**Full Stack Development with MERN**

**Project Documentation**

**1. Introduction**

* **Project Title:** Grocery web App
* **Team Members:**

|  |  |  |  |
| --- | --- | --- | --- |
| S.No | Name | Email Id | Naan Mudhalvan Id |
| 1. | Stephy |  |  |
| 2. | Praveen I |  |  |
| 3. | Raghini |  |  |
| 4. | Dhanush R |  |  |

**2. Project Overview**

To create a robust and user-friendly platform for managing grocery shopping online, addressing the growing demand for e-commerce solutions in this sector.

* **Purpose:**

1. The Grocery Web App serves as an online platform enabling users to Browse and search for groceries.
2. Add items to a virtual shopping cart.
3. Securely checkout and manage orders.
4. Allow admins to add, update, and manage inventory efficiently.

* **Features:**
  1. User Features:

User registration and login.

Browse products by category or search function.

Add/remove items from the cart.

Place orders with secure payment options.

* 1. Admin Features:

Dashboard for managing inventory.

Track and update order statuses.

View customer information and sales reports.

**3. Architecture**

* **Frontend:**
  1. Built with React.js, ensuring modular and reusable components for scalability.
  2. Implements routing with React Router.
  3. State management handled using Redux or Context API for seamless data flow across components.
* **Backend:**
  1. Powered by Node.js and Express.js, delivering RESTful APIs.
  2. Middleware layers for request validation, error handling, and authentication.
* **Database:**

MongoDB serves as the database, offering high performance and scalability.

**Key collections:**

* 1. Users: Stores user credentials and details.
  2. Products: Maintains product information like name, category, price, stock.
  3. Orders: Tracks orders with details on products, customer, and status.

**System Diagram:**

The architecture follows the MERN stack:

Client (React) → Sends requests via Axios/Fetch API.

Server (Node.js) → Processes requests and interacts with MongoDB.

Database (MongoDB) → Stores and retrieves data efficiently.

**4. Setup Instructions**

* **Prerequisites:**
  1. Node.js installed on your machine.
  2. MongoDB database running locally or on a cloud platform (e.g., MongoDB Atlas).
* **Installation:**
  1. Clone the repository:

Git clone [repository\_url]

* 1. Navigate to the directories:

Cd client

Cd server

* 1. Install dependencies:

npm install

* 1. Create a .env file in the server directory and add:

MongoDB connection URI.

JWT secret key.

* 1. Run the development servers (see 6. Running the Application below).

**5. Folder Structure**

* **Client:**

Src/

│ └── components/ # Reusable UI components

│ └── pages/ # Main page layouts

│ └── redux/ # Redux actions and reducers

│ └── utils/ # Helper functions

* **Server:**

Src/

│ └── routes/ # API endpoints

│ └── controllers/ # Business logic

│ └── models/ # Mongoose schemas

│ └── middlewares/ # Authentication and error handlers

**6. Running the Application**

* Provide commands to start the frontend and backend servers locally.
  + **Frontend:** npm start in the client directory.

Cd client

Npm start

* + **Backend:** npm start in the server directory.

Cd server

Npm start

**Full Stack:**

Run both services simultaneously using tools like Concurrently or separate terminals.

**7. API Documentation**

Example API Endpoints:

User Registration

Method: POST

Endpoint: /api/users/register

Request Body:

{

"name": "John Doe",

"email": "john@example.com",

"password": "password123"

}

Response:

{

“message”: “User registered successfully!”

}

**8. Authentication**

1. JWT-based Authentication: Tokens are issued during login and stored on the client (e.g., cookies, localStorage).
2. Protected Routes: Middleware ensures only authenticated users can access sensitive endpoints.

**9. User Interface**

Add screenshots of the following:

1. Homepage: Displays featured products and categories.
2. Product Listing: Shows products with filters (category, price, etc.).
3. Cart Page: Displays items added to the cart with a checkout option.
4. Admin Dashboard: Interface to manage products and orders.

**10. Testing**

1. Unit Testing: Validate individual components and functions using Jest.
2. Integration Testing: Test API endpoints with tools like Postman or Newman.

Example Tools Used:

1. Jest for frontend testing.
2. Mocha/Chai for backend testing

**11. Screenshots or Demo**

1. Screenshots showcasing key features.
2. A link to the live demo or a recording.

**12. Known Issues**

* Pagination Bugs: Pagination on large datasets may not work smoothly.
* Limited Error Handling: Inconsistent user feedback for errors.

**13. Future Enhancements**

* AI Product Recommendations: Personalize the user experience.
* Mobile App Integration: Expand to native mobile platforms.
* Real-time Notifications: Notify users about order updates.