|  |  |
| --- | --- |
| Sowmya R | Frontend Developer |
| Shushmeeta E | Backend Developer |
| Sowmiya K | Database Specialist |
| Bandaru sasidhar | QA Tester |

FOOD ORDERING APP WITH MERN STACK.

Team Members:

Overview:

Purpose:

The Food Ordering App is designed to allow users to browse menus, place orders, and track them in real time. Restaurants can manage their menus and orders through an admin interface.

Features:

User registration and login (authentication).

Browse and search for restaurants and menu items.

Add items to the cart and place an order.

Real-time order tracking.

Admin panel for restaurant owners to manage menus

and orders.

Architecture:

1. Frontend:

Built with React and styled using CSS frameworks.

State management: Context API/Redux to handle cart.

Routing: React Router for navigation.

2.Backend:

Node.js with Express.js for handling API requests.

Middleware for authentication, validation, and error handling.

3.Database:

MongoDB for storing user data, restaurant menus, orders, and

reviews.

Example Collections:

Users: Stores user profiles, hashed passwords, and roles.

Restaurants: Details like name, location, and menu items.

4. Setup Instructions

Prerequisites:

Node.js (version X.X.X)

MongoDB (version X.X.X)

Any other dependencies like npm/yarn

Installation:

1. Clone the repository:

git clone [https://github.com/Sowmiya-icon/Food-Ordering-App.git]

2. Navigate to the project folder.

3.Install dependencies for both frontend and backend:

cd client && npm install

cd ../server && npm install

4. Set up .env files for frontend and backend. Example:

Frontend: API\_BASE\_URL

Backend: MONGO\_URI , JWT\_SECRET

5.Folder Structure

Client (React Frontend):

/src/components: Reusable UI components (e.g., Navbar, Footer, Cart).

/src/pages: Pages like Home, Menu, Checkout, Order Status.

/src/context: Context for managing cart and authentication state.

Server (Node.js Backend):

/routes: API route definitions (e.g., userRoutes, orderRoutes).

/controllers: Logic for handling API requests.

/models: MongoDB schemas (e.g., User, Restaurant, Order).

/middleware: Authentication and error handling middleware.

6. Running the Application

Frontend:

Navigate to the client directory and start the development server:

cd client

npm start

Backend:

Navigate to the server directory and start the Node.js server:

cd server

npm start

7. API Documentation

Document core API endpoints:

Login:

Endpoint: /api/users/login

Method: POST

Parameters:

Email : ksowmiya267@gmail.com

Password : xxxxxxxxxx

Response:

{

"token": "jwt-token",

"user": { "name": "John Doe", "email": "john@example.com" }

}

Place Order:

Endpoint: /api/orders

Method: POST

Request Body:

{

"restaurant\_Id": "12345",

"items": [

{ "menu\_Item\_Id": "67890", "quantity": 2 }

]

}

Response:

{

"order\_Id": "98765",

"status": "Pending"

}

8. Authentication

JWT-based Authentication:

Access tokens issued upon login

Protected routes for user profiles and order history.

Authorization:

Admin roles to manage restaurants and orders.

9. User Interface

Include screenshots or GIFs for:

Home page with restaurant listings.

Menu page showing items.

Checkout process.

Admin dashboard.

10. Testing

Testing Strategy:

Unit testing with Jest for frontend components.

API testing with Postman.

End-to-end testing with Cypress.

Command to Run Tests:

npm test

11. Screenshots or Demo

Provide screenshots or a live demo link (e.g., via Heroku, Netlify, or Vercel).

12. Known Issues

Example:

Payment gateway integration is incomplete.

Real-time order tracking occasionally delays updates.

13. Future Enhancements

Add features like user reviews and ratings.

Implement AI-based food recommendations.

Improve performance and scalability for high user.

This structure provides a clear and organized format tailored for a food ordering app. Let me know if you’d like further refinements!