**Full Stack Development with MERN**

**Project Documentation - ShopEZ - E-Commerce Application**

**1. INTRODUCTION**

**Project Title**: ShopEZ - E-Commerce Application

**Team id**: NM2024TMID00168

**Team Members:**

Rakshitha C - 413021104031 (Team leader)

Chandru M - 413021104004

Komal R - 413021104019

Shekina priya S- 413021104036

**2. Project Overview**

* **Purpose:**

The shopEZ E-Commerce application aims to provide users with a seamless, convenient, and interactive platform to browse, search, compare, and purchase a wide variety of products online. It simplifies the shopping experience by offering real-time product availability, pricing, and user-specific recommendations. Users can shop for products across multiple categories such as electronics, fashion, home goods, and more.

* **Goals:**

1.Enhanced User Experience:

* + - Provide an intuitive and smooth interface for browsing and purchasing products.
    - Simplify the checkout and payment process to minimize the time required for users to complete their transactions.

2. Cost Efficiency and Transparency:

* + - Offer clear and transparent pricing, including taxes, discounts, and shipping costs.
    - Integrate loyalty programs, discounts, and promotions to enhance value for customers.

3. Accessibility and Convenience:

 Ensure the platform is accessible via both web and mobile devices.

* Support multi-currency and multi-language capabilities for a global audience.

4. Secure Transaction :

* secure payment gateways (like PayPal, Stripe, or others) for various payment methods.
* Implement robust security measures to protect user data and privacy.

5.Customer Support:

* + - * offer customer support with live chat, email, and a help center for returns, queries, and issue resolution.
      * Provide real-time updates on order statuses, shipping, and delivery tracking.

**3. Features:**

1. Product Search:

* Users can search for products based on categories, price range, brand, ratings, etc.

2. Advanced Filters:

Filter results by:

* Price range
* Brand
* Rating
* Categories (e.g., electronics, clothing, accessories)
* Availability (in-stock, out-of-stock)

3. Shopping Cart:

* Users can add products to a shopping cart and proceed with the checkout.
* The cart displays item details, pricing, and total cost.

4. User Profile and History:

* Users can create and manage accounts, view order history, and track shipping statuses.

5. Payment Gateway Integration:

* Secure payment processing using services like PayPal, Stripe, or credit/debit cards.

6. Discounts and Coupons:

* Option for applying promotional codes, discounts, and seasonal offers.

7. Real-time Inventory:

* Real-time product availability updates for a smooth shopping experience.

**4. Architecture:**

A robust and scalable architecture is necessary for handling the complex requirements of an e-commerce platform. Here's an updated breakdown:

**a) Front-End:**

* **Web Application**: Built using frameworks like React, Angular, or Vue.js for a responsive and interactive UI.
* **Mobile Application**: Native or hybrid mobile apps (React Native, Flutter) for seamless shopping on mobile devices.
* **Features:**
* **Web Application**: Built using frameworks like React, Angular, or Vue.js for a responsive and interactive UI.
* **Mobile Application**: Native or hybrid mobile apps (React Native, Flutter) for seamless shopping on mobile devices.
* **Features:**
  + User-friendly interface for product search, browsing, and purchasing.
  + Support for multi-language and multi-currency.
  + Real-time notifications for order updates, offers, and discounts.

**b) Back-End:**

* Microservices Architecture:

1. **Product Service**: Handles the product catalog, descriptions, images, categories, and stock levels.

2. **Order Service**: Manages the shopping cart, order processing, and checkout.

3**. User Management Service**: Manages user profiles, authentication, and order history.

4**. Payment Service**: Integrates with payment gateways for processing transactions.

5**. Notifications Service**: Sends email, SMS, and push notifications to users regarding order statuses, promotions, etc.

**c) Database:**

* Relational Database (RDBMS): Stores product details, user profiles, and orders (e.g., MySQL, PostgreSQL).
* NoSQL Database: Stores unstructured data like reviews, search logs, and recommendations (e.g., MongoDB).

**5. Folder Structure:**

Here’s a modified version of your flight booking app's folder structure tailored for an e-commerce platform:

**Static Files:**

* index.html, favicon, manifest files, and other assets.

**Src:**

* **Components:** Reusable UI components (e.g., buttons, modals, product cards, cart).
  + - /commo**:** Generic components like buttons, modals.
    - /layout**:** Header, footer, sidebar, navigation.
    - /search**:** Components for product search and filtering.
    - /cart**:** Cart view, checkout, and order summary.
    - /profile: User account management (order history, profile update).
    - /products: Product details page, product lists.
    - /notifications: Toasts, alerts, messages.
* **Features:**
  + - * /auth: Login, signup, and authentication (JWT tokens).
      * /productSearch: Search and filter logic for products.
      * /checkout: Payment processing and order confirmation.
      * /profile: User profile management and order tracking.
* **Pages:**
  + - /home: Landing page displaying featured products and promotions.
    - /productDetails: Detailed product pages.
    - /cart: Shopping cart page.
    - /checkout: Checkout page.
    - /profile: User profile page.
    - /error: Custom error pages like 404 or 500.

**6. Running the Application:**

To run the full-stack e-commerce app locally:

**Frontend:**

bash

cd client

npm install

npm start

**Backend:**

bash

cd server

npm install

npm start

**7. API Documentation:**

**1. Product Search Endpoint:**

GET /api/products/search

Parameters: category, brand, price range, keywords.

Example Response:

json

[

{

"productId": "123",

"name": "Laptop",

"price": 899.99,

"category": "Electronics",

"image": "laptop.jpg"

},

]

**2. User Signup:**

POST /api/auth/signup

Example Request:

json

{

"name": "John Doe",

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

"password": "securepassword"

}

**3. Order Checkout:**

POST /api/orders/checkout\*\*

Parameters: userId, productList, shippingAddress, paymentDetails.

Example Response:

json

{

"message": "Order placed successfully",

"orderId": "98765"

}

**8. Authentication:**

Authentication verifies the identity of users. In this project, it is implemented using **JSON Web Tokens (JWT)**.

**1**. **User Signup**:

* Endpoint: /api/auth/signup
* Users provide their details (e.g., name, email, and password).
* Passwords are securely hashed using a library like **bcrypt** and stored in the database.
* After successful registration, the user is notified (but no token is issued yet).

**2.** **User Login**:

* Endpoint: /api/auth/login
* Users provide their credentials (email and password).
* The password is verified against the hashed version stored in the database.
* Upon successful authentication:

A **JWT** is generated and returned to the user.

This token contains the user’s ID, email, and other claims (e.g., roles) in its payload.

The token is signed using a secret key to ensure its integrity.

**3. Token Generation**:

* The JWT is generated using libraries like **jsonwebtoken**.
* Claims include:

sub (subject): User ID.

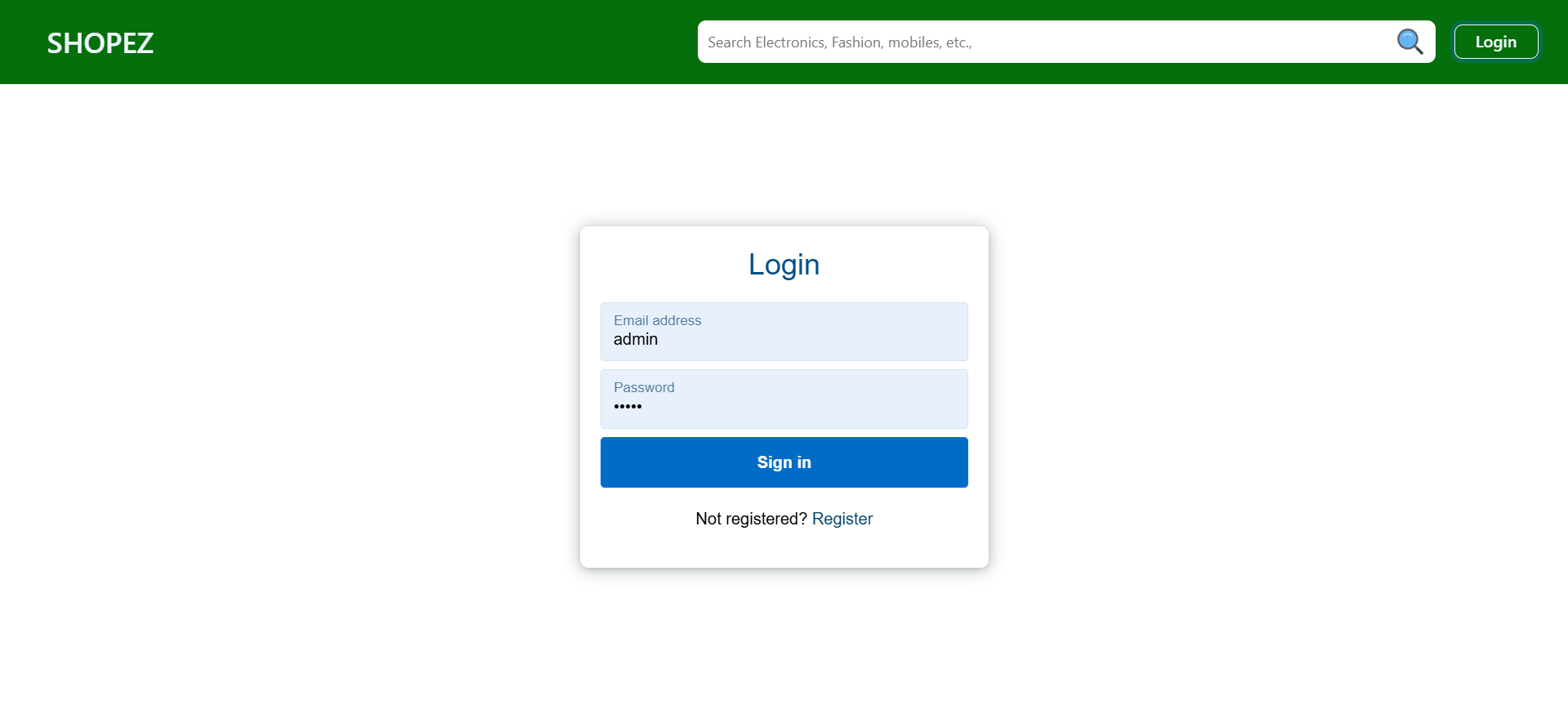
iat (issued at): Timestamp of token issuance.

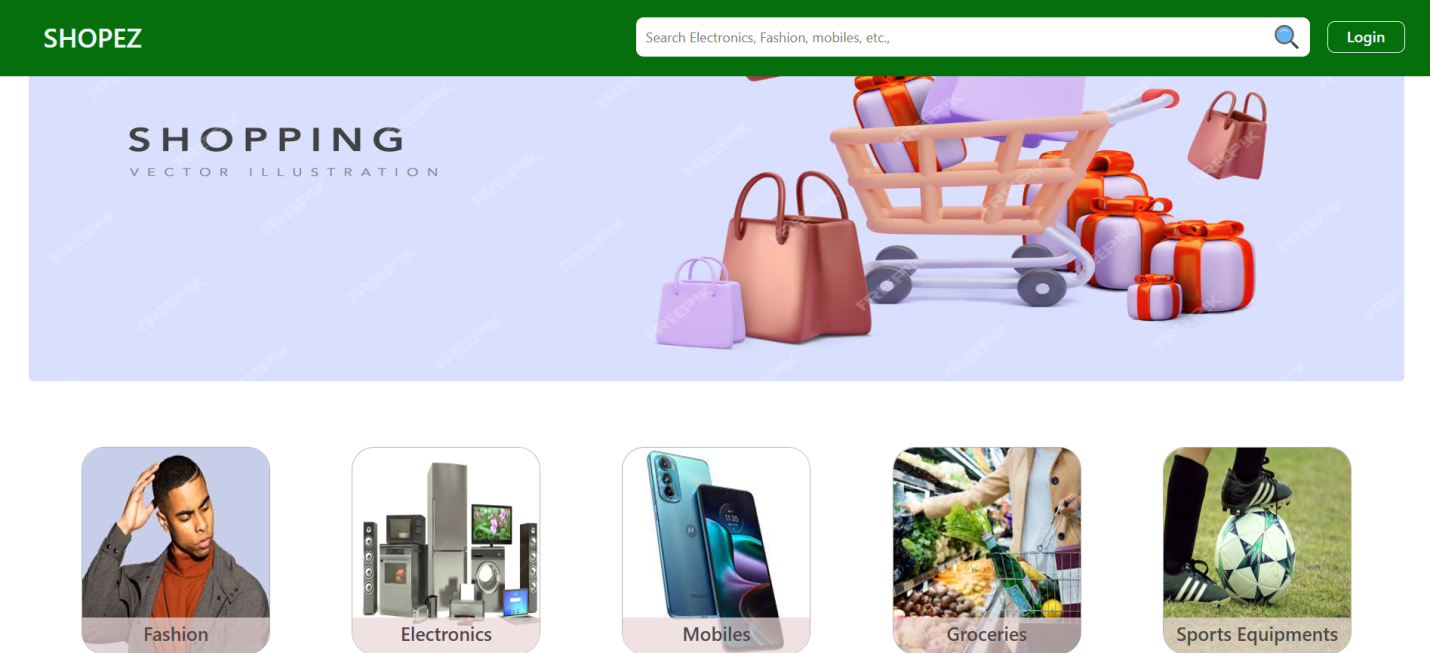
exp (expiration): Token expiry time .

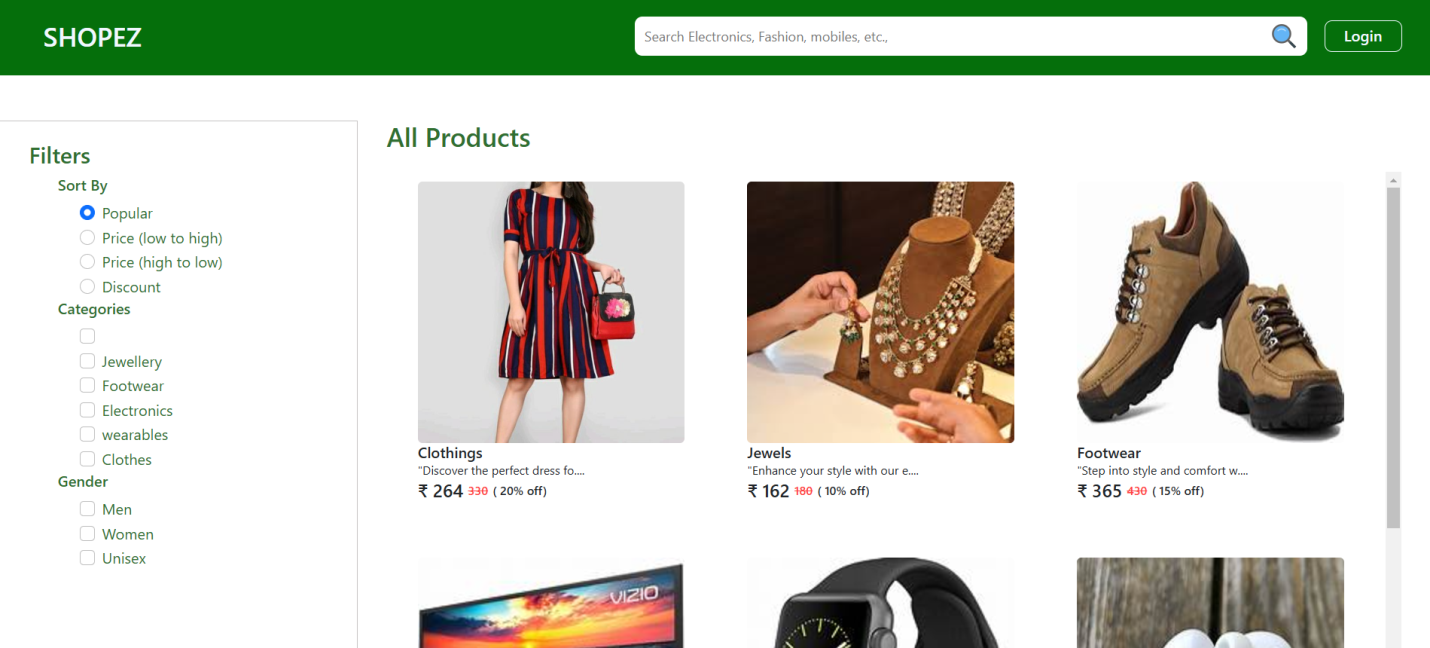
4. **Storing the Token**:

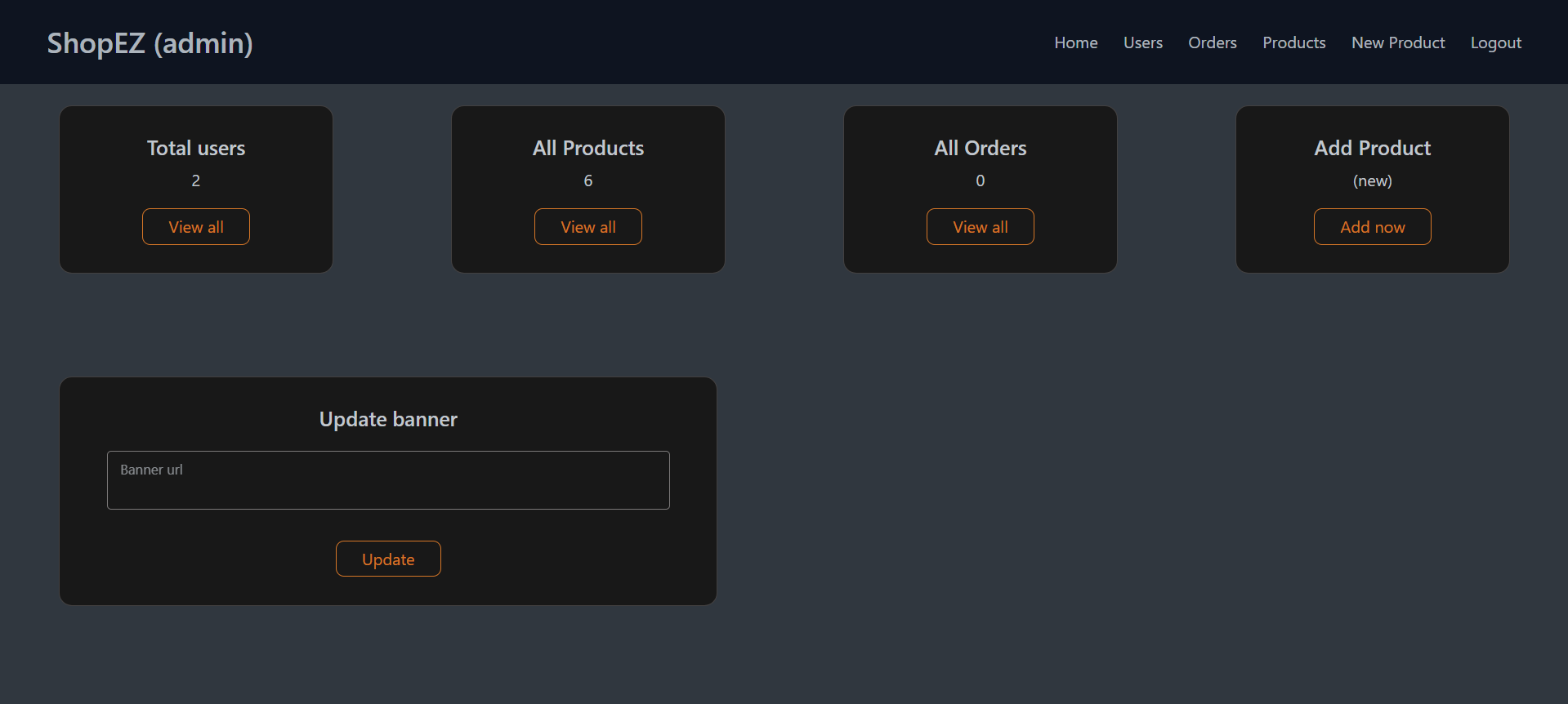
* The client stores the JWT in **localStorage** or **cookies** (with HttpOnly for security).
* For mobile apps, secure storage mechanisms like **Keychain** (iOS) or **Keystore** (Android) are used.

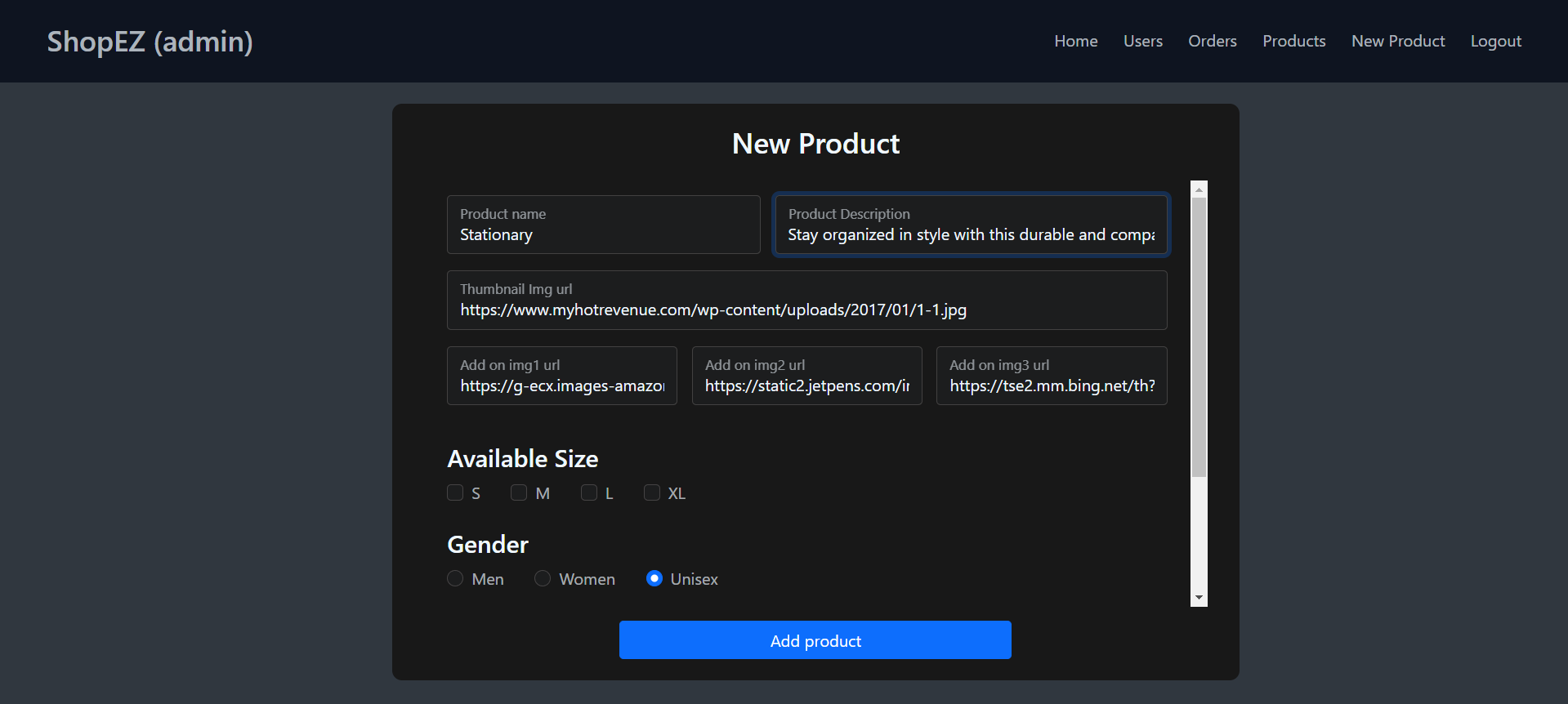
**9. User Interface:**

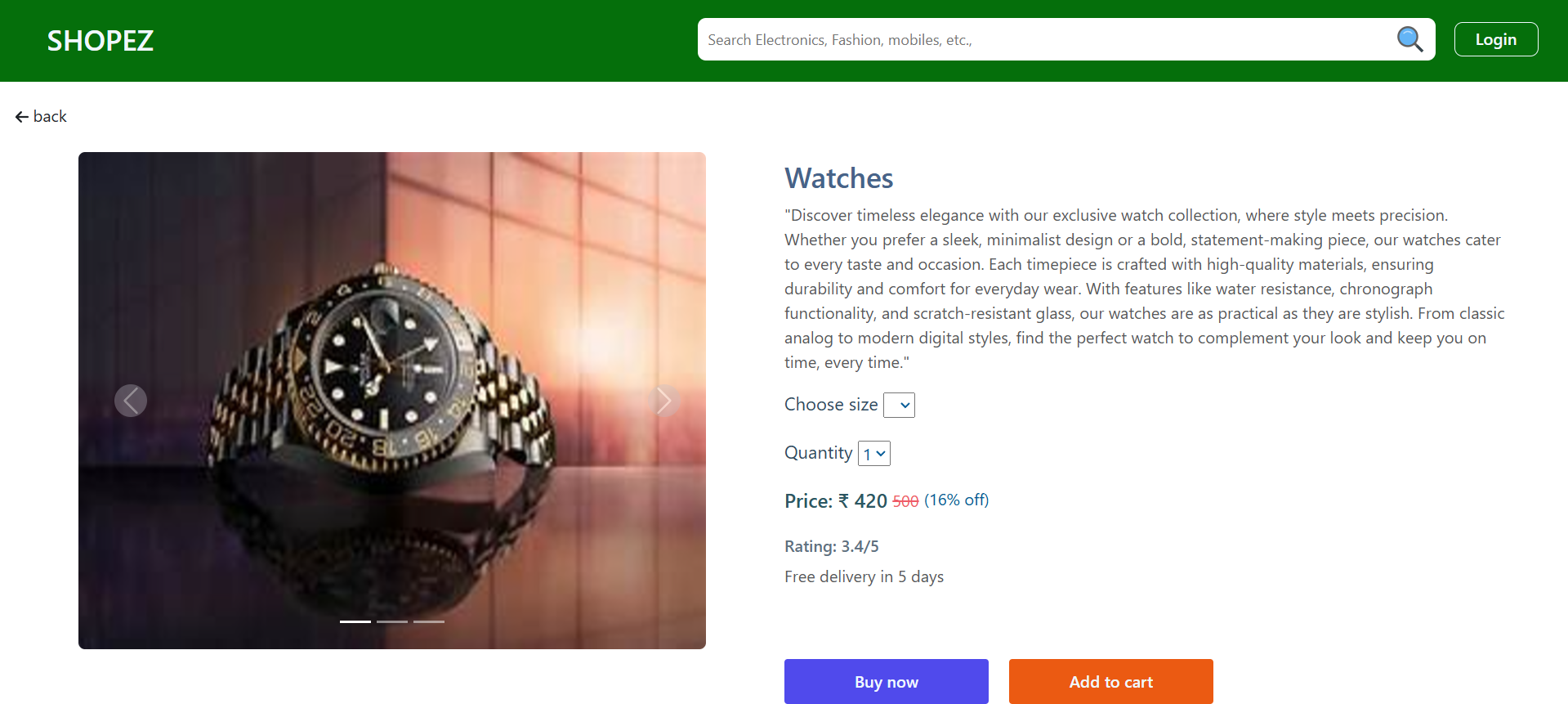
****

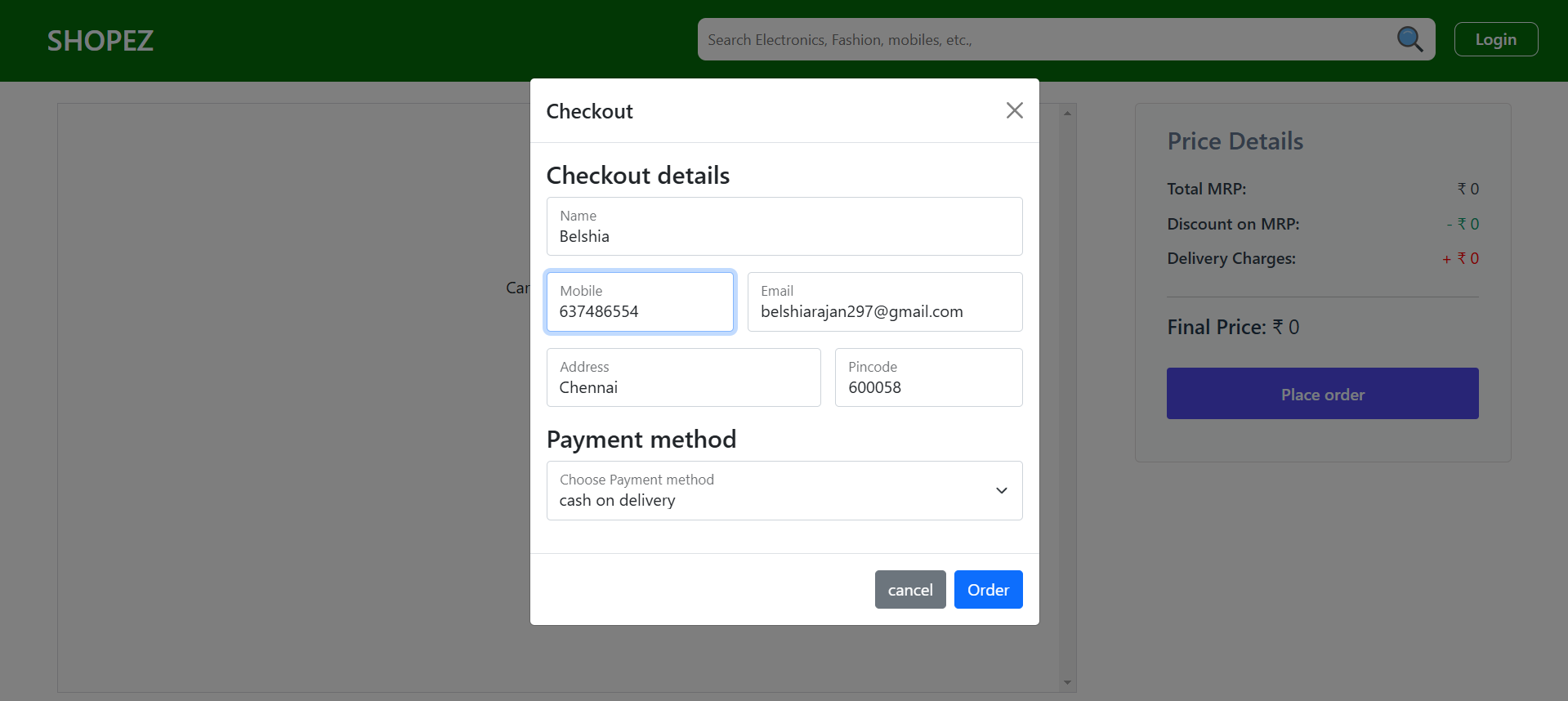
****

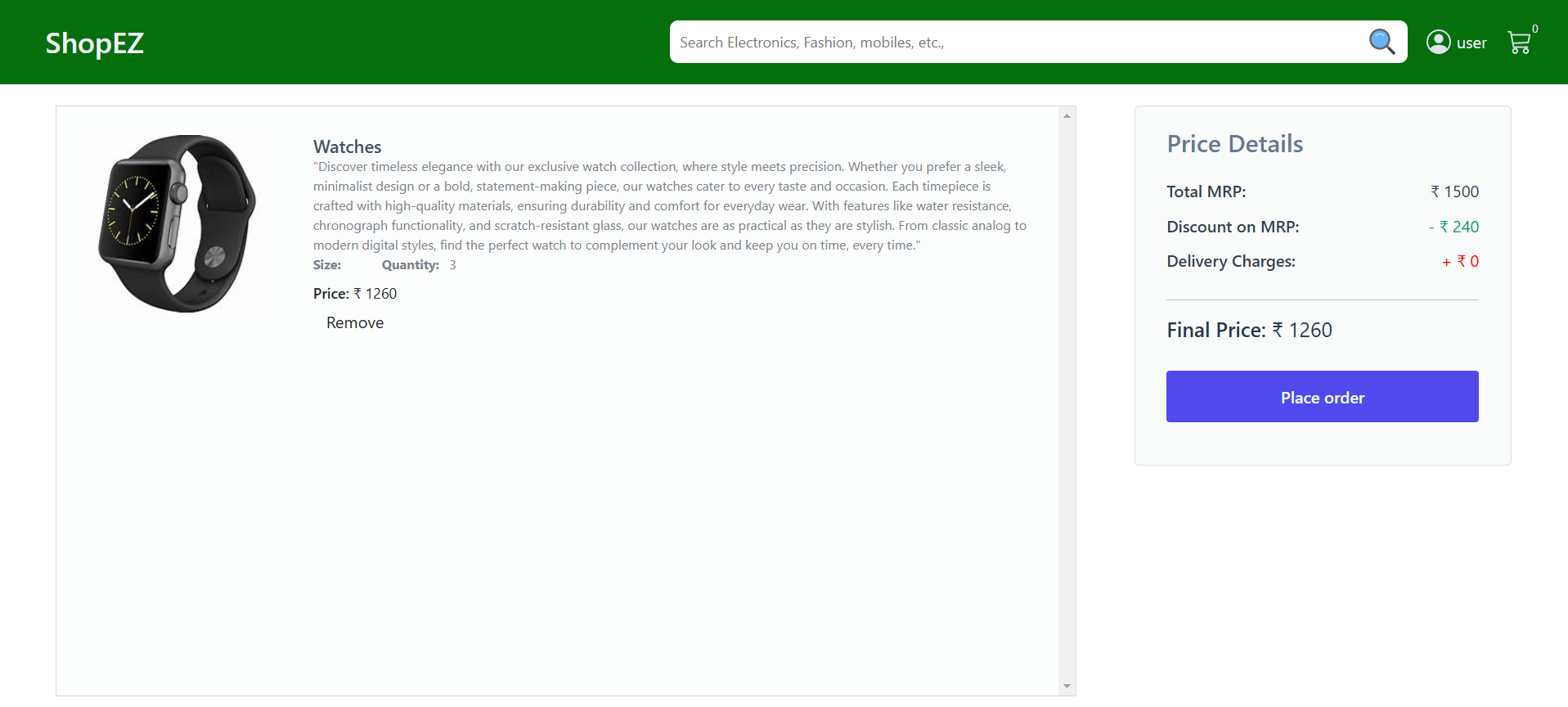
****

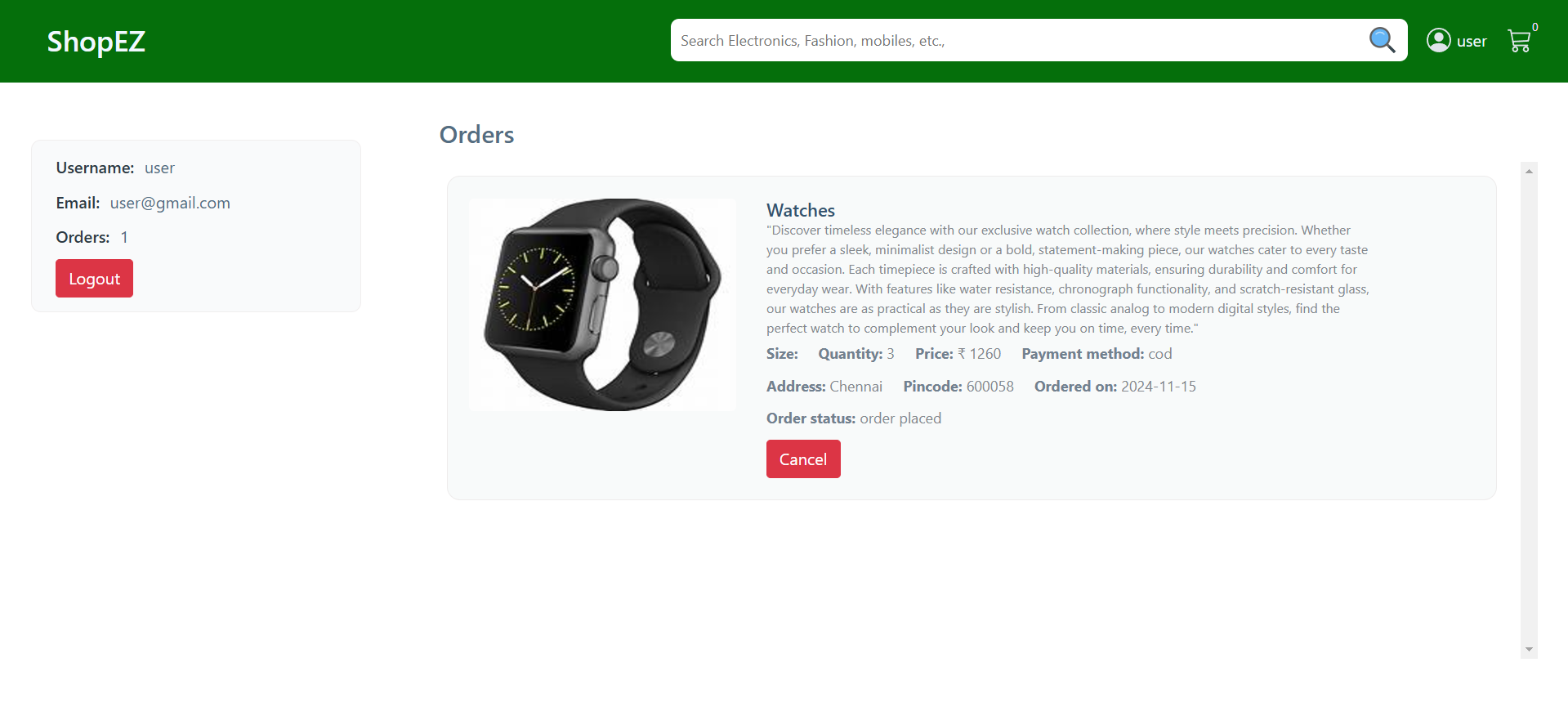
****

****

****

****

****

****

**10. Testing:**

**Manual Testing**: Test product search, cart management, checkout, and payment integration.

**11. Screenshots or Demo:**

Demo link:  
https://github.com/rakshitha-777/shopEZ---E-Commerce-Application.git

**12. Known Issues:**

* Price updates are not reflected immediately after applying discount codes.
* Some users experience lag when browsing large product categories.

**13. Future Enhancements:**

* Multiple Currencies: Support for currency conversion based on the user’s location.
* Product Recommendations: Use machine learning algorithms to recommend products based on user behavior.
* Social Media Integration: Allow users to share their shopping experiences on social platforms.