Project Documentation

SMART RECIPE RECOMMENDER

Project Overview

The Smart Recipe Recommender is a full-stack web application developed using the MERN stack (MongoDB, Express.js, React.js, Node.js). The goal of this project is to assist users in discovering recipes they can cook based on the ingredients they already have at home. It simplifies daily meal planning by offering personalized suggestions and also allows users to manage their kitchen inventory and save their own recipes. The project emphasizes user convenience, efficient data management, and intelligent matching logic to deliver meaningful recommendations.

This system can be especially useful for people who want to reduce food waste, manage their groceries better, or quickly decide on a meal without needing to shop for new ingredients. With a focus on a responsive and intuitive interface, along with secure user authentication, the application aims to be both practical and user-friendly.

Technology Stack

✓ **Backend:**

- Built with **Node.js** and **Express.js** for server-side routing and API logic.
- **MongoDB** used as the database to store user profiles, inventory, and recipes.
- **Mongoose** serves as the ODM (Object Data Modeling) tool to interact with MongoDB.
- **JWT (JSON Web Tokens)** implemented for secure user authentication and session management.

✓ **Frontend:**

- Developed using **React.js** to build dynamic, reusable user interfaces.
- Utilizes React hooks like **useState**, **useEffect**, and **useReducer** for efficient state management.
- **Context API** combined with **useReducer** maintains the global state, including user sessions and inventory data.

✓ **UI/UX Design:**

- **Tailwind CSS** used for responsive, mobile-friendly, and modern styling with utility-first classes.

Project Architecture

- Frontend handles UI and API integration.
- Backend provides endpoints for authentication, recipes, and inventory logic.
- Database stores structured data related to users, recipes, and preferences.

Flow chart

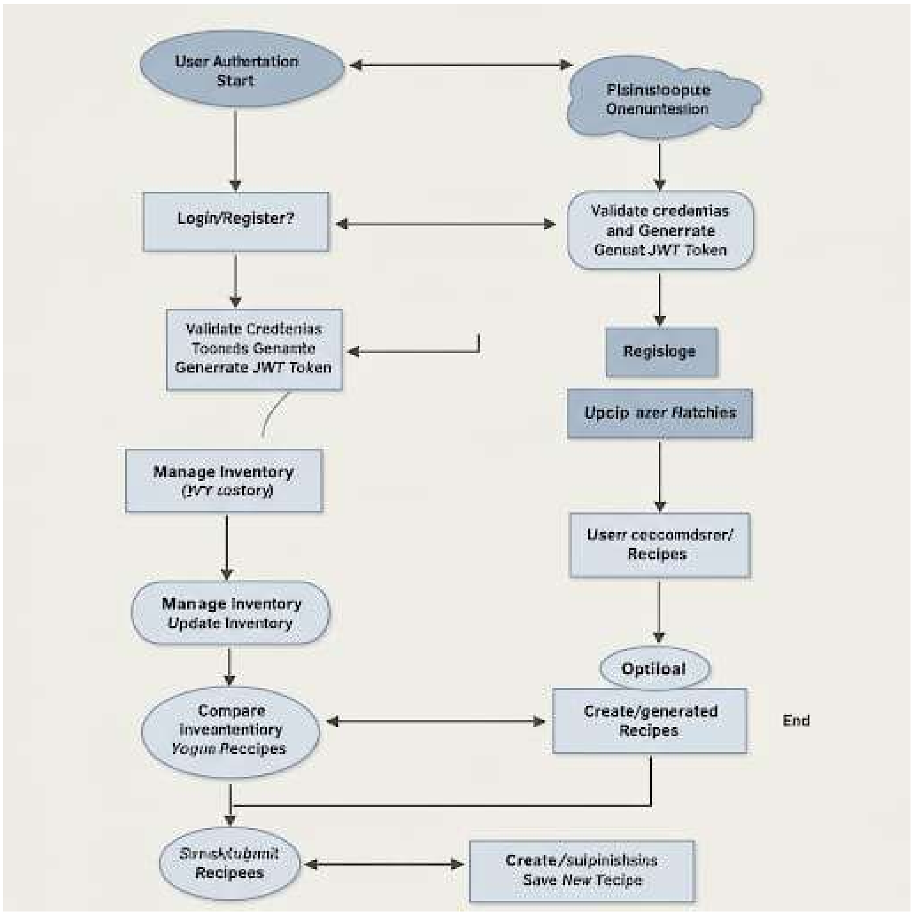

User Authertation Start
Plsinistoopue Onenuntesiion
Login/Register?
Validate credentias and Generrate Genuat JWT Token
Validate Credtenias Tooneds Genamte Generrate JWT Token
Regisloge
Upcip azer Ratchies
Manage Inventory (JVT costory)
Userr ceccomdsrer/ Recipes
Manage Inventory Update Inventory
Optiloal
Compare inveantentiory Yogun Reccipes
Create/generated Recipes
End
Snrisk/ubmit Recipees
Create/sulpinishsins Save New Tecipe

Fig no 1: Design

System Features

- **User Registration and Login:** Securely stores user credentials with password hashing. Manages user sessions using JWT (JSON Web Tokens).
- **User Dashboard:** Allows users to manage their inventory of ingredients. Users can add, update, or delete items from their pantry. Inventory is stored in the database and linked to the user’s profile.
- **Recipe Recommendation Engine:** Fetches recipes from the database. Filters recipes based on the user’s current inventory. Suggests recipes when there is a high match rate (e.g., 80% of ingredients available). displays missing ingredients, if any, to help users complete recipes.
- **Manual Recipe Submission:** Users can add their own recipes by entering the title, ingredients, preparation steps, and an image.User-submitted recipes are stored securely and linked to their profile for future .

File Structure

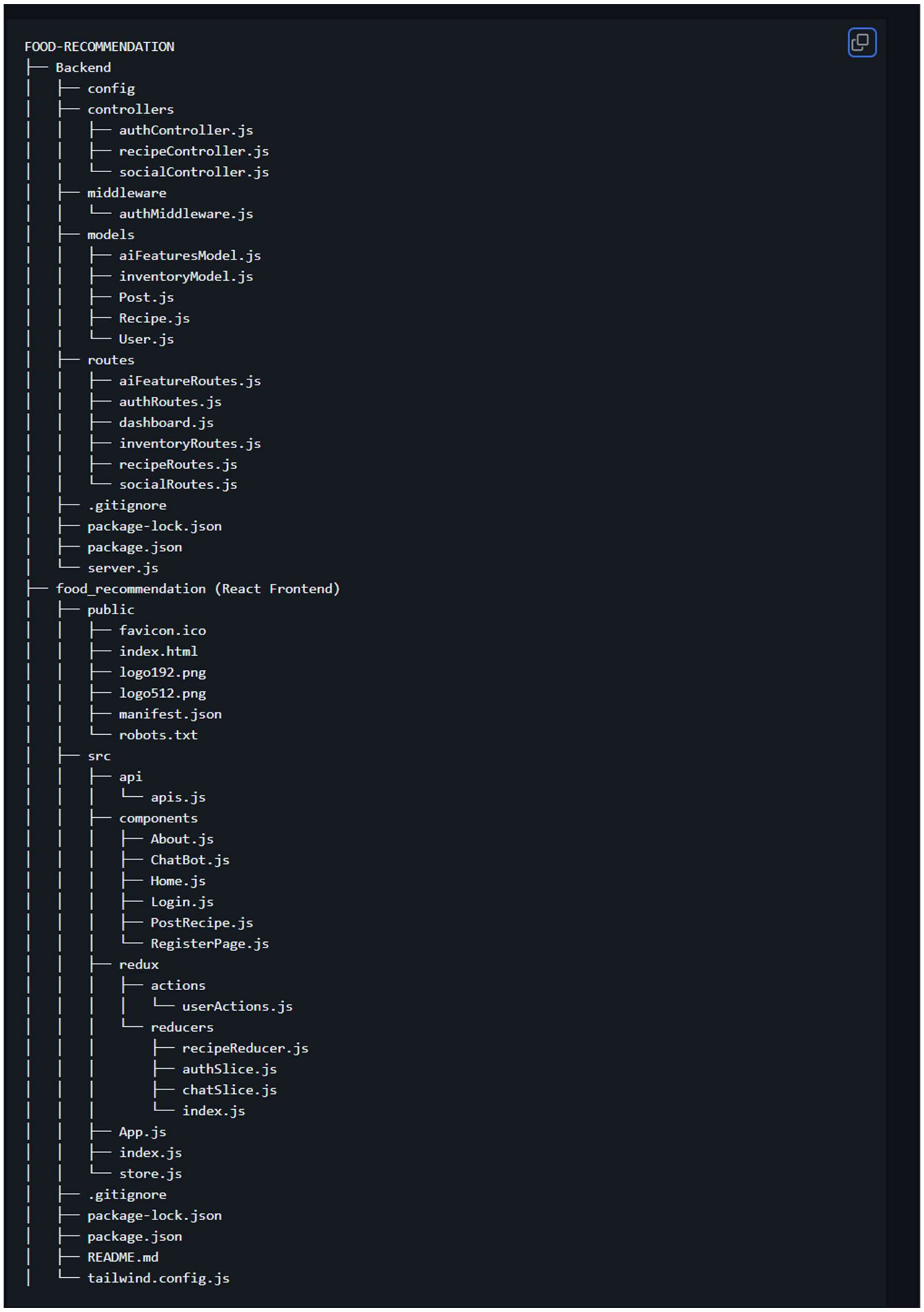

```
FOOD-RECOMMENDATION
├── Backend
│   ├── config
│   ├── controllers
│   │   ├── authController.js
│   │   ├── recipeController.js
│   │   └── socialController.js
│   ├── middleware
│   │   └── authMiddleware.js
│   ├── models
│   │   ├── aiFeaturesModel.js
│   │   ├── inventoryModel.js
│   │   ├── Post.js
│   │   ├── Recipe.js
│   │   └── User.js
│   ├── routes
│   │   ├── aiFeatureRoutes.js
│   │   ├── authRoutes.js
│   │   ├── dashboard.js
│   │   ├── inventoryRoutes.js
│   │   ├── recipeRoutes.js
│   │   └── socialRoutes.js
│   ├── .gitignore
│   ├── package-lock.json
│   ├── package.json
│   └── server.js
├── food_recommendation (React Frontend)
│   ├── public
│   │   ├── favicon.ico
│   │   ├── index.html
│   │   ├── logo192.png
│   │   ├── logo512.png
│   │   ├── manifest.json
│   │   └── robots.txt
│   ├── src
│   │   ├── api
│   │   │   └── apis.js
│   │   ├── components
│   │   │   ├── About.js
│   │   │   ├── ChatBot.js
│   │   │   ├── Home.js
│   │   │   ├── Login.js
│   │   │   ├── PostRecipe.js
│   │   │   └── RegisterPage.js
│   │   ├── redux
│   │   │   ├── actions
│   │   │   │   └── userActions.js
│   │   │   └── reducers
│   │   │       ├── recipeReducer.js
│   │   │       ├── authSlice.js
│   │   │       ├── chatSlice.js
│   │   │       └── index.js
│   │   ├── App.js
│   │   ├── index.js
│   │   └── store.js
│   ├── .gitignore
│   ├── package-lock.json
│   ├── package.json
│   ├── README.md
│   └── tailwind.config.js
```

Fig no 2: Folder Structure

Architecture Overview

Client → API Gateway → Backend Services → MongoDB

- Frontend communicates with the backend using REST APIs.
- AI functionalities are either integrated in the backend or accessed via third-party APIs (e.g., for NLP or image recognition).
- MongoDB stores structured and semi-structured data such as recipes, user preferences, and pantry items.

Backend Implementation

The backend is built using Node.js and Express.js, serving as the core logic and API layer of the application. It is responsible for handling all user requests, processing business logic, and interacting with the database.

Key Technologies and Tools:

- **Node.js:** Server runtime environment for building scalable network applications.
- **Express.js:** Lightweight web framework to manage server routes and middleware.
- **JWT (jsonwebtoken):** Secure user authentication and authorization.
- **Mongoose:** Object Data Modeling (ODM) library for MongoDB, providing schema-based solutions.
- **AWS (Amazon Web Services):**

  EC2: Scalable cloud-based virtual servers used for deploying the backend application.

  Nginx: High-performance web server and reverse proxy for load balancing and serving static assets.

API Structure:

- **/api/auth:** Handles user registration and login, generating JWT tokens for session management.
- **/api/recipes:** Fetches recommended recipes and allows users to add custom ones.
- **/api/chatbot:** Integrates with the Spoonacular API to provide additional recipe suggestions and food-related chatbot interactions**.**

Authentication:

JWT tokens are issued during login and used to protect private routes. Middleware is implemented to verify the token and ensure secure access to user data.

Error Handling & Validation:

The backend includes centralized error handling, input validation using custom middleware, and secure error messages to avoid leaking sensitive information.

Frontend Implementation

The frontend of the application is built using React.js to create a dynamic and responsive user interface. The main objective is to ensure ease of use, efficient state management, and an aesthetically pleasing experience.

Key Technologies and Tools:

- **React.js:** Functional components and hooks for building UI
- **Context API + useReducer:** Global state management for auth, inventory, and recipes
- **Tailwind CSS:** Utility-first CSS framework for styling and responsiveness
- **React Router DOM:** Handles navigation across different pages

Major Components:

- **Auth Pages**: Login and Register pages with form validation and token handling
- **Dashboard:** Displays inventory, recommended recipes, and quick access to actions
- **Inventory Manager:** Lets users add, update, or delete pantry items
- **Recipe Viewer:** Shows suggested recipes and allows adding custom ones

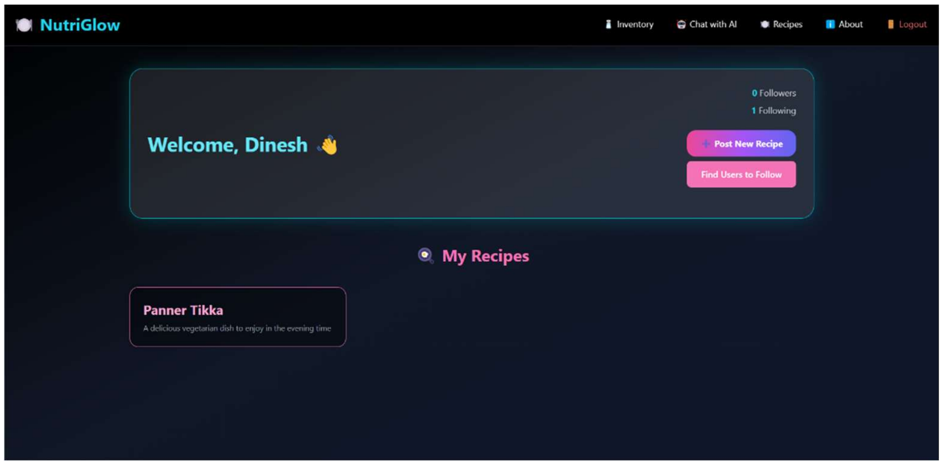
Responsiveness & Accessibility:

The UI is designed to be fully responsive across devices using Tailwind's grid and flex utilities. Components are semantic and accessible for screen readers and keyboard navigation.

Database Design and Management

The application uses MongoDB for storing user data, inventory items, and recipes. Data is organized into three main collections: Users, Inventory, and Recipes, with each user's data isolated for security and personalization. Mongoose is used to define schemas and manage relationships between data. Recipes can be system-defined or user-submitted, and the inventory is linked to each user to support personalized recommendations. The database structure is simple, scalable, and optimized for quick data retrieval and secure access.

Deployment

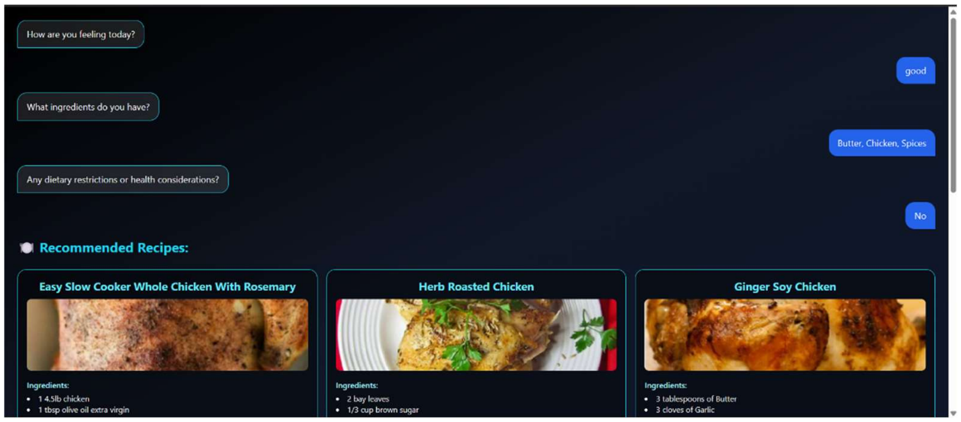


Welcome, Dinesh 👋

Panner Tikka

A delicious vegetarian dish to enjoy in the evening time

Fig no 3: Dashboard


How are you feeling today?

good

What ingredients do you have?

Butter, Chicken, Spices

Any dietary restrictions or health considerations?

No

Recommended Recipes:

Easy Slow Cooker Whole Chicken With Rosemary

Ingredients:
- 1 4.5lb chicken
- 1 tbsp olive oil extra virgin

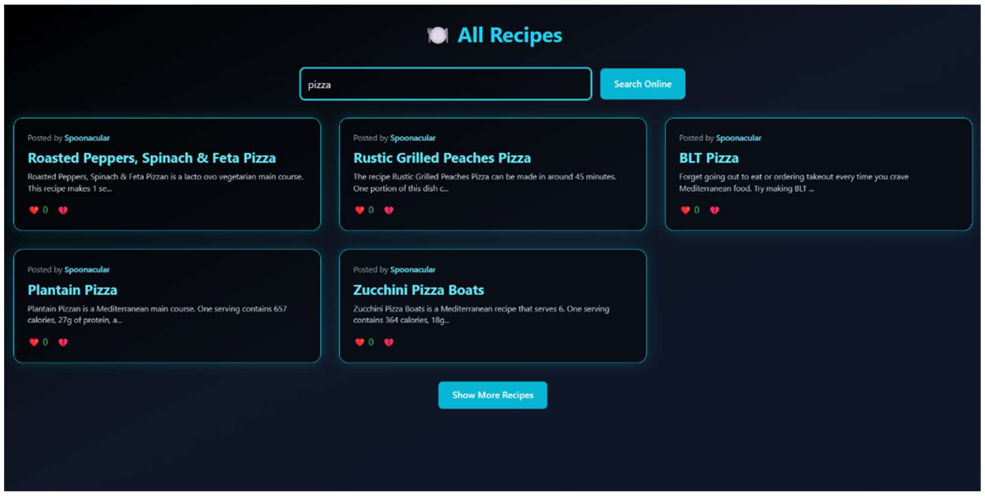
Herb Roasted Chicken

Ingredients:
- 2 bay leaves
- 1/3 cup brown sugar

Ginger Soy Chicken

Ingredients:
- 3 tablespoons of Butter
- 3 cloves of Garlic

Fig no 4: Chatbot


All Recipes

pizza

Posted by Spoonacular

Roasted Peppers, Spinach & Feta Pizza

Roasted Peppers, Spinach & Feta Pizzan is a lacto ovo vegetarian main course. This recipe makes 1 se...

0

Posted by Spoonacular

Rustic Grilled Peaches Pizza

The recipe Rustic Grilled Peaches Pizza can be made in around 45 minutes. One portion of this dish c...

0

Posted by Spoonacular

BLT Pizza

Forget going out to eat or ordering takeout every time you crave Mediterranean food. Try making BLT ...

0

Posted by Spoonacular

Plantain Pizza

Plantain Pizzan is a Mediterranean main course. One serving contains 657 calories, 27g of protein, a...

0

Posted by Spoonacular

Zucchini Pizza Boats

Zucchini Pizza Boats is a Mediterranean recipe that serves 6. One serving contains 364 calories, 18g...

0

Fig no 5: Recipe page

The Smart Recipe Recommender was deployed on AWS Cloud to ensure accessibility, reliability, and performance. The application was hosted on a virtual server, with both the frontend and backend running together in a live environment. The database was managed using a secure cloud-hosted service, allowing seamless data access. Basic server setup, security configurations, and performance monitoring were done to keep the application stable and available to users.

Testing

- Unit and integration tests written for core API endpoints.
- Manual UI testing conducted for user flows.
- Test coverage reports available upon request.

Conclusion

- The Smart Recipe Recommender project addresses real-world challenges around meal planning, health, and food sustainability using modern web development and AI techniques. It is designed to scale with user needs and adapt to various dietary lifestyles, creating a personalized, efficient, and engaging experience.