**Project Setup And Configuration**

**Milestone 1: Project Setup and Configuration:**

**1. Install required tools and software:**

• Node.js.

Reference Article: <https://www.geeksforgeeks.org/installation-of-node-js-on-windows/>

• Git.

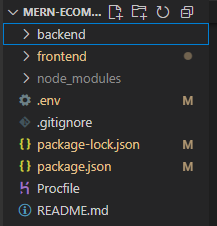
Reference Article: <https://git-scm.com/book/en/v2/Getting-Started-Installing-Git>

**2. Create project folders and files:**

• frontend folders.

• backend folders

Referral Image:



**Backend Development**

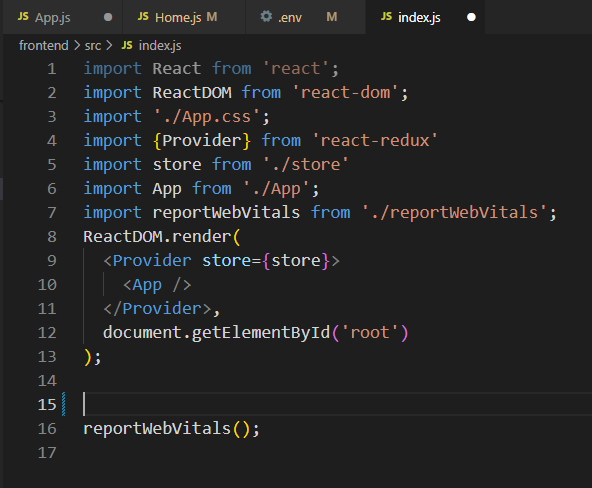
**1. Setup express server:**

• Create index.js file.

• Create an express server on your desired port number.

• Define API’s

Reference Image:



Now your express is successfully created.

**Set Up Project Structure:**

• Create a new directory for your project and set up a package.json file using the npm init command.

• Install necessary dependencies such as Express.js, Mongoose, and other required packages.

Reference Images:

  
**2. Database Configuration:**

• Set up a MongoDB database either locally or using a cloud-based MongoDB service like MongoDB Atlas or use locally with MongoDB compass.

• Create a database and define the necessary collections for admin, users, products,  orders and other relevant data.

**3. Create Express.js Server:**

• Set up an Express.js server to handle HTTP requests and serve API endpoints.

• Configure middleware such as body-parser for parsing request bodies and cors for handling cross-origin requests.

**4. Define API Routes:**

• Create separate route files for different API functionalities such as users, orders, and authentication.

• Define the necessary routes for listing products, handling user registration and  login,managing orders, etc.

• Implement route handlers using Express.js to handle requests and interact with the database.

**5. Implement Data Models:**

• Define Mongoose schemas for the different data entities like products, users,  and orders.

• Create corresponding Mongoose models to interact with the MongoDB database.

 • Implement CRUD operations (Create, Read, Update, Delete) for each model to perform database operations.

**6. User Authentication:**

• Create routes and middleware for user registration, login, and logout.

 • Set up authentication middleware to protect routes that require user authentication.

**7. Handle new products and Orders:**

• Create routes and controllers to handle new product listings, including fetching products data from the database and sending it as a response.

• Implement ordering(buy) functionality by creating routes and controllers to  handle order requests, including validation and database updates.

**8. Admin Functionality:**

• Implement routes and controllers specific to admin functionalities such as adding products, managing user orders, etc.

• Add necessary authentication and authorization checks to ensure only authorized admins can access these routes.

**9. Error Handling:**

• Implement error handling middleware to catch and handle any errors that occur during the API requests.

• Return appropriate error responses with relevant error messages and HTTP status codes.

**Database Development**

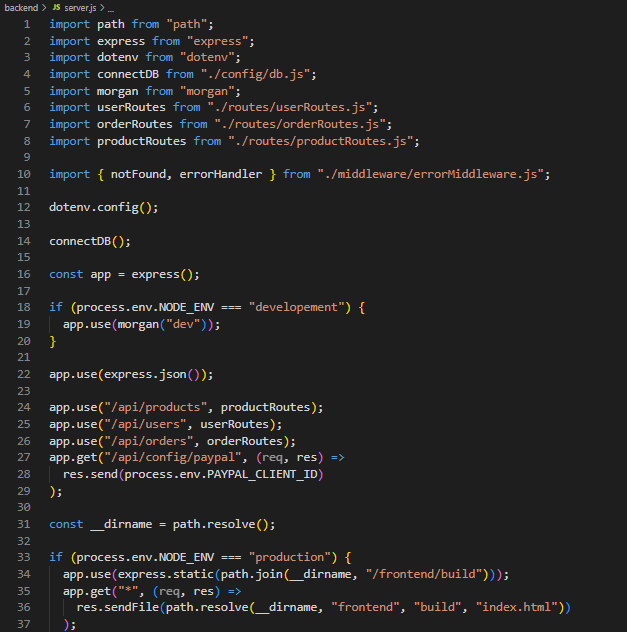
**Create database in cloud**

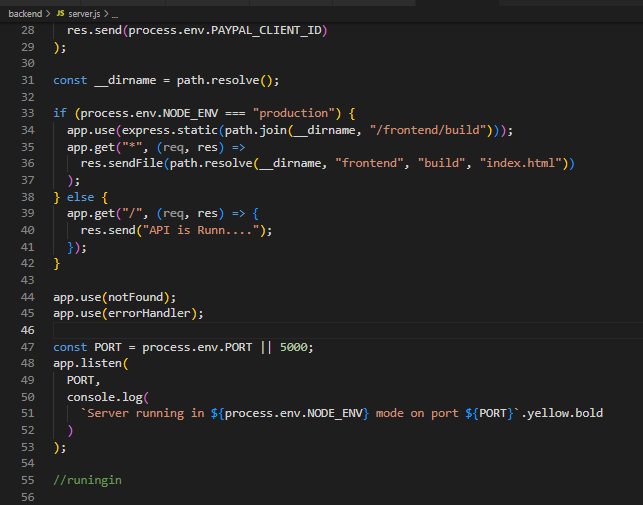
• Install Mongoose.

• Create database connection.

Reference Article: <https://www.mongodb.com/docs/atlas/tutorial/connect-to-your-cluster/>

Reference Image:





**Schema use-case:**

**1. User Schema:**

• Schema: userSchema

• Model: ‘User’

• The User schema represents the user data and includes fields such as username, email, and password.

• It is used to store user information for registration and authentication purposes.

 • The email field is marked as unique to ensure that each user has a unique email address

**2. Product Schema:**

• Schema: productSchema

• Model: ‘Product’

• The Product schema represents the data of all the products in the platform.

• It is used to store information about the product details, which will later be useful for  ordering .

**3. Orders Schema:**

• Schema: ordersSchema

• Model: ‘Orders’

• The Orders schema represents the orders data and includes fields such as userId,  product Id, product name, quantity, size, order date, etc.,

• It is used to store information about the orders made by users.

• The user Id field is a reference to the user who made the order.

**4. Cart Schema:**

• Schema: cartSchema

• Model: ‘Cart’

• The Cart schema represents the cart data and includes fields such as userId, product  Id, product name, quantity, size, order date, etc.,

• It is used to store information about the products added to the cart by users. • The user Id field is a reference to the user who has the product in cart.

**5. Admin Schema:**

• Schema: adminSchema

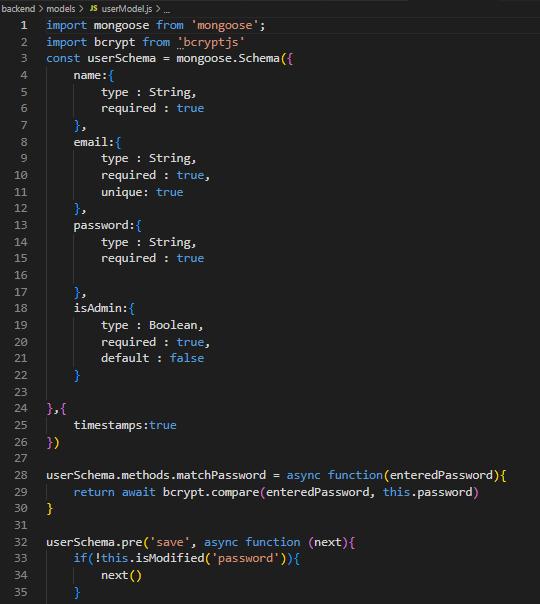
• Model: ‘Admin’

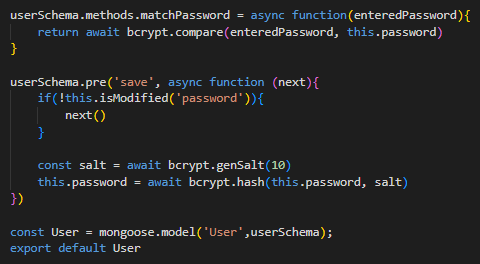
• The admin schema has essential data such as categories, banner.

**Code Explanation:**

**Schemas:**

Now let us define the required schemas





**Frontend development**

**1. Setup React Application:**

• Create a React app in the client folder.

• Install required libraries

• Create required pages and components and add routes.

**2.Design UI components:**

• Create Components.

• Implement layout and styling.

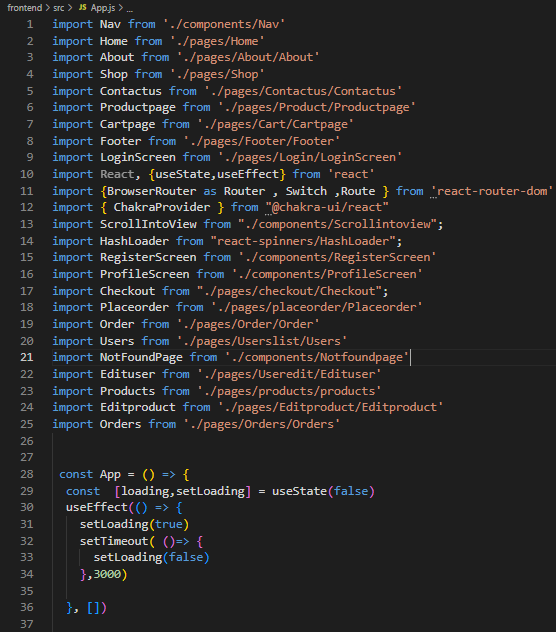
• Add navigation.

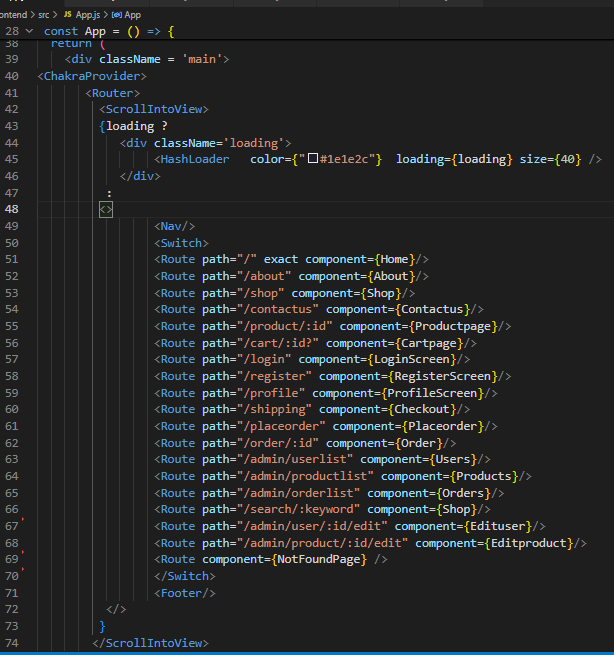
**3.Implement frontend logic:**

• Integration with API endpoints.

• Implement data binding.

Reference Image:



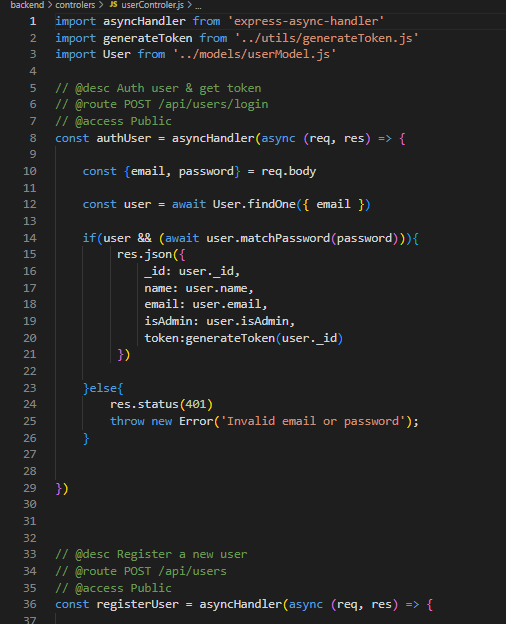


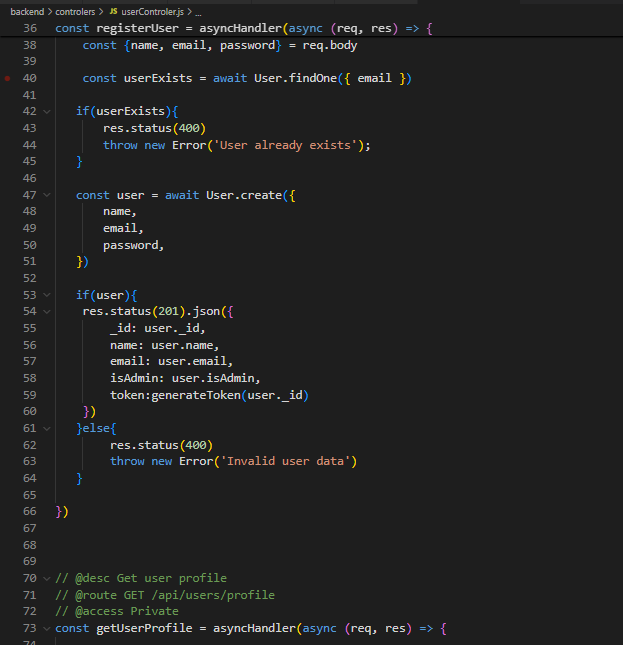
### Project Implementation & Execution

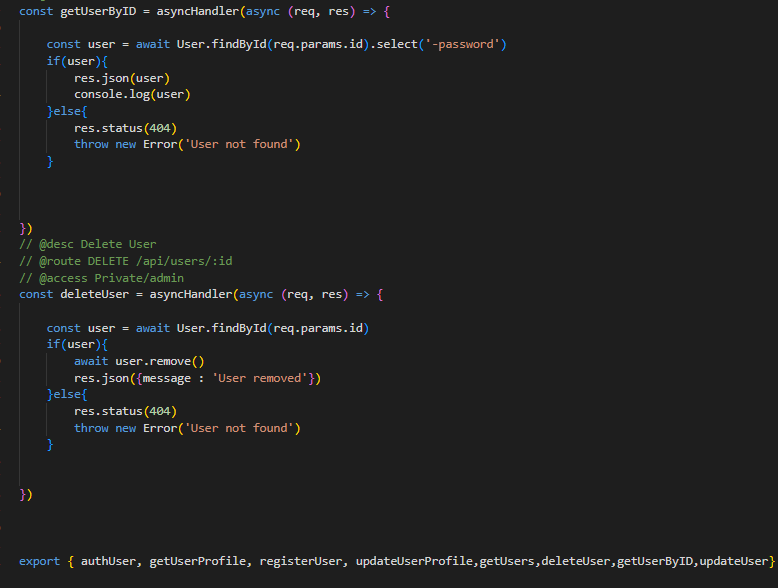
**User Authentication:**

·       **Backend**

Now, here we define the functions to handle http requests from the client for authentication.

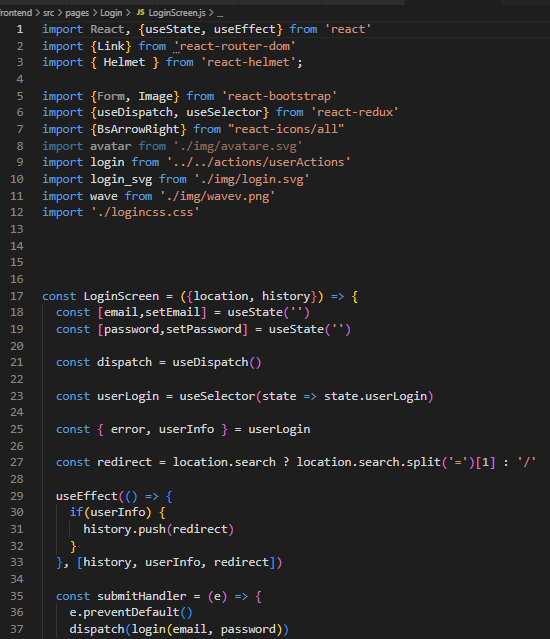






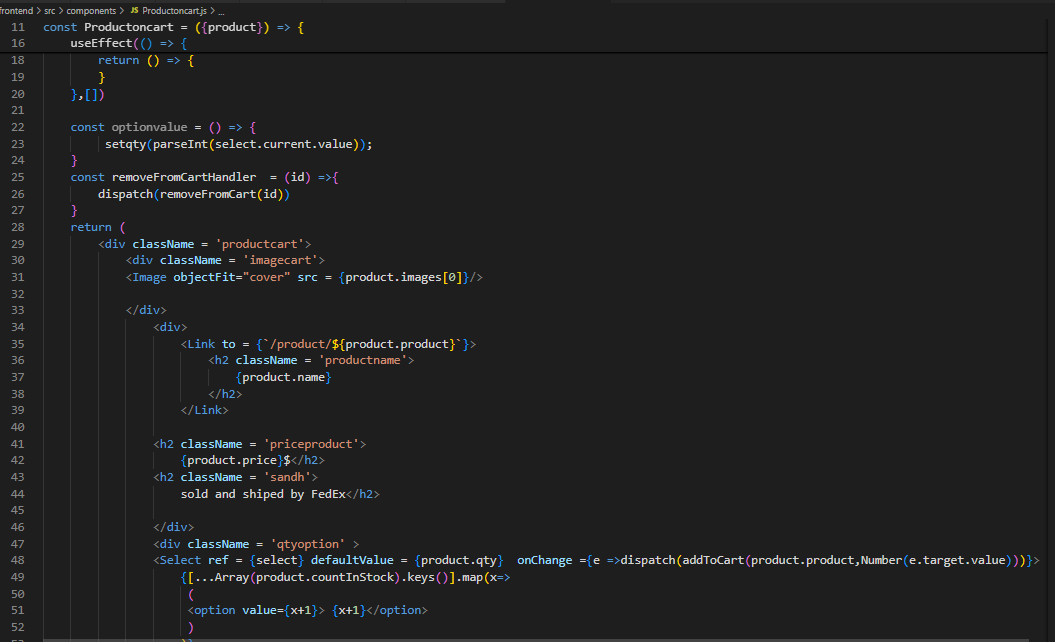
**Frontend**

**Login:**



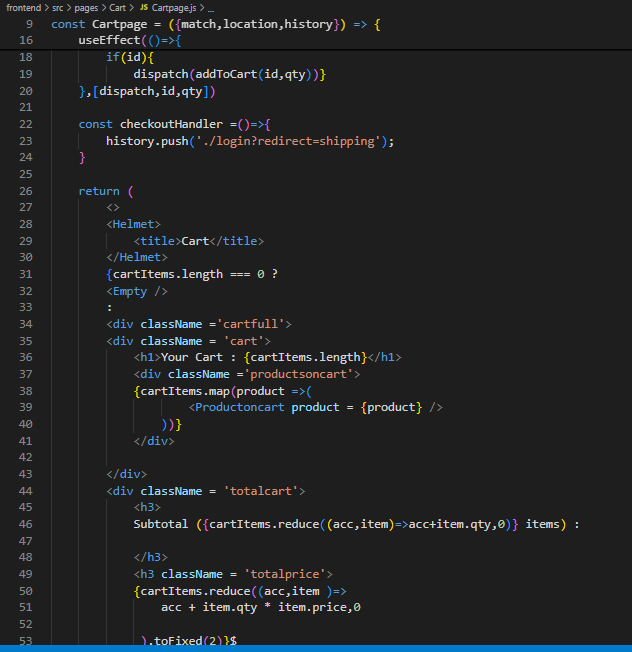
In the home page, we’ll fetch all the products available in the platform along with the filters.

**Fetching products:**



**Add product to cart:**

Here, we can add the product to the cart or can buy directly.



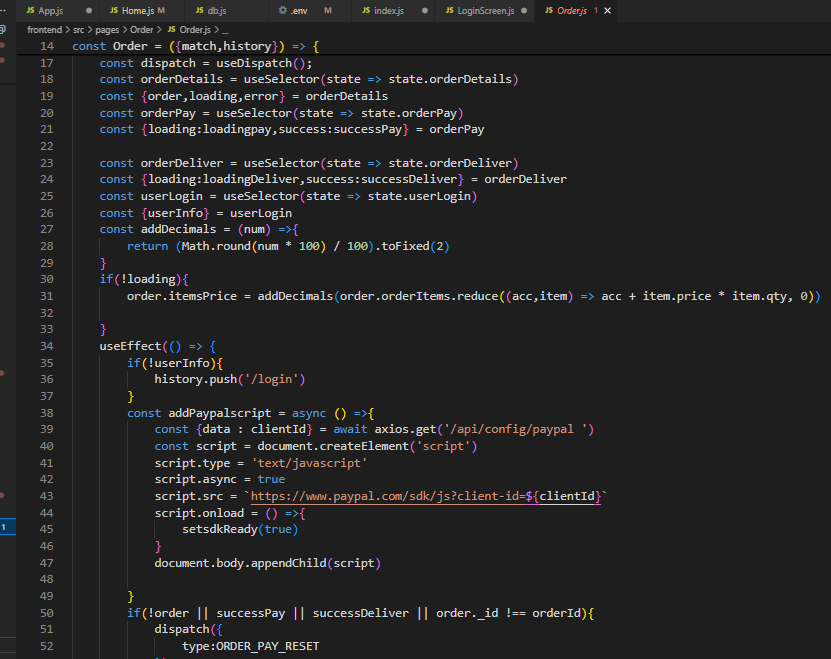
Backend**:** In the backend, if we want to buy, then with the address and payment method, we process buying. If we need to add the product to the cart, then we add the product details along with the user Id to the cart collection.

Buy product:

**Order products:**

Now, from the cart, let’s place the order

·       Frontend

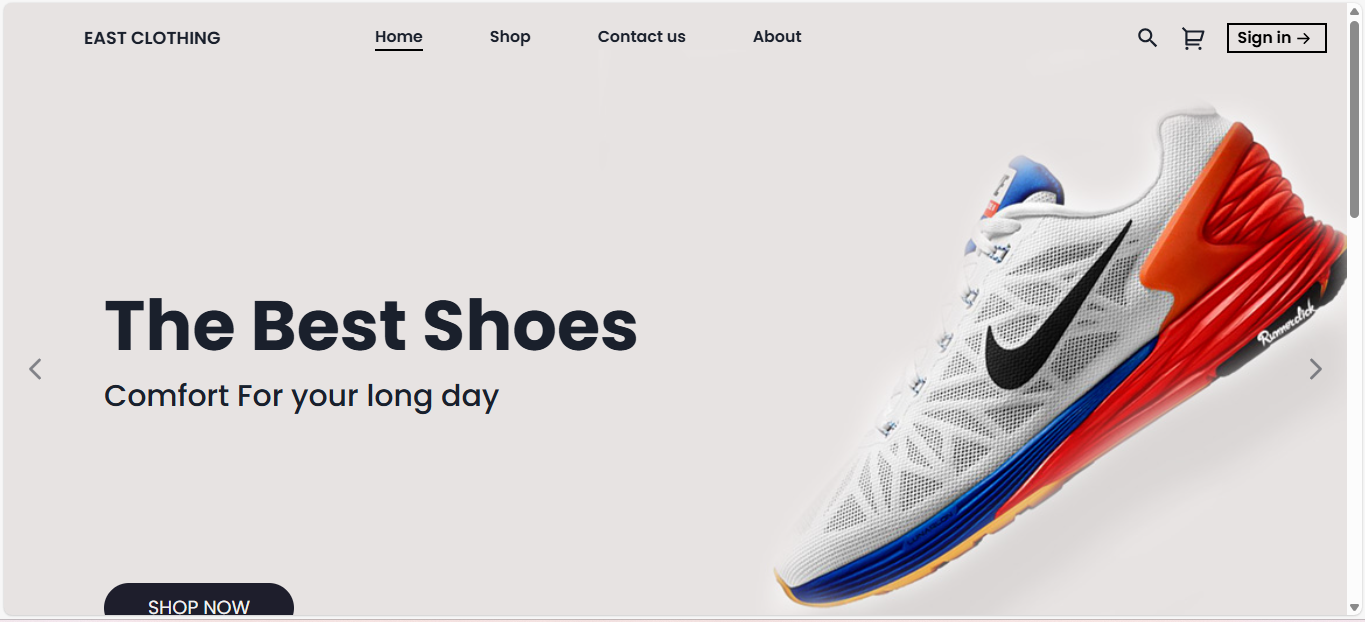


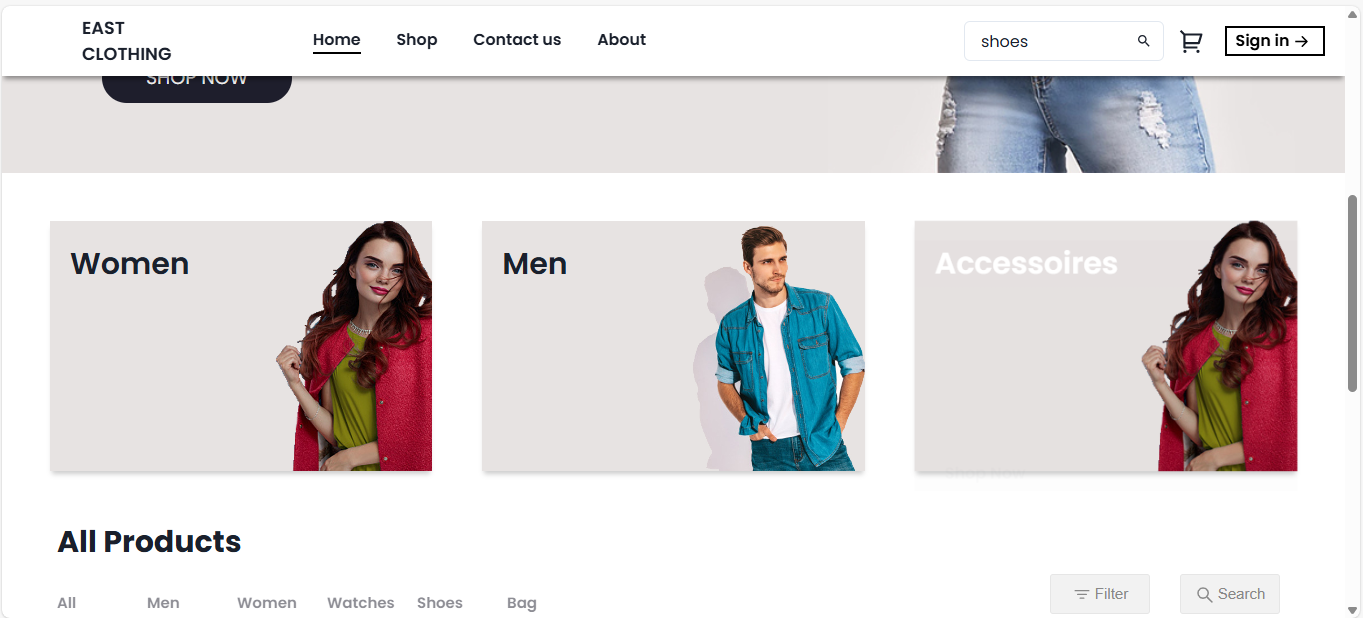
  ·       Backend

 In the backend, on receiving the request from the client, we then place the order for the products in the cart with the specific user Id.

**Demo UI images:**

·       **Landing page**





**Authentication**

