ShopEZ: E-commerce Application

VIT BHOPAL

Team Members	REGISTRATION NO
Sneha yadav	**22BEY10089**
Shivani Gupta	**22BCG10164**
V. Bala Gayatri	**22BEY10035**
Yash Raj Singh	**22BEY10085**

TABLE OF CONTENT

1. INTRODUCTION

1.1 Project Overview

ShopEZ is a full-featured e-commerce web application developed using the MERN stack (MongoDB, Express.js, React.js, and Node.js). The platform is designed to deliver a seamless and efficient online shopping experience for customers while offering robust tools for sellers and administrators. ShopEZ enables users to browse, search, and purchase products with ease, while sellers can manage their inventory and track orders through an intuitive dashboard. Administrators are provided with powerful controls to oversee platform activity, ensuring a secure and well-regulated marketplace.

1.2 Purpose

The primary purpose of ShopEZ is to create a scalable, responsive, and user-centric e-commerce solution that caters to the needs of all stakeholders—customers, sellers, and administrators. By leveraging modern web technologies, ShopEZ aims to bridge the gap between functionality and user experience, ensuring a smooth shopping process, efficient seller operations, and simplified administrative management. The project also serves as a practical implementation of modular development and real-time interactivity within a modern online marketplace.

2. IDEATION PHASE

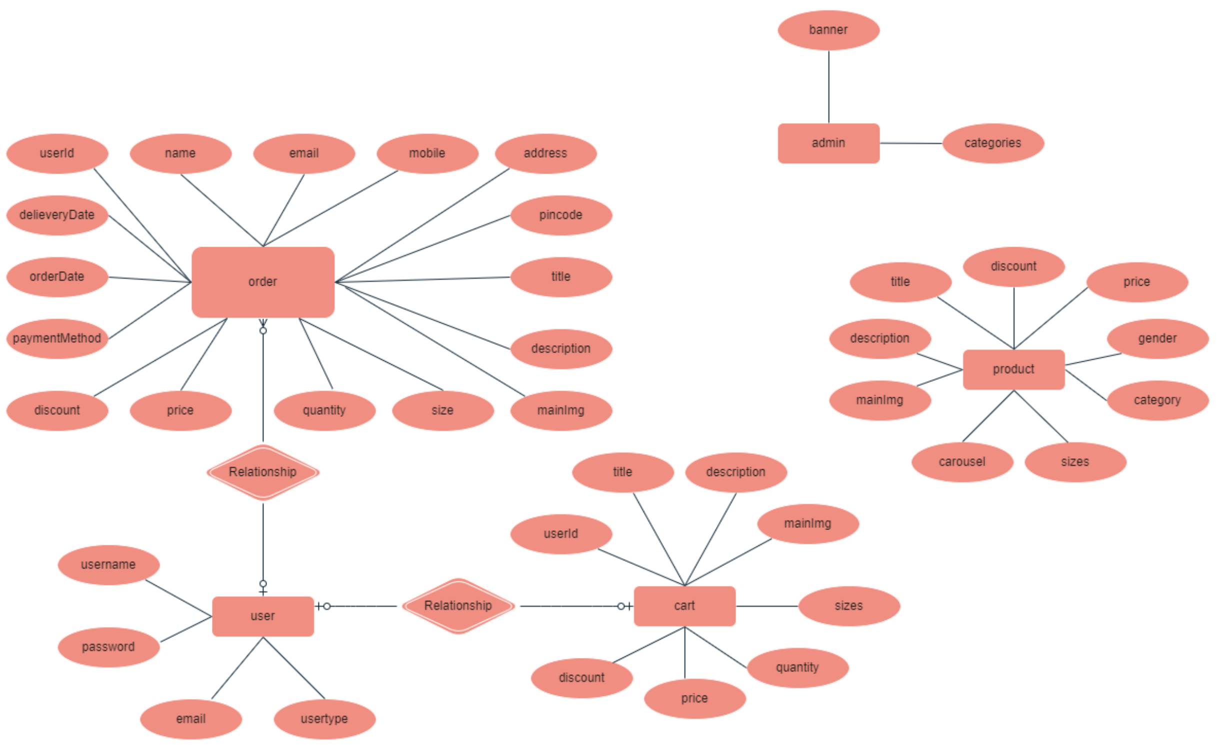
2.1 Problem Statement

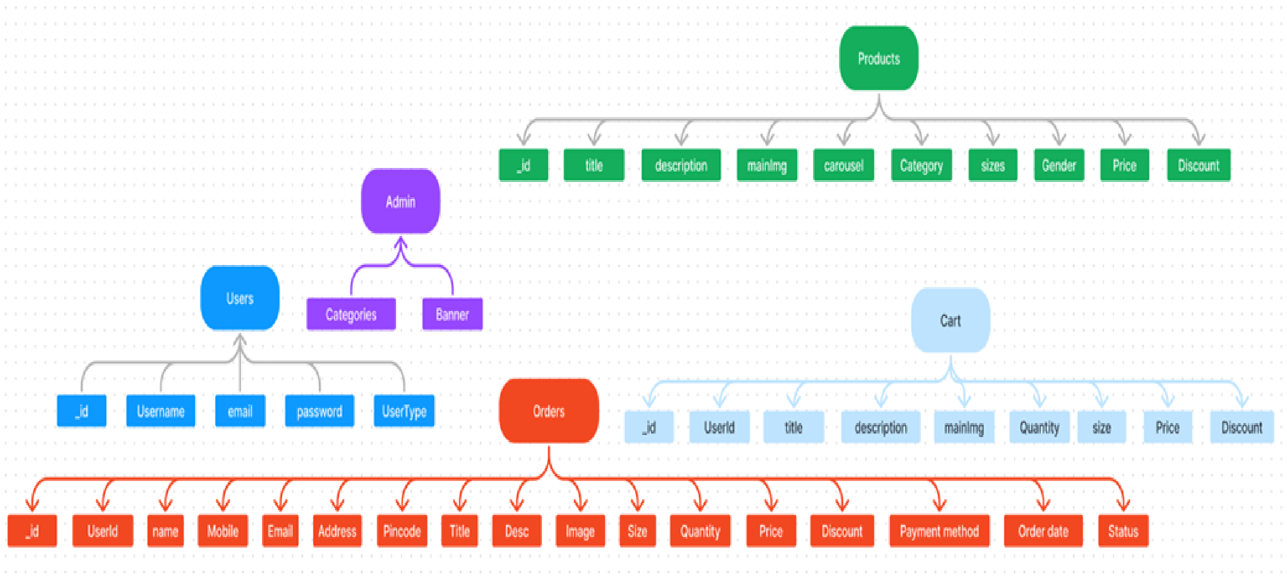
In today's fast-paced digital world, many e-commerce platforms struggle with either limited functionality for sellers or poor user experience for customers. Small to mid-level sellers often lack access to streamlined tools for managing inventory and tracking sales. Meanwhile, customers face issues such as cluttered interfaces, confusing checkout processes, and lack of trust in platform security. Additionally, administrators find it difficult to maintain order and oversee platform integrity effectively. There is a need for a comprehensive, scalable, and user-friendly platform that caters equally to buyers, sellers, and administrators.

2.2 Empathy Map Canvas

To better understand user needs, an Empathy Map Canvas was developed focusing on three user personas—Customer, Seller, and Admin:

ER-MODEL





The **ShopEZ ER-diagram** represents the entities and relationships involved in a modern e-commerce system. It visualizes how users, products, carts, and orders are interlinked, providing clarity on data flow and management. Below is a breakdown of the core entities and their relationships:

- **USER**: Represents individuals who are registered on the platform. Each user can browse products, add items to their cart, and place orders.
- **Admin**: Represents administrative users responsible for managing categories, banners, and overall platform settings. Admins also have visibility and control over product listings and user activity.
- **Products**: Stores details about all the products available on the platform. This includes product name, description, price, image, and category.
- **Cart**: This collection contains the products added by users to their shopping cart. Each cart entry is uniquely identified by the associated user ID.
- **Orders**: Maintains a record of all purchases made by users. Each order is tied to a specific user and contains details such as product information, quantity, shipping address, and payment status.

Key Features:

1. **Comprehensive Product Catalog**:

   ShopEZ offers a wide range of products for users to explore. Each product includes detailed descriptions, customer reviews, pricing, and discounts, making it easier for users to make informed decisions.

2. **"Shop Now" Button**:

   Every product listing includes a clearly visible "Shop Now" button. When clicked, users are directed to the order page to proceed with their purchase.
3. **Order Details Page**:

   After selecting a product, users are taken to an order page where they can input shipping details, choose a payment method, and include any specific preferences related to the order.
4. **Secure and Efficient Checkout**:

   ShopEZ ensures a fast and secure checkout process. User data is handled with care and encrypted to maintain privacy, while minimizing delays during order completion.
5. **Order Confirmation and Tracking**:

Once an order is placed, users receive a confirmation notification. They can then access a detailed view of their order, including status updates, shipping information, and payment details.

In addition to enhancing the buyer experience, ShopEZ equips **administrators** with a powerful backend system to oversee the platform. Admins can:

- Add, edit, or remove products.
- Manage product categories and homepage banners.
- Monitor user activity and order history.
- Ensure smooth and secure functioning of the platform.

ShopEZ is designed to offer a **seamless, secure, and enjoyable shopping experience.** With its intuitive user interface, structured product flow, and efficient admin controls, the platform serves both users and admins in a scalable and reliable manner.

2.3 Brainstorming

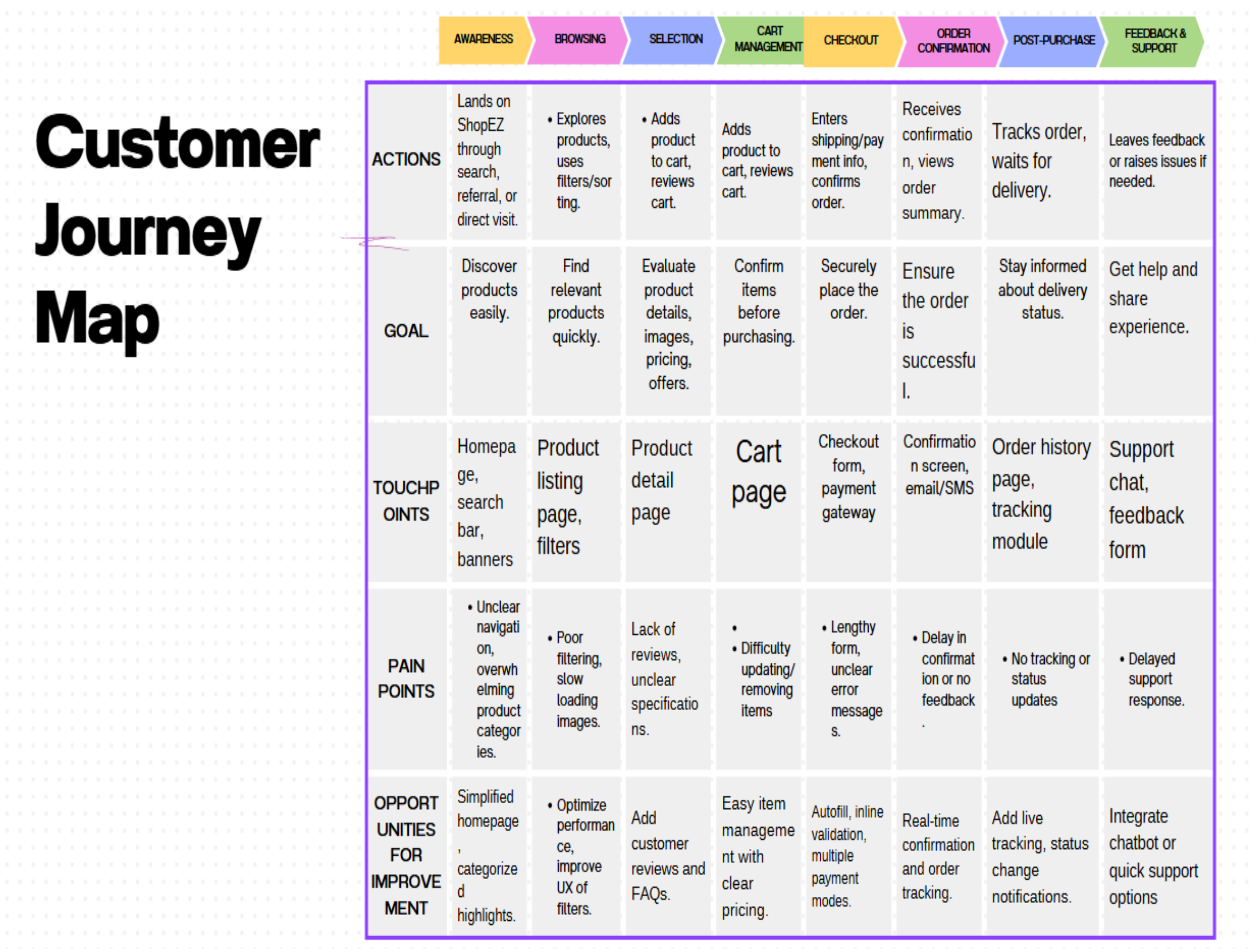
In the brainstorming phase, various ideas were generated to shape ShopEZ into a user-friendly and efficient e-commerce platform. We explored the core needs of buyers, such as smooth product browsing, secure checkout, order tracking, and intuitive user flow. At the same time, we identified administrative requirements like product management, user monitoring, and overall control of the platform.

We mapped out potential features including a dynamic product catalog, cart system, simplified order placement, and a secure authentication system. Emphasis was placed on creating a responsive UI, ensuring data consistency through MongoDB, and implementing seamless communication between frontend and backend using RESTful APIs. This stage helped align the team’s vision and establish a clear development roadmap for ShopEZ.

3. REQUIREMENT ANALYSIS

3.1 Customer Journey Map

The customer journey in ShopEZ is designed to be smooth and intuitive. It begins with a user visiting the platform, browsing the product catalog, selecting desired items, and adding them to the cart. After reviewing the cart, the user proceeds to the checkout, where they enter shipping and payment details. Post-payment, the user receives confirmation and can view their order details. At each stage, the system ensures clarity, responsiveness, and security to enhance the user experience.



Customer Journey Map

	AWARENESS	BROWSING	SELECTION	CART MANAGEMENT	CHECKOUT	ORDER CONFIRMATION	POST-PURCHASE	FEEDBACK & SUPPORT
ACTIONS	Lands on ShopEZ through search, referral, or direct visit.	• Explores products, uses filters/sorting.	• Adds product to cart, reviews cart.	Adds product to cart, reviews cart.	Enters shipping/payment info, confirms order.	Receives confirmation, views order summary.	Tracks order, waits for delivery.	Leaves feedback or raises issues if needed.
GOAL	Discover products easily.	Find relevant products quickly.	Evaluate product details, images, pricing, offers.	Confirm items before purchasing.	Securely place the order.	Ensure the order is successful.	Stay informed about delivery status.	Get help and share experience.
TOUCHPOINTS	Homepage, search bar, banners	Product listing page, filters	Product detail page	Cart page	Checkout form, payment gateway	Confirmation screen, email/SMS	Order history page, tracking module	Support chat, feedback form
PAIN POINTS	• Unclear navigation, overwhelming product categories.	• Poor filtering, slow loading images.	Lack of reviews, unclear specifications.	• • Difficulty updating/removing items	• Lengthy form, unclear error messages.	• Delay in confirmation or no feedback.	• No tracking or status updates	• Delayed support response.
OPPORTUNITIES FOR IMPROVEMENT	Simplified homepage, categorized highlights.	• Optimize performance, improve UX of filters.	Add customer reviews and FAQs.	Easy item management with clear pricing.	Autofill, inline validation, multiple payment modes.	Real-time confirmation and order tracking.	Add live tracking, status change notifications.	Integrate chatbot or quick support options

3.2 Solution Requirement

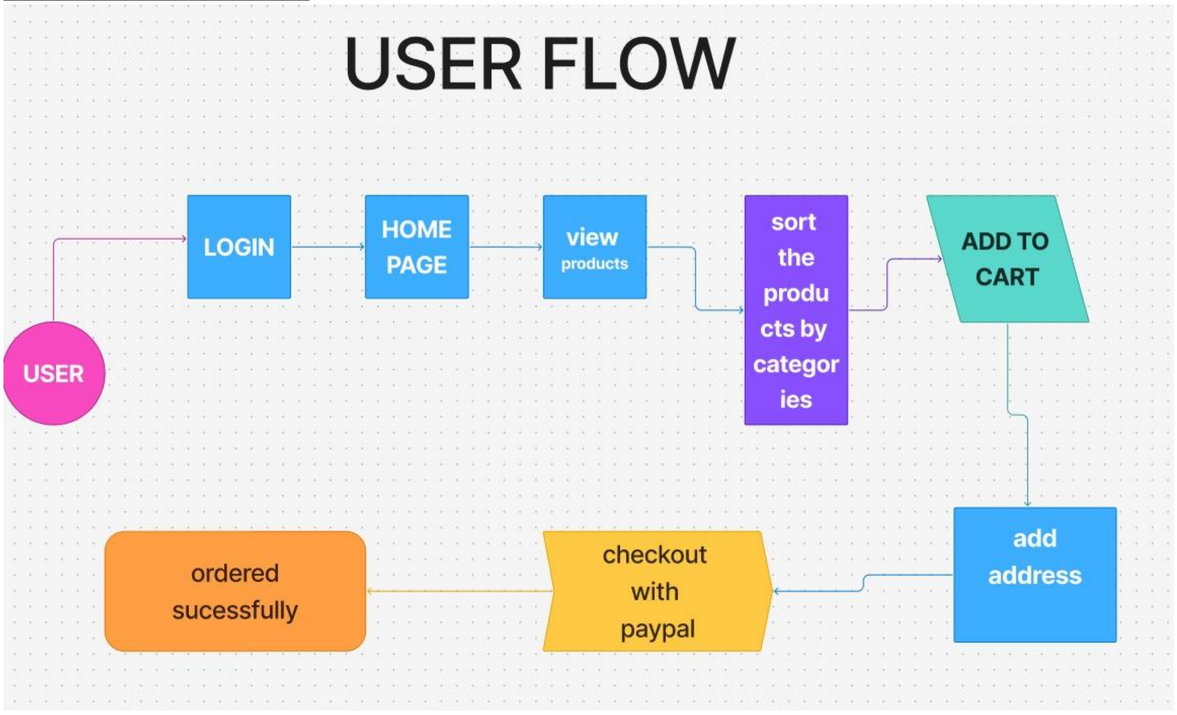
To deliver a seamless experience for buyers and admins, ShopEZ requires:

- A responsive user interface for browsing and ordering products.
- A secure login and authentication mechanism.
- Real-time cart and order management.
- An admin dashboard for managing product listings, categories, and user records.
- A backend system to handle data processing, validation, and API integration.
- Reliable data storage to manage users, products, orders, and cart data.

3.3 Data Flow Diagram

The **ShopEZ** application is designed to accommodate three primary user roles: **Customer**, and **Admin**. Each role has specific responsibilities and permissions, managed through dedicated user flows and API endpoints. Below is an overview of the application flow for each role.

Customer Flow



This diagram represents the **user flow** of an e-commerce platform—from login to placing an order. Here's a step-by-step explanation of the process:

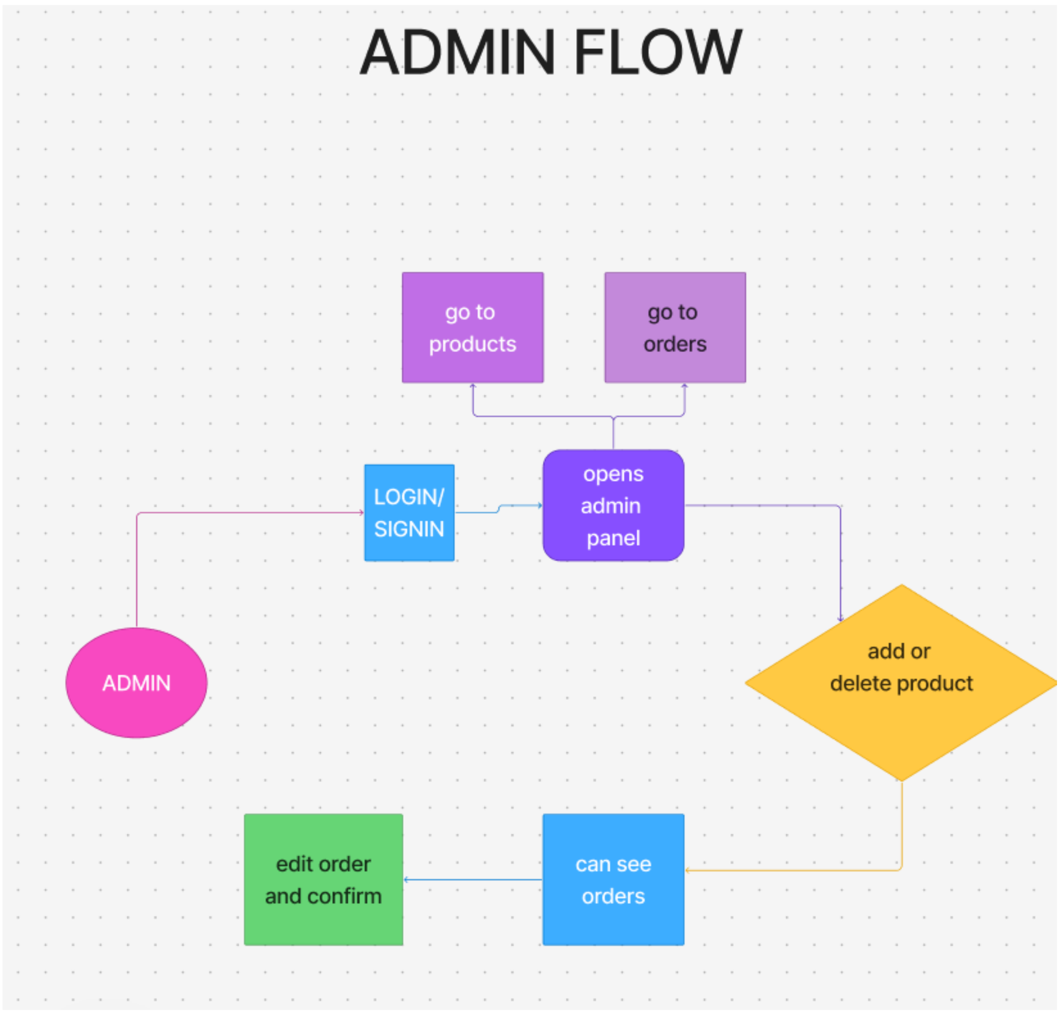
1. **USER**
   - The user initiates the process.
2. **LOGIN**
   - The user logs into their account.
3. **HOME PAGE**
   - After login, they are taken to the homepage of the store.
4. **View Products**
   - From the homepage, the user browses through the products.
5. **Sort the Products by Categories**
   - The user has the option to filter or sort products by categories to narrow down their search.
6. **ADD TO CART**
   - After finding a product, the user adds it to their cart.
7. **Add Address**
   - Before checking out, the user inputs or selects a shipping address.
8. **Checkout with PayPal**
   - The user proceeds to the payment stage and chooses PayPal as the method of payment.
9. **Ordered Successfully**
   - Once the payment is done, the order is confirmed and successfully placed.

Observations:

- The flow is mostly linear but includes a decision step (sorting products).
- It assumes PayPal is the only checkout method.
- The process ends with a success message after checkout.

.

1. **<u>Admin Flow</u>**



1. **ADMIN**
   - The process starts with the admin initiating the workflow.
2. **LOGIN/SIGNIN**
   - Admin logs into the system.
3. **Opens Admin Panel**
   - After a successful login, the admin is directed to the admin panel.

From the Admin Panel, there are 2 main paths:

1. Product Management Path

- **Go to Products** → The admin can manage the products listed.
- **Add or Delete Product** → Admin can either add new products or remove existing ones.

2. Order Management Path

- **Go to Orders** → Leads to viewing all placed orders.
- **Can See Orders** → Admin can browse through all orders.
- **Edit Order and Confirm** → Admin has the option to edit order details and confirm them for processing.

Platform Management:

- Admins can monitor all activities within the application, including user engagement, order volumes, and sales data.
- They ensure that platform operations run smoothly and address any technical or logistical issues that may arise.
- Admins have the power to update platform settings, adjust fees, and oversee security protocols to protect user data and maintain the integrity of the application.

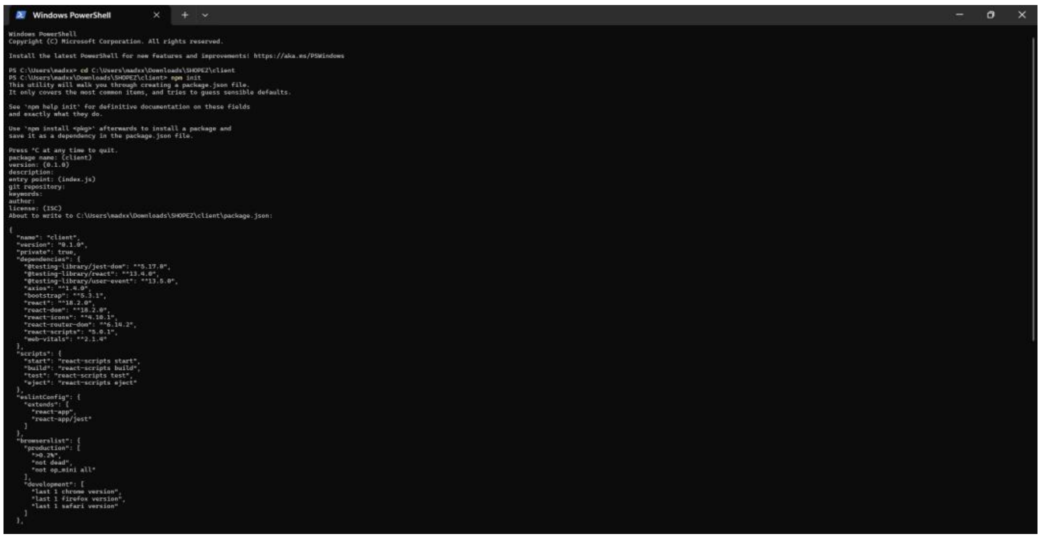
3.4 Technology Stack

To develop the **ShopEZ** e-commerce application using Node.js, Express.js, MongoDB, and React.js, here are the essential prerequisites:

Key Prerequisites

- **Node.js and npm**:
  Node.js is a runtime environment that enables JavaScript to run server-side, offering scalability and efficiency for network applications.
  - **Download**: Node.js

- **Installation Instructions**: Node.js Installation Guide
- Run npm init to set up the project dependencies.

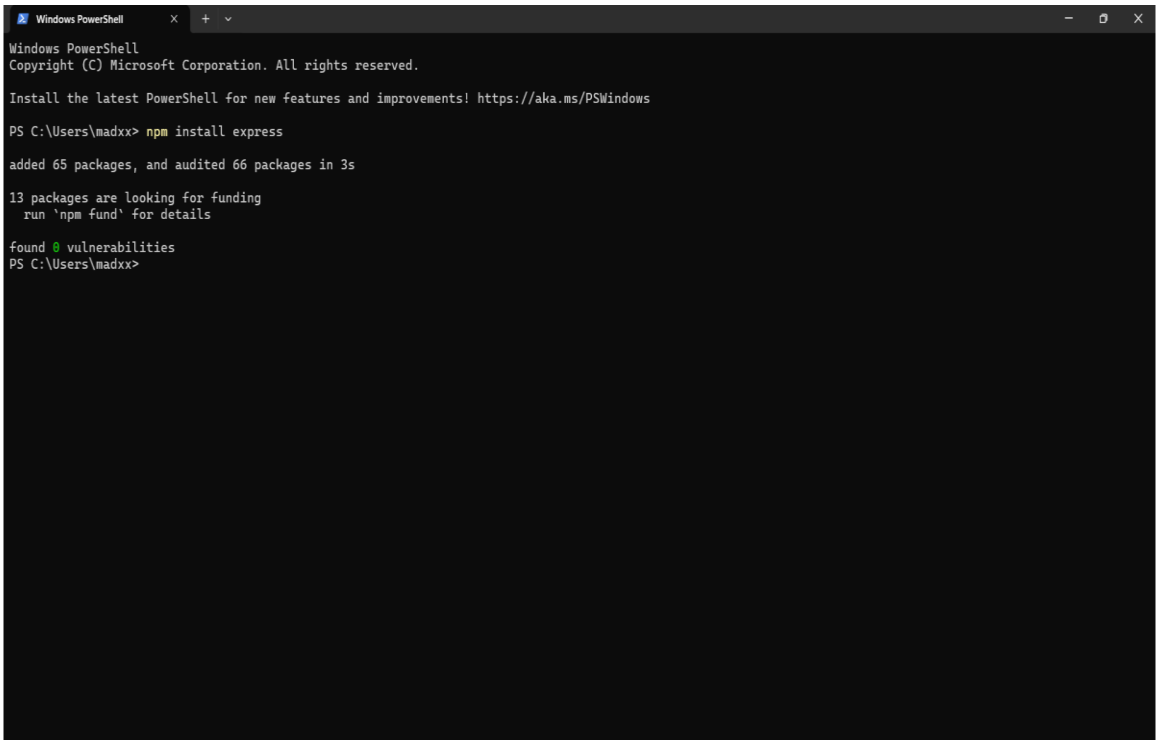


```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Users\madxx> cd C:\Users\madxx\Downloads\SHOPEZ\client
PS C:\Users\madxx\Downloads\SHOPEZ\client> npm init
This utility will walk you through creating a package.json file.
It only covers the most common items, and tries to guess sensible defaults.

See `npm help init` for definitive documentation on these fields
and exactly what they do.

Use `npm install <pkg>` afterwards to install a package and
save it as a dependency in the package.json file.

Press ^C at any time to quit.
package name: (client)
version: (0.1.0)
description:
entry point: (index.js)
git repository:
keywords:
author:
license: (ISC)
About to write to C:\Users\madxx\Downloads\SHOPEZ\client\package.json:

{
  "name": "client",
  "version": "0.1.0",
  "private": true,
  "dependencies": {
    "@testing-library/jest-dom": "^5.17.0",
    "@testing-library/react": "^13.4.0",
    "@testing-library/user-event": "^13.5.0",
    "axios": "^1.4.0",
    "bootstrap": "^5.3.1",
    "react": "^18.2.0",
    "react-dom": "^18.2.0",
    "react-icons": "^4.10.1",
    "react-router-dom": "^6.14.2",
    "react-scripts": "5.0.1",
    "web-vitals": "^2.1.4"
  },
  "scripts": {
    "start": "react-scripts start",
    "build": "react-scripts build",
    "test": "react-scripts test",
    "eject": "react-scripts eject"
  },
  "eslintConfig": {
    "extends": [
      "react-app",
      "react-app/jest"
    ]
  },
  "browserslist": {
    "production": [
      ">0.2%",
      "not dead",
      "not op_mini all"
    ],
    "development": [
      "last 1 chrome version",
      "last 1 firefox version",
      "last 1 safari version"
    ]
  },
```

Express.js:

Express.js is a lightweight and efficient web framework for Node.js that simplifies the creation of APIs and web applications, supporting routing, middleware, and a modular structure.

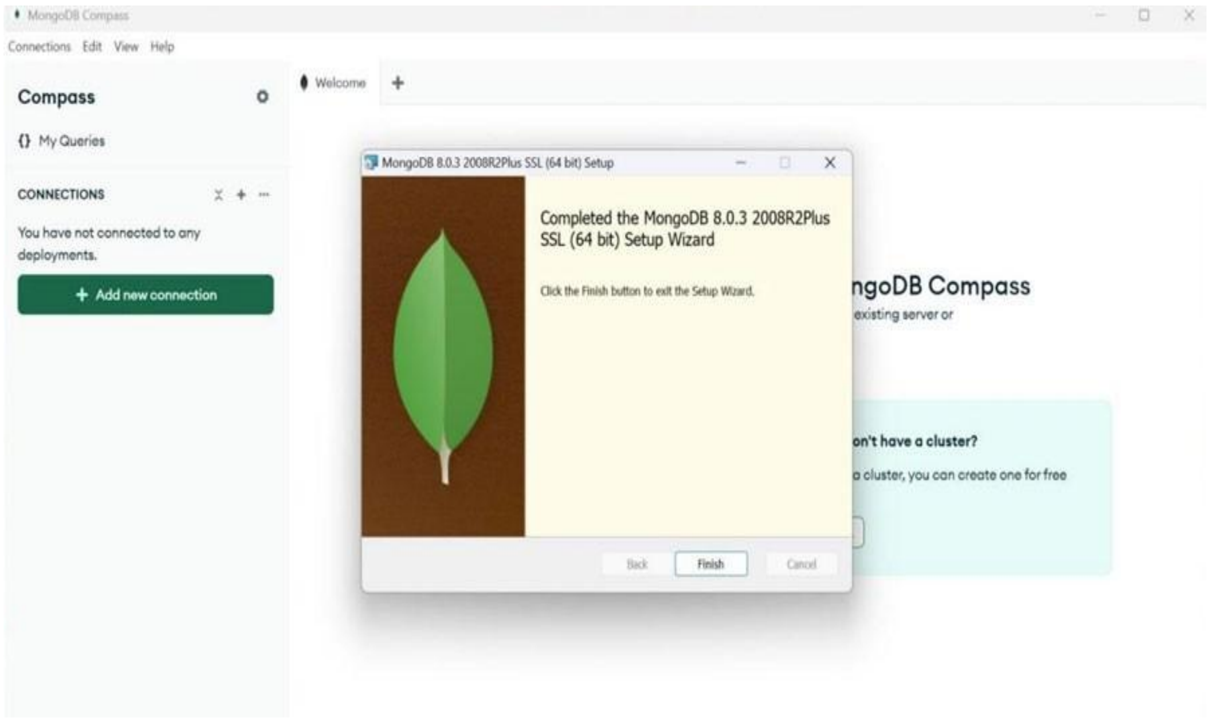
- **Installation**: Open your terminal and run: npm install express



```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Users\madxx> npm install express

added 65 packages, and audited 66 packages in 3s

13 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
PS C:\Users\madxx>
```

- **MongoDB**:
  MongoDB is a NoSQL database ideal for handling diverse data types, offering scalability and flexibility with JSON-like storage. It's highly compatible with Node.js, making it suitable for large- scale applications.
  - **Download**: MongoDB Community Server

Installation Instructions: MongoDB Installation Guide



MongoDB 8.0.3 2008R2Plus SSL (64 bit) Setup

Completed the MongoDB 8.0.3 2008R2Plus SSL (64 bit) Setup Wizard

Click the Finish button to exit the Setup Wizard.

You have not connected to any deployments.

- **React.js**:
  React.js is a widely used JavaScript library for building interactive user interfaces. It enables reusable UI components and dynamic web applications.

- **HTML, CSS, and JavaScript**:
  Basic knowledge of HTML, CSS, and JavaScript is essential for structuring, styling, and enabling interactivity on the client-side.

- **Database Connectivity**:
  Use a MongoDB driver or an ODM like Mongoose to connect the Node.js server with MongoDB for CRUD operations.

- **Additional UI Libraries**:
  Utilize Material-UI and Bootstrap for enhanced styling and responsive components.

Install dependencies:

Client:

cd
client
npm
install

Serve
r: *cd*
server
npm
install

Start the Development Server

- To launch the development server, execute:
  npm start
- Access the **ShopEZ** app at http://localhost:5173/auth/login by default.

4. PROJECT DESIGN

4.1 Problem-Solution Fit

Traditional e-commerce platforms often suffer from complex navigation, slow performance, and poor user feedback integration. These issues can make the shopping process frustrating for users and difficult to manage for platform admins. There is a clear need for a lightweight, scalable, and intuitive e-commerce solution that delivers a smooth experience for buyers while simplifying platform management for administrators.

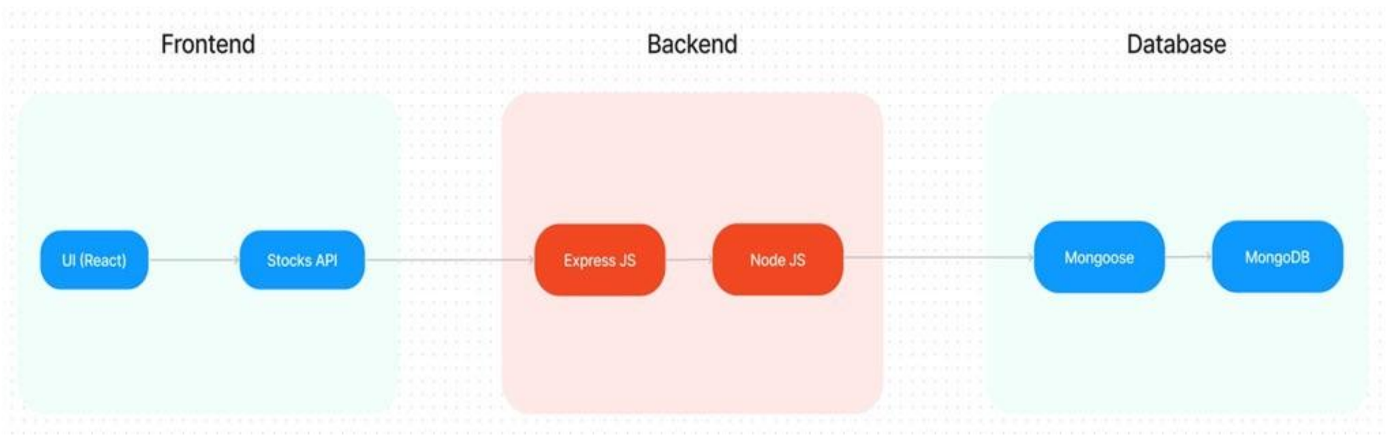
ShopEZ addresses these gaps by focusing on speed, clarity, and user-friendly interfaces, offering a modern shopping platform that ensures buyers can find, purchase, and track products easily. Admins are empowered with a centralized dashboard for managing listings, users, and orders effectively.

4.2 Proposed Solution

The proposed solution, ShopEZ, is a full-stack e-commerce web application built using the **MERN stack (MongoDB, Express.js, React.js, and Node.js)**. It provides:

- A responsive and intuitive **buyer interface** with product search, filters, cart, and a secure checkout process.
- A robust **admin panel** to manage users, products, and orders.
- Seamless **frontend-backend integration**, ensuring real-time updates and efficient data flow.
- Enhanced **data validation and security** to protect user data and transaction integrity.

4.3 Solution Architecture



The technical architecture of the **ShopEZ** application follows a client-server model, with the front-end serving as the client and the back-end as the server. This structure facilitates efficient data flow and responsive interactions for users browsing and purchasing products.

- **Client**:
  The Client, developed in React, handles user interface elements and product presentation. Using libraries like Axios, the front-end communicates seamlessly with the back-end through RESTful APIs. Additionally, the frontend incorporates Material-UI and Bootstrap to create a polished, responsive UI that enhances the user experience across various devices.

- **Server**:
  The server, built with Express.js, manages server-side logic and

database interactions. MongoDB is used for data storage, offering a flexible, scalable solution for handling structured and unstructured data, including user details, product information, and order records.

- **Data Exchange**:
  RESTful APIs facilitate communication between the front-end and back-end, ensuring smooth data retrieval and updates. This modular approach also enables flexibility in expanding or modifying the system in the future.
- **Authentication and Security**:
  JWT (JSON Web Tokens) are implemented to secure user sessions, ensuring role-based access for customers, sellers, and admins. This token-based authentication ensures that only authorized

  users can access restricted actions, such as managing product listings or viewing sensitive order details.
- **Real-Time Updates**:
  The platform provides real-time notifications, such as instant order confirmations and status updates, enhancing user engagement and transparency. For sellers, the dashboard includes analytics on sales trends and customer behavior, supporting data-driven decisions.
- **Third-Party Integrations**:
  Payment gateways are integrated for secure transactions, supporting multiple payment methods for a convenient checkout process. Additionally, analytics tools are incorporated into the admin dashboard to visualize metrics on product engagement, sales performance, and customer demographics.

The **ShopEZ** e-commerce application is structured into **frontend** and **backend** directories, following a modular approach for better scalability and maintainability. This structure organizes components by functionality, supports role-based features, and simplifies code management for both developers and administrators.

Root Folders and Files

public:

Contains static files such as index.html which is the root HTML file for your application. You can also place images, icons, and manifest files here that should be directly accessible by the browser.

src:

The main source folder where your React app's components, logic, styles, and routes are defined. This is where you'll spend most of your development time.

At the root of your project, you have several important configuration files:

- **.env**: This holds sensitive keys like your database URL or secret keys. You should never upload this to GitHub.
- **.gitignore**: Lists files and folders that Git should ignore (e.g., node_modules, .env).
- **components.json**: Probably used for managing or documenting shared UI components (not very common—depends on your setup).
- **eslint.config.js**: Configuration for ESLint, which checks your JavaScript code for bugs and formatting issues.
- **index.html**: The main HTML file used by Vite to serve your React app.
- **jsconfig.json**: Helps your editor (like VS Code) understand JavaScript paths and modules.
- **package.json**: This is your project's "manifest" — it lists dependencies, scripts, and other metadata.
- **package-lock.json**: Automatically generated to lock the versions of all

installed packages.

- **README.md**: Your project documentation.
- **tailwind.config.js**: Configuration file for customizing Tailwind CSS.
- **vite.config.js**: Configuration file for Vite (a fast frontend build tool you're using instead of Webpack).

Frontend Structure

The frontend is built using **React.js** and is housed within the client folder. This directory contains all the files and components required for the user interface of ShopEZ.

Inside the src/ folder, here's what each part does:

- **assets**/: Static files like images, logos, icons, etc.
- **components**/: Reusable UI components grouped by feature or purpose:
  - **admin-view**/: Components for admin dashboard or tools.
  - **auth**/: Components for login, signup, and authentication logic.
  - **shopping-view**/: Components for the shopping experience (e.g., product listings, cart).
  - **common**/: Reusable components like buttons, loaders, cards.
  - **ui**/: Possibly lower-level UI components like modals, dropdowns, form elements.
- **config**/: Config files for constants like base API URLs, app settings, themes.
- **lib**/: Contains logic like API utilities, custom hooks, formatting functions.
- **pages**/: Each folder here maps to a page or route in your app:
  - admin-view/, auth/, not-found/, shopping-view/, unauth-page/
- **store**/: State management, likely using Redux Toolkit:
  - auth-slice.js, admin.js, common-slice.js, etc.

    - store.js ties all of your slices into one Redux store.

- **App.jsx:** The root component that sets up routes and renders your pages.
- **main.jsx:** The entry point where React renders the app into the DOM.
- **App.css and index.css**: Tailwind or custom global styles.

Backend Structure

The backend is built using **Node.js** and **Express.js** to handle server-side logic, data management, and API requests. This part of the project is contained within the server folder.

Folder and File Structure

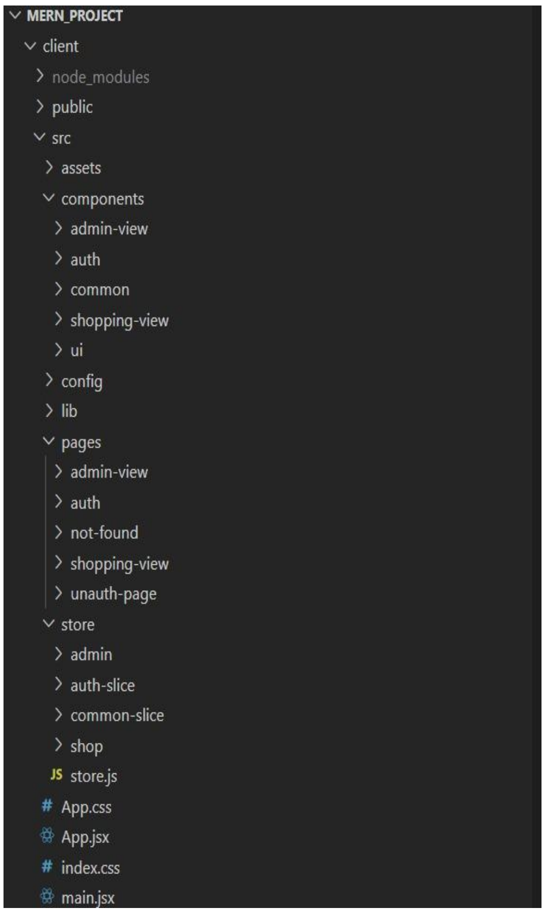
- **controllers**/: Contains logic for handling requests. Each file typically corresponds to a route (e.g., handling product data, user data, etc.).
- **helpers**/: Utility functions used across the server, like data formatting, token creation, error handling, etc.
- **models**/: Mongoose schemas that define how your MongoDB documents look (e.g., User, Product).
- **routes**/: This folder defines API endpoints and connects them to the corresponding controller functions.
- **server.js**: This is the main file that starts your backend server, connects to MongoDB, sets up middleware, and loads routes.

You also have a separate package.json and node_modules/ specific to your backend.

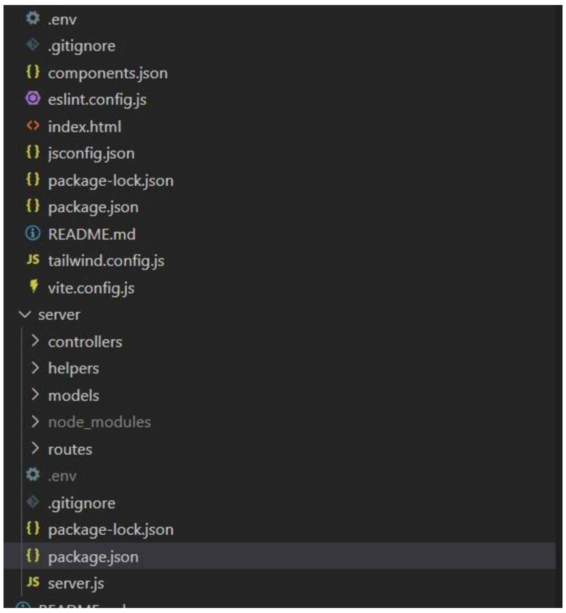
.

Entry Points and Package Files

- **index.js**: The main entry point for the backend server. It initializes the server, connects to the MongoDB database, sets up middleware, and loads routes for handling API requests.
- **package.json**: Contains metadata for the backend project, including dependencies, scripts, and project information. It manages the backend libraries and tools required to run the server.
- **package-lock.json**: Ensures consistency by locking the exact versions of dependencies installed in node_modules.



```
MERN_PROJECT
  client
    node_modules
    public
    src
      assets
      components
        admin-view
        auth
        common
        shopping-view
        ui
      config
      lib
      pages
        admin-view
        auth
        not-found
        shopping-view
        unauth-page
      store
        admin
        auth-slice
        common-slice
        shop
        store.js
      App.css
      App.jsx
      index.css
      main.jsx
```



```
    .env
    .gitignore
    components.json
    eslint.config.js
    index.html
    jsconfig.json
    package-lock.json
    package.json
    README.md
    tailwind.config.js
    vite.config.js
  server
    controllers
    helpers
    models
    node_modules
    routes
    .env
    .gitignore
    package-lock.json
    package.json
    server.js
```

Supporting Files and Documentation

- **README.md**: Contains documentation for the project, including setup instructions, usage guidelines, and general information about the platform. This file is essential for helping developers and contributors understand and work with the codebase.

vite.config.js: The configuration file for Vite, a fast build tool and development server for the frontend. This file allows customization of Vite’s behavior, such as setting up plugins, defining alias paths, and configuring development and production environments.

5. PROJECT PLANNING & SCHEDULING

5.1 Project Planning

The development of **ShopEZ** followed a structured and phased approach to ensure smooth progress and timely completion. The planning process involved breaking down the entire project into manageable stages, each with specific goals and deliverables. Agile methodology was used to accommodate iterative development and incorporate feedback efficiently.

The project was divided into the following key phases:

- **Requirement Gathering**: Identifying user needs, defining features, and preparing design documents.
- **Design Phase**: Creating wireframes, mockups, UI components, and overall system architecture.
- **Development Phase**: Building the frontend and backend modules, setting up the database, and implementing APIs.
- **Integration & Testing**: Connecting modules, performing functionality tests, fixing bugs, and ensuring security.
- **Deployment**: Hosting the application on a cloud platform and ensuring accessibility.
- **Documentation & Final Review**: Preparing technical documentation and project reports, and reviewing the overall performance.

Each phase had predefined timelines and responsibilities assigned, ensuring accountability and consistent progress throughout the development cycle.

6. FUNCTIONAL AND PERFORMANCE TESTING

6.1 Performance Testing

Performance testing for **ShopEZ** was conducted to ensure that the platform functions efficiently under various conditions and maintains a smooth user experience even during peak loads. The primary goals were to assess response time, system stability, and resource usage.

Key areas tested:

- **Page Load Speed:** All core pages (Home, Product Listing, Cart, Checkout) were optimized for fast loading using React's virtual DOM and efficient component rendering.
- **API Response Time:** Backend endpoints were monitored to maintain quick responses under different data loads using optimized queries and Express middleware.
- **Database Operations:** MongoDB queries were indexed and optimized to ensure faster data retrieval and minimal latency in fetching user, product, and order data.
- **Concurrent Users:** The platform was tested with multiple simulated users to check consistency in performance and identify any bottlenecks or slowdowns.

Results indicated that the system consistently handled user actions like browsing, adding to cart, and placing orders within acceptable response times. Performance remained stable, ensuring reliability and scalability for real-world use.

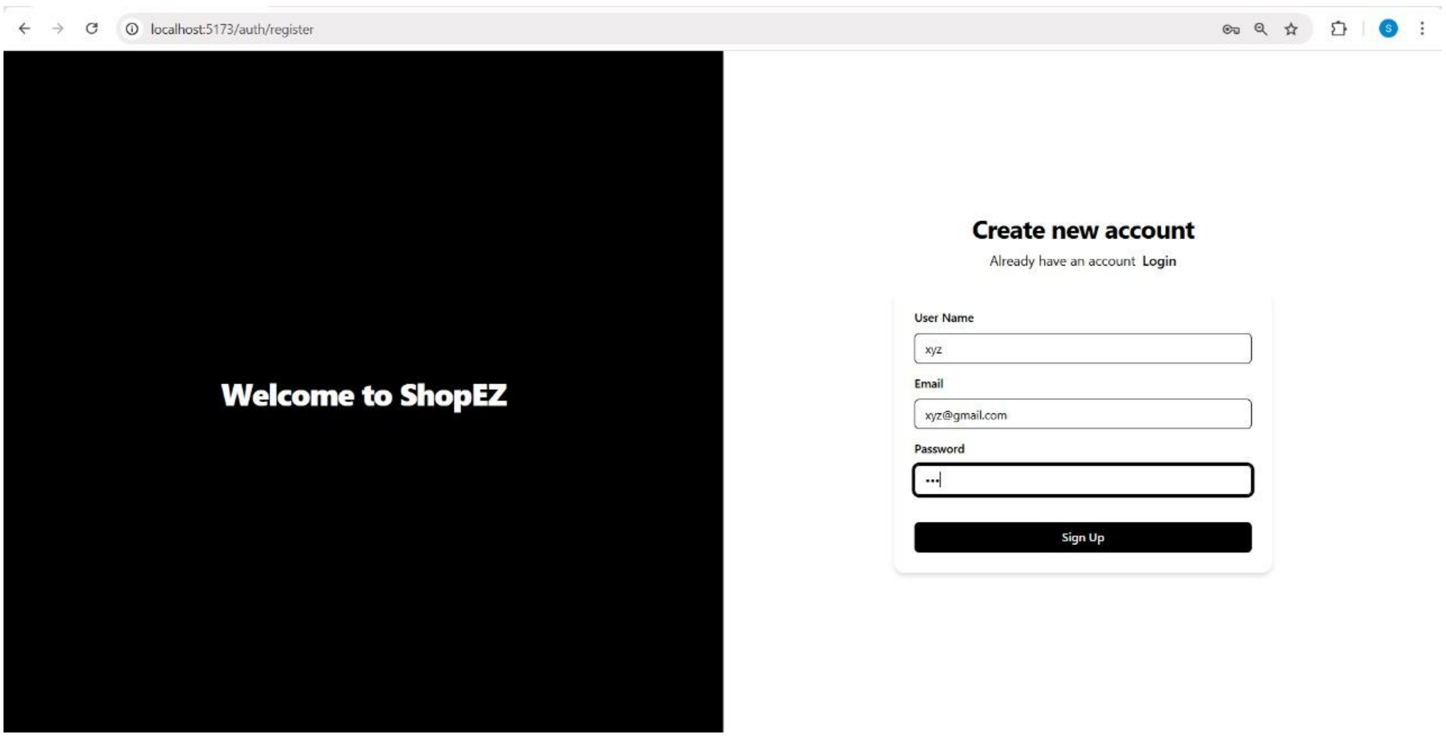
7. RESULTS

7.1 Output Screenshots

The screenshots provided below showcase key pages in the **ShopEZ** application, illustrating the user and admin flows. These screenshots cover critical aspects such as account registration, product browsing, cart management, and the admin dashboard.

User Flow Screenshots

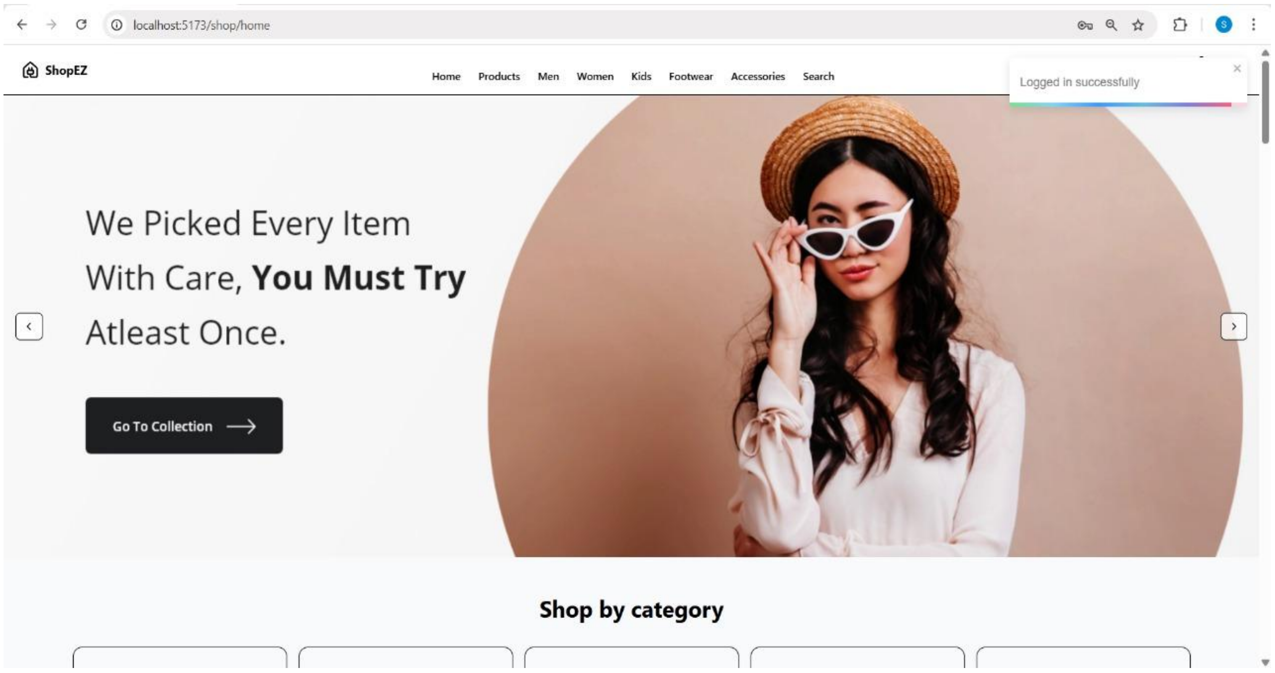
1. **Login Page**
   Existing users can log in using their credentials to access their personalized shopping experience.



xyz

xyz@gmail.com

2)landing page

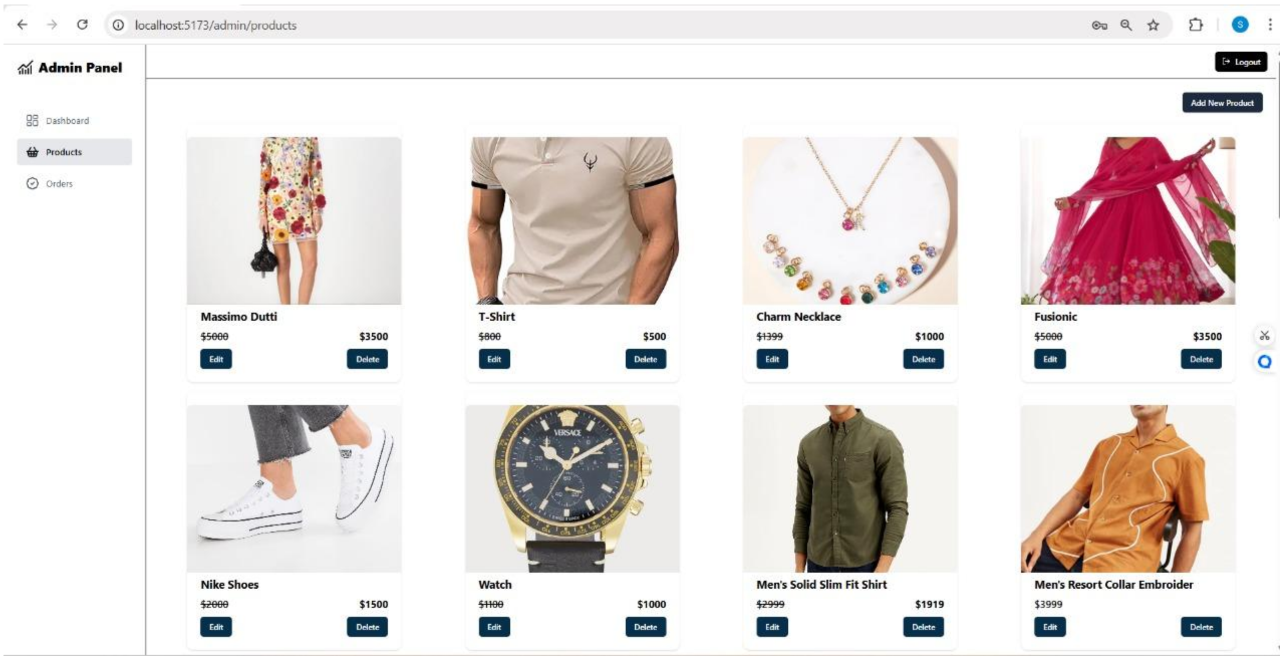


Logged in successfully

We Picked Every Item With Care, **You Must Try** Atleast Once.

Shop by category

3)products page



Massimo Dutti ~~$5000~~ $3500

T-Shirt ~~$800~~ $500

Charm Necklace ~~$1399~~ $1000

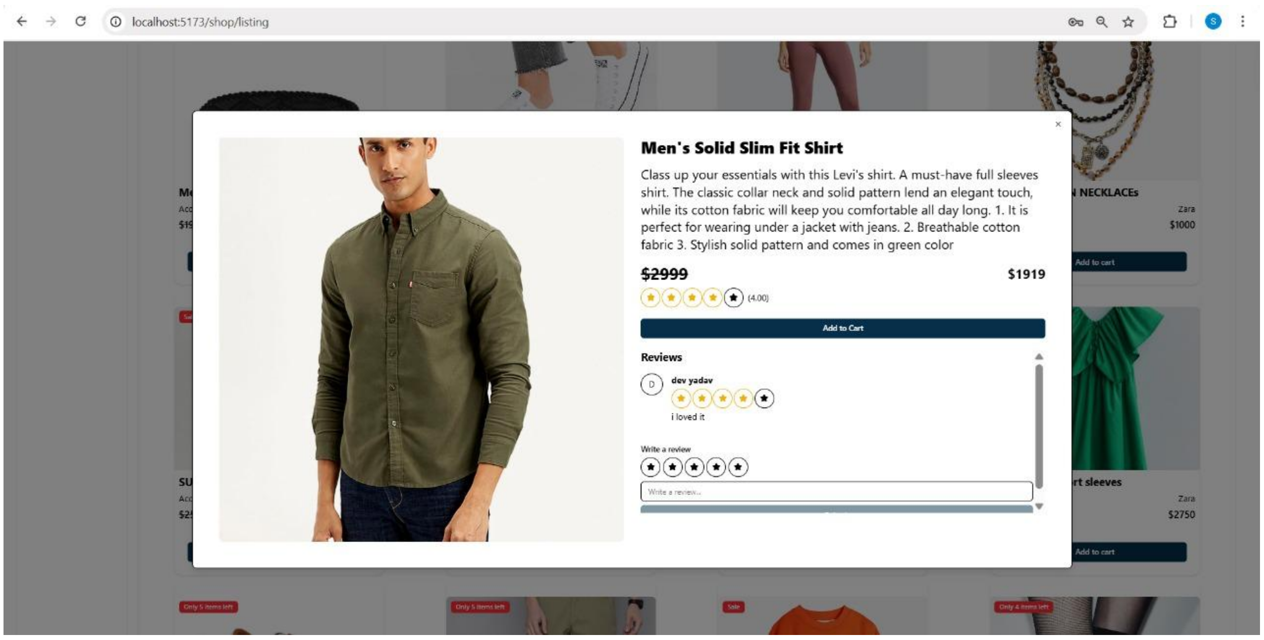
Fusionic ~~$5000~~ $3500

Nike Shoes ~~$2000~~ $1500

Watch ~~$1100~~ $1000

Men's Solid Slim Fit Shirt ~~$2999~~ $1919

Men's Resort Collar Embroider $3999



localhost:5173/shop/listing

Men's Solid Slim Fit Shirt

Class up your essentials with this Levi's shirt. A must-have full sleeves shirt. The classic collar neck and solid pattern lend an elegant touch, while its cotton fabric will keep you comfortable all day long. 1. It is perfect for wearing under a jacket with jeans. 2. Breathable cotton fabric 3. Stylish solid pattern and comes in green color

~~$2999~~ $1919

(4.00)

Add to Cart

Reviews

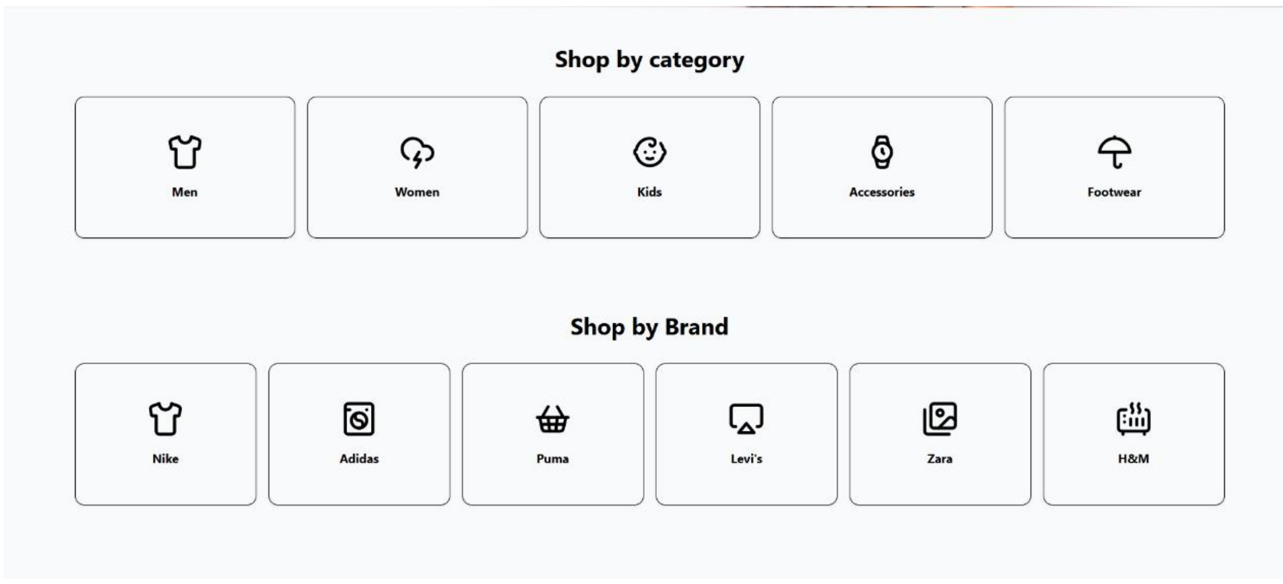
dev yadav

i loved it

Write a review

Write a review...

4) category page



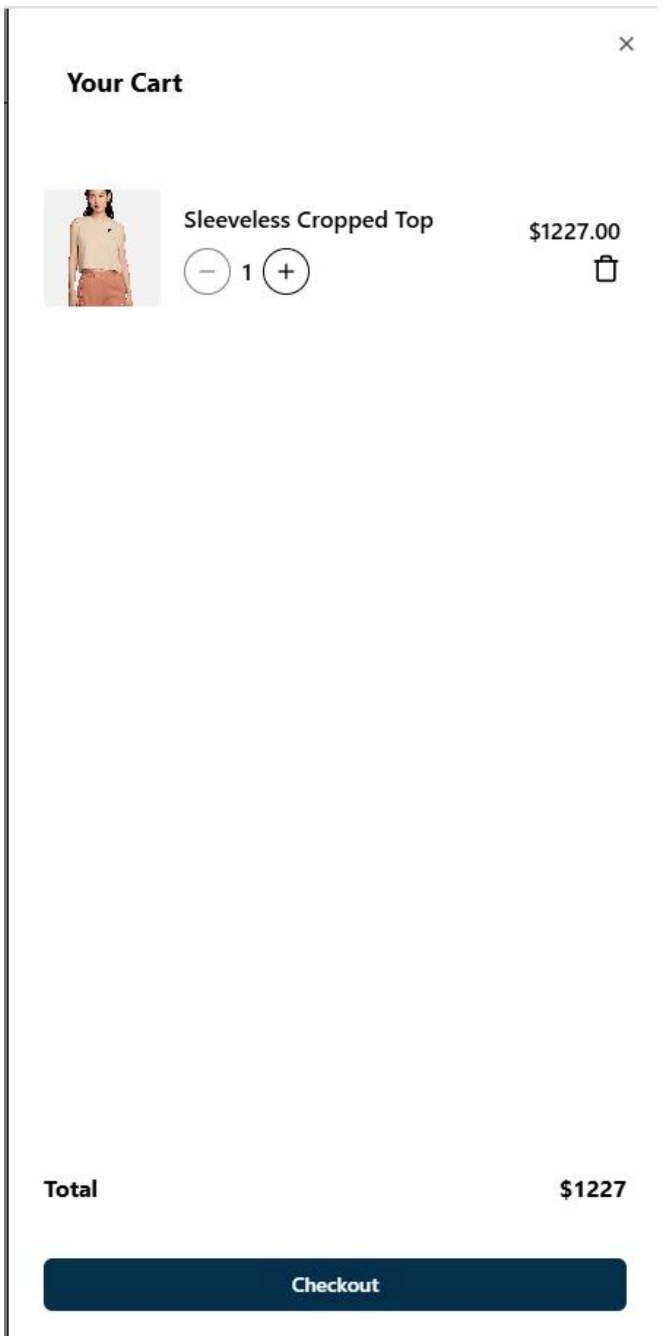
Shop by category

Men | Women | Kids | Accessories | Footwear

Shop by Brand

Nike | Adidas | Puma | Levi's | Zara | H&M

5)adding to cart page


Your Cart

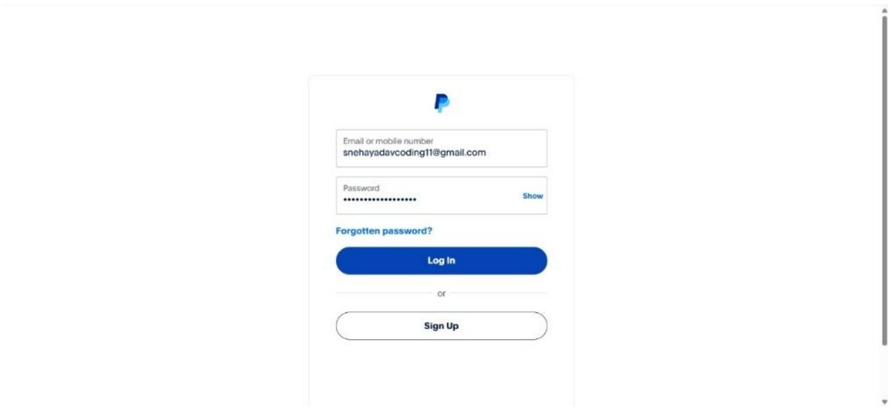
Sleeveless Cropped Top

1

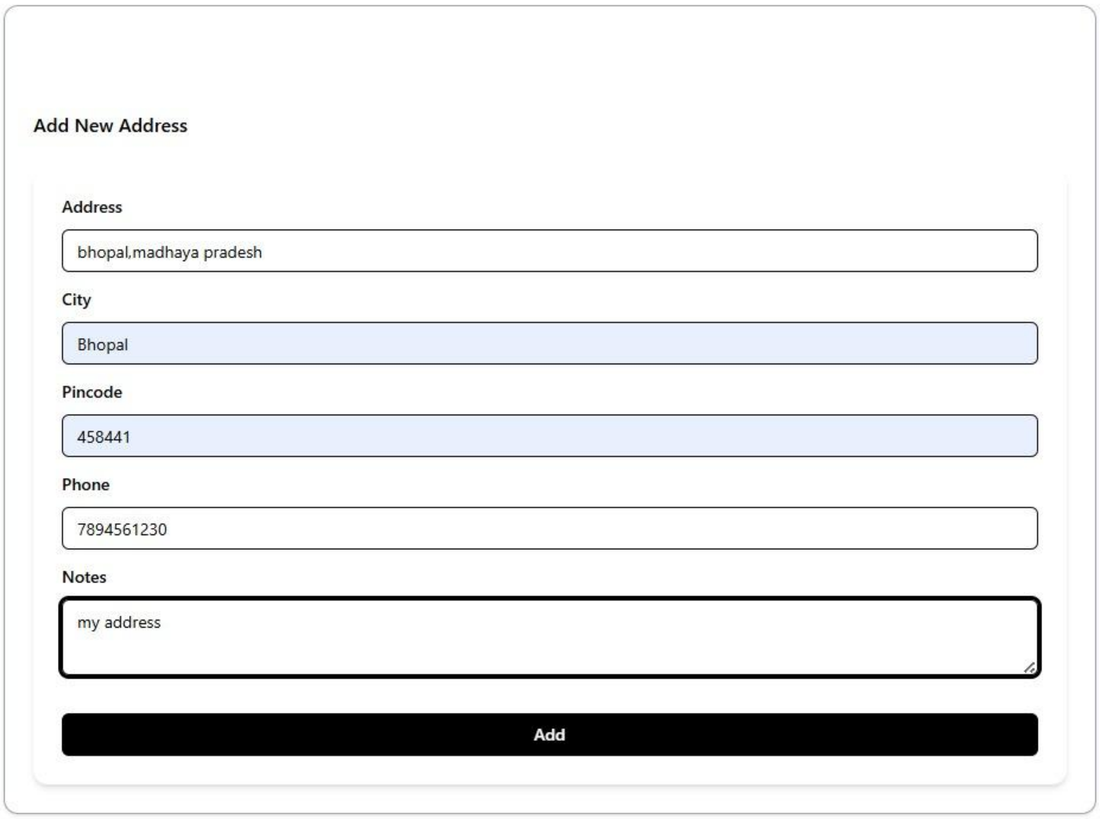
$1227.00

Total $1227

6) payment page


Email or mobile number
snehayadavcoding11@gmail.com
Password
••••••••••••••••••

7) creating or choosing address page


Add New Address

Address: bhopal,madhaya pradesh

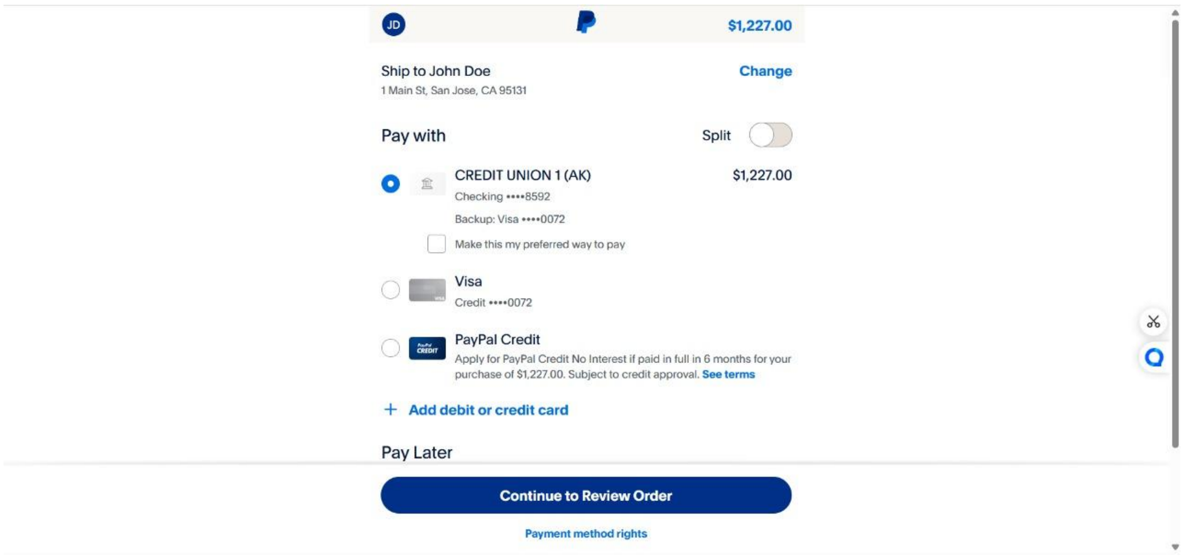
City: Bhopal

Pincode: 458441

Phone: 7894561230

Notes: my address

8) confirmation of payment page


JD $1,227.00

Ship to John Doe
1 Main St, San Jose, CA 95131

Change

Pay with — Split

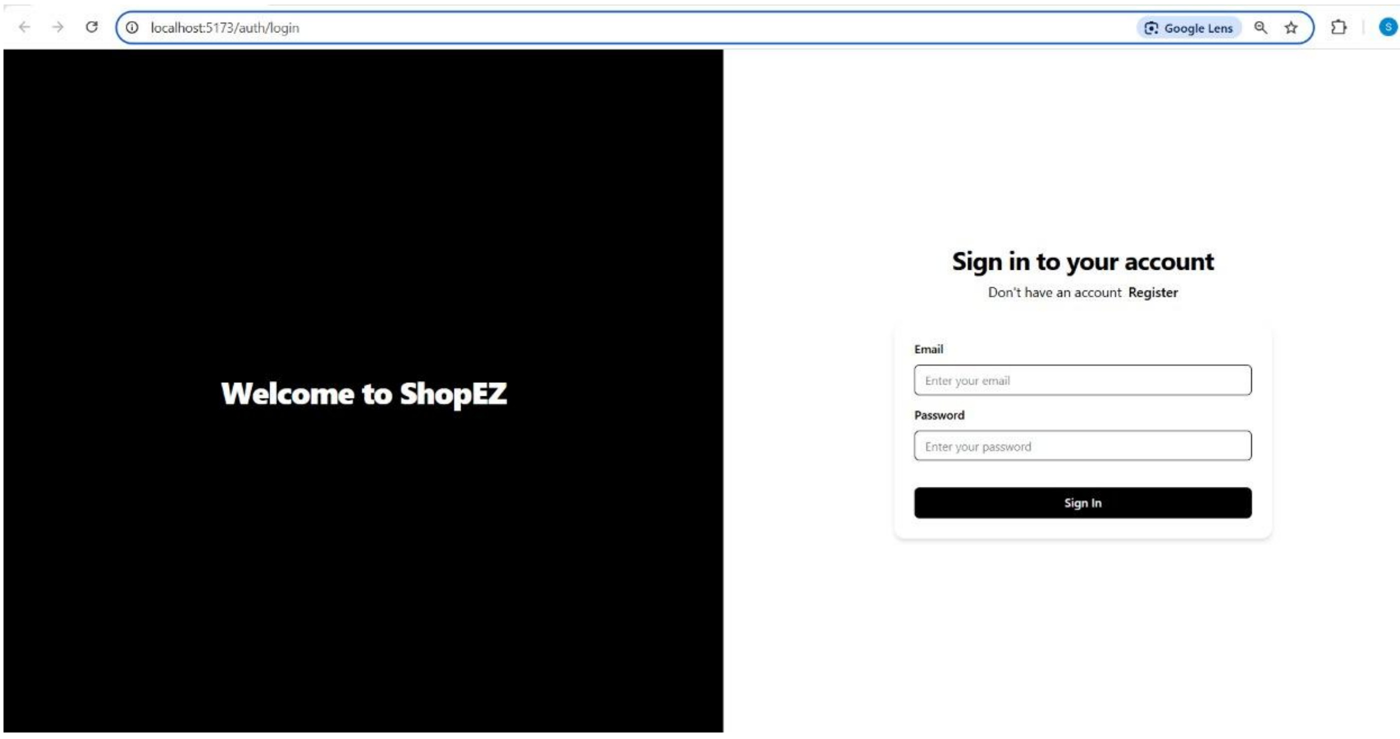
CREDIT UNION 1 (AK) $1,227.00
Checking ••••8592
Backup: Visa ••••0072
Make this my preferred way to pay

Visa
Credit ••••0072

PayPal Credit
Apply for PayPal Credit No Interest if paid in full in 6 months for your purchase of $1,227.00. Subject to credit approval. See terms

ADMIN FLOW SCREENSHOTS:

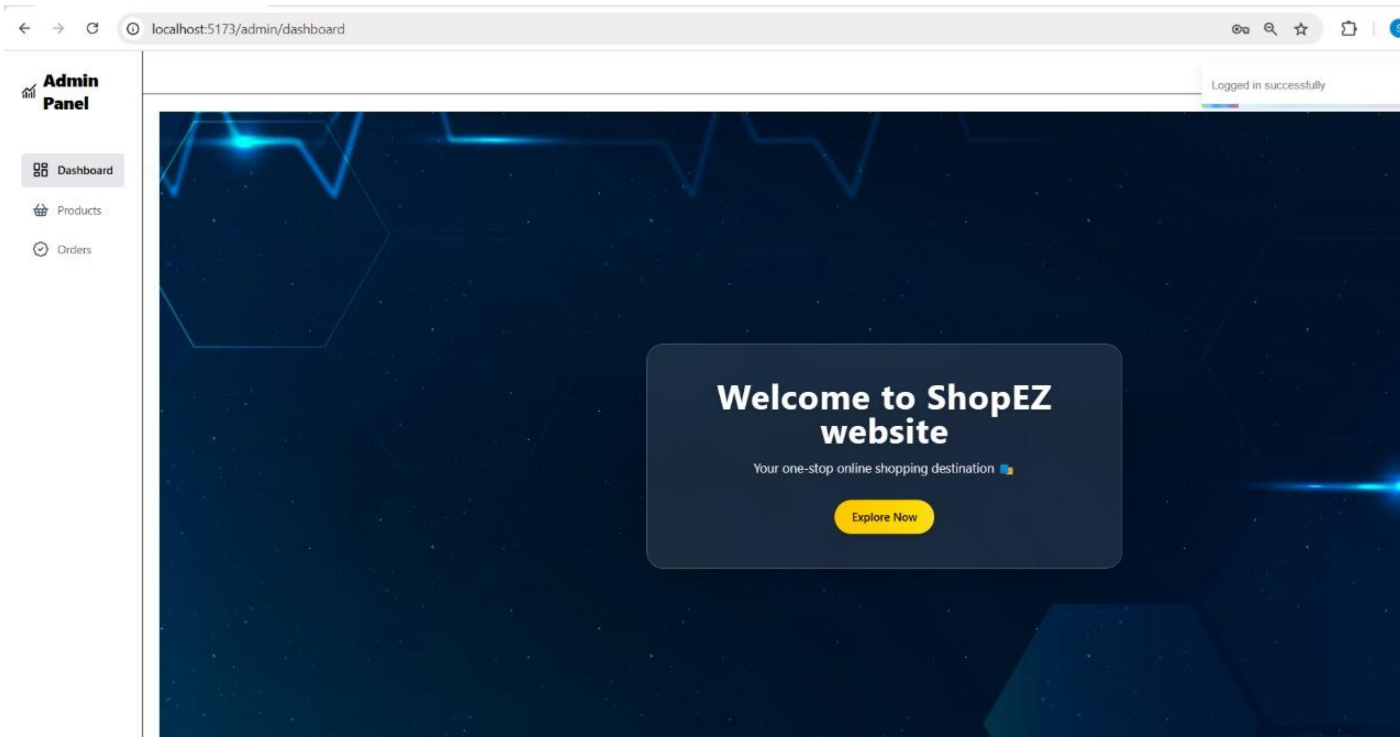
1)login page


localhost:5173/auth/login

Welcome to ShopEZ

Sign in to your account
Don't have an account Register

2)dashboard page

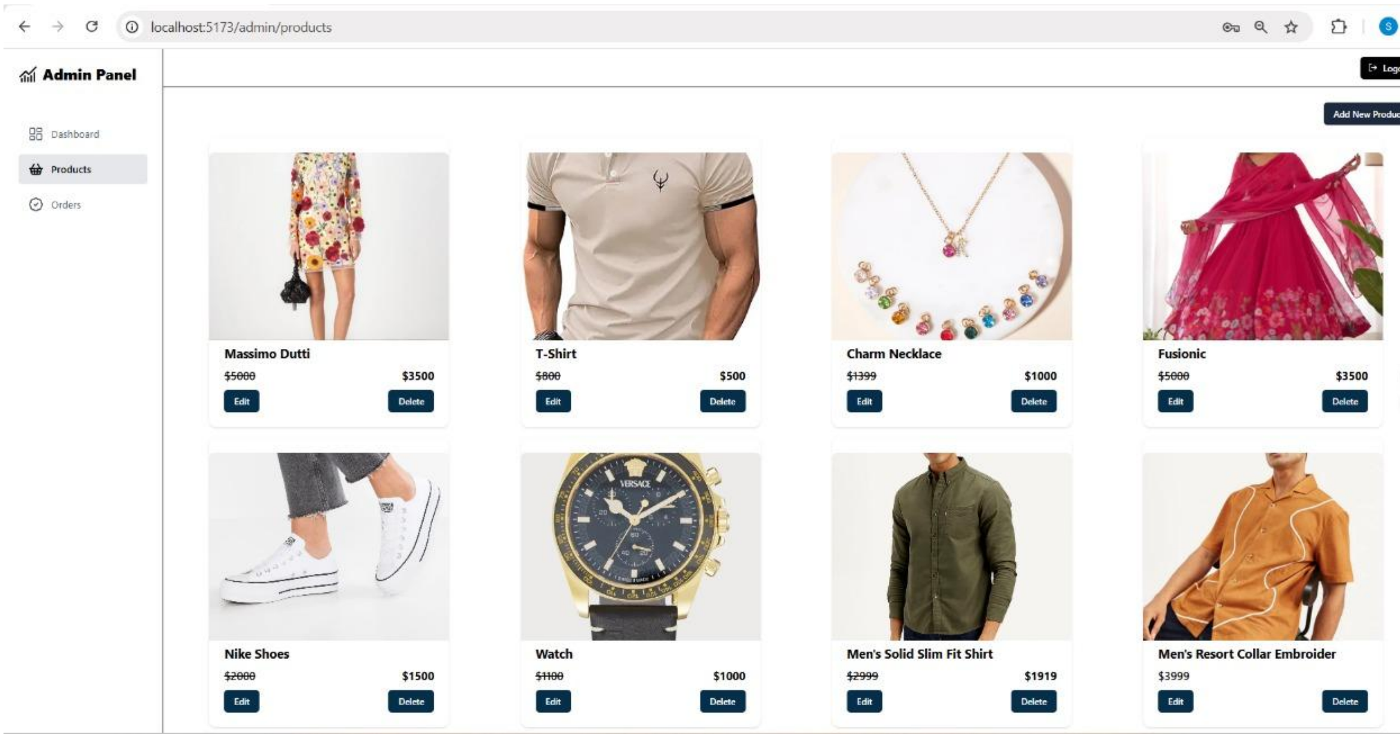


Logged in successfully

Welcome to ShopEZ website

Your one-stop online shopping destination 🛍️

3)products page



Massimo Dutti ~~$5000~~ $3500

T-Shirt ~~$800~~ $500

Charm Necklace ~~$1399~~ $1000

Fusionic ~~$5000~~ $3500

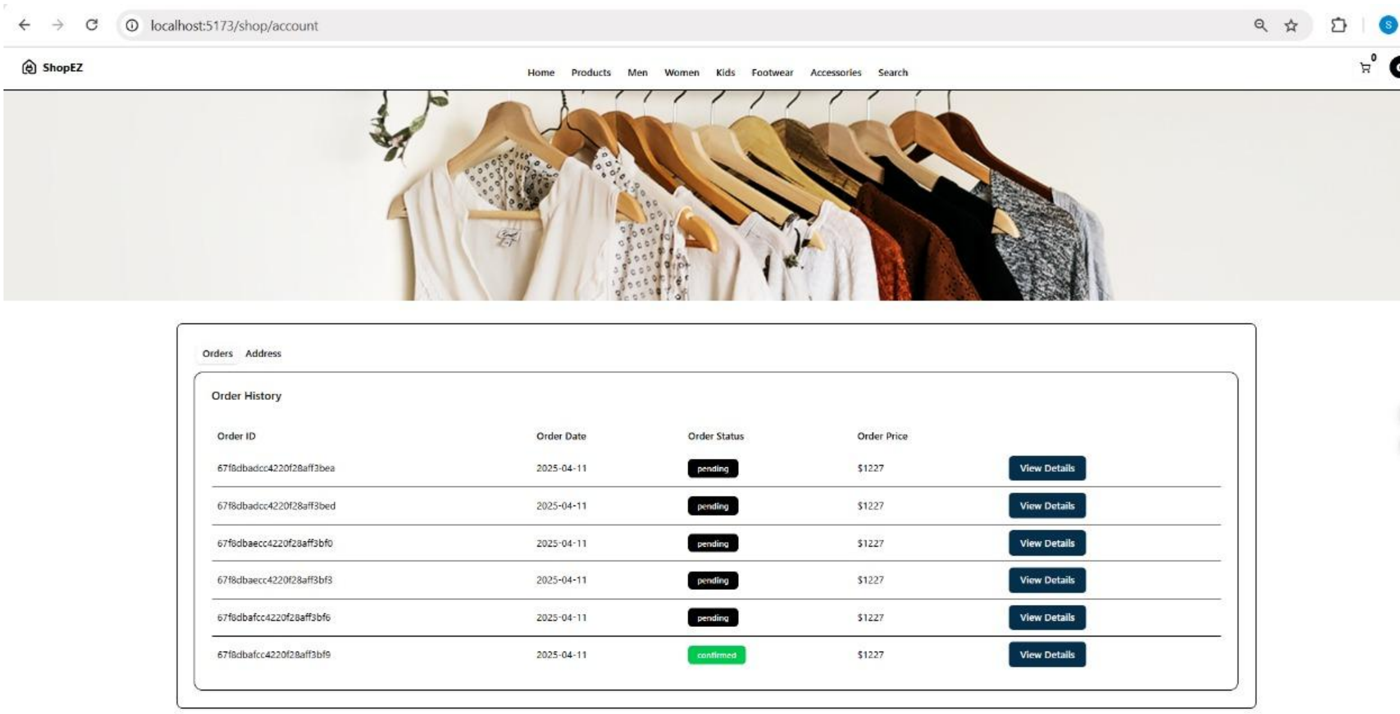
Nike Shoes ~~$2000~~ $1500

Watch ~~$1100~~ $1000

Men's Solid Slim Fit Shirt ~~$2999~~ $1919

Men's Resort Collar Embroider $3999

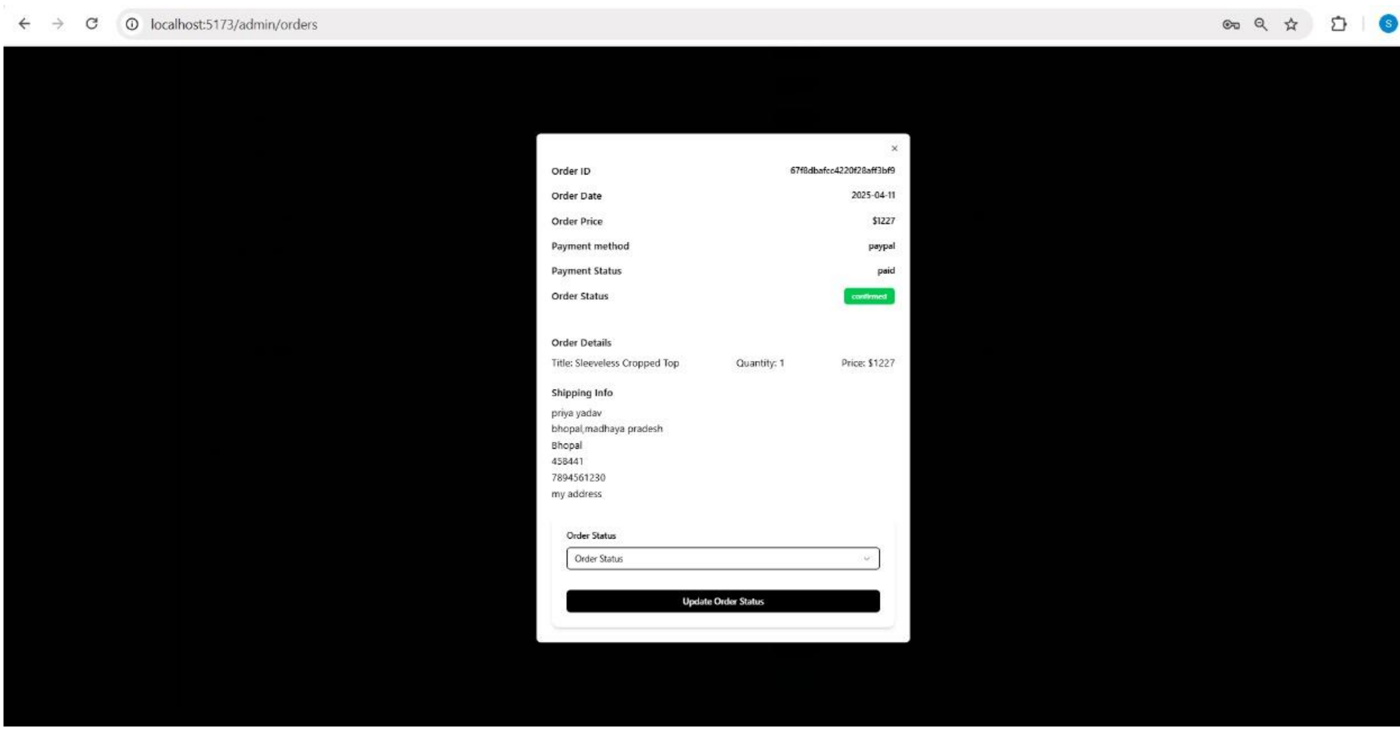
4)orders page



Order History

Order ID	Order Date	Order Status	Order Price	
67f8dbadcc4220f28aff3bea	2025-04-11	pending	$1227	View Details
67f8dbadcc4220f28aff3bed	2025-04-11	pending	$1227	View Details
67f8dbaecc4220f28aff3bf0	2025-04-11	pending	$1227	View Details
67f8dbaecc4220f28aff3bf3	2025-04-11	pending	$1227	View Details
67f8dbafcc4220f28aff3bf6	2025-04-11	pending	$1227	View Details
67f8dbafcc4220f28aff3bf9	2025-04-11	confirmed	$1227	View Details

5)order status and edit page



Order ID	67f8dbafcc4220f28aff3bf9
Order Date	2025-04-11
Order Price	$1227
Payment method	paypal
Payment Status	paid
Order Status	confirmed

Order Details

Title: Sleeveless Cropped Top Quantity: 1 Price: $1227

Shipping Info

priya yadav
bhopal,madhaya pradesh
Bhopal
458441
7894561230
my address

Order Status: Order Status

8. ADVANTAGES & DISADVANTAGES

Advantages:

- User-Friendly Interface: Built using React.js for a responsive and intuitive user experience.
- Secure Checkout Process: Ensures data security through encrypted transactions and validation.
- Efficient Product Browsing: Fast product retrieval and display using optimized MongoDB queries.
- Real-Time Updates: Cart and order information are updated instantly for better user interaction.

- Admin Management: Admins can manage users, products, and monitor activities effectively.

Disadvantages:

- No Seller Module: Current version lacks features for sellers to manage their own product listings.
- Dependent on Internet Connection: As with most web apps, functionality depends on stable connectivity.
- Limited Scalability in Free Hosting: Hosting on free-tier platforms may not handle heavy traffic effectively.

9.CONCLUSION:

ShopEZ is a dynamic and user-friendly e-commerce platform built on the powerful **MERN stack** (MongoDB, Express.js, React.js, Node.js). Designed to deliver a seamless shopping experience for users while providing a robust backend system for sellers and administrators, ShopEZ ensures scalability, security, and performance across all its features.
The platform efficiently serves three core user roles: **customers**, **sellers**, and **administrators.** Customers enjoy an intuitive interface for browsing products, adding items to their cart, and

completing secure checkouts. Sellers are equipped with a dedicated dashboard that simplifies product management, inventory tracking, and order processing, allowing them to scale their operations with ease. Administrators are given powerful tools to oversee platform activity, manage user accounts, and maintain the integrity of product listings—ensuring a safe and efficient marketplace for all.

ShopEZ showcases the strength of modern web technologies in building scalable and engaging digital commerce solutions. With its modular architecture, thoughtful design, and focus on user experience, it lays a solid foundation for ongoing innovation and future expansion.

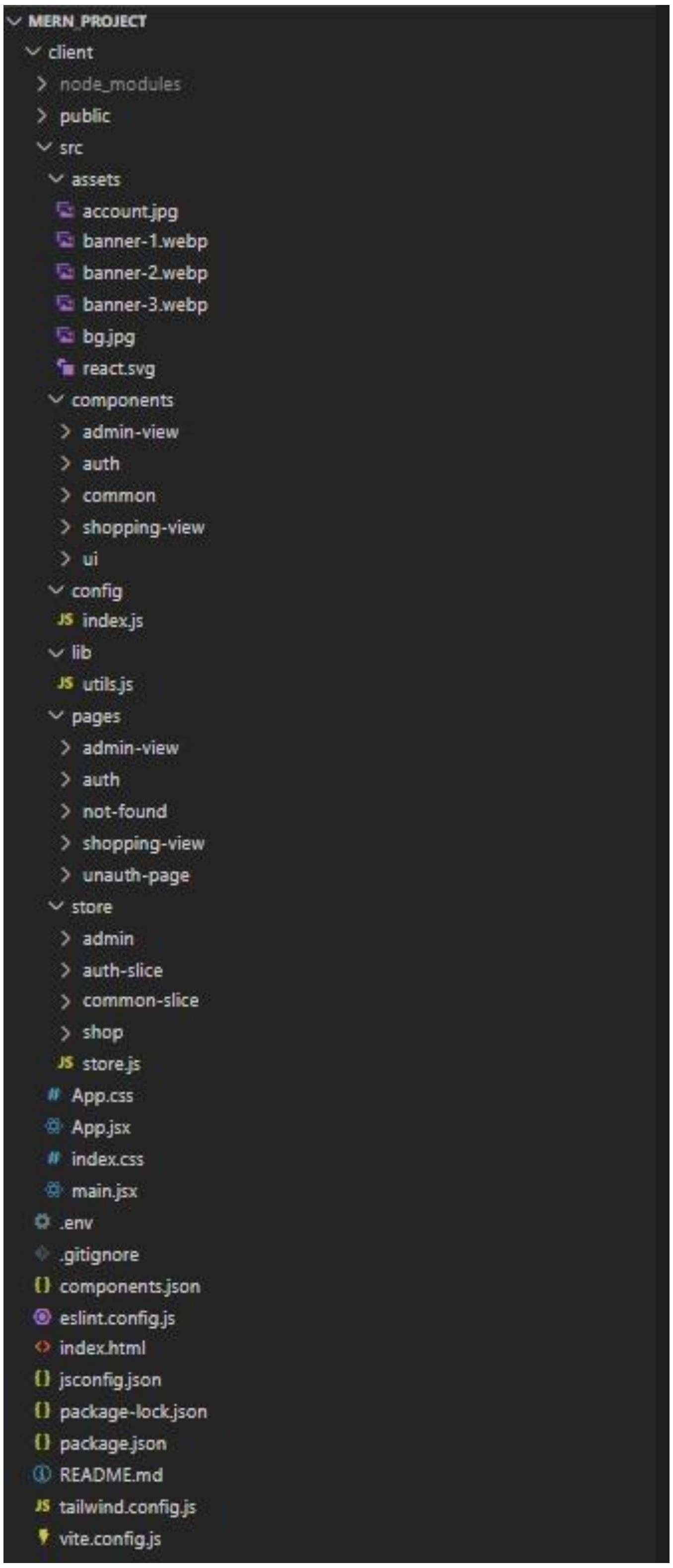
In summary, **ShopEZ not only addresses the current demands of the e-commerce ecosystem but also sets the stage for continued growth—enhancing both the shopping experience and the management capabilities for all users involved.**

10. FUTURE SCOPE

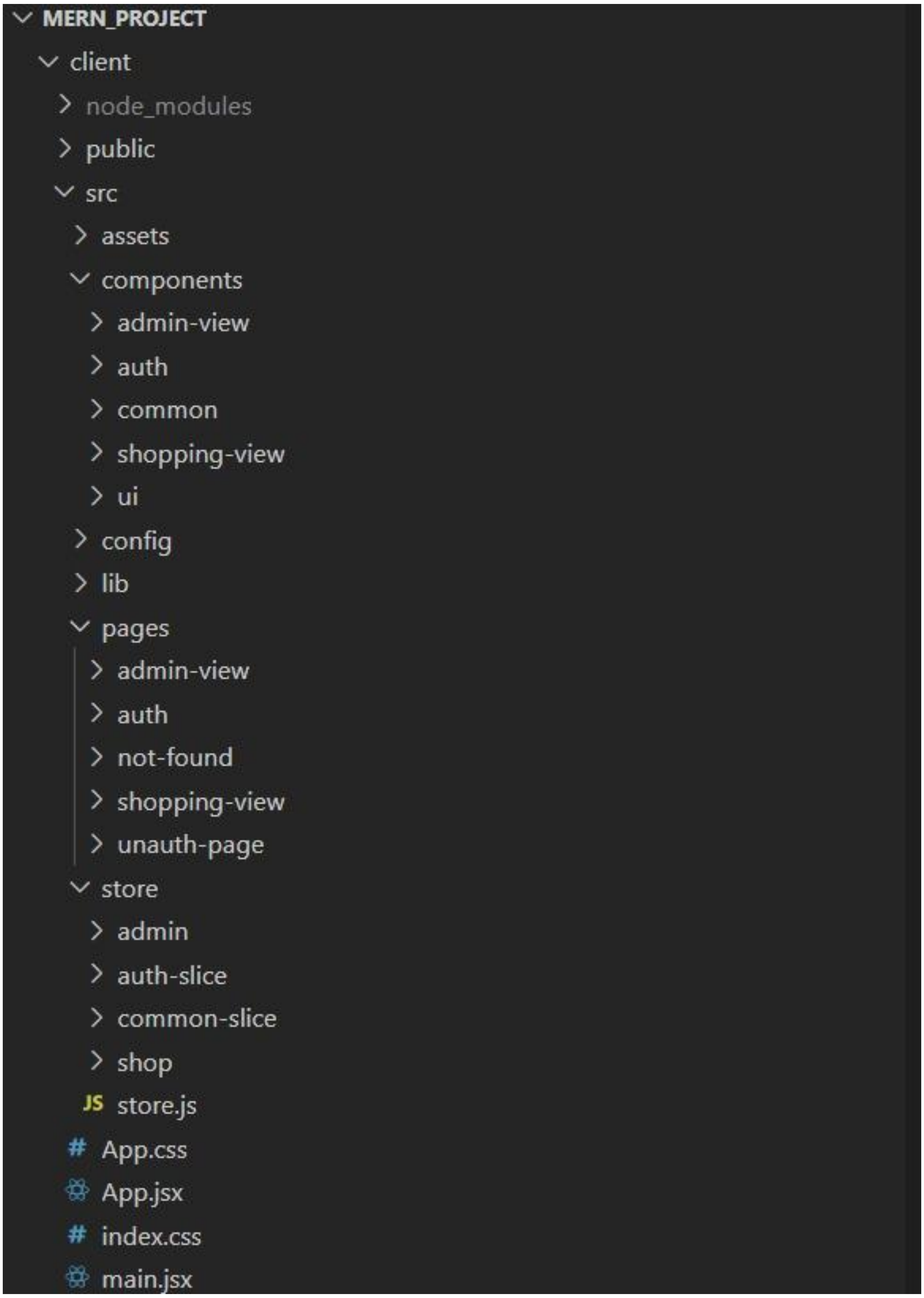
- **Seller Dashboard Integration**: Adding seller-side functionality for inventory and order management.
- **Payment Gateway Integration**: Enabling real-time transactions using services like Razorpay or Stripe.
- **Wishlist and Review System**: Allowing users to save products and share feedback.
- **Mobile App Version**: Extending platform accessibility through a dedicated mobile app.
- **AI-Based Recommendations**: Implementing product suggestions based on user behavior.

11.APPENDIX

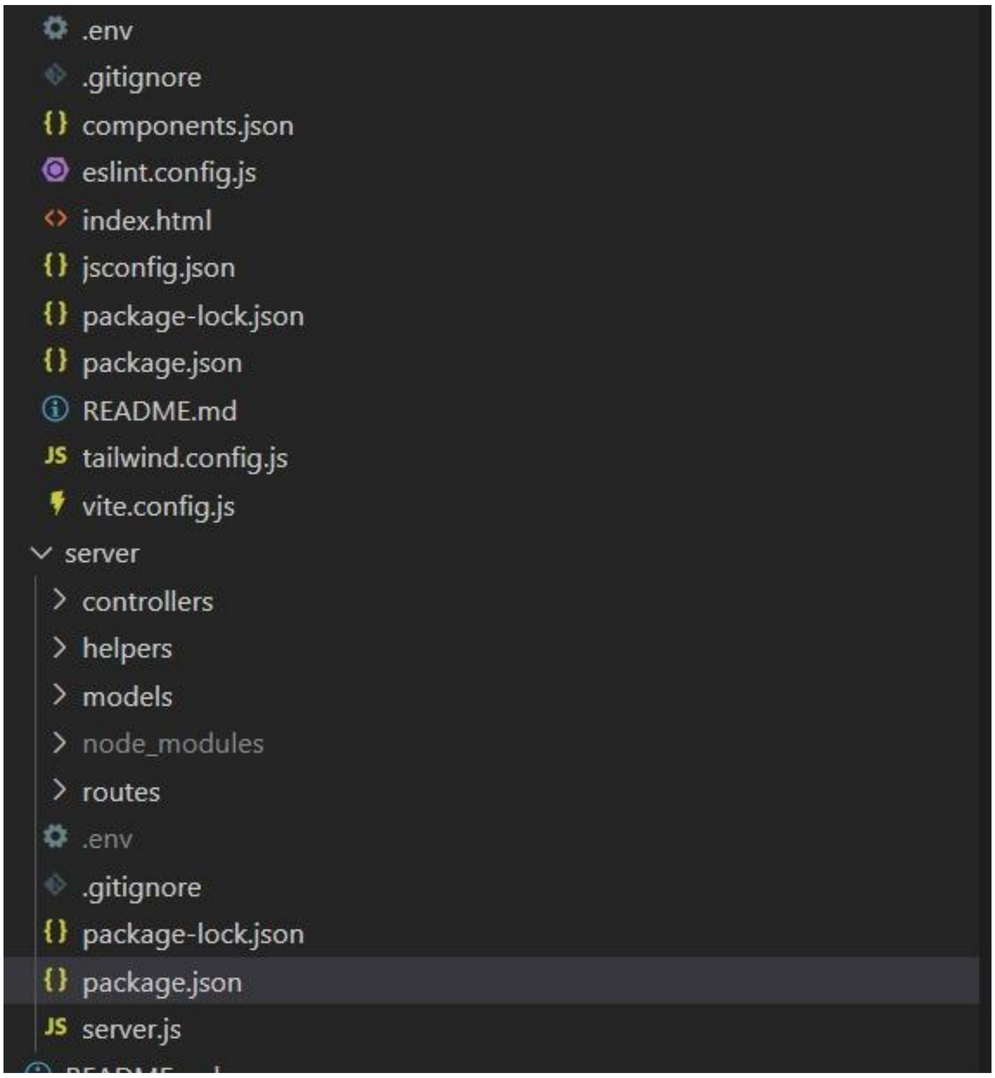
11.1 Source code



```
MERN_PROJECT
  client
    node_modules
    public
    src
      assets
        account.jpg
        banner-1.webp
        banner-2.webp
        banner-3.webp
        bg.jpg
        react.svg
      components
        admin-view
        auth
        common
        shopping-view
        ui
      config
        index.js
      lib
        utils.js
      pages
        admin-view
        auth
        not-found
        shopping-view
        unauth-page
      store
        admin
        auth-slice
        common-slice
        shop
        store.js
      App.css
      App.jsx
      index.css
      main.jsx
    .env
    .gitignore
    components.json
    eslint.config.js
    index.html
    jsconfig.json
    package-lock.json
    package.json
    README.md
    tailwind.config.js
    vite.config.js
```



```
∨ MERN_PROJECT
  ∨ client
    > node_modules
    > public
    ∨ src
      > assets
      ∨ components
        > admin-view
        > auth
        > common
        > shopping-view
        > ui
      > config
      > lib
      ∨ pages
        > admin-view
        > auth
        > not-found
        > shopping-view
        > unauth-page
      ∨ store
        > admin
        > auth-slice
        > common-slice
        > shop
        JS store.js
      # App.css
      App.jsx
      # index.css
      main.jsx
```



```
.env
.gitignore
components.json
eslint.config.js
index.html
jsconfig.json
package-lock.json
package.json
README.md
tailwind.config.js
vite.config.js
server
  controllers
  helpers
  models
  node_modules
  routes
  .env
  .gitignore
  package-lock.json
  package.json
  server.js
```

11.2 GitHub & Project Demo Link:

- **GitHub Repository:** https://github.com/SnehaYadavWorld11/Mern_Stack-ShopEZ.git
- **Live Demo:** https://drive.google.com/file/d/1sf0OhwgIi3d6qpX8PHjiiuWHaaonJbmT/view?usp=sharing
- https://drive.google.com/file/d/1znDPKbi1CsA1oLESelnp5fbzPIrtxCfw/view?usp=sharing