**FORK & FLAME: Your Virtual Kitchen Assistant**

# Introduction

* **Project Title:** FORK & FLAME
* **Team ID**: **: NM2025TMID38967**
* **Team Leader:**

|  |  |
| --- | --- |
| NAME | MAIL ID |
| RISHI KUMAR E | <rk48076911@gmail.com> |

* **Team Members:**

|  |  |
| --- | --- |
| NAME | MAIL ID |
| PRINCE SAMUEL G S | [samprince14325327@gmail.com](mailto:samprince14325327@gmail.com) |
| YOGARAJ K | <kyogaraj28@gmail.com> |
| CHATRESH K | [chatresh96@gmail.com](chatresh96@gmail.com%20) |
| SATHISH P | <satizsathish05@gmail.com> |

# PROJECT OVERVIEW

**2.1 Purpose:** Fork & Flame connects clients and freelancers through project postings, bidding, and real-time communication.

The goal of this project is to create a comprehensive, user-friendly digital flavour diary platform that offers a variety of recipes. The platform will cater to food enthusiasts by providing easy-to-follow recipes, cooking tips, and nutritional information. It will continuously expand its recipe library to become the go-to resource for home cooks, aspiring chefs, and foodies.

**2.2 Key Features**

* **Recipe Library**: A diverse collection of recipes categorized by cuisine, difficulty, occasion.
* **Visuals**: High-quality photos and step-by-step visual guides, including video content for key steps.
* **Meal Planning Tools**: Option for users to plan meals, save their favorite recipes, and generate shopping lists.
* **SEO-Optimized Content**: Recipe descriptions, titles, and meta tags optimized for search engine visibility.

**3. Target Audience**

* **Home Cooks**: Individuals looking for easy, nutritious, or fun recipes for everyday meals.
* **Foodies & Culinary Enthusiasts**: Users who enjoy experimenting with new ingredients and recipes.
* **Families & Busy Professionals**: Those looking for quick, affordable meals that are easy to prepare.

# 3. ARCHITECTURE

1. **Frontend – React.js with Bootstrap and Material UI**  
   The frontend of *FORK & FLAME* is built using **React.js**, ensuring a highly dynamic and responsive user interface. **Bootstrap** provides a robust grid system and layout framework, while **Material UI** adds modern, pre-designed components for consistency and visual appeal. Together, they deliver a smooth, intuitive, and interactive user experience across devices.
2. **Backend – Node.js and Express.js**  
   The backend is powered by **Node.js** for fast, scalable, and event-driven server operations. **Express.js** is used to handle routing, server logic, and API endpoints, ensuring efficient communication between the frontend and database. This combination supports secure data processing and reliable application performance.
3. **Database – MongoDB**  
   **MongoDB** serves as the primary database, chosen for its flexibility in handling unstructured data. It stores user information, project postings, application details, and chat messages in a secure and scalable manner. Its document-oriented structure makes it well-suited for real-time interactions and large-scale data management.

# 4. SETUP INSTRUCTIONS

* 1. **Prerequisites**

Before setting up ***Fork & Flame*** ensure that the following tools and technologies are installed on your system:

* + **Node.js** – JavaScript runtime environment.
  + **MongoDB** – Database for storing user data, projects, and chat messages.
  + **Git** – For cloning and version control.
  + **React.js** – Frontend framework for building the user interface.
  + **Express.js** – Backend framework for handling APIs and server logic.
  + **Mongoose** – ODM (Object Data Modeling) library for MongoDB.
  + **Visual Studio Code** – Recommended IDE for development.
  1. **Installation steps**
     1. **Node.js**
* **Knowledge**: basic JavaScript (variables, functions, promises/async-await), basic command-line use (terminal / PowerShell), JSON, and git (recommended).
* **System**: Windows / macOS / Linux (x64 or arm64); 2GB+ RAM recommended; internet to download installers.
* **Tools to have ready**: a terminal (CMD/PowerShell/WSL on Windows, Terminal on macOS/Linux), a code editor (VS Code recommended).

**Step-by-step installation**

**Windows — easiest: official installer (or use npm-windows for version management)**

**Official installer (quick)**

1. Visit the official Node.js downloads page and download the Windows **.msi** (choose the **LTS** build).
2. Run the MSI and accept defaults (it will install node and npm and add to PATH).
3. Open PowerShell or Command Prompt and verify:

node -v

npm -v

you’re done. [nodejs.org](https://nodejs.org/en/download?utm_source=chatgpt.com)

**If you want multiple Node versions / safer global installs — use npm for Windows**

1. Install **npm-windows** (download and run the installer from the npm-windows releases).
2. After install, open a new admin PowerShell or CMD and use:

npm install lts

npm list

npm use <version> # e.g. npm use 18.16.0 OR `npm use lts` if supported

node -v

npm-windows lets you switch versions without re-installers. [GitHub+1](https://github.com/coreybutler/nvm-windows?utm_source=chatgpt.com)

npm -v

you’re done.

* + 1. **MongoDB**
  + **Knowledge Prerequisites**
  + Basic **database concepts**: collections, documents, CRUD (Create, Read, Update, Delete).
  + Basic **command line/terminal** usage.
  + JSON knowledge (MongoDB stores data in BSON, similar to JSON).
  + (Optional) Networking basics: ports, IP binding, authentication.

**System Prerequisites**

* + **OS**: Windows (x64/arm64 supported).
  + **Processor**: 64-bit, 1 GHz+ recommended.
  + **RAM**: Minimum 2 GB (4 GB+ recommended for dev; higher for prod).
  + **Disk space**: At least 5–10 GB free (MongoDB stores large amounts of data).
  + **Port**: MongoDB runs on 27017 by default (make sure it’s free/unblocked).

**Software Prerequisites**

* + **Administrator** (to install packages/services).
  + **Package manager** (apt, yum, Homebrew) OR direct installer.
  + **MongoDB Shell (mongosh)** – required to interact with MongoDB.
  + **MongoDB Compass (GUI)** – optional but recommended for beginners.
  + **Docker** (optional) – for containerized setup.

**4.2.3.Git**

**System prerequisites**

* + **OS**: Windows
  + **Permissions**: Administrator (Windows)
  + **Network**: Internet access (for cloning/pushing remote repos like GitHub, GitLab, Bitbucket).

**Software prerequisites**

* + Text/code editor (VS Code recommended).
  + SSH client (optional, for GitHub/GitLab SSH connections).

**Github account creation:**

### 1. **Sign in to GitHub**

* Go to [GitHub](https://github.com).
* Log in to your GitHub account. If you don't have one, you can sign up for free.

### 2. **Create a New Repository**

* Once logged in, click on the **+** icon in the top right corner of the page.
* Select **"New repository"** from the dropdown menu.

### 3. **Fill in Repository Details**

You'll need to fill out the following fields:

* **Repository Name**: Choose a name for your repository (e.g., my-project).
* **Description** (optional): Provide a brief description of your project (e.g., "A cool web app").
* **Public or Private**: Choose whether your repository will be public (anyone can see it) or private (only you and collaborators can access it).
* **Initialize with README**: If you want to add a README file (which is helpful for explaining your project), check this box.
* **Add .gitignore** (optional): You can choose a template for your .gitignore file, depending on your project's language or framework (e.g., Python, Node, etc.).
* **Choose a License** (optional): You can choose a license if you'd like to specify how others can use your code.

### 4. **Create the Repository**

* Once you've filled everything out, click the **Create repository** button.

### 5. **Clone the Repository (Optional)**

* After the repository is created, you'll be taken to the new repository's page.
* To clone it to your local machine, copy the **clone URL** (either HTTPS or SSH) from the "Code" button.
* In your terminal, run the following command:
* git clone<repository-url>

### 6. **Start Adding Files**

* Now you can start adding files to your repository either through the GitHub website or by pushing files from your local machine.

That's it! You've created a new GitHub repository. Let me know if you need help with the next steps, like pushing files to GitHub.

* + 1. **React.js**

**System prerequisites**

* + **OS**: Windows
  + **Node.js & npm**:
  + Install latest **LTS version of Node.js** (includes npm).
  + Verify with:
  + node -v
  + npm -v
  + **Code Editor**: VS Code recommended.
  + **Browser**: Chrome/Edge/Firefox for developer tools.
  + A.Using Create React App (CRA) [Beginner Friendly]
  + Open terminal → navigate to project folder.
  + Run:
  + npx create-react-app my-app
  + (Here npx comes with npm 5.2+, so no extra install needed.)
  + Go into project folder:
  + cd my-app
  + Start development server:
  + npm start
  + 👉 App runs at <http://localhost:3000>.
    1. **Express.js – Mongoose – Visual Studio Code**
  + **Prerequisites (Theory)**
  + **JavaScript & Node.js** → needed since Express runs on Node.
  + **MongoDB knowledge** → Mongoose is an ODM for MongoDB.
  + **REST API concepts** → to design routes/endpoints.
  + **Visual Studio Code** → editor to write and manage the project.
  + **Node.js & npm installed** → to run server and install packages.
  + **MongoDB installed or Atlas account** → for database.
* **Installation Steps:**

# Clone the repository git clone

# Install client dependencies cd client npm install

# Install server dependencies cd

../server npm install

# FOLDER STRUCTURE

Fork & Flame-Work/

│

├── client/ # React Frontend

│ ├── components/ # Reusable UI components

│ └── pages/ # Application pages and views

│

├── server/ # Node.js Backend

│ ├── routes/ # API endpoint definitions

│ ├── models/ # Mongoose schemas for MongoDB

│ └── controllers/ # Business logic and request handling

This structure ensures a clear separation between the **frontend (React)** and **backend (Node.js + Express)**, making development and maintenance more efficient.

# RUNNING THE APPLICATION

The document provides commands to run a frontend and backend application:

* **Frontend**:
  + **cd client**: This command likely navigates to the frontend directory of the project.
  + **npm start**: Starts the frontend application using Node Package Manager (npm).
* **Backend**:
  + **cd server**: Navigates to the backend server directory.
  + **npm start**: Starts the backend server using npm.

**Accessing the Application**:

* The application can be accessed via http://localhost:3000, which is a local address where the frontend application can be viewed in a browser.

# API DOCUMENTATION

API documentation serves as a vital part of any application or service, especially when it comes to integrating with backend systems. It provides a clear and structured guide on how to interact with the application through its **API endpoints**. In this case, the API documentation appears to outline how to interact with user management, project handling, chat functionalities, authentication, and routing security. Here's a breakdown:

### **1. API Endpoints:**

The endpoints listed in the documentation represent various functionalities provided by the backend of the application. Each endpoint corresponds to a specific action or resource. Here's a closer look at what the documentation suggests:

#### ****User Endpoints:****

* **/api/user/register**
  + **Purpose**: This endpoint is for registering new users. It likely requires some user data (e.g., email, password, name) to create a new account.
  + **HTTP Method**: Typically, a POST request is used here, sending user data to the server to create a new account.
  + **Response**: It could return a success message or an error message if the registration fails (e.g., email already exists).
* **/api/user/login**
  + **Purpose**: This endpoint handles user login. The user submits credentials (email and password), and if the login is successful, a session token (such as a JWT) is returned.
  + **HTTP Method**: A POST request is commonly used here.
  + **Response**: The response typically includes a **JWT token** or a session identifier if the login is successful. If the credentials are invalid, an error message will be returned.

#### ****Project Endpoints:****

* **/api/projects/create**
  + **Purpose**: This endpoint allows users to create new projects within the system. It would typically involve sending project details (such as project title, description, and other parameters).
  + **HTTP Method**: A POST request, sending data in the body to create the project.
  + **Response**: Returns a confirmation message or the details of the newly created project.
* **/api/projects/:id**
  + **Purpose**: Fetch details of a specific project based on its ID.
  + **HTTP Method**: A GET request is usually used to retrieve the details of the project.
  + **Response**: It returns the project data (such as title, description, and other attributes) or an error if the project ID doesn't exist.
* **/api/projects/:id/apply**
  + **Purpose**: Allows a user to apply to a project, which means associating the user with the project in some way (e.g., applying for a job or requesting involvement).
  + **HTTP Method**: A POST request would be used to submit the application.
  + **Response**: Confirmation of the application or error if something goes wrong.

# USER INTERFACE

The **User Interface (UI)** section of a Fork & Flame application provides a description of how the user will interact with the app or website. It includes the structure, design elements, and features that allow users to search for recipes, view ingredients, and create a meal plan, among other actions.

**8.1. Landing Page**

**Overview**: The landing page is the first thing users see when they visit the flavour diary app. It should provide a welcoming experience, easy navigation, and access to key functionalities.

**Key Elements**:

* **Logo/Brand Name**: Clear visibility of the app’s name and logo.
* **Search Bar**: Allows users to search for recipes by ingredients, cuisines, or meal types.
* **Category Navigation**: Displays popular categories like “Breakfast,” “Lunch,” “Dinner,” or “Desserts.”
* **Featured Recipes**: A carousel or grid showcasing trending or recommended recipes.

#### ****. 8.2. Recipe Search Page****:

* **Overview**: This page allows users to search for recipes by ingredients, cuisine, or type of meal. It is designed to help users find what they need quickly.
* **Key Elements**:
  + **Search Filters**: Filters for cuisine type, cooking time, difficulty, and dietary restrictions (e.g., vegetarian, gluten-free).
  + **Recipe Cards**: Each recipe is displayed with a brief title, image, and cooking time. Clicking on a recipe card takes the user to the recipe details page.
  + **Sort Options**: Sorting options based on popularity, ratings, or most recent.

#### ****8.3.RecipeDetails Page****:

* **Overview**: This page shows the full details of a selected recipe, including ingredients, instructions, and optional nutritional information.
* **Key Elements**:
  + **Recipe Title**: The name of the recipe.
  + **Image Gallery**: A section showing high-quality images of the finished dish and step-by-step pictures.
  + **Ingredients List**: A list of required ingredients with quantities.
  + **Cooking Instructions**: Detailed steps for preparing the dish.
  + **Difficulty & Time**: Information on the complexity of the recipe and estimated cooking time.
  + **User Reviews and Ratings**: A section where users can leave ratings and comments.
  + **Save to Favorites**: Option for users to save the recipe for future reference.

### **UI Design Considerations**:

#### ****1. Visual Design****:

* **Colors**: Use warm, inviting colors, typically associated with food (e.g., orange, red, green) to create an appetizing and user-friendly interface.
* **Typography**: Use legible fonts for easy reading of recipe instructions and ingredients.
* **Images**: High-quality images of food that engage the user and inspire them to try out new recipes.

#### ****2. User Experience (UX)****:

* **Responsive Design**: Ensure the UI adapts to different screen sizes, from desktop to mobile.
* **Navigation**: Easy-to-use navigation menus that lead users to the most important sections (search, meal planner, profile).
* **Interactive Elements**: Buttons should be clearly defined for receipes.

#### ****3. Accessibility****:

* Ensure that the UI is accessible to all users, including those with disabilities. Use appropriate color contrasts, readable font sizes, and alt text for images.

**9. TESTING**

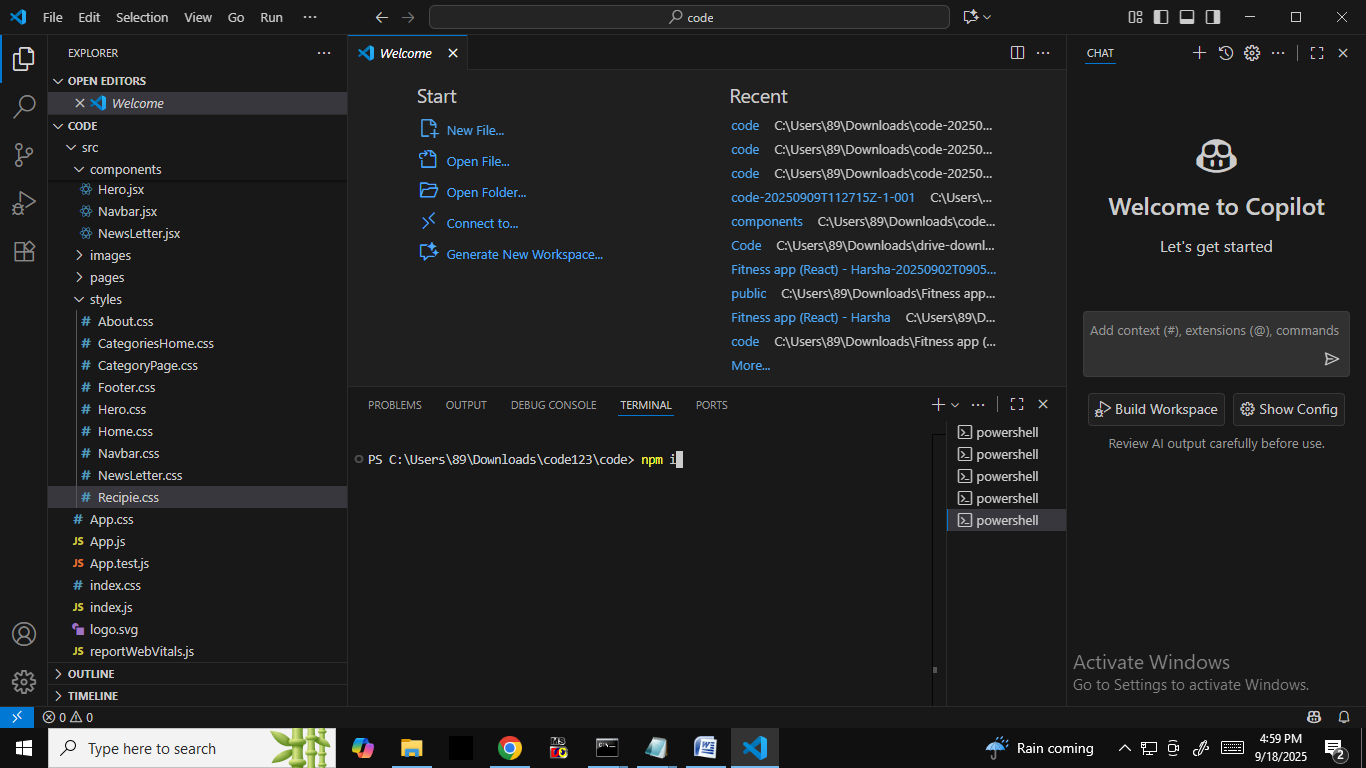
**Manual testing during milestones**

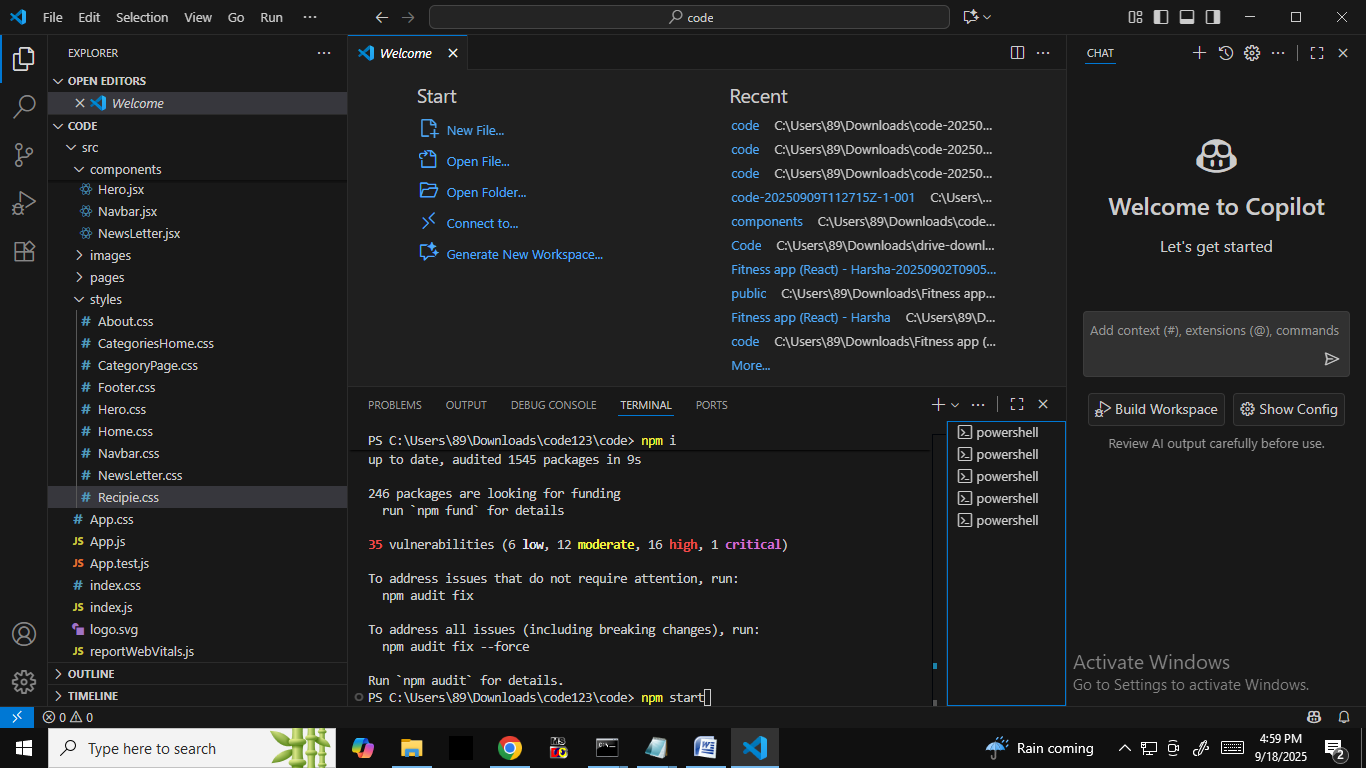
### Manual Testing

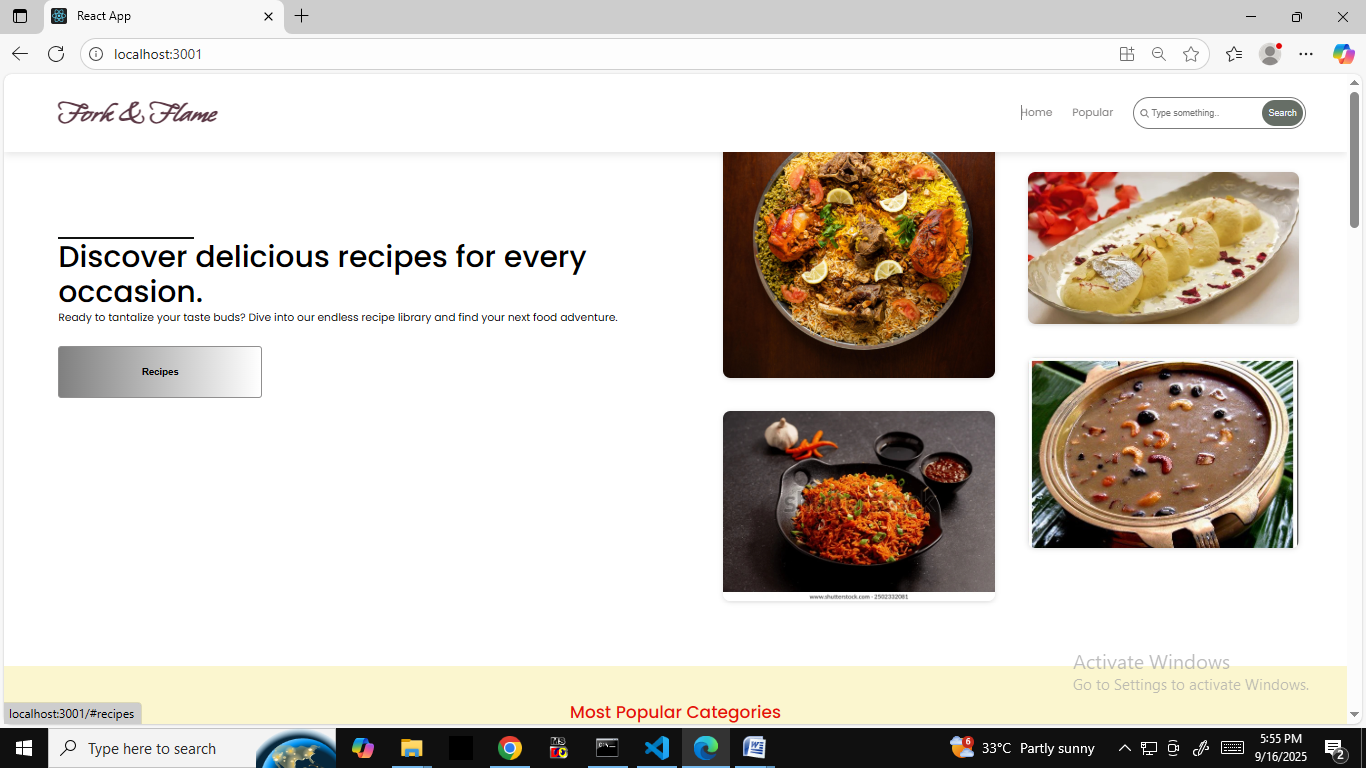
Manual testing is a type of software testing where human testers manually execute test cases to find defects and ensure the software meets its requirements. This process relies on human observation, creativity, and critical thinking.

The general steps in manual testing include:

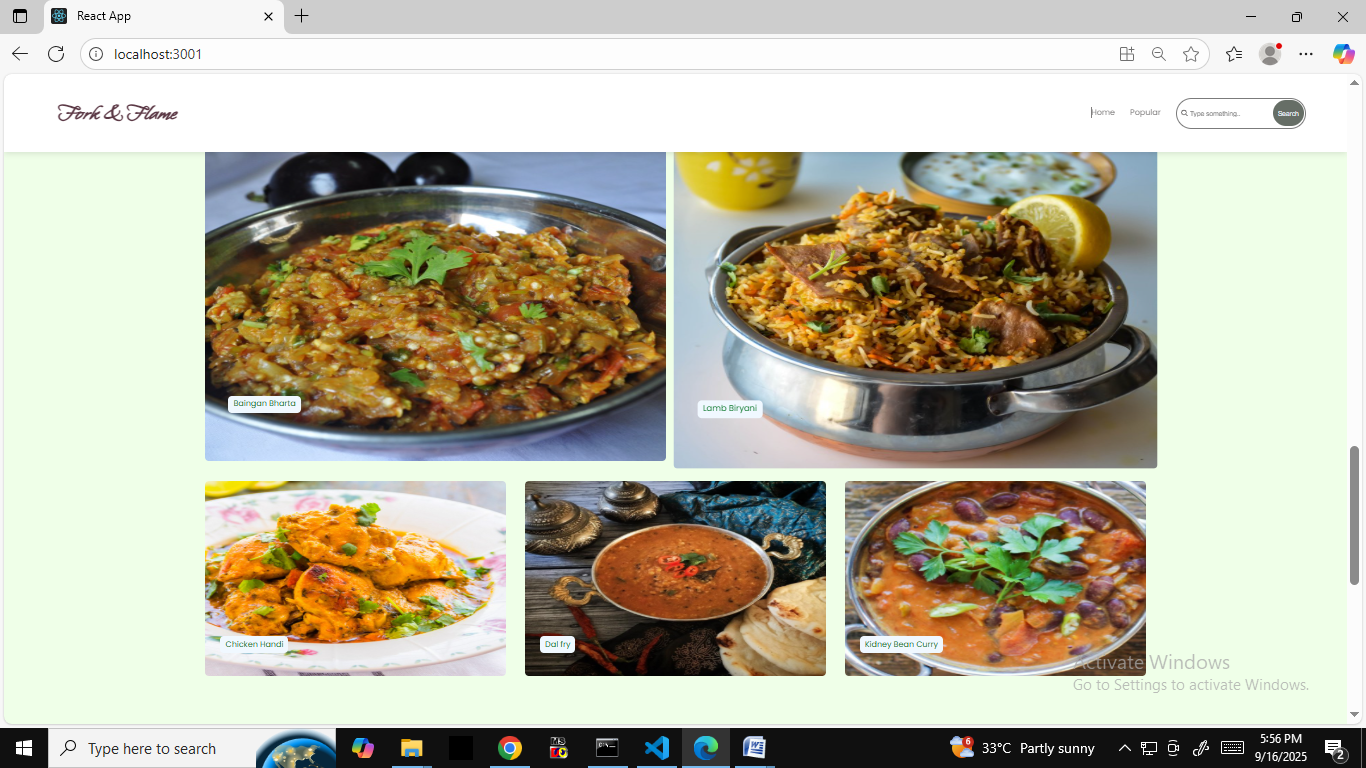
1. **Understanding Requirements**: Reviewing and understanding all software requirements, user stories, and design files.
2. **Creating a Test Pla**n: Developing a roadmap for the entire testing process, including objectives, scope, and resources.
3. **Writing Test Case**s: Creating detailed step-by-step instructions for testing specific features and scenarios.
4. **Setting up the Test Environment**: Installing the necessary software, databases, and other tools to perform the tests.
5. **Executing Test Cases:** Manually interacting with the software to verify it behaves as expected and documenting the results.
6. **Logging Defects:** Reporting any issues or bugs found during testing into a defect tracking tool.
7. **Retesting and Regression Testing:** Re-checking the areas where bugs were fixed and running regression tests to ensure new issues were not introduced.
8. **Analyzing Results and Reporting**: Reviewing the entire process, documenting findings, and creating a final report.
9. **SCREENSHOTS**

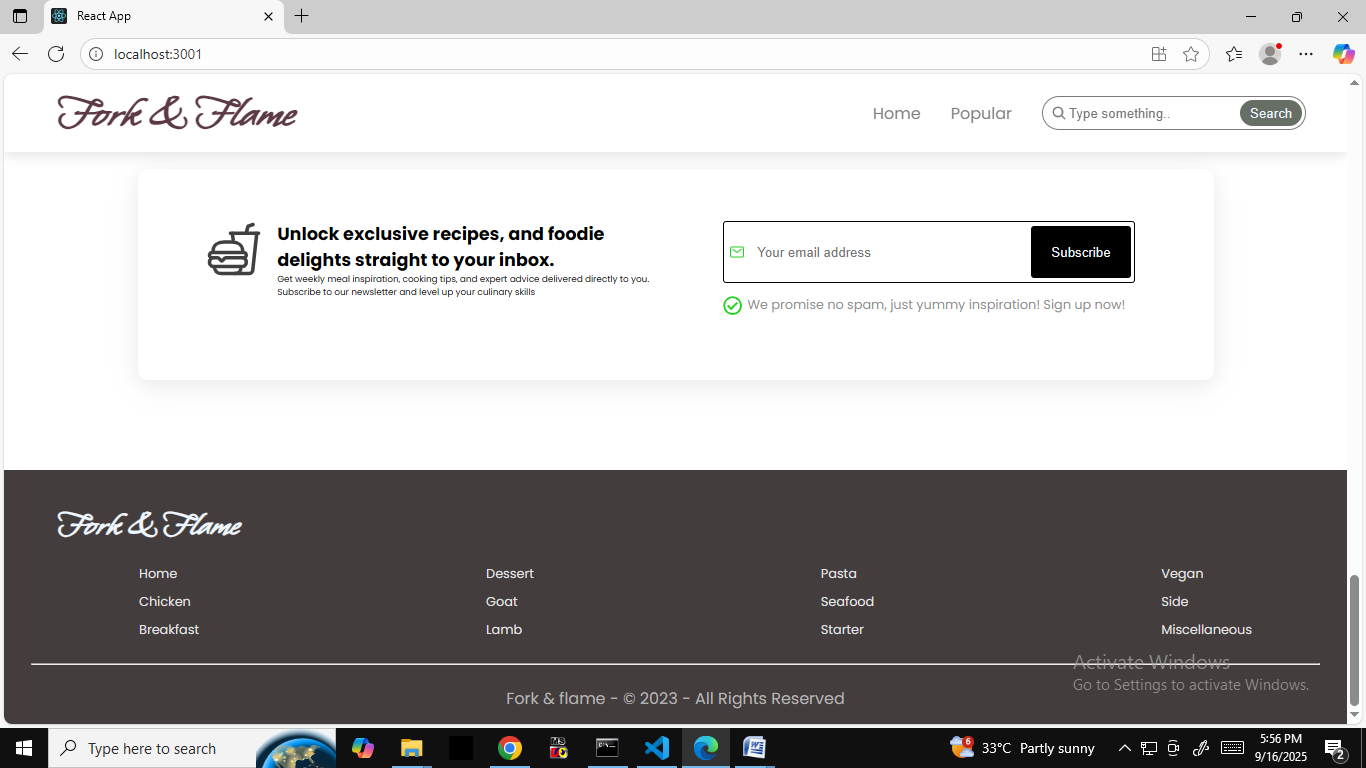


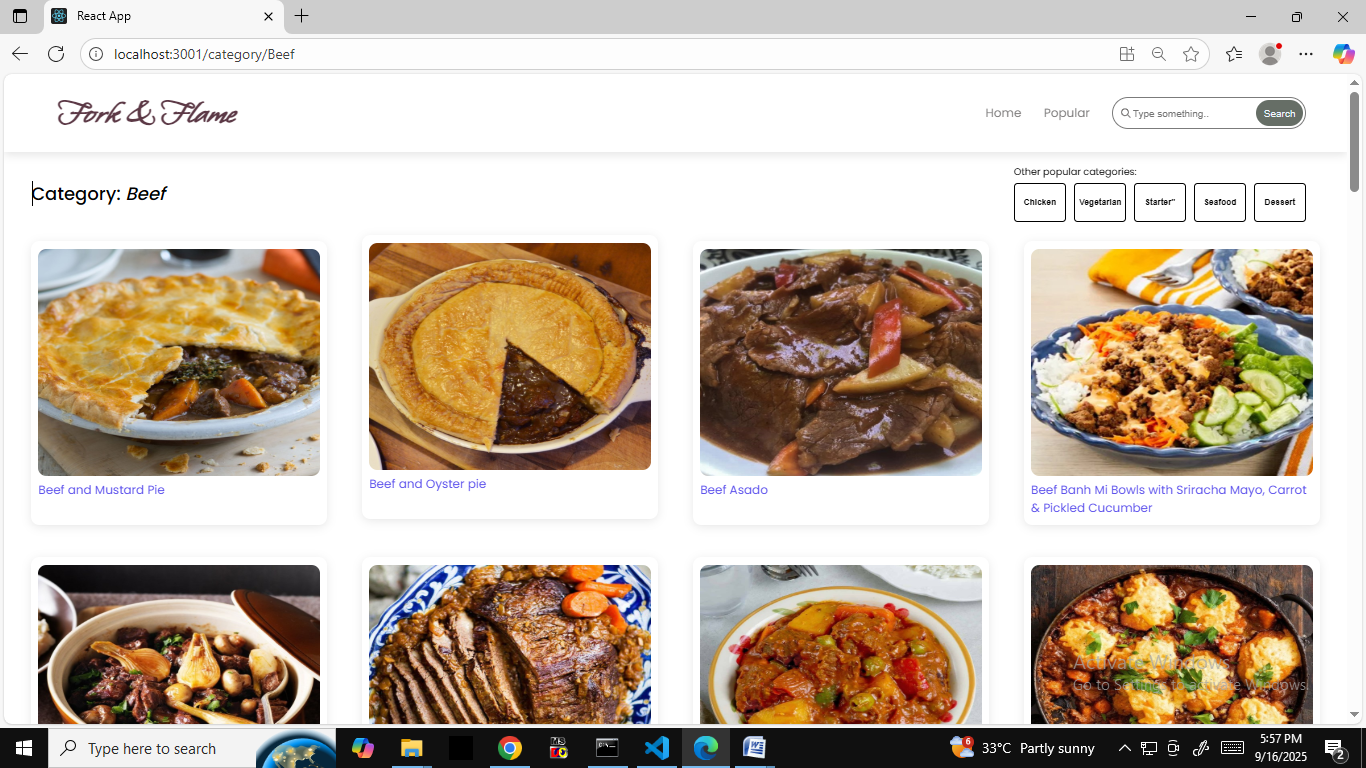


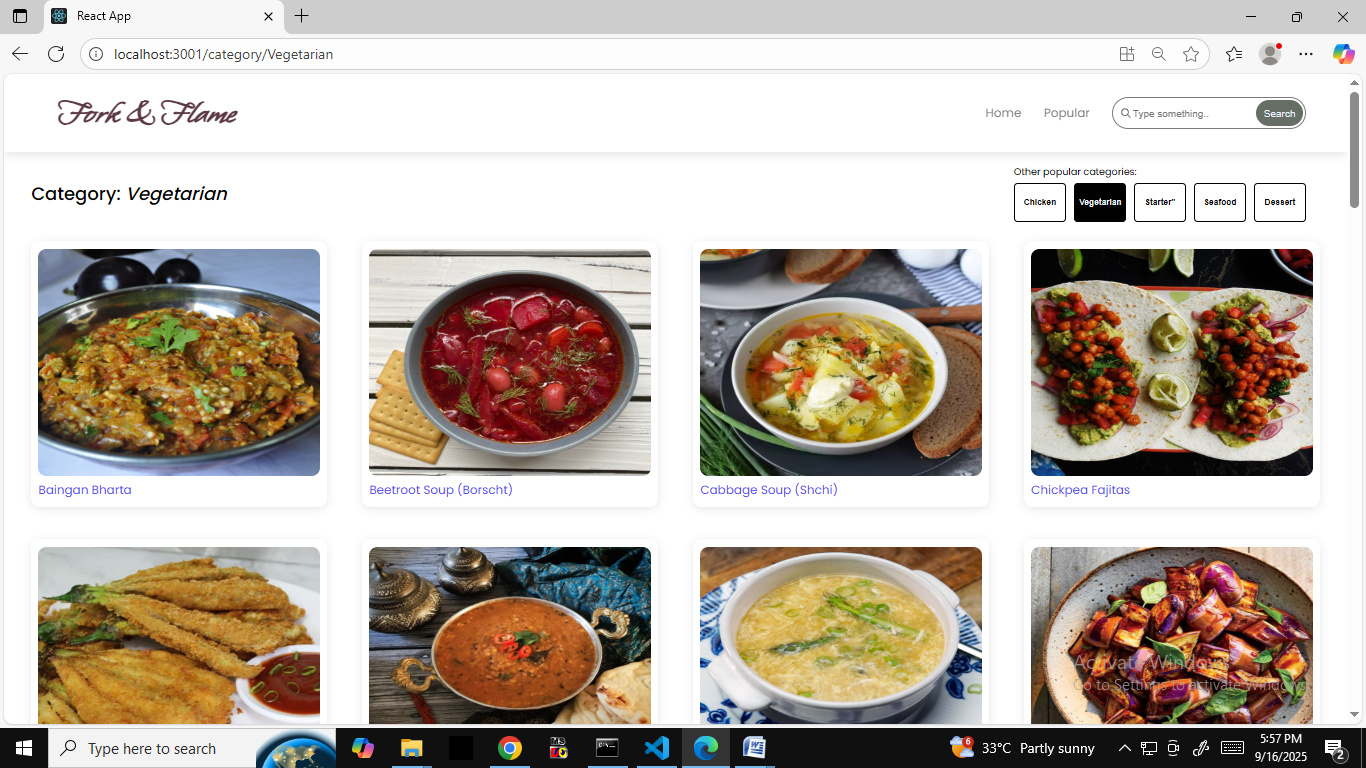






****

****

****

1. **KNOWN ISSUES :**

This section highlights some of the common challenges currently faced by our Fork & Flame website.

Our Fork & Flame website is currently facing a few challenges related to content . One of the main issues is the limited number of recipes available, which can make the site feel incomplete or underdeveloped. In addition, while SEO optimization is essential for improving visibility, finding the right balance is difficult. Search engines prefer detailed, keyword-rich content, but over-optimization can lead to unnatural or overly commercial writing that detracts from the user experience. Furthermore, maintaining a steady flow of high-quality content is resource-intensive. Writing detailed recipes, including ingredients, instructions, and tips, requires significant time and effort, and the continuous addition of fresh content demands ongoing resources. We are actively working to resolve these issues by expanding our recipe collection and fine-tuning our content strategy.

# 12. FUTURE ENHANCEMENTS

To improve the overall user experience and expand the functionality of our cookbook website, several key enhancements are planned. **User Contributions** will be introduced, allowing users to submit their own recipes, review others, and provide feedback, fostering a community-driven approach to content creation. We’ll also enhance **Search and Filtering** capabilities, making it easier for users to find recipes based on ingredients, time, difficulty, and other preferences. Additionally, we aim to cater to **health-conscious individuals** by offering specialized recipes for specific diets such as keto, vegan, and gluten-free.

Content growth will be streamlined through **Batch Content Creation**, enabling us to add multiple recipes at once, and by encouraging **User-Generated Content**, with a moderation system in place. To improve **SEO**, we plan to implement AI-assisted tools for optimization, structured data (Schema Markup) for better search visibility, and a **Topic Clustering** strategy to improve internal linking. Efficiency will also be a priority, with the introduction of **Templated Formats**, a **Content Calendar**, and **Collaboration Tools** for smoother content production workflows.

User engagement will be enhanced through features like a **Comment & Rating System**, **Save & Share Options**, and the inclusion of **Step-by-Step Visuals** or videos for key recipe steps. We’ll also leverage **Analytics** to track popular recipes and gain insights via **User Surveys** to guide content priorities. The introduction of a **Meal Planner** will allow users to organize meals, generate shopping lists, and receive personalized recipe suggestions. Finally, an upgraded **Admin Panel** and **User Profile Page** will offer better management of content, user accounts, and personal recipe preferences, ensuring a more customized and interactive experience.

* 1. **CONCLUSION**

The Flavour Diary project successfully establishes a digital platform designed to inspire, guide, and support food enthusiasts, home cooks, and culinary learners through an accessible recipe-sharing and meal-planning experience. By integrating a robust frontend with React.js, a scalable backend powered by Node.js and Express.js, and a flexible MongoDB database, the application ensures a seamless, secure, and user-friendly interaction.

The platform’s key strengths lie in its diverse recipe library, intuitive UI, and tools that simplify meal planning while fostering engagement through features such as interactive recipe exploration. Although challenges remain, such as expanding recipe content and balancing SEO optimization with natural content flow, the groundwork laid in this version provides a solid foundation for continuous improvement.

Ultimately, Flavour Diary has demonstrated the potential to become a comprehensive digital companion for modern cooking. With planned enhancements such as user contributions, advanced filtering, AI-assisted content optimization, and personalized meal planners, the application is poised to evolve into a vibrant, community-driven ecosystem that enriches everyday cooking experiences.