Full Stack Development with MERN

INTODUCTION :

Project Title

SHOPEZ : ONE-STOP SHOP FOR ONLINE PURCHASES

Team ID: LTVIP2025TMID21133

Team Leader: M.Naga Venkata Balaji

Team member: M.Charishma

Team member: M.Sahithi

Team member: M.Reshma Teja Sri

PURPOSE

The purpose of this project is to design, develop, and analyze . **ShopEZ**, an integrated e-commerce platform that serves as a one-stop shop for online purchases. The platform is intended to simplify the online shopping experience by offering a wide range of products across multiple categories—including fashion, electronics, home essentials, and more—within a single unified system. This project aims to:

- Address the growing consumer demand for convenience, speed, and variety in online shopping.
- Explore and implement essential features such as secure user authentication, product browsing, cart management, order processing, and admin-level control.
- Create an efficient and seamless shopping experience that enhances customer satisfaction while streamlining backend operations for administrators.

By undertaking this project, the objective is not only to deliver a technically sound application but also to understand the business, functional, and user-experience challenges associated with building a full-fledged e-commerce system.

FEATURES

It consists of some key features which include:

1. Wide product range across categories like fashion, electronics, and home essentials.
2. Secure user authentication with role-based access control.
3. User-friendly shopping cart with dynamic add, update, and remove functionality.
4. Seamless checkout process with order summary and delivery details.
5. Real-time order tracking and detailed order history for users.
6. Admin panel for inventory management including product CRUD operations.
7. Dashboard with analytics for tracking users, orders, and revenue.
8. Automated invoice generation after successful purchases.
9. Responsive design optimized for all screen sizes and devices.
10. Customer support section with contact form and order-related queries.

DESCRIPTION

ShopEZ is a full-stack e-commerce web application designed to serve as a one-stop online shopping platform for users. It provides a seamless and intuitive shopping experience by offering a wide variety of products across multiple categories such as fashion, electronics, groceries, and home essentials. Built using the MERN stack (MongoDB, Express.js, React.js, and Node.js), ShopEZ allows users to register, log in, browse products, add items to their cart, and securely place orders.

The platform also includes an admin dashboard for managing inventory, viewing orders, and tracking business performance through analytics. With features like secure authentication, responsive design, real-time order tracking, and invoice generation, ShopEZ aims to deliver both customer satisfaction and operational efficiency. This project showcases how modern web technologies can be integrated to create a scalable, user-friendly, and business-ready online retail solution.

SCENARIO

Sarah's Birthday Gift

Sarah, a busy professional, is scrambling to find the perfect birthday gift for her best friend, Emily. She knows Emily loves fashion accessories, but with her hectic schedule, she hasn't had time to browse through multiple websites to find the ideal present. Feeling overwhelmed, Sarah turns to ShopEZ to simplify her search.

1. Effortless Product Discovery: Sarah opens ShopEZ and navigates to the fashion accessories category. She's greeted with a diverse range of options, from chic handbags to elegant jewelry. Using the filtering options, Sarah selects "bracelets" and refines her search based on Emily's preferred style and budget.

2. Personalized Recommendations: As Sarah scrolls through the curated selection of bracelets, she notices a section labeled "Recommended for You." Intrigued, she clicks on it and discovers a stunning gold bangle that perfectly matches Emily's taste. Impressed by the personalized recommendation, Sarah adds it to her cart.

3.Seamless Checkout Process: With the bracelet in her cart, Sarah proceeds to checkout. She enters Emily's address as the shipping destination and selects her preferred payment method. Thanks to ShopEZ's secure and efficient checkout process, Sarah completes the transaction in just a few clicks.

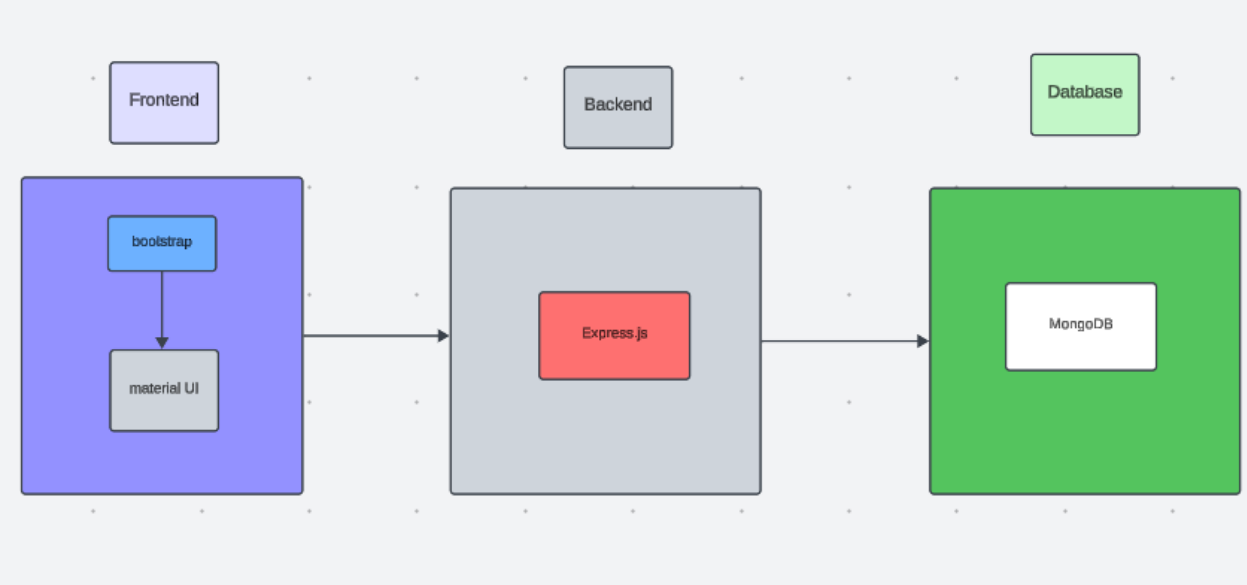
4. Order Confirmation: Moments after placing her order, Sarah receives a confirmation email from ShopEZ. Relieved to have found the perfect gift for Emily, she eagerly awaits its arrival.

5. Efficient Order Management for Sellers: Meanwhile, on the other end, the seller of the gold bangle receives a notification of Sarah's purchase through ShopEZ's seller dashboard. They quickly process the order and prepare it for shipment, confident in ShopEZ's streamlined order management system.

6. Celebrating with Confidence: On Emily's birthday, Sarah presents her with the beautifully packaged bracelet, knowing it was chosen with care and thoughtfulness. Emily's eyes light up with joy as she adorns the bracelet, grateful for Sarah's thoughtful gesture.

In this scenario, ShopEZ proves to be the perfect solution for Sarah's busy lifestyle, offering a seamless and personalized shopping experience. From effortless product discovery to secure checkout and efficient order management, ShopEZ simplifies the entire process, allowing Sarah to celebrate Emily's birthday with confidence and ease.

TECHNICAL ARCHITECTURE



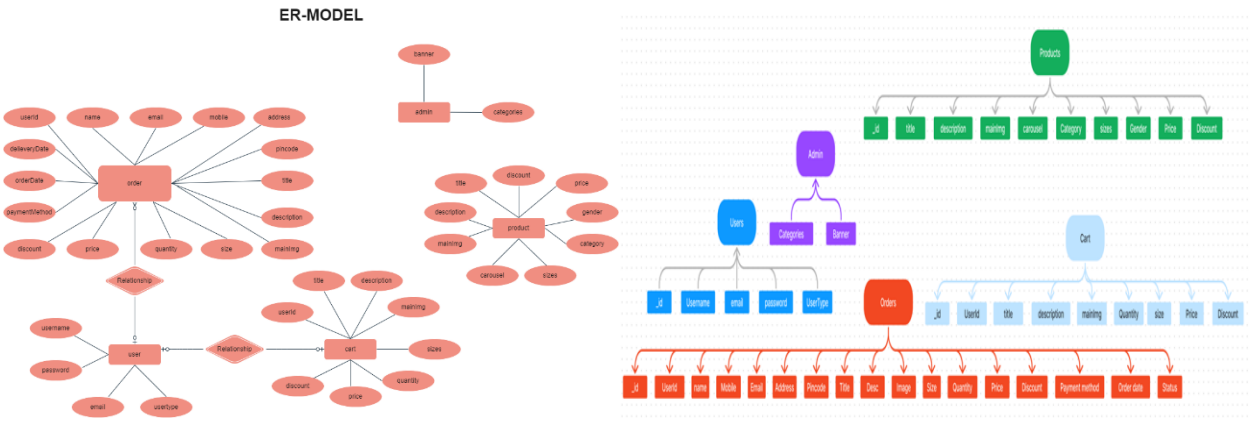
The **ShopEZ platform** is built using the **MERN stack**, which combines MongoDB for the database, Express.js for the server-side framework, React.js for the frontend interface, and Node.js as the backend runtime environment. The application follows a **modular structure** separating concerns across frontend components, backend routes, and database models.

- **Frontend**: Developed with React.js, it uses reusable components, client-side routing, and responsive design for a smooth user experience.
- **Backend**: Powered by Express.js and Node.js, it handles API routing, user authentication, and communication with the database.
- **Database**: MongoDB stores all product, user, and order data in a flexible, JSON-like format using Mongoose models.

- **Authentication**: Implements secure login/register functionality using JWT (JSON Web Tokens) and bcrypt for password hashing.
- **Admin Panel**: Accessible only to admins for managing products, users, and orders efficiently.

This architecture ensures a **scalable**, **maintainable**, and **full-featured** e-commerce application suitable for real-world use.

ER DIAGRAM



The ShopEZ ER-diagram represents the entities and relationships involved in an e-commerce system. It illustrates how users, products, cart, and orders are interconnected. Here is a breakdown of the entities and their relationships:
USER: Represents the individuals or entities who are registered in the platform.
Admin: Represents a collection with important details such as Banner image and Categories.
Products: Represents a collection of all the products available in the platform.
Cart: This collection stores all the products that are added to the cart by users. Here, the elements in the cart are differentiated by the user Id.
Orders: This collection stores all the orders that are made by the users in the platform.

SETUP PHASE

PRE-REQUISITES:

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

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/

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 complaints, status of the complaints, 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.

• 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:

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

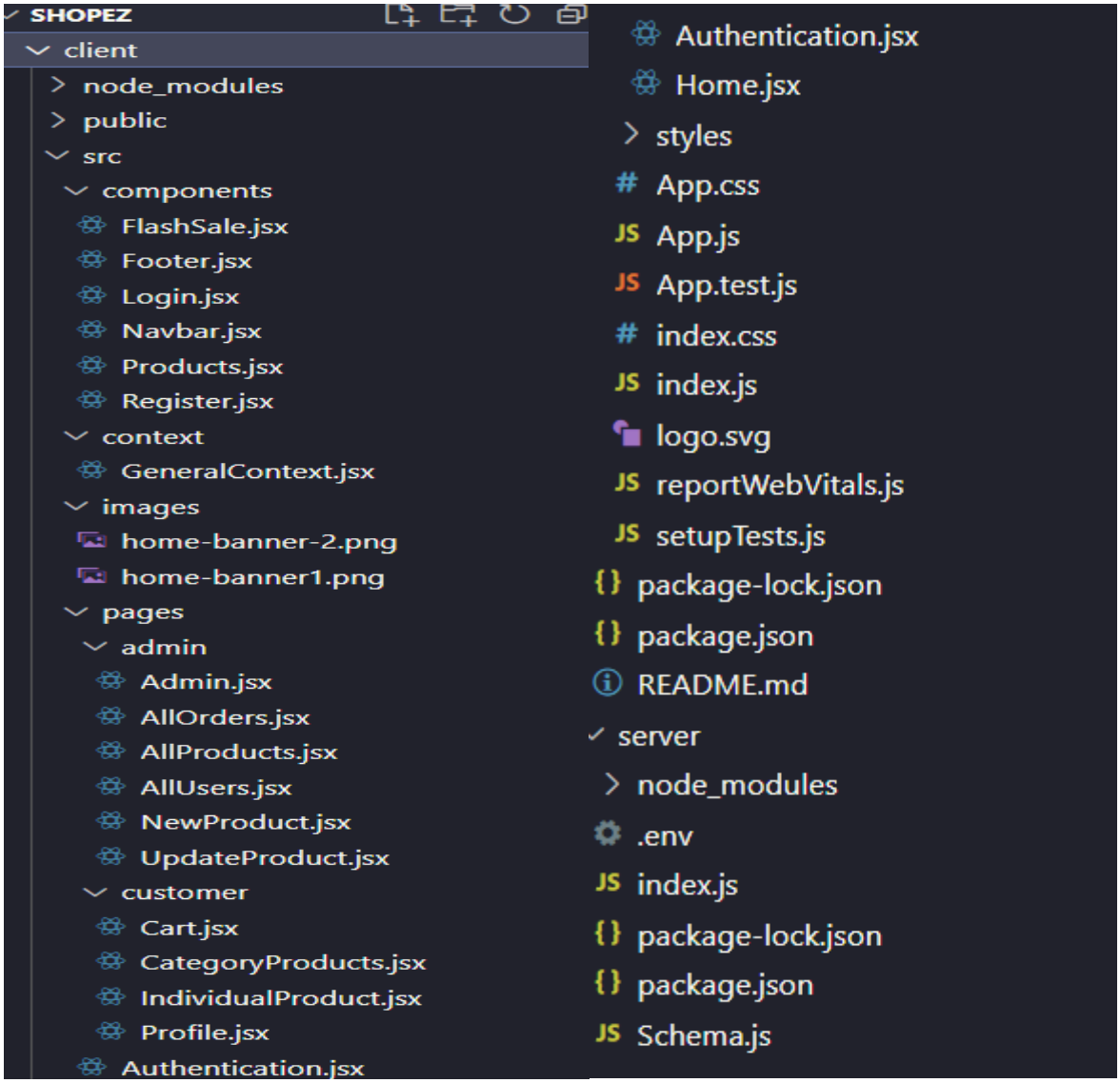
git clone: https://github.com/tanusreep8/ShopEZ.git

Install Dependencies:

• Navigate into the cloned repository directory:
cd complaint-registery
• Install the required dependencies by running the following commands:
cd client
npm install
cd server
npm install
Start the Development Server:
• To start the development server, execute the following command:
npm start
• The SHOPEZ: One-stop shop for online purchases app will be accessible at http://localhost:3001

You have successfully installed and set up the SHOPEZ: One-stop shop for online purchases app on your local machine. You can now proceed with further customization, development, and testing as needed.

PROJECT STRUCTURE:



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

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

APPLICATION FLOW:

1. User Registration / Login
    - New users can sign up and existing users can log in.
    - JWT tokens are generated for authentication and authorization.

2. Home Page Display
    - Users are presented with featured products and categories.
    - Navigation options guide users to explore products.
3. Product Browsing
    - Users can view all products or filter by category, search by name, or sort by price.
    - Each product has a detailed view with images, price, and description.
4. Add to Cart
    - Users can add products to their cart, update quantities, or remove items.
    - Cart state is maintained for each logged-in user.
5. Checkout Process
    - Users proceed to checkout where they enter shipping details.
    - An order summary is shown before confirming the order.
6. Order Placement
    - Once confirmed, the order is saved in the database.
    - A success page is shown, and an invoice is generated.
7. Order History & Tracking
    - Users can view all their past orders and track current order status (e.g., pending, shipped).
8. Admin Panel Access
    - Admins log in separately and access a dashboard with metrics.
    - Admins can add, update, or delete products, manage users, and view all orders.
9. Logout
    - Users and admins can securely log out, ending their session.

This flow ensures a seamless shopping experience from login to order management, while also providing full backend control through the admin panel.

API Documentation

Base URL: http://localhost:3001

General

- **GET /** – Check if server is working.
  Response: { "message": "working" }

Auth

- **POST /login**
  Body: { "type":"user|admin", "user":"string", "pwd":"string" }
- **POST /signup**
  Body: { "type":"user", "user":"string", "pwd":"string", "email":"string" }

Project Flow:

Use the code in: https://github.com/Charishmamothukuri/shopEZ
or follow the videos below for better understanding.

Milestone 1:

Project Setup and Configuration:

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

   Now, firstly create the folders for frontend and backend to write the respective code and install the essential libraries.

   - Client folders.
   - Server folders

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

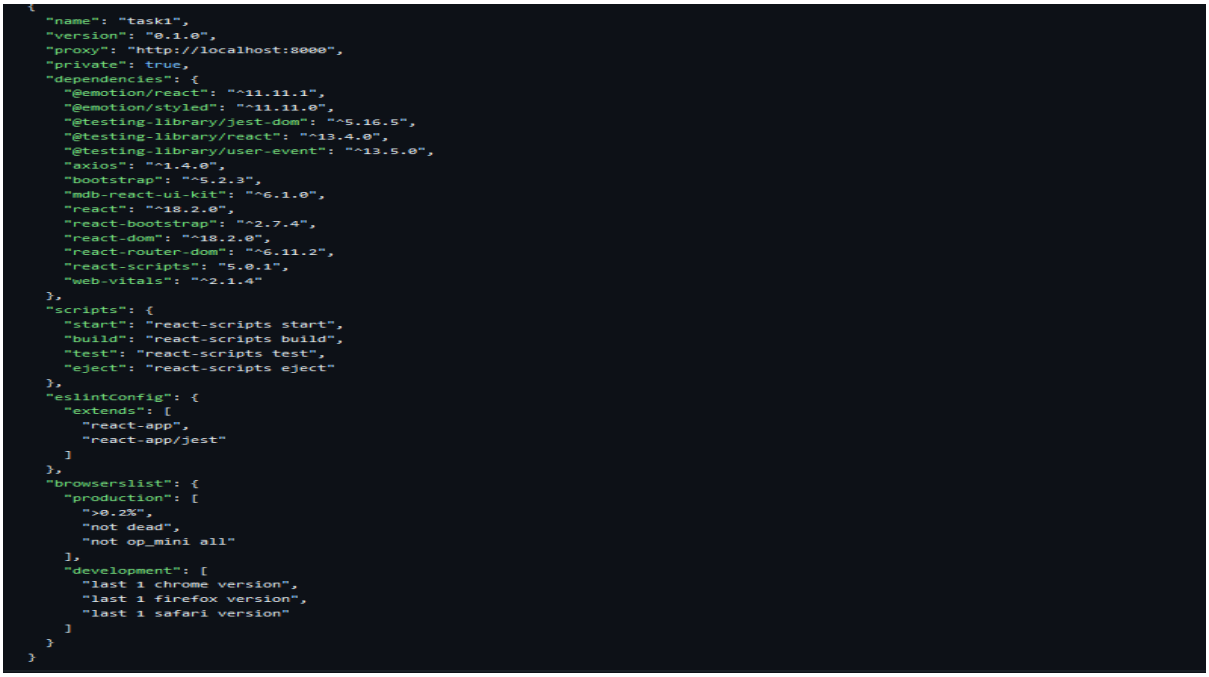
   For the backend to function well, we use the libraries mentioned in the prerequisites. Those libraries includes

   - Node.js.
   - MongoDB.
   - Bcrypt
   - Body-parser

   Also, for the frontend we use the libraries such as

   - React Js.
   - Material UI
   - Bootstrap
   - Axios

   After the installation of all the libraries, the package.json files for the frontend looks like the one mentioned below.



```
{
  "name": "task1",
  "version": "0.1.0",
  "proxy": "http://localhost:8000",
  "private": true,
  "dependencies": {
    "@emotion/react": "^11.11.1",
    "@emotion/styled": "^11.11.0",
    "@testing-library/jest-dom": "^5.16.5",
    "@testing-library/react": "^13.4.0",
    "@testing-library/user-event": "^13.5.0",
    "axios": "^1.4.0",
    "bootstrap": "^5.2.3",
    "mdb-react-ui-kit": "^6.1.0",
    "react": "^18.2.0",
    "react-bootstrap": "^2.7.4",
    "react-dom": "^18.2.0",
    "react-router-dom": "^6.11.2",
    "react-scripts": "5.0.1",
    "web-vitals": "^2.1.4"
  },
  "scripts": {
    "start": "react-scripts start",
    "build": "react-scripts build",
    "test": "react-scripts test",
    "eject": "react-scripts eject"
  },
  "eslintConfig": {
    "extends": [
      "react-app",
      "react-app/jest"
    ]
  },
  "browserslist": {
    "production": [
      ">0.2%",
      "not dead",
      "not op_mini all"
    ],
    "development": [
      "last 1 chrome version",
      "last 1 firefox version",
      "last 1 safari version"
    ]
  }
}
```

After the installation of all the libraries, the package.json files for the backend looks like the one mentioned below.

```
{
  "name": "backend",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "start": "nodemon index.js"
  },
  "keywords": [],
  "author": "",
  "license": "ISC",
  "dependencies": {
    "bcrypt": "^5.1.0",
    "cors": "^2.8.5",
    "express": "^4.18.2",
    "express-session": "^1.17.3",
    "mongoose": "^7.1.1",
    "nodemon": "^2.0.22"
  }
}
```

Milestone 2:

Backend Development:

- **Set Up Project Structure:**
    - Create a new directory for your project and set up a package.json file using npm initcommand.
    - Install necessary dependencies such as Express.js, Mongoose, and other requiredpackages.

- **Set Up Project Structure:**
    - Create a new directory for your project and set up a package.json file using npm init command.
    - Install necessary dependencies such as Express.js, Mongoose, and other requiredpackages.
- **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 forhandling cross-origin requests.
- **Define API Routes:**
    - Create separate route files for different API functionalities such as authentication, stock actions, and transactions.
    - Implement route handlers using Express.js to handle requests and interact with thedatabase.
- **Implement Data Models:**
    - Define Mongoose schemas for the different data entities like Bank, users, transactions, deposits and loans.
    - Create corresponding Mongoose models to interact with the MongoDB database.
    - Implement CRUD operations (Create, Read, Update, Delete) for each model toperform database operations.
- **User Authentication:**
    - Implement user authentication using strategies like JSON Web Tokens (JWT) orsession-based authentication.
    - Create routes and middleware for user registration, login, and logout.
    - Set up authentication middleware to protect routes that require user authentication.

- **Handle new transactions:**
    - Allow users to make transactions to other users using the user's account id.
    - Update the transactions and account balance dynamically in real-time.
- **Admin Functionality:**

- Implement routes and controllers specific to admin functionalities such as fetching all the data regarding users, transactions, stocks and orders.

- **Error Handling:**
  - Implement error handling middleware to catch and handle any errors that occurduring the API requests.
  - Return appropriate error responses with relevant error messages and HTTP statuscodes.

Milestone 3:

Database Development :

This milestone focuses on designing and implementing the core database schemas required to support **user management**, **order processing**, and **administrative control** within the ShopEZ platform. **MongoDB** serves as the primary database for storing structured and scalable e-commerce data.

1)User Schema

The **User Schema** defines the structure for all registered users of the ShopEZ platform, including both customers and administrators.

Key Fields:

- **username** – Name of the user
- **email** – User's unique email address
- **password** – Encrypted password for login security
- **role** – User role: either "user" or "admin"

Notes:

- All users must register with a **username**, **email**, and **password**.
- By default, users are assigned the role "user".
- Admins are identified by setting the role field to "admin".
- All user data is stored in the **users** collection in MongoDB.

2)Order Schema

The **Order Schema** defines how customer orders are structured and stored in the database.

Key Fields:

- **userId** – MongoDB ObjectId referencing the user
- **username** – Name of the user placing the order
- **products** – Array of ordered products (each with id, name, and price)
- **address** – Full delivery address
- **phone** – Contact number of the customer
- **paymentMode** – Method of payment (e.g., "COD")
- **orderDate** – Timestamp of when the order was placed

Notes:

- Orders are linked to users via the **userId** field.
- All order records are stored in the **orders** collection in MongoDB.

3)Admin Dashboard Access *(No New Schema)*

No separate schema is needed for admins. They use the same **User Schema** with the role field set to "admin".

Admin Capabilities:

- View all registered users
- View all placed orders
- See which user placed which product order

Admins can manage platform data securely through a protected dashboard interface.

4)Future Extension – Chat or Complaint Support *(Optional)*

Though not implemented in the current version, the system design supports future integration of a **chat** or **complaint support module**.

Suggested Schema Fields:

- **userId** – ID of the user raising the issue
- **productId** *(optional)* – Product involved (if applicable)
- **message** – Complaint or support message
- **timestamp** – Time when the message was submitted

These messages can be stored in a new **chats** or **complaints** collection to enhance customer service functionality.

Milestone 4:

1. Setup React Application

The first step in frontend development is creating a solid foundation using React.

Key Activities:

- React App Initialization: Create the base React application using Create React App to provide the necessary project scaffolding.
- Library Installation: Install essential libraries such as react-router-dom for routing, axios for API calls, and state management tools like Redux Toolkit.
- Project Structure Setup: Organize the directory structure for scalability and maintainability.

2. Design UI Components

This phase focuses on building and styling the visible parts of the platform that users interact with.

Key Activities:

- Reusable Components: Develop modular components such as buttons, forms, cards, product displays, and navigation bars.
- Layout and Styling: Implement a consistent visual style using CSS, component libraries like Material UI, or custom design systems to ensure branding and user-friendliness.
- Navigation System: Use React Router to build an intuitive and seamless navigation experience between key sections, such as:
    - Home page
    - Product details

- Categories
- Login/Register
- Cart and Checkout

3. Implement Frontend Logic

The final stage bridges the gap between the UI and backend data, allowing for interactivity and dynamic behavior.

Key Activities:

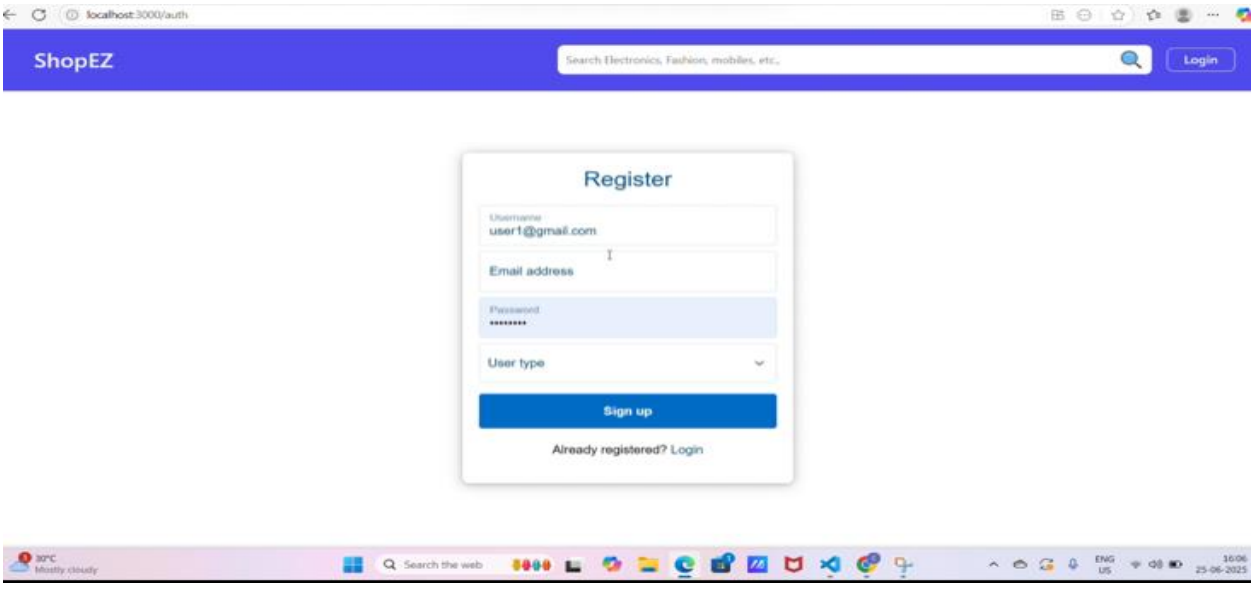
- API Integration: Connect frontend components to backend RESTful API endpoints to fetch or send data (e.g., retrieving product lists, submitting login forms, managing the cart).
- State Management: Use Redux Toolkit to manage and share data across components, including authentication state, product listings, and cart items.
- Dynamic Rendering: Implement data binding and conditional rendering to display real-time content based on user actions and API responses.
- Error Handling and Feedback: Display loading indicators, success messages, and error alerts to enhance user experience.

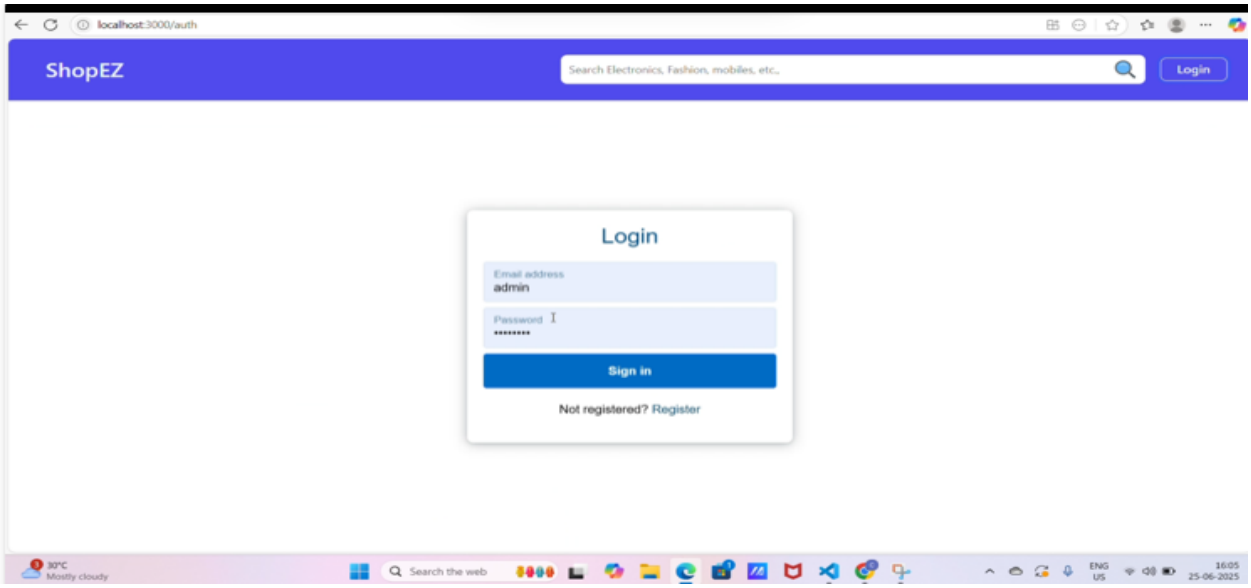
Milestone 5:

Project Implementation:

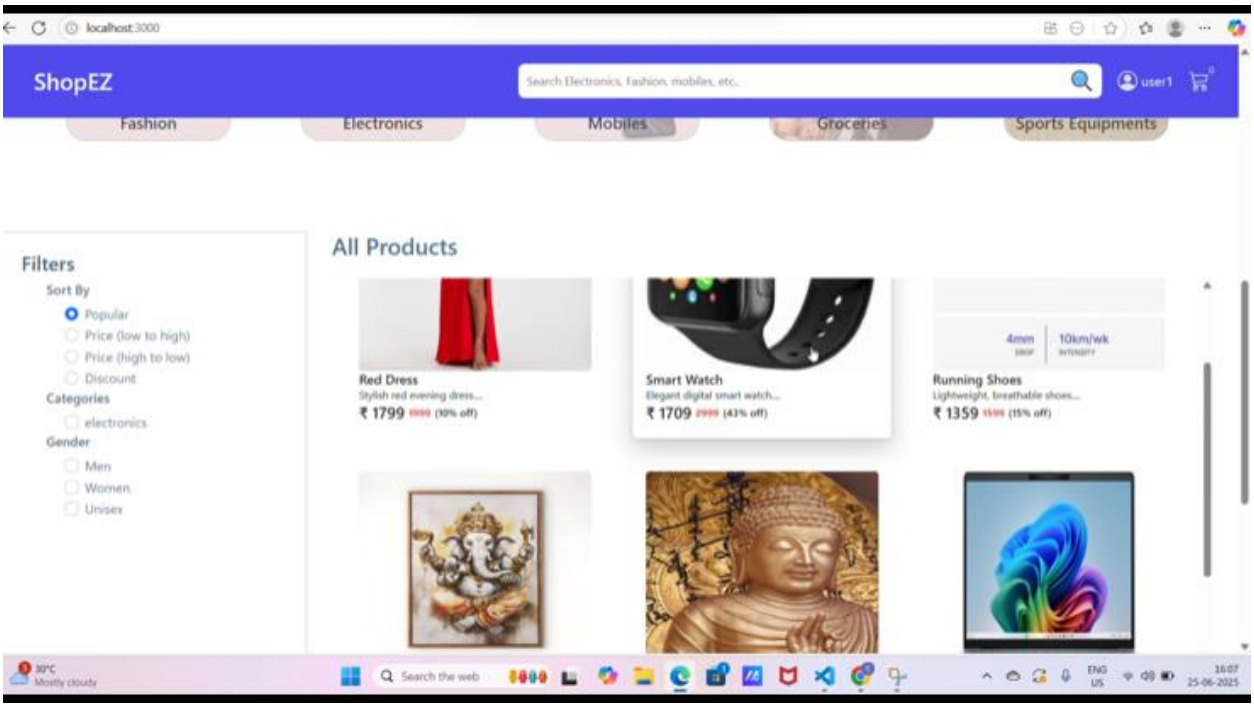
On completing the development part, we then run the application one last time to verify all the functionalities and look for any bugs in it. The user interface of the application looks a bit like the one's provided below.

- User Login Page

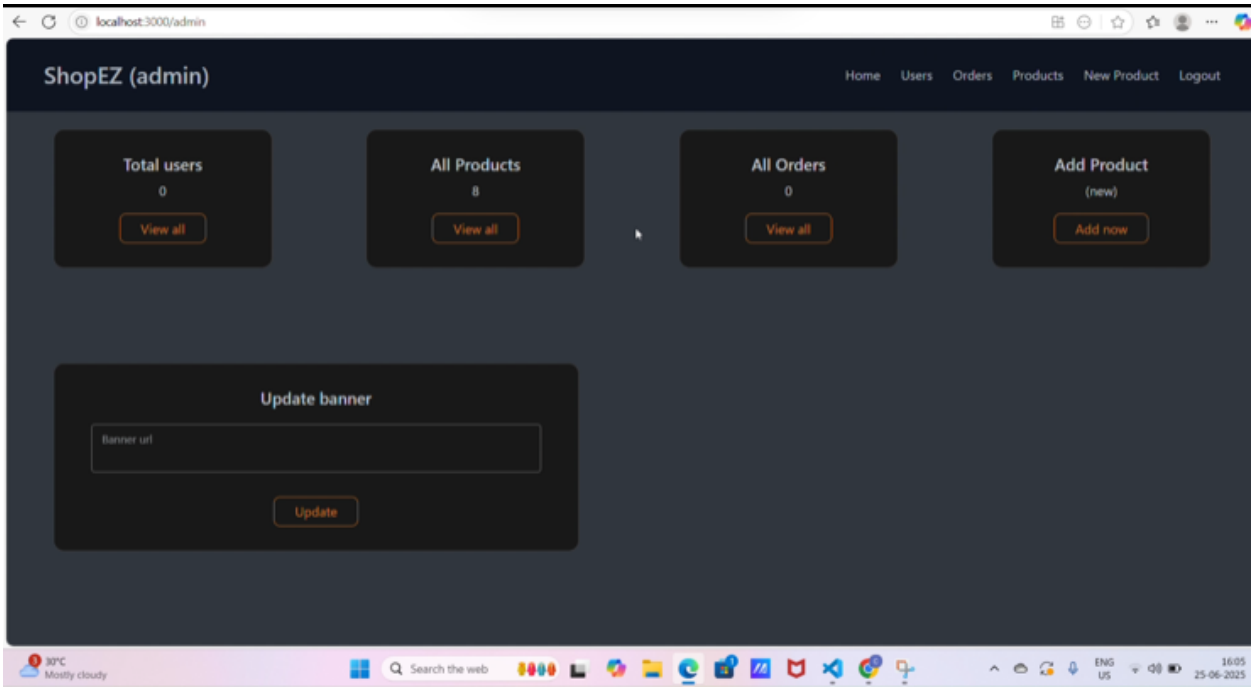




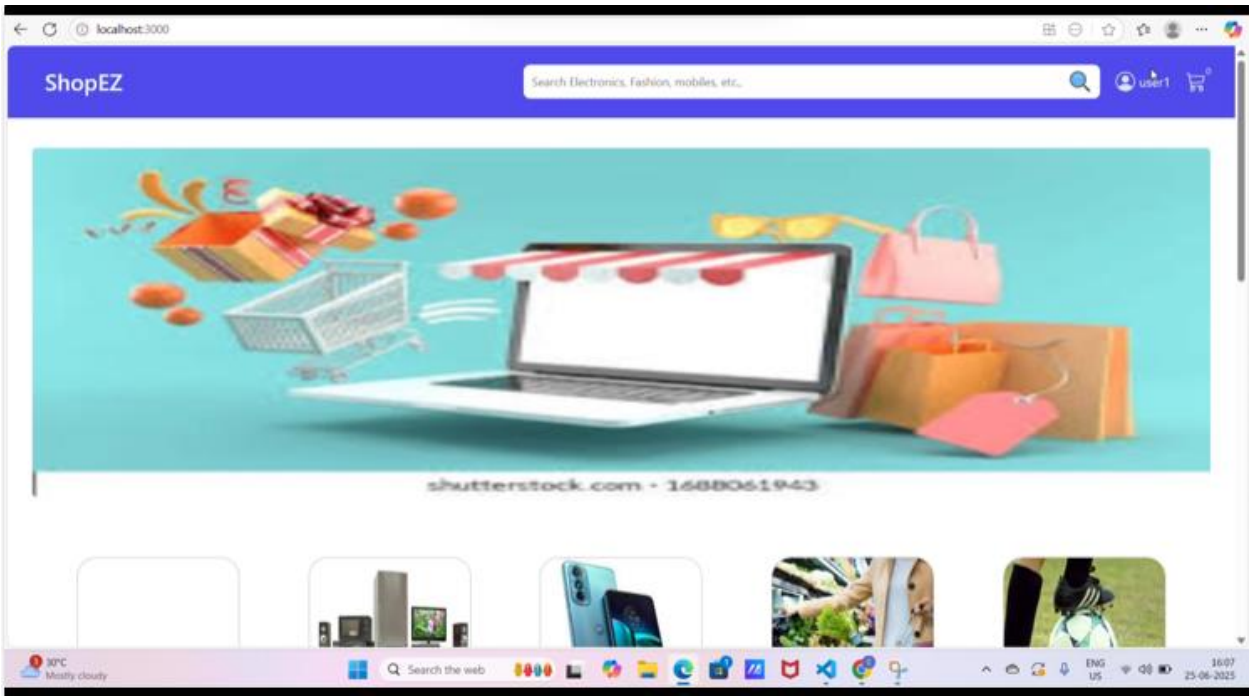
- Product Catalog



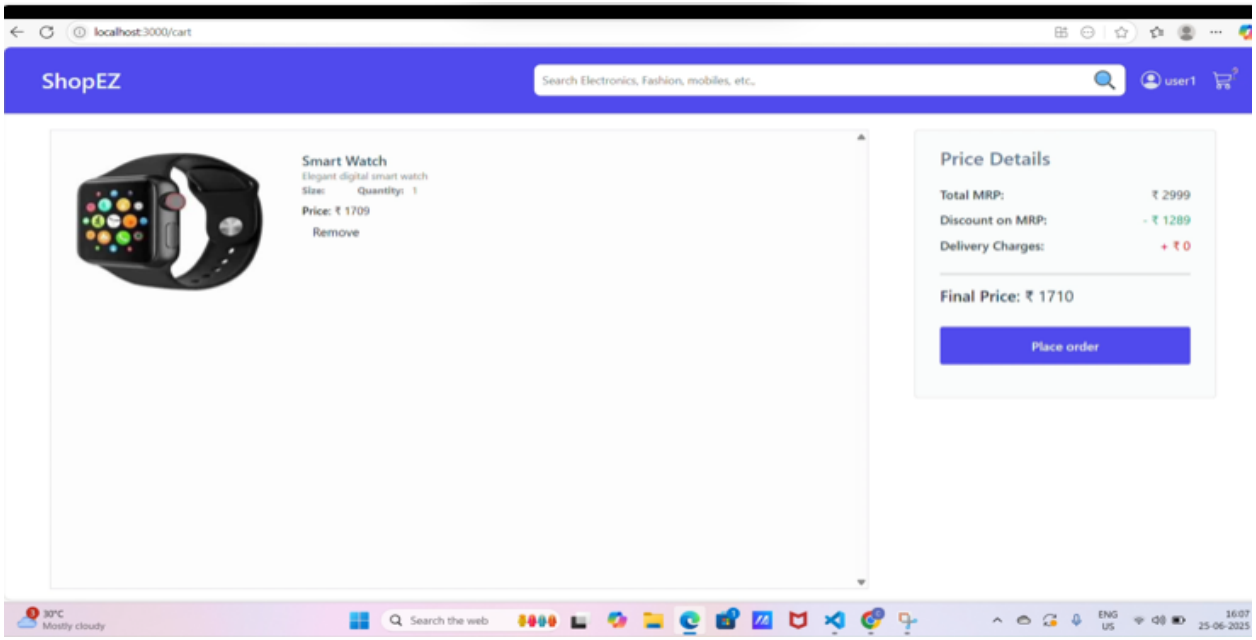
- Admin Dashboard



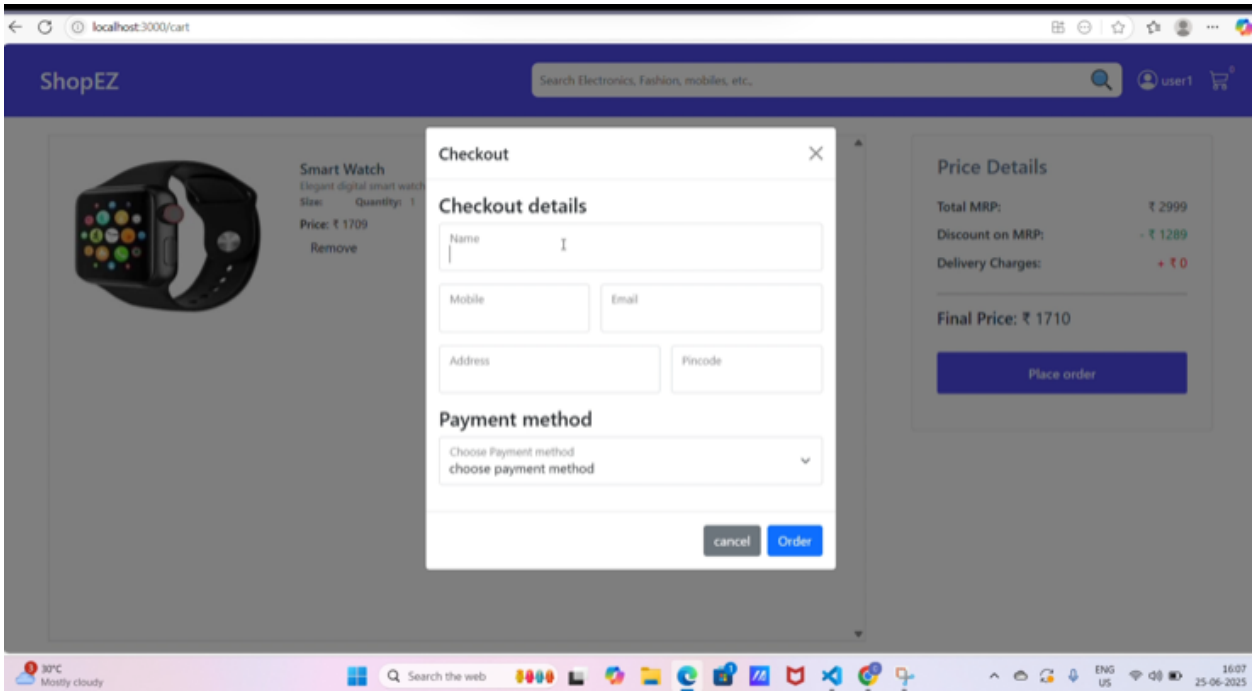
- Landing Page



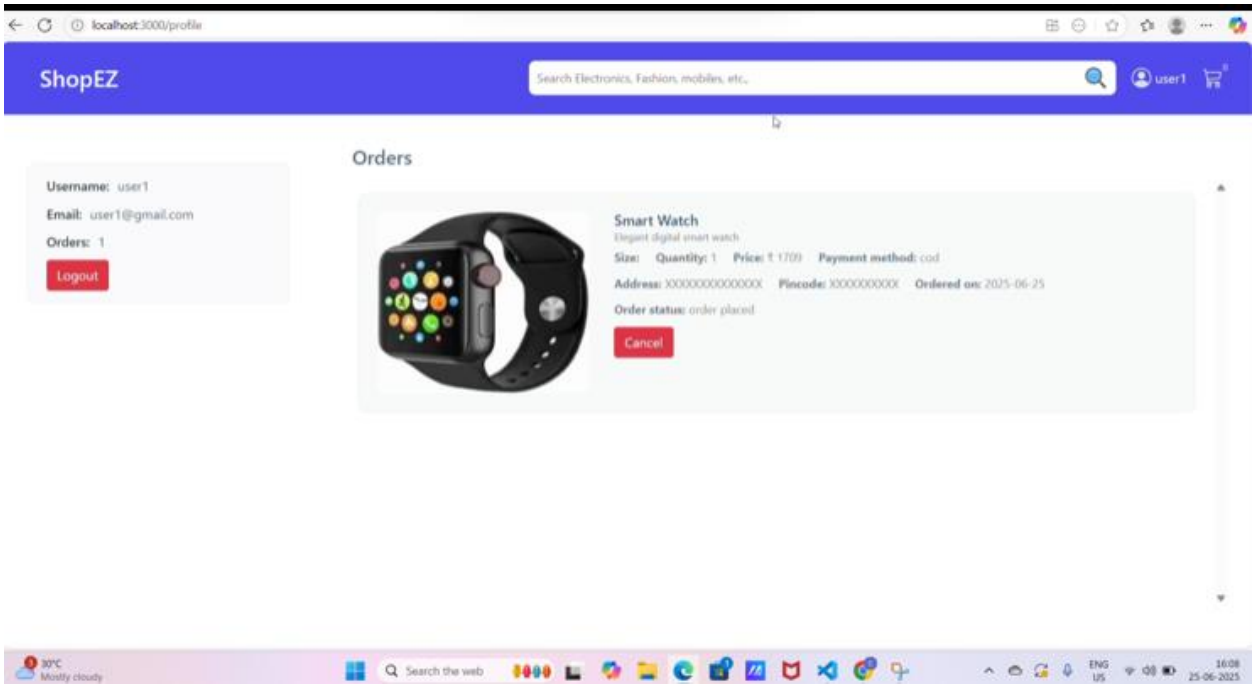
- Cart



Checkout : payment details



Orders :



REFERENCE LINKS :

DEMO LINK VIDEO :

https://youtu.be/aLXn_ay4bBM?si=bscPiYZvwH8h6Sea

GITHUB CODES :

https://github.com/Charishmamothukuri/shopEZ