Project overview

# Introduction:

Shopez is an e-commerce platform that allows users to browse and purchase products online. Shopez is a leading e-commerce platform that offers a wide range of products at unbeatable prices. With a mission to provide a seamless and enjoyable shopping experience, we strive to make online shopping easy, convenient, and accessible to everyone. It is 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.

**Team Members**

1. **Ch. Abhiram**
2. **V. Harish**
3. **P. Revathi**
4. **K. Satya**

## Wide Product Range:

Competitive Pricing: We provide competitive pricing and discounts to ensure that our customers get the best value for their money.

Wide Product Range: We offer a vast selection of products across various categories, including electronics, fashion, home decor, and more.

Fast and Reliable Delivery: We partner with reliable logistics providers to ensure that our customers receive their orders quickly and efficiently.

Secure Payment Options: We offer secure payment options, including credit/debit cards, net banking, and cash on delivery.

## Why Choose Shoppez:

Convenience: Shop from the comfort of your own home, 24/7.

Wide Selection: Browse through our vast collection of products and find what you're looking for.

Competitive Pricing: Get the best value for your money with our competitive pricing and discounts.

Reliable Delivery: Receive your orders quickly and efficiently with our fast and reliable delivery service.

## Features:

Product Variety: Shoppez offers a wide range of products across various categories, including electronics, fashion, home decor, and more.

User-Friendly Interface: The platform provides an intuitive and easy-to-use interface, making it simple for customers to find and purchase products.

Secure Payment Options: Shoppez offers various secure payment options, including credit/debit cards, net banking, and cash on delivery.

Fast Delivery: The platform partners with reliable logistics providers to ensure fast and timely delivery of products.

## Key Features:

Zero Investment: Start selling online with zero investment.

Easy Registration: Register using your phone number and start selling.

Product Catalogs: Access catalogs of over 15 crore products.

Share and Earn: Share products with customers and earn commissions.

Social Media Integration: Share products on social media platforms.

## Target Audience:

Small Business Owners: Entrepreneurs and small business owners who want to start their online journey.

Individual Sellers: Individuals who want to sell products online and earn commissions.

## Availability:

Mobile App: Available as a mobile app for Android and iOS devices.

Website: Also accessible through the website.

## Goals:

### Primary Goals:

Democratize E-commerce: Make e-commerce accessible to everyone, especially small businesses and entrepreneurs.

Simplify Online Selling: Provide a simple and easy-to-use platform for individuals to start selling online.

Empower Small Businesses: Enable small businesses to reach a wider audience and grow their customer base.

### Secondary Goals

Increase Digital Adoption: Encourage more people to adopt digital commerce and online shopping.

Create New Income Opportunities: Provide individuals with new opportunities to earn income through online sales.

Expand Product Reach: Make a wide range of products available to customers across India.

### Long-term Goals:

Become a Leading E-commerce Platform: Establish Shopez as a leading e-commerce platform in India.

Enable Millions of Sellers: Enable millions of individuals and small businesses to start selling online through Shopez.

Contribute to Digital India: Contribute to the government's Digital India initiative by promoting digital commerce and online shopping.

## Functionalities:

### Core Functionalities

User Registration: Users can register on the platform using their phone numbers.

Product Catalogs: Users can access catalogs of over 15 crore products across various categories.

Product Sharing: Users can share products with customers via social media, messaging apps, and other channels.

Order Management: Users can manage orders, track status, and receive updates.

Commission Tracking: Users can track their commissions earned on sales.

### Product-Related Functionalities

Product Search: Users can search for products by name, category, or brand.

Product Filtering: Users can filter products by price, brand, and other criteria.

Product Details: Users can view detailed product information, including descriptions, images, and prices.

### Order-Related Functionalities

Order Placement: Users can place orders on behalf of customers.

Order Tracking: Users can track order status, including shipping and delivery updates.

Order Cancellation: Users can cancel orders, if necessary.

# Architecture

## Frontend:

The frontend is the user-facing part of the application, where customers interact with the platform.

### Technologies Used:

HTML/CSS/JavaScript: Core technologies for building the user interface.

React.js/Angular/Vue.js: Modern JavaScript frameworks for building dynamic and responsive UIs.

### Home Screen:

Header: Shopez logo, search bar, and navigation menu.

Hero Section: Showcase featured products, promotions, or categories.

Product Categories: Display top-level categories (e.g., Electronics, Fashion, Home).

Call-to-Action (CTA): Encourage users to start selling or explore products.

### Product Details Page:

Product Image: High-quality product image.

Product Information: Display product name, description, price, and specifications.

CTA: Add to cart, buy now, or share product.

Product Variations: Display available colors, sizes, or other variations.

### Order Management:

Order List: Display list of orders, including order ID, date, and status.

Order Details: View detailed order information, including products, pricing, and customer details.

Order Actions: Update order status, cancel order, or mark as shipped.

### Seller Dashboard:

Overview: Display seller performance metrics, including sales, commissions, and orders.

Product Management: Manage product listings, including adding, editing, and deleting products.

Order Management: Manage orders, including updating status and communicating with customers.

### User Profile:

Profile Information: Display user name, email, and phone number.

Order History: View list of past orders.

Commission History: View list of past commissions earned.

### Technical Requirements:

Responsive Design: Ensure a smooth user experience across various devices and screen sizes.

Fast Loading Times: Optimize images, code, and database queries for fast page loads.

Secure Payment Gateway: Integrate a secure payment gateway to protect user transactions.

Search Engine Optimization (SEO): Optimize meta tags, titles, and descriptions for improved search engine rankings.

### Frontend Frameworks and Libraries

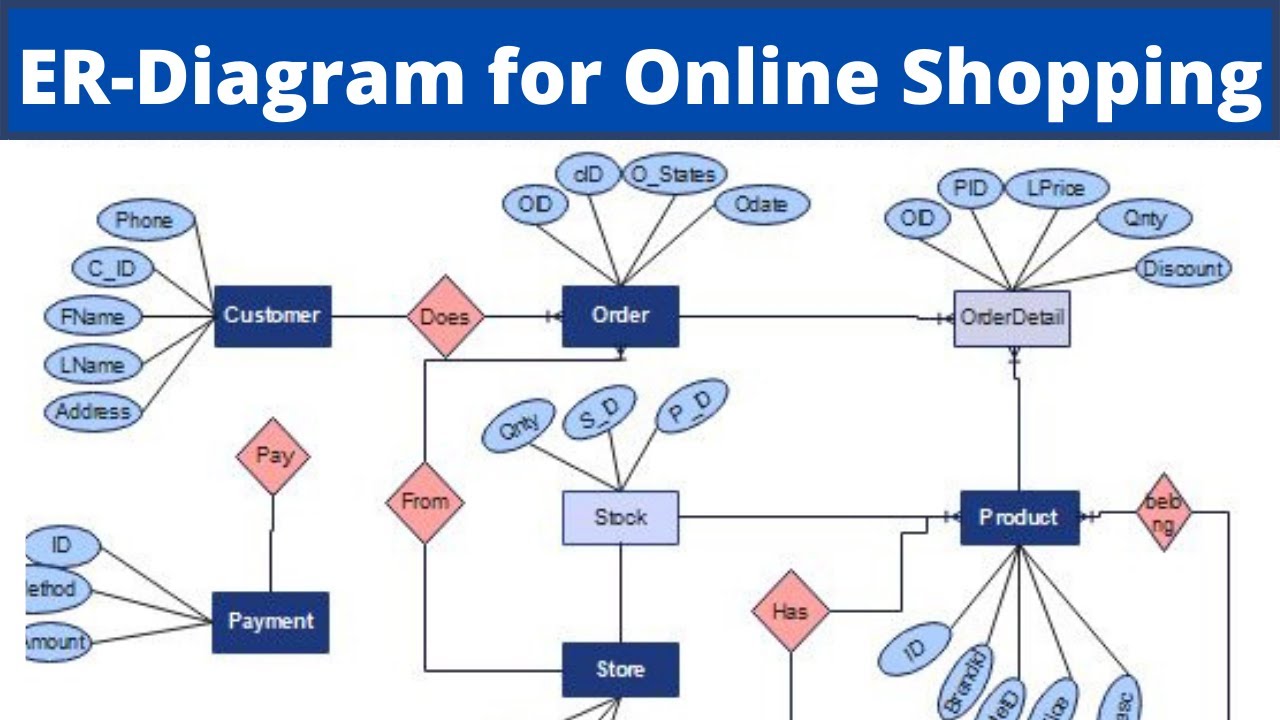
React: Utilize React for building reusable UI components.

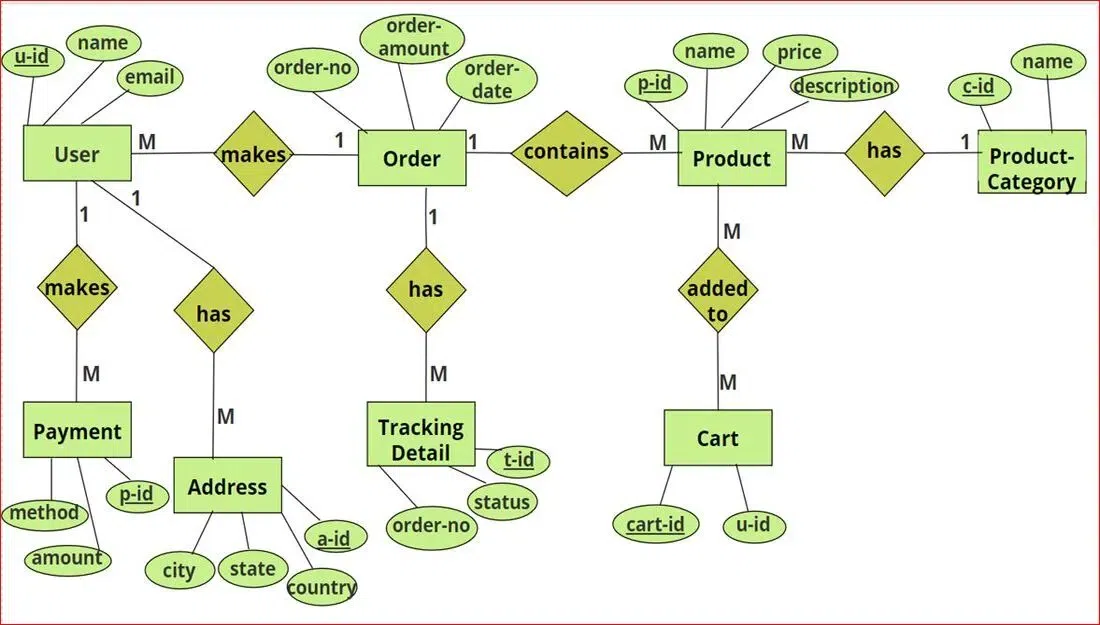
Redux: Manage global state with Redux.

Bootstrap: Use Bootstrap for responsive design and UI components.

Axios: Handle HTTP requests with Axios.

**ER Diagrams**





## Backend Frameworks and Libraries:

Node.js: Utilize Node.js as the server-side runtime environment.

Express.js: Build the backend API using Express.js.

MongoDB: Use MongoDB as the NoSQL database for storing user data, products, and orders.

Mongoose: Interact with MongoDB using Mongoose.

### API Endpoints

#### User Endpoints:

POST /users: Create a new user.

GET /users: Retrieve a list of all users.

GET /users/:id: Retrieve a specific user by ID.

PATCH /users/:id: Update a specific user by ID.

DELETE /users/:id: Delete a specific user by ID.

#### Endpoints:

POST /products: Create a new product.

GET /products: Retrieve a list of all products.

GET /products/:id: Retrieve a specific product by ID.

PATCH /products/:id: Update a specific product by ID.

DELETE /products/:id: Delete a specific product by ID.

#### Order Endpoints:

POST /orders: Create a new order.

GET /orders: Retrieve a list of all orders.

GET /orders/:id: Retrieve a specific order by ID.

PATCH /orders/:id: Update a specific order by ID.

DELETE /orders/:id: Delete a specific order by ID.

#### Database Schema:

##### Users Collection:

id (ObjectId)

name (String)

email (String)

phone (String)

password (String)

##### Products Collection:

id (ObjectId)

name (String)

description (String)

price (Number)

category (String)

##### Orders Collection:

id (ObjectId)

userId (ObjectId)

productId (ObjectId)

quantity (Number)

total (Number)

status (String)

### Authentication and Authorization:

JSON Web Tokens (JWT): Use JWT for authentication and authorization.

Password Hashing: Hash passwords using bcrypt.

### Payment Gateway Integration:

Razorpay: Integrate Razorpay as the payment gateway.

### Server-Side Rendering:

Handlebars: Use Handlebars for server-side rendering.

### Logging and Error Handling:

Winston: Use Winston for logging.

Error Handling Middleware: Implement error handling middleware to catch and handle errors.

# PREREQUISITES:

To develop a full-stack e-commerce app using React JS, Node.js, and MongoDB, there are several prerequisites you should consider. Here are the key prerequisites for developing such an application:

## Node.js and npm:

Install Node.js, which includes npm (Node Package Manager), on your development machine. Node.js is required to run JavaScript on the server side. • Download: https://nodejs.org/en/download/

* Installation instructions: https://nodejs.org/en/download/package-manager/

MongoDB:

Set up a MongoDB database to store hotel and booking information.

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

* Download: https://www.mongodb.com/try/download/community
* Installation instructions: https://docs.mongodb.com/manual/installation/

Express.js:

Express.js is a web application framework for Node.js. Install Express.js to handle server-side routing,middleware, and API development.

* Installation: Open your command prompt or terminal and run the following command: **npm install express**

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. To 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.

Front-end Framework:

Utilize Angular to build the user-facing part of the application, including product listings, booking forms, and user interfaces for the admin dashboard.

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 • Sublime Text: Download from https://www.sublimetext.com/download

WebStorm:

Download from <https://www.jetbrains.com/webstorm/download>

**To Connect the Database with Node JS go through the below provided link:** •

Link: https://www.section.io/engineering-education/nodejsmongoosejs-mongodb/

**To run the existing ShopEZ App project downloaded from github:**:

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/harsha-vardhan-reddy-07/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: **npm install**

Start the Development Server:

To start the development server, execute the following command: npm run dev or npm run start

The e-commerce app will be accessible at http://localhost:3000 by default. You can change the port configuration in the .env file if needed.

Access the App:

Open your web browser and navigate to http://localhost:3000.

You should see the flight booking app's homepage, indicating that the installation and setup were successful.

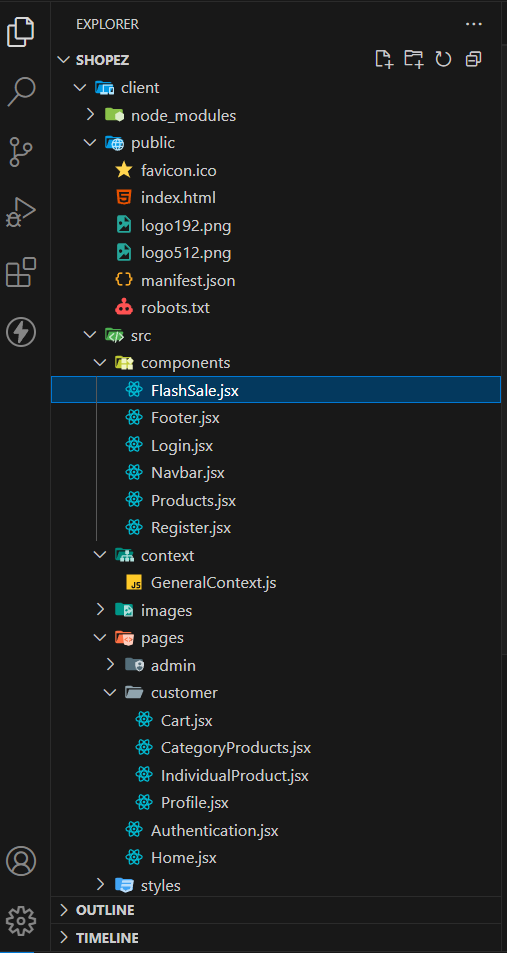
You have successfully installed and set up the ShopEZ app on your local machine. You can now proceed with further customization, development, and testing as needed.

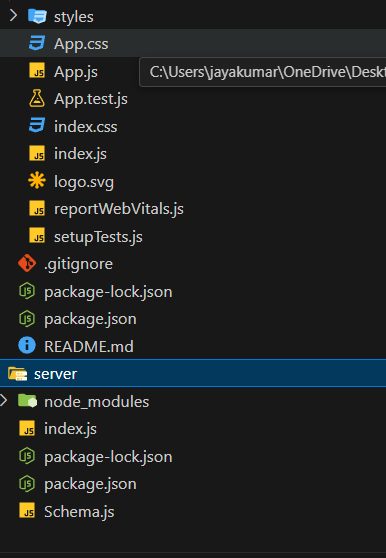
# PROJECT STRUCTURE:

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 Flow:

Milestone 1: Project Setup and Configuration:

Install required tools and software:

* Node.js.

Reference Article: <https://www.geeksforgeeks.org/installation-of-node-js-on-windows/>

* Git.

Reference Article: <https://git-scm.com/book/en/v2/Getting-Started-Installing-Git>

Create project folders and files:

• Client folders. • Server folders

Referral Video Link:

<https://drive.google.com/file/d/1uSMbPIAR6rfAEMcb_nLZAZd5QIjTpnYQ/view?usp=sharing>





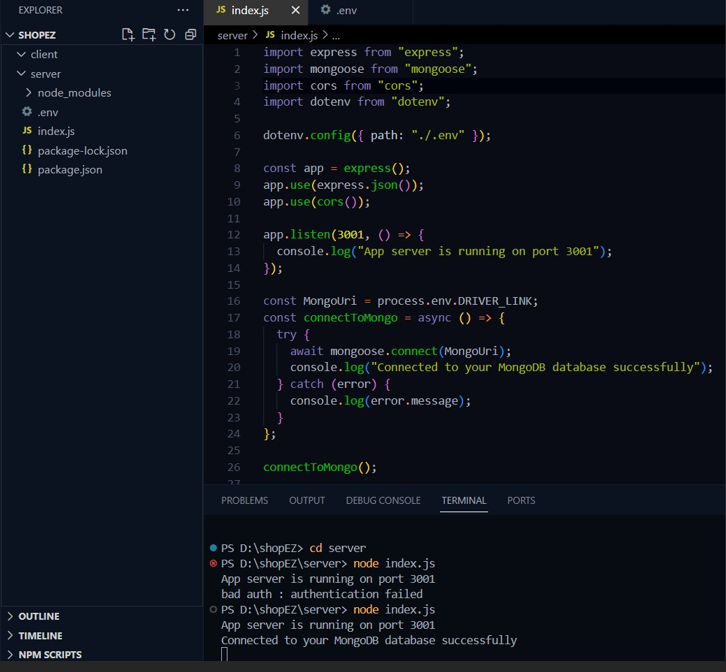
# 6. Configure MongoDB:

Create database in cloud video link:<https://drive.google.com/file/d/1CQil5KzGnPvkVOPWTLP0h-Bu2bXhq7A3/view>

Install Mongoose.

Create database connection.

Reference Video of connect node with mongoDB database: <https://drive.google.com/file/d/1cTS3_-EOAAvDctkibG5zVikrTdmoY2Ag/view?usp=sharing>Reference Article: <https://www.mongodb.com/docs/atlas/tutorial/connect-to-your-cluster/>



**Implement API endpoints:**

• Implement CRUD operations.

* Test API endpoints.

# Backend:

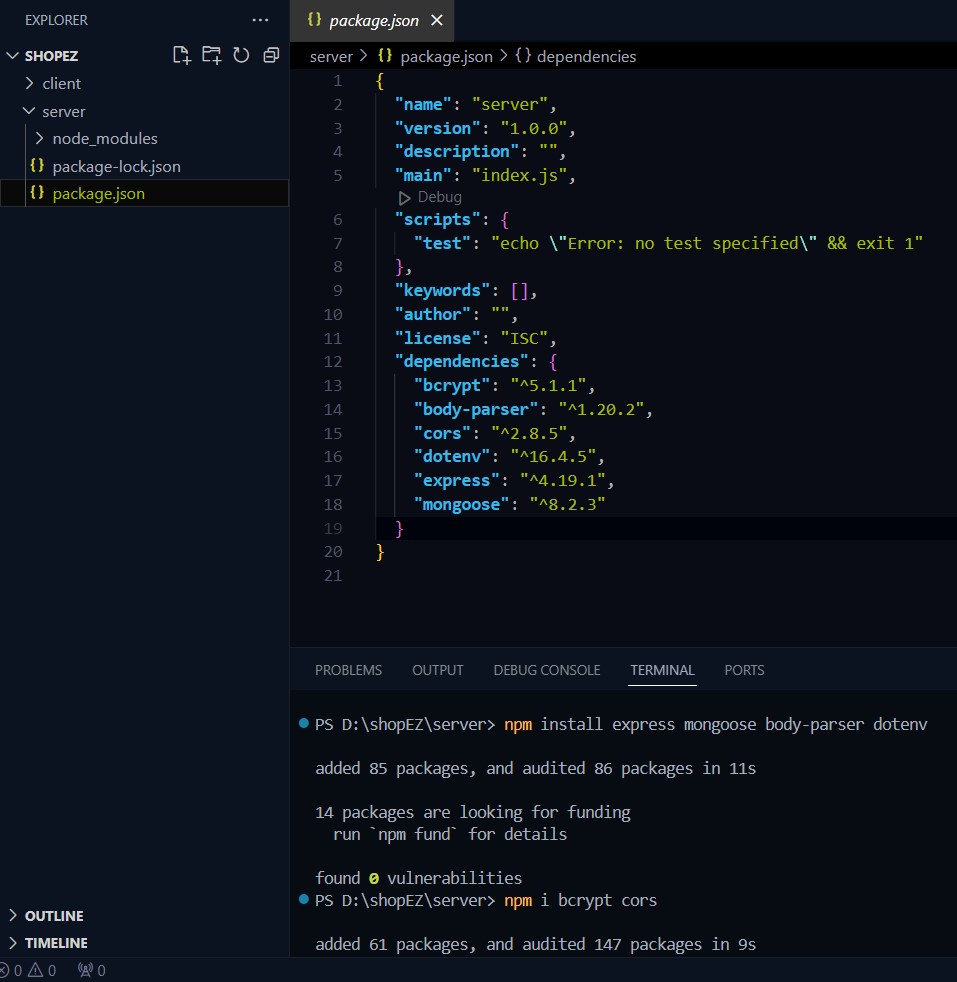
Set Up Project Structure:

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-gQK3DO6wr7ooAfJYIQwnem ZoF/view?usp=sharing](https://drive.google.com/file/d/19df7NU-gQK3DO6wr7ooAfJYIQwnemZoF/view?usp=sharing) Reference Image:



Database Configuration:

* + Set up a MongoDB database either locally or using a cloud-based MongoDB service like MongoDB Atlas or use locally with MongoDB compass.
  + Create a database and define the necessary collections for admin, users, products, orders and other relevant data.

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 for handling cross-origin requests.

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 orders.
  + Create corresponding Mongoose models to interact with the MongoDB database.
  + Implement CRUD operations (Create, Read, Update, Delete) for each model to perform database operations.

User Authentication:

* + Create routes and middleware for user registration, login, and logout. • Set up authentication middleware to protect routes that require user authentication.

Handle new products and Orders:

* + Create routes and controllers to handle new product listings, including fetching products data from the database and sending it as a response.
  + Implement ordering(buy) functionality by creating routes and controllers to handle order requests, including validation and database updates.
  + Admin Functionality:
    - Implement routes and controllers specific to admin functionalities such as adding products, managing user orders, etc.
    - Add necessary authentication and authorization checks to ensure only authorized admins can access these routes.

Error Handling:

* + - Implement error handling middleware to catch and handle any errors that occur during the API requests.
    - Return appropriate error responses with relevant error messages and HTTP status codes.

7.API DOCUMENTATION

ShopEZ is a comprehensive e-commerce application developed using the MERN stack (MongoDB, Express.js, React.js, Node.js). The backend exposes several RESTful API endpoints to manage various resources. Here's a documentation of the primary endpoints:

**User Endpoints:**

* **POST /api/users/register**: Registers a new user.
* **POST /api/users/login**: Authenticates a user and returns a token.
* **GET /api/users/profile**: Retrieves the profile of the authenticated user.
* **PUT /api/users/profile**: Updates the profile of the authenticated user.
* **GET /api/users/:id**: Fetches details of a user by ID.
* **DELETE /api/users/:id**: Deletes a user by ID.

**Product Endpoints:**

* **GET /api/products**: Retrieves a list of all products.
* **POST /api/products**: Creates a new product (admin only).
* **GET /api/products/:id**: Fetches details of a product by ID.
* **PUT /api/products/:id**: Updates a product by ID (admin only).
* **DELETE /api/products/:id**: Deletes a product by ID (admin only).

**Order Endpoints:**

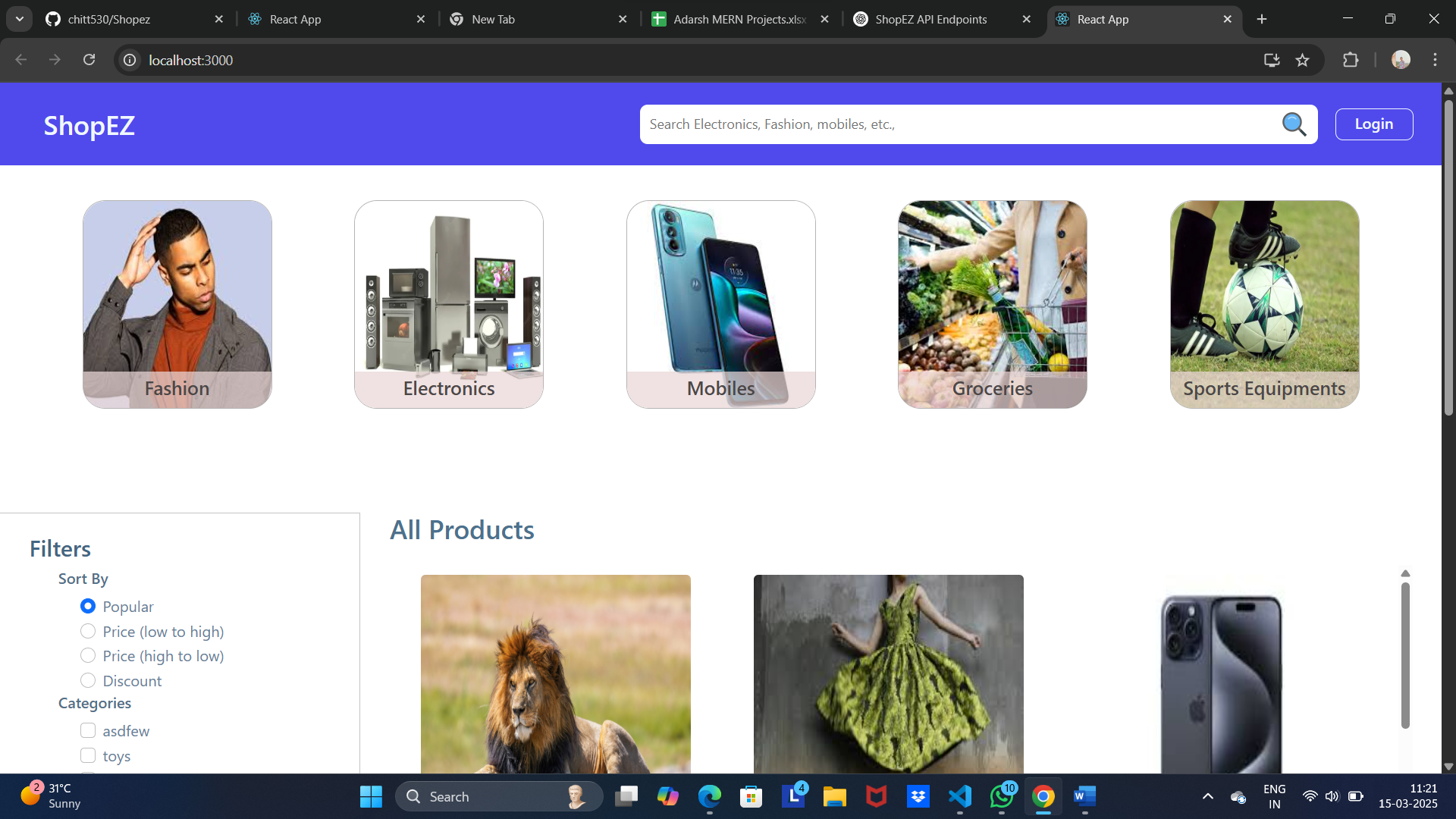
* **POST /api/orders**: Creates a new order.
* **GET /api/orders/:id**: Retrieves details of an order by ID.
* **GET /api/orders/user/:userId**: Fetches all orders for a specific user.
* **PUT /api/orders/:id/pay**: Marks an order as paid.
* **PUT /api/orders/:id/deliver**: Marks an order as delivered (admin only).

**Cart Endpoints:**

* **GET /api/cart**: Retrieves the current cart of the authenticated user.
* **POST /api/cart**: Adds an item to the cart.
* **PUT /api/cart/:itemId**: Updates the quantity of a cart item.
* **DELETE /api/cart/:itemId**: Removes an item from the cart.
* **Admin Endpoints:**
* **GET /api/admin/users**: Retrieves a list of all users (admin only).
* **GET /api/admin/orders**: Fetches all orders (admin only).
* **GET /api/admin/products**: Retrieves all products (admin only).

These endpoints facilitate the core functionalities of the ShopEZ platform, enabling both user and administrative operations.

**User interface**

****

**1. Principles of User Interface Design**

UI design is governed by several fundamental principles:

**a. Clarity**

* The interface should be clear and easy to understand.
* Text, icons, and visual elements should be self-explanatory.

**b. Consistency**

* UI elements should behave predictably across different screens and applications.
* Consistent color schemes, fonts, and layouts enhance usability.

**c. Feedback**

* The system should provide immediate responses to user actions (e.g., button clicks, form submissions).
* Visual cues like animations, sound, or tooltips can improve feedback.

**d. Efficiency**

* The interface should allow users to perform tasks quickly and with minimal effort.
* Features like shortcuts and predictive text improve efficiency.

**e. Affordance**

* Elements should suggest their function (e.g., buttons look clickable, sliders look draggable).

**f. Accessibility**

* UI should be inclusive for users with disabilities (e.g., screen readers, color contrast adjustments).
* Supports different input methods (keyboard, mouse, touch, voice).

**2. UI Components**

**a. Input Controls**

* Buttons, text fields, checkboxes, radio buttons, dropdowns

**b. Navigation Components**

* Menus, breadcrumbs, search bars, pagination

**c. Informational Components**

* Tooltips, notifications, progress bars, modals

**d. Containers**

* Cards, grids, sections, accordions

**3. UI Design Patterns**

Common UI design patterns that improve usability:

* **Direct Manipulation:** Drag-and-drop, resizing, zooming
* **Skeuomorphism vs. Flat Design:** Realistic elements vs. minimalist aesthetics
* **Mobile-First Design:** Prioritizing mobile usability before scaling to larger screens
* **Dark Mode & Light Mode:** Providing different themes for user preference

**4. UI Evaluation Methods**

* **User Testing:** Observing real users interacting with the interface
* **Heuristic Evaluation:** Checking against established usability principles
* **A/B Testing:** Comparing different UI versions to find the most effective one

1. **a. Functional Testing**

Ensures that all features work as expected.

* User authentication (login, registration, password reset)
* Product search and filtering
* Shopping cart functionality
* Payment gateway integration
* Order processing and tracking

1. **b. Performance Testing**

Evaluates the speed and stability of the system under various conditions.

* Load Testing: How the site handles multiple users at once
* Stress Testing: How the site performs under extreme conditions
* Response time for page loads and transactions

1. **c. Security Testing**

Ensures user data and transactions are secure.

* SSL/TLS encryption verification
* SQL injection and cross-site scripting (XSS) checks
* Secure payment processing compliance (PCI-DSS)
* User session security

1. **d. Usability Testing**

Checks if the shopping experience is user-friendly.

* Intuitive navigation and search
* Checkout process simplicity
* Mobile responsiveness
* Accessibility for disabled users

1. **e. Integration Testing**

Ensures smooth interaction between different modules.

* Payment gateway communication
* Inventory management updates
* API connections with third-party services (e.g., shipping providers)

1. **f. Regression Testing**

Ensures that new updates do not break existing functionality.

* Re-run critical test cases after updates
* Ensure previous bugs remain fixed

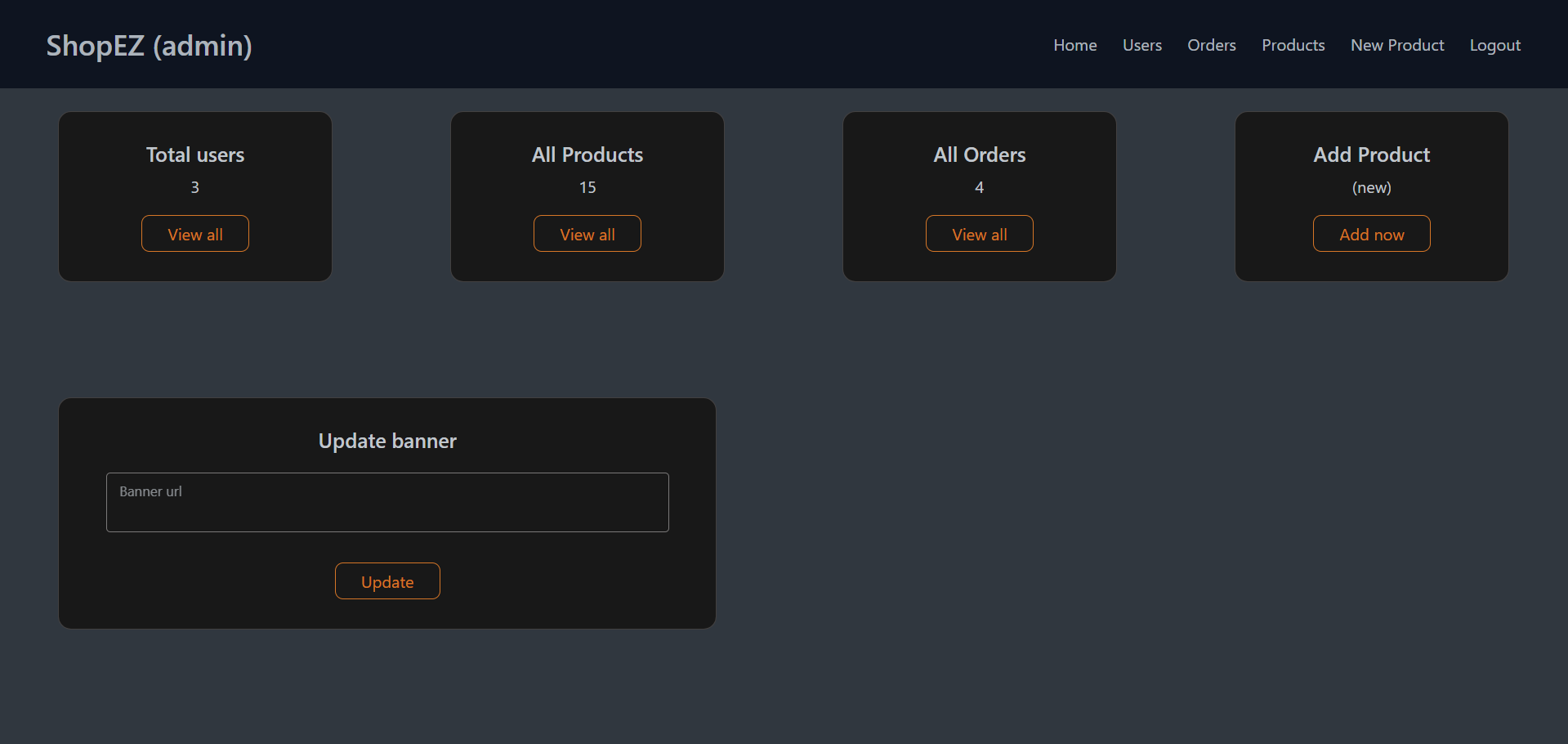
1. **g. A/B Testing**

Compares two versions of a feature to optimize conversion rates.

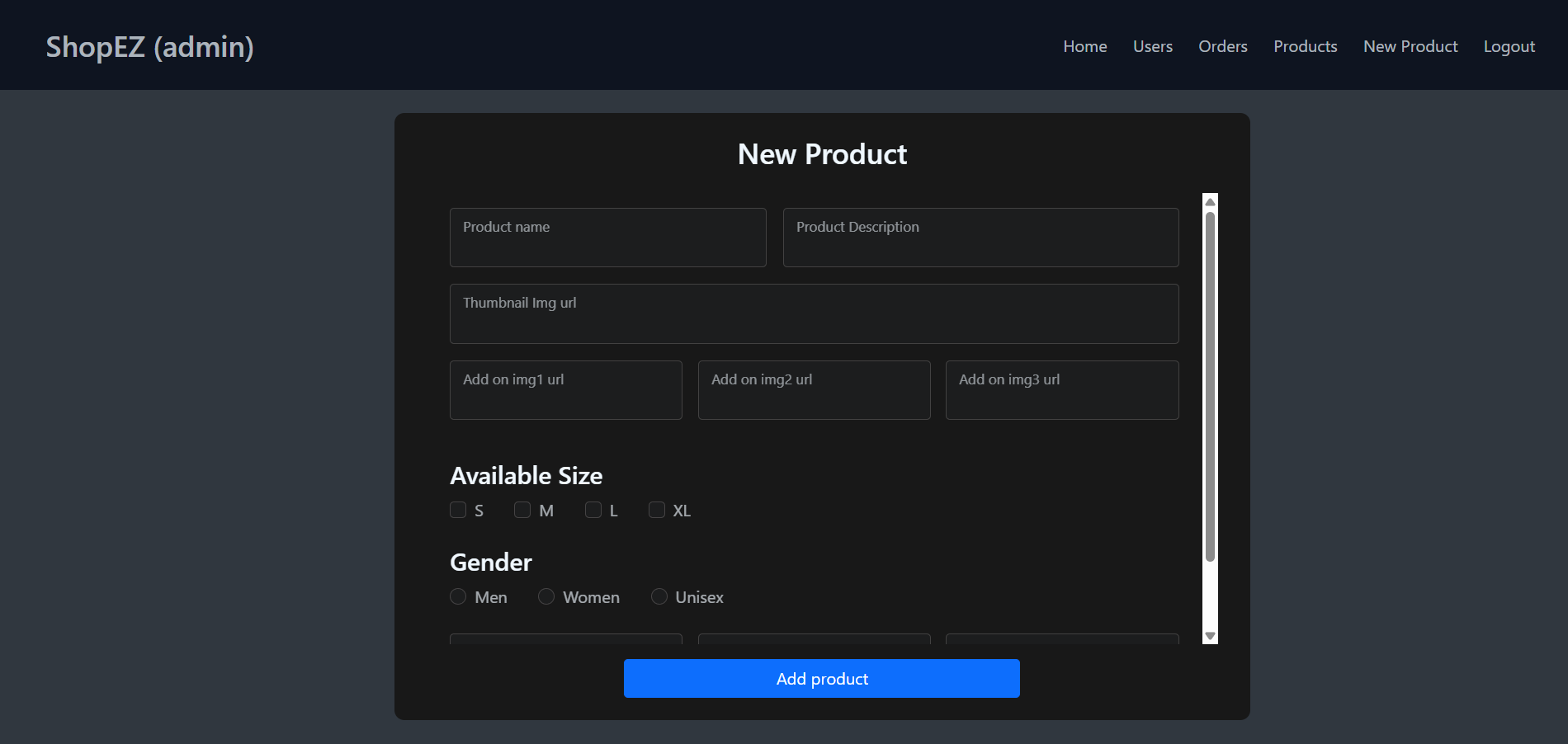
* Different checkout flows to improve sales
* UI layout variations for better engagement

**Demo**

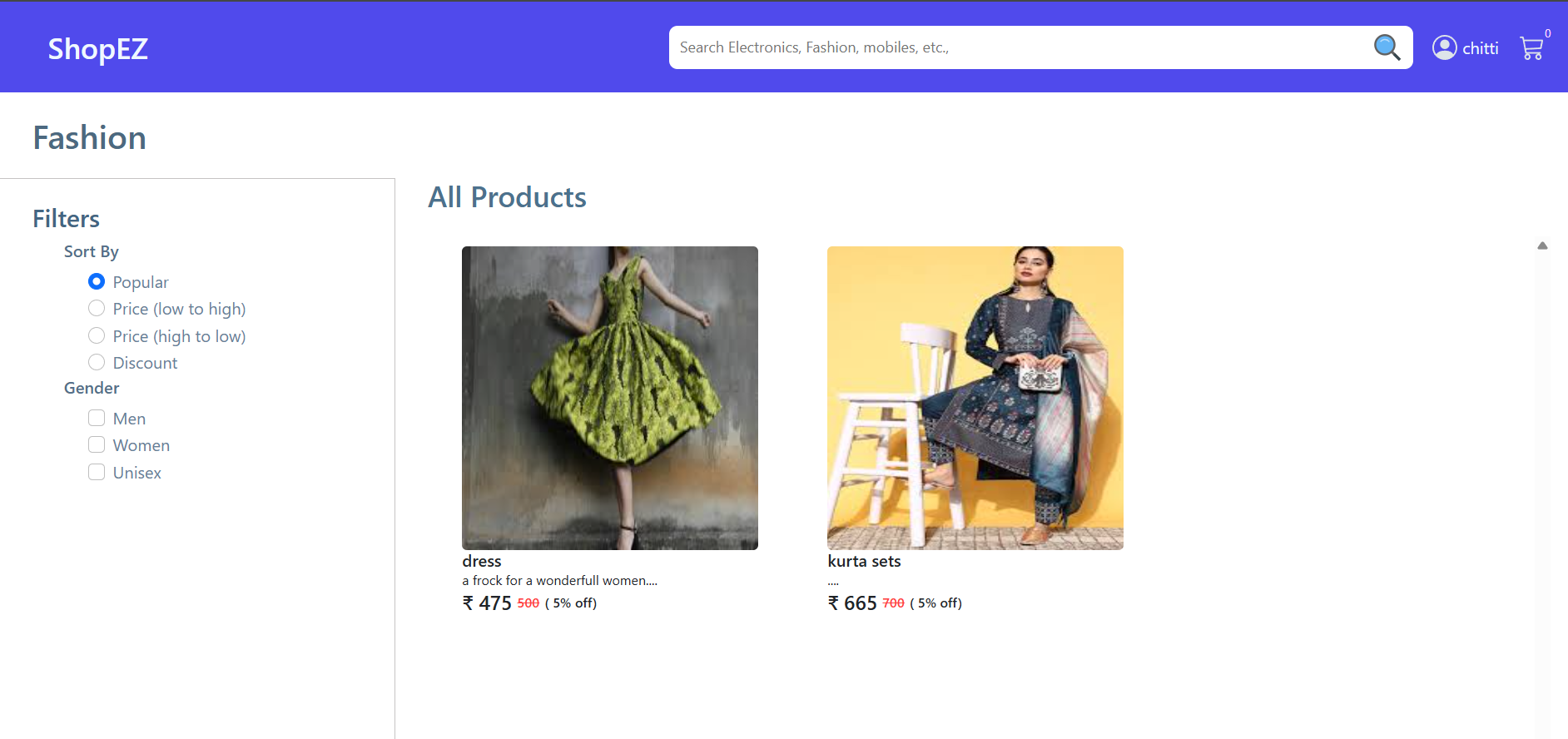
**Admin Home**

****

**Admin Product**

****

**User Interface**

****

**Know Issues**

1. Functional Issues

🔸 Cart and Checkout Failures – Items disappear from the cart, or users experience errors during checkout.  
🔸 Payment Gateway Errors – Transactions fail due to incorrect API integrations or payment provider downtime.  
🔸 Search and Filtering Bugs – Users may not find relevant products due to poor search algorithms or broken filters.  
🔸 Inventory Mismatch – Products shown as "In Stock" are actually unavailable due to outdated stock data.  
🔸 Order Tracking Issues – Delayed or missing tracking updates from logistics providers.

2. Performance Issues

🔸 Slow Page Load Times – Large images, unoptimized scripts, and poor caching can slow down the site.  
🔸 Downtime During High Traffic – Websites crash during peak sales (e.g., Black Friday) due to inadequate server resources.  
🔸 Latency in Processing Orders – Slow backend processing can delay order confirmation and shipping.

3. Security Issues

🔸 Data Breaches & Leaks – Poor security practices may expose customer data (e.g., emails, passwords, payment details).  
🔸 Fraudulent Transactions – Fake orders, stolen credit cards, or bot-based purchases can harm businesses.  
🔸 Weak Authentication – Lack of strong password policies or multi-factor authentication (MFA) makes accounts vulnerable.  
🔸 SQL Injection & XSS Attacks – Poorly sanitized inputs can allow hackers to manipulate the system.

4. Usability & UI/UX Issues

🔸 Complex Checkout Process – Long, multi-step checkouts increase cart abandonment rates.  
🔸 Non-Responsive Design – The website doesn’t adapt well to mobile devices or different screen sizes.  
🔸 Inconsistent Navigation – Users struggle to find key features like account settings, order history, or return policies.  
🔸 Lack of Accessibility – Missing features like screen reader support, proper color contrast, or keyboard navigation for disabled users.

5. Integration & API Issues

🔸 Payment Gateway Failures – API downtime or incorrect configurations can block transactions.  
🔸 Shipping Provider Integration Errors – Tracking data is not updated due to API failures.  
🔸 Third-Party Plugin Conflicts – Conflicting extensions or updates can break existing functionalities.

6. Customer Support Issues

🔸 Delayed Response from Support – Slow ticket resolutions lead to negative customer experiences.  
🔸 Automated Bots with Poor Responses – Chatbots may fail to provide relevant answers or escalate issues properly.  
🔸 Refund & Return Process Delays – Long refund processing times frustrate customers.

Future Enhancement

1. **Faster & Smarter Delivery Options**

🔹 **Drone & Autonomous Vehicle Deliveries** – Faster shipping with minimal human intervention.  
🔹 **Same-Day & One-Hour Delivery** – Optimized logistics for ultra-fast deliveries.  
🔹 **Smart Lockers & Pickup Stations** – Secure pickup points for convenient order collection

1. **Web3 & Decentralized Shopping Experiences**

🔹 **NFT-Based Digital Goods** – Selling digital collectibles and fashion.  
🔹 **Metaverse Shopping Malls** – Virtual stores where users can explore and buy products in a 3D space.  
🔹 **Decentralized Marketplaces** – Peer-to-peer shopping without intermediaries.