Design Document

for

Recipe Realm

(Recipe Management System)

Table Of Contents

Introduction

Recipe Realm enables users to create, edit, view, and share recipes. The backend uses Express.js for the server framework and MongoDB as the database. This document outlines the architecture, components, data models, and relationships necessary for implementation.

Overview

Recipe Realm is built using the MERN stack (MongoDB, Express.js, React, Node.js). The application is structured to provide scalability, maintainability, and a responsive user experience across devices.

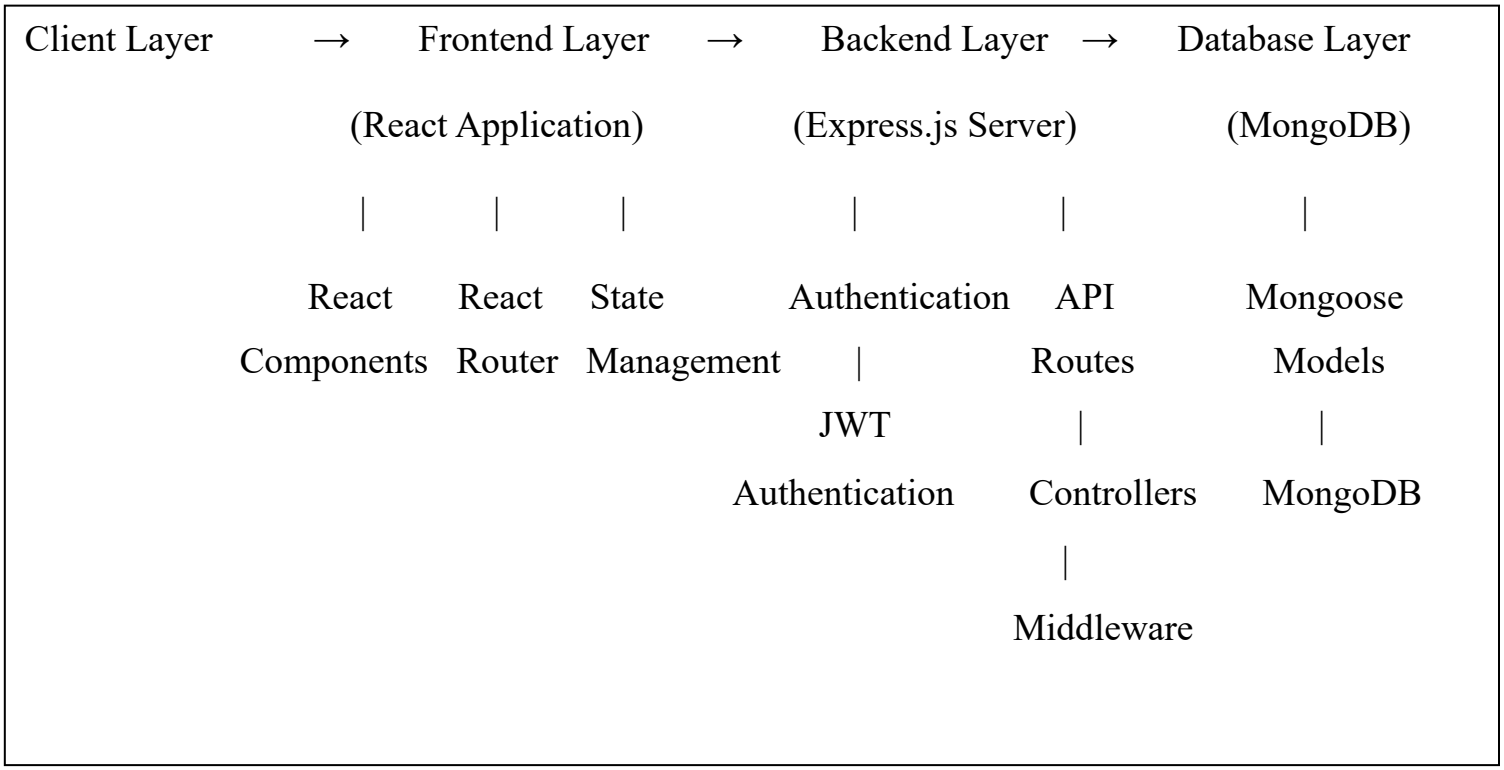
Objectives

1. Provide functionality for users to add, edit, delete, and view recipes.
2. Allow users to upload and manage images for recipes.
3. Support user authentication and profile management.
4. Facilitate tagging and categorization of recipes.
5. Ensure scalability and maintainability of the system.

Architechture

The system follows a three-tier architechture:

1. Frontend: Developed with React.js, responsible for the client side user interface and interactions.
2. Backend API: Built with Express.js and Node.js. handling server side logic, API endpoints, authorization etc.
3. Database: Uses MongoDB for storing user data, recipes and related information.



Backend Design

Technologies Used

- **Node.js**: JavaScript runtime environment.
- **Express.js**: Web application framework for building APIs.
- **MongoDB**: NoSQL database for data storage.
- **Mongoose**: ODM (Object Data Modeling) library for MongoDB.
- **JWT**: JSON Web Tokens for authentication.
- **bcrypt**: Library for hashing passwords.

Project Layout

- **index.js**: entry point of the server application
- **models/**: Schemas for users and recipes models.
- **routes.js**: defines endpoints for various pages

API Design

- Authentication
- Users (/api/users):
    1. POST/: create a user
    2. GET/id: get user details
    3. PUT/id: update user details
- Recipes(/api/recipes):

1. POST/: add new recipes
2. GET/id: get recipe details
3. PUT/id: update recipe details
4. GET/tags: get recipes under a particular tag
5. DELETE/id: delete a recipe

Database Schema

User Model

Fields:

- name(String, required): user's full name
- email(String, required, unique): email address
- phoneNumber(String, required): contact number
- password(String, required): encrypted password

Fields	**Object_Id**	
Name	String	
Email	String	Foreign key
Phone Number	String	
Password	String	
userId(CreatedAt)	Date.now()	Primary key

Recipes Model

Fields:

- recipeName(String, required): name of the recipe
- coverImage(Image, required): front image of the recipe
- recipeImage(Image, required): image in the recipeInnerPage
- prepTime(String, required): time taken to prepare the ingredients
- cookTime(String, required): time taken to cook the dish
- servings(Number, required): numbers of persons the dish can be served to
- description(String, required): a short description of the recipe
- ingredients(String, required): an array of required ingredients
- method(String, required): an array of number of steps to make the recipe
- tags(String, required): different tags the recipe belongs to.

Fields	**Object_Id**	
recipeId	Date.now()	Primary Key
user	User_email	Foreign Key
recipeName	String	
coverImage	Image	
recipeImage	Image	
prepTime	String	
cookTime	String	
Servings	Number	

Description	String	
Ingredients	String	
Method	String	
Tags	String	

Data Relationships

- **Users and Recipes**: One-to-Many relationship (a user can have many recipes).
- **Recipes and Tags**: Many-to-Many relationship (a recipe can have many tags, and a tag can belong to multiple recipes).
- MongoDB stores references (e.g., ObjectId) to establish these relationships efficiently.

Frontend Design

Technologies Used

- **React.js**: JavaScript library
- **CSS**: styling
- **React-Router-Dom**: for routing

Project Layout

- **index.jsx**: entry point of the react application
- **routes**.**jsx**: client side routing
- **app**.**jsx:** main application component
- **components/**: resuable components
- **public**/: images used for recipes

Routing

- /: home page
- /addRecipe: page containg form to add a new recipe.
- /profilePage: profile page of logged in user
- /recipes: overview of recipes under different tags.
- /recipes/tags/tag_name: page containing recipes of a particular tag
- /recipes/recipeId: inner recipe page

Security

Authentication

- JWT Tokens:
    - Tokens are securely created and signed using a secret key.
    - Stored safely in the client's localStorage.
- Password Handling:
    - Passwords are encrypted with bcrypt before saving in the database.
    - Plaintext passwords are never saved or logged.

Authorization

- Protected Routes:
    - Backend endpoints are accessible only with valid JWT tokens.
    - Frontend routes implement higher-order components to restrict unauthorized access.
- **Input Validation**:
    - Validation is enforced both on the client and server side using Regex.

CORS Configuration

- Access Control Policies:
    - Configured to accept requests only from trusted domains.
    - Proper headers are set for Access-Control-Allow-Origin, allowed methods, and headers.

Future Scope

1. Advanced User Features

- Social Interaction:
    - Add features like recipe commenting, reviews, and ratings to foster a sense of community.
    - Enable users to follow each other and view a personalized feed of recipes from their favorite chefs or users.
- Recipe Sharing:
    - Integration with social media platforms for seamless sharing of recipes.
    - Option to generate printable recipe cards or share via email.

2. Localization and Global Reach

- Language Support:
    - Translate the platform into multiple languages to cater to international users.
- Regional Recipes:
    - Highlight traditional and regional recipes from around the world.
- Dietary Preferences:
    - Cater to specific dietary needs, such as vegan, keto, paleo, or culturally specific dietary restrictions.

3. Enhanced Image and Video Support

- Media Features:
    - Allow users to upload and embed cooking videos.

    - Introduce step-by-step recipe video tutorials.
- AI-based Food Recognition:
    - Let users upload images of dishes and suggest recipes based on the image.

Conclusion

The **Recipe Realm** project is a full-stack web application designed to make recipe management simple and engaging. Built with **React.js** on the frontend, **Express.js** and **Node.js** on the backend, and **MongoDB** as the database, the platform provides a seamless and efficient user experience.

Throughout development, I focused on creating a clear and modular architecture, with well-organized components, APIs, and database models. Emphasis was placed on scalability, security, and intuitive design to ensure the platform can grow while maintaining a user-friendly experience.

Working on Recipe Realm has been a valuable journey, teaching me the intricacies of full-stack development and the significance of creating solutions with the user in mind. It has been both an educational and enjoyable project.