**Full Stack Development with MERN**

**API Development and Integration Report**

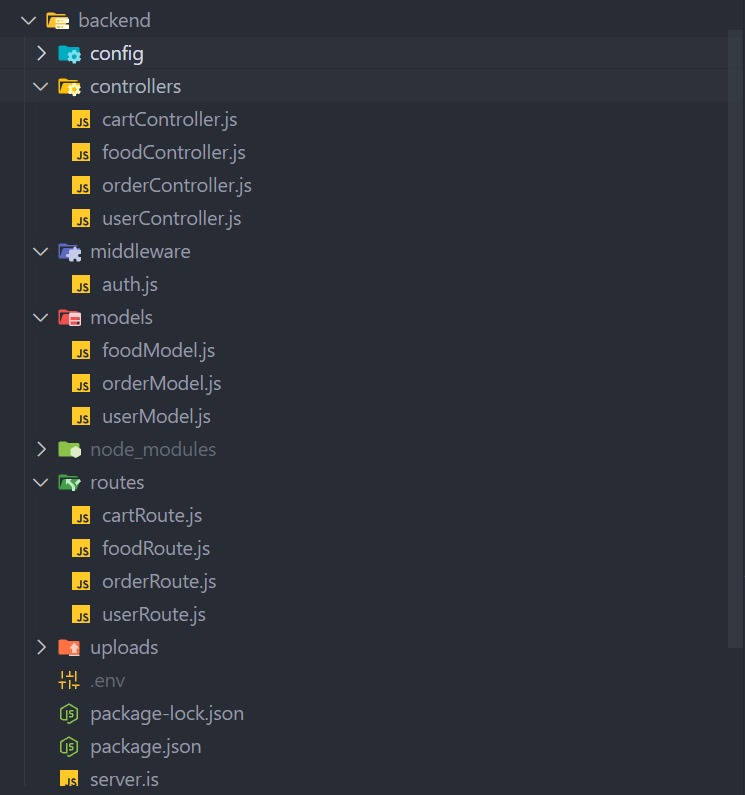
| Date | 11th July, 2024 |
| --- | --- |
| Team ID | SWTID1720019605 |
| Project Name | Project - Grocery web app |
| Maximum Marks |  |

**Project Title:** Farmzi  
**Date:**11/07/24  
**Prepared by:** SWTID1720019605

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

**Technologies Used**

* **Backend Framework:** Node.js with Express.js
* **Database:** MongoDB
* **Authentication:** Validator , JSONWebtoken , bcrypt

**Project Structure**  


**Key Directories and Files**

1. **/controllers**
   * cartController.js
   * foodController.js
   * orderController.js
   * userController.js
2. **/models**
   * foodModel.js
   * orderModel.js
   * userModel.js
3. **/routes**
   * cartRoute.js
   * foodRoute.js
   * orderRoute.js
   * userRoute.js
4. **/middlewares**
   * auth.js
5. **/config**
   * db.js

**API Endpoints**  
A summary of the main API endpoints and their purposes:

**Food Management:**

/api/food/add : Adds a new food item to the database.

/api/food/list : Retrieves a list of all available food items.

/api/food/remove : Removes a specific food item from the database.

**User Management:**

/api/user/register : Registers a new user.

/api/user/login : Authenticates a user and provides a token.

**Cart Management:**

/api/cart/add : Adds an item to the user's cart.

/api/cart/remove : Removes an item from the user's cart.

/api/cart/get : Retrieves the current contents of the user's cart.

**Order Management:**

/api/order/place : Place a new order for the items in the user's cart.

/api/order/verify **:** Verifies the details of an order.

/api/order/userorders : Retrieves a list of all orders placed by the user.

/api/order/list : Retrieves a list of all orders.

/api/order/status : Checks the status of a specific order.

**User Authentication**

* **POST /api/user/register** - Registers a new user.
* **POST /api/user/login** - Authenticates a user and returns a token.

**User Management**

* **GET /api/user/-** Retrieves user information by ID.
* **PUT /api/user/**- Updates user information by ID.

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

* **User Authentication:** JSON Tokens
* **Data Fetching:** CORS

**Error Handling and Validation**  
Describe the error handling strategy and validation mechanisms:

* **Error Handling:** Middleware is used to catch and handle errors related to invalid or expired JSON Web Tokens (JWTs) before they disrupt the application flow.
* **Validation:** express-validator is utilized to ensure that incoming requests contain valid and correctly formatted data by defining validation rules and checking for errors in the request.

**Security Considerations**  
Outline the security measures implemented:

* **Authentication:** JSON web token
* **Data Encryption:** bcrypt for hashing the passwords