**My Recipe Book**

**Team ID : SWTID1741244104158473**

**TEAM MEMBERS :**

**Team Size : 5**

**Team Leader :** R vinodha ( email : [vinodharamadhass@gmail.com](mailto:vinodharamadhass@gmail.com) )

**Team member :** A sheela rani ( email : [sheelaraniselvi@gmail.com](mailto:sheelaraniselvi@gmail.com) )

**Team member :** B sathya ( email : [sathyaviji135@gmail.com](mailto:sathyaviji135@gmail.com) )

**Team member :** R saranya ( email : [saranyarameshsaranya5@gmail.com](mailto:saranyarameshsaranya5@gmail.com) )

**Team member :** C sripriya ( email : [sripriyap65@gmail.com](mailto:sripriyap65@gmail.com) )

# INTRODUCTION

The given HTML, CSS, and JavaScript code represents a Recipe Book Web Application that allows users to browse different categories of recipes, such as meals, drinks, and desserts. Each recipe includes ingredients, preparation steps, and images. The project uses HTML for structure, CSS for styling, and JavaScript for interactivity (if needed in future development).

My Recipe Book is a web application designed to allow users to browse various categories of recipes, including meals, drinks, and desserts. Each recipe provides detailed information such as ingredients, preparation steps, and images. The goal is to create an intuitive and user-friendly interface where users can easily navigate and manage their favorite recipes.

The primary goal of this application is to provide users with an easy-to-navigate interface to view and manage their favorite recipes. The design follows a structured format with a homepage, category pages (meals, drinks, desserts), and individual recipe pages.

# **Languages Used :**

1. **HTML (HyperText Markup Language)**

HTML is used to create the structure of the web pages. It defines elements such as headings, paragraphs, images, lists, links, and buttons. In this project, multiple HTML files are used to organize different recipe categories and pages.

# CSS (Cascading Style Sheets)

CSS is used to style the website and enhance the user experience. It controls the layout, colors, fonts, and responsive behavior of the pages. This project uses external CSS (style.css) to maintain consistency across different pages.

# JavaScript (Planned but Not Implemented Yet)

Although the provided code does not contain JavaScript, it can be used to add interactivity, such as dynamically displaying ingredients, filtering recipes, or adding animations.

# PROJECT OVERVIEW

# **1. Purpose**

# The My Recipe Book web application is designed to help users easily browse, view, and manage different types of recipes, including meals, drinks, and desserts. The main goal is to provide a well-structured, visually appealing, and user-friendly platform where users can access detailed recipes, including ingredients and preparation steps. This project aims to make cooking more convenient by offering an organized digital recipe collection.

# **2. Features**

# ✔ User-Friendly Interface: A simple and clean design that allows easy navigation between different recipe categories.

# ✔ Recipe Categories: Recipes are divided into categories such as meals, drinks, and desserts, making it easy for users to find what they need.

# ✔ Detailed Recipe Pages: Each recipe includes:

# Ingredients List: Users can check off ingredients as they prepare the dish.

# Preparation Steps: Step-by-step instructions for making the dish.

# Images: Visual representation of the dish for reference.

# ✔ Navigation System: Users can smoothly switch between different pages with a structured menu and navigation buttons.

# ✔ Responsive Design: The website is designed to be accessible on different devices, including desktops and mobile phones.

# **3. Architecture Overview**

# The application will follow a component-based architecture using React.js. It will consist of:

# UI Components (Navbar, Recipe Card, etc.)

# Pages (Homepage, Meals, Drinks, Desserts, Recipe Details)

# State Management (for handling recipes and user interactions)

# Routing (for navigation between pages

# **Component Structure**

# Main Components & Their Roles

# 1. App.js – The main component that includes the router and manages the overall structure.

# 2. Navbar.js – A reusable navigation bar for switching between categories.

# 3. Home.js – Displays the homepage with links to recipe categories.

# 4. CategoryPage.js – Displays a list of recipes for each category (Meals, Drinks, Desserts).

# 5. RecipeDetails.js – Shows detailed information about a selected recipe.

# 6. RecipeCard.js – A reusable component to display recipe previews.

# 7. Footer.js – A footer component for branding or additional links.

# **2.State Management**

# For managing state, we can use Context API or Redux, depending on complexity.

# Context API Approach (Recommended for Simplicity)

# A RecipeContext.js file will store recipe data and allow access throughout the app.

# useContext hook will be used in components like CategoryPage.js and RecipeDetails.js to fetch data dynamically.

# Redux Approach (For Advanced State Management)

# Actions & reducers will manage recipe data globally.

# Useful if you plan to add user authentication or dynamic recipe storage in the future.

# **3.Routing Structure (Using React Router)**

# Navigation between different pages will be handled using react-router-dom.

# Example Routes in App.js

# import { BrowserRouter as Router, Route, Routes } from 'react-router-dom';

# import Home from './pages/Home';

# import CategoryPage from './pages/CategoryPage';

# import RecipeDetails from './pages/RecipeDetails';

# function App() {

# return (

# <Router>

# <Routes>

# <Route path="/" element={<Home />} />

# <Route path="/category/:type" element={<CategoryPage />} />

# <Route path="/recipe/:id" element={<RecipeDetails />} />

# </Routes>

# </Router>

# );

# }

# export default App;

# Routing Explanation

# / → Homepage

# /category/:type → Displays recipes for meals, drinks, or desserts based on type.

# /recipe/:id → Shows details of a specific recipe using a dynamic id.

# **4.SETUP INSTRUCTIONS**

# **1. Prerequisites**

# Before setting up the project, ensure you have the following installed:

# ✅ For the current static version (HTML, CSS, JS):

# A web browser (Chrome, Firefox, Edge, etc.)

# A code editor (VS Code, Sublime Text, etc.)

# ✅ If using React & Node.js (for future development):

# Node.js (LTS version) – Required for running a React project.

# Git – For cloning the project from a repository.

# VS Code – Recommended code editor.

# **2. Installation & Setup**

# (A) Setup for the Current Static Version (HTML, CSS, JavaScript)

# Download the Project Files

# If you have the project as a ZIP file, extract it to a preferred location.

# If using GitHub, clone the repository:

# git clone <repository\_url>

# 2. Open the Project in a Browser

# Locate index.html.

# Right-click and open it in a browser (Chrome, Edge, Firefox, etc.).

# The website will load and work as expected.

# 3.Editing the Code (Optional)

# Open the project folder in VS Code.

# Modify HTML, CSS, or JS files as needed.

# (B) Setup for a React Version (If Upgrading)

# 1. Clone the Repository

# git clone <repository\_url>

# cd my-recipe-book

# 2. Install Dependencies

# npm install

# 3.Start the Development Server

# npm run dev

# 4.Open in Browser

# Navigate to http://localhost:5173/ (if using Vite for React).

# **5. FOLDER STRUCTURE**

# 1. Client: Organizing the React Application

# (a) Components (src/components/)

# These are reusable UI components:

# Navbar.js – Navigation bar

# RecipeCard.js – Displays a preview of a recipe

# Footer.js – Footer section

# (b) Pages (src/pages/)

# Each page represents a section of the app:

# Home.js – Displays homepage

# CategoryPage.js – Lists meals, drinks, or desserts

# RecipeDetails.js – Displays a full recipe

# (c) Context API for State Management (src/context/)

# RecipeContext.js – Stores and manages recipe data globally

# 2. Utilities: Helper Functions & Custom Hooks

# (a) Helper Functions (src/utils/)

# formatDate.js – Formats dates (if timestamps are used)

# truncateText.js – Shortens long text descriptions

# (b) Custom Hooks (src/hooks/)

# useFetch.js – A hook for fetching recipes from an API

# **RUNNING THE APPLICATION**

# **Running the Application Locally**

# For the Current Static Version (HTML, CSS, JavaScript)

# Download the project files (or clone the repository).

# Open the project folder.

# Right-click on index.html → Open with Browser.

# ✅ The project will run without any server setup.

# **For the React Version (If Upgrading)**

# Install Dependencies

# npm install

# 2. Start the Development Server

# npm run dev

# Open the App in Browser

# Once the server starts, open:

# http://localhost:5173/

# (If using Vite for React development)

# **COMPONENT DOCUMENTATION**

# **1. Key Components**

# These are the main components in the My Recipe Book Web Application and their roles:

# App.js – The main application component that manages routing and overall structure.

# Navbar.js – A navigation bar that provides links to different recipe categories. It may receive a currentPage prop to highlight the active section.

# Home.js – Displays the homepage with category links such as Meals, Drinks, and Desserts.

# CategoryPage.js – Lists recipes based on the selected category. It receives a categoryType prop to display the relevant recipes (e.g., "Meals", "Drinks").

# RecipeDetails.js – Displays the full recipe details, including ingredients, preparation steps, and an image. It takes a recipeId prop to fetch the correct recipe.

# Footer.js – A simple footer that contains links or branding information.

# **2. Reusable Components**

# These components are used multiple times throughout the application for consistency and reusability:

# RecipeCard.js – A reusable card component that displays a preview of a recipe, including an image, title, and short description. It accepts props like title, image, description, and onClick.

# Button.js – A customizable button component that can be styled differently based on props like label, onClick, and type (e.g., primary or secondary button).

# LoadingSpinner.js – Displays a loading animation when data is being fetched.

# ErrorMessage.js – Shows an error message when something goes wrong, with a message prop for customization.

# **STATE MANAGEMENT**

# **Global State Management**

# For managing state across the entire application, you can use Context API or Redux.

# Using Context API (Recommended for Simplicity)

# Context API allows global state sharing across multiple components without prop drilling.

# The RecipeContext.js file will manage the list of recipes and user-selected items.

# **2. Local State Management**

# Local state is used inside individual components for UI interactions, such as toggling checkboxes or handling search filters.

# **USER INTERFACE**

# **Homepage Screenshot**

# Show the main landing page with navigation links to different recipe categories (Meals, Drinks, Desserts).

# Highlight any interactive elements like hover effects on category links.

# **Category Page Screenshot**

# Display a grid/list of available recipes under a selected category.

# Include recipe images, titles, and brief descriptions.

# **Recipe Details Page Screenshot**

# Show a full recipe with ingredients, preparation steps, and an image.

# Capture checkbox interactions where users can mark ingredients as used.

# **Navigation Bar Screenshot**

# Show the navbar with links to different pages.

# If the active page is highlighted, include that in the screenshot.

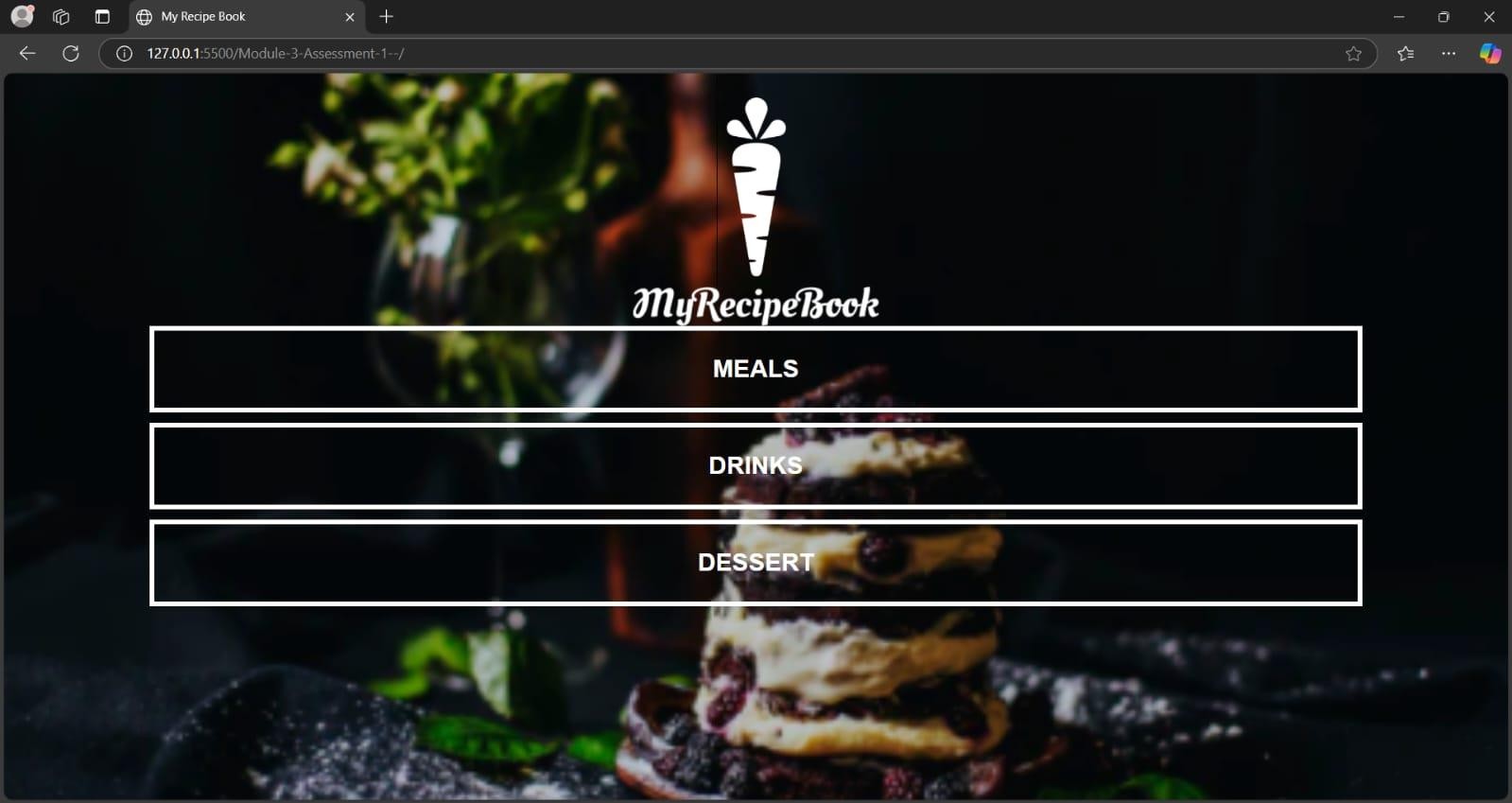
# **Mobile Responsive View**

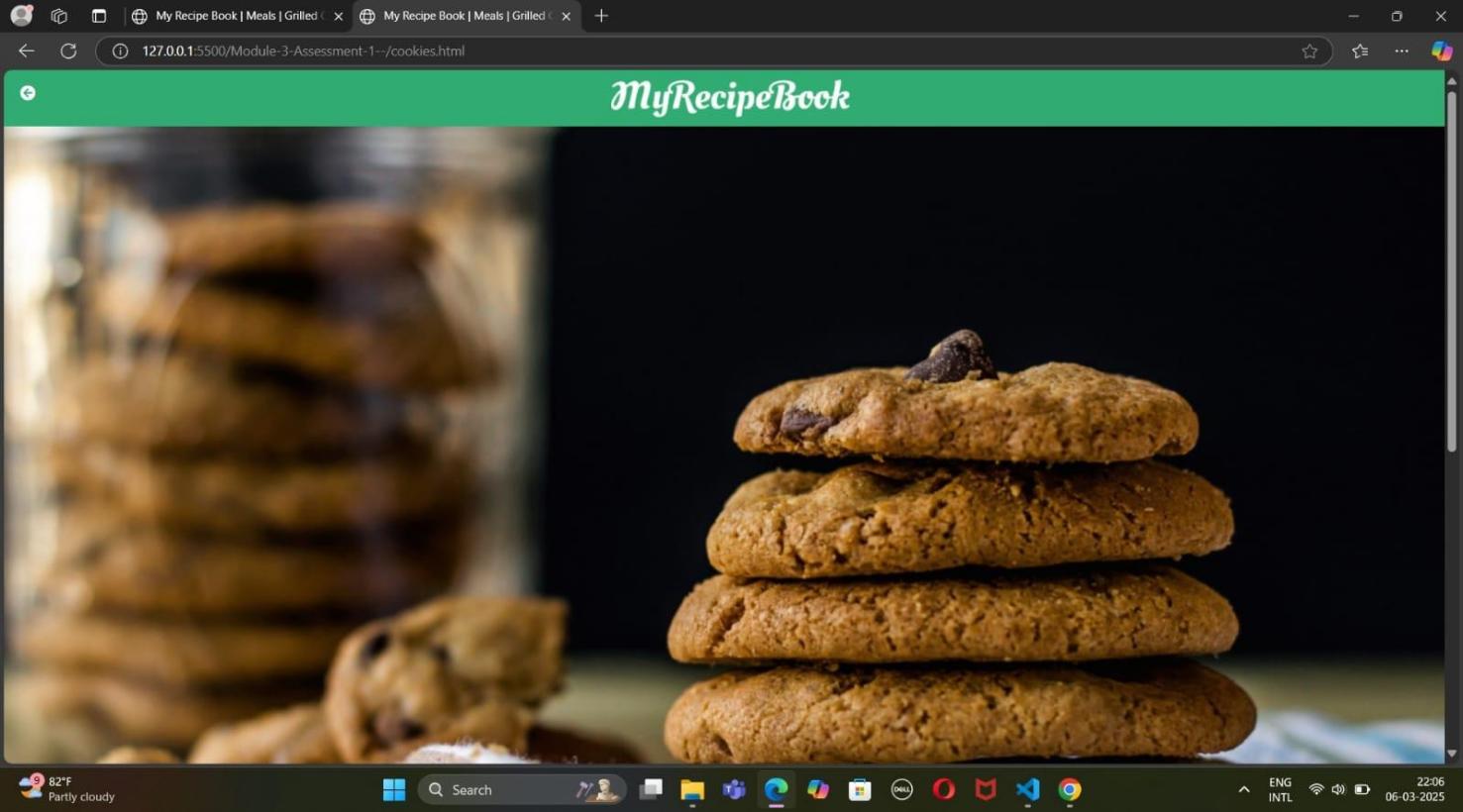
# Take a screenshot of how the app looks on a mobile device to showcase responsiveness.

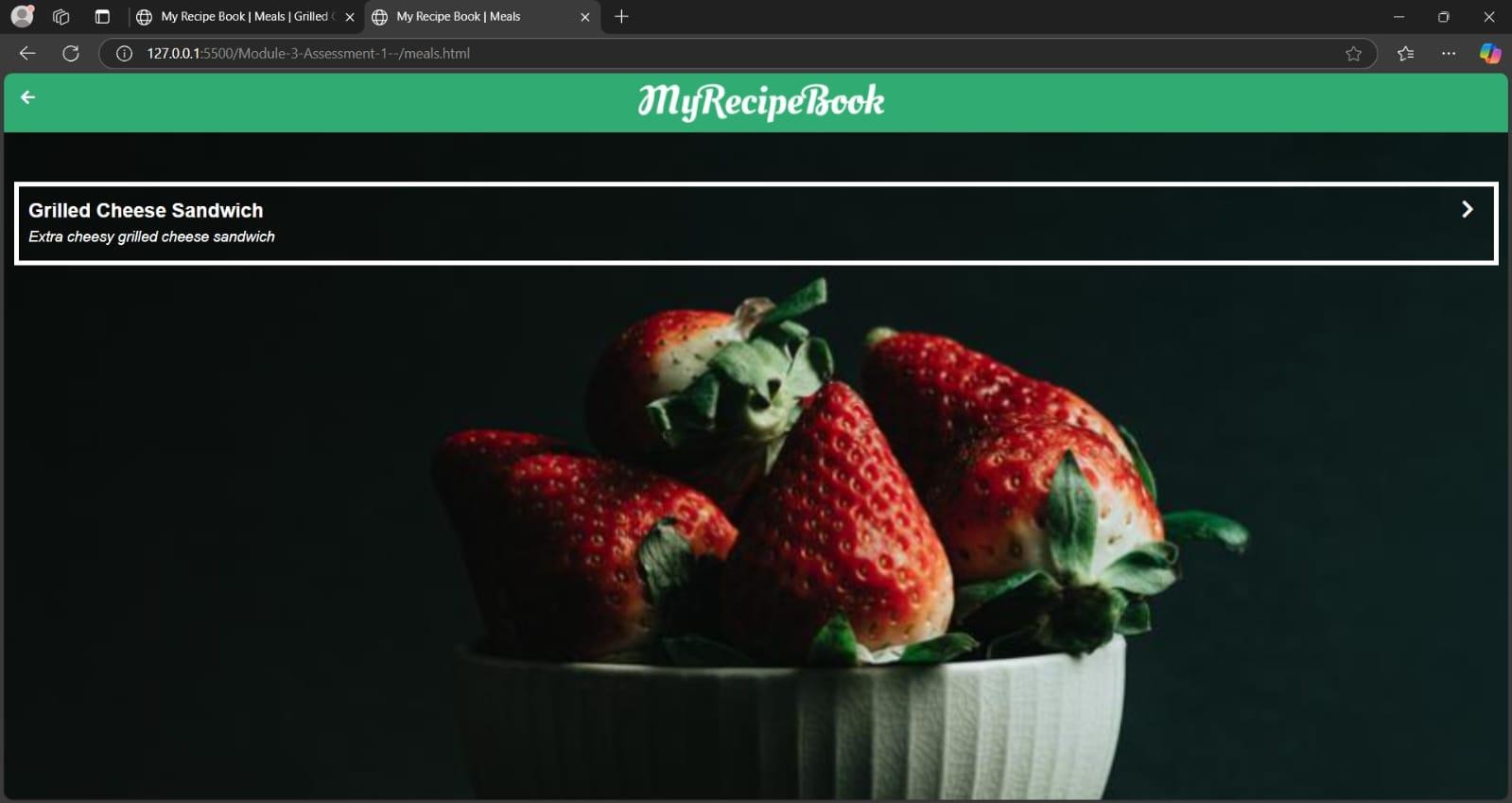
# **GIF for Interactions (Optional)**

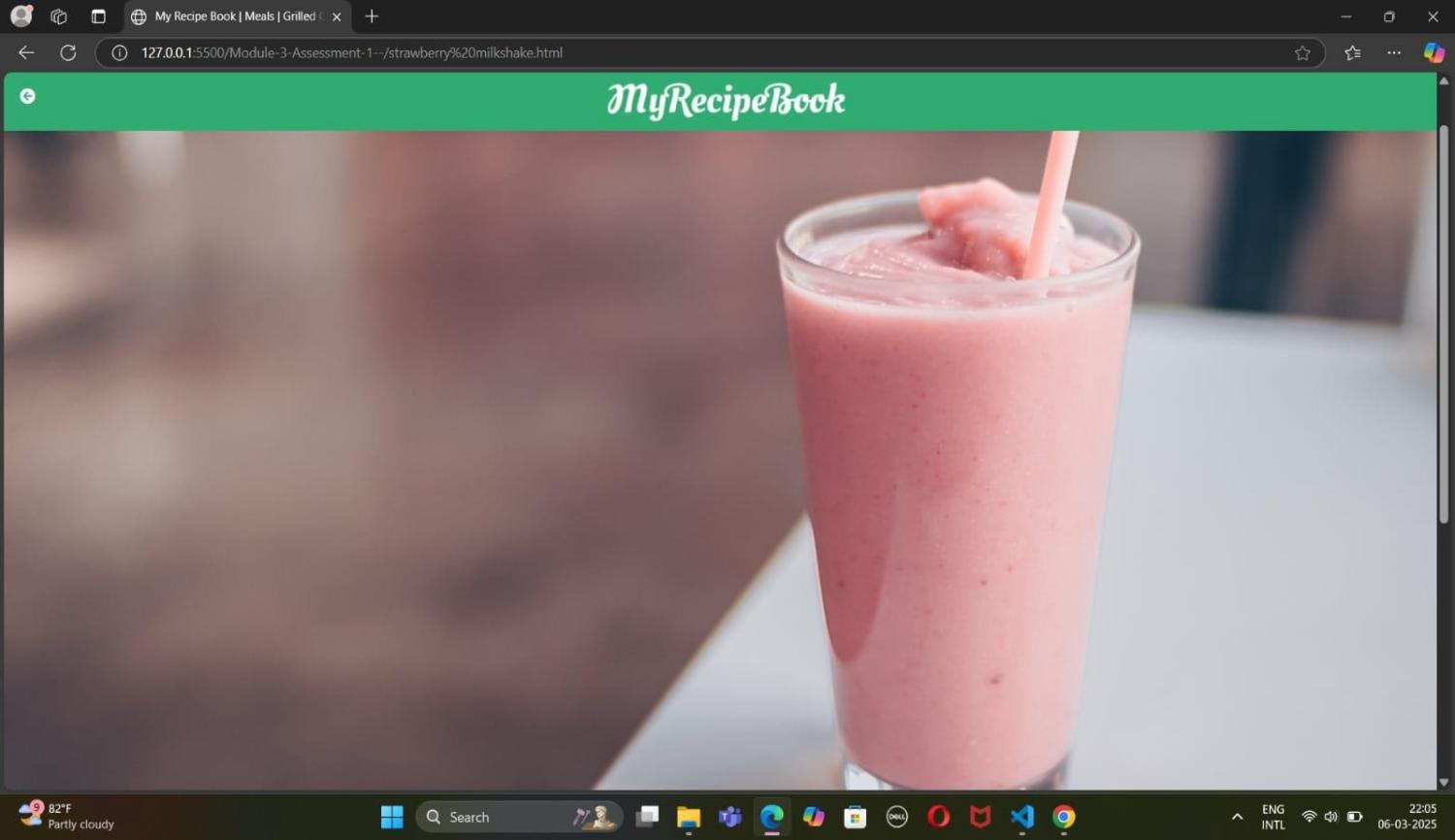
# Create a GIF showing a user selecting a recipe, clicking on it, and navigating to the recipe details page.

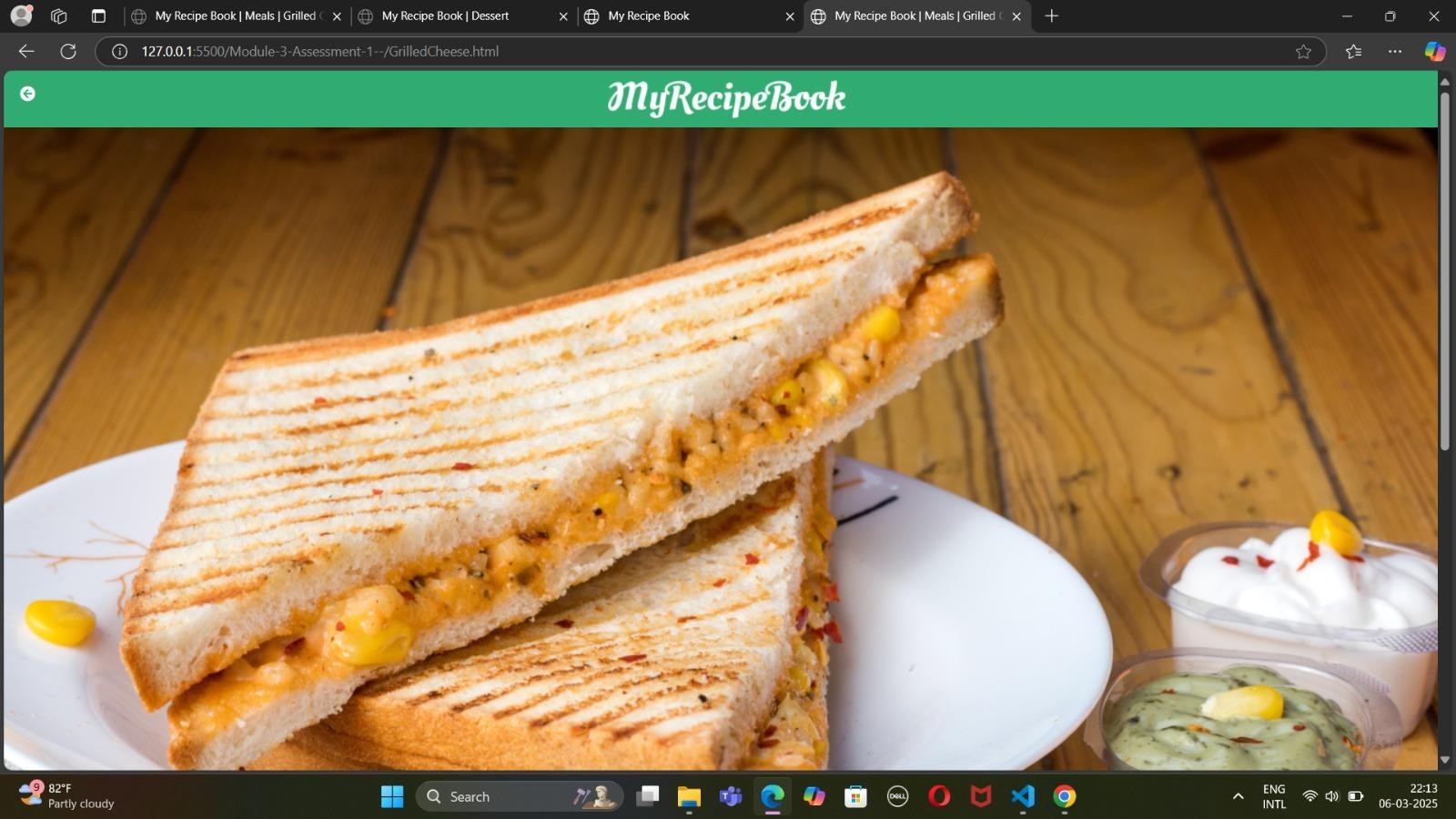
# Show a GIF of checking off ingredients while cooking.











1. **STYLING**

**1. CSS Frameworks/Libraries**

Your current project uses CSS for styling without a framework. However, if you plan to enhance it, you can use CSS frameworks like Tailwind CSS or Bootstrap for better styling and responsiveness.

✅ Current CSS Approach

Uses plain CSS (style.css) for layout, colors, fonts, and responsiveness.

Includes custom styles for homepage, category pages, and recipe details pages.

✅ Future Enhancements (Optional)

Tailwind CSS – A utility-first CSS framework for faster styling.

npm install -D tailwindcss

Bootstrap – Provides pre-designed UI components for buttons, cards, and grids.

npm install bootstrap

**2. Theming**

Currently, the project does not have a dark/light theme switch or a custom design system. However, you can implement:

✅ Custom Theming with CSS Variables (Example)

:root {

--primary-color: #30ac72;

--background-color: #f5f5f5;

--text-color: #333;

}

body {

background: var(--background-color);

color: var(--text-color);

}

This approach allows easy theme customization without modifying individual styles.

1. **TESTING**

**1. Testing Strategy**

For a React-based version of My Recipe Book, testing can be divided into three levels:

✅ Unit Testing (Jest + React Testing Library)

Focuses on individual components to ensure they render correctly.

Example: Testing if RecipeCard.js correctly displays a recipe title and image.

Example Test for RecipeCard.js

import { render, screen } from "@testing-library/react";

import RecipeCard from "../components/RecipeCard";

test("renders recipe title and image", () => {

render(<RecipeCard title="Grilled Cheese" image="/test-img.png" />);

expect(screen.getByText("Grilled Cheese")).toBeInTheDocument();

expect(screen.getByRole("img")).toHaveAttribute("src", "/test-img.png");

});

✅ Integration Testing (React Testing Library + Mock Data)

Ensures multiple components work together correctly.

Example: Testing if clicking a recipe card navigates to the RecipeDetails page.

Example Test for CategoryPage.js

import { render, screen } from "@testing-library/react";

import { MemoryRouter } from "react-router-dom";

import CategoryPage from "../pages/CategoryPage";

test("renders recipes from a category", () => {

render(

<MemoryRouter>

<CategoryPage categoryType="Meals" />

</MemoryRouter>

);

expect(screen.getByText("Meals")).toBeInTheDocument();

});

✅ End-to-End (E2E) Testing (Cypress or Playwright)

Tests user interactions like clicking buttons, navigating pages, and checking ingredients.

Example: Simulating a user searching for a recipe and marking ingredients as used.

Example Cypress Test

describe("Recipe App User Flow", () => {

it("should navigate to a recipe and check ingredients", () => {

cy.visit("/");

cy.get(".recipe-card").first().click();

cy.get("input[type='checkbox']").first().check();

});

});

**2. Code Coverage**

Ensuring that all components and logic are tested properly can be done using Jest’s built-in coverage tool.

✅ Run Code Coverage Report

npm test -- --coverage

This will generate a report showing which parts of the code are covered by tests.

1. **SCREENSHOT OR DEMO**

https://drive.google.com/file/d/1KJPoi8wAa7uF2T14UmrehtMOE4XOtUWL/view?usp=drivesdk

1. **KNOWN ISSUES**

Known Issues in My Recipe Book Web Application

1. Page refresh resets selections – Checked ingredients or selections are lost on reload.

Fix: Use localStorage to save selections.

1. No search or filtering – Users cannot easily find specific recipes.

Fix: Add JavaScript for dynamic search.

1. Recipes are hardcoded – Cannot add new recipes easily.

Fix: Use a database (Firebase/MySQL) for storage.

1. Not fully mobile-friendly – Some layouts may break on small screens.

Fix: Improve CSS for better responsiveness.

5.Future React version issues – State may reset, and routing may fail on refresh.

Fix: Use useContext for state management and handle API errors properly.

These are the main issues to be aware of. Let me know if you need help fixing them!

1. **FUTURE ENHANCEMENTS**
2. Search & Filter Feature – Allow users to search for recipes by name or filter by category.
3. User Login & Favorites – Let users create accounts and save their favorite recipes.

3. Dynamic Recipe Storage – Store and manage recipes in a database (Firebase/MySQL) instead of hardcoded HTML.

4. Interactive Animations – Add hover effects, transitions, and loading animations for a better user experience.

5.Dark Mode Support – Add a theme switcher for dark and light mode.

6. Mobile-Friendly Design – Improve CSS responsiveness for a smoother experience on small screens.

7.Recipe Submission Form – Allow users to add their own recipes to the website.

8. Voice-Assisted Cooking Mode – Guide users through recipes with voice instructions.

***THANK YOU !!***