google Fonts -->
  <link href="https://fonts.googleapis.com/css2?family=Poppins:wght@400;600&display=swap"
rel="stylesheet">
  <!-- Font Awesome for Icons -->
  <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/font-
awesome/6.0.0/css/all.min.css" integrity="sha512-..." crossorigin="anonymous"
referrerpolicy="no-referrer" />
  <!-- Custom CSS -->
  <link rel="stylesheet" href="styles.css">
</head>
<body>
  <header>
    <div class="navbar">
      </a>
      <a href="about.html">About</a>
      <a href="contact.html">Contact</a>
      <a href="service.html">Service</a>
      <a href="index.html">Home</a>
    </div>
  </header>

  <br><br><br>
  <br><br><br><br>
  <div class="search-bar">
    <input type="text" id="food-input" placeholder="Search for a recipe...">
    <button id="search-button"><i class="fas fa-search"></i></button>
  </div>
```

```
<button onclick="window.location.href='favorites.html'">Favorites</button>

<main>
  <div id="loader" class="loader hidden"></div>
  <div id="results" class="results"></div>
</main>

<!-- Recipe Details Modal -->
<div id="recipe-modal" class="modal hidden">
  <div class="modal-content">
    <span class="close-button">&times;</span>
    <div id="recipe-details"></div>
  </div>
</div>

<!-- <footer>
  <p>&copy; 2023 Food Race in India. All rights reserved.</p>
</footer> -->

<!-- Custom JS -->
<script src="script.js"></script>
</body>
</html>
```

script.js

```
document.getElementById('search-button').addEventListener('click', function() { const
  query = document.getElementById('food-input').value;
  if (query) { searchRecipes(query);
  }
});

function searchRecipes(query) {
  const apiKey = 'a9fffb156669410ca87e41c585f0999a'; const url
  =
  `https://api.spoonacular.com/recipes/complexSearch?apiKey=${apiKey}&query=${query}
  &number=10&addRecipeInformation=true`;

  fetch(url)
    .then(response => response.json())
    .then(data => { displayResults(data.results);
    })
    .catch(error => { console.error('Error:',
      error);
    });
}

function displayResults(results) {
  const resultsDiv = document.getElementById('results');
  resultsDiv.innerHTML = "";

  results.forEach(recipe => {
    const recipeCard = document.createElement('div');
    recipeCard.classList.add('recipe-card');

    const recipeImage = document.createElement('img');
    recipeImage.src = recipe.image;
    recipeCard.appendChild(recipeImage);

    const recipeTitle = document.createElement('h3');
    recipeTitle.textContent = recipe.title;
    recipeCard.appendChild(recipeTitle);
```



```

const viewRecipeButton = document.createElement('button');
viewRecipeButton.textContent = 'View Recipe';
viewRecipeButton.addEventListener('click', () => {
  getRecipeDetails(recipe.id);
});
recipeCard.appendChild(viewRecipeButton);

resultsDiv.appendChild(recipeCard);
});
}

function getRecipeDetails(recipeId) {
  const apiKey = 'a9fffb156669410ca87e41c585f0999a'; const url
  =
  `https://api.spoonacular.com/recipes/${recipeId}/information?apiKey=${apiKey}&include
  Nutrition=true`;

  fetch(url)
    .then(response => response.json())
    .then(data => { displayRecipeDetails(data);
    })
    .catch(error => { console.error('Error:',
      error);
    });
}

function displayRecipeDetails(recipe) {
  const modal = document.getElementById('recipe-modal');
  const recipeDetailsDiv = document.getElementById('recipe-details');

  recipeDetailsDiv.innerHTML = "";

  const recipeImage = document.createElement('img');
  recipeImage.src = recipe.image;
  recipeDetailsDiv.appendChild(recipeImage);

  const recipeTitle = document.createElement('h3');
  recipeTitle.textContent = recipe.title;

```

```

recipeDetailsDiv.appendChild(recipeTitle); const

calories = document.createElement('p');
calories.textContent = `Calories: ${recipe.nutrition?.nutrients?.find(n => n.name ===
'Calories')?.amount || 'N/A'} kcal`;
recipeDetailsDiv.appendChild(calories);

const fat = document.createElement('p');
fat.textContent = `Fat: ${recipe.nutrition?.nutrients?.find(n => n.name === 'Fat')?.amount ||
'N/A'} g`;
recipeDetailsDiv.appendChild(fat);

const carbs = document.createElement('p');
carbs.textContent = `Carbohydrates: ${recipe.nutrition?.nutrients?.find(n => n.name
=== 'Carbohydrates')?.amount || 'N/A'} g`;
recipeDetailsDiv.appendChild(carbs);

const protein = document.createElement('p');
protein.textContent = `Protein: ${recipe.nutrition?.nutrients?.find(n => n.name ===
'Protein')?.amount || 'N/A'} g`;
recipeDetailsDiv.appendChild(protein);

const instructions = document.createElement('div'); instructions.classList.add('instructions');
const instructionsTitle = document.createElement('h4'); instructionsTitle.textContent =
'Instructions'; instructions.appendChild(instructionsTitle);

// Ensure `analyzedInstructions` exists and is not empty
if (recipe.analyzedInstructions && recipe.analyzedInstructions.length > 0) {
  recipe.analyzedInstructions[0].steps.forEach(step => {
    const stepElement = document.createElement('p'); stepElement.textContent =
    `${step.number}. ${step.step}`; instructions.appendChild(stepElement);
  });
} else {
  const noInstructions = document.createElement('p'); noInstructions.textContent =
  'No instructions available for this recipe.'; instructions.appendChild(noInstructions);
}

```

```

recipeDetailsDiv.appendChild(instructions);

modal.style.display = 'flex';
}

document.querySelector('.close-button').addEventListener('click', function() { const
  modal = document.getElementById('recipe-modal'); modal.style.display = 'none';
});

// Close modal when clicking outside of it
window.onclick = function(event) {
  const modal = document.getElementById('recipe-modal'); if
  (event.target == modal) {
    modal.style.display = 'none';
  }
};

// Modify the 'View Recipe' button's event listener function
displayResults(results) {
  const resultsDiv = document.getElementById('results');
  resultsDiv.innerHTML = "";

  results.forEach(recipe => {
    const recipeCard = document.createElement('div');
    recipeCard.classList.add('recipe-card');

    const recipeImage = document.createElement('img');
    recipeImage.src = recipe.image;
    recipeCard.appendChild(recipeImage);

    const recipeTitle = document.createElement('h3');
    recipeTitle.textContent = recipe.title;
    recipeCard.appendChild(recipeTitle);

    // View Recipe Button
    const viewRecipeButton = document.createElement('button');
    viewRecipeButton.textContent = 'View Recipe';
    viewRecipeButton.addEventListener('click', () => {

```

```

        window.location.href = `recipe.html?id=${recipe.id}`;
    });
    recipeCard.appendChild(viewRecipeButton);

    // Add to Favorites Button
    const favoritesButton = document.createElement('button');
    favoritesButton.innerHTML = '<i class="fas fa-heart"></i>';
    favoritesButton.addEventListener('click', () => {
        addToFavorites(recipe.id);
    });
    recipeCard.appendChild(favoritesButton);

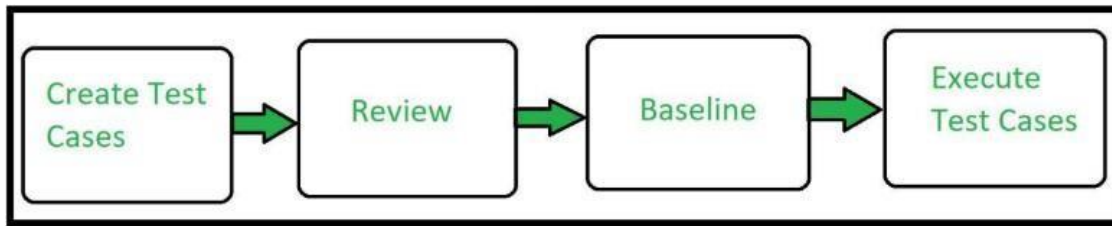
    resultsDiv.appendChild(recipeCard);
});
}

// Add to Favorites Function function
addToFavorites(recipeId) {
    let favorites = JSON.parse(localStorage.getItem('favorites')) || [];
    if (!favorites.includes(recipeId)) {
        favorites.push(recipeId);
        localStorage.setItem('favorites', JSON.stringify(favorites)); alert('Recipe
        added to favorites!');
    } else {
        alert('Recipe is already in favorites!');
    }
}
}

```

5.3 Testing

Unit Testing is a type of software testing where individual units or components of a software are tested. The purpose is to validate that each unit of the software code performs as expected. Unit Testing is done during the development (coding phase) of an application by the developers. Unit Tests isolate a section of code and verify its correctness. A unit may be an individual function, method, procedure, module, or object.



5.3.1 UNIT TESTING

Unit Testing is a software testing technique by means of which individual units of software i.e. group of computer program modules, usage procedures and operating procedures are tested to determine whether they are suitable for use or not. It is a testing method using whichever independent modules are tested to determine if there are any issue by the developer himself. It is correlated with functional correctness of the independent modules

Advantages of Unit Testing

1. **Early Bug Detection** – Issues are identified early in development, reducing the cost of fixing them later.
2. **Improved Code Quality** – Encourages developers to write modular and maintainable code.
3. **Faster Development** – Helps detect and fix problems quickly, speeding up the development process.
4. **Easier Debugging** – Since only a single function/module is tested at a time, debugging is straightforward.
5. **Enhances Code Reusability** – Properly tested units can be reused in different parts of the project with confidence.

Disadvantages of Unit Testing

1. **Time-Consuming** – Writing and maintaining test cases takes additional time.
2. **Cannot Catch All Errors** – Unit testing only checks individual components, not how they interact together.
3. **Increased Development Cost** – If not planned properly, unit testing may require extra resources.
4. **Requires Skilled Testers** – Writing effective test cases requires knowledge of testing frameworks and methodologies.

5.3.2 INTEGRATION TESTING

Integration testing is a software testing technique that focuses on verifying the interactions and data exchange between different components or modules of a software application. The goal of integration testing is to identify any problems or bugs that arise when different components are combined and interact with each other. Integration testing is typically performed after unit testing and before system testing. It helps to identify and resolve

integration issues early in the development cycle, reducing the risk of more severe and costly problems later on.



Advantages

- **It is convenient for small systems.**
- **Simple and straightforward approach.**
- **Can be completed quickly.**
- **Does not require a lot of planning or coordination.**
- **May be suitable for small systems or projects with a low degree of interdependence between components.**

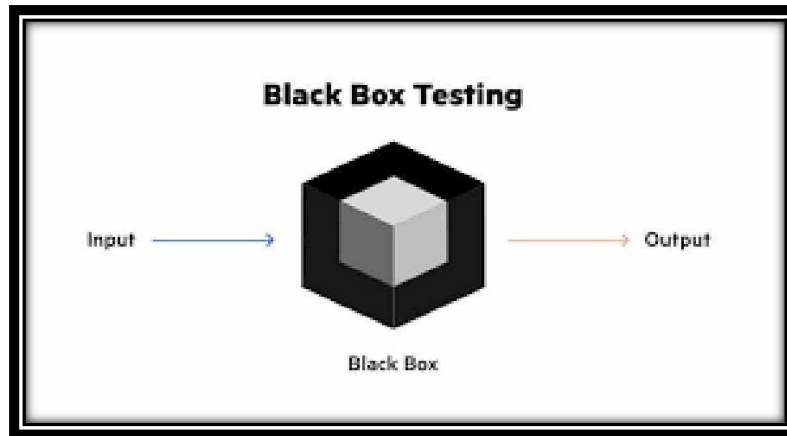
Disadvantages

- **There will be quite a lot of delay because you would have to wait for all the modules to be integrated.**
- **High risk critical modules are not isolated and tested on priority since all modules are tested at once.**
- **Not Good for long Projects.**
- **High risk of integration problems that are difficult to identify and diagnose.**

5.3.3 Black box testing.

Black box testing is a technique of software testing which examines the functionality of software without peering into its internal structure or coding. The primary source of black box testing is a specification of requirements that is stated by the customer.

In this method, tester selects a function and gives input value to examine its functionality, and checks whether the function is giving expected output or not. If the function produces correct output, then it is passed in testing, otherwise failed. The test team reports the result to the development team and then tests the next function. After completing testing of all functions if there are severe problems, then it is given back to the development team for correction.



Techniques of Black Box Testing

1. Equivalence Class Testing

- Used to minimize the number of test cases while maintaining sufficient test coverage.
- Divides input data into equivalence classes, where one representative value is tested instead of every possible input.

2. Boundary Value Testing

- Focuses on testing the boundary values of input ranges.
- Determines whether the system correctly handles edge cases (e.g., minimum, maximum, and just beyond valid input ranges).
- Useful for systems where inputs fall within specific numerical or categorical ranges.

3. Decision Table Testing

1. Uses a decision table to represent inputs (causes) and expected outputs (effects) in a structured manner.
2. Each column represents a unique combination of inputs and their corresponding outputs, ensuring thorough testing of different conditions.

Advantages of Black Box Testing

- **Efficient for large systems** – Useful for testing complex applications without needing access to source code.
- **Unbiased testing** – Since testers and developers work independently, results are more objective.
- **Non-technical testers** – Testers do not need programming knowledge, making it accessible to a wider range of professionals.
- **Focuses on user experience** – Since testing is based on functional requirements, it ensures that the system meets end-user expectations.
- **Detects ambiguities in requirements** – Helps uncover contradictions or missing details in the functional specifications.

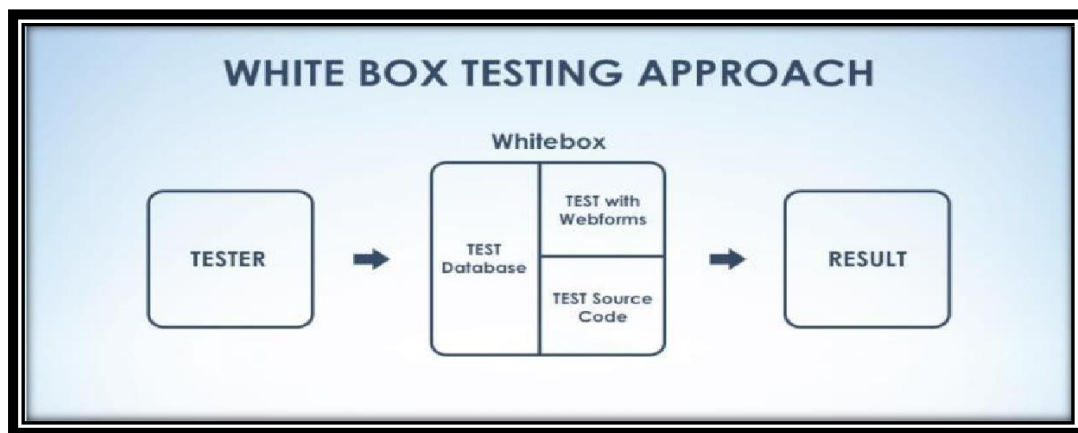
Disadvantages of Black Box Testing

- **Difficult test case design** – Without clear functional specifications, designing effective test cases is challenging.
- **Limited coverage** – It may be hard to test all possible inputs, especially with time constraints.
- **Hidden paths** – Some execution paths may go untested, leading to undiscovered defects.
- **Duplicate testing** – There is a risk of repeating tests already performed by developers.

5.3.4 WHITE BOX TESTING

White box testing techniques analyse the internal structures the used data structures, internal design, code structure, and the working of the software rather than just the functionality as in black box testing. It is also called glass box testing or clear box testing or structural testing. White Box Testing is also known as transparent testing, open box testing.

Developers do white box testing. In this, the developer will test every line of the code of the program. The developers perform the White-box testing and then send the application or the software to the testing team, where they will perform the black box testing and verify the application along with the requirements and identify the bugs and sends it to the developer.



Advantages

- Code optimization by revealing hidden errors
- Transparency of the internal coding structure which is helpful in deriving the type of input
- data needed to test an application effectively
- Covers all possible paths of a code thereby, empowering a software engineering team to
- conduct thorough application testing
- Enables programmer to introspect because developers can carefully describe any new implementation
- Test cases can be easily automated
- Gives engineering-based rules to stop testing an application

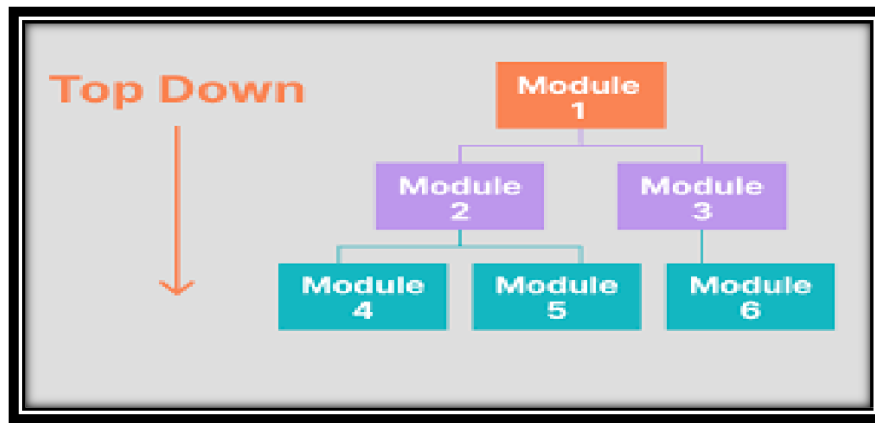
Disadvantages

- A complex and expensive procedure which requires the adroitness of a seasoned professional, expertise in programming and understanding of internal structure of a code
- Updated test script required when the implementation is changing too often
- Exhaustive testing becomes even more complex using the white box testing method if the application is of large size
- Some conditions might be untested as it is not realistic to test every single one
- Necessity to create full range of inputs to test each path and condition make the white box testing method time-consuming
- Defects in the code may not be detected or may be introduced considering the ground rule of analysing each line by line or path by path.

5.3.5 TOP DOWN APPROACH

The top down integration testing method is an incremental approach that involves joining two or more logically related modules. The process involves using dummy programs called Stubs and Drivers to stimulate the behaviour of unintegrated lower-level modules.

This technique follows the control flow or architectural structure to test it from top to bottom. Unit testing is performed on the top module alone, followed by integration of the lower modules. In this way, all the modules are integrated and tested simultaneously



Advantages

- The tested product is very consistent, as the integration testing is basically performed in an environment that is almost similar to that of reality.
- Stubs can be written within lesser time. Because when compared to the drivers, Stubs are simpler to author.
- Through Top-Down integration it is possible to obtain an early prototype.
- Critical Modules are tested on priority.
- Major design flaws could be found and fixed first.
- Depth first strategy allows a complete function of the software to be implemented.
- Verifies major control or decision early in the test process.

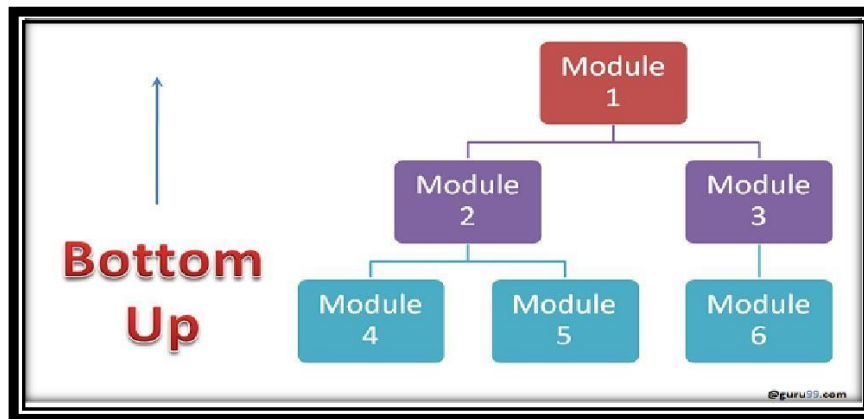
Disadvantages

The basic functionality of the software is tested at the end of the cycle.

- It requires many stubs.
- Modules at the lower level tested inadequately.

5.3.6 BOTTOM UP APPROACH

Bottom-up Testing is a type of incremental integration testing approach in which testing is done by integrating or joining two or more modules by moving upward from bottom to top through control flow of architecture structure. In these, low-level modules are tested first, and then highlevel modules are tested. This type of testing or approach is also known as inductive reasoning and is used as a synthesis synonym in many cases. Bottom-up testing is user-friendly testing and results in an increase in overall software development. This testing results in high success rates with longstanding results.



Advantages of Bottom Up Approach

- If the low level modules and their combined functions are often invoked by other modules, then it is more useful to test them first so that meaningful effective integration of other modules can be done.
- Appropriate for applications where bottom up design methodology is used.
- Advantageous if major flaws occur towards the bottom of the program.
- Test conditions are easier to create.
- Observation of test results is easier.
- Always starting at the bottom of the hierarchy again means that the critical modules are generally built and tested first and therefore any errors or mistakes in these forms of modules are identified early in the process.

Disadvantages of Bottom Up Approach

- Test engineers cannot observe system level functions from a partly integrated system. They cannot observe the system level functions until the top level test driver is in place.
- The program as an entity does not exist until the last module is added.
- One big disadvantage of bottom up strategy is that in this sort of testing no working model can be represented as far as several modules have been built

5.4 SYSTEM TESTING

System Testing includes testing of a fully integrated software system. Generally, a computer system is made with the integration of software (any software is only a single element of a computer system). The software is developed in units and then interfaced with other software and hardware to create a complete computer system. In other words, a computer system consists of a group of software to perform the various tasks, but only software cannot perform the task; for that software must be interfaced with compatible hardware. System testing is a series of different type of tests with the purpose to exercise and examine the full working of an integrated software computer system against requirements.

5.5 FUNCTIONAL TESTING

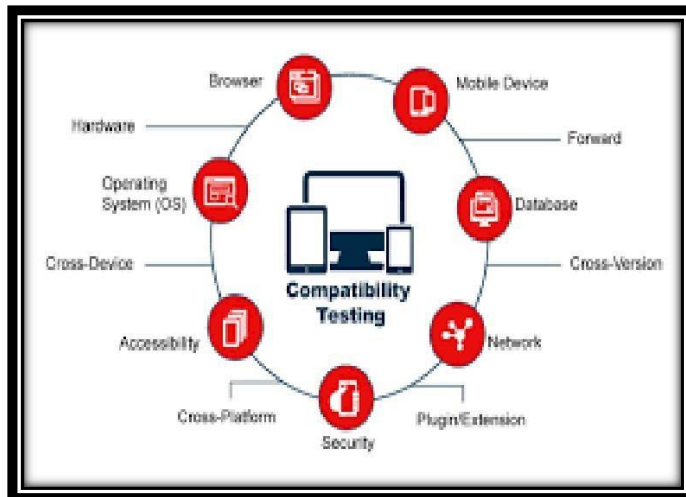


Functional testing is a type of testing that seeks to establish whether each application feature works as per the software requirements. Each function is compared to the corresponding requirement to ascertain whether its output is consistent with the end user's expectations. The testing is done by providing sample inputs, capturing resulting outputs, and verifying that actual outputs are the same as expected outputs.

Test in Functional Testing

- Mainline functions: Testing the main functions of an application
- Basic Usability: It involves basic usability testing of the system. It checks whether a user can freely navigate through the screens without any difficulties.
- Accessibility: Checks the accessibility of the system for the user
- Error Conditions: Usage of testing techniques to check for error conditions. It checks whether suitable error messages are displayed.

5.6 COMPATIBILITY TESTING



Compatibility testing examines the compatibility of the application and the product with different computing environments. It is a part of non-functional testing. It tests the usability, reliability and performance of the application and the product. The ISO 25010 standard defines it as a characteristic or extent to which a software system can exchange information with other systems whilst sharing the same software and hardware.

Backward compatibility

Also called downward compatibility is when older versions of the application or software are tested for compatibility with newer hardware and software. It is relevant when some users may operate the application on old devices.

Forward compatibility

Testing tests an application or software in new versions of hardware and software. It verifies if existing hardware and software perform smoothly with the newer version. Within these two types of compatibility testing are several, more specific categories

How does Compatibility test work?

In compatibility testing, we define the set of environments or platforms the application is expected to work in. Following this, a test plan is developed to determine the most important issue(s) faced by the application so that they can be prioritized in these tests. It is critical to set up the environments to simulate what the end-user would experience, such as desktops, smartphones, tablets etc. It is also important for the tester to have sufficient knowledge of various software, hardware and platforms and how they respond in various

configurations. Once the environment is set up, the tests can be run and any bugs and defects that are detected should be reported.

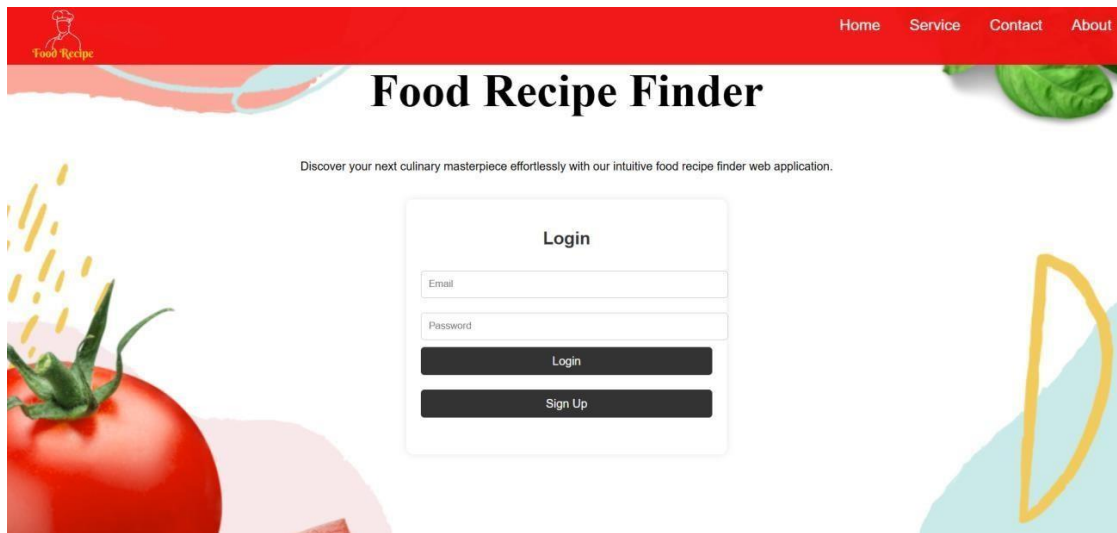
5.7. TEST CASES

A test case is a defined format for software testing required to check if a particular application/software is working or not. A test case consists of a certain set of conditions that need to be checked to test an application or software i.e. in more simple terms when conditions are checked it checks if the resultant output meets with the expected output or not. A test case consists of various parameters such as Id, condition, steps, input, expected result, result, status, and remarks.

Test Case ID	Module (Unit to be tested)	Description	Desired Output	Observed Output	Pass or Fail
TC001	User Registration	Verify if a new user can register successfully	User registered and redirected to homepage	As expected	Pass
TC002	User Login	Verify if a registered user can log in with valid credentials	User logged in and redirected to dashboard	As expected	Pass
TC003	User Login	Verify login with incorrect credentials	Display error message	As expected	Pass
TC004	Recipe Search	Check if users can search for recipes by name	Relevant recipes displayed	As expected	Pass
TC005	View Recipe	Verify if clicking 'View Recipe' opens the correct	Recipe details displayed correctly	As expected	Pass
TC006	View Recipe	Verify if images and ingredients are loaded properly	Images, ingredients, and instructions displayed	As expected	Pass
TC007	Add to Favorites	Check if a user can add a recipe to favorites	Recipe added to favorites list	As expected	Pass
TC008	View Favorites	Verify if users can see their favorite recipes list	Favorites list displayed	As expected	Pass

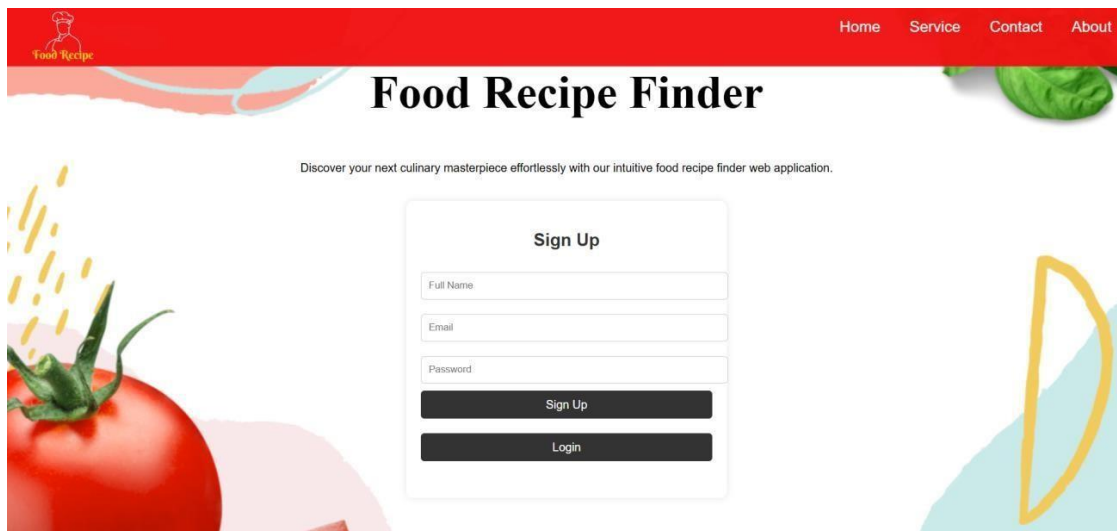
6.2 USER DOCUMENTATION

1. User Login



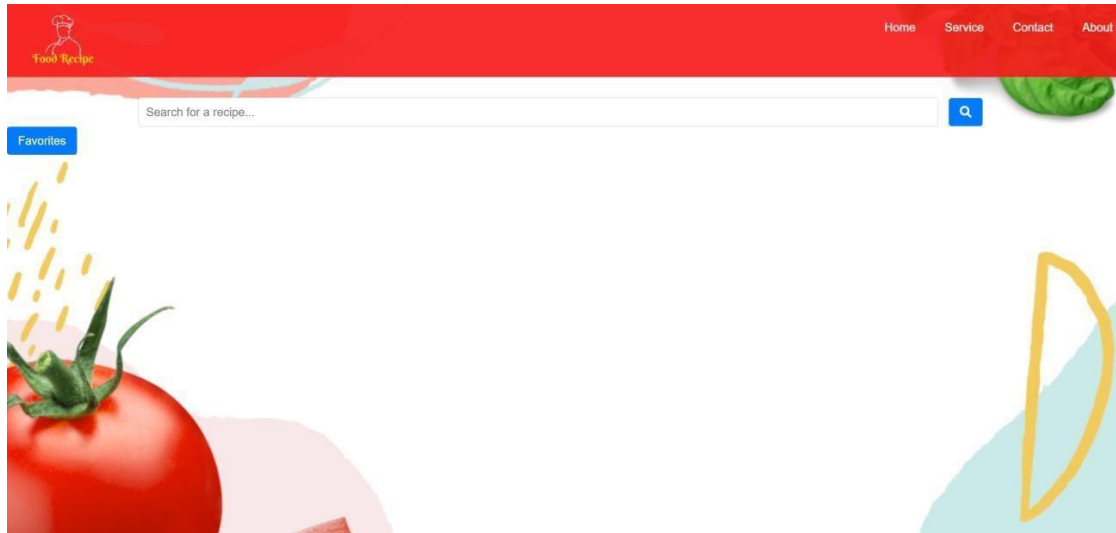
The screenshot displays the 'Food Recipe Finder' web application. The header is red with a logo on the left and navigation links (Home, Service, Contact, About) on the right. The main title 'Food Recipe Finder' is centered in a large, bold, black font. Below the title, a subtitle reads: 'Discover your next culinary masterpiece effortlessly with our intuitive food recipe finder web application.' The central focus is a white 'Login' form with a title 'Login' at the top. It contains two input fields: 'Email' and 'Password'. Below these fields are two dark gray buttons: 'Login' and 'Sign Up'. The background features a red tomato on the left and a yellow lemon slice on the right, both with a light blue water splash effect.

2. User sign up

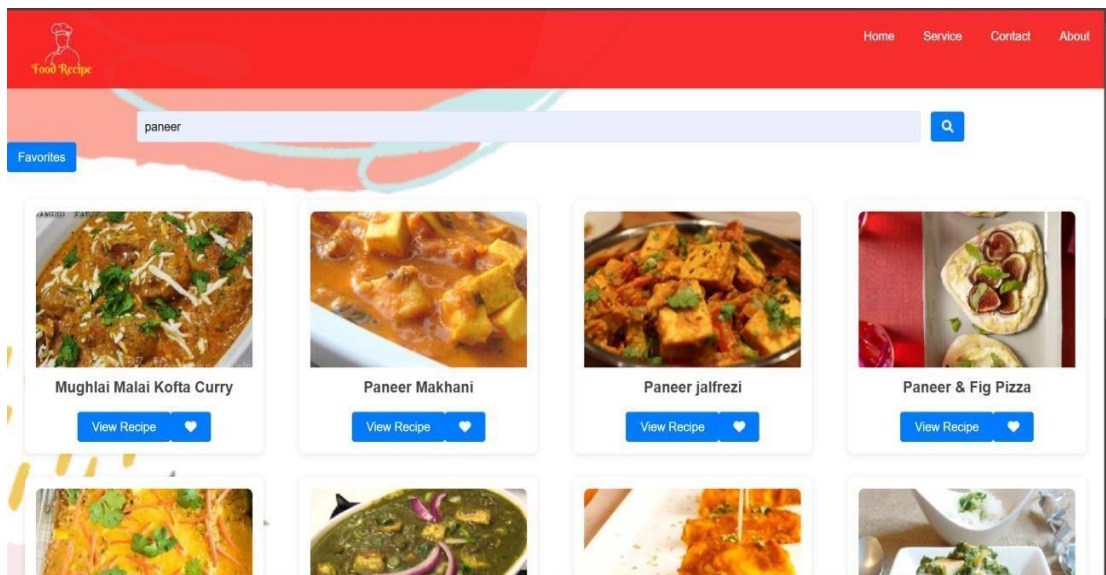


The screenshot displays the 'Food Recipe Finder' web application. The header is red with a logo on the left and navigation links (Home, Service, Contact, About) on the right. The main title 'Food Recipe Finder' is centered in a large, bold, black font. Below the title, a subtitle reads: 'Discover your next culinary masterpiece effortlessly with our intuitive food recipe finder web application.' The central focus is a white 'Sign Up' form with a title 'Sign Up' at the top. It contains three input fields: 'Full Name', 'Email', and 'Password'. Below these fields are two dark gray buttons: 'Sign Up' and 'Login'. The background features a red tomato on the left and a yellow lemon slice on the right, both with a light blue water splash effect.

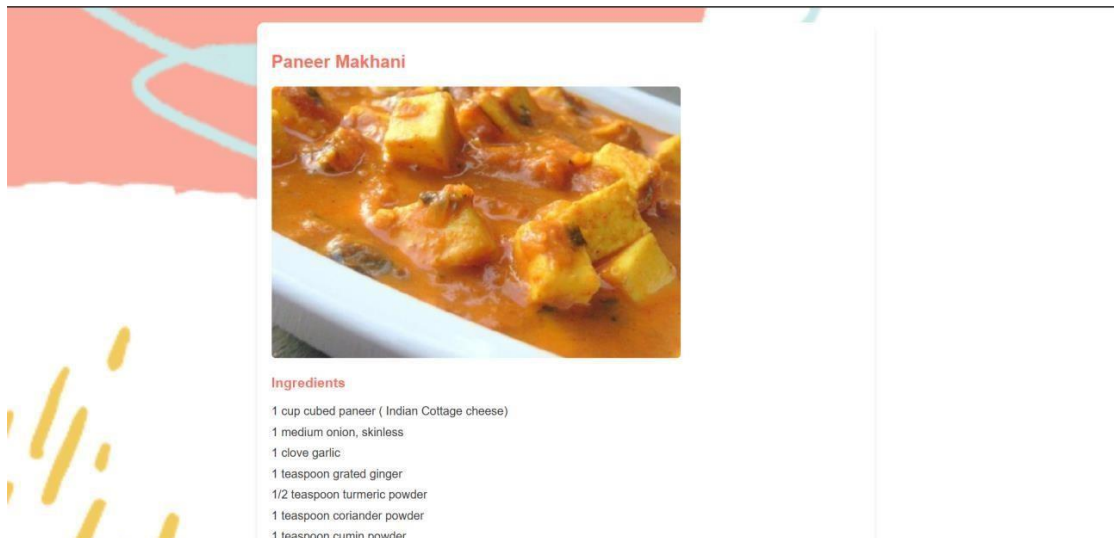
3. Home page



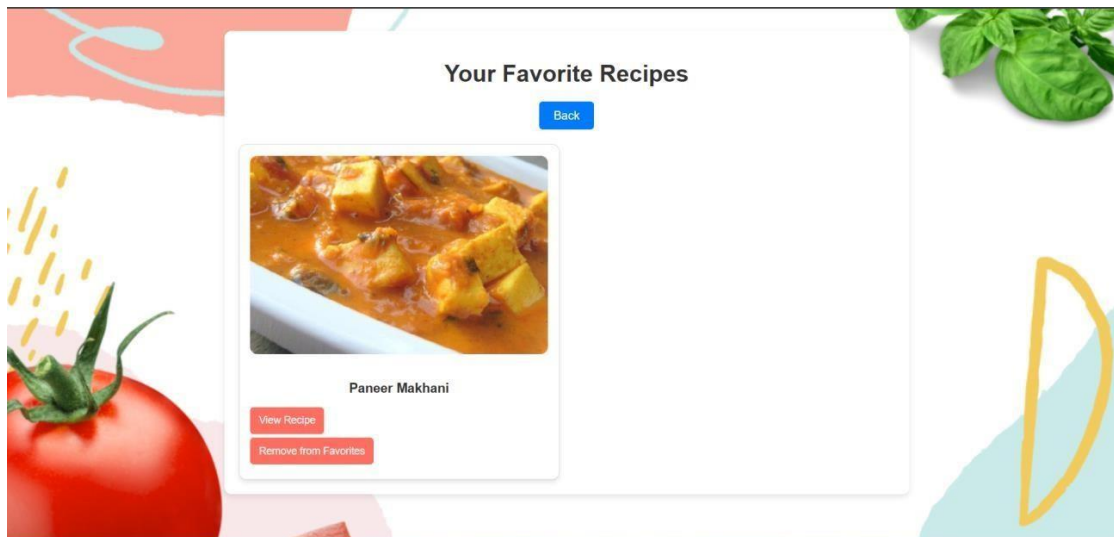
4. Search result



5. Recipe detail page



6. Favourite page



Chapter 7:

Conclusion and Future Scope

7.1 Conclusion

The Food Recipe Web Application successfully provides users with an efficient way to search for recipes, view detailed instructions, and save their favorite recipes. With an intuitive user interface and seamless API integration, the application enhances the cooking experience by offering accurate and relevant food recipes based on user queries. The project achieves its primary goal of simplifying recipe discovery while ensuring ease of use and accessibility. Overall, the system demonstrates a well-structured approach to modern digital recipe management.

7.2 Limitations of the System

Despite its success, the application has certain limitations:

1. **Dependence on Third-Party APIs** – The application relies on the Spoonacular API for fetching recipe data, making it vulnerable to API downtime, rate limits, or changes in API policies.
2. **Limited Customization** – Users cannot modify or add their own recipes directly within the application.
3. **No Offline Access** – The application requires an internet connection to fetch and display recipe details.
4. **Basic Search Functionality** – The search is limited to text-based queries, and may not always return the most relevant results.
5. **Image Recognition Accuracy** – The image-based recipe search depends on the accuracy of the image recognition API, which may not always produce precise results.

7.3 Future Scope

The future development of the Food Recipe Web Application can address these limitations and expand its capabilities. Some potential enhancements include:

1. **User-Generated Recipes** – Implementing a feature that allows users to add, modify, and share their own recipes within the application.
2. **Offline Mode** – Enabling offline access to previously searched recipes through caching techniques.

3. Enhanced Search Functionality – Improving the search algorithm with AI-based recommendations and voice search.
4. Advanced Image Recognition – Utilizing more robust machine learning models to improve food image recognition accuracy.
5. Mobile App Development – Extending the application into a mobile app for Android and iOS users to enhance accessibility and user engagement.
6. Personalized Recommendations – Implementing a recommendation system based on user preferences and past searches.
7. Integration with Smart Kitchen Devices – Allowing users to sync their recipes with smart kitchen appliances for a seamless cooking experience.