Grocery Web app using MERN

A grocery web app is an online platform that allows users to purchase groceries and related products conveniently from their devices. These apps have gained popularity due to the increasing demand for convenient shopping solutions and the ability to have groceries delivered directly to one's doorstep. Below are some key features and a description of a typical grocery web app:

User-Friendly Interface

Grocery web apps are designed with an intuitive and user-friendly interface to ensure that users, regardless of their technical proficiency, can easily navigate and find the products they need.

Product Management

Retailers can upload, organize, and manage product listings. Users can browse through various categories, view detailed product descriptions, and add items to their shopping cart.

Shopping Cart and Checkout

Users can add products to their virtual shopping cart and proceed to a streamlined checkout process. Features like saved payment methods and address autofill enhance the checkout experience.

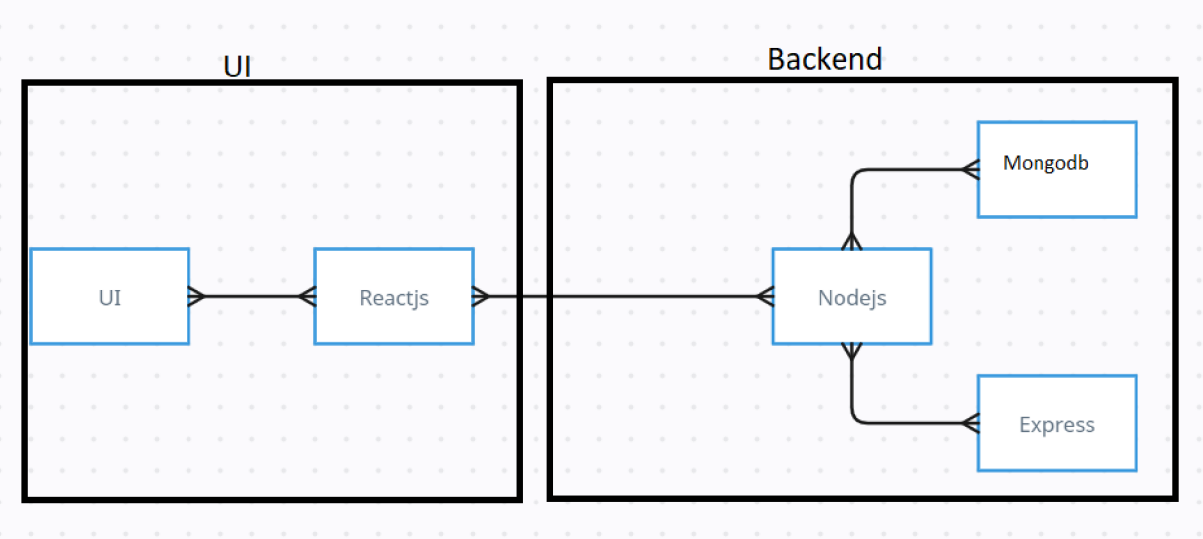
Order Tracking

Users can track their orders in real-time, from order confirmation to delivery. Notifications and updates keep customers informed about the status of their delivery.

Payment Methods

Multiple payment options, such as credit/debit cards, digital wallets, and cash on delivery, are usually available to accommodate different user preferences.

TECHNICAL ARCHITECTURE

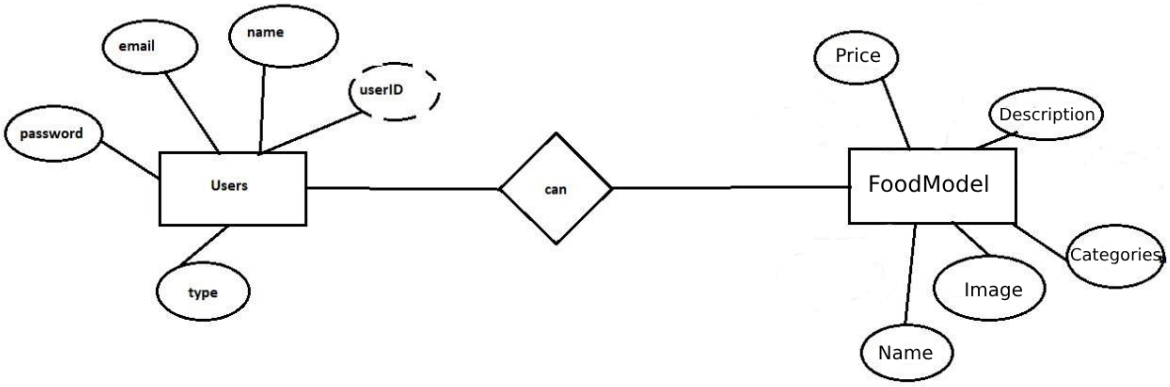


The technical architecture of grocery app follows a client-server model, where the frontend serves as the client and the backend acts as the server. The frontend encompasses not only the user interface and presentation but also incorporates the axios library to connect with backend easily by using RESTful Apis.

On the backend side, we employ Express.js frameworks to handle the server-side logic and communication.
For data storage and retrieval, our backend relies on MongoDB. MongoDB allows for efficient and scalable storage of user data and necessary information about the place.

Together, the frontend and backend components, along with Express.js, and MongoDB, form a comprehensive technical architecture for our grocery app. This architecture enables real-time communication, efficient data exchange, and seamless integration, ensuring a smooth and immersive blogging experience for all users.

ER DIAGRAM

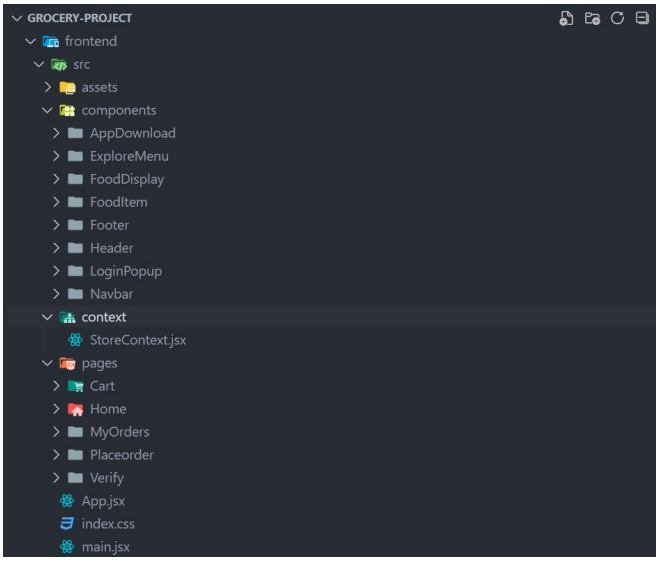


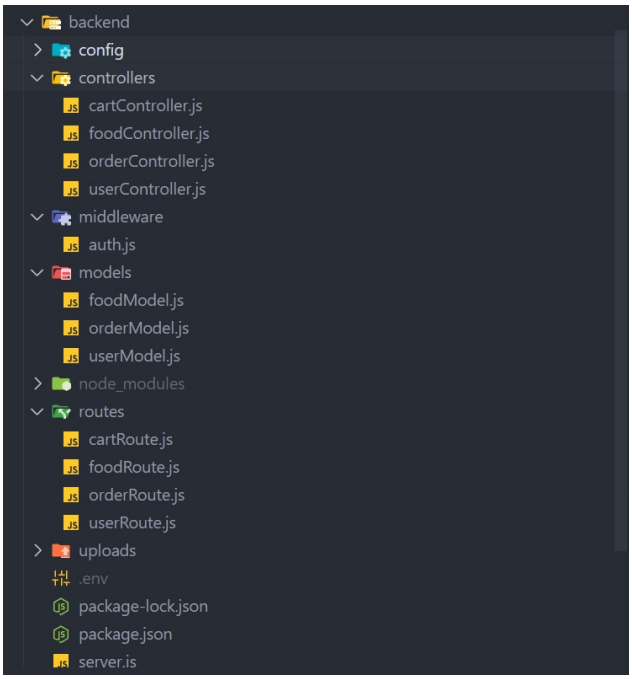
PROJECT STRUCTURE

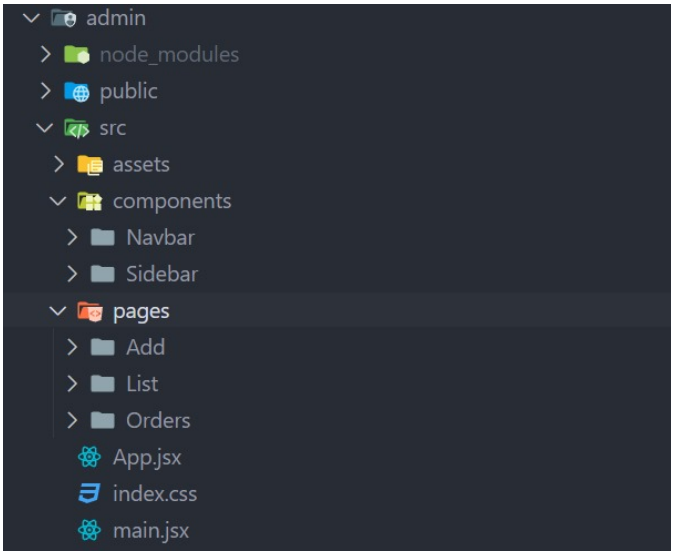
The first image is of frontend part which is showing all the files and folders that have been used in UI development

The second image is of Backend part which is showing all the files and folders that have been used in backend development

The third image is of admin panel which is showing all the files and folders that have been used in admin panel development.







modular architecture.

PRE-REQUISTES:

Here are the key prerequisites for developing a full-stack application using Node.js, Express.js, MongoDB, React.js:

✔Vite:
Vite is a new frontend build tool that aims to improve the developer experience for development with the local machine, and for the build of optimized assets for production (go live). Vite (or ViteJS) includes: a development server with ES _native_ support and Hot Module Replacement; a build command based on rollup.

```
npm create vite@latest
```

✔Node.js and npm:
Node.js is a powerful JavaScript runtime environment that allows you to run JavaScript code on the server-side. It provides a scalable and efficient platform for building network applications. Install Node.js and npm on your development machine, as they are required to run JavaScript on the server-side.
Download: https://nodejs.org/en/download/
Installation instructions: https://nodejs.org/en/download/package-manager/

```
npm init
```

✔Express.js:

Express.js is a fast and minimalist web application framework for Node.js. It simplifies the process of creating robust APIs and web applications, offering features like routing, middleware support, and modular architecture.
Install Express.js, a web application framework for Node.js, which handles server-side routing, middleware, and API development.
Installation: Open your command prompt or terminal and run the following command:

```
npm install express
```

✔**MongoDB:**

MongoDB is a flexible and scalable NoSQL database that stores data in a JSON-like format. It provides high performance, horizontal scalability, and seamless integration with Node.js, making it ideal for handling large amounts of structured and unstructured data.

Set up a MongoDB database to store your application's data.
Download: https://www.mongodb.com/try/download/community Installation instructions: https://docs.mongodb.com/manual/installation/

✔**React.js:**
React.js is a popular JavaScript library for building user interfaces. It enables developers to create interactive and reusable UI components, making it easier to build dynamic and responsive web applications.
Install React.js, a JavaScript library for building user interfaces.
Follow the installation guide: https://reactjs.org/docs/create-a-new-react-app.html

✔**HTML, CSS, and JavaScript**: Basic knowledge of HTML for creating the structure of your app, CSS for styling, and Javascript for client side interactivity is essential.

✔**Database Connectivity**: Use a MongoDB driver or an Object-Document Mapping (ODM) library like Mongoose to connect your Node.js server with the MongoDB database and perform CRUD (Create, Read, Update, Delete) operations. To Connect the Database with Node JS go through the below provided link: https://www.section.io/engineering-education/nodejs- mongoosejs-mongodb/

✔ **Front-end Framework**: Utilize Reactjs to build the user-facing part of the application, including entering booking room, status of the booking, and user interfaces for the admin dashboard.
For making better UI we have also used some libraries like material UI and boostrap.

✔ **Version Control**: Use Git for version control, enabling collaboration and tracking changes throughout the development process. Platforms like GitHub or Bitbucket can host your repository.
Git: Download and installation instructions can be found at: https://git-scm.com/downloads

✔**Development Environment**: Choose a code editor or Integrated Development Environment (IDE) that suits your preferences, such as Visual Studio Code, Sublime Text, or WebStorm.

1. Visual Studio Code: Download from https://code.visualstudio.com/download

To run the existing Video Conference App project downloaded from GitHub:
Follow below steps:
Clone the Repository:

a. Open your terminal or command prompt.
b. Navigate to the directory where you want to store the e-commerce app.
c. Execute the following command to clone the repository:

d. Git clone https://github.com/Drockz-Tech/Grocery-webapp.git

Install Dependencies:

2. Navigate into the cloned repository directory:
   cd containment-zone
3. Install the required dependencies by running the following commands:
   cd frontend npm
   install cd
   ../backend
   npm install
   cd admin npm
   install

Start the Development Server:

4. To start the development server, execute the following command: npm start
5. The grocery app will be accessible at http://localhost:5173

You have successfully installed and set up the Containment Zone app on your local machine. You can now proceed with further customization, development, and testing as needed.

Roles and Responsibilities:

The roles and responsibilities of the users can be inferred from the API endpoints defined in the code. Here is a summary:

Admin:

a. Can add food items for the customer.
b. Also delete the food items
c. Can update order status.

Customer:

a. Can signup and login
b. Can add and remove items from the cart.
c. Can place order.
d. Can track the order.

Project Flow:

Before starting to work on this project, let's see the demo. **Project demo:** https://drive.google.com/file/d/1TFm5LgohmHeLw-xrjx8kbYOA2pTp1GSP/view?usp=sharing

Use the code in**:** **https://github.com/Drockz-Tech/Grocery-webapp.git**

or follow the videos below for better understanding.

Milestone 1: Project setup and configuration.

a. **Folder setup:**
    i. Create frontend and
    ii. Backend folders and
    iii. Admin folder
b. **Installation of required tools:**

i. Open the frontend folder to install necessary tools

For frontend, we use: ▪
React Vite
- React-dom
- Axios
- react-router-dom

ii. Open the backend folder to install necessary tools

For backend, we use:

- Cors
- Bcrypt
- Body-parser
- Dotenv
- Mongoose
- Express
- Stripe
- Validator
- Multer
- Nodemon
- jsonwebtoken

Milestone 2: Backend Development

c. **Setup express server**
   i. Create server.js file in the server (backend folder).
   ii. Edit the mongo db connection string in db.js file inside config folder
   iii. define port number, stripe secret key and JWT key in env file to access it.
   iv. Configure the server by adding cors, body-parser.

d. **Configure MongoDB**

e. 1. Import mongoose.
2.Add database connection from db.js file present in config folder 3.Create a model folder to store all the DB schemas user, food, and orders.

f. **Add authentication:** for this,
You need to make middleware folder and in that make auth.js file for the authentication of the projects and can use in.