Full Stack Development with MERN

API Development and Integration Report

Date	19 July 2024
Team ID	SWTID1720150432
Project Name	EagerEats-Food Ordering App
Maximum Marks	

Project Title: Eager Eats
Date: 19 July 2024
Prepared by:

- Kishore S
- Sanjay R
- Praveen S
- Rishikeshwaran K R

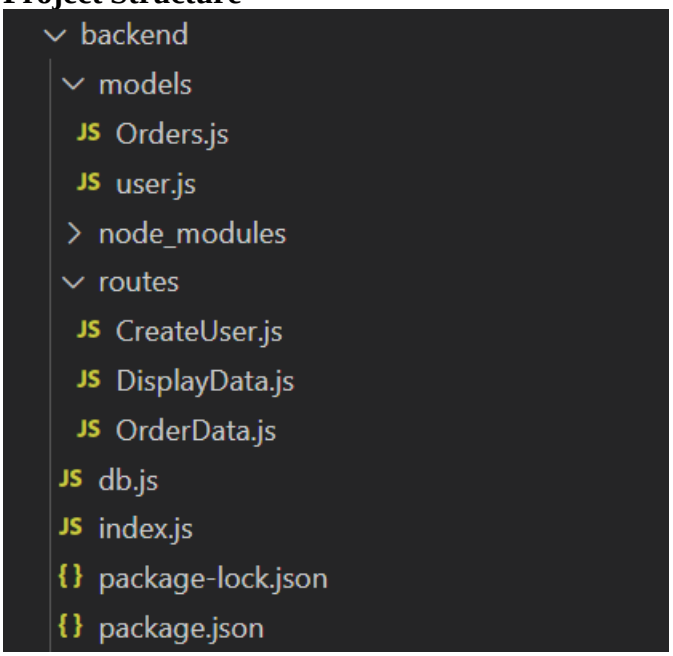
Objective

The objective of this report is to document the API development progress and key aspects of the backend services implementation for the Eager Eats project.

Technologies Used

- **Backend Framework:** Node.js with Express.js
- **Database:** MongoDB
- **Authentication:** [e.g., JWT, OAuth]

Project Structure



Key Directories and Files

1. **/models:**

   Orders.js: Contains the Mongoose schema and model for the "Orders" MongoDB collection.

   user.js: Contains the Mongoose schema and model for the "User" MongoDB collection.

2. **/routes:**

CreateUser.js: Defines the API endpoint for creating a new user and links it to the appropriate controller function.

DisplayData.js: Defines the API endpoint for displaying data and links it to the appropriate controller function.

OrderData.js: Defines the API endpoint for handling order data and links it to the appropriate controller function.

3. **db.js:**

   Handles the database connection configuration and initialization.

4. **index.js:**

   The main entry point of the application, sets up the Express server, and configures routes and middleware.

5. **package-lock.json** and **package.json**

   package-lock.json: Automatically generated file that locks the versions of the installed npm packages.

   package.json: Contains metadata about the project and lists the dependencies required by the application.

API Endpoints
Here is a summary of the API endpoints shown in the image, along with their purposes:

User Authentication

- POST /api/loginuser - Authenticates a user and returns a token.

- POST /api/createuser - Registers a new user.

Order Data

- POST /api/myorderData - Retrieves the order data for a specific user.

- POST /api/OrderData - Retrieves all order data.

Food Data

- POST /api/foodData - Retrieves all food data.

Integration with Frontend:

The backend communicates with the frontend via RESTful APIs. Key points of integration include:

User Authentication:
- Tokens: Secure tokens (JWT) are passed between frontend and backend to handle authentication. Tokens are stored securely in HttpOnly cookies or secure storage.

Data Fetching:
- API Calls: Frontend components make API calls to fetch necessary data for display and interaction, including restaurant listings, menu items, order details, and user profiles.
- Real-time Updates: WebSockets or Server-Sent Events (SSE) for real-time order status updates.

Error Handling and Validation:

Describe the error handling strategy and validation mechanisms:

Error Handling:
- Centralized Error Handling: Use middleware to handle errors globally. Custom error classes can be defined for different types of errors (e.g., validation errors, authentication errors, server errors).
- Error Logging: Implement logging (e.g., using Winston) to capture and store error details for debugging and monitoring.

Validation:
- Input Validation: Use libraries like Joi or express-validator to validate incoming data. Ensure that user inputs are sanitized to prevent injection attacks.
  - User Input: Validate registration, login details, order information, payment details, and feedback.
  - API Responses: Validate data before sending it to the frontend to ensure consistency and correctness.

Security Considerations:

Outline the security measures implemented:

Authentication:
- Secure Token-Based Authentication: Implement JWT for user authentication. Ensure tokens are securely stored and managed.
- Multi-Factor Authentication (MFA): Optional MFA for an additional layer of security.

Data Encryption:
- Encryption at Rest: Use AES encryption for sensitive data stored in the database (e.g., user passwords, payment details).
- Encryption in Transit: Use HTTPS (SSL/TLS) to encrypt data transmitted between the frontend and backend.