1. INTRODUCTION

Cookbook is a modern, web-based application designed to transform the way individuals interact with recipes, whether they are novice cooks or seasoned chefs. As a user-friendly platform, Cookbook helps users discover, organize, and create recipes with ease, offering an intuitive experience that appeals to a broad range of culinary enthusiasts. The application is built using **React.js**, ensuring a fast, responsive, and dynamic interface that adapts seamlessly to the needs of users. With its vast collection of recipes sourced from the **Meals DB API**, Cookbook serves as an all-in-one virtual kitchen assistant, allowing users to explore new dishes, manage their favourite recipes, and experiment with various cuisines. At the core of Cookbook is a powerful search engine that allows users to find recipes based on ingredients, cuisine type, dietary preferences, and more. Whether you’re looking for quick dinner ideas, preparing for a special event, or trying to adhere to specific dietary restrictions, Cookbook’s advanced search options make it easy to filter recipes that match your needs. Users can also save their favourite recipes, create meal plans, and share their culinary creations with a like-minded community. Cookbook’s visual recipe browsing feature offers users the ability to explore recipe categories through curated image galleries, making the recipe discovery process not only efficient but also enjoyable. The clean and modern design ensures a pleasant experience for all users, with easy navigation and clear, concise categories. Whether you are a beginner looking for simple meals or a professional chef looking for new challenges, Cookbook provides a platform to inspire culinary creativity. The platform’s integration with the **Meals DB API** offers an expansive library of international recipes, giving users access to a wide variety of dishes from around the world. In addition, Cookbook’s ability to connect to external data feeds through **Rapid API** ensures a continually updated and diverse collection of recipes. As the digital recipe management tool of the future, Cookbook makes it easier than ever to store and share recipes with friends, family, and a growing community of cooking enthusiasts. Cookbook’s vision is to redefine the recipe discovery and management process by making cooking fun, engaging, and accessible to anyone, regardless of their skill level. The application fosters a sense of community where users can share their culinary experiences, collaborate on recipes, and support each other in their cooking endeavours. With Cookbook, every recipe becomes an adventure waiting to be discovered, explored, and savoured. Whether you're looking to recreate an old favourite or discover something new, Cookbook is the ultimate kitchen companion that will elevate your cooking experience to new heights.

2.SYSTEM STUDY

EXISTING SYSTEM

The existing systems for recipe management and culinary exploration mainly revolve around individual cooking websites, recipe blogs, and mobile applications. Many of these systems allow users to discover and save recipes but lack the integrated features and streamlined experience offered by Cookbook. Here's a breakdown of existing systems in the recipe discovery and management space:

**1. Recipe Websites and Blogs**

A large portion of recipe discovery today happens on websites and blogs where users can search for specific recipes based on ingredients, cuisine, or dietary needs. These sites often have basic search functionality and allow users to save or print recipes. However, they generally lack advanced features such as meal planning, personalised recommendations, or integration with other tools like grocery shopping apps.

* **Pros:**
  + Extensive variety of recipes.
  + Can be very detailed, with step-by-step instructions and pictures.
  + Access to user reviews and ratings.
* **Cons:**
  + Limited personalization: Users generally have to manually search through categories without much customization.
  + User experience is often fragmented across different sites, with no unified interface.
  + Recipe saving is often cumbersome and may require bookmarking or third-party tools.
  + Limited interaction with other users or cooking communities.

**2. Recipe Apps (e.g., Yummly, All Recipes, Epicurious)**

These mobile applications focus on delivering a curated recipe collection that users can browse, save, and share. Some apps offer meal planning, grocery list generation, and integration with smart devices (e.g., smart ovens or fridges). Yummly, for instance, provides personalized recipe recommendations based on a user’s preferences and dietary needs.

* **Pros:**
  + Mobile-first experience allows users to easily browse recipes on the go.
  + Some apps offer personalized recommendations and meal planning.
  + Users can create shopping lists and manage recipes within the app.
* **Cons:**
  + Many apps require users to create an account or subscribe for premium features.
  + The user interface can sometimes feel cluttered or overly complex, making it difficult to find what you need quickly.
  + Limited interaction with the broader cooking community beyond recipe sharing.
  + Some apps do not offer an extensive database or have limited recipe diversity.

**3. Social Media Platforms (e.g., Pinterest, Instagram)**

Social media platforms like Pinterest and Instagram are frequently used by food enthusiasts to discover new recipes. Users can follow their favourite food bloggers, chefs, and culinary influencers to explore a variety of recipes and ideas.

* **Pros:**
  + Access to a wide range of user-generated content and influencer-driven recipes.
  + Visually appealing and interactive, especially on platforms like Instagram.
  + Easy sharing and collaborative features with friends and communities.
* **Cons:**
  + Searching for specific recipes can be time-consuming and inefficient.
  + There’s no structured way to save or organize recipes outside of bookmarking or pinning.
  + Not a dedicated recipe platform, so recipe details may be incomplete, and sourcing is often unclear.

**4. Recipe Management Tools (e.g., Paprika, Evernote, My Cookbook)**

These tools allow users to save and organise recipes, create meal plans, and generate shopping lists. They often provide more structured ways to manage culinary content, allowing users to store recipes offline, tag recipes by categories, and even scale ingredients based on the number of servings.

* **Pros:**
  + Great for personal organization: recipes are neatly categorized, and meal planning is made simple.
  + Some tools allow users to add custom recipes and modify existing ones.
  + Features such as grocery lists and shopping integration are common.
* **Cons:**
  + Recipe discovery is often not as intuitive as other platforms, and there’s limited access to an expansive, curated recipe database.

2.2 PROPOSED SYSTEM

**Proposed System for Cookbook: Your Virtual Kitchen Assistant**

The **Cookbook: Your Virtual Kitchen Assistant** aims to elevate the current landscape of recipe discovery and management by integrating multiple features into a single, streamlined platform.

**1. Key Features of the Proposed System**

**Recipe Discovery with Advanced Search**

The Cookbook platform will feature an intuitive search system, allowing users to quickly find recipes based on multiple criteria:

* **Ingredient-based Search:** Users can search for recipes based on ingredients they already have in their kitchen.
* **Cuisine-based Search:** Recipes can be filtered by different global cuisines (e.g., Italian, Mexican, Chinese).
* **Dietary-based Search:** Filters for dietary restrictions (e.g., gluten-free, vegetarian, keto) will be available.
* **Meal Type Search:** Users can easily search for breakfast, lunch, dinner, desserts, and more.

This advanced search feature, powered by the **Meals DB API**, ensures that users can discover the exact recipes they want, efficiently narrowing down the options based on their preferences.

**Recipe Management and Personalization**

* **Save and Organize:** Users can save their favourite recipes to a personal recipe box, categorizing them by type, difficulty, or occasion.
* **Meal Planning:** The app will allow users to create meal plans for the week or month, helping them organize their meals and grocery shopping.
* **Personalized Recommendations:** Based on saved recipes, preferences, and browsing history, Cookbook will offer personalized recipe suggestions.
* **Custom Recipe Creation:** Users will have the option to create and upload their own recipes, which can be shared with others in the Cookbook community.

**Visual Recipe Browsing**

Cookbook will offer a visual browsing experience where users can view curated galleries of recipes based on different categories (e.g., seasonal, trending, dietary preferences). This feature enhances the discovery process, making it more engaging and visually appealing.

**User Interaction and Community Features**

* **Recipe Sharing:** Users will be able to share their favourite recipes with friends and the wider Cookbook community.
* **Comments and Ratings:** A rating system and comment section will allow users to give feedback on recipes, helping others decide which dishes to try.
* **Social Interaction:** Users can follow other cooks or chefs, engage in discussions, and discover new recipes from those they admire.

**Grocery List Integration**

When users select a recipe, the platform will automatically generate a grocery list with all the ingredients needed to make the dish. This list can be checked off as users shop, ensuring they don’t miss anything. It can also be exported to mobile apps or shared with others.

**Seamless Cross-Platform Experience**

The Cookbook platform will be a responsive web application built using **React.js**, providing users with a consistent and smooth experience across all devices — desktop, tablet, and mobile. Users will be able to access their saved recipes, meal plans, and shopping lists on the go, ensuring the platform is always available in the kitchen.

**Integration with Smart Devices (Future Scope)**

In the future, Cookbook could integrate with smart kitchen devices (e.g., smart ovens, refrigerators, or voice assistants like Amazon Alexa) to further enhance the cooking experience. For example, the app could provide step-by-step instructions while users are cooking, send alerts when it's time to check the dish, or even preheat the oven automatically for specific recipes.

**2. System Architecture**

The proposed system is designed with scalability and user experience in mind. The architecture of Cookbook will consist of the following key components:

**Frontend**

* **React.js:** The frontend will be built using **React.js** to create a dynamic, responsive, and modern single-page application (SPA). Reacts component-based structure ensures that the interface is modular, making it easy to update and expand with new features.
* **Redux or Context API:** For state management across different components, either **Redux** or **Context API** will be used to store global data such as user preferences, saved recipes, and meal plans.
* **CSS & Styled Components:** The app will use modern styling techniques such as **CSS3** and **Styled Components** to ensure a clean, attractive, and responsive design.

**Backend**

* **API Integration (Meals DB API via Rapid API):** The app will fetch data from the **Meals DB API** using **Rapid API**. This will provide access to a large database of recipes with images, ingredients, preparation instructions, and nutritional information.
* **User Authentication (Optional for Future Updates):** The backend can be extended with user authentication for saving personal recipes, preferences, and meal plans. Technologies like **JWT (JSON Web Tokens)** can be used to implement secure logins.

**Database (Future Scope)**

In the future, Cookbook may use a database to store user information such as:

* **Saved Recipes**
* **Meal Plans**
* **Shopping Lists**
* **User Comments and Ratings**

This would allow users to sign in, access their personalized data, and continue their cooking journey seamlessly across different devices.

**Hosting and Deployment**

The app will be deployed on cloud services like **Heroku**, **Vencel**, or **Netlify**, ensuring that it is scalable, secure, and always available to users.

**3. Proposed Workflow**

1. **User Registration (Optional):**
   * Users can create an account or continue as guests. Registered users can save their recipes, create meal plans, and share their creations with the community.
2. **Search for Recipes:**
   * Users can search for recipes based on ingredients, cuisine, meal type, or dietary preferences using the advanced search bar.
3. **View Recipe Details:**
   * Clicking on a recipe will open detailed instructions, ingredient lists, preparation time, and nutritional information.
4. **Save and Organize Recipes:**
   * Users can save recipes they like and categorize them into different folders (e.g., “Quick Meals,” “Healthy Options,” “Desserts”).
5. **Create Meal Plans:**
   * Users can create weekly or monthly meal plans, ensuring they always have a recipe ready for every occasion.
6. **Generate Grocery List:**
   * A shopping list is automatically generated based on the ingredients of the recipes added to the meal plan.
7. **User Interaction and Community:**
   * Users can leave ratings, share their experiences in the comment section, and engage with other community members.

**4. Benefits of the Proposed System**

* **Enhanced Discovery Experience:** With multiple search filters and personalized recommendations, Cookbook ensures users quickly find the recipes they need.
* **Organized Recipe Management:** The ability to save and categorize recipes, along with meal planning features, helps users stay organized and efficient.
* **Cross-Platform Consistency:** The web application is fully responsive, making it accessible on various devices, from mobile phones to desktops.
* **Community Engagement:** By allowing users to interact, share, and collaborate on recipes, Cookbook builds a vibrant cooking community.
* **Smart Integration (Future Scope):** Potential integration with smart kitchen devices and AI-driven features will further elevate the cooking experience.

4. FEASIBILITY STUDY

A feasibility study is conducted to evaluate whether the proposed system, Cookbook: Your Virtual Kitchen Assistant, can be developed and implemented successfully within the available constraints of time, resources, and technology. It assesses the technical, operational, economic, and legal feasibility of the project to ensure that the project is viable and sustainable. Below is the feasibility study for Cookbook:

1. Technical Feasibility

1.1 Technological Requirements

The proposed system will leverage modern web technologies that are widely accepted in the development community. Here’s a breakdown of the required technologies:

* Frontend:
  + React.js for building a dynamic and interactive user interface.
  + CSS3 and Styled Components for modern, responsive, and attractive design.
  + Redux or Context API for state management.
  + Axios for HTTP requests to interact with APIs.
* Backend (API Integration):
  + Meals DB API for accessing a vast collection of recipes from diverse cuisines.
  + Rapid API to act as a gateway for API calls.
  + Future database for user information (e.g., MongoDB, Firebase).
* Deployment and Hosting:
  + Heroku, Netlify, or Vencel for hosting and cloud services.
* Future Scope:
  + Integration with smart kitchen devices (e.g., voice assistants, smart ovens) via compatible APIs.
  + Use of cloud-based databases (e.g., MongoDB, Firebase) to store user data.

1.2 Availability of Technology

The technologies selected are mature and widely used in the industry:

* React.js is a powerful, well-documented, and flexible JavaScript library for building user interfaces, with a large community and ample resources available.
* Meals DB API and Rapid API are easily accessible and reliable platforms that allow integration with external data sources, offering recipes, ingredients, and instructions.
* Hosting platforms like Netlify and Vencel are cost-effective and well-suited for deploying modern web applications.
* State management solutions like Redux or Context API are tried and tested tools for managing complex state in large React applications.

Given these factors, the technical feasibility of the project is high.

2. Operational Feasibility

2.1 User Accessibility

Cookbook aims to provide a user-friendly interface that caters to both novice and experienced cooks. The app will be accessible via a browser on any device (desktop, tablet, mobile), ensuring that users can access the platform at their convenience. Features such as dynamic search, saved recipes, meal planning, and grocery list generation will cater to a variety of user needs.

2.2 Scalability

The platform is designed to scale as the user base grows. Features like meal planning, advanced search, and recipe management can be enhanced with additional functionalities as needed. The architecture will allow for easy future updates and scalability:

* Backend: The integration of Rapid API ensures scalability by connecting to external data sources. The system can handle increased user interactions and data fetch requests with the current architecture.
* Frontend: Reacts component-based architecture ensures that new features can be added incrementally without disrupting the overall experience.

2.3 User Adoption

The target audience for Cookbook includes home cooks, food enthusiasts, professional chefs, and anyone interested in exploring new recipes. The user-friendly design, personalized recipe suggestions, and meal planning features provide significant value to the users, encouraging adoption. In addition, the potential for social interactions and sharing within the community will foster user engagement.

2.4 Support and Maintenance

Cookbook will need ongoing support and maintenance to:

* Address user issues and bugs.
* Implement regular updates to keep the app up to date with new features and improvements.
* Ensure the app’s compatibility with evolving web standards and API updates.

This can be handled by a dedicated team of developers, and by leveraging cloud-based infrastructure like Heroku or Netlify, maintenance becomes more manageable.

3. Economic Feasibility

3.1 Cost Analysis

* Development Costs:
  + Frontend Development: A React.js developer will be required to build the user interface. Estimated cost: $5000–$7000.
  + Backend/API Integration: API calls to Meals DB API and Rapid API will be charged based on usage. The cost is minimal for initial use, but as the user base grows, the cost may scale.
  + Database and Hosting Costs: Cloud hosting on Netlify, Vencel, or Heroku comes at a low cost for small-scale operations. As the app scales, additional storage and premium plans may be required.
* Operational Costs:
  + API Usage: Meals DB API and Rapid API offer free tiers, but as traffic grows, you may need to subscribe to premium tiers for more requests.
  + Database: The initial use of a cloud database (like MongoDB or Firebase) would have low costs, but scaling up the database to accommodate a larger number of users will incur additional costs.

3.2 Revenue Generation (Optional)

To make Cookbook sustainable in the long term, potential monetization strategies include:

* Premium Memberships: Offering a subscription-based model for exclusive features like ad-free browsing, advanced search filters, and premium recipes.
* Affiliate Marketing: Partnering with food brands, cookware manufacturers, or grocery delivery services to promote products within the app. Users could click on affiliate links to buy ingredients or kitchen tools.
* Advertisements: Displaying non-intrusive ads within the app to generate additional revenue without disrupting the user experience.

3.3 Return on Investment (ROI)

Given the low initial development cost and the scalability of cloud-based technologies, the system can be developed on a relatively modest budget. Revenue generation through premium memberships or affiliate marketing could cover ongoing operational costs and generate a profit over time, making the system economically viable in the long run.

4. Legal Feasibility

4.1 Copyright and Licensing Issues

* Meals DB API and Rapid API will need to be used in accordance with their terms of service. It is essential to ensure that Cookbook complies with their usage policies, particularly around data access, API calls, and the use of recipe content.
* Recipe Content: Any recipes provided by third-party sources through the API should be attributed correctly to avoid copyright infringement. Users uploading their own recipes should agree to the platform’s terms of use regarding intellectual property.
* User Data Privacy: Cookbook will need to comply with data privacy regulations such as GDPR or CCPA, depending on the target user base. This includes collecting only necessary data, securing personal information, and providing users with the ability to manage their data.

4.2 Terms of Use and User Consent

* User Terms: A detailed Terms of Service (TOS) agreement should be provided to users that outline the rights and responsibilities of both the platform and the user.
* Data Privacy Policy: A transparent privacy policy will ensure users understand how their data will be used and protected.

4.SYSTEM ANALYSIS

**System Analysis for Cookbook: Your Virtual Kitchen Assistant**

System analysis is the process of studying and evaluating the current system or proposed system to identify the functional requirements and performance needs. In the case of **Cookbook: Your Virtual Kitchen Assistant**, the system analysis involves assessing the app's objectives, identifying user needs, defining functional requirements, and analysing system components to ensure they meet the overall goals of the project.

**1. Requirements Analysis**

**1.1 Functional Requirements**

The system must provide the following core functionalities to support the Cookbook experience:

* **User Account Management** (Optional for Future Updates)
  + Registration and login functionalities for users.
  + Option for guest access without creating an account.
  + User profile management, including preferences, saved recipes, and meal plans.
* **Recipe Discovery & Search**
  + Search recipes based on ingredients, cuisine, meal type, or dietary needs.
  + Ability to filter recipes by dietary restrictions (e.g., gluten-free, vegan).
  + Display recipe details including ingredients, instructions, and nutritional information.
* **Recipe Organization & Personalization**
  + Users can save, organize, and categorize favourite recipes (e.g., Quick Meals, Desserts, Healthy Options).
  + Personalized recipe suggestions based on user preferences and browsing history.
* **Meal Planning**
  + Users can create meal plans for the week or month.
  + The ability to assign specific recipes to days of the week and plan meals in advance.
* **Grocery List Generation**
  + Automatic generation of a grocery list based on the ingredients required for selected recipes.
  + Option to check off ingredients as users purchase them.
* **Recipe Sharing and Community Features**
  + Users can share their favourite recipes with other community members.
  + A comment section where users can rate and provide feedback on recipes.
  + A follow feature to keep up with favourite chefs or users.
* **Interactive UI with Visual Experience**
  + Visually engaging recipe browsing experience, with image galleries and categories for easy discovery.
  + Smooth user interface, with a focus on intuitive navigation for both novice and experienced cooks.

**1.2 Non-Functional Requirements**

In addition to the core functionalities, the system must meet the following non-functional requirements to ensure a positive user experience:

* **Performance:**
  + The system should load quickly and provide instant search results for recipes.
  + The application should support multiple concurrent users without performance degradation.
* **Security:**
  + User data (if stored) should be encrypted, ensuring privacy.
  + Secure authentication and authorization processes for user accounts.
* **Usability:**
  + The user interface should be easy to use, with clear navigation and minimal learning curve.
  + The application should be mobile-responsive, accessible from different devices (smartphones, tablets, desktops).
* **Scalability:**
  + The system should be able to scale as the number of users increases, ensuring high availability and performance.
  + Integration with third-party APIs (Meals DB, Rapid API) should be designed to handle a high volume of data requests.

**2. System Design**

The system will be developed using a modern tech stack, which will enable both flexibility and scalability. Below is an overview of the proposed system design:

**2.1 System Architecture**

* **Frontend Architecture:**
  + **React.js:** Used to build the dynamic, user-interactive front end of the application. React ensures the app is fast and responsive by utilizing its component-based architecture.
  + **Redux/Context API:** For managing global state, such as user preferences, saved recipes, and meal plans, Redux or Context API will be used.
  + **CSS3/Styled Components:** Styling the components to create a clean and responsive layout that works on multiple devices.
* **Backend Architecture:**
  + **Meals DB API** will be integrated into the system through **Rapid API** to fetch recipes, ingredients, and meal details.
  + **Node.js** (future implementation) might be used for server-side functionalities like user authentication and managing dynamic data.
  + **Cloud Storage:** User data, such as saved recipes and meal plans, will be stored in a cloud-based database such as **MongoDB** or **Firebase** for scalability.
* **External Integrations:**
  + **Third-Party APIs** like **Meals DB API** will be used to fetch and display recipe data.
  + Future integrations may involve smart kitchen devices (e.g., voice assistants, smart ovens) to enhance the cooking experience.

**2.2 Data Flow**

The data flow will be simple yet effective, allowing users to interact with the app seamlessly:

1. **User Requests Recipe Data:**
   * The user interacts with the frontend (React.js), where they input search queries (e.g., ingredients, cuisines, etc.).
   * React sends a request to the **Meals DB API** through **Rapid API** to fetch recipe data.
   * Recipe data (ingredients, instructions, nutritional details) is returned to the frontend and displayed.
2. **User Saves Recipes and Meal Plans:**
   * When a user saves a recipe or creates a meal plan, the app stores this data in the local state (using Redux or Context API).
   * For registered users, the app can save this data in the cloud database (**MongoDB** or **Firebase**).
3. **Grocery List Generation:**
   * Based on the recipes added to the meal plan, a grocery list is automatically generated, listing all required ingredients.
   * This list can be downloaded, shared, or printed for use during shopping.
4. **User Interaction and Community Sharing:**
   * Users can share recipes and leave ratings/comments. This interaction will be managed in the system's backend, either stored locally or within the cloud database.

**3. Use Case Analysis**

**4.1 Use Case 1: User Registration and Login**

* **Actors:** New User, Existing User
* **Precondition:** User opens the Cookbook application.
* **Scenario:**
  1. A new user can register using their email and create a password.
  2. An existing user can log in by entering their credentials.
  3. Once logged in, users can save recipes, create meal plans, and track their cooking history.

**4.2 Use Case 2: Recipe Search**

* **Actors:** User
* **Precondition:** User is logged in or browsing as a guest.
* **Scenario:**
  1. The user enters a keyword, ingredient, or meal type in the search bar.
  2. The app fetches matching recipes from the **Meals DB API**.
  3. The user can filter results by cuisine or dietary needs.

**4.3 Use Case 3: Saving a Recipe**

* **Actors:** Registered User
* **Precondition:** The User is logged in.
* **Scenario:**
  1. The user selects a recipe from the search results.
  2. The user clicks "Save" to add the recipe to their profile.
  3. The recipe is added to their saved recipe collection for future reference.

**4.4 Use Case 4: Meal Plan Creation**

* **Actors:** Registered User
* **Precondition:** User is logged in.
* **Scenario:**
  1. The user selects multiple recipes and adds them to their weekly meal plan.
  2. The app automatically generates a grocery list based on the meal plan.
  3. The user can view and edit the meal plan at any time.

5.SYSTEM REQUIREMENTS AND SPECIFICATION

System requirements and system specifications outline the necessary hardware, software, and network resources required to build, deploy, and run the Cookbook application efficiently. These requirements ensure that the system performs optimally, is scalable, secure, and meets all necessary technical and operational needs.

1. System Requirements

1.1 Hardware Requirements

* Development Environment:
  + Processor: Intel i5 or higher / AMD Ryen 5 or higher.
  + RAM: 8 GB or more.
  + Storage: At least 100 GB of available storage for development and resources.
  + Graphics Card: A standard integrated graphics card is sufficient (Intel HD Graphics or equivalent).
  + Internet Connection: Stable high-speed internet connection (for API interactions, downloading libraries, etc.).
* Production (Hosting/Deployment) Environment:
  + Server: Cloud-based server, preferably with scaling capabilities.
    - Recommended Providers: AWS, Google Cloud, Microsoft Azure, or any other cloud provider supporting Node.js apps.
  + RAM: 2 GB minimum for a basic instance. Increase as needed based on user load.
  + CPU: Minimum of 1 vCPU (Virtual CPU), scalable as per demand.
  + Storage: Cloud storage with scalability (for large data, media assets, and database).
  + Backup: Regular backup services (e.g., Amazon S3 for media files, database backup for MongoDB/Firebase).
* User Device:
  + Desktop/Laptop/Tablet: Any modern device capable of running a web browser with internet access.
  + Mobile Devices: Any modern smartphone capable of running a web browser.

1.2 Software Requirements

* Development Tools:
  + Operating System: Windows 10 or 11 / macOS / Linux (any modern OS supporting development).
  + Code Editor/IDE:
    - Visual Studio Code (Viscose)
    - Sublime Text
    - WebStorm (optional)
  + Version Control: Git and GitHub (or GitLab, Bitbucket) for version control and code collaboration.
  + Package Manager:
    - nm. (Node Package Manager) for managing JavaScript libraries and dependencies.
  + Web Browser: Google Chrome, Mozilla Firefox, Safari, or Microsoft Edge (for testing during development).
* Backend Tools:
  + Node.js: JavaScript runtime for server-side logic and handling API requests.
  + Express.js: Framework to help build API endpoints, handle routing, and manage backend functionalities.
  + Database:
    - MongoDB: A NoSQL database for storing user data and recipes.
    - Firebase (Optional): An alternative for real-time databases and easier authentication.
  + API Integration:
    - Meals DB API via Rapid API for recipe data (food, ingredients, etc.).
  + Authentication & Authorization:
    - JWT (JSON Web Tokens) or OAuth 2.0 for user authentication.
* Frontend Tools:
  + React.js: For building the user interface with reusable components and efficient rendering.
  + Redux/Context API: For state management (to handle global state such as saved recipes, user preferences, etc.).
  + CSS3/SCSS: For responsive styling and layout management.
  + Styled Components (Optional): For scoped CSS styling in React components.
* Deployment Tools:
  + Heroku/Netlify/Vencel: Cloud platforms for hosting the application.
  + Docker (Optional): To containerize the application for easier deployment and scaling.
  + CI/CD Tools: Jenkins, GitHub Actions, or Circles for continuous integration and continuous deployment.
* Other Services:
  + Cloud Storage: Amazon S3 or similar for storing media files (recipe images, profile pictures, etc.).
  + Email Service: Mail gun or SendGrid for sending notification emails to users (e.g., registration confirmation, password reset).

1.3 Network Requirements

* Internet Connection:
  + Stable internet connection for cloud services, API calls, and real-time data retrieval.
  + Fast data transfer rates to ensure the smooth and responsive behaviour of the application.
* API Calls:
  + The system will need access to the Meals DB API via Rapid API for fetching recipe data.
  + API rate limits and pricing models for Meals DB API and Rapid API should be considered, especially for premium access to more extensive data.

2. System Specifications

2.1 Functional Specifications

The functional specifications describe what the system should do in terms of services and features. Below are the core functionalities Cookbook should provide:

* User Management:
  + Registration and Login: New users can register, and existing users can log in with email/password or third-party authentication (Google/Facebook).
  + User Profile Management: Users can create and manage their profile, save favourite recipes, and view their meal plans.
  + Guest Mode: Users can browse and use the app without signing up (optional).
* Recipe Search & Discovery:
  + Search Recipes: Users can search for recipes based on ingredients, meal types, cuisines, and dietary needs.
  + Recipe Filters: Users can filter recipes based on different parameters like dietary restrictions (vegan, gluten-free, etc.).
  + Recipe Details: Each recipe displays detailed information, including ingredients, cooking instructions, and nutritional facts.
* Meal Planning:
  + Weekly Meal Plan Creation: Users can plan meals for the upcoming week by selecting and saving recipes.
  + Grocery List Generation: The app generates a shopping list based on selected recipes for the week.
* User Interaction:
  + Recipe Rating & Comments: Users can rate and leave feedback on recipes.
  + Sharing Recipes: Users can share recipes within the app or on social media platforms.
  + Save Recipes: Users can save their favourite recipes and categorize them for later access.
* Social Features (Optional):
  + Follow Other Users or Chefs: Users can follow other community members or chefs to stay updated with their latest recipes and posts.
  + Recipe Sharing: Users can upload their own recipes and share them with others.

2.2 Non-Functional Specifications

These specifications define the performance and reliability requirements for the system:

* Performance:
  + The application should have a response time of less than 2 seconds for API requests.
  + The system should handle at least 500 concurrent users with minimal lag or performance issues.
* Security:
  + Data Encryption: User passwords and sensitive data should be encrypted both during transmission (SSL/TLS) and at rest (AES encryption).
  + Authentication: Implement strong authentication methods (OAuth, JWT) to ensure user security.
  + Role-Based Access Control (RBAC): Different levels of access for users and administrators to control content and features.
* Scalability:
  + The system should scale easily to accommodate growing user data, meal plans, recipes, and activity.
  + Cloud hosting should allow automatic scaling based on user demand (e.g., AWS Auto Scaling).
* Usability:
  + The application should have an intuitive, clean, and easy-to-use interface.
  + It should be mobile-friendly, responsive, and accessible on different devices.
* Availability:
  + Uptime: The system should have a target uptime of 99.9% to ensure the application is consistently available to users.
  + Backup: Regular database backups should be taken to prevent data loss.

2.3 Database Specifications

The application will use a cloud-based NoSQL database to store user profiles, saved recipes, meal plans, and interaction data:

* MongoDB/Firebase: A flexible, scalable NoSQL database to store data. Data structure includes:
  + User Data: User profiles, login credentials, and preferences.
  + Recipe Data: Recipe details, ingredient lists, and instructions.
  + Meal Plans: User-generated meal plans.
  + Reviews and Ratings: User feedback on recipes.

The database should ensure quick data retrieval for real-time functionalities such as searching and browsing recipes.

6.SYSTEM DESIGN

The table design outlines the structure of the database to store and manage different entities of the CookBook application. These tables represent the core components such as users, recipes, saved recipes, and meal plans. The design follows a relational database approach (if using SQL like MySQL or PostgreSQL) or a NoSQL structure (if using MongoDB). Below is a relational database table design for simplicity and scalability.

**1. Users Table**

| **Column Name** | **Data Type** | **Description** | **Constraints** |
| --- | --- | --- | --- |
| user\_id | INT (PK) | Unique identifier for each user. | Primary Key, Auto-increment |
| username | VARCHAR(255) | Username chosen by the user. | Unique, Not NULL |
| email | VARCHAR(255) | Email address of the user. | Unique, Not NULL |
| password\_hash | VARCHAR(255) | Encrypted password for user authentication. | Not NULL |
| first\_name | VARCHAR(100) | First name of the user. |  |
| last\_name | VARCHAR(100) | Last name of the user. |  |
| profile\_picture | VARCHAR(255) | URL to the profile picture (if available). |  |
| account\_type | ENUM('guest', 'user', 'admin') | Type of user (admin, registered user, guest). | Default = 'user' |
| created\_at | TIMESTAMP | Date and time when the user account was created. | Default = CURRENT\_TIMESTAMP |
| updated\_at | TIMESTAMP | Date and time when the user account was last updated. |  |

**2. Recipes Table**

| **Column Name** | **Data Type** | **Description** | **Constraints** |
| --- | --- | --- | --- |
| recipe\_id | INT (PK) | Unique identifier for each recipe. | Primary Key, Auto-increment |
| title | VARCHAR(255) | Title of the recipe. | Not NULL |
| description | TEXT | Short description of the recipe. |  |
| category | VARCHAR(100) | Recipe category (e.g., breakfast, lunch, dessert). |  |
| cuisine | VARCHAR(100) | Cuisine type (e.g., Italian, Indian, American). |  |
| instructions | TEXT | Step-by-step instructions to prepare the recipe. |  |
| ingredients | TEXT | Ingredients required to make the recipe. |  |
| image\_url | VARCHAR(255) | URL to the image of the recipe. |  |
| prep\_time | INT | Preparation time in minutes. |  |
| cook\_time | INT | Cooking time in minutes. |  |
| servings | INT | Number of servings the recipe makes. |  |
| nutrition\_info | TEXT | Nutritional information of the recipe. |  |
| created\_at | TIMESTAMP | Date and time when the recipe was added. | Default = CURRENT\_TIMESTAMP |
| updated\_at | TIMESTAMP | Date and time when the recipe was last updated. |  |

**3. Saved Recipes Table**

| **Column Name** | **Data Type** | **Description** | **Constraints** |
| --- | --- | --- | --- |
| saved\_recipe\_id | INT (PK) | Unique identifier for saved recipes. | Primary Key, Auto-increment |
| user\_id | INT (FK) | Foreign Key linking to the Users table. | Foreign Key, Not NULL |
| recipe\_id | INT (FK) | Foreign Key linking to the Recipes table. | Foreign Key, Not NULL |
| saved\_at | TIMESTAMP | Date and time when the recipe was saved by the user. | Default = CURRENT\_TIMESTAMP |

**4. Meal Plans Table**

| **Column Name** | **Data Type** | **Description** | **Constraints** |
| --- | --- | --- | --- |
| meal\_plan\_id | INT (PK) | Unique identifier for each meal plan. | Primary Key, Auto-increment |
| user\_id | INT (FK) | Foreign Key linking to the Users table. | Foreign Key, Not NULL |
| plan\_name | VARCHAR(255) | Name of the meal plan (e.g., "Week 1", "Healthy Meals"). | Not NULL |
| start\_date | DATE | The start date of the meal plan. |  |
| end\_date | DATE | The end date of the meal plan. |  |
| created\_at | TIMESTAMP | Date and time when the meal plan was created. | Default = CURRENT\_TIMESTAMP |
| updated\_at | TIMESTAMP | Date and time when the meal plan was last updated. |  |

**5. Meal Plan Recipes Table (Associative Table for Meal Plan to Recipes)**

| **Column Name** | **Data Type** | **Description** | **Constraints** |
| --- | --- | --- | --- |
| meal\_plan\_recipe\_id | INT (PK) | Unique identifier for each meal plan recipe entry. | Primary Key, Auto-increment |
| meal\_plan\_id | INT (FK) | Foreign Key linking to the Meal Plans table. | Foreign Key, Not NULL |
| recipe\_id | INT (FK) | Foreign Key linking to the Recipes table. | Foreign Key, Not NULL |
| day\_of\_week | VARCHAR(10) | Day of the week this recipe is planned for (e.g., Monday, Tuesday). | Not NULL |

**6. Reviews Table**

| **Column Name** | **Data Type** | **Description** | **Constraints** |
| --- | --- | --- | --- |
| review\_id | INT (PK) | Unique identifier for each review. | Primary Key, Auto-increment |
| user\_id | INT (FK) | Foreign Key linking to the Users table. | Foreign Key, Not NULL |
| recipe\_id | INT (FK) | Foreign Key linking to the Recipes table. | Foreign Key, Not NULL |
| rating | INT | Rating given by the user (1 to 5 stars). |  |
| comment | TEXT | Review comment from the user. |  |
| created\_at | TIMESTAMP | Date and time when the review was submitted. | Default = CURRENT\_TIMESTAMP |

**7. Ingredients Table (If Ingredients Need Separate Management)**

| **Column Name** | **Data Type** | **Description** | **Constraints** |
| --- | --- | --- | --- |
| ingredient\_id | INT (PK) | Unique identifier for each ingredient. | Primary Key, Auto-increment |
| name | VARCHAR(255) | Name of the ingredient (e.g., "Tomato", "Sugar"). | Not NULL |
| category | VARCHAR(100) | Category of the ingredient (e.g., "Vegetable", "Spice"). |  |

**8. Recipe Ingredients Table (Associative Table for Recipe and Ingredients)**

| **Column Name** | **Data Type** | **Description** | **Constraints** |
| --- | --- | --- | --- |
| recipe\_ingredient\_id | INT (PK) | Unique identifier for each recipe ingredient entry. | Primary Key, Auto-increment |
| recipe\_id | INT (FK) | Foreign Key linking to the Recipes table. | Foreign Key, Not NULL |
| ingredient\_id | INT (FK) | Foreign Key linking to the Ingredients table. | Foreign Key, Not NULL |

6.2 FORM DESIGN

**Form Design for CookBook: Your Virtual Kitchen Assistant**

The form design for CookBook is integral to providing a user-friendly interface for interactions within the web application. These forms will allow users to add, edit, and interact with recipes, user profiles, and meal plans. Below is the **form design** for various key features in the CookBook application.

**1. User Registration Form**

**Purpose:** To allow new users to create an account in the CookBook system.

**Form Fields:**

* **First Name** (Text Field):  
  *Label:* First Name  
  *Placeholder:* Enter your first name  
  *Required:* Yes
* **Last Name** (Text Field):  
  *Label:* Last Name  
  *Placeholder:* Enter your last name  
  *Required:* Yes
* **Username** (Text Field):  
  *Label:* Username  
  *Placeholder:* Choose a username  
  *Required:* Yes
* **Email** (Email Field):  
  *Label:* Email Address  
  *Placeholder:* Enter your email address  
  *Required:* Yes
* **Password** (Password Field):  
  *Label:* Password  
  *Placeholder:* Create a password  
  *Required:* Yes
* **Confirm Password** (Password Field):  
  *Label:* Confirm Password  
  *Placeholder:* Re-enter your password  
  *Required:* Yes
* **Profile Picture** (File Upload):  
  *Label:* Profile Picture  
  *Required:* No

**Buttons:**

* **Submit** (Button):  
  *Text:* Register  
  *Action:* Submits the form for registration
* **Cancel** (Button):  
  *Text:* Cancel  
  *Action:* Cancels the registration process

**2. User Login Form**

**Purpose:** To allow registered users to log into the CookBook system.

**Form Fields:**

* **Username/Email** (Text Field):  
  *Label:* Username or Email  
  *Placeholder:* Enter your username or email  
  *Required:* Yes
* **Password** (Password Field):  
  *Label:* Password  
  *Placeholder:* Enter your password  
  *Required:* Yes

**Buttons:**

* **Login** (Button):  
  *Text:* Log In  
  *Action:* Logs the user into the system
* **Forgot Password?** (Link/Button):  
  *Text:* Forgot Password  
  *Action:* Redirects the user to the password reset page

**3. Add Recipe Form**

**Purpose:** To allow users to add a new recipe to the CookBook database.

**Form Fields:**

* **Recipe Title** (Text Field):  
  *Label:* Recipe Title  
  *Placeholder:* Enter the title of the recipe  
  *Required:* Yes
* **Category** (Dropdown List):  
  *Label:* Recipe Category  
  *Options:* Breakfast, Lunch, Dinner, Dessert, Snacks, etc.  
  *Required:* Yes
* **Cuisine** (Text Field):  
  *Label:* Cuisine Type  
  *Placeholder:* Enter the cuisine type (e.g., Italian, Mexican, etc.)  
  *Required:* Yes
* **Ingredients** (Text Area):  
  *Label:* Ingredients  
  *Placeholder:* List the ingredients with quantities (e.g., 1 cup sugar, 2 eggs)  
  *Required:* Yes
* **Instructions** (Text Area):  
  *Label:* Cooking Instructions  
  *Placeholder:* Enter the step-by-step cooking instructions  
  *Required:* Yes
* **Preparation Time** (Number Field):  
  *Label:* Prep Time (in minutes)  
  *Placeholder:* Enter preparation time  
  *Required:* Yes
* **Cooking Time** (Number Field):  
  *Label:* Cooking Time (in minutes)  
  *Placeholder:* Enter cooking time  
  *Required:* Yes
* **Serving Size** (Number Field):  
  *Label:* Number of Servings  
  *Placeholder:* Enter the number of servings  
  *Required:* Yes
* **Recipe Image** (File Upload):  
  *Label:* Upload Recipe Image  
  *Required:* No

**Buttons:**

* **Submit** (Button):  
  *Text:* Add Recipe  
  *Action:* Submits the recipe to the database
* **Cancel** (Button):  
  *Text:* Cancel  
  *Action:* Cancels adding the recipe

**4. Edit Recipe Form**

**Purpose:** To allow users to edit an existing recipe.

**Form Fields:**

The fields in this form would be similar to the "Add Recipe Form" with pre-filled values pulled from the existing recipe's data.

* **Recipe Title** (Text Field):  
  *Label:* Recipe Title  
  *Required:* Yes
* **Category** (Dropdown List):  
  *Label:* Recipe Category  
  *Required:* Yes
* **Cuisine** (Text Field):  
  *Label:* Cuisine Type  
  *Required:* Yes
* **Ingredients** (Text Area):  
  *Label:* Ingredients  
  *Required:* Yes
* **Instructions** (Text Area):  
  *Label:* Cooking Instructions  
  *Required:* Yes
* **Preparation Time** (Number Field):  
  *Label:* Prep Time (in minutes)  
  *Required:* Yes
* **Cooking Time** (Number Field):  
  *Label:* Cooking Time (in minutes)  
  *Required:* Yes
* **Serving Size** (Number Field):  
  *Label:* Number of Servings  
  *Required:* Yes
* **Recipe Image** (File Upload):  
  *Label:* Upload New Image (optional)  
  *Required:* No

**Buttons:**

* **Submit** (Button):  
  *Text:* Update Recipe  
  *Action:* Updates the recipe with the new information
* **Cancel** (Button):  
  *Text:* Cancel  
  *Action:* Cancels the edit process

**5. Search Recipe Form**

**Purpose:** To allow users to search for recipes based on their preferences.

**Form Fields:**

* **Search Bar** (Text Field):  
  *Label:* Search Recipes  
  *Placeholder:* Search by recipe name, ingredients, or cuisine  
  *Required:* No
* **Category Filter** (Dropdown List):  
  *Label:* Filter by Category  
  *Options:* Breakfast, Lunch, Dinner, Dessert, Snacks, etc.  
  *Required:* No
* **Cuisine Filter** (Text Field):  
  *Label:* Filter by Cuisine  
  *Placeholder:* Enter cuisine type (e.g., Italian, Mexican, etc.)  
  *Required:* No
* **Dietary Preferences** (Checkboxes):  
  *Label:* Dietary Preferences  
  *Options:* Vegetarian, Vegan, Gluten-Free, Dairy-Free  
  *Required:* No

**Buttons:**

* **Search** (Button):  
  *Text:* Search Recipes  
  *Action:* Executes the search based on the entered criteria

**6. User Profile Update Form**

**Purpose:** To allow users to update their profile information (e.g., change password, update profile picture).

**Form Fields:**

* **First Name** (Text Field):  
  *Label:* First Name  
  *Required:* Yes
* **Last Name** (Text Field):  
  *Label:* Last Name  
  *Required:* Yes
* **Email** (Email Field):  
  *Label:* Email Address  
  *Required:* Yes
* **Profile Picture** (File Upload):  
  *Label:* Update Profile Picture  
  *Required:* No
* **New Password** (Password Field):  
  *Label:* New Password (optional)  
  *Required:* No
* **Confirm New Password** (Password Field):  
  *Label:* Confirm New Password (optional)  
  *Required:* No

**Buttons:**

* **Save Changes** (Button):  
  *Text:* Save Changes  
  *Action:* Saves the updated profile information
* **Cancel** (Button):  
  *Text:* Cancel  
  *Action:* Cancels the update process

**7. Meal Plan Form**

**Purpose:** To allow users to create a meal plan for a specified period.

**Form Fields:**

* **Meal Plan Name** (Text Field):  
  *Label:* Meal Plan Name  
  *Placeholder:* Enter a name for your meal plan (e.g., "Week 1").  
  *Required:* Yes
* **Start Date** (Date Picker):  
  *Label:* Start Date  
  *Required:* Yes
* **End Date** (Date Picker):  
  *Label:* End Date  
  *Required:* Yes
* **Add Recipes to Plan** (Searchable Dropdown List or Multi-Select):  
  *Label:* Add Recipes  
  *Required:* Yes

**Buttons:**

* **Create Meal Plan** (Button):  
  *Text:* Create Meal Plan  
  *Action:* Creates the meal plan and links the selected recipes to it
* **Cancel** (Button):  
  *Text:* Cancel  
  *Action:* Cancels the meal plan creation