HCLTech
Supercharging Progress™

Career Shaper™

Project Title: Online Recipe Finder

Team Members,

RAUSHAN KUMAR

(3502110549)

PRAGYA AASTHA

(3502110542)

SAURABH KR. SINGH

(3502110555)

PROJECT DOCUMENTATION

Submitted By:

NAME	RAUSHAN KUMAR
REG NO.	3502110549
BRANCH	B.E-C.S. E
DATE OF SUBMISSION	27/03/2025

HCLTech

HCL Tech

VINAYAKA MISSION'S RESEARCH FOUNDATION

VISION • WISDOM • UNITY

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

One of the most important human needs is food. It provides the energy needed for daily life. Today, food is more than just a necessity—it is an art. Innovative recipes and cooking techniques continue to emerge, transforming simple ingredients into creative dishes. Many of these innovations use minimal raw materials to create something unique.

A recipe is a set of instructions that guide food preparation and cooking. It includes essential details such as ingredients, steps, tools, cooking time, and portion sizes. However, not everyone can remember every recipe they come across. Limited ingredients can also make cooking challenging, especially for those who are not passionate about it.

In this digital age, web apps help users search for recipes, often based on a dish's name. However, many apps lack a feature that suggests recipes based on available ingredients. That's where our solution comes in.

We are developing a web application that allows users to search for recipes based on the ingredients they have. Using the Spoonacular API, the app will scan and identify available ingredients, then suggest suitable recipes. This is making cooking easier, smarter, and more convenient.

1.1 ABSTRACT

The Recipe Finder Web Application is designed to help users discover and explore new recipes based on their personal preferences, dietary needs, and available ingredients. Built using Python and Flask for the backend and React for the frontend, this application provides a seamless and intuitive user experience.

Users can search for recipes by entering criteria such as ingredients, cooking time, cuisine type, and dietary restrictions (e.g., vegan, gluten-free, low-carb). The application pulls recipe data from external APIs like Spoonacular, offering a wide variety of recipe options along with detailed instructions, nutritional information, and high-quality images.

In addition to searching for recipes, users can also submit their recipes, contributing to a growing community of food enthusiasts. The app includes features such as saving favorite recipes, generating shopping lists, and sharing recipes on social media. Users can create an account to personalize their experience further and keep track of their recipe collection. Ultimately, the Recipe Finder web application provides a convenient, interactive, and personalized platform for discovering, sharing, and enjoying meals tailored to users' tastes and dietary preferences.

1.2 Aims & Objectives

The main aim of our application is to provide recipes to the consumers based on the ingredients already available with them, unlike other recipe providing applications where the ingredients available with the consumer is not taken into consideration.

The objectives of our project are as follows:

- To help the user decide a recipe to cook from the ingredients available with him/her.
- To guide the user to the recipe based on the user's choices and needs.
- To help save the user money and time by tediously referencing cook books and buying ingredients he/she does not need.

1.3 Problem Statement

In today's fast-paced world, finding the right recipe based on available ingredients, dietary preferences, or cooking time can be a challenge. Many existing recipe finder applications allow users to search for recipes by dish names but lack a feature that suggests recipes based on the ingredients they have.

Additionally, people with dietary restrictions (such as vegan, gluten-free, or low-carb diets) often struggle to find suitable meal options. Manually searching for recipes that match these requirements is time-consuming and inefficient.
Furthermore, users may want to save favorite recipes, generate shopping lists, or contribute their own recipes to a community platform. Existing solutions do not always provide these functionalities in a single, user-friendly interface.

Thus, there is a need for an Online Recipe Finder that offers a personalized, ingredient-based search system with additional features like user authentication, saving favorite recipes, filtering by dietary preferences, and API integration for fetching real-time recipe data.

1.3.1 Application Modules

The web application has four major functionalities in this application. They are browsing all the favourite recipes, filtering and displaying the recipes-based ingredients selected by the user, displaying all the recipes and adding a new recipe or an ingredient to the database.

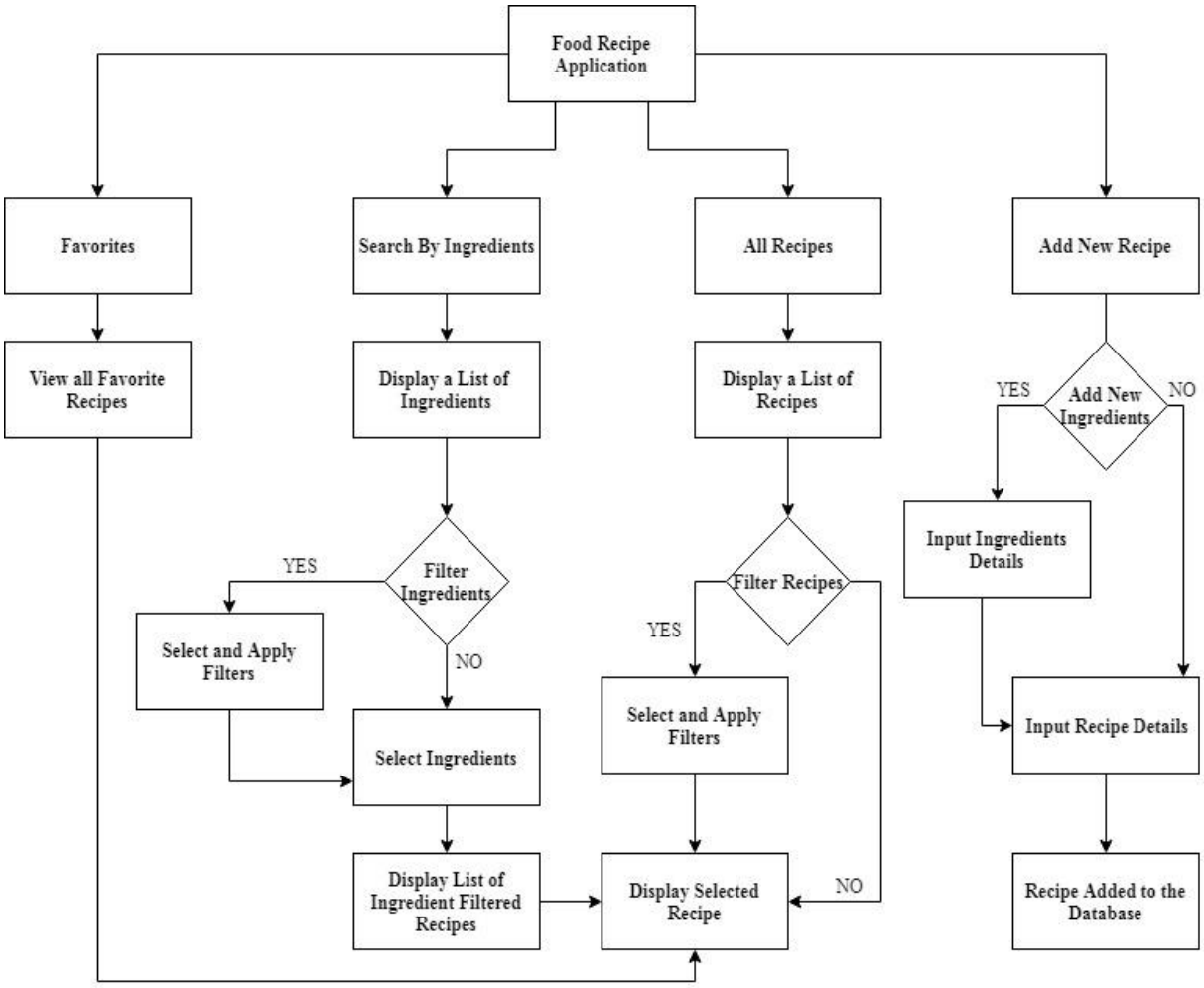


Fig No: 1

1.4 Scope of the Project

The Online Recipe Finder is a web-based application that helps users find and manage recipes efficiently. It offers the following key features:

Features in Scope:

- Ingredient-Based Search – Users can input ingredients to get recipe suggestions.
- Recipe Filtering – Search by cuisine, dietary preference, cooking time, and meal type.
- Save Favorite Recipes – Users can bookmark and manage recipes.
- Recipe Submission – Users can add their own recipes to the platform.
- API Integration – Fetches real-time recipes, images, and nutrition details via Spoonacular API.

CHAPTER 2

LITERATURE SURVEY

2.1 Overview of Existing Recipe Finder Applications

Several web-based recipe finder platforms, such as AllRecipes, Yummly, and Tasty, allow users to discover new recipes. These platforms offer features like search by dish name, user ratings, video tutorials, and AI-based recommendations. Some applications integrate APIs like Spoonacular to fetch real-time recipe data.

2.2 Comparative Analysis of Similar Platforms

Platform	**Key Features**	**Limitations**
AllRecipes	Large recipe database, user reviews, video tutorials	Lacks an effective ingredient-based search, limited dietary filtering
Yummly	AI-based recommendations, extensive filtering options	Requires user login, some features are paid
Tasty	Step-by-step video tutorials, social media integration	No advanced filtering, limited personalized recommendations
Spoonacular API	Provides real-time recipes, nutrition data, and ingredient analysis	API request limits, dependency on external data sources

Table No: 1

2.3 Limitations of Existing Solutions

Despite the availability of recipe finder applications, several limitations exist:

- Limited Ingredient-Based Search – Many apps do not effectively suggest recipes based on available ingredients.
- Lack of Personalized Features – Few platforms allow users to save recipes, track preferences, or contribute their own recipes.
- API Rate Limitations – Platforms relying on Spoonacular or other APIs may face restrictions on API calls and performance delays.
- No Custom Recipe Contributions – Users cannot add personal recipes to many platforms.

2.4 How This Project Improves Upon Previous Works

The Online Recipe Finder addresses these issues by leveraging the Spoonacular API while adding additional functionalities:

- Ingredient-Based Recipe Search – Users can input available ingredients, and the system suggests suitable recipes.
- Advanced Filtering – Users can search recipes based on cuisine, dietary preferences (vegan, keto, gluten-free), and cooking time.
- User Recipe Contribution – Enables users to submit their own recipes to expand the database.
- Personalized Experience – Registered users can save favorite recipes and receive tailored suggestions.
- Optimized API Usage – Implements caching and optimized API calls to reduce Spoonacular API rate limits.

CHAPTER 3

SYSTEM DESIGN & METHODOLOGY

3.1 System Architecture

Overview

The **Online Recipe Finder** follows a **three-tier architecture**, ensuring scalability, security, and efficient data flow between components. The architecture consists of:

- **Frontend (Client-Side)** – Built using React.js, it provides an interactive UI and communicates with the backend via API requests.
- **Backend (Server-Side)** – Developed using Flask (Python), it handles user authentication, processes API requests, and interacts with the database.
- **Database (Storage Layer)** – Managed using MySQL, it stores user details, saved recipes, and user-submitted recipes.

Architectural Diagram

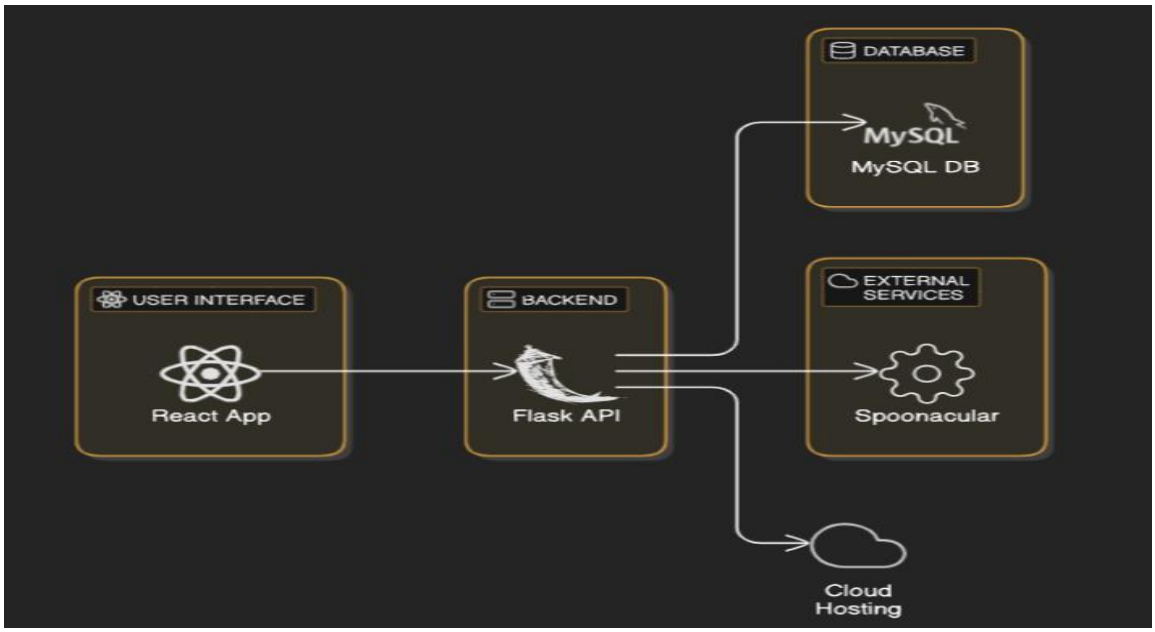


Fig No: 2

3.2 Technology Stack

It is built using a combination of modern web technologies to ensure efficiency, scalability, and security.

The key components and their technologies are as follows:

Frontend: The frontend of the application is developed using **React.js**. It provides a dynamic and responsive user interface, enabling users to search for recipes, filter results, and interact

with the application. The frontend also handles API communication by sending requests to the backend and displaying the fetched data.

Backend: The backend is built using Flask (Python). It manages user authentication, processes API requests, and integrates with the database. Flask provides a lightweight yet powerful framework for handling business logic and ensuring seamless communication between the frontend and database.

Database: The system uses MySQL to store user-related data, including login credentials, saved recipes, and user-submitted recipes. The relational database structure allows efficient querying and ensures data integrity.

External API: The application leverages the **Spoonacular API** to fetch real-time recipe data, images, and nutritional information. The API integration enhances the recipe search functionality by providing a vast collection of recipes from external sources.

Authentication: For user authentication and session management, the system implements JWT (JSON Web Tokens**)**. JWT ensures secure login, allowing users to access personalized features such as saving recipes and submitting their own.

Hosting: Deployed on Heroku or AWS for cloud-based scalability and high availability.

Version Control: The project is managed using GitHub, which facilitates collaboration, version control, and tracking of code changes. GitHub allows developers to maintain different versions of the application and work efficiently in a team environment.

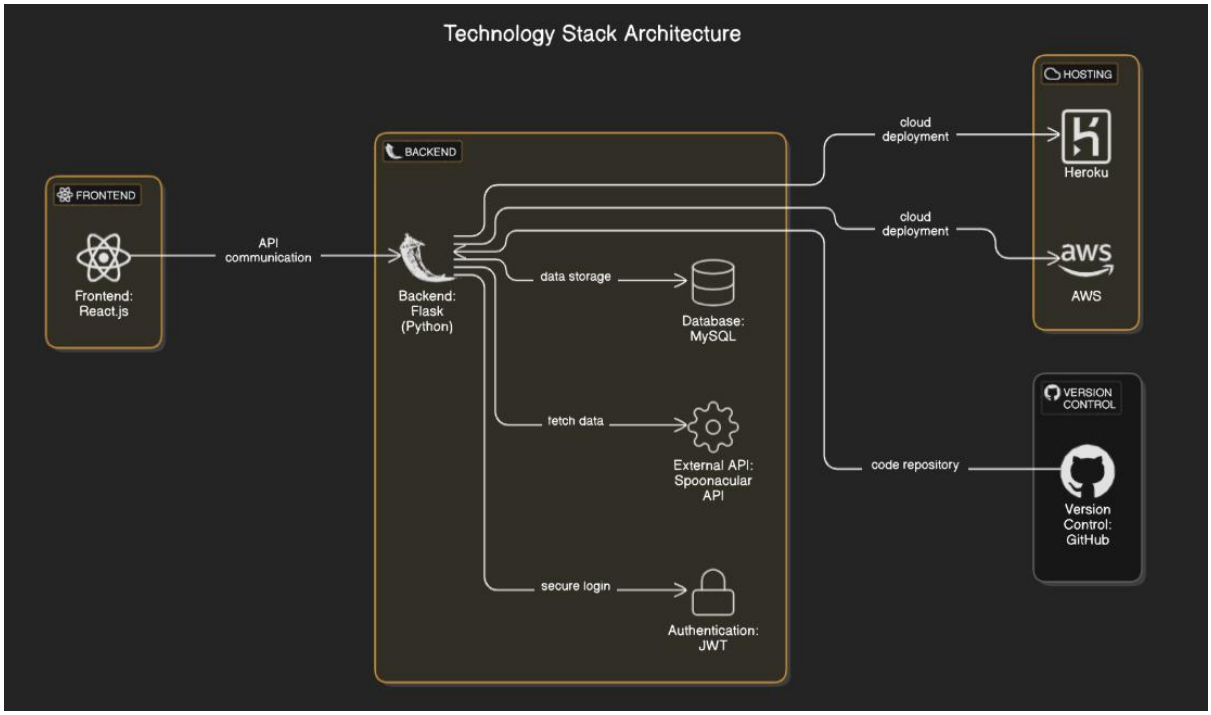


Fig No: 3

3.3 Methodology (Waterfall Model)

A method can be defined as a systematic and orderly procedure or process for attaining a specific objective. Methodology does not describe specific methods but outlines a structured process that must be followed to achieve the desired results. These processes form a generic framework that can be broken down into sub-processes, combined, or rearranged as needed. However, every task must adhere to these processes in some form. Methodology may simply describe a process or expand into a collection of theories, concepts, and ideas related to a particular discipline or field of study. One of the methodologies applied in the Recipe Finder Web Application is the Waterfall Model. This model provides a structured and sequential approach to system development. By following the Waterfall Methodology, the system is developed through the following phases:

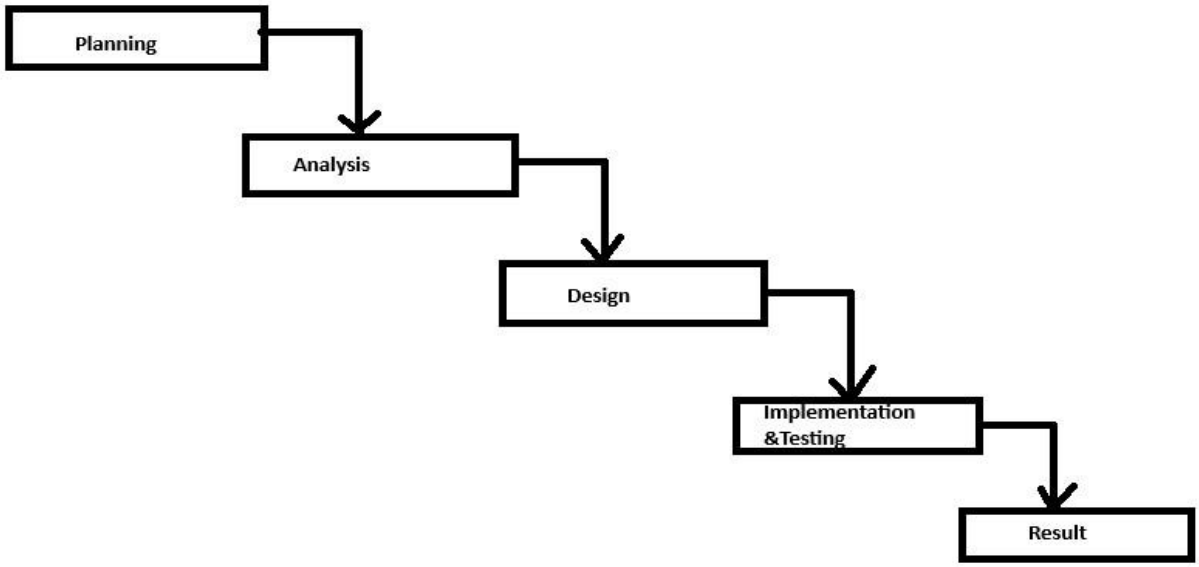


Fig No: 4

Planning – Defining the project scope, goals, and functionalities of the recipe finder web application.

Analysis – Understanding user needs, ingredient-based search requirements, and Spoonacular API integration.

Design – Structuring the web application's UI/UX, database, and API interactions.

Implementation – Developing the web application using technologies such as React.js, Python, and MySQL.

Testing – Ensuring the application functions correctly, validating API responses, and checking for bugs.

Deployment & Maintenance – Deploying the application on a web server and continuously improving it based on user feedback.

CHAPTER 4

IMPLEMENTATION & FEATURES

4.1 Features & Functionalities

The Online Recipe Finder is intended to give users a fast and personalized way of finding and organizing recipes. The main features are:

Ingredient-Based Recipe Search

Users can enter ingredients they have, and the system recommends suitable recipes retrieved from the Spoonacular API and some of ther recipes from Databases.

Recipe Filtering & Categorization

Users can refine their search by:

- Cuisine type (e.g., Italian, Indian, Chinese)
- Dietary preferences (e.g., Vegan, Gluten-Free, Keto)
- Cooking time (Quick meals vs. elaborate dishes)
- Meal type (Breakfast, Lunch, Dinner, Snacks)

User Authentication & Profile Management

- Secure login and signup using JWT authentication.
- Users can create and manage profiles with personalized preferences.

Saving & Managing Favorite Recipes

- Users can bookmark their favorite recipes for future reference.
- Saved recipes are stored in the MySQL database for easy retrieval.

Recipe Submission by Users

- Users can add their recipes, which are stored in the database and accessible to other users.

API Integration with Spoonacular

- Fetches real-time recipes, nutritional data, and cooking instructions.
- Ensures access to a vast database of recipes beyond user-submitted content.

Responsive & User-Friendly Interface

- Developed with React.js, ensuring smooth navigation across desktops, tablets, and mobile devices.

4.2 Database Schema & Design (ER Diagram & Table Structures)

The Online Recipe Finder follows a relational database structure using MySQL. The database consists of multiple tables to efficiently store and manage user data, recipes, and interactions.

Entity-Relationship (ER) Diagram

The system includes the following entities:

- **Users** – Stores user details and authentication data.
- **Recipes** – Stores user-submitted recipes and API-fetched recipes.
- **Saved_Recipes** – Links users to their saved recipes.
- **Comments** – Stores user comments on recipes.
- **Ingredients** – Stores the list of ingredients for each recipe, including quantity and measurement units.
- **Feedback** – Stores user ratings and feedback on recipes, helping improve recommendations.
- **Directions** – Stores the step-by-step cooking instructions for recipes, ensuring a structured format.

ER Diagram

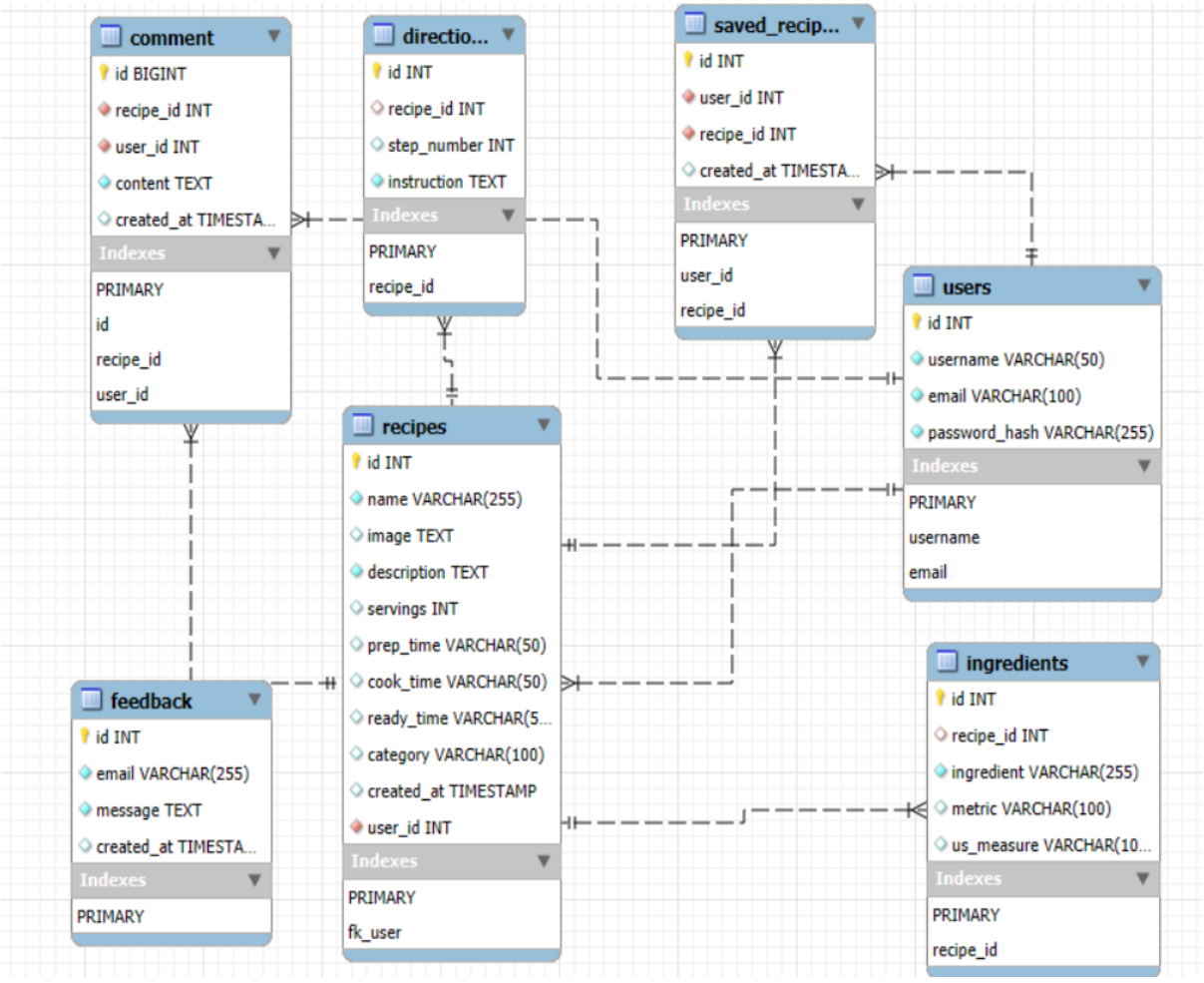


Fig No: 5

4.3 API Integration (Spoonacular API)

The **Spoonacular API** is used to fetch real-time **recipes, images, ingredients, cooking instructions, and nutritional details**.

- **How the API Works:**
  - The frontend sends a request to the backend to search for recipes based on ingredients.
  - The backend invokes the Spoonacular API, obtains the data, and forwards it to the frontend.
  - The UI shows the recipes with images, steps, and other information.

- **Example API Call:**

  A GET request to fetch recipes using Spoonacular:

Fig No: 6

4.4 Implementation Details

4.4.1 Frontend (React.js)

The frontend is built using React.js, offering a dynamic and interactive user experience.

- Key Features:
  - Search Bar: Allows users to enter ingredients and retrieve matching recipes.
  - Recipe Display Page: Shows recipe details, images, and instructions.
  - User Authentication: Implements JWT-based login/signup.
  - Saved Recipes Section: Enables users to bookmark their favorite recipes.

- Libraries Used:
  - Axios (for API calls)
  - React Router (for navigation)
  - Redux (for state management)

4.4.2 Backend (Flask & Python)

The backend is built with Flask, serving as the API layer to handle business logic and database interactions.

- Key Features:
  - Handles API requests from the frontend.
  - Processes user authentication (JWT).
  - Fetches recipes from Spoonacular API.
  - Manages user-submitted recipes and comments.

Example Flask API Route:

```
from flask import Flask, request, jsonify
import requests

app = Flask(__name__)

@app.route('/search_recipes', methods=['GET'])
def search_recipes():
    ingredients = request.args.get('ingredients')
    api_url = f"https://api.spoonacular.com/recipes/findByIngredients?ingredients=
{ingredients}&apiKey=your_api_key"
    response = requests.get(api_url)
    return jsonify(response.json())

if __name__ == '__main__':
    app.run(debug=True)
```

Fig No: 7

The given Flask API code creates a route to find recipes from the Spoonacular API by searching using ingredients provided by the user. The script then imports the modules required, namely Flask for handling web, requests for sending HTTP requests, and jsonify for outputting JSON response. The Flask app is set up, and a route /search_recipes is established for handling GET requests. When a query to this route includes an ingredients parameter, the method takes the value from the query string and forms a URL to Spoonacular's API. The API URL has the dynamic components of the ingredients and an API key with which to search for matching recipes. A request to Spoonacular's API is issued, and it is first transformed to a JSON format

upon its return. Lastly, the script starts the Flask application in debug mode with automatic reloading for ease of development.

This API allows users to search for recipes based on ingredients and dietary preferences.

```python
from flask import Flask, Blueprint, request, jsonify, session
import requests

def get_apirecipes(ingredients, diet):
    url = "https://api.spoonacular.com/recipes/complexSearch"
    params = {
        "apiKey": "9cfad92bb48d4c398dbf066de10f40f5",
        "query": ingredients,
        "diet": diet if diet != "None" else "",
        "number": 10,
        "addRecipeInformation": True
    }
    response = requests.get(url, params=params)
    return response.json() if response.status_code == 200 else {"error": "API request failed"}

# Create a blueprint for authentication-related routes
auth_routes = Blueprint("auth_routes", __name__)

# Route to handle recipe search
@auth_routes.route('/api/search', methods=['POST'])
def search():
    data = request.get_json()
    ingredients = data.get("ingredients")
    diet = data.get("diet")

    recipes = get_apirecipes(ingredients, diet)
    return jsonify(recipes)
```

Explanation:

This Flask API helps users find recipes using the **Spoonacular API** based on provided ingredients and dietary preferences.

- ### Function get_apirecipes(ingredients, diet)
  - Sends a request to Spoonacular API with the given ingredients and diet type.
  - Retrieves up to 10 matching recipes along with extra details.
  - Returns the recipes in JSON format.
- ### Blueprint auth_routes
  - Creates a route group for handling authentication-related requests.

➢ Route /api/search (POST request)

- Receives JSON data from the user (ingredients & diet).
- Calls get_apirecipes() to fetch recipes.
- Returns the recipe list as a JSON response.

Recipe Display Section

```
{!loading && searchSubmitted && (
  <div className="recipes-container">
    <h1>Great recipes start with simple ingredients!</h1>
    {recipes.length === 0 ? (
      <p>No recipes found. Please try different ingredients or diets.</p>
    ) : (
      <div className="recipe-grid">
        {recipes.map((recipe, index) => (
          <div key={index} className="recipe">
            <img src={recipe.image} alt={recipe.title} />
            <h3>{recipe.title}</h3>
            <p>⏳ Ready in {recipe.readyInMinutes} mins | 🍽 Serves: {recipe.servings}</p>
            <a href={recipe.sourceUrl} target="_blank" rel="noopener noreferrer" className="btn">🔗
View Recipe</a>
          </div>
        ))}
      </div>
    )}
  </div>
)}
```

Add Recipe code

```
@recipe_routes.route("/api/recipes", methods=["POST"])
def add_recipe():
    data = request.get_json()

    if "user_id" not in data:
        return jsonify({"error": "user_id is required"}), 400  # Ensure user_id is included

    try:
        sql = """
        INSERT INTO recipes (name, image, description, servings, prep_time, cook_time, ready_time,
category, user_id)
        VALUES (%s, %s, %s, %s, %s, %s, %s, %s, %s)
        """
        values = (
            data["name"], data["image"], data["description"], data["servings"],
            data["prep_time"], data["cook_time"], data["ready_time"], data["category"],
            data["user_id"]
        )
        cursor.execute(sql, values)
        recipe_id = cursor.lastrowid

        # Insert Ingredients
        for ing in data["ingredients"]:
            cursor.execute("INSERT INTO ingredients (recipe_id, ingredient, metric, us_measure) VALUES
(%s, %s, %s, %s)",
                           (recipe_id, ing["ingredient"], ing["metric"], ing["us"]))

        # Insert Directions
        for step_number, instruction in enumerate(data["directions"], start=1):
            cursor.execute("INSERT INTO directions (recipe_id, step_number, instruction) VALUES (%s,
%s, %s)",
                           (recipe_id, step_number, instruction))

        db.commit()
        return jsonify({"message": "Recipe added successfully!"}), 201

    except Exception as e:
        db.rollback()
        return jsonify({"error": str(e)}), 500
```

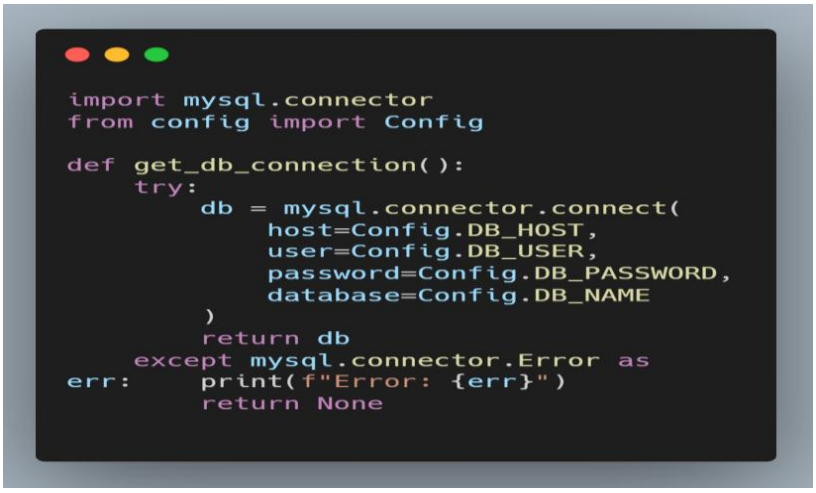
This script creates a Flask route "/api/recipes" that accepts POST requests to create a new recipe in a MySQL database. The function first fetches JSON information from the request and checks whether a user_id is included; otherwise, it returns a 400 error. It then builds an SQL INSERT statement to insert a new recipe into the recipes table, saving information like the name, image, description, servings, preparation time, cooking time, total ready time, category, and the user it was created by. Once the insert is executed, it collects the recipe_id of the inserted recipe. The function next loops through the ingredients list from the request data and inserts every ingredient into the ingredients table and associates them with the recipe_id. Likewise, it loops through the directions list and inserts each step with its respective step number into the directions table. Provided all the insertions are complete, the updates are saved in the database and a success message is sent with a status code of 201. In case of any error, the transaction is reversed to avoid half insertion of the data and an error message with status code 500 is returned.

4.4.3 Database (MySQL)

The MySQL database stores user accounts, saved recipes, and user-submitted recipes.

- Key Features:
  - Stores user authentication data securely.
  - Manages user-generated recipes and interactions.
  - Ensures quick retrieval and filtering of recipes.

Flask MySQL DATABASE CONNECTION CODE

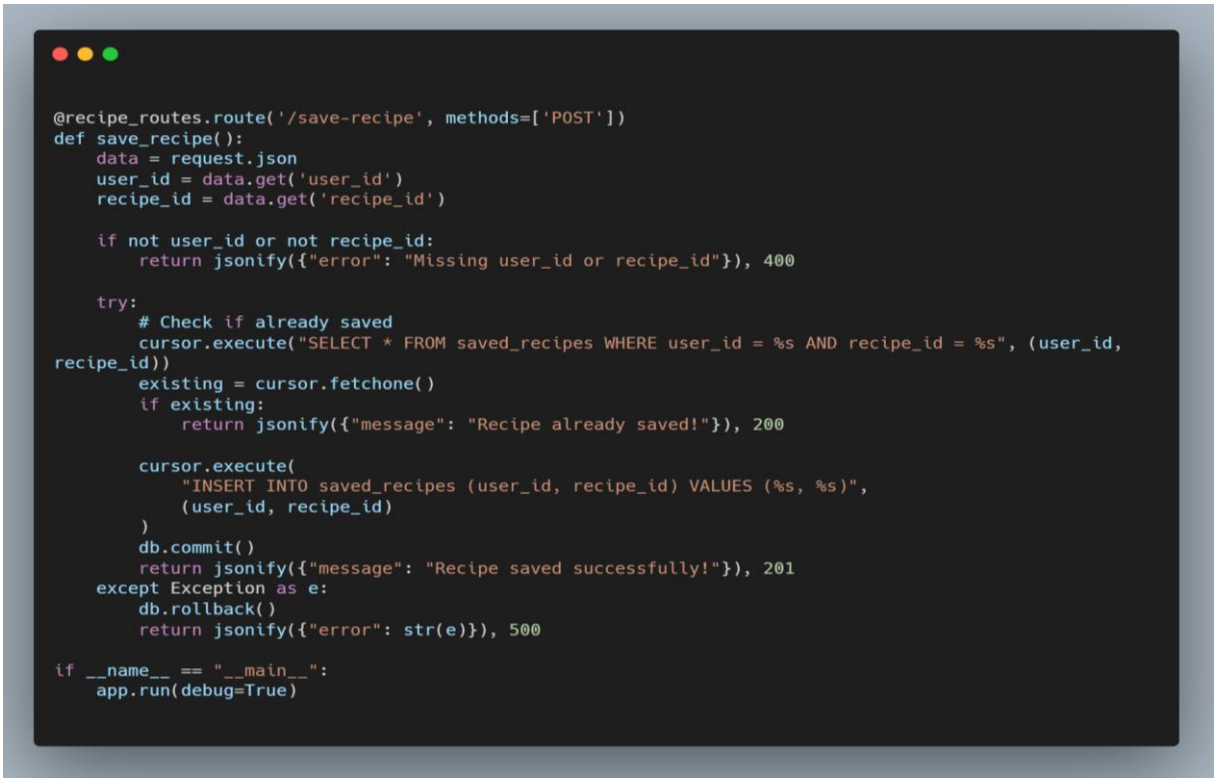


```
import mysql.connector
from config import Config

def get_db_connection():
    try:
        db = mysql.connector.connect(
            host=Config.DB_HOST,
            user=Config.DB_USER,
            password=Config.DB_PASSWORD,
            database=Config.DB_NAME
        )
        return db
    except mysql.connector.Error as
err:     print(f"Error: {err}")
        return None
```

This code creates a function, get_db_connection(), that creates a connection to a MySQL database with the mysql.connector module. The database credentials, such as the host, user,

password, and database name, are imported from a configuration file (Config). The function tries to connect to the database and, if successful, returns the database connection object. If there is a connection error while establishing the connection, it catches the exception, prints an error message, and returns None to allow the application to fail the database connection smoothly.

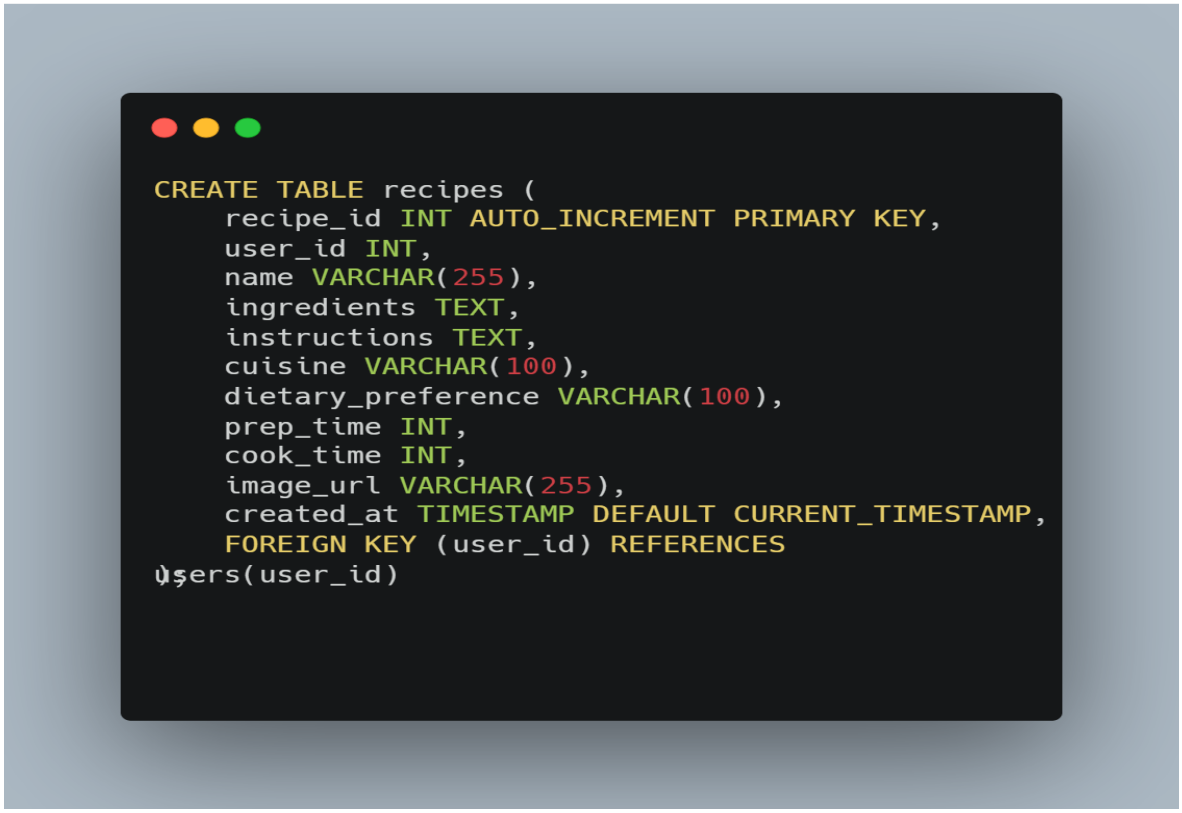
Code to save Favourite Recipe



```
@recipe_routes.route('/save-recipe', methods=['POST'])
def save_recipe():
    data = request.json
    user_id = data.get('user_id')
    recipe_id = data.get('recipe_id')

    if not user_id or not recipe_id:
        return jsonify({"error": "Missing user_id or recipe_id"}), 400

    try:
        # Check if already saved
        cursor.execute("SELECT * FROM saved_recipes WHERE user_id = %s AND recipe_id = %s", (user_id,
recipe_id))
        existing = cursor.fetchone()
        if existing:
            return jsonify({"message": "Recipe already saved!"}), 200

        cursor.execute(
            "INSERT INTO saved_recipes (user_id, recipe_id) VALUES (%s, %s)",
            (user_id, recipe_id)
        )
        db.commit()
        return jsonify({"message": "Recipe saved successfully!"}), 201
    except Exception as e:
        db.rollback()
        return jsonify({"error": str(e)}), 500

if __name__ == "__main__":
    app.run(debug=True)
```

This Flask route allows users to save recipes to their account. It starts by defining the /save-recipe endpoint under the recipe_routes blueprint, accepting only POST requests. The function extracts user_id and recipe_id from the JSON request body, ensuring both are provided; otherwise, it returns a 400 Bad Request error. Next, it checks whether the recipe is already saved by querying the saved_recipes table. If the recipe exists, it returns a 200 OK response indicating that the recipe is already saved. If not, it inserts the user_id and recipe_id into the database and commits the transaction, returning a 201 Created response with a success message. In case of errors, the transaction is rolled back, and a 500 Internal Server Error is returned. Finally, the Flask app runs in debug mode, enabling automatic reload and error tracking. This implementation ensures that users can save recipes without duplication while handling errors efficiently

Example Table Schema for Recipes, Users And Saved Recipes:



```
CREATE TABLE recipes (
    recipe_id INT AUTO_INCREMENT PRIMARY KEY,
    user_id INT,
    name VARCHAR(255),
    ingredients TEXT,
    instructions TEXT,
    cuisine VARCHAR(100),
    dietary_preference VARCHAR(100),
    prep_time INT,
    cook_time INT,
    image_url VARCHAR(255),
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (user_id) REFERENCES
);sers(user_id)
```

```
CREATE TABLE users (
    id INT AUTO_INCREMENT PRIMARY KEY,
    username VARCHAR(50) UNIQUE NOT NULL,
    email VARCHAR(100) UNIQUE NOT NULL,
    password_hash VARCHAR(255) NOT NULL
);eated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (user_id) REFERENCES
);sers(user_id)
```

```
CREATE TABLE saved_recipes (
    id INT AUTO_INCREMENT PRIMARY KEY,
    user_id INT NOT NULL,
    recipe_id INT NOT NULL,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (user_id) REFERENCES users(id) ON DELETE CASCADE,
    FOREIGN KEY (recipe_id) REFERENCES recipes(id) ON DELETE
);ASCADE
```

Fig No: 8

CHAPTER 5

TESTING, DEPLOYMENT & CHALLENGES

5.1 Testing & Debugging

To ensure the reliability and performance of the Online Recipe Finder, the system undergoes thorough testing and debugging. The testing process covers unit testing, integration testing, and UI testing to validate individual components, interactions between services, and the user experience.

5.1.1 Unit Testing

Unit testing focuses on verifying the smallest testable parts of the application, such as individual functions and API endpoints.

Frontend (React.js):

- Tests for correct rendering of UI components.
- Checks if form validations (e.g., login, recipe submission) work correctly.
- Ensures API calls return the expected responses.

Backend (Flask API & MySQL):

- Verifies recipe search API endpoints return accurate data.
- Ensures database queries function properly (CRUD operations).
- Tests user authentication (JWT) to prevent unauthorized access.

5.1.2 Integration Testing & UI Testing

Integration Testing ensures that different components of the application (frontend, backend, database, and API) work together correctly.

- Testing API calls between React.js frontend and Flask backend.
- Verifying database interactions (e.g., saving favorite recipes).
- Ensuring Spoonacular API responses are correctly parsed and displayed.

UI Testing verifies that the user interface is functional and user-friendly.

- Testing navigation flow (e.g., login → search → save recipes).
- Checking form validations (e.g., recipe submission fields).
- Ensuring responsiveness across desktop, tablet, and mobile devices.

Example Unit Test (Flask API):

```python
import unittest
from app import app

class TestAPI(unittest.TestCase):
    def setUp(self):
        self.client = app.test_client()

    def test_search_recipes(self):
        response = self.client.get('/search_recipes?ingredients=tomato,cheese')
        self.assertEqual(response.status_code, 200)
        self.assertIn('recipes', response.json)

if __name__ == '__main__':
    unittest.main()
```

Fig No: 9

5.2 Deployment Process

The deployment of the Online Recipe Finder involves hosting the application on a cloud platform, ensuring continuous integration and delivery, and optimizing performance.

5.2.1 Hosting Platform (Heroku, AWS, etc.)

The backend and database are deployed on a cloud hosting platform, such as:

- **Heroku:** Ideal for Flask applications, offering easy deployment and database management.
- **AWS (EC2, RDS)**: Provides scalability and high availability for the backend and database.
- **Vercel / Netlify:** Used for deploying the React.js frontend.

Deployment Steps:

- **Push Code to GitHub** – Version control & repository management.
- **Set Up Environment Variables** – Secure API keys and database credentials.
- **Deploy Backend (Flask)** – Hosted on Heroku/AWS with API endpoints exposed.
- **Deploy Frontend (React.js)** – Hosted on Vercel or Netlify.
- **Database Configuration** (MySQL) – Connected via AWS RDS or Heroku Postgres.
- **Test & Monitor** – Verify API responses and monitor performance.

5.2.2 CI/CD Pipeline (if applicable)

To ensure smooth continuous integration and deployment (CI/CD), the project can use GitHub Actions or Jenkins.

CI/CD Workflow:

- Code Commit: Developer pushes changes to GitHub.
- Automated Testing: Runs unit & integration tests to detect errors.
- Build Process: Packages frontend & backend applications.
- Deployment: Automatically deploys updates to Heroku/AWS.

5.3 Challenges & Solutions

Challenges	Issues Faced	Solutions Implemented
API Rate Limiting	Spoonacular API had request limits.	Implemented **caching** to reduce API calls.
Database Connection Issues	MySQL connections were timing out.	Used **connection pooling** for efficient queries.
UI Performance Lag	Fetching large datasets caused slow UI.	Added **pagination and lazy loading** for recipes.
Authentication Security	Risk of unauthorized access.	Implemented **JWT authentication** with token expiry.
Deployment Downtime	Server occasionally crashed on high traffic.	Optimized backend **resource management & load balancing**.

Table No: 2

CHAPTER 6

USER GUIDE

This user guide provides step-by-step instructions on how to use the Online Recipe Finder effectively.

6.1 How to Use the Application

The Online Recipe Finder allows users to search for recipes, filter results based on ingredients, save favorite recipes, and submit their own recipes. Users can also manage their profiles and customize their preferences.

Key Features:

- Search for recipes by entering ingredients.
- Save and manage favorite recipes.
- Submit personal recipes to the platform.
- Edit user profiles and preferences.

6.2 Registration & Login Process

Registering a New Account	Logging In
➢ Click on the "Sign Up" button on the homepage. ➢ Enter your name, email, and password. ➢ Click "Register" to create an account. ➢ .confirmation email may be sent for verification.	➢ Click on the "Login" button. ➢ Enter your email and password. ➢ Click "Login" to access your account

Table No: 3

6.3 Searching for Recipes

- **Search by Ingredients**
  - Navigate to the search bar on the homepage.
  - Enter available ingredients (e.g., "chicken, garlic, tomato").
  - Click "Search", and the app will suggest matching recipes.
  - Click on a recipe to view its ingredients, instructions, and nutritional info.

🔹 **Advanced Filtering**

- Filter by cuisine (Italian, Indian, Chinese, etc.).
- Filter by dietary preference (Vegan, Gluten-Free, Keto, etc.).
- Filter by cooking time (Quick meals vs. elaborate dishes).

6.4 Saving Favorite Recipes

🔹 **How to Save Recipes**

- Click the ❤️ (heart) icon on any recipe.
- The recipe is added to your Favorites list.
- Access saved recipes anytime under "My Favorites" in your profile.

🔹 **Removing Saved Recipes**

- Go to "My Favorites" in your profile.
- Click the ❌ (remove) button to delete a saved recipe**.**

6.5 Submitting a Recipe

🔹 **How to Add Your Own Recipe**

- Navigate to "Submit Recipe" in your profile.
- Fill in the required details:
- Recipe name
- List of ingredients
- Step-by-step instructions
- Cooking time & cuisine type
- Upload an image (optional)
- 3Click "Submit" to add the recipe to the database.
- Your recipe will appear in the user-submitted section for others to view.

6.6 User Settings & Profile Management

🔹 **Editing Profile**

- Go to "Profile Settings".
- Update name, email, or password.
- Click "Save Changes".

6.7 Troubleshooting & FAQ

Common Issues & Solutions

Issue	Solution
Forgot Password	Click "Forgot Password" on the login page and follow the instructions.
Recipe Not Displaying	Ensure correct ingredient names are entered in the search bar.
Cannot Save Recipe	Ensure you are logged in before saving recipes.
API Error	Refresh the page; if the issue persists, try again later.
Login Issues	Double-check email/password or reset password if needed.

Table No: 4

Registration, Login, and Recipe Management flowchart

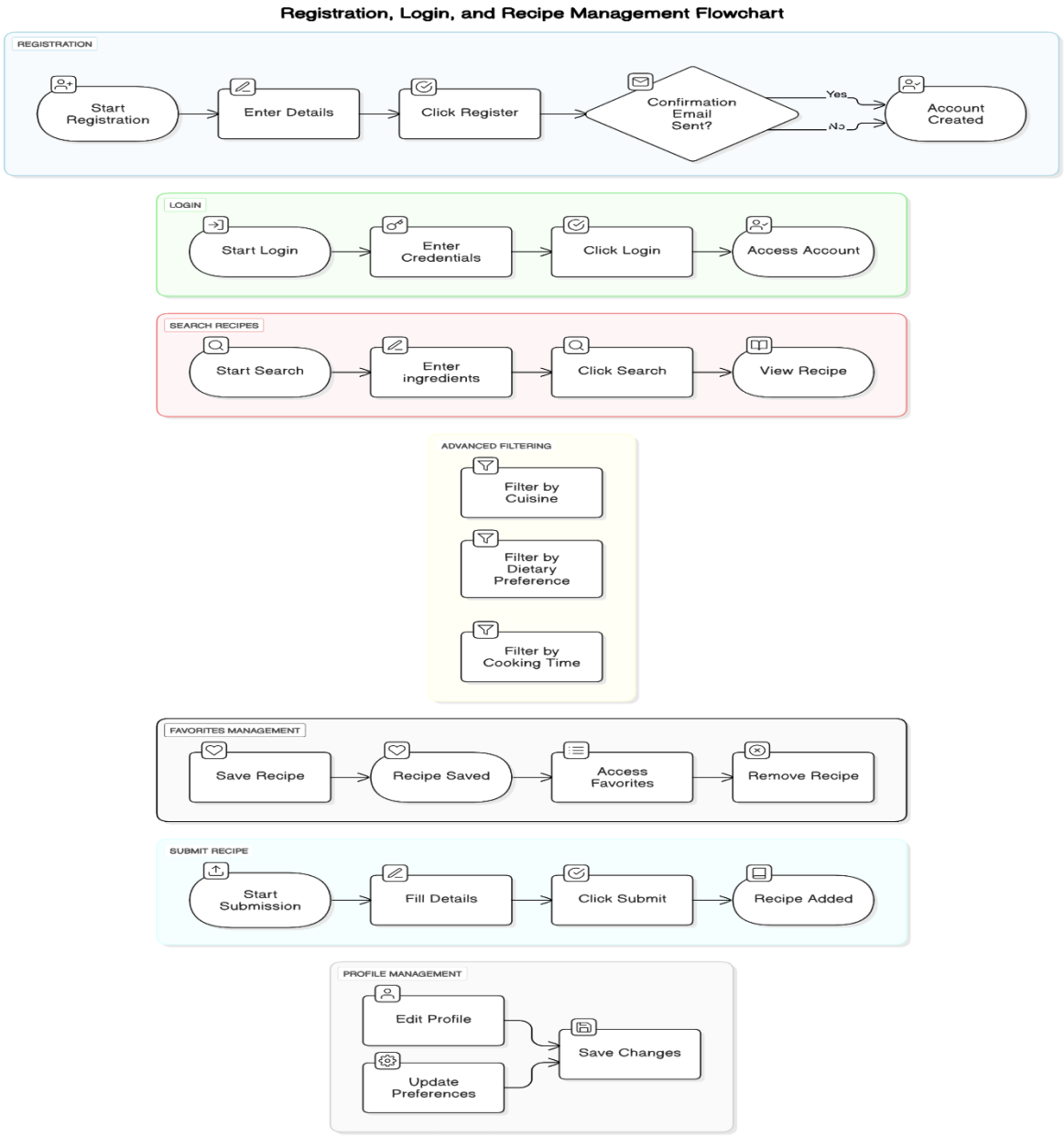


Fig No: 10

Search page & User Profile Page

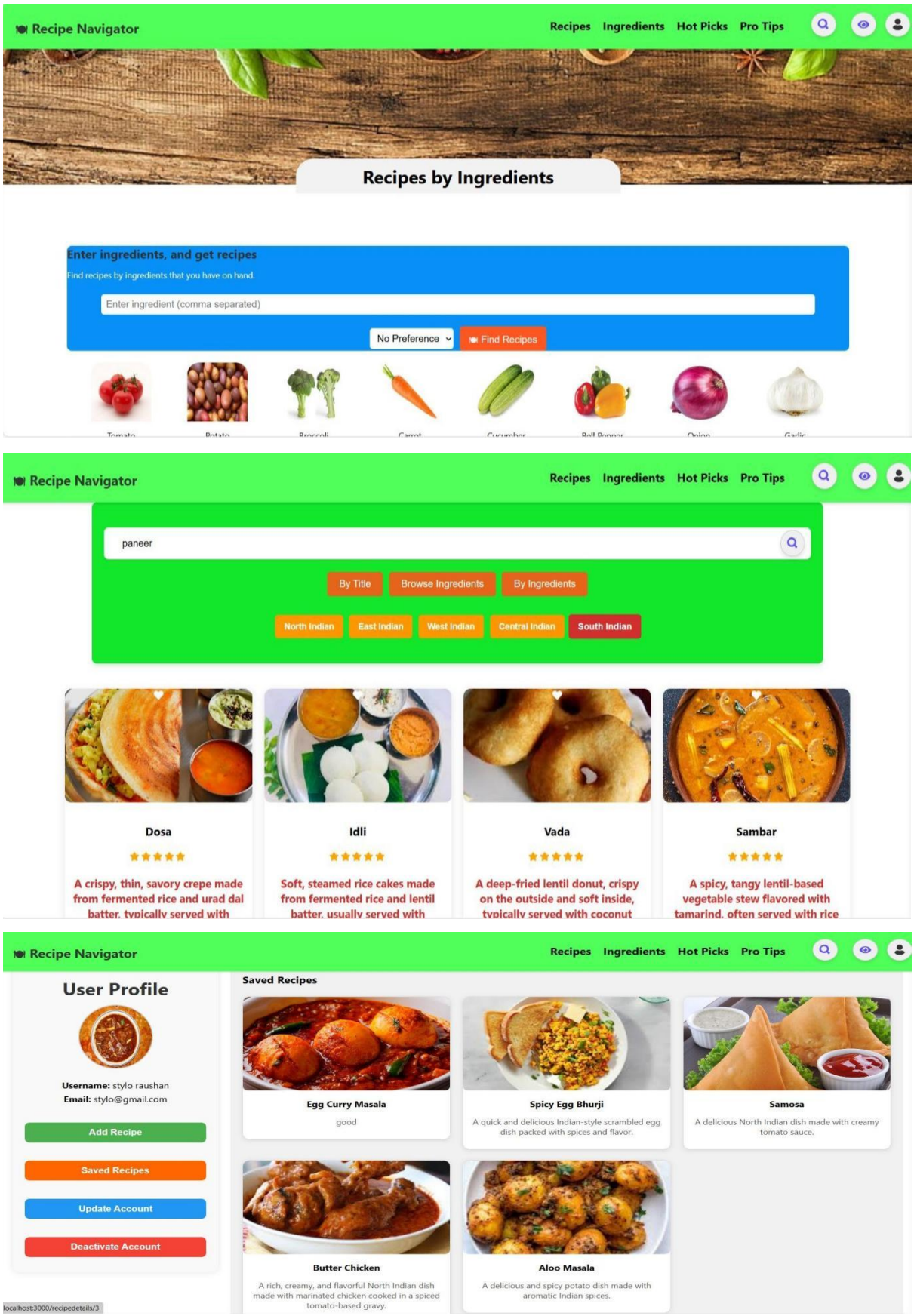
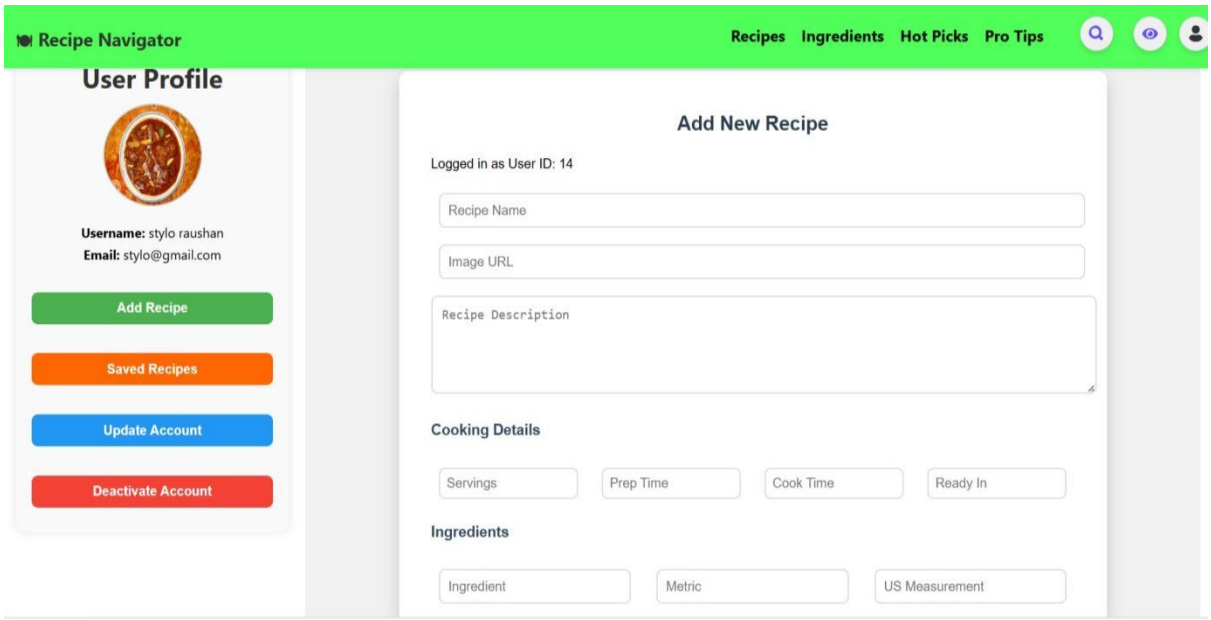
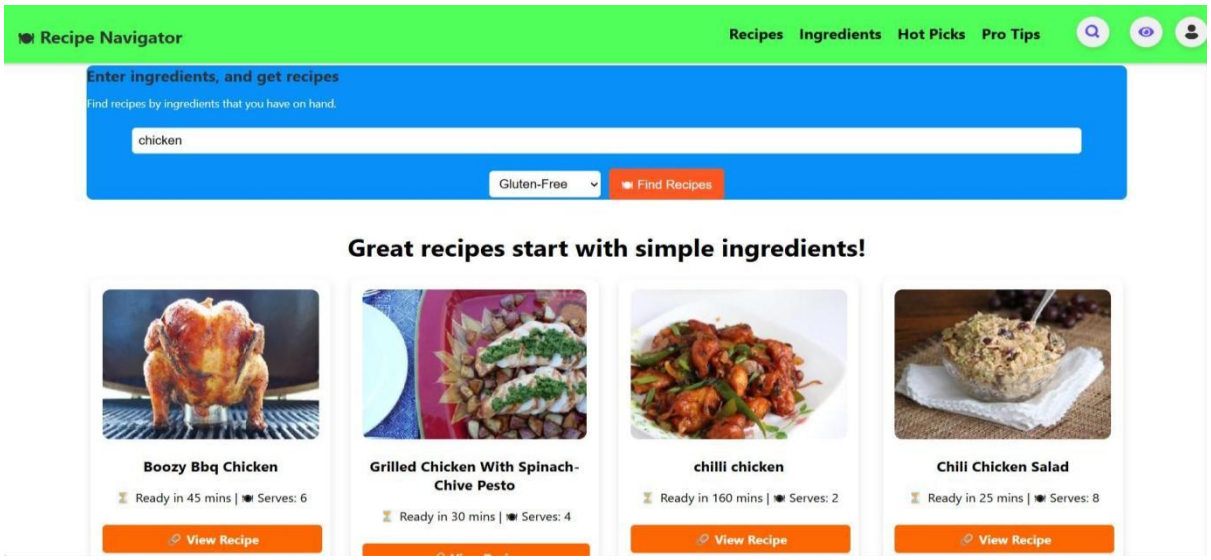


Fig No: 11

Add recipe & fetching recipe & login/Reg page





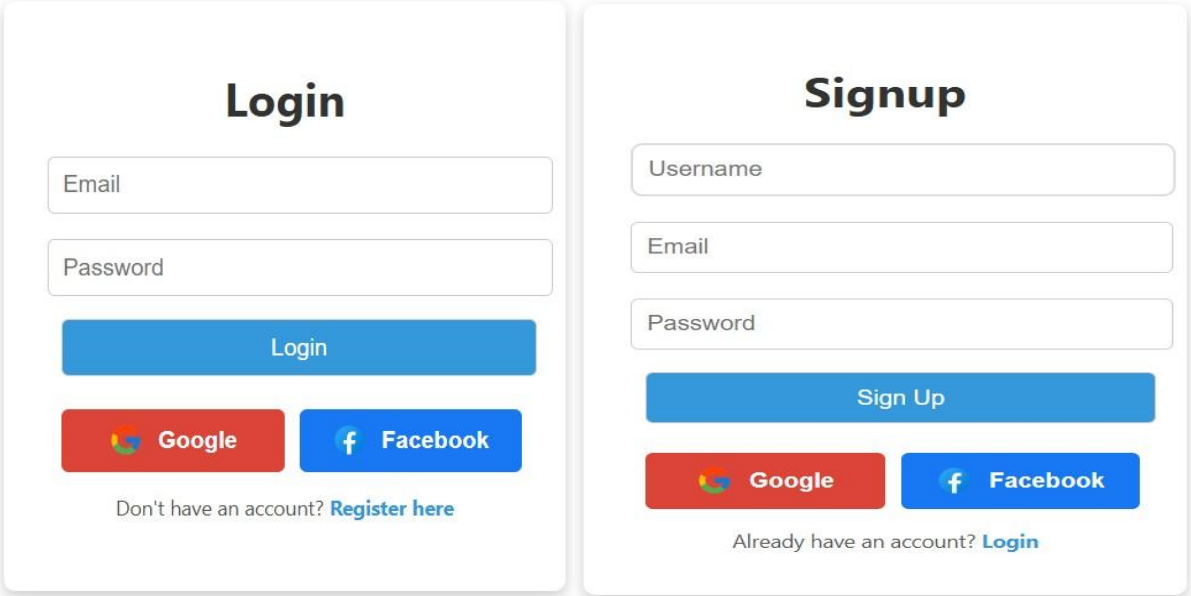


Fig No: 12

Recipe Info page

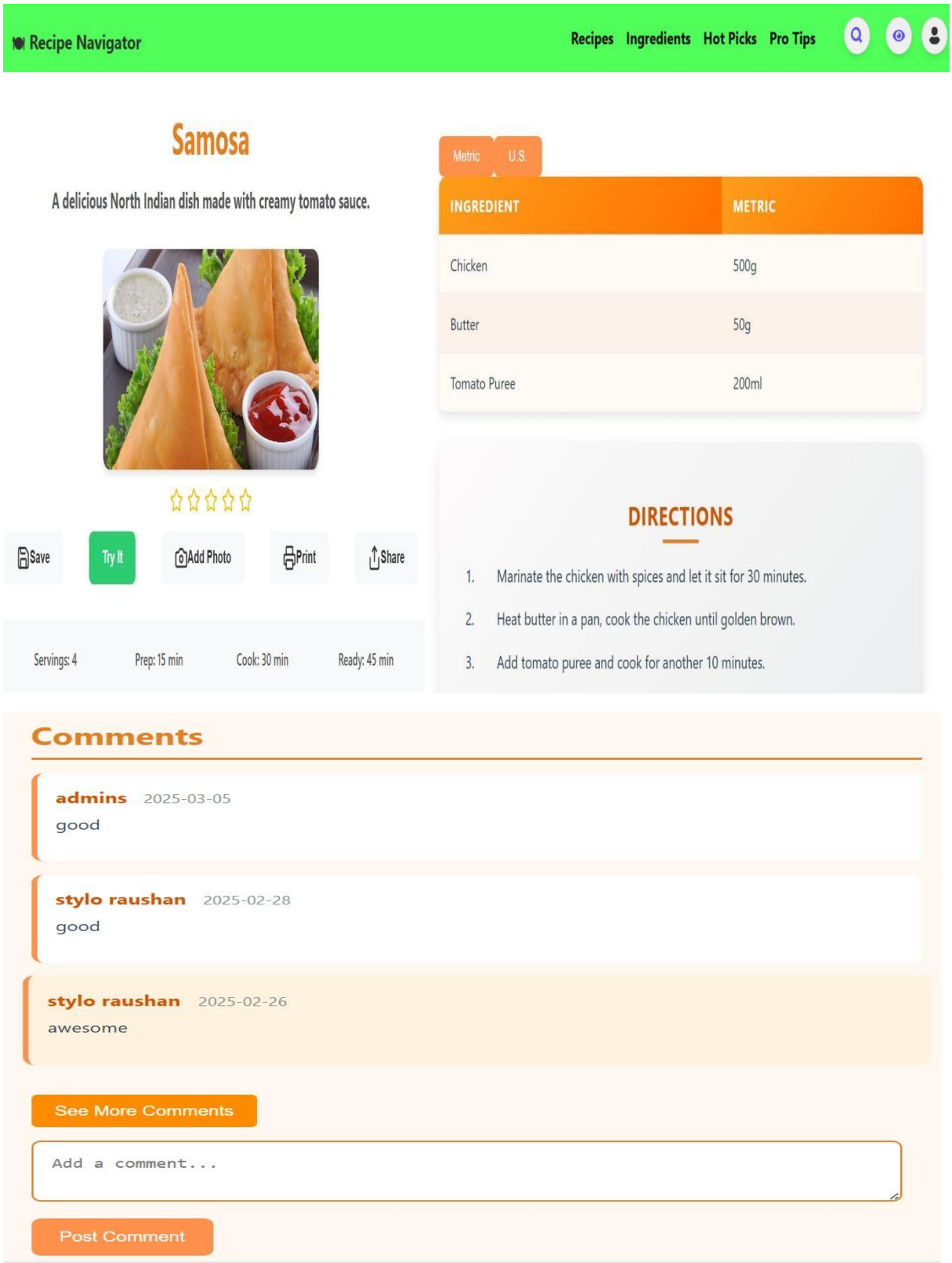


Fig No: 13

CHAPTER 7

CONCLUSION & FUTURE ENHANCEMENTS

7.1 Summary of Project Outcomes

The Online Recipe Finder successfully provides users with a seamless experience in searching, saving, and managing recipes based on available ingredients and dietary preferences. The integration of the Spoonacular API ensures access to a vast database of recipes, while the user-generated content feature enhances community engagement.

7.2 Key Takeaways & Lessons Learned

During the development of the Online Recipe Finder, several technical and project management insights were gained:

Technical Learnings:

API Integration: Implementing the Spoonacular API required optimization to handle API rate limits efficiently.

Frontend-Backend Communication: Using React.js (frontend) and Flask (backend) improved data handling and user interactions.

Database Management: Structuring an optimized MySQL database enhanced data retrieval efficiency.

Security Measures: Implementing JWT authentication and data validation ensured secure user access.

Project Management Takeaways:

Effective Planning: Using the Waterfall Model helped in breaking the development process into structured phases.

Debugging & Optimization: Regular testing helped identify and fix UI/UX performance bottlenecks.

User-Centric Approach: Prioritizing user-friendly design and navigation improved usability.

7.3 Future Enhancements

To further improve the Online Recipe Finder, several enhancements can be implemented:

Feature Enhancements:

- AI-Powered Recipe Recommendations – Implementing machine learning algorithms to suggest recipes based on user preferences and history.
- Voice Search Integration – Allowing users to search recipes using voice commands.

- Meal Planner Feature – Enabling users to plan weekly meals and generate shopping lists.
- Recipe Rating & Reviews – Allowing users to rate recipes and leave detailed reviews.
- Offline Mode – Saving favorite recipes locally for offline access.

Performance & Scalability Improvements:

- Optimize API Calls – Implementing caching mechanisms to reduce Spoonacular API requests.
- Mobile App Development – Expanding the project into a mobile application (Android/iOS) for better accessibility.
- Cloud-Based Hosting – Migrating the backend to a scalable cloud platform for enhanced performance.

7.4 References

1) Spoonacular API Documentation – https://spoonacular.com/food-api/docs
2) React.js Official Documentation – https://react.dev/
3) Flask Framework Documentation – https://flask.palletsprojects.com/
4) MySQL Documentation – https://dev.mysql.com/doc/umen
5) Navathe, Elmasri, Fundamentals of Database Systems, Pearson Education, Inc., California, 2000.
6) TATLI, Ipek, Food Recommendation System Project Report, 2009.
7) [7] Richard Fairley, Software Engineering Concept, Tata McGraw-Hill Education, 2001.
8) Roger S. Pressman, Software Engineering: A Practitioner's Approach (First Edition), 1982.
9) Roger S. Pressman, Software Engineering: A Beginner's Guide, 1988.
10) De Almeida, Jorge Miguel Tavares Soares, Personalized Food Recommendations, 2015.
11) Lee Cheng, Teh; Yusof, Umi; Khalid, Mohd Nor Akmal, Content-Based Filtering Algorithm for Mobile Recipe Application, 2014 8th Malaysian Software Engineering Conference (MySEC 2014).