# Cook Book: Your Virtual Kitchen Assistant

**A**
**Project Report**

**Full stack developer**

**Bachelor of Computer Science and Engineering**

**Submitted by**

**Sk.Sana-AP23110010393**
**Ch.Rahitya-AP23110010407**
**B.lakshmi poojitha-AP23110011560**
**M.Sravani-AP23110011656**

**Submitted to**

**Mr.Arshad syed**

**SRM UNIVERSITY-AP**
**MANGALAGIRI(M)**
**KURAGALLU VILLAGE**
**ANDHRA PRADESH - 522540**

## Introduction

In a people there was busy schedules yet still crave delicious homemade meals, finding the right recipe at the right time can be overwhelming. *Cook Book: Your Virtual Kitchen Assistant* aims to simplify this daily challenge by offering a smart, interactive, and easy-to-use platform that brings cooking inspiration right to the user's fingertips. This application transforms the traditional cookbook into a modern digital experience, helping users explore new dishes, organize recipes, and plan meals with ease.

The project's frontend, built using **HTML, CSS, JavaScript, and React.js**, focuses on delivering a clean and responsive interface that adapts smoothly to desktops, tablets, and smartphones. With features such as recipe search, ingredient-based filtering, recipe categories, and detailed step-by-step instructions, the platform ensures users can navigate and cook without confusion. Visual elements like recipe cards, pop-up modals, and interactive ingredient checklists enhance the overall experience, making cooking more enjoyable and less time-consuming.

*Cook Book: Your Virtual Kitchen Assistant* is designed for users of all cooking levels—from beginners learning their first dishes to home chefs trying innovative meals. By combining modern design with practical functionality, the project aims to make everyday cooking smarter, faster, and more organized.

## Scenario-Based Intro

Imagine it's a busy evening and you're standing in your kitchen, unsure of what to cook with the few ingredients you have. Instead of wasting time searching online or flipping through old recipe books, you open **Cook Book: Your Virtual Kitchen Assistant**. With just a few clicks, the app shows you recipes based on the ingredients already in your kitchen.

The clean and simple interface, built using **HTML, CSS, JavaScript, and React.js**, helps you quickly explore different dishes. You can filter recipes by ingredients, cuisine, or cooking time and instantly get step-by-step instructions. As

you cook, you can check off ingredients one by one, making the process easier and more organized.

Whether you're trying a new recipe for fun, planning meals for the week, or just want something quick, the responsive design works smoothly on your phone, tablet, or computer. Interactive recipe cards, pop-ups, and helpful suggestions make cooking feel more enjoyable and less stressful.

Instead of guessing what to cook, Cook Book becomes your smart kitchen partner—helping you turn simple ingredients into delicious meals with confidence.

## Target Audience

*Cook Book: Your Virtual Kitchen Assistant* is designed for anyone who wants to make cooking easier, more enjoyable, and interactive. Its intuitive frontend appeals to a wide range of users:

1. **Busy Professionals & Students** – People with limited time who need quick, easy-to-follow recipes and meal planning tools at their fingertips.

2. **Beginner Cooks** – Individuals learning to cook who benefit from step-by-step instructions, interactive ingredient checklists, and visual guidance.

3. **Home Chefs & Food Enthusiasts** – Users looking to explore new cuisines, try creative recipes, and organize their favorite dishes in a visually appealing, interactive interface.

4. **Health & Diet-Conscious Users** – Those who want to filter recipes based on dietary preferences, ingredients, or nutrition goals.

5. **Tech-Savvy Food Lovers** – Users who enjoy engaging with digital tools and responsive web applications that enhance their cooking experience across devices.

By focusing on a clean, dynamic frontend, the application ensures that all users, regardless of skill level or lifestyle, can navigate seamlessly, find inspiration, and enjoy a personalized culinary journey.

## Project Goals and Objectives

The primary goal of *Cook Book: Your Virtual Kitchen Assistant* is to create an engaging and interactive platform that makes cooking easier, more organized, and enjoyable for everyone. Our objectives include:

**Intuitive User Interface:** Build a clean and responsive frontend that allows users to explore recipes, follow step-by-step instructions, and navigate the platform effortlessly, enhancing the overall cooking experience.

**Efficient Recipe Management:** Offer tools to save favorite recipes, plan meals, and filter dishes by ingredients, cuisine, or dietary needs, helping users find what they want quickly and efficiently.

**Interactive and Responsive Design:** Utilize modern web technologies like HTML,CSS,JavaScript and React to ensure the platform is visually appealing, interactive, and fully functional across desktops, tablets, and mobile devices.

**Enhanced Cooking Experience:** Enable users to track ingredients, get recommendations, and engage with recipes in a way that makes cooking fun, creative, and personalized.

## Key Features

1. **Smart Recipe Search** – Quickly find recipes by ingredients, cuisine, or meal type with an intuitive search bar.

2. **Interactive Recipe Cards** – Visual cards displaying images, cooking time, difficulty level, and brief descriptions.

3. **Step-by-Step Cooking Instructions** – Clear, interactive guides with checklists to make following recipes easy and enjoyable.

4. **Favorites & Meal Planning** – Save favorite recipes and plan weekly meals for personalized culinary management.

5. **Responsive Design** – Optimized for desktops, tablets, and smartphones to ensure a seamless experience on any device.

6. **Ingredient-Based Recommendations** – Suggest recipes based on available ingredients, reducing food waste and enhancing convenience.

7. **Dynamic Modals & Pop-Ups** – Access full recipe details without leaving the main interface for smooth navigation.

8. **User-Friendly Navigation** – Organized layout for quick access to searches, categories, saved recipes, and personal plans.

## PRE-REQUISITES

The development of *Cook Book: Your Virtual Kitchen Assistant*, ensure you have the following tools and knowledge in place:

**1.Node.js and npm:**

Node.js is a powerful JavaScript runtime environment that allows you to run JavaScript code on the local environment. It provides a scalable and efficient platform for building network applications.

Install Node.js and npm on your development machine, as they are required to run JavaScript on the server-side.

● Download: https://nodejs.org/en/download/

● Installation instructions: https://nodejs.org/en/download/package-manager/

**2.React.js:**

React.js is a popular JavaScript library for building user interfaces. It enables developers to create interactive and reusable UI components, making it easier to build dynamic and responsive web applications.

Install React.js, a JavaScript library for building user interfaces.

● **Create a new React app:**

```
npm create vite@latest
```

Enter and then type project-name and select preferred frameworks and then enter

● **Navigate to the project directory:**

```
cd project-name

npm install
```

● **Running the React App:** With the React app created, you can now start the development server and see your React application in action.

● **Start the development server:**

```
Npm run serve:api

npm run dev
```

This command launches the development server, and you can access your React app at http://localhost:5173 in your web browser.

**3.Basic Web Development Knowledge:**

- Familiarity with **HTML** for creating page structure.
- Understanding of **CSS** for styling and layout.
- Basic **JavaScript** knowledge for interactivity and DOM manipulation.

**4.Code Editor or IDE:**

- Recommended: **Visual Studio Code**
- These provide syntax highlighting, extensions, and a smooth development workflow.

**Visual Studio Code:** Download from https://code.visualstudio.com/download

**5.Web Browser:**

- A modern browser such as **Google Chrome**, **Firefox**, or **Edge** for testing and debugging your application

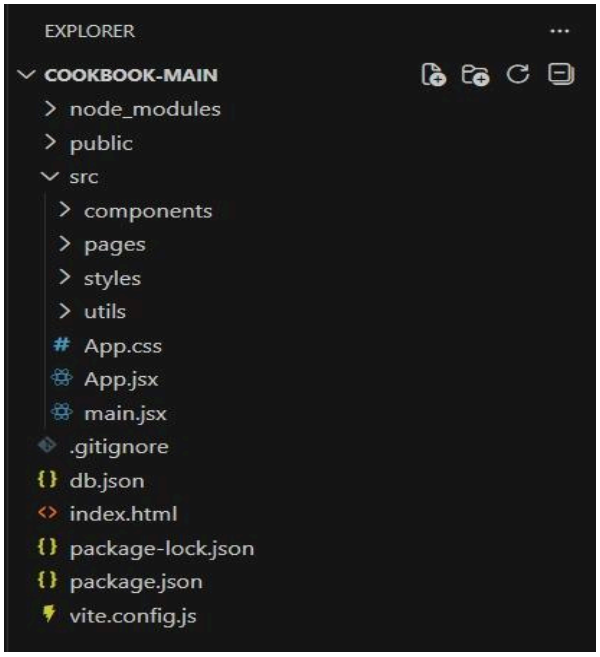
**6.Version Control:**

- **Git** for tracking changes and managing code versions.
- Hosting platforms like **GitHub** for collaboration.

**7.Basic Knowledge of Responsive Design:**

- Understanding how to use **media queries** and flexible layouts to make the application work on **desktop, tablet, and mobile devices**.

## Project structure




**public/** → Contains the main HTML file and public assets like images.

**src/** → Main folder where all React code lives.

**components/** → Small reusable frontend pieces (navbar, cards, etc.).

**pages/** → Full screens such as Home, Favorites, Recipe Details.

**styles/** → CSS files for consistent styling.

**utils/** → Helper functions for filtering, fetching, or processing recipes.

**db.json** → Sample recipe data for testing.

**App.jsx** → Manages routes and overall app structure.

**main.jsx** → Starting point that loads the React app.

**Project flow**

**Demolink:**[https://drive.google.com/drive/folders/1MKXKPvecdlBXfDs9M9X-qwlIvwBENUns?usp=sharing](https://drive.google.com/drive/folders/1MKXKPvecdlBXfDs9M9X-qwlIvwBENUns?usp=sharing)

**Code:**[https://drive.google.com/file/d/1l1Ps4OKU9v2uE86_OHn648107TeDvli0/view?usp=sharing](https://drive.google.com/file/d/1l1Ps4OKU9v2uE86_OHn648107TeDvli0/view?usp=sharing)

## Milestone 1: Project Setup and Configuration:

### 1. Install required tools and software:

1.Open the project folder to install necessary tools In this project, we use:

- npm install
- npm run dev
- npm install framer-motion
- npm install react-router-dom
- npm install @fortawesome/fontawesome-free

## Milestone 2: Web Development:

### 1. Setup React Application:

- Create React application.
- Configure Routing.
- Install required libraries.

### App.jsx client:-

```
import React from 'react';
import ReactDOM from 'react-dom/client';
import App from './App';
import './styles/global.css';
import './styles/theme.css';

ReactDOM.createRoot(document.getElementById('root')).render(
  <React.StrictMode>
    <App />
  </React.StrictMode>
);
```

**Code Description:-**

● import React from 'react'; Loads the React library, which is required to create React components.

● import ReactDOM from 'react-dom/client'; Imports ReactDOM's modern rendering API (createRoot) used in React 18+ to render components in the browser.

● import App from './App'; Imports the main App component, which contains the structure and logic of your application.

● import './styles/global.css'; Loads global CSS styles that apply across the entire application.

● import './styles/theme.css'; Loads theme-related styling (such as colors, fonts, spacing, etc.).

● ReactDOM.createRoot(document.getElementById('root')).render(...) Creates a root React node and tells React to render the application inside the HTML element with the id root.

● <React.StrictMode> A React wrapper that helps highlight potential problems in your application during development. It does not affect production.

● <App /> The main component of the application that contains all other components. Everything on the screen is rendered from the App component.

**2.App.jsx**

```
import React, { useEffect, useState } from 'react';
import { BrowserRouter as Router, Routes, Route, useNavigate } from 'react-router-dom';
import { initTheme } from './utils/theme';
import Navbar from './components/Navbar';
import Footer from './components/Footer';
import Home from './pages/Home';
import MyCookbook from './pages/MyCookbook';
import UploadRecipe from './pages/UploadRecipe';
import Community from './pages/Community';
import EditRecipe from './pages/EditRecipe';
import About from './pages/About';
import SignIn from './pages/SignIn';
import SignUp from './pages/SignUp';
import RecipeDetails from './components/RecipeDetails';
import './App.css';

const AppContent = () => {
  const navigate = useNavigate();
  const [searchQuery, setSearchQuery] = useState('');

  useEffect(() => {
    initTheme();
  }, []);

  const handleVoiceSearch = (query) => {
    setSearchQuery(query);
    navigate('/');
    // Trigger search after navigation
    setTimeout(() => {
      const event = new CustomEvent('voiceSearch', { detail: query });
      window.dispatchEvent(event);
    }, 100);
  };

  return (
    <div className="app">
```

```
  return (
    <div className="app">
      <Navbar onVoiceSearch={handleVoiceSearch} />
      <main className="main-content">
        <Routes>
          <Route path="/" element={<Home />} />
          <Route path="/my-cookbook" element={<MyCookbook />} />
          <Route path="/upload" element={<UploadRecipe />} />
          <Route path="/community" element={<Community />} />
          <Route path="/community/edit/:id" element={<EditRecipe />} />
          <Route path="/about" element={<About />} />
          <Route path="/signin" element={<SignIn />} />
          <Route path="/signup" element={<SignUp />} />
          <Route path="/recipe/:id" element={<RecipeDetails />} />
        </Routes>
      </main>
      <Footer />
    </div>
  );
};

const App = () => {
  return (
    <Router>
      <AppContent />
    </Router>
  );
};

export default App;
```

**Code Description**

- React, useState, and useEffect are imported to manage component behavior and application state.
- BrowserRouter (Router), Routes, Route, and useNavigate from react-router-dom are imported to enable client-side routing and page navigation.
- initTheme is imported to initialize the website's theme (e.g., dark/light mode).

- Multiple components and pages are imported, including Navbar, Footer, Home, MyCookbook, UploadRecipe, Community, EditRecipe, About, SignIn, SignUp, and RecipeDetails.
- App.css is imported for global styling.
- Wraps the main routing and layout logic of the application.
- useNavigate() is used to programmatically navigate between pages.
- searchQuery state stores search input received via voice search.
- useEffect runs once to initialize the theme using initTheme().
- handleVoiceSearch(query):
- Updates the searchQuery state.
- Navigates back to the home page ('/').
- Dispatches a custom voiceSearch event with the search query, allowing the Home page or other components to listen for voice search triggers.
- The component returns the main structure of the app:
- Navbar (supports voice search input).
- A <main> section holding all defined routes.
- Footer at the bottom of the page.
- Routes define the different pages of the application.
- / → Home
- /my-cookbook → User's saved recipes
- /upload → UploadRecipe page
- /community → Community recipes
- /community/edit/:id → EditRecipe page
- /about → About page
- /signin and /signup → Authentication pages
- /recipe/:id → RecipeDetails page
- Wraps the entire application inside a Router, enabling navigation between pages.
- Renders the AppContent component within the router context.
- The App component is exported as the default export for use in main.js or other entry files.

**3. Design UI components:**

- Create Components.
- Implement layout and styling.
- Add navigation.

**4.Code HeroSection.jsx**

```
import React from 'react';
import { motion } from 'framer-motion';
import SearchBar from './SearchBar';
import './HeroSection.css';

const HeroSection = ({ onSearch, onCuisineSelect }) =>
  return (
    <section className="hero-section">
      {/* 🖼️ Background Image */}
      <div className="hero-background">
        <img
          src="/assets/hero-bg.jpeg"
          alt="Delicious food background"
          className="background-image"
        />
        <div className="image-overlay"></div>
      </div>

      {/* 🌟 Hero Content */}
      <div className="hero-content container">
        {/* 🍴 Floating Icon */}
        <motion.div
          className="hero-icon"
          initial={{ opacity: 0, scale: 0.8 }}
          animate={{ opacity: 1, scale: 1 }}
          transition={{ duration: 0.8 }}
        >
          <i class="fas fa-utensils"></i>
        </motion.div>

        {/* ✨ Title and Subtitle */}
        <motion.div
          className="hero-text"
          initial={{ opacity: 0, y: 30 }}
          animate={{ opacity: 1, y: 0 }}
          transition={{ duration: 0.8, delay: 0.2 }}
```

```
      <div style={{ position: 'relative', height: '220px', width: '100%' }}>
        <h1 className="hero-title gradient-title">COOKBOOK!</h1>
      </div>
      {/* 🌈 Highlighted Subtitle */}
      <p className="hero-subtitle">Discover the Art of Flavors.</p>
    </motion.div>

    {/* 🔍 Search Bar */}
    <motion.div
      className="hero-search"
      initial={{ opacity: 0, y: 30 }}
      animate={{ opacity: 1, y: 0 }}
      transition={{ duration: 0.8, delay: 0.4 }}
    >
      <SearchBar onSearch={onSearch} placeholder="Search recipes..." />
    </motion.div>

    {/* 🍛 Cuisine Tags */}
    <motion.div
      className="hero-cuisine-tags"
      initial={{ opacity: 0, y: 30 }}
      animate={{ opacity: 1, y: 0 }}
      transition={{ duration: 0.8, delay: 0.6 }}
    >
      {['Thai', 'Chinese', 'Italian', 'Indian'].map((cuisine) => (
        <button
          key={cuisine}
          type="button"
          className="hero-cuisine-button"
          onClick={() =>
            onCuisineSelect ? onCuisineSelect(cuisine) : onSearch?.(cuisine)
          }
        >
          {cuisine}
```

```
                onClick={() =>
                  onCuisineSelect ? onCuisineSelect(cuisine) : onSearch?.(cuisine)
                }
              >
                {cuisine}
              </button>
            ))}
          </motion.div>
        </div>
      </section>
    );
};

export default HeroSection;
```

**Code Description**

- React is imported to create the component.
- Framer Motion (motion) is imported to add smooth animations to the hero section elements.
- SearchBar is imported to display a search input field for users to search recipes.
- HeroSection.css is imported to style the entire section.
- The HeroSection component receives two props:
- onSearch: Triggered when the search bar is used or when a cuisine tag is clicked.
- onCuisineSelect: Triggered specifically when a cuisine tag is selected (optional).
- The component returns a <section> containing:
- A background image with a dark overlay for improved visibility.
- An animated floating utensils icon using Framer Motion.
- A title ("COOKBOOK!") with gradient styling and an animated subtitle (“Discover the Art of Flavors”).
- An animated SearchBar component that allows users to search for recipes.
- A set of animated cuisine category buttons (Thai, Chinese, Italian, Indian).
- Clicking a button calls onCuisineSelect(cuisine) if provided, otherwise falls back to onSearch(cuisine).

**5. Code IngredientChecklist.js:-**

```
import React, { useState, useEffect } from '
import { motion } from 'framer-motion';
import './IngredientChecklist.css';

const IngredientChecklist = ({ ingredients }
  const [checkedItems, setCheckedItems] = us

  useEffect(() => {
    // Load saved checklist state from local
    const saved = localStorage.getItem('ingr
    if (saved) {
      try {
        setCheckedItems(JSON.parse(saved));
      } catch (e) {
        console.error('Error loading checkli
      }
    }
  }, []);

  useEffect(() => {
    // Save checklist state to localStorage
    localStorage.setItem('ingredient_checkli
  }, [checkedItems]);

  const toggleItem = (index) => {
    setCheckedItems(prev => ({
      ...prev,
      [index]: !prev[index]
    }));
  };

  const checkedCount = Object.values(checked
  const totalCount = ingredients.length;

  return (
    <div className="ingredient-checklist">
```

```
const IngredientChecklist = ({ ingredients }) => {
    <div className="checklist-header">
      <h3>
        <i className="fas fa-list-check"></i> Ingredients Checklist
      </h3>
      <span className="checklist-progress">
        {checkedCount} / {totalCount}
      </span>
    </div>
    <ul className="checklist-items">
      {ingredients.map((ingredient, index) => (
        <motion.li
          key={index}
          className={`checklist-item ${checkedItems[index] ? 'checked' : ''}`}
          initial={{ opacity: 0, x: -20 }}
          animate={{ opacity: 1, x: 0 }}
          transition={{ delay: index * 0.05 }}
        >
          <label>
            <input
              type="checkbox"
              checked={checkedItems[index] || false}
              onChange={() => toggleItem(index)}
            />
            <span className="checkmark"></span>
            <span className="ingredient-text">
              <strong>{ingredient.measure}</strong> {ingredient.name}
            </span>
          </label>
        </motion.li>
      ))}
    </ul>
  </div>
);
};
```

# Code Description

- React and its hooks (useState, useEffect) are imported to manage component state and lifecycle behavior.
- Framer Motion is imported to animate each checklist item when it appears.
- IngredientChecklist.css is imported for styling the checklist layout and appearance.
- The IngredientChecklist component receives an ingredients prop, which contains a list of ingredient objects (each with measure and name).
- A checkedItems state object keeps track of which ingredients are checked.
- Keys represent ingredient indices.
- Values are booleans (true for checked, false for unchecked).
- The first useEffect runs on mount and loads previously saved checklist data from localStorage (if available), restoring the user's saved progress
- The second useEffect saves the updated checklist state to localStorage whenever checkedItems changes.
- The toggleItem function flips the checked state for the ingredient at the given index.
- checkedCount calculates how many ingredients are checked, while totalCount shows the full list length.
- The component displays:A header with a title and a progress indicator (checkedCount / totalCount).
- A list of ingredients, where each item:

  Animates into view using Framer Motion.

  Contains a checkbox to mark the item as completed.

**6. Code Navbar.jsx**

```
import React, { useState, useEffect } from 'react';
import { Link, useLocation, useNavigate } from 'react-router-dom';
import { motion } from 'framer-motion';
import ThemeToggle from './ThemeToggle';
import './Navbar.css';
import { getCurrentUser, signOut } from '../utils/localStorage';

const Navbar = () => {
  const location = useLocation();
  const isActive = (path) => location.pathname === path;
  const [user, setUser] = useState(null);
  const [menuOpen, setMenuOpen] = useState(false);
  const navigate = useNavigate();

  useEffect(() => {
    setUser(getCurrentUser());
  }, [location]);

  // Close mobile menu on navigation
  useEffect(() => {
    setMenuOpen(false);
  }, [location.pathname]);

  const handleSignOut = () => {
    signOut();
    setUser(null);
    navigate('/');
  };

  return (
    <motion.nav
      className="navbar"
      initial={{ y: -100 }}
      animate={{ y: 0 }}
      transition={{ duration: 0.5 }}
    >
```

```
  return (
    <motion.nav
      className="navbar"
      initial={{ y: -100 }}
      animate={{ y: 0 }}
      transition={{ duration: 0.5 }}
    >
      <div className="navbar-container">
        <Link to="/" className="navbar-brand">
          <i className="fas fa-utensils"></i>
          <span>CookBook</span>
        </Link>

        <div className={`navbar-menu ${menuOpen ? 'open' : ''}`}>
          <Link
            to="/"
            className={`navbar-link ${isActive('/') ? 'active' : ''}`}
            onClick={() => setMenuOpen(false)}
          >
            Home
          </Link>
          <Link
            to="/my-cookbook"
            className={`navbar-link ${isActive('/my-cookbook') ? 'active' : ''}`}
            onClick={() => setMenuOpen(false)}
          >
            My Cookbook
          </Link>
          <Link
            to="/upload"
            className={`navbar-link ${isActive('/upload') ? 'active' : ''}`}
            onClick={() => setMenuOpen(false)}
          >
            Upload
```

```
          Upload
        </Link>
        <Link
          to="/community"
          className={`navbar-link ${isActive('/community') ? 'active' : ''}`}
          onClick={() => setMenuOpen(false)}
        >
          Community
        </Link>
        <Link
          to="/about"
          className={`navbar-link ${isActive('/about') ? 'active' : ''}`}
          onClick={() => setMenuOpen(false)}
        >
          About
        </Link>
      </div>

      <div className="navbar-actions">
        <ThemeToggle />
        {user ? (
          <>
            <span className="navbar-user">{user.name || user.email}</span>
            <button className="btn btn-link" onClick={handleSignOut}>Sign Out</butto
          </>
        ) : (
          <Link to="/signin" className="navbar-link auth-link">
            Sign In
          </Link>
        )}
      </div>

      <button
        className="navbar-toggle"
```

```
            </>
          ) : (
            <Link to="/signin" className="navbar-link auth-link">
              Sign In
            </Link>
          )}
        </div>

        <button
          className="navbar-toggle"
          aria-label="Toggle menu"
          aria-expanded={menuOpen}
          onClick={() => setMenuOpen((s) => !s)}
        >
          <span className="hamburger" aria-hidden="true">
            <span className="line" />
            <span className="line" />
            <span className="line" />
          </span>
        </button>
      </div>
    </motion.nav>
  );
};

export default Navbar;
```

**Code Description**

- React, useState, and useEffect are imported to manage component state and lifecycle behavior.
- Link, useLocation, and useNavigate from react-router-dom handle navigation and detect the current page.
- Framer Motion is used to animate the navbar sliding down when loaded.
- ThemeToggle handles light/dark mode switching.
- Navbar.css contains styling for layout, colors, and responsiveness.
- getCurrentUser and signOut are imported from localStorage utilities to manage user login state.
- user: Stores the currently logged-in user (retrieved from localStorage).
- menuOpen: Tracks whether the mobile hamburger menu is open.
- The current URL path is obtained using useLocation().
- When the location changes, the component updates the user state with the latest user data from localStorage.
- Another useEffect automatically closes the mobile menu whenever the route changes.
- handleSignOut(): Calls signOut() to remove the user from localStorage. Clears the user state. Navigates back to the home page.
- The <motion.nav> element animates downward on mount using Framer Motion.
- Brand link with an icon and the text “CookBook”.
- Navigation links (Home, My Cookbook, Upload, Community, About).
- The active page is highlighted using a conditional class (active).
- Clicking a link closes the mobile menu.
- Navbar actions section: Theme toggle button. Shows the user’s name/email and a “Sign Out” button if logged in. Otherwise shows a “Sign In” link. Hamburger menu button for mobile screens, which toggles menuOpen.
- The Navbar component is exported as the default export.

# 7.Code RatingStars.jsx

```jsx
const RatingStars = ({ recipeId, onRatingChange }) => {

  const displayRating = hoveredRating || userRating;

  return (
    <div className="rating-stars">
      <div className="rating-stars-display">
        {[1, 2, 3, 4, 5].map((star) => (
          <motion.button
            key={star}
            className={`star ${star <= displayRating ? 'active' : ''}`}
            onClick={() => handleRatingClick(star)}
            onMouseEnter={() => setHoveredRating(star)}
            onMouseLeave={() => setHoveredRating(0)}
            whileHover={{ scale: 1.2 }}
            whileTap={{ scale: 0.9 }}
            aria-label={`Rate ${star} stars`}
          >
            <i className={`fas fa-star ${star <= displayRating ? 'filled' : ''}`}></i>
          </motion.button>
        ))}
      </div>
      {averageRating > 0 && (
        <div className="rating-info">
          <span className="average-rating">
            {averageRating.toFixed(1)} / 5.0
          </span>
          <span className="rating-count">
            ({ratingCount} {ratingCount === 1 ? 'rating' : 'ratings'})
          </span>
        </div>
      )}
    </div>
  );
};
```

```jsx
  const displayRating = hoveredRating || userRating;

  return (
    <div className="rating-stars">
      <div className="rating-stars-display">
        {[1, 2, 3, 4, 5].map((star) => (
          <motion.button
            key={star}
            className={`star ${star <= displayRating ? 'active' : ''}`}
            onClick={() => handleRatingClick(star)}
            onMouseEnter={() => setHoveredRating(star)}
            onMouseLeave={() => setHoveredRating(0)}
            whileHover={{ scale: 1.2 }}
            whileTap={{ scale: 0.9 }}
            aria-label={`Rate ${star} stars`}
          >
            <i className={`fas fa-star ${star <= displayRating ? 'filled' : ''}`}></i>
          </motion.button>
        ))}
      </div>
      {averageRating > 0 && (
        <div className="rating-info">
          <span className="average-rating">
            {averageRating.toFixed(1)} / 5.0
          </span>
          <span className="rating-count">
            ({ratingCount} {ratingCount === 1 ? 'rating' : 'ratings'})
          </span>
        </div>
      )}
    </div>
  );
};

export default RatingStars;
```

**Code Description**

- React, along with useState and useEffect, is imported to manage rating state and lifecycle updates.
- Framer Motion is used to animate star buttons with hover and tap effects.
- Local storage helper functions — getRecipeRating, setUserRating, and getUserRating — are imported to store and retrieve rating data for each recipe.
- RatingStars.css is imported to style the star icons and rating display.
- hoveredRating: Tracks which star the user is currently hovering over.
- userRating: Stores the rating the current user has previously given this recipe.
- averageRating: Stores the average rating calculated from all users
- ratingCount: Stores how many total ratings the recipe has.
- useEffect (Load Ratings):Runs when the component mounts or when recipeId changes.Retrieves the current user's rating via getUserRating(recipeId).Retrieves the recipe's overall average rating and rating count using getRecipeRating(recipeId).
- handleRatingClick(rating):Saves the user's new rating using setUserRating(recipeId, rating).Updates the local user rating state.Refreshes the recipe's average rating and total rating count.Calls onRatingChange(rating) if the parent component wants to respond to rating changes.
- displayRating:Determines which rating should be visually displayed:If the user is hovering over a star → show the hovered rating.Otherwise → show the user's saved rating.
- UI Structure:A row of five clickable, animated star buttons created using Framer Motion.Stars light up (active) if they are less than or equal to the displayed rating.Hovering enlarges the stars; clicking sets a rating.

## 8.Code RecipeDetails.jsx

```
import React, { useState, useEffect } from "react";
import { useParams, useNavigate } from "react-router-dom";
import { motion } from "framer-motion";
import {
  getRecipeById,
  parseRecipeIngredients,
  getUploadedRecipeFromServer,
} from "../utils/api";
import {
  saveRecipe,
  removeSavedRecipe,
  isRecipeSaved,
  getUploadedRecipes,
} from "../utils/localStorage";
import IngredientChecklist from "./IngredientChecklist";
import Timer from "./Timer";
import RatingStars from "./RatingStars";
import "./RecipeDetails.css";

const RecipeDetails = () => {
  const { id } = useParams();
  const navigate = useNavigate();
  const [recipe, setRecipe] = useState(null);
  const [loading, setLoading] = useState(true);
  const [saved, setSaved] = useState(false);
  const [ingredients, setIngredients] = useState([]);
  const [instructions, setInstructions] = useState([]);

  useEffect(() => {
    const loadRecipe = async () => {
      setLoading(true);

      // Handle user-uploaded recipe
      if (id.startsWith("user_")) {
        const uploadedRecipes = getUploadedRecipes();
        const recipeData = uploadedRecipes.find((r) => r.id === id);
        if (recipeData) {
          setRecipe({
            ...recipeData,
            strMeal: recipeData.name || recipeData.strMeal,
            strMealThumb: recipeData.image || recipeData.strMealThumb,
            strCategory: recipeData.category || recipeData.strCategory,
            strArea: recipeData.cuisine || recipeData.strArea,
            strInstructions: recipeData.steps
```

```
import React, { useState, useEffect } from "react";
import { useParams, useNavigate } from "react-router-dom";
import { motion } from "framer-motion";
import {
  getRecipeById,
  parseRecipeIngredients,
  getUploadedRecipeFromServer,
} from "../utils/api";
import {
  saveRecipe,
  removeSavedRecipe,
  isRecipeSaved,
  getUploadedRecipes,
} from "../utils/localStorage";
import IngredientChecklist from "./IngredientChecklist";
import Timer from "./Timer";
import RatingStars from "./RatingStars";
import "./RecipeDetails.css";

const RecipeDetails = () => {
  const { id } = useParams();
  const navigate = useNavigate();
  const [recipe, setRecipe] = useState(null);
  const [loading, setLoading] = useState(true);
  const [saved, setSaved] = useState(false);
  const [ingredients, setIngredients] = useState([]);
  const [instructions, setInstructions] = useState([]);

  useEffect(() => {
    const loadRecipe = async () => {
      setLoading(true);

      // Handle user-uploaded recipe
      if (id.startsWith("user_")) {
        const uploadedRecipes = getUploadedRecipes();
        const recipeData = uploadedRecipes.find((r) => r.id === id);
        if (recipeData) {
          setRecipe({
            ...recipeData,
            strMeal: recipeData.name || recipeData.strMeal,
            strMealThumb: recipeData.image || recipeData.strMealThumb,
            strCategory: recipeData.category || recipeData.strCategory,
            strArea: recipeData.cuisine || recipeData.strArea,
            strInstructions: recipeData.steps
```

```
            strInstructions: recipeData.steps
              ? recipeData.steps.join("\n")
              : recipeData.strInstructions,
          });
          setSaved(isRecipeSaved(id));
          setIngredients(recipeData.ingredients || []);
          setInstructions(recipeData.steps || []);
        }
      } else {
        // Handle API recipe
        let recipeData = await getRecipeById(id);
        if (recipeData) {
          setRecipe(recipeData);
          setSaved(isRecipeSaved(id));
          const parsedIngredients = parseRecipeIngredients(recipeData);
          setIngredients(parsedIngredients);
          const steps = recipeData.strInstructions
            .split("\n")
            .filter((step) => step.trim());
          setInstructions(steps);
        } else {
          // Fallback to uploaded recipe from server
          const uploaded = await getUploadedRecipeFromServer(id).catch(
            () => null
          );
          if (uploaded) {
            setRecipe({
              ...uploaded,
              strMeal: uploaded.name || uploaded.strMeal,
              strMealThumb: uploaded.image || uploaded.strMealThumb,
              strCategory: uploaded.category || uploaded.strCategory,
              strArea: uploaded.cuisine || uploaded.strArea,
              strInstructions: uploaded.steps
                ? uploaded.steps.join("\n")
                : uploaded.strInstructions,
            });
            setSaved(isRecipeSaved(id));
            setIngredients(uploaded.ingredients || []);
            const steps =
              uploaded.steps ||
              (uploaded.strInstructions
```

```
      </div>
    );
  }

  const prepTime = recipe.prepTime || 30; // Default 30 minutes

  return (
    <motion.div
      className="recipe-details"
      initial={{ opacity: 0 }}
      animate={{ opacity: 1 }}
      transition={{ duration: 0.5 }}
    >
      <div className="container">
        <button
          className="btn btn-outline back-button"
          onClick={() => navigate(-1)}
        >
          <i className="fas fa-arrow-left"></i> Back
        </button>

        <div className="recipe-details-header">
          <div className="recipe-details-image">
            <img
              src={
                recipe.strMealThumb ||
                recipe.image ||
                "https://via.placeholder.com/400x300?text=Recipe+Image"
              }
              alt={recipe.strMeal || recipe.name}
              onError={(e) => {
                e.target.src =
                  "https://via.placeholder.com/400x300?text=Recipe+Image";
              }}
            />
          </div>

          <div className="recipe-details-info">
            <h1>{recipe.strMeal || recipe.name}</h1>
            <div className="recipe-meta">
              {(recipe.strCategory || recipe.category) && (
                <span className="meta-item">
                  <i className="fas fa-tag"></i>{" "}
```

```
                <i className="fas fa-list"></i> Ingredients
              </h2>
              <IngredientChecklist ingredients={ingredients} />
            </div>

            <div className="recipe-section">
              <h2>
                <i className="fas fa-clock"></i> Prep Time
              </h2>
              <Timer initialMinutes={prepTime} />
            </div>

            <div className="recipe-section">
              <h2>
                <i className="fas fa-book-open"></i> Instructions
              </h2>
              <ol className="instructions-list">
                {instructions.map((step, index) => (
                  <motion.li
                    key={index}
                    initial={{ opacity: 0, x: -20 }}
                    animate={{ opacity: 1, x: 0 }}
                    transition={{ delay: index * 0.1 }}
                  >
                    {step}
                  </motion.li>
                ))}
              </ol>
            </div>
          </div>
        </div>
      </motion.div>
    );
  };

  export default RecipeDetails;
```

**Code Description**

- Imports React along with useState and useEffect to manage component state and lifecycle logic.
- Imports useParams and useNavigate from react-router-dom to read the recipe ID from the URL and enable navigation.
- Imports Framer Motion for fade-in animations.
- Imports API utility methods (getRecipeById, parseRecipeIngredients, getUploadedRecipeFromServer) and localStorage helpers (saveRecipe, removeSavedRecipe, isRecipeSaved, getUploadedRecipes).
- Imports UI components: IngredientChecklist, Timer, and RatingStars, plus CSS styling.
- The component reads the recipe ID from the URL and loads either:A user-uploaded recipe (IDs starting with "user_") from localStorage or server,
- A regular API recipe using getRecipeById.After loading, it sets recipe details, saved status, parsed ingredients, and instruction steps.
- Loading state and error states are handled:Shows a spinner while loading,Shows a friendly error message if recipe not found.handleSave() toggles whether a recipe is saved in the user's cookbook using localStorage.prepTime uses the recipe value or defaults to 30 minutes.
- The UI structure includes:A back button using navigate(-1).Recipe image, fallback image logic, recipe name, category, origin, and prep time metadata.Save/Unsave button that updates localStorage.RatingStars component for rating the recipe.Ingredients section with IngredientChecklist.Prep time section with a Timer component.Instructions section rendered as an animated ordered list with motion effects.The whole layout fades in using Framer Motion.The component exports RecipeDetails as the default.

**9.Code RecipeList.jsx**

```
import React from 'react';
import RecipeCard from './RecipeCard';
import './RecipeList.css';

const RecipeList = ({ recipes, loading, error }) => {
  if (loading) {
    return (
      <div className="recipe-list-loading">
        <div className="spinner"></div>
        <p>Loading recipes...</p>
      </div>
    );
  }

  if (error) {
    return (
      <div className="recipe-list-error">
        <i className="fas fa-exclamation-triangle"></i>
        <p>{error}</p>
      </div>
    );
  }

  if (!recipes || recipes.length === 0) {
    return (
      <div className="recipe-list-empty">
        <i className="fas fa-search"></i>
        <p>No recipes found. Try a different search!</p>
      </div>
    );
  }

  return (
    <div className="recipe-list">
      {recipes.map((recipe, index) => (
        <RecipeCard key={recipe.idMeal || recipe.id} recipe={recipe} index={index} />
      ))}
    </div>
  );
};

export default RecipeList;
```

## Code Description

- Imports React, RecipeCard, and the CSS file.
- Accepts three props: recipes, loading, error.
- Shows a loading spinner and “Loading recipes…” text.
- Shows an error icon and the error message.
- If recipes is empty or undefined:
- Shows a “No recipes found” message with a search icon.

- Maps through each recipe.Renders a RecipeCard for every recipe.Uses recipe.idMeal or recipe.id as the key.
- Wraps all recipe cards inside a grid container with the class recipe-list

## 10.Code SearchBar.jsx

```jsx
import React, { useState } from 'react';
import { motion } from 'framer-motion';
import './SearchBar.css';

const SearchBar = ({ onSearch, placeholder = 'Search recipes...' }) => {
  const [query, setQuery] = useState('');

  const handleSubmit = (e) => {
    e.preventDefault();
    if (query.trim()) {
      onSearch(query.trim());
    }
  };

  const handleSearchClick = () => {
    if (query.trim()) {
      onSearch(query.trim());
    }
  };

  return (
    <motion.form
      className="search-bar"
      onSubmit={handleSubmit}
      initial={{ opacity: 0, y: -20 }}
      animate={{ opacity: 1, y: 0 }}
      transition={{ duration: 0.5 }}
    >
      <div className="search-input-wrapper">
        <input
          type="text"
          className="search-input"
          value={query}
          onChange={(e) => setQuery(e.target.value)}
          placeholder={placeholder}
        />
        <button
          type="submit"
          className="search-button"
          onClick={handleSearchClick}
        >
          <i className="fas fa-search"></i>
        </button>
      </div>
    </motion.form>
  );
};

export default SearchBar;
```

**Code Description**

- Imports React, useState, framer-motion, and the CSS file.
- Accepts two props:onSearch (function to execute a search).placeholder (optional text for the input, default: “Search recipes…”)
- Uses useState to store the search query entered by the user.
- handleSubmit prevents form refresh and triggers onSearch if the query isn’t empty.
- handleSearchClick runs the same search logic when the button is clicked.
- Renders a motion-enhanced <form> with fade-and-slide animation.
- Inside the form:An input field bound to the query state.
- A search button with a search icon.
- Submitting the form or clicking the button triggers the search.

**11.Code UploadRecipeForm.jsx**

```
            name="name"
            value={formData.name}
            onChange={handleChange}
            placeholder="Enter recipe name"
            required
          />
        </div>

        <div class="form-row">
          <div class="form-group">
            <label>Cuisine</label>
            <input
              type="text"
              name="cuisine"
              value={formData.cuisine}
              onChange={handleChange}
              placeholder="e.g., Indian"
              required
            />
          </div>
          <div class="form-group">
            <label>Category</label>
            <input
              type="text"
              name="category"
              value={formData.category}
              onChange={handleChange}
              placeholder="e.g., Dessert"
              required
            />
          </div>
          <div class="form-group">
            <label>Prep Time (minutes)</label>
            <input
              type="number"
              name="prepTime"
              value={formData.prepTime}
              onChange={handleChange}
              placeholder="30"
              min="1"
              required
            />
          </div>
        </div>

        <div class="form-group">
          <label>Image URL</label>
          <input
            type="url"
            name="image"
            value={formData.image}
            onChange={handleChange}
            placeholder="https://example.com/image.jpg"
          />
        </div>

        <div class="form-group">
          <div class="form-group-header">
            <label>Ingredients</label>
            <button type="button" className="btn btn-outline" onClick={addIngredient}>
              <i class="fas fa-plus"></i> Add Ingredient
            </button>
          </div>
          {formData.ingredients.map((ingredient, index) => (
            <div key={index} class="ingredient-row">
              <input
                type="text"
                placeholder="e.g., 1 cup"
                value={ingredient.measure}
                onChange={(e) =>
                  handleIngredientChange(index, "measure", e.target.value)
                }
              />
              <input
                type="text"
                placeholder="e.g., flour"
                value={ingredient.name}
                onChange={(e) =>
```

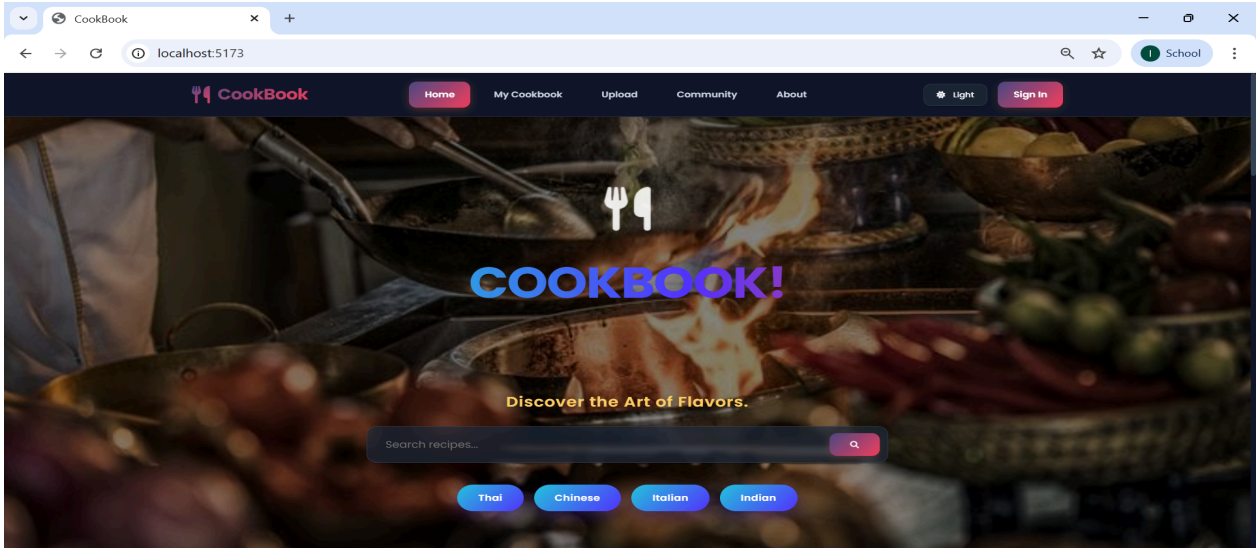
```
            <input
              type="text"
              placeholder="e.g., 1 cup"
              value={ingredient.measure}
              onChange={(e) =>
                handleIngredientChange(index, "measure", e.target.value)
              }
            />
            <input
              type="text"
              placeholder="e.g., flour"
              value={ingredient.name}
              onChange={(e) =>
                handleIngredientChange(index, "name", e.target.value)
              }
              required
            />
            {formData.ingredients.length > 1 && (
              <button
                type="button"
                className="btn-remove"
                onClick={() => removeIngredient(index)}
              >
                <i className="fas fa-times"></i>
              </button>
            )}
          </div>
        ))}
      </div>

      <div className="form-group">
        <div className="form-group-header">
          <label>Instructions</label>
          <button type="button" className="btn btn-outline" onClick={addStep}>
            <i className="fas fa-plus"></i> Add Step
          </button>
        </div>
        {formData.steps.map((step, index) => (
          <div key={index} className="step-row">
            <span className="step-number">{index + 1}</span>
            <textarea
              placeholder="Enter step instructions"
              value={step}
              onChange={(e) => handleStepChange(index, e.target.value)}
              required
            />
            {formData.steps.length > 1 && (
              <button
                type="button"
                className="btn-remove"
                onClick={() => removeStep(index)}
              >
                <i className="fas fa-times"></i>
              </button>
            )}
          </div>
        ))}
      </div>

      <button type="submit" className="btn btn-primary" disabled={submitting}>
        {submitting ? (
          <>
            <div className="spinner-small"></div> Submitting...
          </>
        ) : (
          <>
            <i className="fas fa-upload"></i> Submit Recipe
          </>
        )}
      </button>
    </motion.form>
  );
};

export default UploadRecipeForm;
```

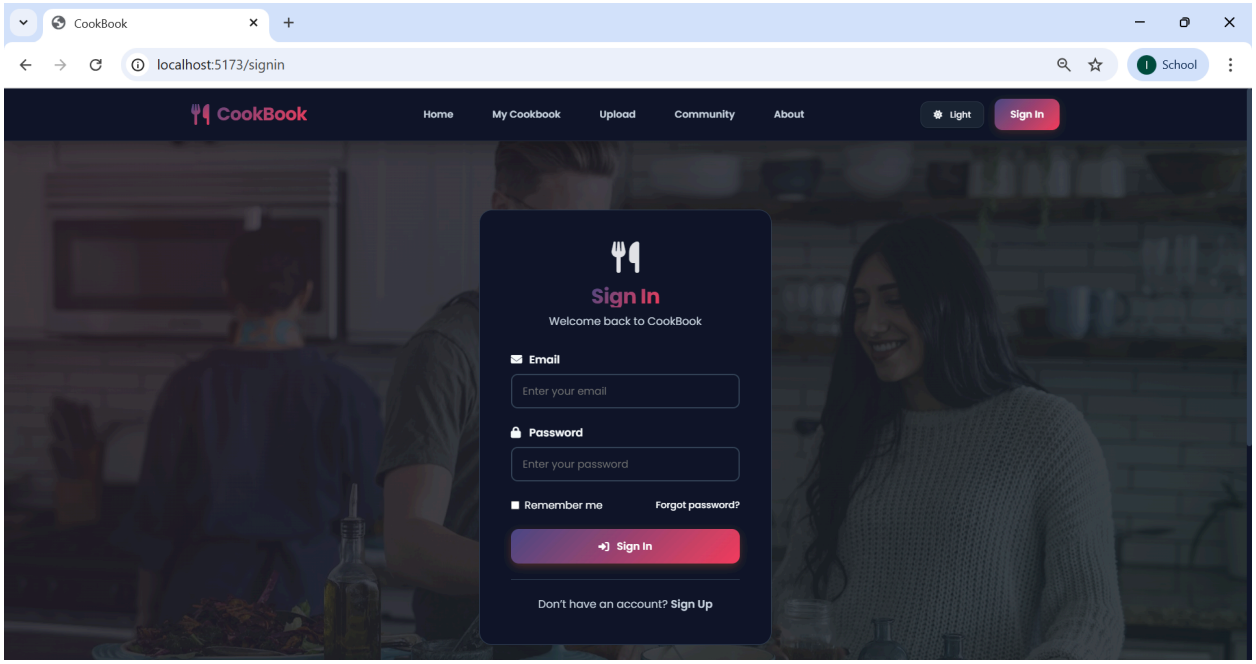
```
            <input
              type="text"
              placeholder="e.g., 1 cup"
              value={ingredient.measure}
              onChange={(e) =>
                handleIngredientChange(index, "measure", e.target.value)
              }
            />
            <input
              type="text"
              placeholder="e.g., flour"
              value={ingredient.name}
              onChange={(e) =>
                handleIngredientChange(index, "name", e.target.value)
              }
              required
            />
            {formData.ingredients.length > 1 && (
              <button
                type="button"
                className="btn-remove"
                onClick={() => removeIngredient(index)}
              >
                <i className="fas fa-times"></i>
              </button>
            )}
          </div>
        ))}
      </div>

      <div className="form-group">
        <div className="form-group-header">
          <label>Instructions</label>
          <button type="button" className="btn btn-outline" onClick={addStep}>
            <i className="fas fa-plus"></i> Add Step
          </button>
        </div>
        {formData.steps.map((step, index) => (
          <div key={index} className="step-row">
            <span className="step-number">{index + 1}</span>
            <textarea
              placeholder="Enter step instructions"
              value={step}
              onChange={(e) => handleStepChange(index, e.target.value)}
              required
            />
            {formData.steps.length > 1 && (
              <button
                type="button"
                className="btn-remove"
                onClick={() => removeStep(index)}
              >
                <i className="fas fa-times"></i>
              </button>
            )}
          </div>
        ))}
      </div>

      <button type="submit" className="btn btn-primary" disabled={submitting}>
        {submitting ? (
          <>
            <div className="spinner-small"></div> Submitting...
          </>
        ) : (
          <>
            <i className="fas fa-upload"></i> Submit Recipe
          </>
        )}
      </button>
    </motion.form>
  );
};

export default UploadRecipeForm;
```

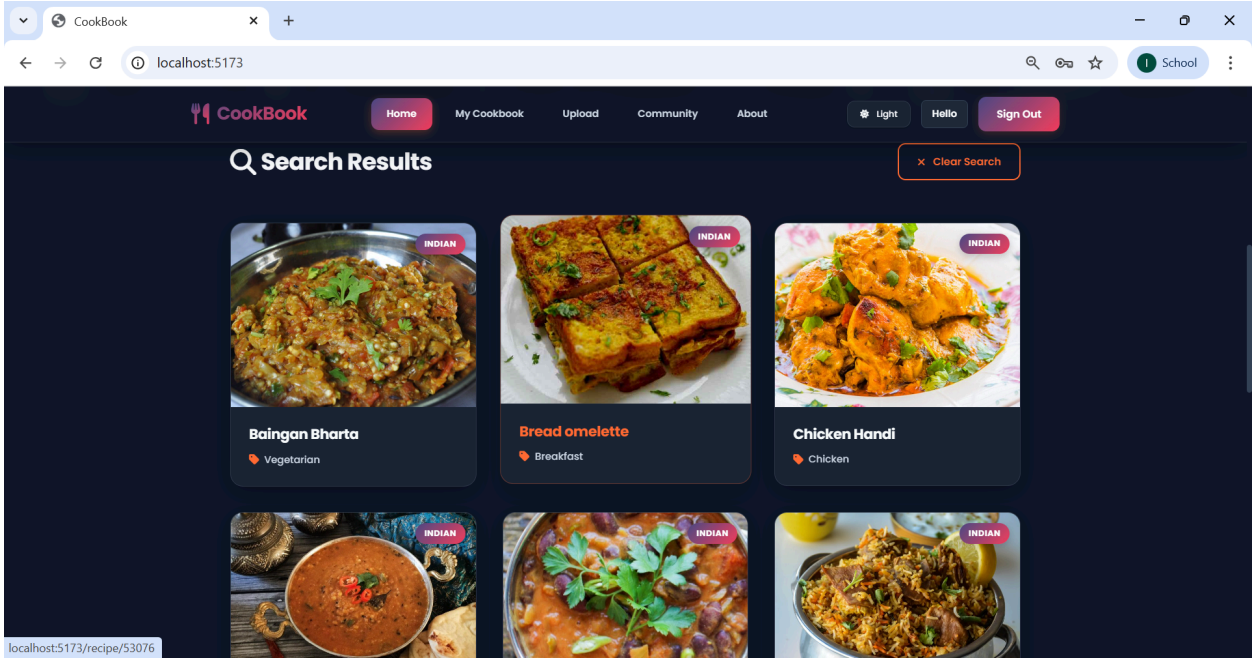
**Code Description**

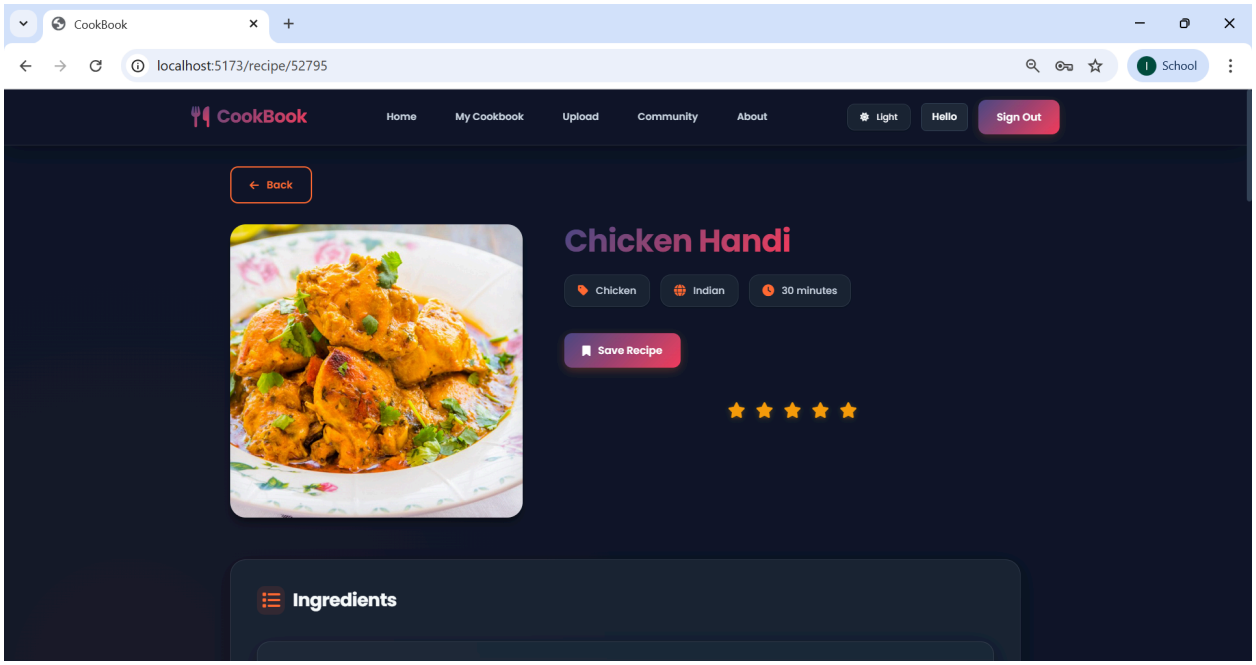
- Imports React hooks, Framer Motion, local storage helpers, API helpers, and CSS.
- onSuccess → callback after successful save
- initialData → pre-filled recipe when editing
- editMode → toggles between create and edit mode
- Component state:formData holds all recipe fields.submitting indicates form submission status
- useEffect loads existing recipe data into the form when editing.
- Handles input updates:handleChange for basic inputs.handleIngredientChange, addIngredient, removeIngredient
- handleStepChange, addStep, removeStep
- handleSubmit builds a recipe object, processes ingredients/steps, and:Sends PUT or updates local storage when editing.Sends POST or saves locally when creating
- Resets form and runs onSuccess after successful save.
- Renders animated <form> containing:
- Recipe name, cuisine, category, prep time, image URL
- Dynamic ingredient list with add/remove buttons
- Dynamic steps list with add/remove buttons
- Submit button with loading state
- Exports UploadRecipeForm as default.

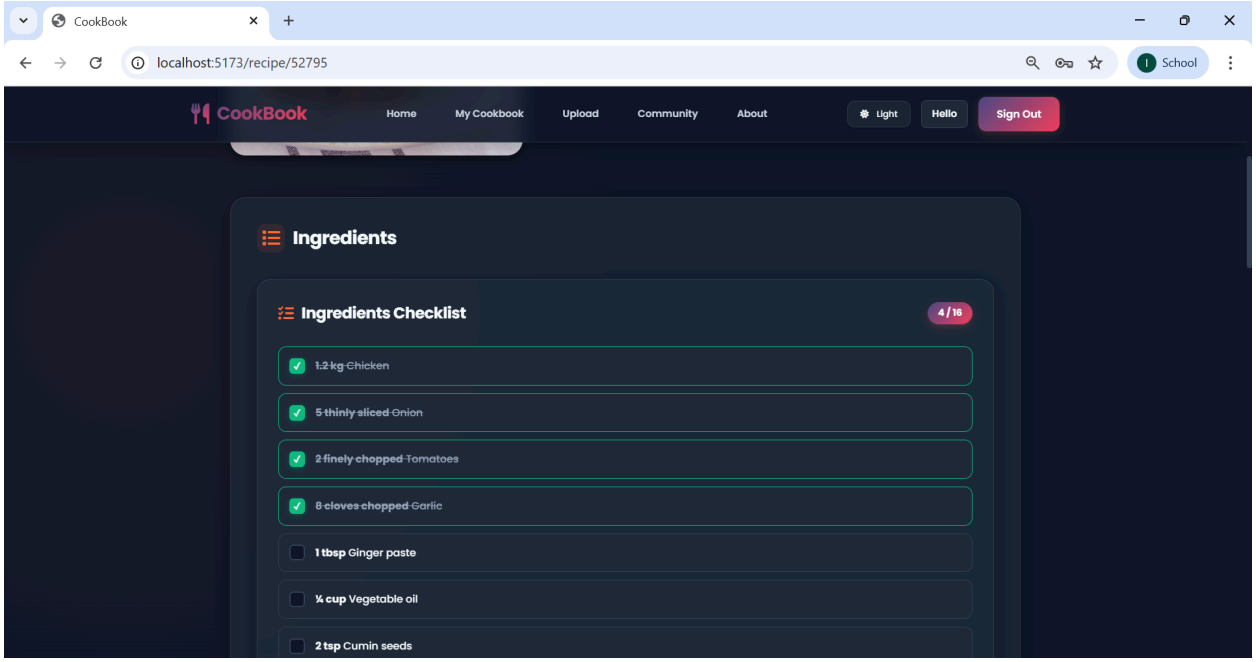
**Project Execution & Output:**

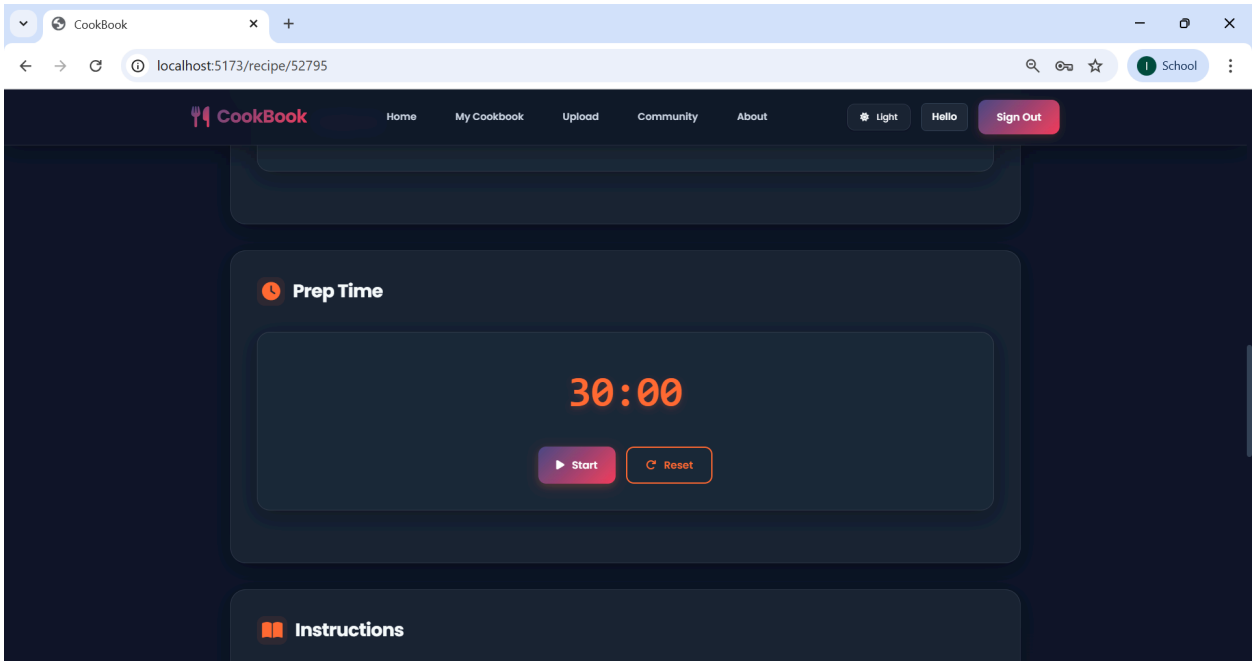

CookBook
localhost:5173
School
CookBook
Home
My Cookbook
Upload
Community
About
Light
Sign In
COOKBOOK!
Discover the Art of Flavors.
Search recipes...
Thai
Chinese
Italian
Indian

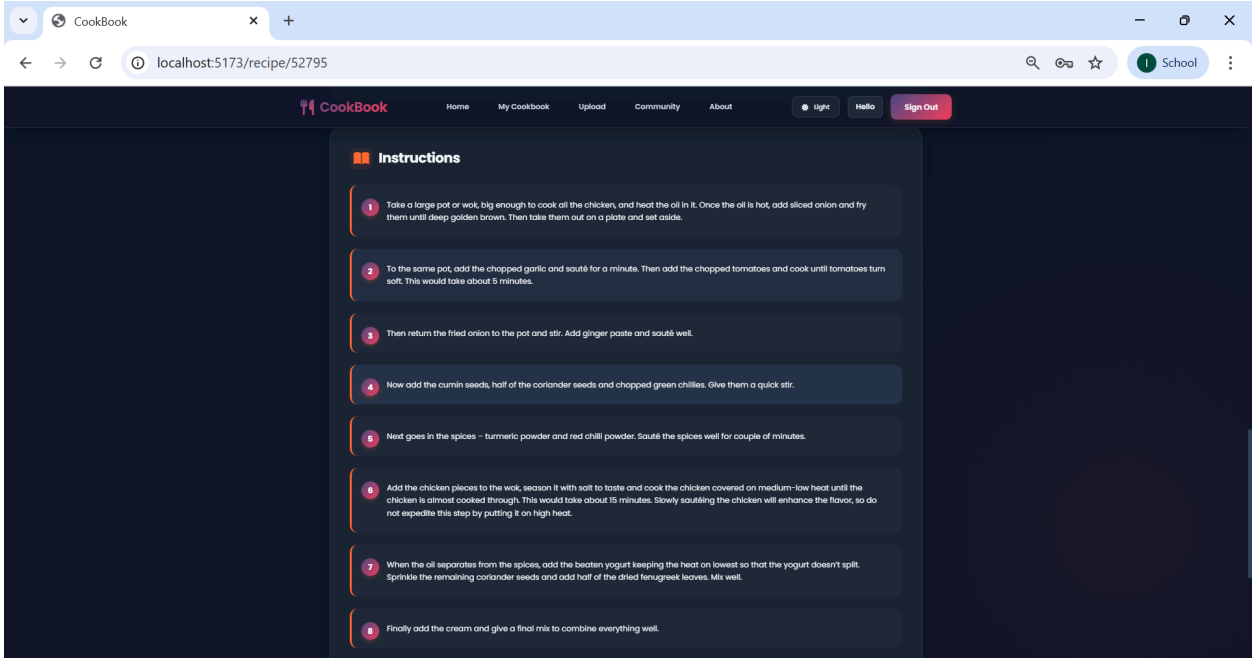

CookBook
localhost:5173/signin
School
CookBook
Home
My Cookbook
Upload
Community
About
Light
Sign In
Sign In
Welcome back to CookBook
Email
Enter your email
Password
Enter your password
Remember me
Forgot password?
Sign In
Don't have an account? Sign Up

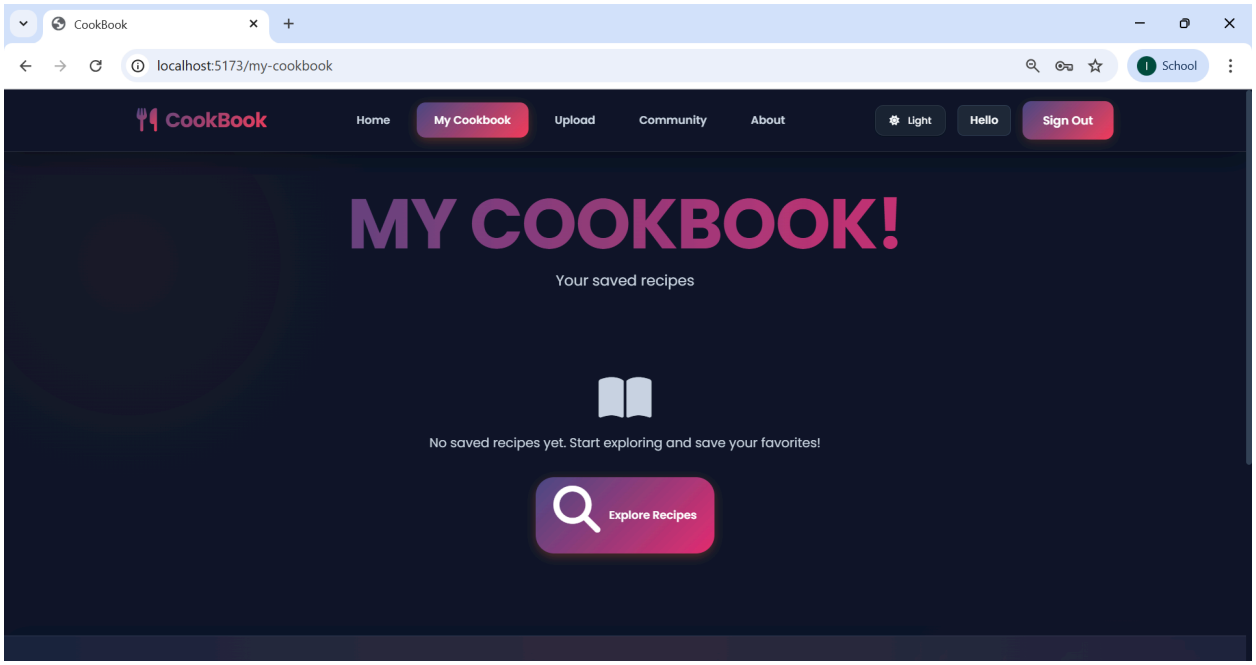

CookBook
localhost:5173
School
CookBook
Home
My Cookbook
Upload
Community
About
Light
Hello
Sign Out
Search Results
Clear Search
INDIAN
Baingan Bharta
Vegetarian
Bread omelette
Breakfast
Chicken Handi
Chicken
localhost:5173/recipe/53076

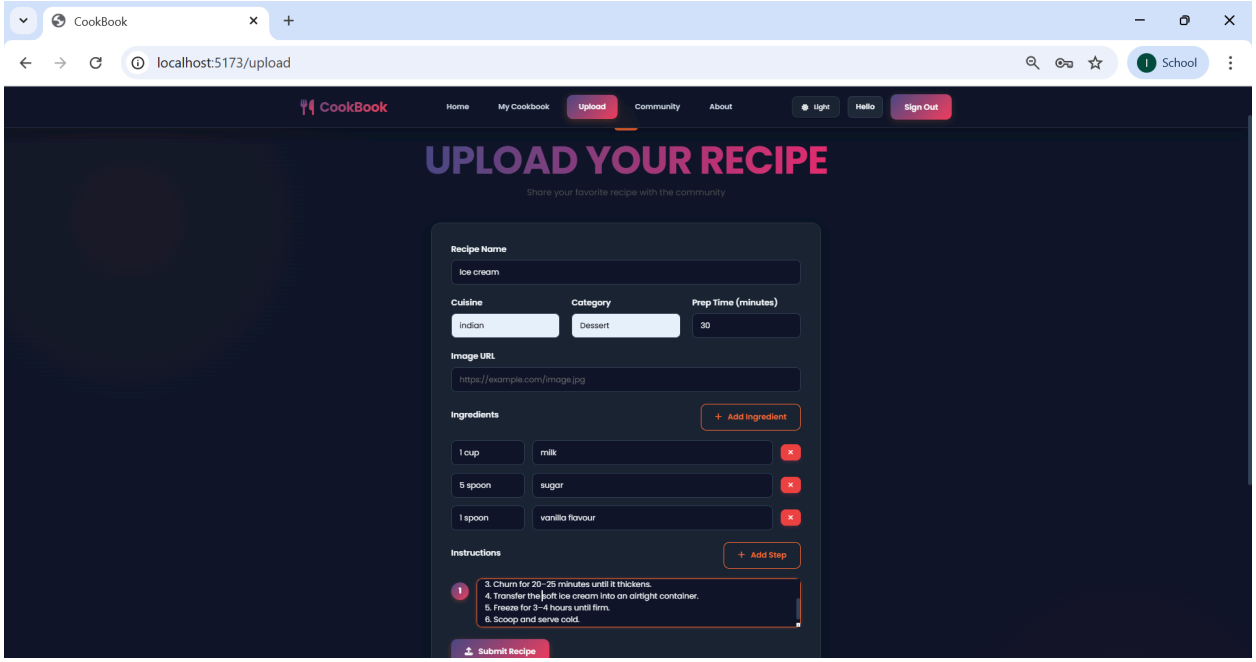

CookBook
localhost:5173/recipe/52795
School
CookBook
Home
My Cookbook
Upload
Community
About
Light
Hello
Sign Out
Back
Chicken Handi
Chicken
Indian
30 minutes
Save Recipe
Ingredients

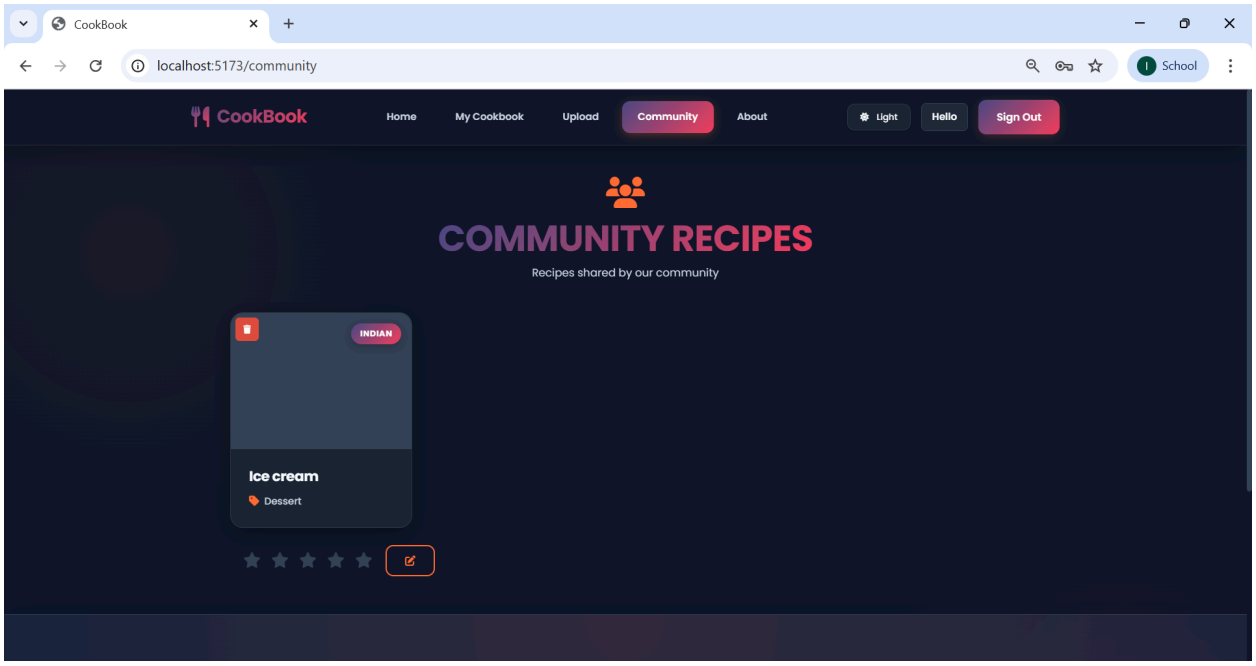

CookBook
localhost:5173/recipe/52795
School
CookBook
Home
My Cookbook
Upload
Community
About
Light
Hello
Sign Out
Ingredients
Ingredients Checklist
4 / 16
1.2 kg Chicken
5 thinly sliced Onion
2 finely chopped Tomatoes
8 cloves chopped Garlic
1 tbsp Ginger paste
¼ cup Vegetable oil
2 tsp Cumin seeds

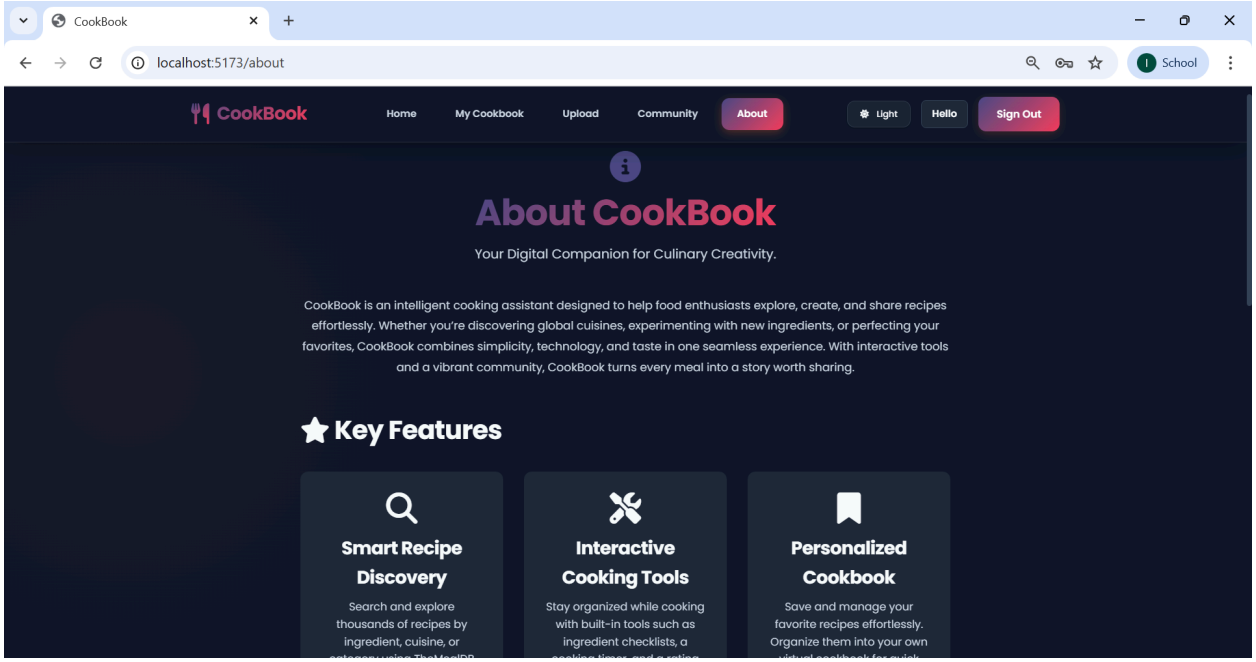

CookBook
localhost:5173/recipe/52795
School
CookBook
Home
My Cookbook
Upload
Community
About
Light
Hello
Sign Out
Prep Time
30:00
Start
Reset
Instructions


CookBook
localhost:5173/recipe/52795
School
CookBook
Home
My Cookbook
Upload
Community
About
Light
Hello
Sign Out
Instructions
1 Take a large pot or wok, big enough to cook all the chicken, and heat the oil in it. Once the oil is hot, add sliced onion and fry them until deep golden brown. Then take them out on a plate and set aside.
2 To the same pot, add the chopped garlic and sauté for a minute. Then add the chopped tomatoes and cook until tomatoes turn soft. This would take about 5 minutes.
3 Then return the fried onion to the pot and stir. Add ginger paste and sauté well.
4 Now add the cumin seeds, half of the coriander seeds and chopped green chillies. Give them a quick stir.
5 Next goes in the spices – turmeric powder and red chilli powder. Sauté the spices well for couple of minutes.
6 Add the chicken pieces to the wok, season it with salt to taste and cook the chicken covered on medium-low heat until the chicken is almost cooked through. This would take about 15 minutes. Slowly sautéing the chicken will enhance the flavor, so do not expedite this step by putting it on high heat.
7 When the oil separates from the spices, add the beaten yogurt keeping the heat on lowest so that the yogurt doesn't split. Sprinkle the remaining coriander seeds and add half of the dried fenugreek leaves. Mix well.
8 Finally add the cream and give a final mix to combine everything well.


CookBook
localhost:5173/my-cookbook
School
CookBook
Home
My Cookbook
Upload
Community
About
Light
Hello
Sign Out
MY COOKBOOK!
Your saved recipes
No saved recipes yet. Start exploring and save your favorites!
Explore Recipes


CookBook
localhost:5173/upload
School
CookBook
Home
My Cookbook
Upload
Community
About
Light
Hello
Sign Out
UPLOAD YOUR RECIPE
Share your favorite recipe with the community
Recipe Name
Ice cream
Cuisine
indian
Category
Dessert
Prep Time (minutes)
30
Image URL
https://example.com/image.jpg
Ingredients
+ Add Ingredient
1 cup
milk
5 spoon
sugar
1 spoon
vanilla flavour
Instructions
+ Add Step
1
3. Churn for 20-25 minutes until it thickens.
4. Transfer the soft ice cream into an airtight container.
5. Freeze for 3-4 hours until firm.
6. Scoop and serve cold.
Submit Recipe


CookBook
localhost:5173/community
School
CookBook
Home
My Cookbook
Upload
Community
About
Light
Hello
Sign Out
COMMUNITY RECIPES
Recipes shared by our community
INDIAN
Ice cream
Dessert




## Project DemoLink:

[https://drive.google.com/drive/folders/1MKXKPvecdlBXfDs9M9X-qwlIvwBENUns?usp=sharing](https://drive.google.com/drive/folders/1MKXKPvecdlBXfDs9M9X-qwlIvwBENUns?usp=sharing)