ShopEZ:One-Stop Shop for Online Purchases

INTRODUCTION

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.

In today's fast-paced digital world, consumers expect convenience, speed, and personalization when shopping online. **ShopEZ** is a dynamic full-stack e-commerce platform built using the **MERN stack (MongoDB, Express.js, React.js, Node.js)** that meets these expectations by providing a seamless and efficient online shopping experience.

ShopEZ is designed to cater to both customers and sellers. For customers, it offers **effortless product discovery** through category-based browsing, advanced search filters, and **personalized recommendations**. Users can easily explore products like fashion accessories, electronics, and more, based on their preferences and budget.

The platform ensures a **seamless checkout process**, where customers can securely place their orders, select delivery addresses, and choose from multiple payment options—all in just a few clicks. Post-purchase, users receive instant **order confirmations**, giving them confidence that their transaction was successful.

For sellers, ShopEZ provides an **efficient order management dashboard** that allows them to track orders, manage inventory, and process shipments quickly. The dashboard also offers **insightful analytics** on sales trends and customer behavior, helping sellers optimize their business strategies.

The real-world utility of ShopEZ is reflected in scenarios like Sarah's experience. As a busy professional, she leveraged ShopEZ's intuitive interface, personalized product recommendations, and fast checkout to select the perfect birthday gift for her friend Emily—showcasing how ShopEZ simplifies shopping even in time-sensitive situations.

With its user-friendly design, robust backend, and real-time data management, ShopEZ redefines the future of online shopping for both buyers and sellers.

Scenario-Based Case Study:

Scenario: Shop for online purchases

User Registration:

Sarah, a college student with a keen interest in online shopping, visits the **ShopEZ E-commerce Platform** for the first time. She creates a new user account by providing her **email address**, **mobile number**, and **password**. After successful registration, she logs in and lands on a **user-friendly home page**.

Browsing Products:

Once logged in, Sarah explores a wide variety of **product categories**, ranging from **fashion accessories** and **electronics** to **home essentials**. The homepage displays **best-selling products**, **trending deals**, and **new arrivals**.

Using **filtering and search options**, Sarah browses through the "**Fashion Accessories → Bracelets**" category to find products that match her preferences and budget.

Product Discovery and Recommendations:

As Sarah scrolls through the product list, she notices a "**Recommended for You**" section. The platform uses her browsing history and preferences to suggest a **Gold-Plated Bangle**, which catches her eye.

She clicks on the product to view **detailed descriptions**, **customer reviews**, **available discounts**, and **delivery options**.

Adding to Cart and Checkout:

Impressed with the product, Sarah adds the bangle to her **shopping cart**. She reviews her cart and proceeds to the **seamless checkout process**, where she:

- Enters **Emily's (her friend's) address** for delivery
- Selects her **preferred payment method (e.g., UPI or Credit Card)**
- Confirms her order with a simple click

Order Confirmation and Tracking:

Immediately after placing the order, Sarah receives an **email and SMS confirmation** with her **Order ID**, **estimated delivery date**, and a link to **track her shipment status**.

She feels confident knowing that her order is being processed efficiently.

Seller's Role:

Meanwhile, the seller of the Gold-Plated Bangle, **John**, logs into the **ShopEZ Seller Dashboard**.
Through the dashboard, John can:

- View all **new orders**
- Check the **customer details and shipping address**
- **Update the order status** to "Packed" and "Shipped"
- Manage **inventory** and track **sales reports** using built-in **sales analytics**

John quickly processes Sarah's order and hands it over to the courier service.

Admin Oversight:

On the administrative side, the **ShopEZ Admin** team monitors the platform's overall activity. They ensure:

- **Smooth transactions** between customers and sellers
- **Order processing and delivery tracking**
- **Product listing management**
- Handling any **customer complaints or order disputes**
- Maintaining **system performance and security**

The admin panel gives insights into **daily sales**, **user registrations**, and **order fulfillment rates**.

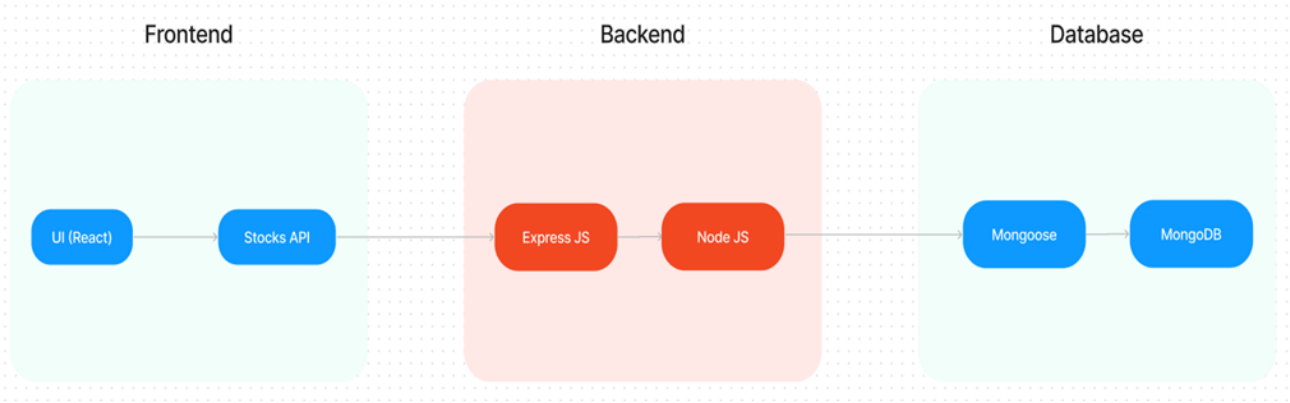
Product Delivery and Customer Satisfaction:

A few days later, Emily receives the beautifully packaged bracelet on her birthday.
Sarah is delighted that the gift arrived on time and in perfect condition.
She leaves a **positive review and 5-star rating** for the seller on ShopEZ.

Paid Services for Sellers:

John, the seller, explores **ShopEZ's premium seller features** (paid services), like **highlighting products in featured listings** and **running promotional offers**. This helps him improve visibility and increase sales.

TECHNICAL ARCHITECTURE

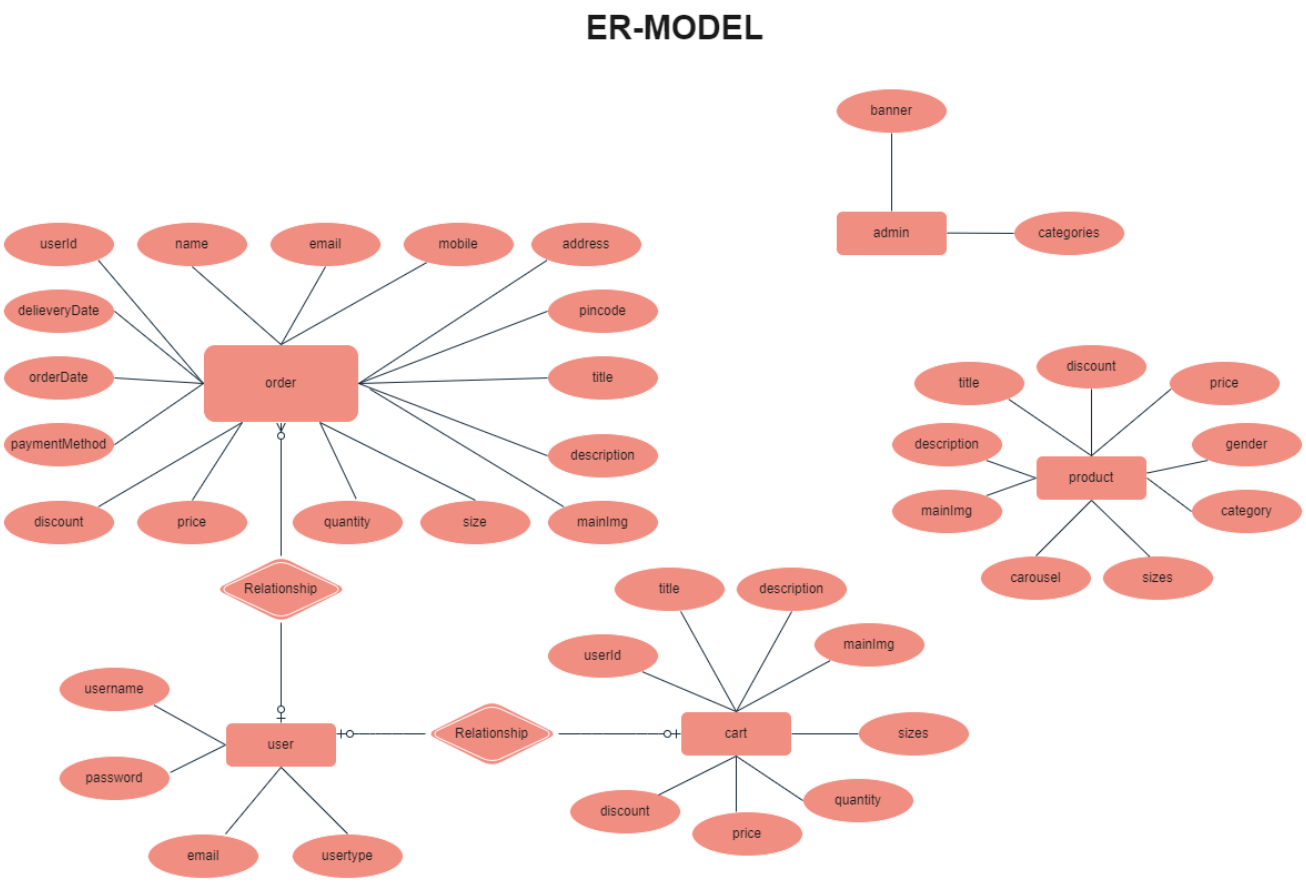


The technical architecture of OLP app follows a client-server model, where the frontend serves as the client and the backend acts as the server. The frontend encompasses not only the user interface and presentation but also incorporates the axios library to connect with backend easily by using RESTful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

The front end utilizes the bootstrap and material UI library to establish a real-time and better UI experience for any user. 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 OLP 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



Here there are 6 collections and that have their fields in

1. User Table (user):
Attributes:

- userId
- username
- password
- email
- usertype

2. Order Table (order):
Attributes:

- userId
- name
- email
- mobile
- address
- pincode
- title
- description
- mainImg
- size
- quantity
- price
- discount
- paymentMethod
- orderDate
- deliveryDate

3. Product Table (product):
Attributes:

- title
- description
- mainImg
- discount
- price
- gender

- category
- carousel
- sizes

4. Cart Table (cart):
Attributes:

- userId
- title
- description
- mainImg
- sizes
- discount
- price
- quantity

5. Admin Table (admin):
Attributes:

- banner
- categories

PRE-REQUISITES:

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

Install Dependencies:

- Navigate into the cloned repository directory:

```
cd containment-zone
```

- Install the required dependencies by running the following commands:

```
cd frontend

npm install

cd ../backend

npm install
```

Start the Development Server:

- To start the development server, execute the following command:

```
npm start
```

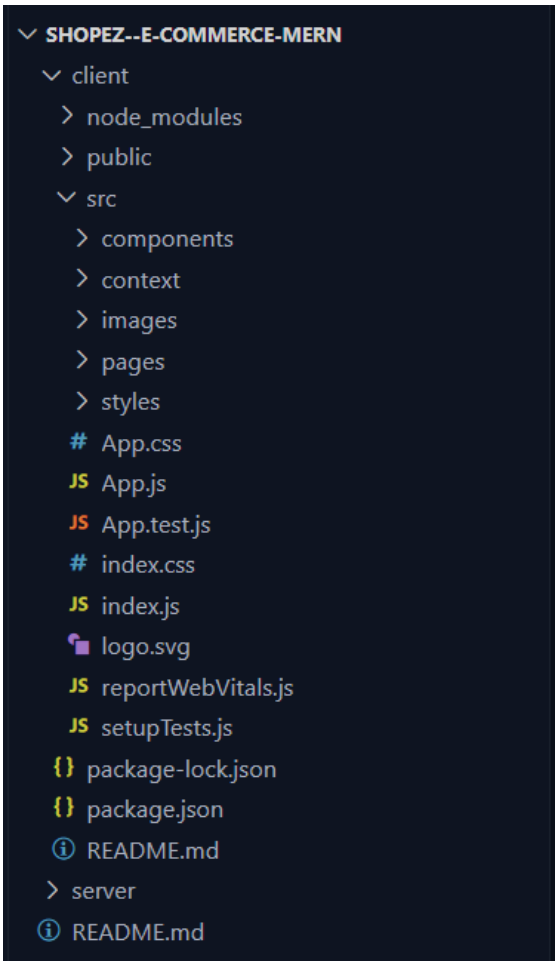
- The OLP app will be accessible at http://localhost:5173

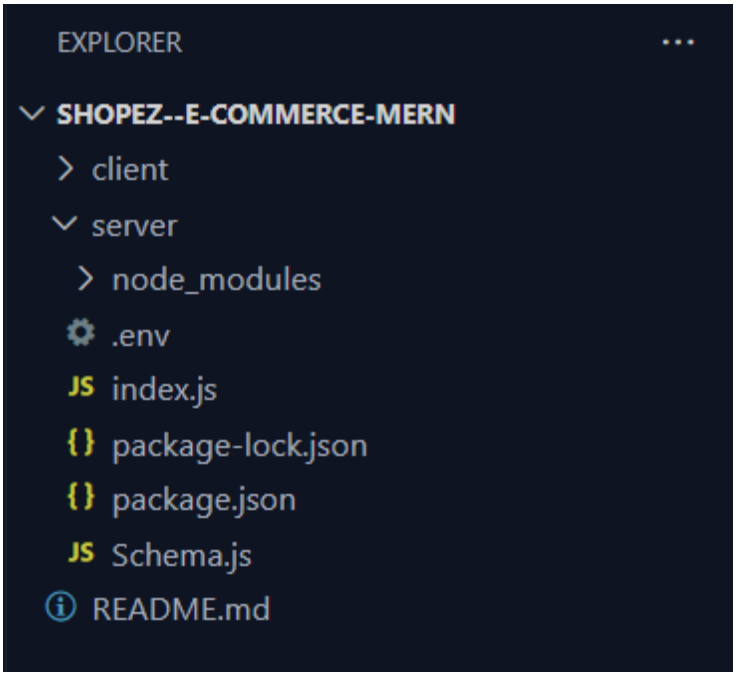
You have successfully installed and set up the Online learning app on your local machine. You can now proceed with further customization, development, and testing as needed.

PROJECT STTRUCTURE

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

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





APPLICATION FLOW:

USER & ADMIN FLOW:

1. User Flow:

• Users start by registering for an account.

• After registration, they can log in with their credentials.

• Once logged in, they can check for the available products in the platform. • Users can add the products they wish to their carts and order.

• They can then proceed by entering address and payment details. • After ordering, they can check them in the profile section.

2. Admin Flow:

• Admins start by logging in with their credentials.

• Once logged in, they are directed to the Admin Dashboard.

• Admins can access the users list, products, orders, etc.,

Milestone 1-Setup & configuration

● **Folder setup:**

1. Create frontend and

2. Backend folders .

Open the backend folder to install the necessary tools

For backend, we use:

● cors

● bcryptjs

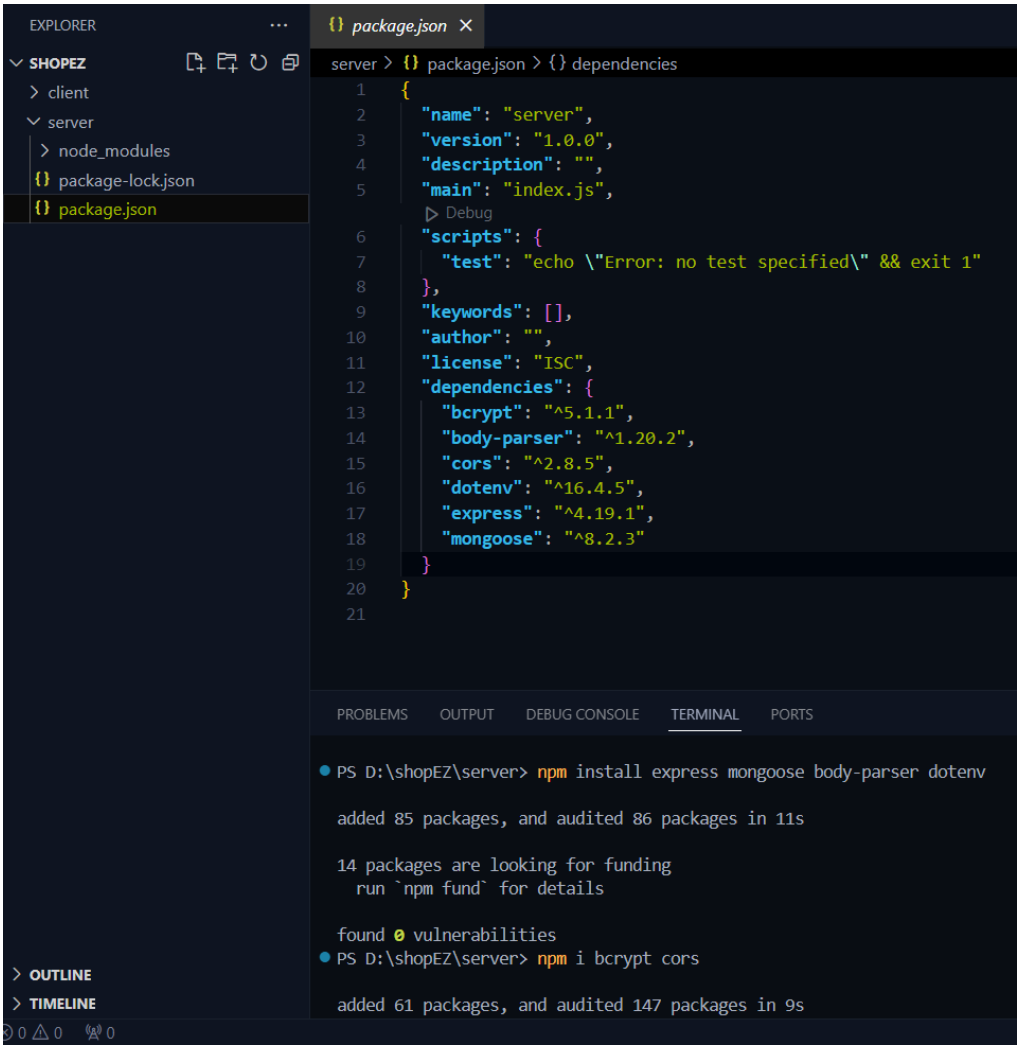
● express

● dotenv

● mongoose

● Multer

● body-parser



Milestone 2- Backend Development

- **Setup express server**

1. Create index.js file in the server (backend folder).

2. define the port number, MongoDB connection string, and JWT key in the env file to access it.
3. Configure the server by adding cors, and body-parser.

- **Add authentication:** for this,

1. You need to make a middleware folder and in that make authMiddleware.js file for the authentication of the projects and can use in.

Ref:backend.mp4

Milestone 3- Database

- **Configure MongoDB**

1. Import mongoose.

2. Add database connection from config.js file present in the config folder

3. Create a model folder to store all the DB schemas.

Ref:database.mp4

Milestone 4- Frontend Development

- **Installation of required tools:**

- For frontend, we use:

1. React

2. Bootstrap

3. Material UI

4. Axios

5. Antd

6. mdb-react-ui-kit

7. react-bootstrap

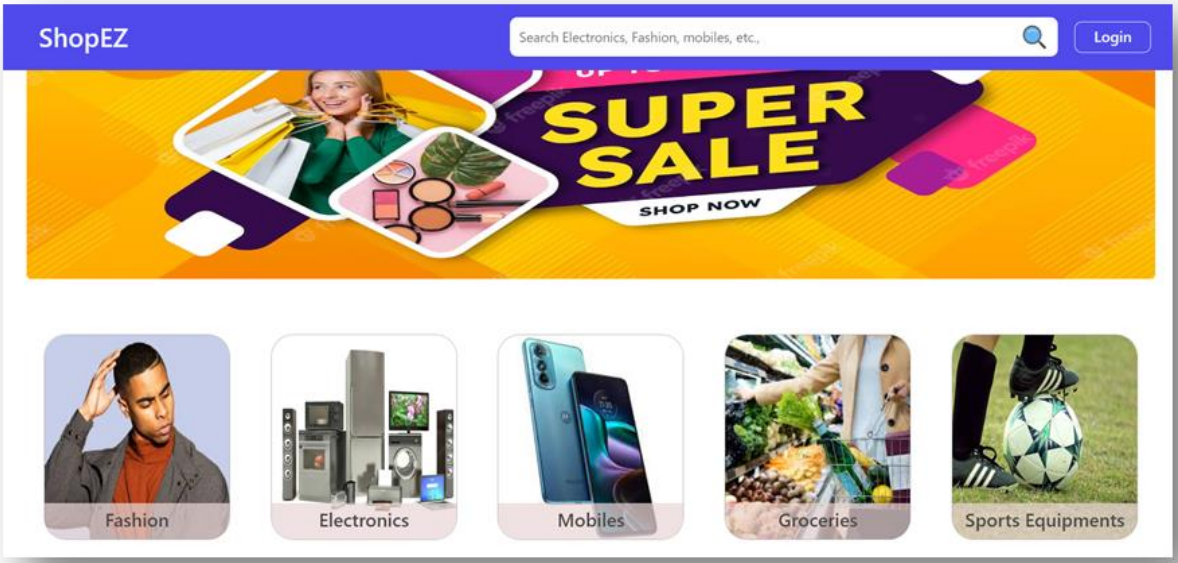
Ref:frontend.mp4

Milestone 5: Project Implementation:

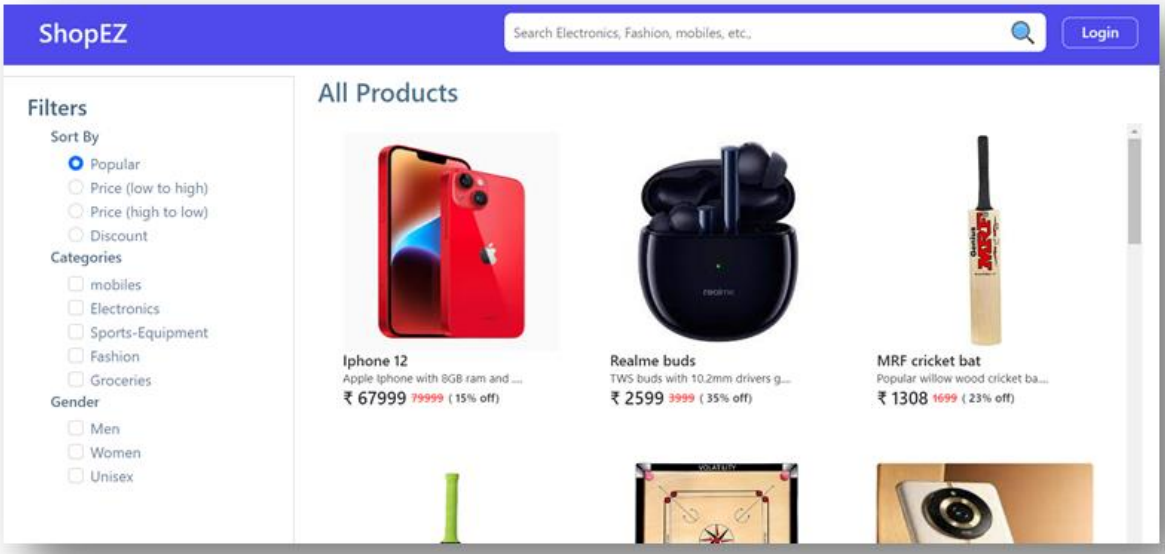
On completing the development part, we then ran the application one last time to verify all the functionalities and look for any bugs in it. The user interface of the application looks a bit like the one's provided below

.

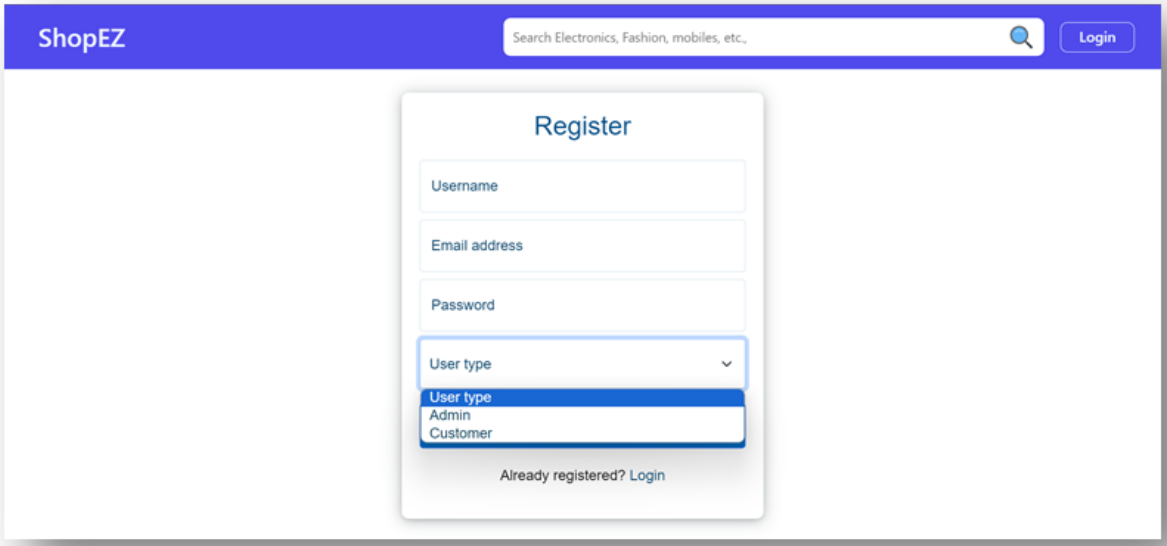
Landing Page



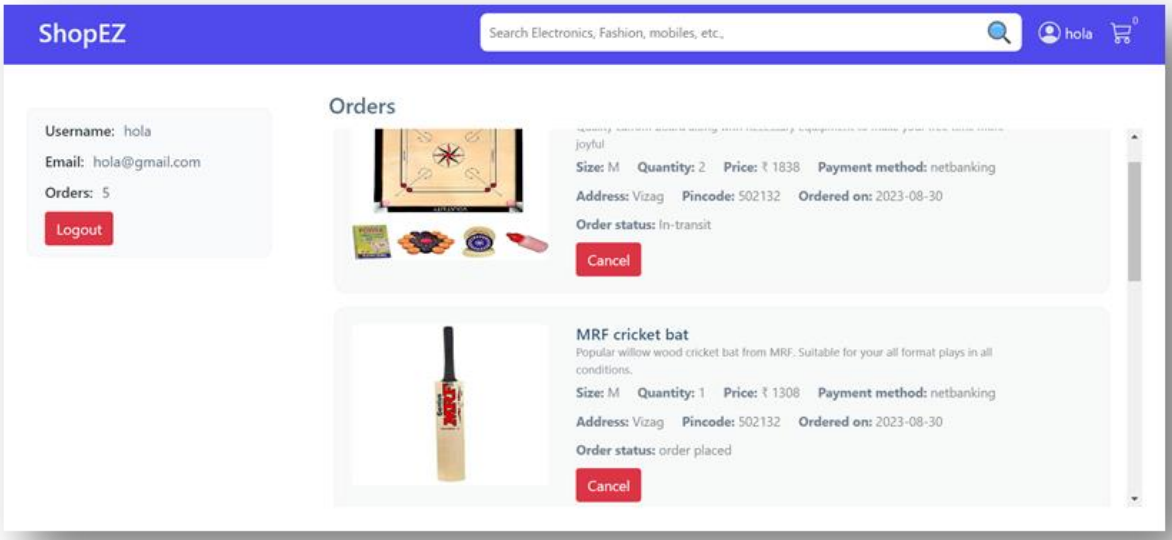
Products



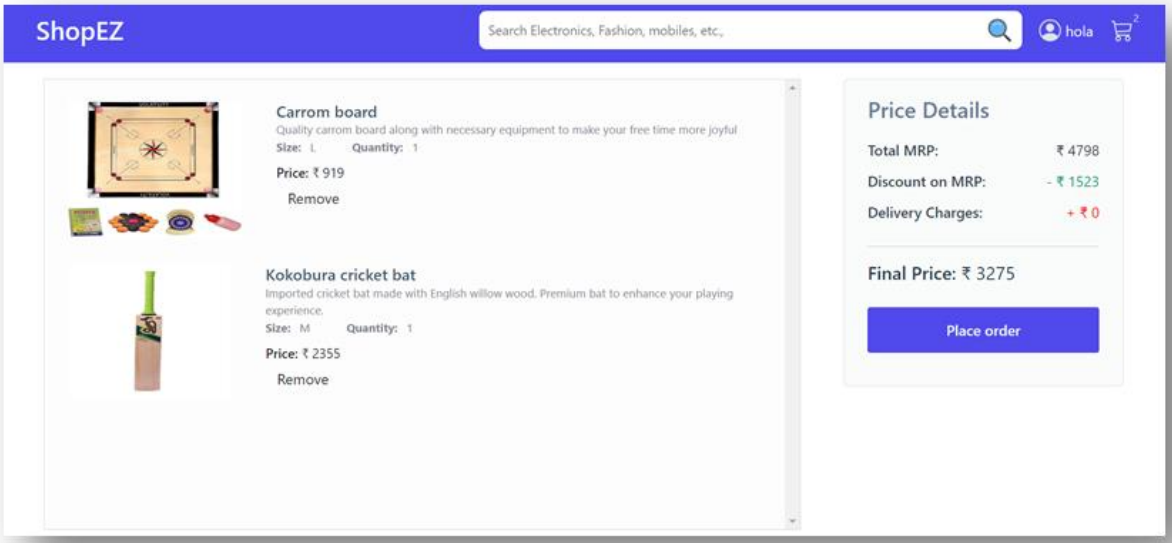
Authentication



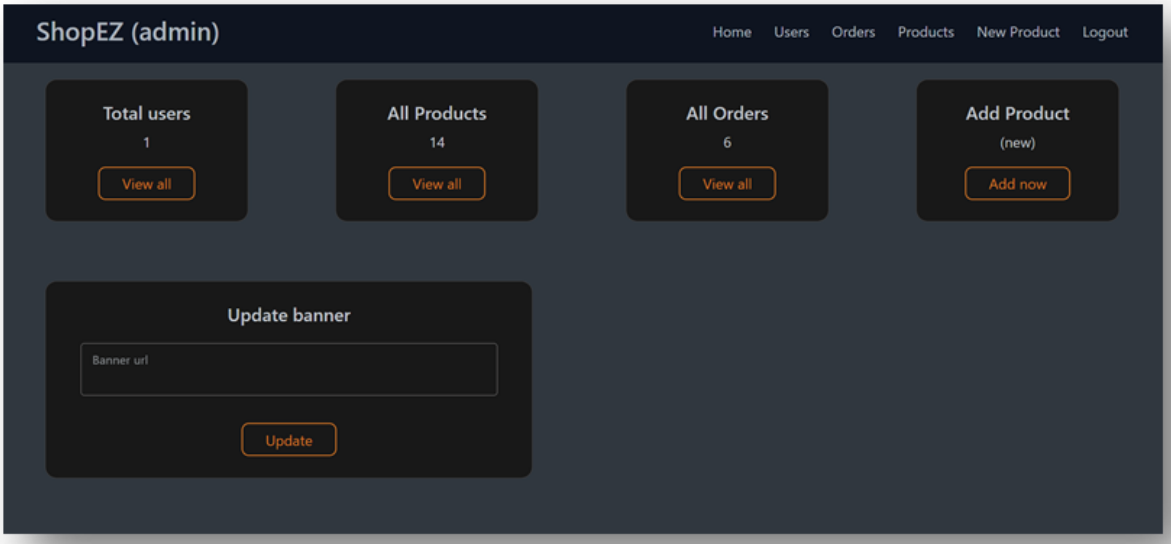
User Profile



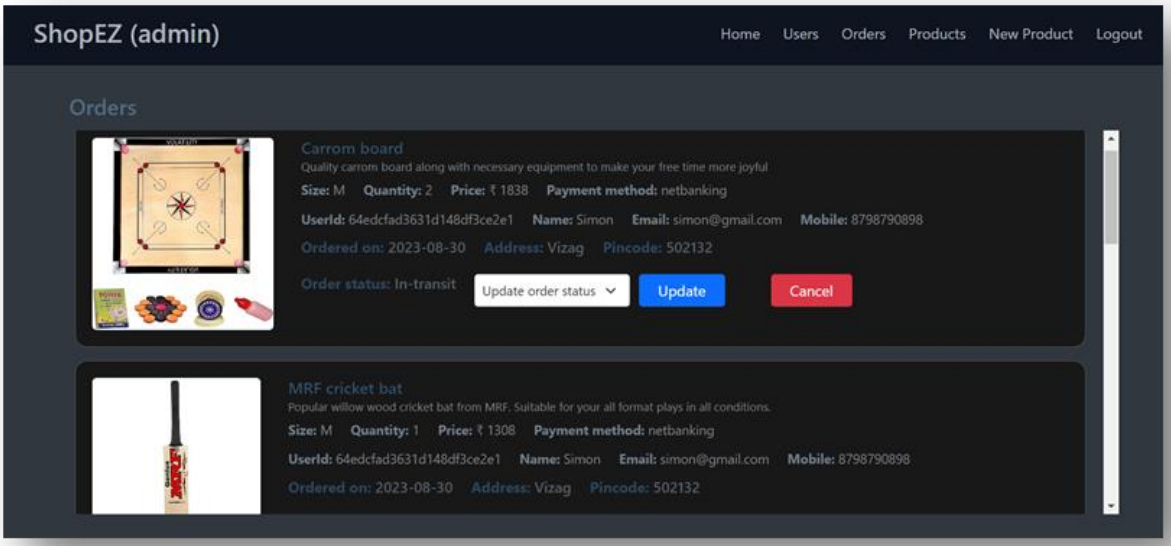
Cart



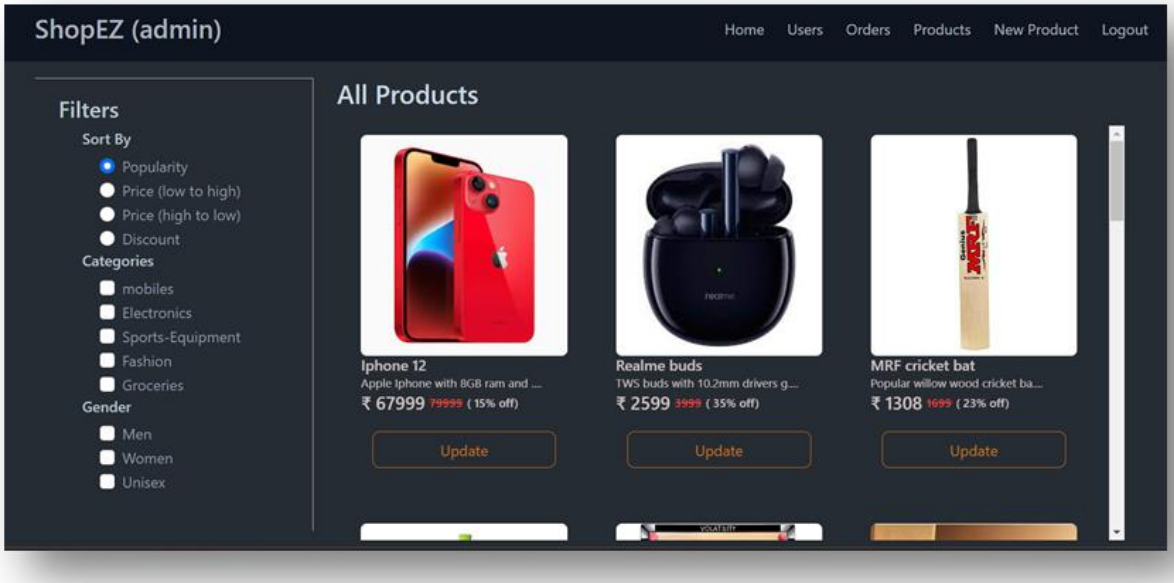
Admin Dashboard



All Orders



New Product Page



Github Repository : https://github.com/Asif0718/Shop-EZ

Demo Video :
https://drive.google.com/file/d/1kVcKsjUsufssZcMQNWC8Spm4jG-A8sMc/view?usp=sharing