**ShopEZ: E-commerce**

**Application**

### 1. INTRODUCTION:

PROJECT TITLE: SHOPEZ ONE- STOP SHOP FOR ONLINE PURCHASE

ShopEZ is your one-stop destination for effortless online shopping. With a user-friendly interface and a comprehensive product catalog, finding the perfect items has never been easier. Seamlessly navigate through detailed product descriptions, customer reviews, and available discounts to make informed decisions. Enjoy a secure checkout process and receive instant order confirmation. For sellers, our robust dashboard provides efficient order management and insightful analytics to drive business growth. Experience the future of online shopping with ShopEZ today.

Seamless Checkout Process

Effortless Product Discovery

Personalized Shopping Experience

Efficient Order Management for Sellers

### TEAM MEMBERS:

1. P.B.S.S KARTHIKEYINI (22MH1A05D4)

2. K.HARISH (22MH1A05B1)

3. G.B. CHATURYA (22MH1A0519)

4. DIPENDRA GADERI(22MH1A0515)

### 2. PROJECT OVERVIEW

##### Purpose:

ShopSmart is an online shopping web application that allows users to browse, search, and

purchase from the comfort of their homes. The goal is to simplify shopping with user

friendly and fast delivery service.

#### **F**EATURES**:**

1. **Comprehensive Product Catalog:** ShopEZ boasts an extensive catalog of products, offering a diverse range of items and options for shoppers. You can effortlessly explore and discover various products, complete with detailed descriptions, customer reviews, pricing, and available discounts, to find the perfect items for your needs.

**2. Shop Now Button:** Each product listing features a convenient "Shop Now" button. When you find a product that aligns with your preferences, simply click on the button to initiate the purchasing process.

3. **Order Details Page**: Upon clicking the "Shop Now" button, you will be directed to an order details page. Here, you can provide relevant information such as your shipping address, preferred payment method, and any specific product requirements.

4. **Secure and Efficient Checkout Process:** ShopEZ guarantees a secure and efficient checkout process. Your personal information will be handled with the utmost security, and we strive to make the purchasing process as swift and trouble-free as possible.

5. **Order Confirmation and Details:** After successfully placing an order, you will receive a confirmation notification. Subsequently, you will be directed to an order details page, where you can review all pertinent information about your order, including shipping details, payment method, and any specific product requests you specified.

### 3.ARCHITECTURE:

#### 

##### Frontend:

Build using [React.js](http://react.js) the frontend includes user interface components such as User

Admin Authentication and an Admin Dashboard.

* create interactive and reusable UI components, making it easier to build dynamic and responsive web applications.

### 

###### Components:

* Navbar, Footer: Shared layout elements
* Home, ProductDetails, AllProducts: Page-level components
* ProductCard, CategoryCard: Reusable UI pieces
* Filters, SortOptions: Dynamic control panels

**Page Layer:**

The pages/ folder contains route-level components organized by feature or user role:

* Admin: Admin dashboard pages (manage users, orders, products)
* customer: Customer pages (product browsing, checkout)
* Home.jsx: Landing page combining categories, banners, and product previews
* Authentication.jsx: Login and registration UI.

##### Backend:

The backend consisting of API endpoints for Users, Orders, Products, etc.,It also

includes Admin Authentication and an Admin Dashboard.

* Developed with Node.js and Express.js, it handles user authentication, product management, and order processing.

###### User Authentication:

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

• Set up authentication middleware to protect routes that require user

authentication.

###### 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.

**Implement Data Models:**

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

and models.

• Create corresponding Mongoose models to interact with the MongoDB

database.

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

perform database operations.

#### Database:

The Database section represents the database that stores collections for Users, cart,

Orders and Product.

* Uses MongoDB to store data such as user info, product details, orders, and admin

Credentials.

###### 1. Configure MongoDB

● Install Mongoose

. ● Create database connection.

● Create Schemas & Models.

## ER DIAGRAM:

## 

**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.

### 4.SETUP INSTRUCTIONS:

### PREREQUISITES:

1. Node.js and npm: Node.js is required to run JavaScript on the server side.

2. MongoDB : Install MongoDB locally or use a cloud-based MongoDB service .

###### INSTALLATION:

###### 

**Clone the repository:**

• Open your terminal or command prompt.

• Navigate to the directory where you want to store the-commerce

• Execute the following command to clone the repository

**Git clone**: https://github.com/username/shopEZ--e-commerce-MERN

##### Install Dependencies:

• Navigate into the cloned repository directory:

**cd ShopEZ—e-commerce-App-MERN**

• Install the required dependencies by running the following command

cd client

npm install

* Axios
* Bootstrap
* React
* React-router-dom
* React-icons
* React-scripts
* web-vitals

**Start the Development Server:**

cd…/server

npm install

* Bcrypt
* Body-parser
* Cors
* Dontev
* Express
* mongoose

###### Environment variables (env.file) :

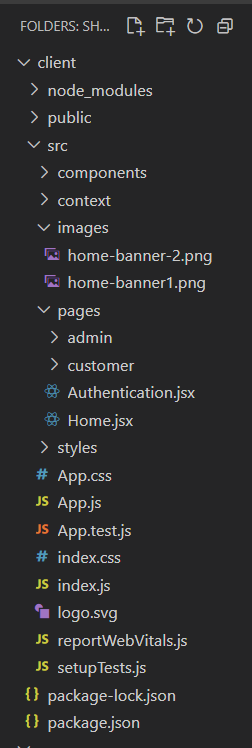
You can change the port configuration in the .env file if needed.

MONGODB\_URI=mongodb://localhost:27017/ecommerce

PORT=6001

##### 5.FOLDER STRUCTURE:

* Client Folder:

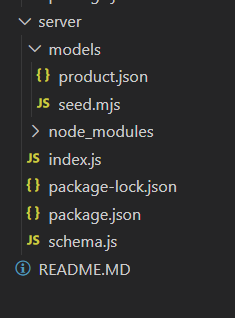


This structure assumes a React app and follows a modular approach. Here's a brief explanation of the main directories and files:

• src/components: Contains components related to the application such as, register, login, home, etc.,

• src/pages has the files for all the pages in the application.

* Server Folder:



###### Setup express server:

[Node.js](http://node.js) is a runtime environment that allows you to run javascript code on the server .

• Create index.js file.

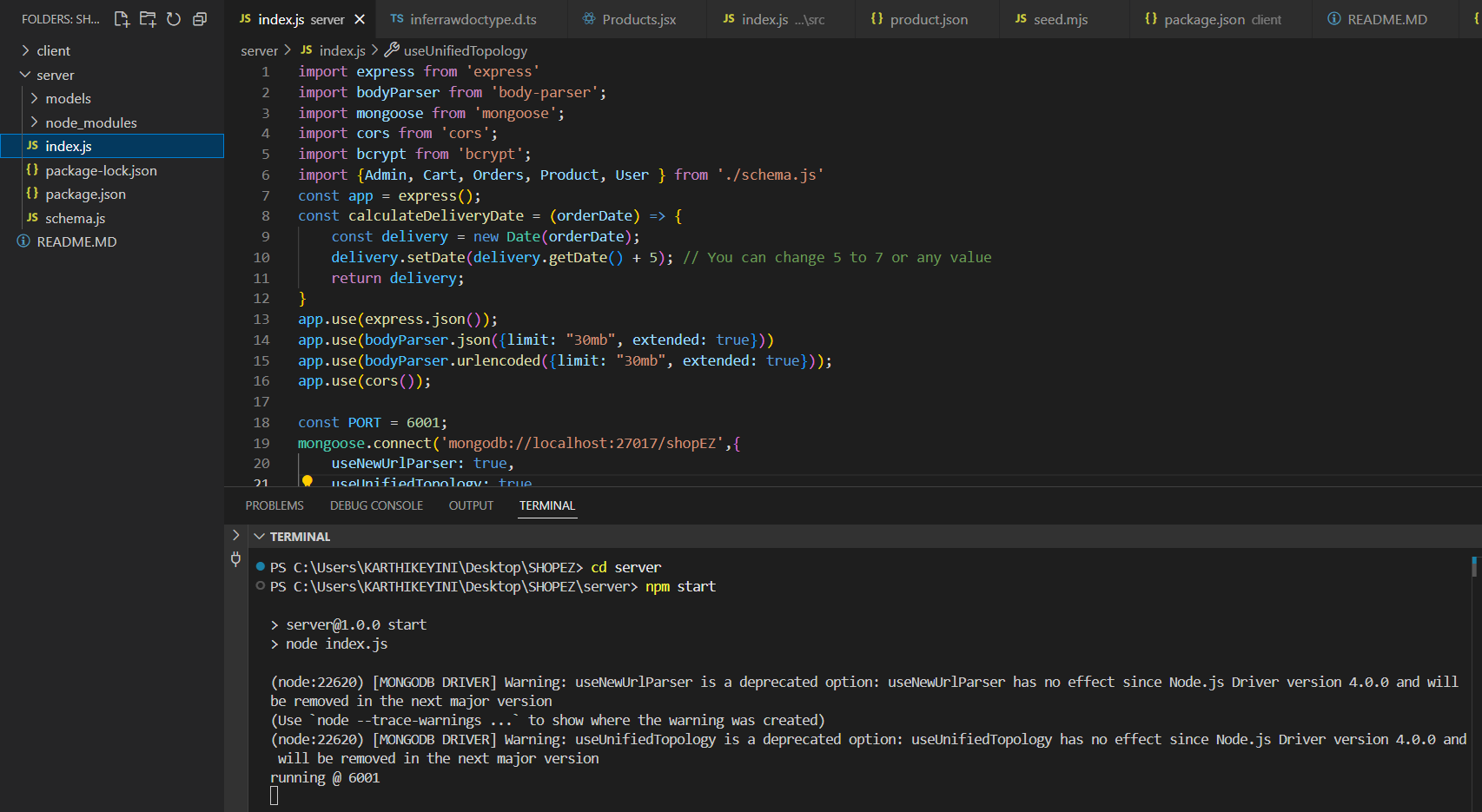
• Create an express server on your desired port number.

• Define API’s

###### 6.RUNNING THE APPLICATION:

Backend: cd server

npm start



Frontend: cd client

npm start

##### 7.API DOCUMENTATION:

##### USER API:

**Register User**

* Endpoint: /register
* Method: POST
* Body:

{

"username": "John",

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

"usertype": "customer",

"password": "123456"

}

#### Login User:

* Endpoint: /login
* Method: POST
* Body:

{

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

"password": "123456"

}

#### Fetch All Users:

* Endpoint: /fetch-users
* Method: GET

### Product APIs

Fetch All Products

* Endpoint: /fetch-products
* Method: GET

#### Fetch Product By ID

* Endpoint: /fetch-product-details/:id
* Method: GET

#### Add New Product

* Endpoint: /add-new-product
* Method: POST

#### Update Product

* Endpoint: /update-product/:id
* Method: PUT

### Order APIs

Place Order (Single Product)

* Endpoint: /buy-product
* Method: POST

###### Place Order (From Cart)

* Endpoint: /place-cart-order
* Method: POST

#### Fetch All Orders

* Endpoint: /fetch-orders
* Method: GET

#### Cancel Order

* Endpoint: /cancel-order
* Method: PUT

#### Update Order Status

* Endpoint: /update-order-status
* Method: PUT

### Cart APIs

#### Fetch Cart Items

* Endpoint: /fetch-cart
* Method: GET

#### Add Item to Cart

* Endpoint: /add-to-cart
* Method: POST

#### Increase Quantity

* Endpoint: /increase-cart-quantity
* Method: PUT

#### Decrease Quantity

* Endpoint: /decrease-cart-quantity
* Method: PUT

#### Remove Item

* Endpoint: /remove-item
* Method: PUT

### Admin APIs

#### Fetch Banner

* Endpoint: /fetch-banner
* Method: GET

#### Update Banner

* Endpoint: /update-banner
* Method: POST

#### Fetch Categories

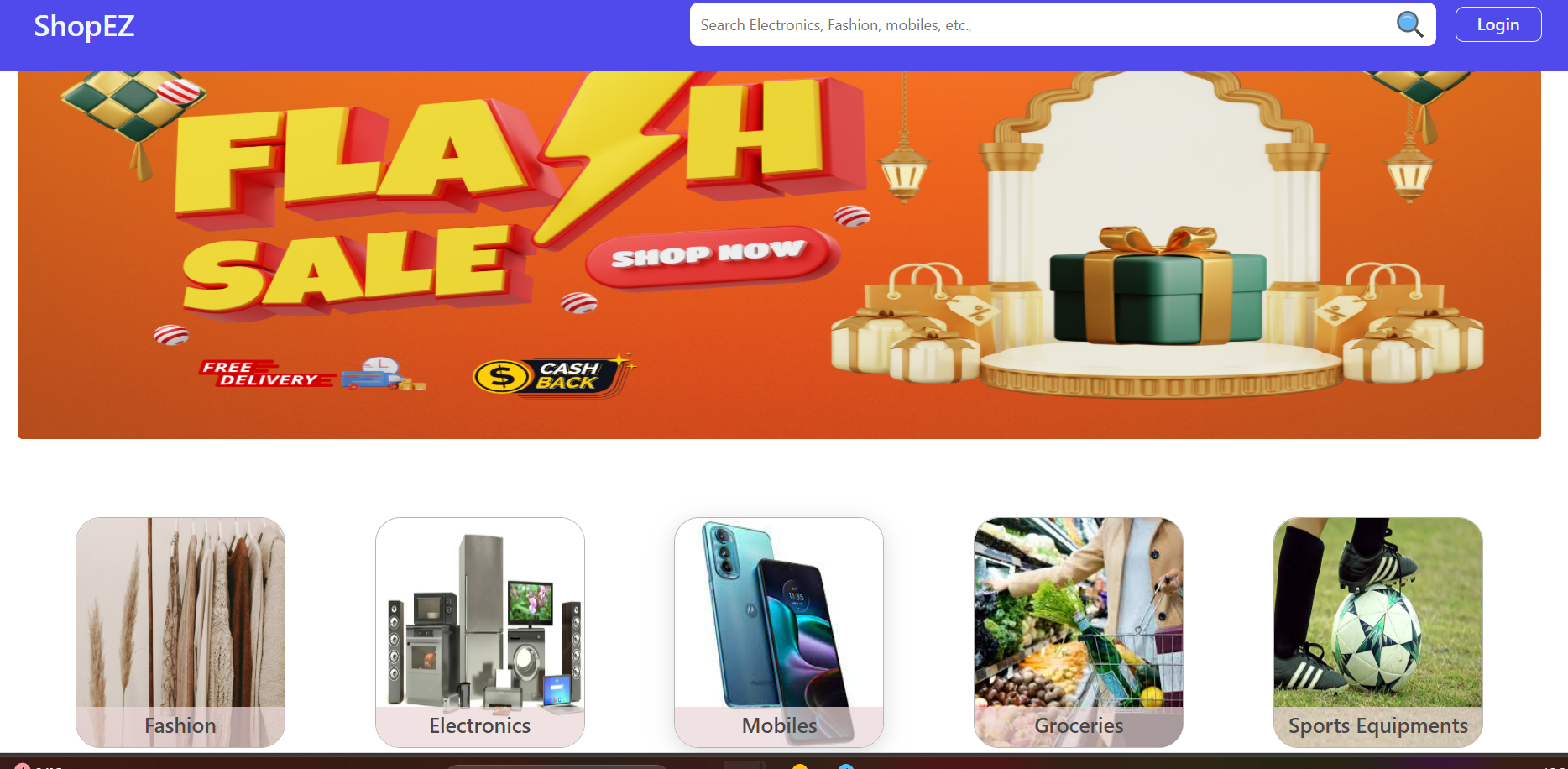
* Endpoint: /fetch-categories
* Method: GET

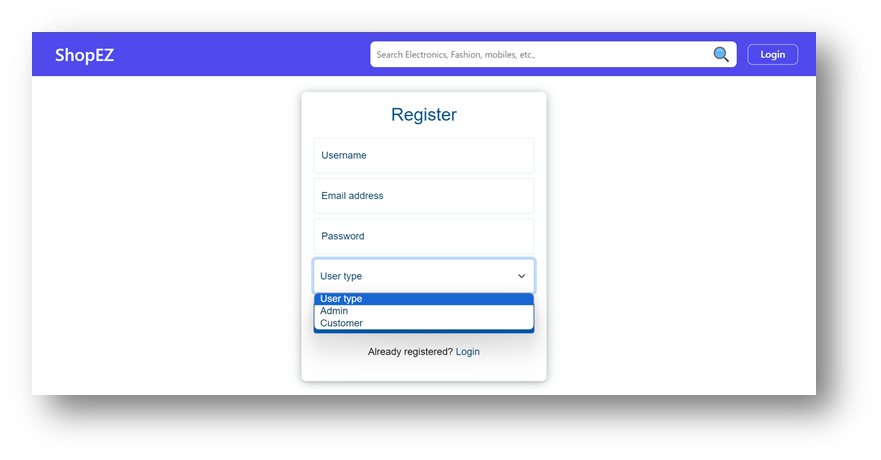
###### 8.AUTHENTICATION:

* Authentication is handled via hashed passwords using bcrypt.
* No JWT is currently used; sessions or token handling can be added in future.

**9. UI images:**

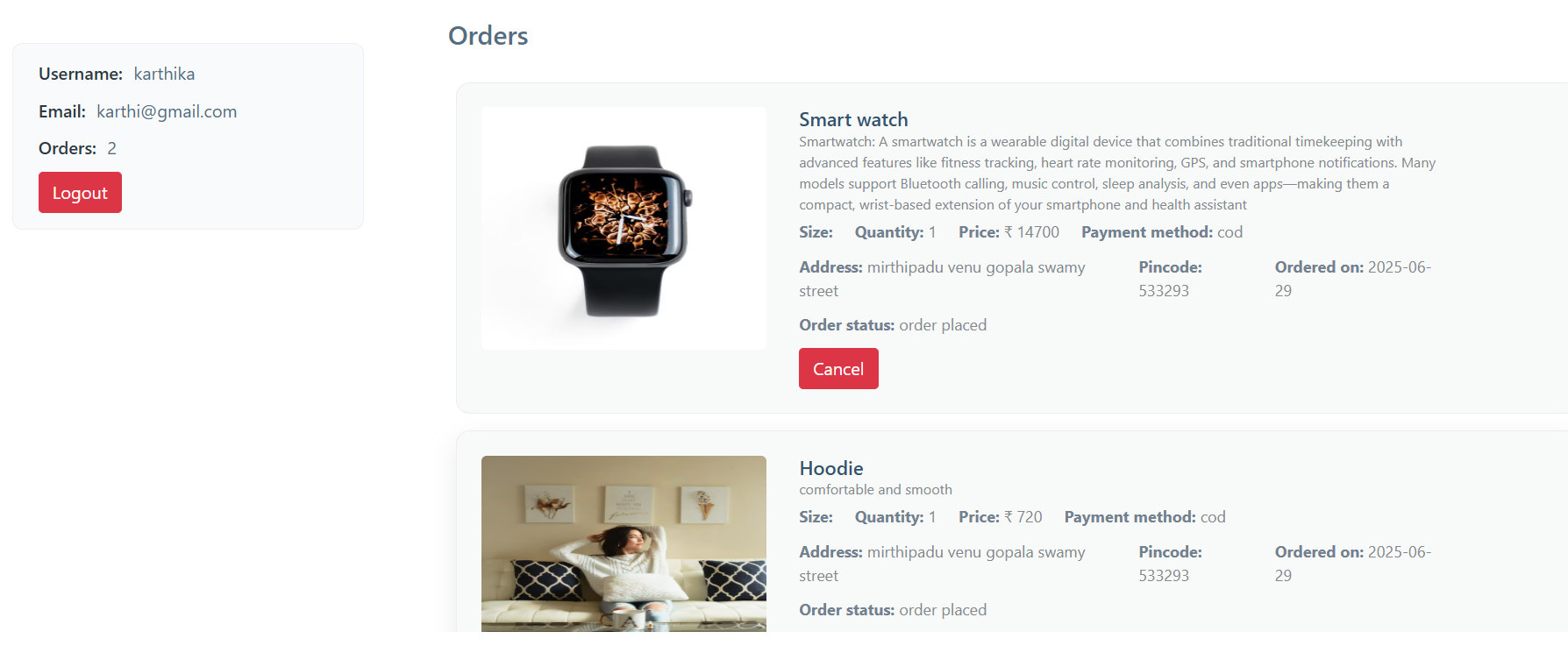
· **Landing page**



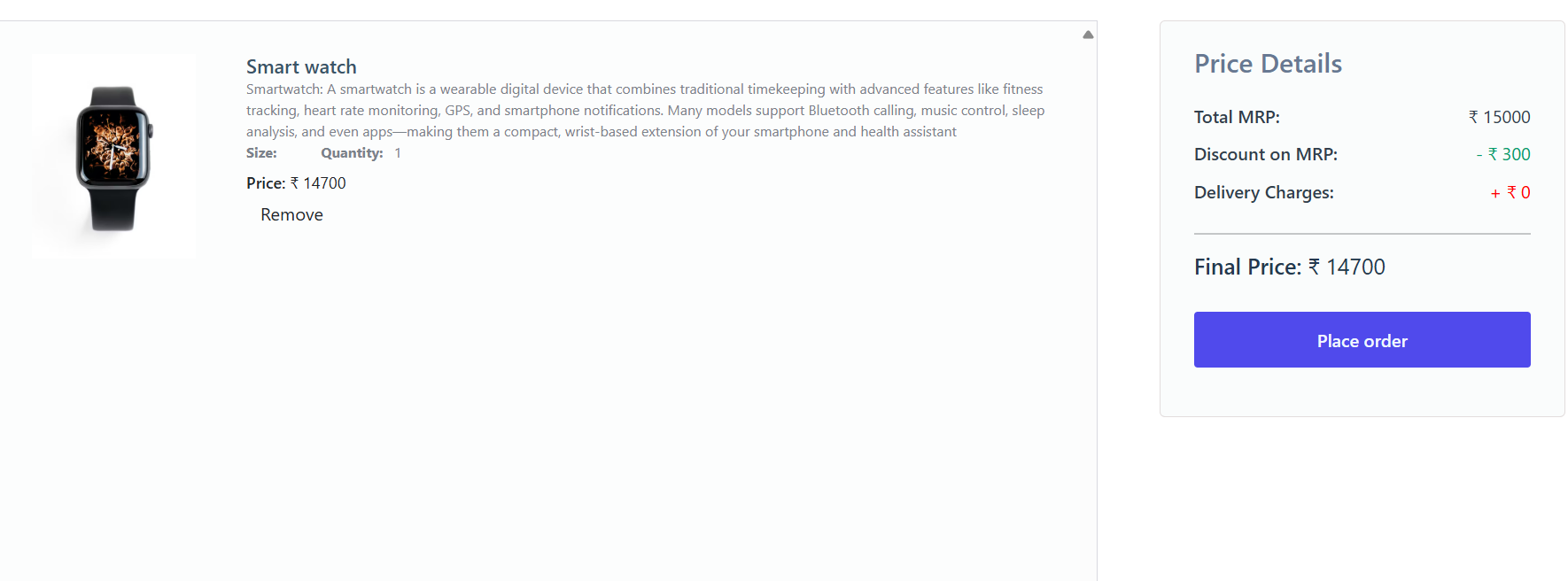
**Authentication**

·

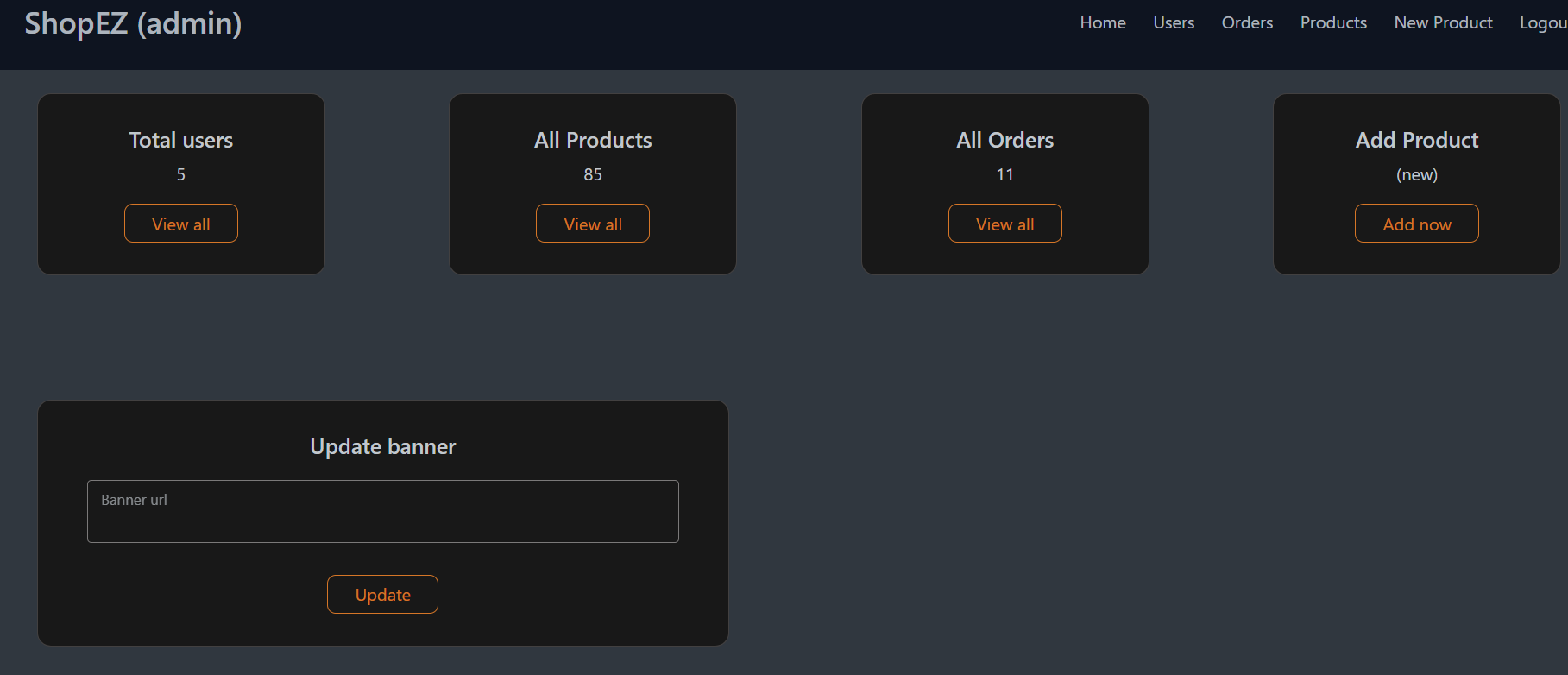
**UserProfile**



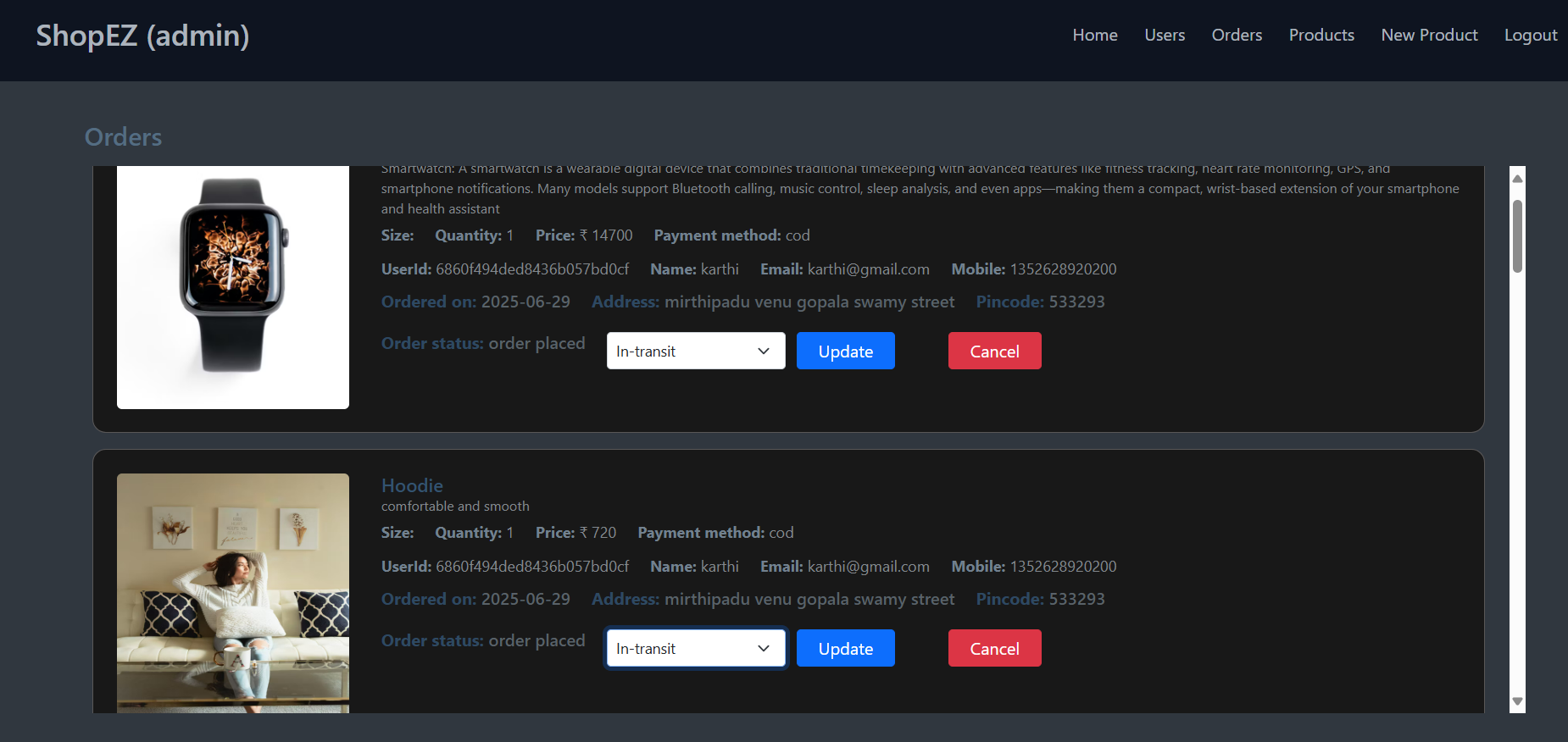
· **Cart**

****

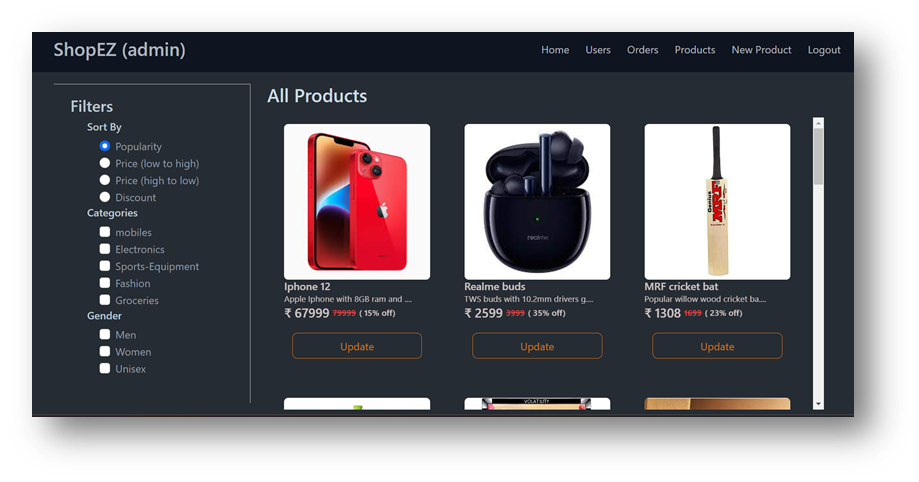
· **Admin dashboard**

****

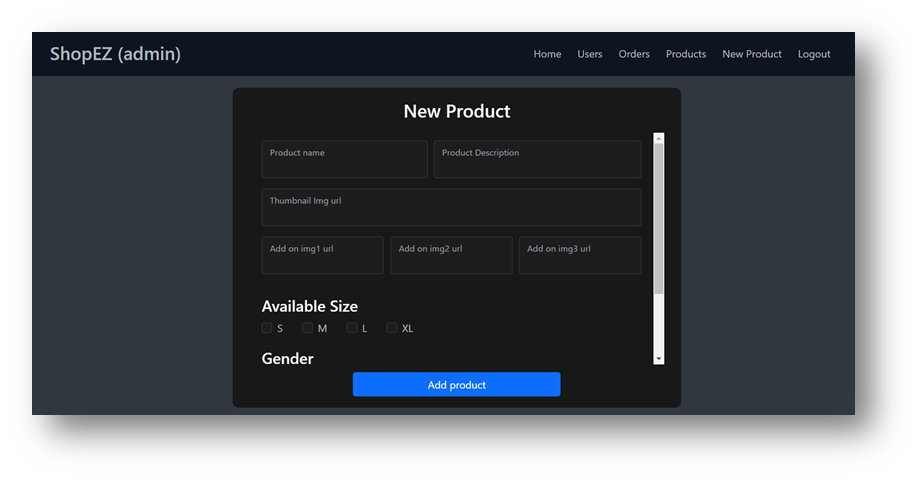
· **All Orders**

****

· **All Products**

****

· **New Product Page**

****

###### Conclusion:

The development of the ShopEz Web App using the MERN stack has successfully

provided an efficient and user-friendly platform for online shopping. This application

streamlines the entire shopping experience by allowing users to browse products, add

them to a cart, and place orders with ease. The use of MongoDB for data storage,

Express.js and Node.js for backend API services, and React.js for dynamic frontend UI

ensures that the application is responsive, scalable, and efficient. This project

demonstrates how modern full-stack technologies can be integrated to build real-time,

feature-rich e-commerce solutions tailored for needs .

###### 

###### 10. Known Issues

* Sorting does not work correctly if filters are applied simultaneously
* Some images hosted externally may fail to load if URLs break
* JWT-based authentication is not implemented yet

##### 

##### 11. Future Enhancements

* Integrate Stripe or Razorpay for payments
* Add product reviews and ratings
* Implement pagination and search
* Create mobile app version using React Native
* Add JWT-based session management
* Add inventory tracking for products