**ShopEZ: E-Commerce Application**

**INTRODUCTION**

ShopEZ makes online shopping simple and enjoyable with its easy-to-use interface. Quickly and what you need with smart search and personalized recommendations. Explore detailed product info, reviews, and deals to shop confidently. Enjoy a smooth, secure checkout and instant order confirmation. Sellers benefit from an efficient dashboard to manage orders with ease. Get valuable insights and analytics to grow your business smarter.

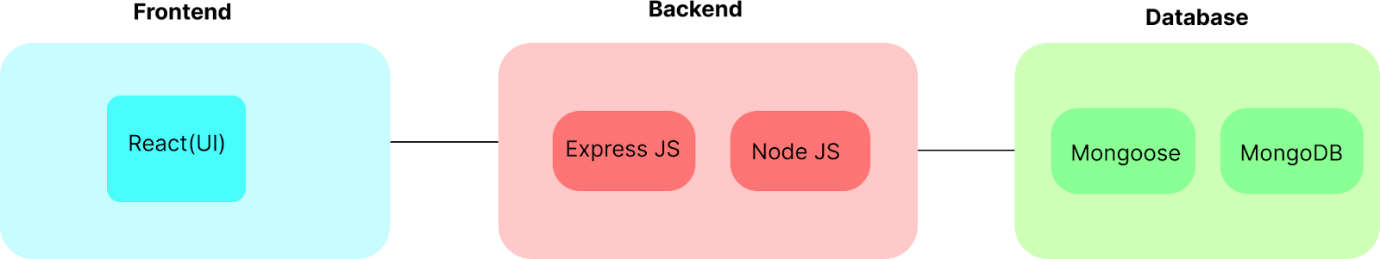
**SCENARIO**: **Mark’s Last-Minute Anniversary Surprise**

Mark, a working dad with two kids, realizes he forgot his wedding anniversary is just two days away. Wanting to surprise his wife with something meaningful, he turns to ShopEZ for help**.**

1. **Effortless Product Discovery**: Mark quickly searches "gifts for her" and narrows down options using filters for jewellery, price, and delivery time.
2. **Personalized Recommendations**: ShopEZ suggests a beautiful heart-shaped pendant based on his past searches and trending anniversary picks.
3. **Quick Product Details**: Mark reads reviews, views product images, and checks the estimated delivery date—all in one place.
4. **Seamless Checkout Process**: He enters express delivery and completes payment securely in minutes.
5. **Instant Order Confirmation:** Mark receives a confirmation and tracking details in his inbox right away.
6. **Seller Coordination:** The seller processes the order promptly through ShopEZ’s dashboard, ensuring it ships the same day.
7. **On-Time Delivery:** The pendant arrives just in time, beautifully packaged.
8. **Celebrating with Joy**: Mark surprises his wife, who is touched by the thoughtful gift—making the day special despite the last-minute rush.

With ShopEZ, Mark turns a potential disaster into a heartfelt celebration, thanks to the platform's speed, personalization, and reliability.

**TECHNICAL ARCHITECTURE:**

****

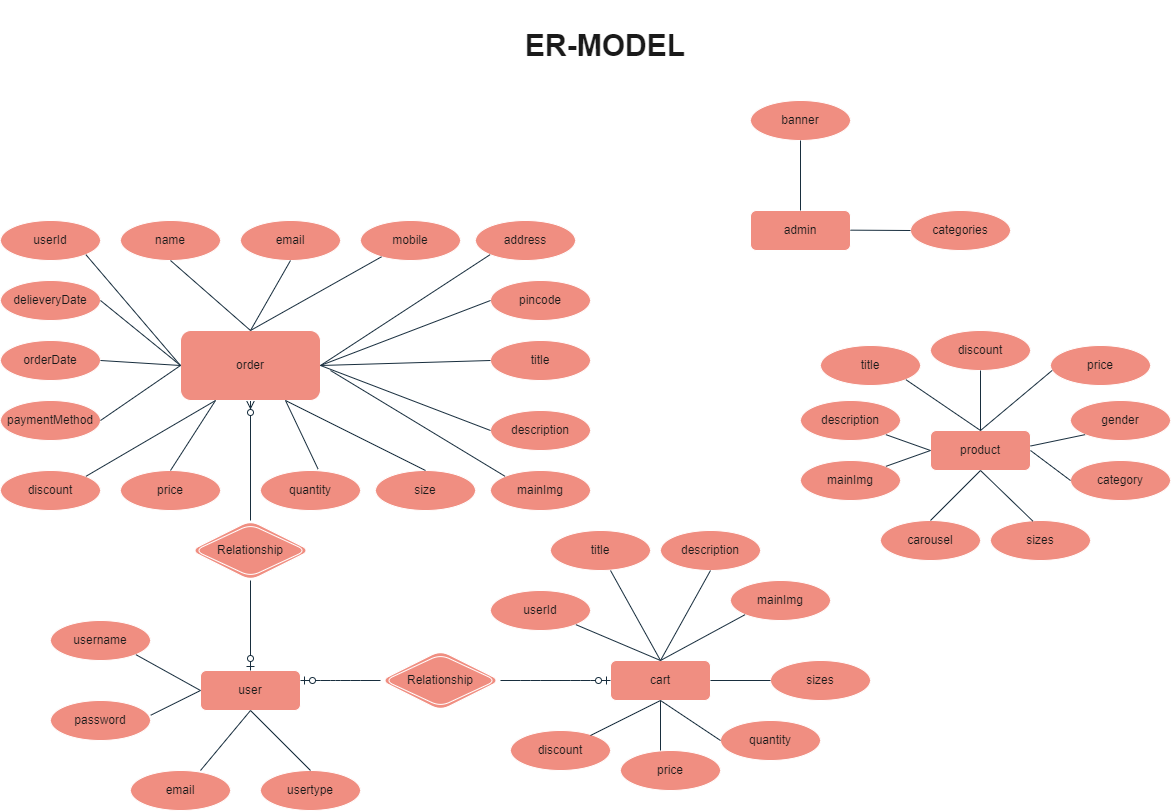
In this architecture diagram:

• The frontend is represented by the "Frontend" section, including user interface components such as User Authentication, Cart, Products, Profile, Admin  dashboard, etc.,

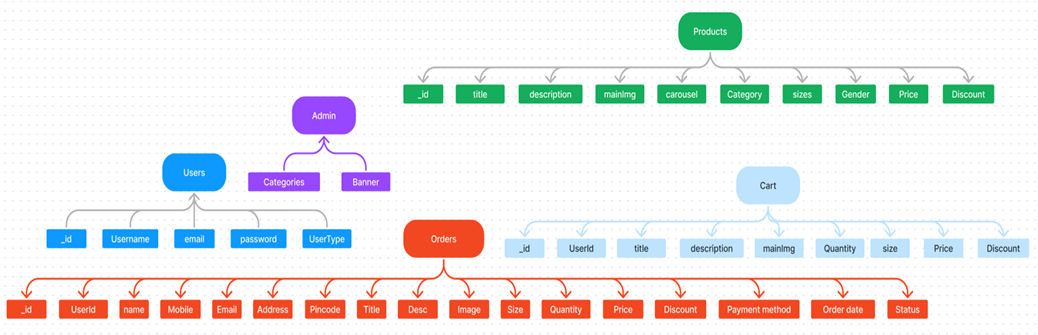
•The backend is represented by the "Backend" section, consisting of API endpoints for Users, Orders, Products, etc.,It also includes Admin Authentication and an Admin Dashboard.

•  The Database section represents the database that stores collections for Users, cart, Orders and Product.

**ER DIAGRAM:**



• The Database section represents the database that stores collections for Users, Admin, Cart, Orders and products.

****

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.

**FEATURES:**

**Comprehensive Product Catalog**

ShopEZ offers a wide range of products with detailed descriptions, customer reviews, pricing, and available discounts, making it easy to find exactly what you need.

**Quick Purchase with "Shop Now"**

Every product listing includes a convenient **"Shop Now"** button, allowing you to begin the buying process instantly.

**Order Details Page**

After clicking "Shop Now," you're taken to a page where you can input shipping information, choose a payment method, and add specific product notes.

**Secure & Efficient Checkout**

ShopEZ ensures a safe and hassle-free checkout process, protecting your personal data while keeping things fast and simple.

**Instant Order Confirmation**

Once your order is placed, you'll receive an immediate confirmation along with access to an order summary page for tracking and review.

**Robust Seller Dashboard**

Sellers benefit from a powerful dashboard where they can manage listings, monitor orders, track customer activity, and view sales history.

**Seamless Shopping Experience**

With its user-friendly design and smart features, ShopEZ makes online shopping and selling convenient, secure, and enjoyable for all users.

**PREREQUISITES:**

* **Node.js & npm**:  
  Install Node.js (includes npm) for backend development.  
  [Download](https://nodejs.org/en/download/) | [Install Guide](https://nodejs.org/en/download/package-manager/)
* **MongoDB**:  
  Set up MongoDB locally or use a cloud service to store data.  
  [Download](https://www.mongodb.com/try/download/community) | [Install Guide](https://docs.mongodb.com/manual/installation/)
* **Express.js**:  
  Backend framework for routing and API handling.  
  Install via terminal: npm install express
* **React.js**:  
  Front-end library for building UI components.  
  [React Setup Guide](https://reactjs.org/docs/create-a-new-react-app.html)
* **Basic Web Technologies**:  
  Knowledge of HTML, CSS, and JavaScript is essential.
* **Database Connectivity**:  
  Use **Mongoose** to connect Node.js with MongoDB.  
  Guide
* **Version Control (Git)**:  
  Track changes and collaborate using Git and GitHub.  
  [Download Git](https://git-scm.com/downloads)
* **Development Environment**:  
  Recommended editors:
  + [VS Code](https://code.visualstudio.com/download)
  + [Sublime Text](https://www.sublimetext.com/download)
  + [WebStorm](https://www.jetbrains.com/webstorm/download)

**2. Running the ShopEZ App**

* **Clone the Repository**:

bash

CopyEdit

git clone https://github.com/harsha-vardhan-reddy-07/shopEZ--e-commerce-MERN

* **Install Dependencies**:

bash

CopyEdit

cd ShopEZ—e-commerce-App-MERN

npm install

* **Start the Development Server**:

bash

CopyEdit

npm run dev # or use npm start

* **Access the App**:  
  Open your browser and go to: <http://localhost:5173>

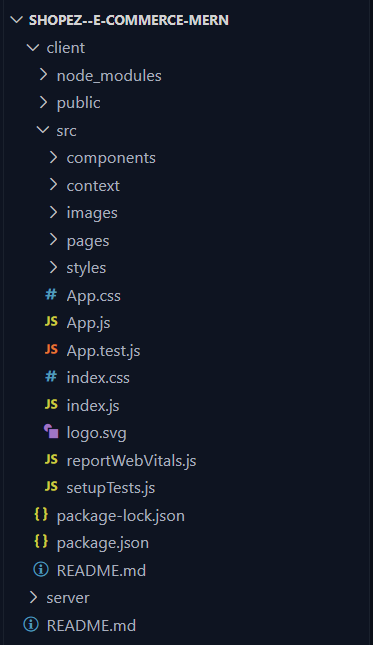
**USER & ADMIN FLOW:**

**User Flow:**

1. Users register and log in using their credentials.
2. After login, they browse available products on the platform.
3. Desired products can be added to the cart.
4. Users proceed to checkout by entering address and payment details.
5. Order history and details are accessible in the profile section.

**Admin Flow:**  
 6. Admins log in to access the backend.  
 7. They are redirected to the Admin Dashboard.  
 8. From the dashboard, admins can manage users, products, and orders.  
 9. Admins can edit listings, track sales, and view customer activity.  
 10. This flow ensures smooth operations and management of the platform.

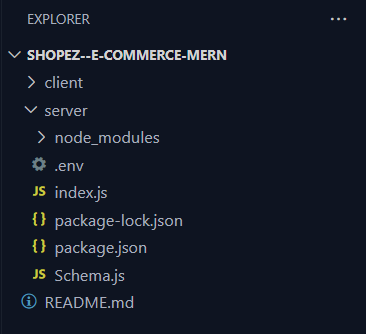
**PROJECT STRUCTURE:**

****

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.



**PROJECT SETUP AND CONFIGURATION:**

**Install required tools and software:**

**• Node.js.**

**Reference Article:** [**https://www.geeksforgeeks.org/installation-of-node-js-on-windows/**](https://www.geeksforgeeks.org/installation-of-node-js-on-windows/)

**• Git.**

**Reference Article:** [**https://git-scm.com/book/en/v2/Getting-Started-Installing-Git**](https://git-scm.com/book/en/v2/Getting-Started-Installing-Git)

**Create project folders and files:**

**• Client folders.**

**• Server folders**

**ReferralVideoLink:** [**https://drive.google.com/file/d/1uSMbPIAR6rfAEMcb\_nLZAZd5QIjTpnYQ/view?usp=sharing**](https://drive.google.com/file/d/1uSMbPIAR6rfAEMcb_nLZAZd5QIjTpnYQ/view?usp=sharing)

**Schema use-case:**

**1. User Schema**

* **Schema**: userSchema
* **Model**: 'User'
* Stores user data including username, email, and password.
* The email field is unique to prevent duplicate registrations.
* Used for registration, login, and authentication.

**2. Product Schema**

* **Schema**: productSchema
* **Model**: 'Product'
* Stores all product-related data like name, price, and description.
* Essential for displaying products and managing orders.

**3. Orders Schema**

* **Schema**: ordersSchema
* **Model**: 'Orders'
* Contains order details such as userId, productId, productName, quantity, size, and orderDate.
* userId references the user who placed the order.

**4. Cart Schema**

* **Schema**: cartSchema
* **Model**: 'Cart'
* Stores data for items added to the cart: userId, productId, productName, quantity, etc.
* userId links to the user owning the cart.

**5. Admin Schema**

* **Schema**: adminSchema
* **Model**: 'Admin'
* Holds admin-related data such as product categories and homepage banners.
* Used for managing content and platform layout from the admin side.

**BACKEND DEVELOPMENT:**

* **Setup express server:**
* • Create index.js file.
* • Create an express server on your desired port number.
* • Define API’s
* Reference Video: <https://drive.google.com/file/d/1-uKMIcrok_ROHyZl2vRORggrYRio2qXS/view?usp=sharing>
* **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 Video: <https://drive.google.com/file/d/19df7NU-gQK3DO6wr7ooAfJYIQwnemZoF/view?usp=sharing>

**2. Database Configuration**

* Set up MongoDB (locally or via Atlas).
* Create collections for users, admins, products, and orders.

**3. Create Express.js Server**

* Set up Express server with middleware like body-parser and cors.

**4. Define API Routes**

* Create separate route files for users, orders, and auth.
* Implement product listing, registration, login, and order APIs.

**5. Implement Data Models**

* Define Mongoose schemas and models for users, products, and orders.
* Implement CRUD operations for each entity.

**6. User Authentication**

* Create routes for user login, registration, and logout.
* Add middleware to protect authenticated routes.

**7. Handle Products & Orders**

* Create routes to fetch/add products and place orders.
* Include validation and database update logic.

**8. Admin Functionality**

* Add routes for admin-specific tasks like product management.
* Apply role-based access control for secure admin access.

**9. Error Handling**

* Use middleware to catch and handle API errors.
* Return appropriate HTTP status codes and messages.

**Frontend Development (React.js)**

**1. Setup React Application**

* Create the React app under a client folder.
* Install libraries and define app structure.

**2. Design UI Components**

* Build reusable components (Navbar, ProductCard, etc.).
* Implement layout, styling, and navigation.

**3. Implement Frontend Logic**

* Connect to backend API using Axios or Fetch.
* Handle data binding, form inputs, and dynamic rendering.

**PROJECT IMPLEMENTATION & EXECUTION:**

**User Authentication:**

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

**·** Frontend:

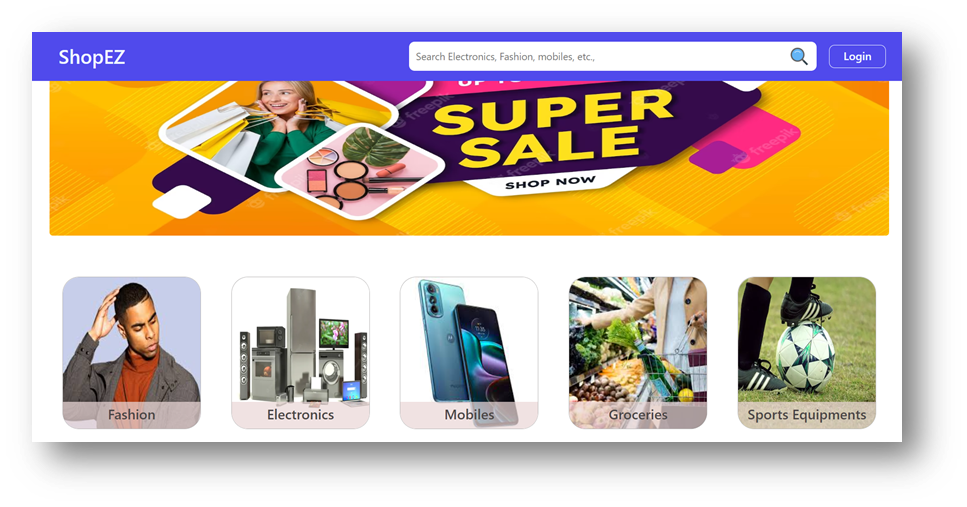
* Login
* Logout
* Register

**·**All Products (User): In the home page, we’ll fetch all the products available in the platform along with the filters.

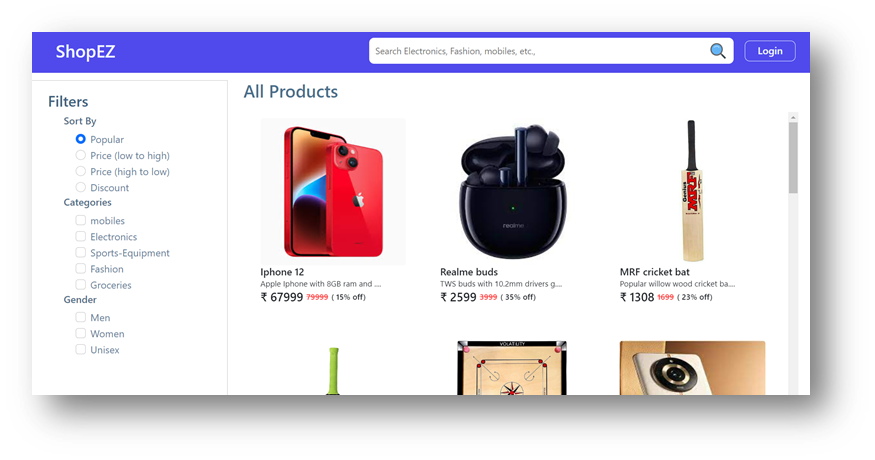
**·**Add product to cart: Here, we can add the product to the cart or can buy directly.

**Demo UI images:**

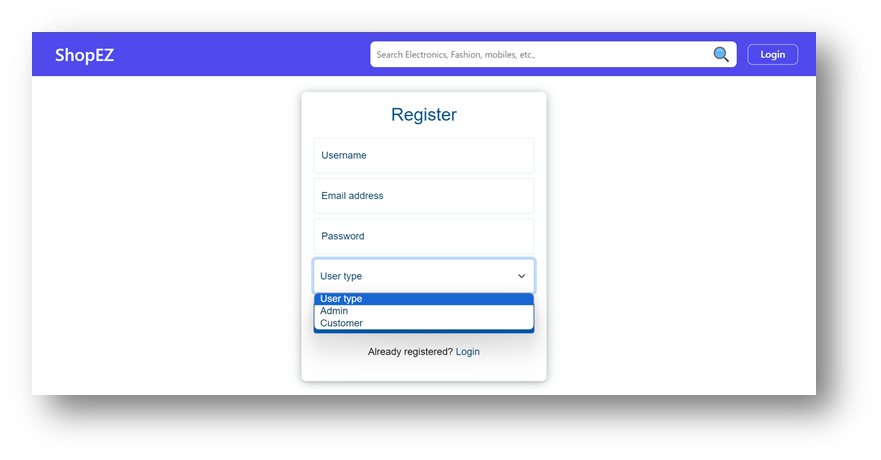
·       **Landing page**

****

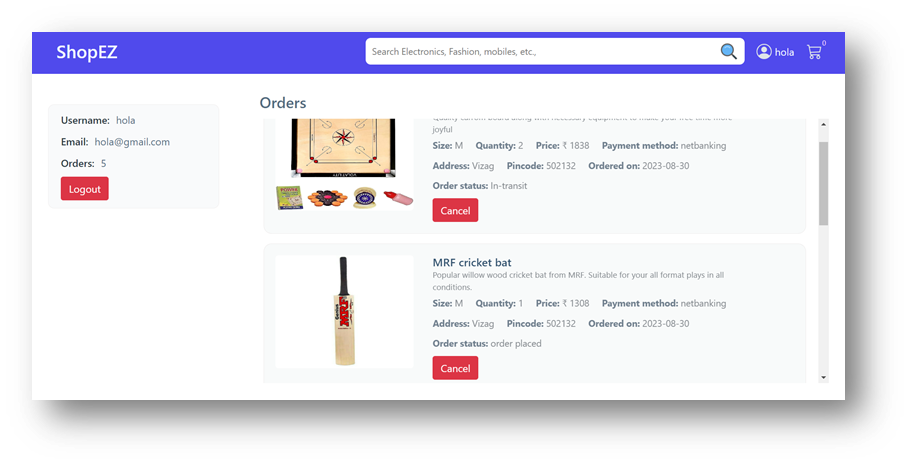
·       **Products**

****

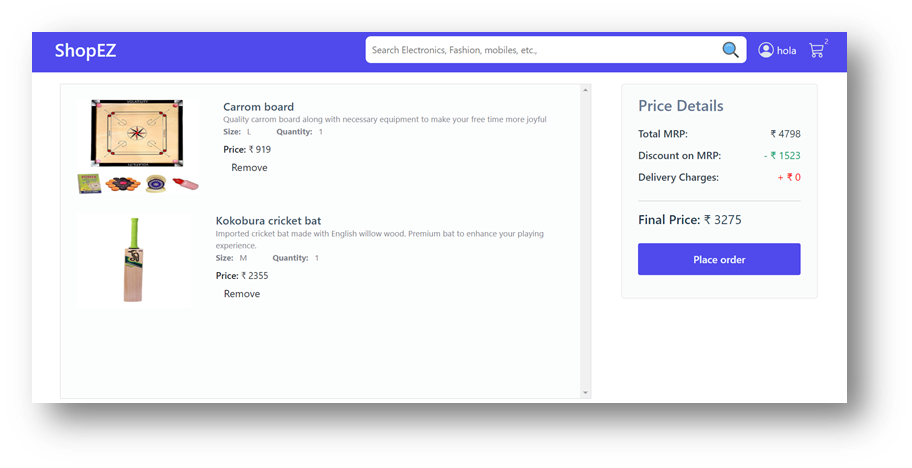
·  **Authentication**

****

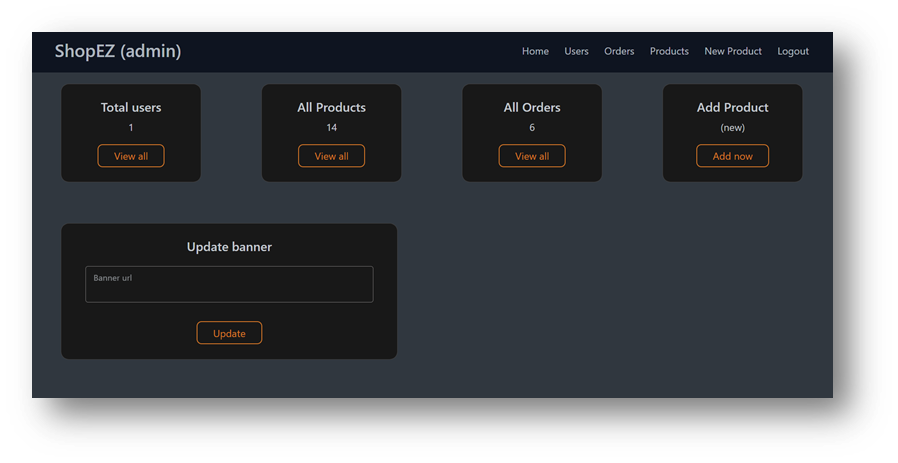
·       **User Profile**

****

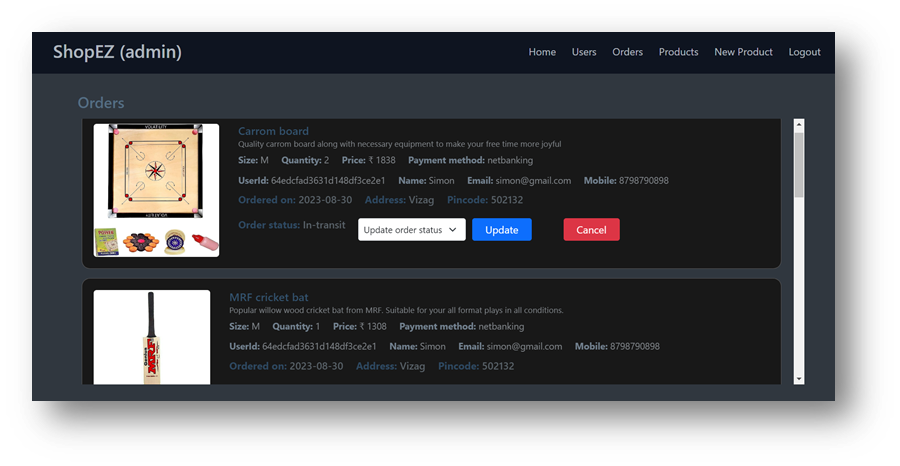
·       **Cart**

****

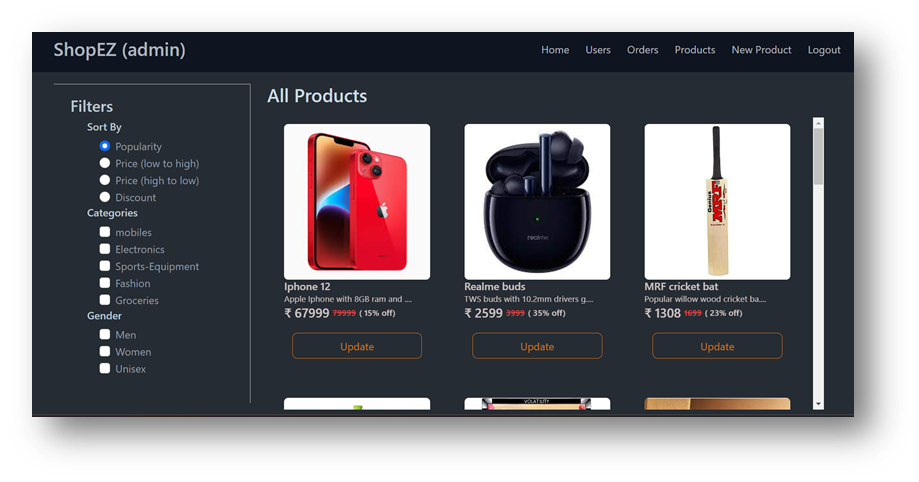
·       **Admin dashboard**

****

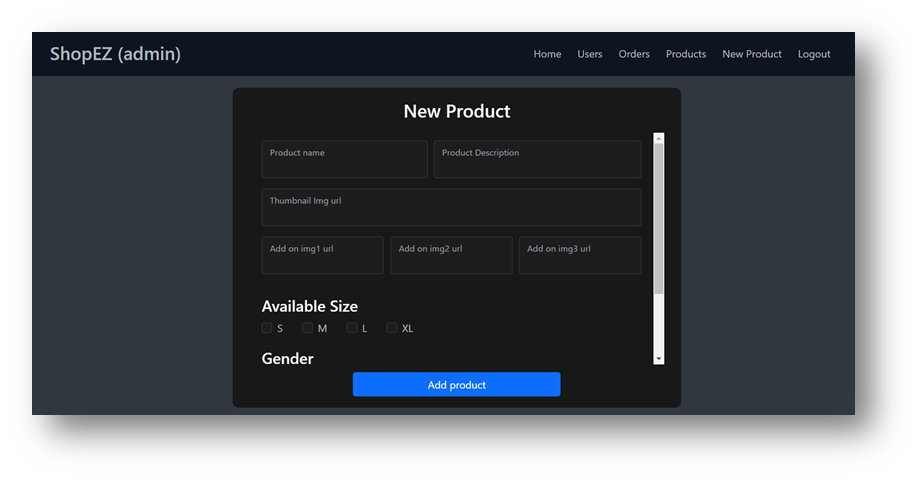
·       **All Orders**

****

·       **All Products**

****

·       **New Product Page**

****

For any further doubts or help, please consider the GitHub repo,

<https://github.com/Harshitha9087/ShopEZ>

The demo of the app is available at:

<https://drive.google.com/drive/folders/1PWRCwAH22t9daUlPWqf-fbUzzYwMhfxC?usp=sharing>