Full Stack Development with MERN

API Development and Integration Report

Date	11th July 2024
Team ID	SWTID1720076571
Project Name	Project - Food Ordering System
Maximum Marks	

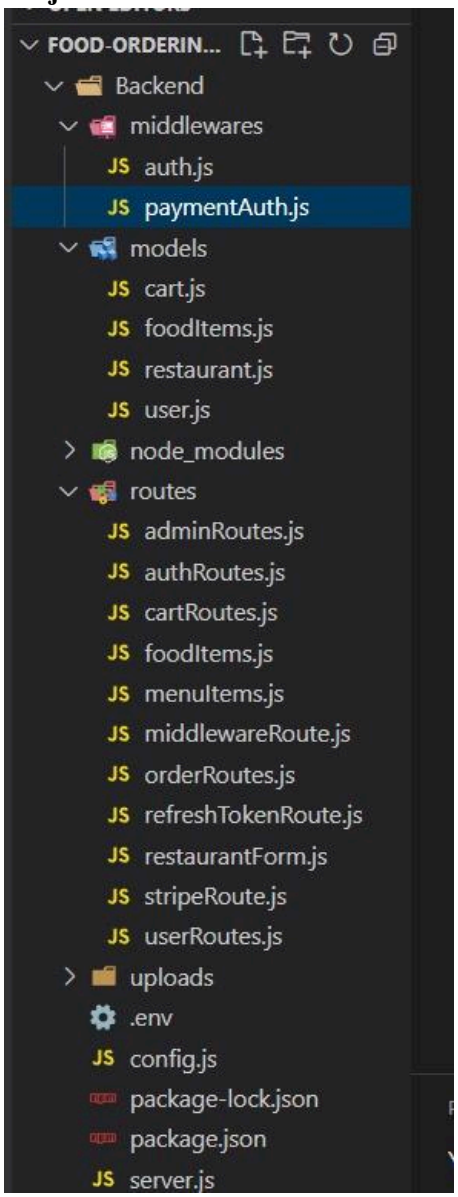
Project Title: Food Ordering System
Date:11th July 2024
Prepared by: Sankalp Sharma

Objective
This report aims to document the API development progress and key aspects of the backend services implementation for the [Your Project Title] project.

Technologies Used

- **Backend Framework:** Node.js with Express.js
- **Database:** MongoDB
- **Authentication:** JWT

Project Structure



The backend project is organized to maintain a clean and modular structure, enabling scalability and ease of maintenance. The primary components of the project are middleware, models, routes, and configuration files.

<u>Key Directories and Files</u>

1. **middlewares/**
    - **auth.js: Handles authentication-related middleware functions.**
    - **paymentAuth.js: Manages middleware for payment authentication processes.**
2. **models/**
    - **cart.js: Defines the schema for cart items.**
    - **foodItems.js: Defines the schema for food items.**

    - **restaurant.js: Defines the schema for restaurant details.**
    - **user.js: Defines the schema for user details.**
3. **routes/**
    - **adminRoutes.js: Routes for admin functionalities.**
    - **authRoutes.js: Routes for authentication processes (login, register).**
    - **cartRoutes.js: Routes for managing the cart.**
    - **foodItems.js: Routes for food items management.**
    - **menuItems.js: Routes for menu items.**
    - **middlewareRoute.js: Routes that require middleware processing.**
    - **orderRoutes.js: Routes for order management.**
    - **refreshTokenRoute.js: Routes for refreshing authentication tokens.**
    - **restaurantForm.js: Routes for restaurant form submissions.**
    - **stripeRoute.js: Routes for handling Stripe payments.**
    - **userRoutes.js: Routes for user-related functionalities.**
4. **uploads/: Directory for file uploads.**
5. **Configuration Files**
    - **.env: Environment variables configuration.**
    - **config.js: General project configuration settings.**
    - **package.json: Project dependencies and scripts.**
    - **server.js: Main server file that starts the application.**

API Endpoints

A summary of the main API endpoints and their purposes:

1. **Authentication**
    - **POST /api/user/login**: Authenticates a user and returns a token.
    - **POST /api/user/register**: Registers a new user.
2. **User Management**
    - **GET /api/user/profile**: Retrieves user profile information.
    - **PUT /api/user/profile**: Updates user profile information.
3. **Admin Management**
    - **GET /api/admin/dashboard**: Retrieves data and metrics for the admin dashboard.
4. **Restaurant Management**
    - **GET /api/restaurant**: Retrieves restaurant information.
    - **POST /api/restaurant**: Creates a new restaurant entry.
5. **Menu and Food Items**
    - **GET /api/menuItems**: Retrieves menu items.
    - **POST /api/menuItems**: Adds a new menu item.
    - **GET /api/foodItems**: Retrieves food items.
    - **POST /api/foodItems**: Adds a new food item.
6. **Order Management**
    - **GET /api/orders**: Retrieves orders for the authenticated user or admin.
    - **POST /api/orders**: Creates a new order.
7. **Cart Management**
    - **GET /api/cart**: Retrieves items in the user's cart.

    - **POST /api/cart**: Adds items to the user's cart.
8. **Payment Processing**
    - **POST /api/payment**: Processes a payment using Stripe.

Integration with Frontend

- The frontend communicates with the backend APIs hosted on [backend URL]. Key API interactions include user authentication, order management, and payment processing. Axios is used in the frontend for making HTTP requests to the backend.

Error Handling and Validation

Error handling and validation are implemented at both middleware and route levels to ensure robustness and security.

1. **Validation**
    - **User Input Validation**: Ensures all required fields are provided and valid.
    - **Data Validation**: Validates data against defined schemas in the models.
2. **Error Handling**
    - **Middleware Error Handling**: Captures and processes errors that occur during middleware execution.
    - **Route Error Handling**: Captures and responds to errors that occur during route handling, ensuring meaningful error messages are returned to the client.

Security Considerations
Security is a critical aspect of the backend implementation. Several measures are taken to ensure data protection and secure communication.

1. **Authentication and Authorization**
    - **JWT Tokens**: Secure user authentication using JSON Web Tokens (JWT).
    - **Role-Based Access Control**: Differentiate access levels for users, admins, and restaurants.
2. **Data Validation and Sanitization**
    - **Input Validation**: Ensure all user inputs are validated and sanitized to prevent SQL injection and XSS attacks.
    - **Library Usage**: Use libraries like `express-validator` to enforce strong validation rules.
3. **Environment Variables**
    - **Sensitive Data**: Store sensitive information like database credentials and API keys in environment variables.
    - **.env File**: Use a `.env` file to manage environment variables securely.