MERN STACK PROJECT ON

"RECIPE HOUSE"

Submitted in partial fulfilment for the award of the degree in

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE AND ENGINEERING

Submitted By:

Deepak Paliwal |AP22110010591

Jyothi Sai Swaroop Mareedu|AP22110010602

Manoj Kumar Devarapu|AP22110010629

Sri Vyshnavi Gadamsetty|AP22110010632

Under the guidance of

"Yatharth Shahrawat"

Recipe House Application (React)

Introduction:

Step into the delightful world of culinary inspiration with Recipe House, a modern React-based web application designed for food enthusiasts and home chefs. This application revolutionizes how you discover, save, and share recipes, making it easier than ever to explore global cuisines, manage your favorite recipes, and craft your culinary journey.

Built with React.js and Tailwind CSS, the Recipe House application leverages real-time API integration and modern web technologies to provide a seamless and dynamic user experience. Whether you're looking for a quick meal idea or embarking on an elaborate cooking adventure, this app caters to all your needs.

Target Audience:

The Recipe Book application is thoughtfully designed to cater to a diverse group of individuals who share a common passion for cooking and exploring the culinary arts. Our primary target audience includes:

Home Chefs: Individuals who enjoy cooking regularly for their families or themselves. These users are often looking for ways to add variety to their daily meals and experiment with new dishes. Whether it's perfecting a classic recipe or introducing a fusion twist, the Recipe Book empowers home chefs to expand their culinary horizons effortlessly.

Food Enthusiasts: This group includes passionate individuals who view cooking as more than just a necessity—it's a creative and enjoyable activity. They enjoy exploring global cuisines, experimenting with unique flavor combinations, and crafting dishes that reflect their personality. With features like real-time recipe searches and detailed ingredient breakdowns, the Recipe Book supports their adventurous culinary endeavors.

Amateur Cooks: Beginners who are new to the kitchen and looking for simple, step-by-step guidance to help them learn the art of cooking. From easy-to-follow recipes to recommendations for popular dishes, the Recipe Book serves as an accessible and supportive platform for building their confidence and skills.

Health-Conscious Individuals: Users focused on maintaining a balanced and healthy lifestyle. With health labels and dietary filters, the application ensures they can easily find recipes tailored to their nutritional goals, such as low-carb, vegan, or gluten-free options.

Busy Professionals: Those with limited time but a love for homemade meals. The Recipe Book offers quick and easy recipes that fit into their busy schedules while maintaining quality and taste

Event Planners and Hosts: For individuals planning special gatherings or celebrations, the Recipe Book provides inspiration and practical options for preparing meals that impress their guests.

With its intuitive interface and wide range of recipes, the Recipe House is a versatile platform designed to meet the needs of all kinds of cooks, from novices to seasoned culinary experts. Whether you're looking to whip up a quick snack or prepare a feast, this application is your trusted companion in the kitchen.

Project Goals and Objectives:

The Recipe Book application aims to deliver an engaging and efficient platform for users to explore, manage, and enjoy recipes. Its goals and objectives are:

User-Friendly Interface: Design an intuitive and visually appealing interface that allows users to effortlessly search for, save, and view recipes.

Interactive Features: Integrate with the Edamam API to enable real-time recipe fetching, ensuring users receive dynamic and up-to-date content.

Customizable Experience: Provide functionality for users to create and manage a personalized collection of their favorite recipes, enhancing user engagement and satisfaction.

These objectives combine to create a seamless and enjoyable cooking companion for users of all experience levels.

Key Features:

The Recipe Book application offers a range of features designed to enhance the cooking experience:

Real-Time Recipe Search: Quickly search and discover recipes using the Edamam API, providing dynamic and accurate results.

Favorite Recipes: Save your preferred recipes to a dedicated favorites section, making it easy to revisit and organize them.

Recipe Details Page: Access comprehensive recipe details, including ingredients, nutritional information, and step-by-step preparation instructions.

Mobile-Friendly Design: Enjoy a seamless and responsive interface optimized for desktops, tablets, and smartphones.

Pre-Requisites:

To set up and run the Recipe Book application, ensure you have the following tools and libraries installed:

1. Node.js and npm:

Node.js is required to run JavaScript locally, and npm is essential for managing dependencies.

Install Node.js and npm:
Download and install Node.js from the official website: Node.js Download.

2. React.js Setup:

React is the core framework used in this application, and Vite is used for faster development.

Steps to Install React using Vite:

```
npm create vite@latest
cd project-name
npm install
npm run dev
```

Start the development server by running

```
npm run dev
```

3. API Access:

Obtain API keys from Edamam to fetch recipes dynamically. Add these keys to a `.env` file in your project:

```
REACT_APP_EDAMAM_APP_ID=your_app_id
REACT_APP_EDAMAM_APP_KEY=your_app_key
```

4. Tailwind CSS and DaisyUI:

For styling, Tailwind CSS and DaisyUI are used to create a modern and visually appealing user interface.

Install Tailwind CSS:
Follow the Tailwind CSS Installation Guide for your React project.

Add DaisyUI:
DaisyUI is a plugin for Tailwind CSS that simplifies the creation of UI components. Install it using npm:

```
npm install daisyui
```

Update Tailwind Configuration:
Add DaisyUI to the `plugins` array in your `tailwind.config.js` file:

```
module.exports = {
  content: ['./src/**/*.{js,jsx,ts,tsx}'],
  theme: {
    extend: {},
  },
  plugins: [require('daisyui')],
};
```

These pre-requisites ensure that your environment is set up for smooth development and optimal performance of the Recipe Book application.

Project Structure:

1. Components:

- **RecipeCard.jsx:** Displays individual recipe details.
- **Sidebar.jsx:** Handles navigation for desktop and mobile views.
- **Other Components:** Reusable elements across pages.

2. Pages:

- **HomePage.jsx:** Displays recipes and includes a search bar.
- **FavoritesPage.jsx:** Shows saved recipes from localStorage.
- **RecipePage.jsx:** Provides detailed information on a selected recipe.

3. Main Files:

- **App.jsx:** Manages routing and overall layout.
- **main.jsx:** Entry point for the React application.
- **index.css:** Defines global styles for the app.

Code Description:

RecipeCard Component:

The `RecipeCard` component is responsible for displaying individual recipe details in a card format. It includes:

- **Favorite Management:** Allows users to add or remove recipes from their favorites.
- **State Management:** Uses useState to track the favorite status of a recipe.
- **Persistence:** Leverages localStorage to save and persist the list of favorite recipes across sessions.
- **User Interaction:** Displays recipe title, image, and serving size with an option to navigate to the detailed recipe page.

CODE:-

```
import { Heart, HeartPulse, Soup } from "lucide-react";

import { useState } from "react";

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



const RecipeCard = ({ recipe, bg, badge }) => {

const healthLabels = [recipe.healthLabels[0],recipe.healthLabels[0]];

let favorites = JSON.parse(localStorage.getItem("favorites")) || [];

let isFav=false;

favorites.forEach((fav) => {

if (fav.label === recipe.label) {

isFav = true;

}
```

```
});

const [isFavorite, setIsFavorite] = useState(isFav);

const addRecipeToFavorites = () => {

let favorites = JSON.parse(localStorage.getItem("favorites")) || [];

let isAlreadyFav=false;

favorites.forEach((fav) => {

if (fav.label === recipe.label) {

isAlreadyFav = true;

}

});

if (isAlreadyFav) {

favorites = favorites.filter((fav) => fav.label !== recipe.label);

setIsFavorite(false);

} else {

favorites.push(recipe);

setIsFavorite(true);

}

localStorage.setItem("favorites", JSON.stringify(favorites));

};
```

```
return (
<div className={`flex flex-col rounded-md ${bg} overflow-hidden p-3 relative`}>
<Link to={"/recipePage"} state={{ recipe, bg }} className='relative h-32'>
<div className="skeleton absolute inset-0" />
<img
src={recipe.image}
alt="recipe img"
className="rounded-md w-full h-full object-cover cursor-pointer opacity-0 transition-opacity duration-500"
onLoad={(e) => {
e.currentTarget.style.opacity = 1;
e.currentTarget.previousElementSibling.style.display = "none";
}}
/>
<div
className="absolute bottom-2 left-2 bg-white rounded-full p-1 cursor-pointer flex items-center gap-1 text-sm"
>
<Soup size={16} /> {recipe.yield} Servings
</div>
```

```
<div
className="absolute top-1 right-2 bg-white rounded-full p-1 cursor-pointer"
onClick={(e) => {
e.preventDefault();
addRecipeToFavorites();
}}
>
{!isFavorite && <Heart size={20} className="hover:fill-red-500 hover:text-red-500" />}
{isFavorite && <Heart size={20} className="fill-red-500 text-red-500" />}
</div>
</Link>
<div className="flex mt-1">
<p className="font-bold tracking-wide">{recipe.label}</p>
</div>
<p className="my-2">
{recipe.cuisineType[0].charAt(0).toUpperCase() + recipe.cuisineType[0].slice(1)}
Kitchen
</p><div className="flex gap-2 mt-auto">
{healthLabels.map((label, idx) => (
<div key={idx} className={`flex gap-1 ${badge} items-center p-1 rounded-md`}>
```

```
<HeartPulse size={16} />

<span className="text-sm tracking-tighter font-semibold">{label}</span>

</div>

))}

</div>

</div>

);

};

export default RecipeCard;
```

Sidebar Component:

The `Sidebar` component provides navigation functionality for the application, optimizing the layout for both desktop and mobile devices:

Desktop View:

-Displays a vertical navigation menu with labeled icons for major sections like Home and Favorites.

-Ensures a user-friendly experience on larger screens.

Mobile View:

-Features a compact, horizontal navigation bar at the bottom of the screen.

-Provides easy access to essential links without cluttering the screen on smaller devices.

```
import { Heart, Home } from "lucide-react";

import { Link } from "react-router-dom";
```

```
const Sidebar = () => {
return (
<>
<DesktopSidebar />
<MobileSidebar />
</>
);
};
export default Sidebar;
```

```
const DesktopSidebar = () => {
return (
<div className='p-3 md:p-10 border-r min-h-screen w-24 md:w-64 hidden sm:block'>
<div className='flex flex-col gap-20 sticky top-10 left-0'>
<div className='w-full'>
<img src='/j.png' alt='logo' className='hidden md:block' />
<img src='/mobile-logo.svg' alt='logo' className='block md:hidden' />
</div>
<ul className='flex flex-col items-center md:items-start gap-8'>
```

```
<Link to={"/"} className='flex gap-1'>

<Home size={"24"} />

<span className='font-bold hidden md:block'>Home</span>

</Link>

<Link to={"/favorites"} className='flex gap-1'>

<Heart size={"24"} />

<span className='font-bold hidden md:block'>Favorites</span>

</Link>

</ul>

</div>

</div>

);

};
```

```
const MobileSidebar = () => {

return (

<div

className='flex justify-center gap-10 border-t fixed w-full

bottom-0 left-0 bg-white z-10 p-2 sm:hidden

'
```

```
>

<Link to={"/"}>

<Home size={"24"} className='cursor-pointer' />

</Link>

<Link to={"/favorites"}>

<Heart size={"24"} className='cursor-pointer' />

</Link>

</div>

);

};
```

HomePage Component:

The HomePage component is the landing page where users can search for and view recipes:

Dynamic Recipe Search:

Uses the Edamam API to fetch recipes based on user input, allowing real-time updates as search terms are entered.

Display of Recipes:

Recipes are displayed in a grid layout, making it easy for users to browse through options.

Default Recipe Fetching:

Uses useEffect to load a default set of recipes, such as "pizza," when the page first loads, giving users immediate access to content.

```
import { Search } from "lucide-react";
```

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

import { useEffect, useState } from "react";

import { getRandomColor } from "../lib/utils";

const APP_ID = "2f26d578";

const APP_KEY = "5d5f2265ddc5695214f78d350a248e67";

const HomePage = () => {

const [recipes, setRecipes] = useState([]);

const fetchRecipes = async (searchQuery) => {

setRecipes([]);

try {

const res = await fetch(

`https://api.edamam.com/api/recipes/v2/?app_id=${APP_ID}&app_key=${APP_KEY}&q=${se
archQuery}&type=public`

);

const data = await res.json();

setRecipes(data.hits);

console.log(data.hits);

}

catch (error) {

console.log(`Error: ${searchQuery} recipe not found`);
```

```
}};
```

```
useEffect(() => {

fetchRecipes("pizza");

}, []);

function handleSearch(){

const val = document.getElementById("find").value;

if (val === "") {

fetchRecipes("ice cream");

} else {

fetchRecipes(val);

}

}

return (

<div className='bg-[#faf9fb] p-10 flex-1'>

<div className='max-w-screen-lg mx-auto'>

<div>

<label className='input shadow-md flex items-center gap-2'>

<Search size={"24"} />

<input
```

```
id="find"

type='text'

className='text-sm md:text-md grow'

placeholder='What do you want to cook today?'

/>

<button type="button" class="px-3 py-1 font-medium text-white bg-yellow-400 hover:bg-yellow-
500 focus:outline-none focus:ring-4 focus:ring-yellow-300 font-medium rounded-full text-sm px-
5 py-2.5 text-center me-2 mb-2 dark:focus:ring-yellow-900"

onClick={handleSearch}>

Search</button></label></div>

<h1 className='font-bold text-3xl md:text-5xl mt-4'>Recommended Recipes</h1>

<p className='text-slate-500 font-semibold ml-1 my-2 text-sm tracking-tight'>Popular
choices</p>

<div className='grid gap-3 grid-cols-1 md:grid-cols-2 lg:grid-cols-3'>

{

recipes.map(({ recipe }, index) => (

<RecipeCard key={index} recipe={recipe} {...getRandomColor()} />

))

}

</div>
```

```
</div>

</div>

);

};

export default HomePage;
```

Favorites Page:

The FavoritesPage component shows a grid of recipes that the user has saved as favorites:

Favorite Storage:

Retrieves the list of saved recipes from localStorage, allowing users to see their personalized recipe collection.

Recipe Grid Layout:

Displays the saved recipes in a clean, organized grid for easy access and management.

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

import { getRandomColor } from "../lib/utils";

const FavoritesPage = () => {

const favorites = JSON.parse(localStorage.getItem("favorites")) || [];

return (

<div className='bg-[#faf9fb] flex-1 p-10 min-h-screen'>

<div className='max-w-screen-lg mx-auto'>

<p className='font-bold text-3xl md:text-5xl my-4'>My Favorites</p>
```

```
{favorites.length === 0 && (
<div className='h-[80vh] flex flex-col items-center gap-4'>
<img src='/404.svg' className='h-3/4' alt='404 svg' />
</div>
)}
<div className='grid grid-cols-1 md:grid-cols-2 lg:grid-cols-3 gap-4'>
{favorites.map((recipe) => (
<RecipeCard key={recipe.label} recipe={recipe} {...getRandomColor()} />
))}
</div>
</div>
</div>
);
};
export default FavoritesPage;
```

Recipe Page :

The RecipePage component provides a detailed view of a selected recipe, offering users in-depth information:

Ingredient List:

Displays all the ingredients required to make the recipe, with additional details such as measurements and quantities.

Shows step-by-step preparation instructions to help users follow along easily.

Dynamic Data Loading:

Uses useLocation from react-router-dom to dynamically load the recipe data passed from the previous page (such as from the recipe card or favorites page).

This structure and approach ensure a smooth user experience and easy access to essential features like searching, saving favorites, and viewing detailed recipes.

```
import React from "react";

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

const RecipePage = () => {

const location = useLocation();

const { recipe, bg } = location.state;

const ingredients = recipe.ingredients;

console.log(bg);

return (

<div className="ml-5">

<div className="p-6 max-w-screen-lg mx-auto space-y-10 ml-10 mt-10">

<section className="text-center">
```

```
<div className="bg-gray-200 w-full h-60 rounded-lg shadow-md flex items-center
justify-center">

<img

src={recipe.image}

alt="Recipe Image"

className="w-full h-full object-cover rounded-lg"

/>

</div>

<h1 className="text-2xl font-extrabold mt-4">{recipe.label}</h1>

<a

href={`https://www.youtube.com/results?search_query=${recipe.label} recipe`}

target="_blank"

rel="noopener noreferrer"

className="relative h-32"

>

<p className="text-gray-600 mt-2 hover:text-blue-500">

Discover content, explore features, and dive into details with ease.

</p>

</a>

</section>
```

```
{/* Grid container */}

<div className="grid grid-cols-3 gap-6">

{ingredients.map((ingredient, index) => (

<div

key={index}

// style={{ backgroundColor: '#F9EFE1' }}

className={`${bg} shadow-lg rounded-lg p-4 flex flex-col items-start space-y-4`}

>

<div className="bg-gray-200 w-16 h-16 rounded-md flex items-center justify-center">

<img

src={ingredient.image}

alt={`${ingredient.food} image`}

className="w-full h-full object-cover rounded-md"

/>

</div>

<div>

<h3 className="text-lg font-bold">{ingredient.food}</h3>

<ul className="text-gray-600 mt-2 space-y-1">

<li>{ingredient.text}</li>
```

```
<li>Measure: {ingredient.measure}</li>

<li>Quantity: {ingredient.quantity}</li>

<li>Weight: {ingredient.weight} gram</li>

</ul>

</div>

</div>

))}

</div>

</div>

</div>

);

};

export default RecipePage;
```

Main Files:

App.jsx:
Handles the overall application layout and routing.

Uses react-router-dom to define routes for pages like Home, Favorites, and Recipe Details.

Includes the Sidebar for consistent navigation.

```
import { Route, Routes } from "react-router-dom";

import Sidebar from "./components/Sidebar";

import HomePage from "./pages/HomePage";

import FavoritesPage from "./pages/FavoritesPage";

import RecipePage from "./pages/RecipePage";

function App() {

return (

<div className='flex'>

<Sidebar />

<Routes>

<Route path='/' element={<HomePage />} />

<Route path='/favorites' element={<FavoritesPage />} />

<Route path='/recipePage' element={<RecipePage />} />

</Routes>

</div>

);

}

export default App;
```

main.jsx:
The entry point of the application.

- Configures the React application with the BrowserRouter for routing.
- Mounts the App component to the root DOM node.

```
import React from "react";

import ReactDOM from "react-dom/client";

import App from "./App.jsx";

import "./index.css";

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

ReactDOM.createRoot(document.getElementById("root")).render(

<React.StrictMode>

<BrowserRouter>

<App />

</BrowserRouter>

</React.StrictMode>

);
```

Setup Instructions:

1) Clone the repository:

git clone <repository_url>

cd project-directory

npm install

2)Add API credentials in .env:

REACT_APP_EDAMAM_APP_ID=your_app_id

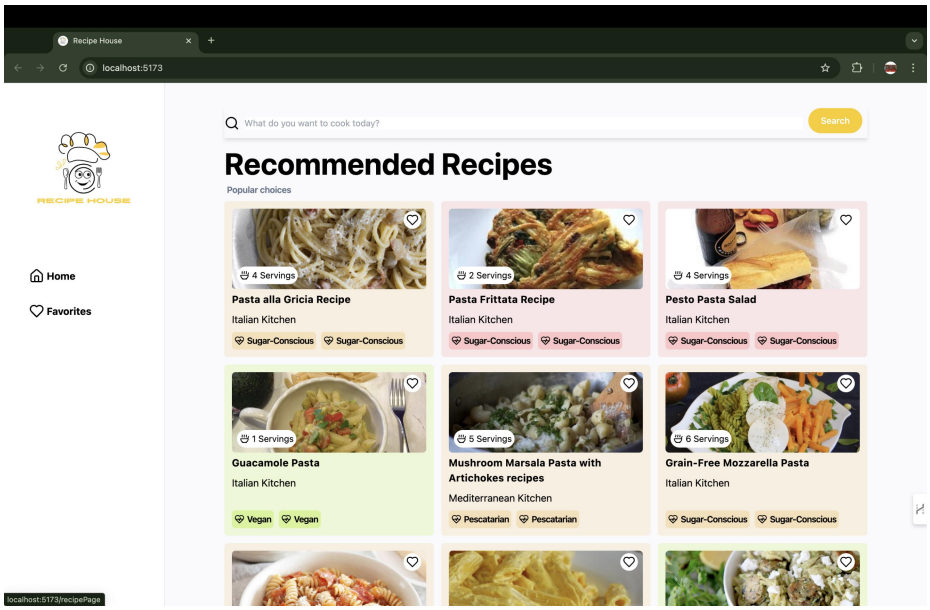
REACT_APP_EDAMAM_APP_KEY=your_app_key

3)Start the development server:

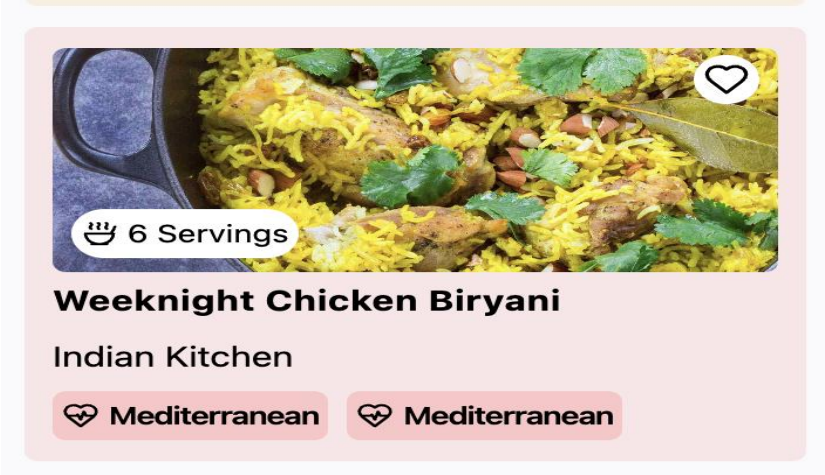
npm run dev

Screenshots:

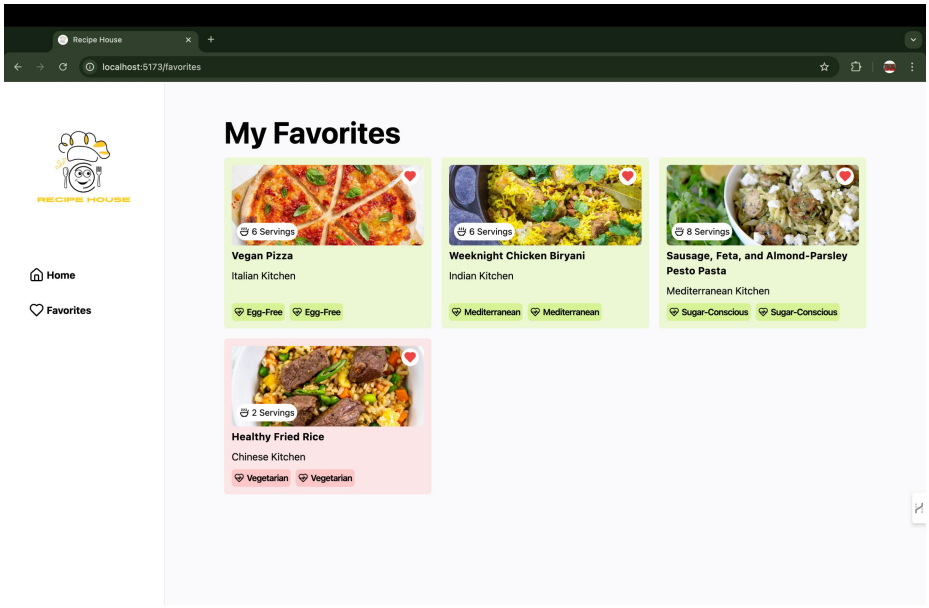
Home Page:*Search bar with popular recipe suggestions.*



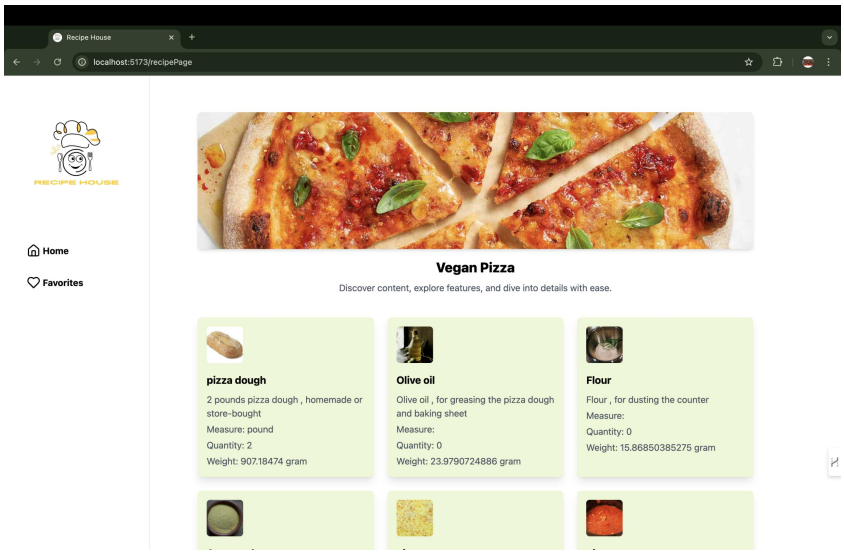
Recipe Card:*Displays image, title, and cuisine type.*



Favorites Page:*Grid of saved recipes.*



Recipe Details:*List of ingredients with nutritional details.*



DEMO LINK:-

https://drive.google.com/file/d/1h8HR5QixhD-6MD5cTsV8FTx3Q8BSxBzq/view?usp=drive_link

THANKYOU