SHOPSY E-COMMERCE APP USING MERN PROJECT REPORT

Submitted by

TEAM NM ID: NM2024TMID12629

NAME:	**NM ID:**
ANNIE FIORINA S.A.	9CF039AD818995FBF194CBF91C90976F
ANITHA M	425F6163F777C7E8B674ABF400C184EB
ANANTH PANDIAN P	B546B719A3FF27C94D64B149D861EE2F
AJITHKUMAR S	3B6B9195631226380E542A76FA8D400E

In partial completion towards the attainment of the degree of

BACHELOR OF TECHNOLOGY IN INFORMATION TECHNOLOGY

T.J. INSTITUTE OF TECHNOLOGY
KARAPAKKAM, CHENNAI – 600097

ANNA UNIVERSITY
CHENNAI – 600025

TABLE OF CONTENTS

Shopsy: E-Commerce Application Built on MERN Stack

1. Project Overview

1.1 Project Description

Shopsy is a fully functional e-commerce application built using the **MERN stack** (MongoDB, Express.js, React.js, and Node.js). The platform aims to provide users with a seamless shopping experience, including browsing a diverse product catalog, managing a shopping cart, securely checking out, and tracking their orders. Additionally, the application includes an admin panel for product and order management, enabling business owners to efficiently manage their e-commerce store.

Shopsy integrates modern features such as secure user authentication, payment gateway integration, and responsive design for a mobile-friendly experience.

1.2 Project Objectives

- **User-Centric Features**: To enable users to view products, add them to their cart, place orders, and view their order history.
- **Admin Features**: To provide administrators with tools to manage products, view sales analytics, and manage user accounts.
- **Payment Gateway**: To integrate Stripe or PayPal for secure, seamless payment processing.
- **Responsive and Scalable**: To ensure the application is optimized for both desktop and mobile devices while being scalable as traffic and data grow.

1.3 Technologies Used

- **Frontend**: React.js, React Router, Redux (optional for state management), Axios (for HTTP requests)
- **Backend**: Node.js, Express.js, JWT (for authentication), Stripe/PayPal (for payments)
- **Database**: MongoDB (using Mongoose for schema definition)
- **Testing**: Jest (Frontend), Mocha/Chai (Backend)

1.4 Features of Shopsy

- **User Features**:
    - Product catalog browsing
    - Cart management
    - User registration and login
    - Order history and profile management
    - Secure checkout via Stripe/PayPal
- **Admin Features**:
    - Product management (CRUD)
    - Order management
    - User account management
    - Admin dashboard with sales analytics

2. Architecture

2.1 Overall Architecture

The **Shopsy** application follows a **client-server architecture** where the frontend (React.js) and backend (Node.js with Express.js) communicate through RESTful APIs. The database (MongoDB) stores all application data, and JWT is used for secure authentication. This separation of concerns ensures flexibility and scalability.

Frontend (Client-Side)

- **React.js**: React is used to build the user interface with a component-based architecture, which makes the app modular and easy to maintain.
- **React Router**: For navigating between pages like home, product details, cart, and checkout.
- **Redux** (optional): For managing application-wide state, including user authentication, cart items, and order details.
- **Axios**: Used to send HTTP requests from the frontend to the backend API.

Backend (Server-Side)

- **Node.js & Express.js**: The backend is powered by Node.js and Express.js, handling API requests such as fetching products, user authentication, and order management.
- **JWT Authentication**: For securing routes and ensuring that only authenticated users can perform sensitive actions (e.g., placing an order, viewing order history).
- **Stripe/PayPal Integration**: For processing secure payments at checkout.

Database

- **MongoDB**: A NoSQL database used to store various collections like products, users, orders, and shopping cart data. MongoDB's flexibility allows for fast development and scalability.
- **Mongoose**: An ODM (Object Data Modeling) library for MongoDB that helps manage and define the schema for data.

2.2 Database Structure

The database is structured using MongoDB collections for products, users, orders, and carts. Below is an overview of the key collections:

1. **Users Collection**:
   a. name, email, password, role (user/admin), address, phoneNumber
2. **Products Collection**:
   a. name, description, price, category, imageURL, inventoryCount
3. **Orders Collection**:
   a. userId, products (array of product IDs), shippingAddress, paymentStatus, orderStatus, totalAmount
4. **Cart Collection**:
   a. userId, products (array of product IDs and quantities), totalPrice

3. Setup Instructions

3.1 Prerequisites

Before setting up the Shopsy application, ensure that you have the following installed:

- **Node.js** (with npm)
- **MongoDB** (locally or MongoDB Atlas for cloud hosting)
- **Stripe or PayPal Account** (for payment integration)
- **Git** (for version control)

3.2 Installation Steps

1. Clone the Repository

```
git clone  https://github.com/annieanto/E-commerce-application.git
```

2. **Install Backend Dependencies** Navigate to the backend folder and install the necessary packages:

```
cd backend
npm install
```

3. **Install Frontend Dependencies** Navigate to the frontend folder and install the necessary packages:

```
cd frontend
npm install
```

4. **Set Up Environment Variables** Create .env files in both the backend and frontend directories:

   a. **Backend .env**:

```
MONGODB_URI=mongodb://localhost:27017/shopsy
JWT_SECRET=your_jwt_secret
STRIPE_SECRET_KEY=your_stripe_secret_key
PORT=5000
```

   b. **Frontend .env**:

```
REACT_APP_API_URL=http://localhost:5000/api
```

5. **Start the Application**
    a. **Backend**: cd backend
       npm run dev

    b. **Frontend**: cd frontend
       npm start

4. Folder Structure

```
shopsy/
│
├── backend/
│   ├── config/                 # Database, JWT, Payment
gateway configuration
│   ├── controllers/            # API route handlers (user,
product, order, cart)
│   ├── models/                 # Mongoose models for data
(user, product, order)
│   ├── routes/                 # Express.js routes for API
endpoints
│   ├── server.js               # Main server file
│   ├── .env                    # Backend environment variables
│
├── frontend/
│   ├── public/                 # Static assets (images,
favicon, etc.)
│   ├── src/
│   │   ├── components/         # Reusable UI components
```

```
(Navbar, ProductCard, Cart)
│   │   ├── pages/            # React pages (Home, Product
Details, Checkout)
│   │   ├── redux/            # Redux actions and reducers
(optional)
│   │   ├── App.js            # Main React component
│   │   ├── index.js          # React entry point
│   │   └── .env              # Frontend environment
variables
│
└── package.json              # Project dependencies and
scripts
```

5. Running the Application

Once the application is set up, you can run both the frontend and backend simultaneously.

1. **Backend**:
   a. Run the backend server (e.g., Express) on port 5000: cd backend
      npm run dev

2. **Frontend**:
   a. Start the React development server on port 3000: cd frontend
      npm start

The application will be accessible at http://localhost:3000 in your browser.

6. API Documentation

6.1 Authentication API

- **POST /api/auth/register**: Register a new user.
  - Request Body: { name, email, password, address }
  - Response: { token }
- **POST /api/auth/login**: Log in a user.
  - Request Body: { email, password }
  - Response: { token }

6.2 Product API

- **GET /api/products**: Fetch all products.
- **GET /api/products/:id**: Fetch a single product by ID.
- **POST /api/products**: Add a new product (Admin only).
  - Request Body: { name, description, price, imageURL, category, inventoryCount }

6.3 Cart API

- **GET /api/cart**: Retrieve the current user's cart.
- **POST /api/cart**: Add an item to the cart.
  - Request Body: { productId, quantity }

6.4 Order API

- **POST /api/orders**: Create a new order.
  - Request Body: { products, totalPrice, shippingAddress }

7. Authentication

7.1 JWT Authentication

- **JWT (JSON Web Token)** is used to authenticate users. After logging in or registering, a JWT token is generated and sent back to the client. The client includes this token in the headers of subsequent requests to access protected resources such as placing an order or viewing order history.

7.2 Protected Routes

- Routes like **POST /api/orders** and **POST /api/products** require a valid JWT token in the request header to authorize access.

8. User Interface

The **Shopsy** user

interface is designed with simplicity and usability in mind. It is built using React.js and follows modern UI/UX design principles:

- **Homepage**: Displays product categories, featured products, and promotions.
- **Product Details**: Shows detailed information about a product, including price, description, and image.
- **Shopping Cart**: Allows users to view and manage cart items before checkout.
- **Checkout**: Secure checkout page for payment and order confirmation.

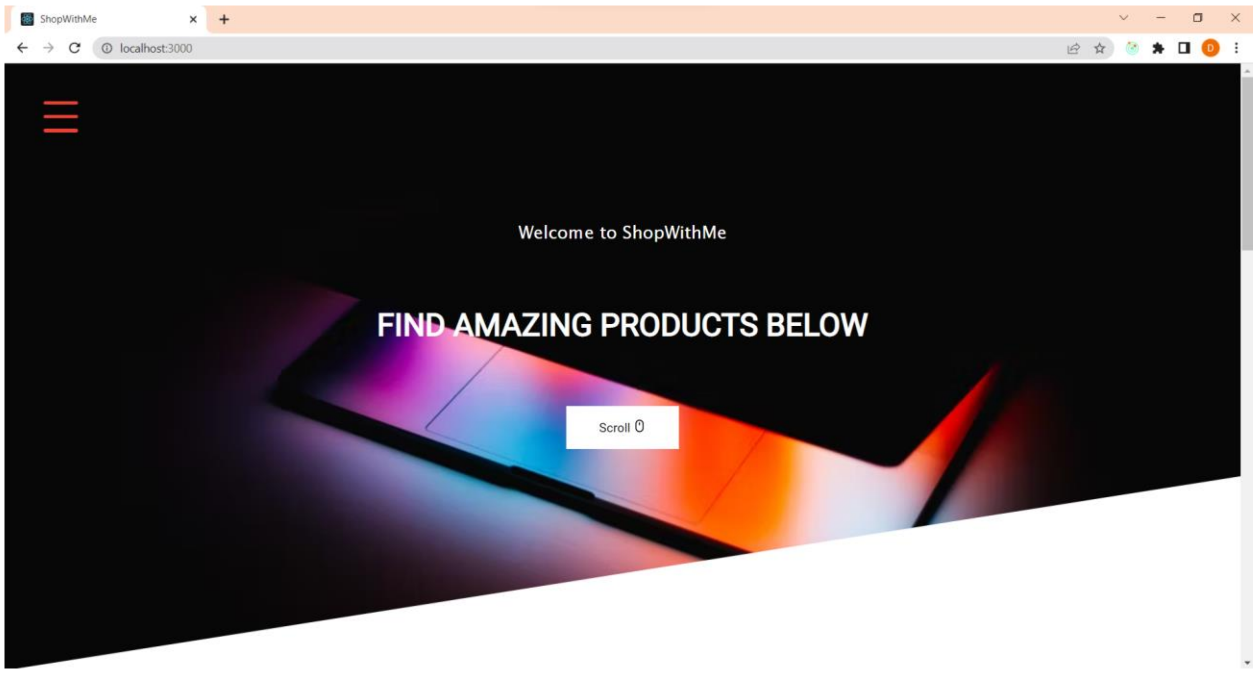
- **Admin Dashboard**: Provides tools for admins to manage products, orders, and users.

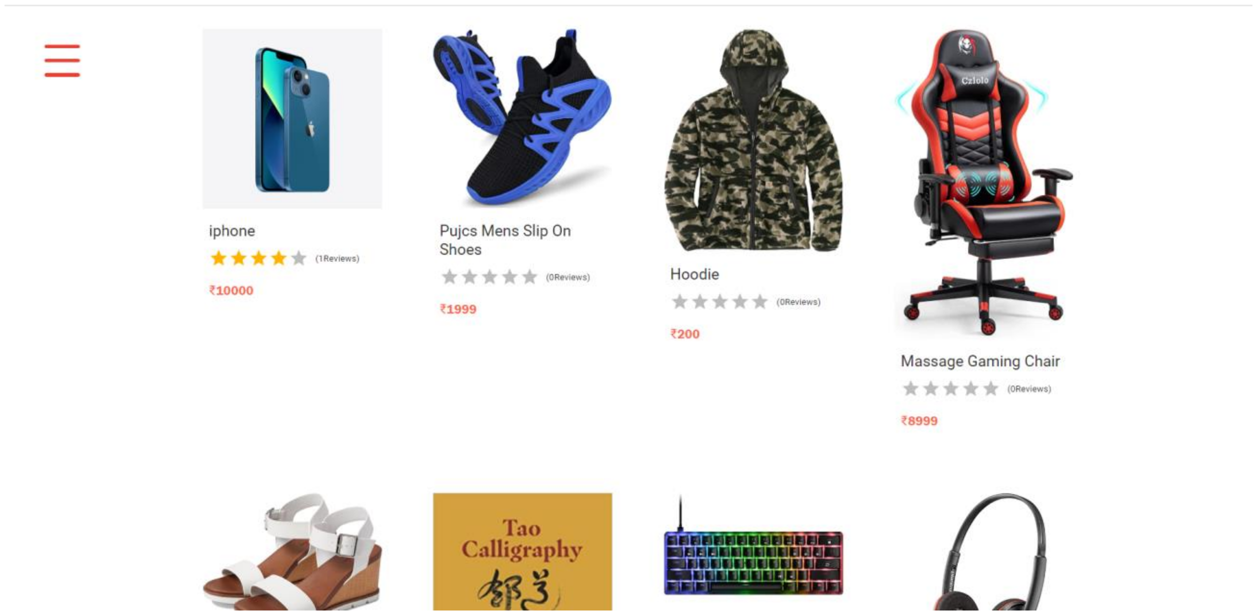
9. Testing

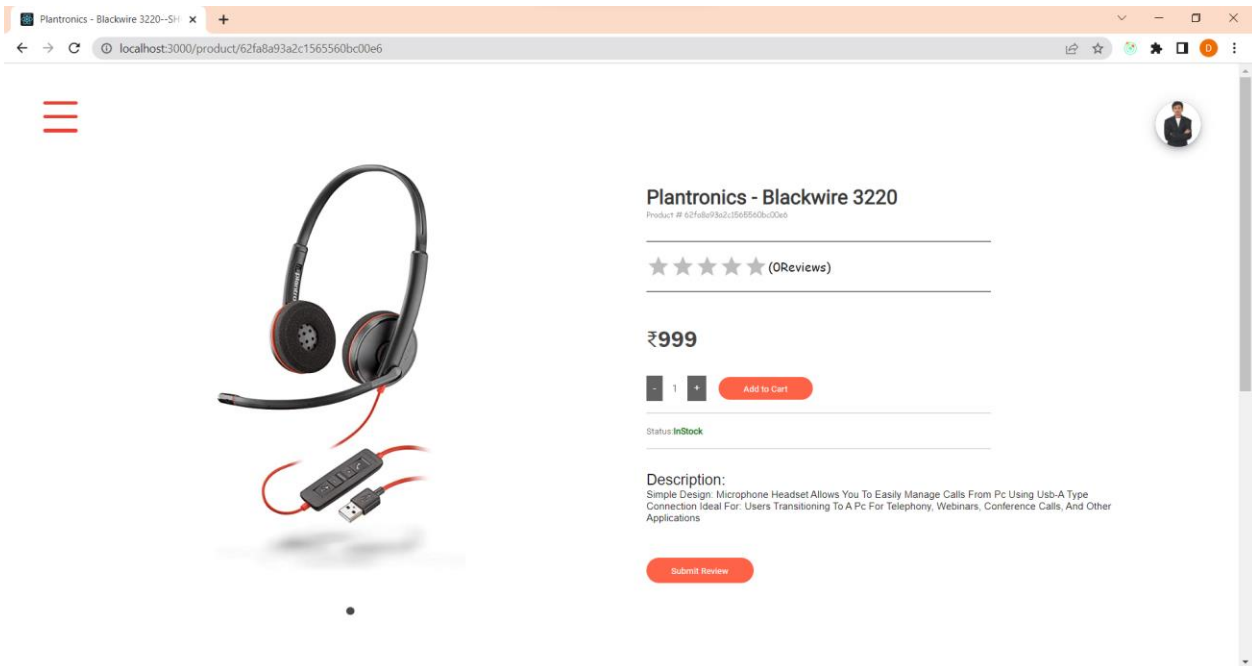
- **Frontend Testing**: **Jest** is used to write unit and integration tests for React components to ensure that the user interface works as expected.
- **Backend Testing**: **Mocha** and **Chai** are used to write tests for API endpoints and business logic to ensure that the backend functions correctly.

10. Demo

A demo version of **Shopsy** can be hosted on platforms such as **Heroku** or **Netlify**, showcasing the full user experience of browsing products, managing the cart, and completing a purchase.


ShopWithMe
localhost:3000
Welcome to ShopWithMe
FIND AMAZING PRODUCTS BELOW
Scroll


iphone
(1Reviews)
₹10000
Pujcs Mens Slip On Shoes
(0Reviews)
₹1999
Hoodie
(0Reviews)
₹200
Massage Gaming Chair
(0Reviews)
₹8999
Tao Calligraphy


Plantronics - Blackwire 3220--SH
localhost:3000/product/62fa8a93a2c1565560bc00e6

Plantronics - Blackwire 3220

Product # 62fa8a93a2c1565560bc00e6

(0Reviews)

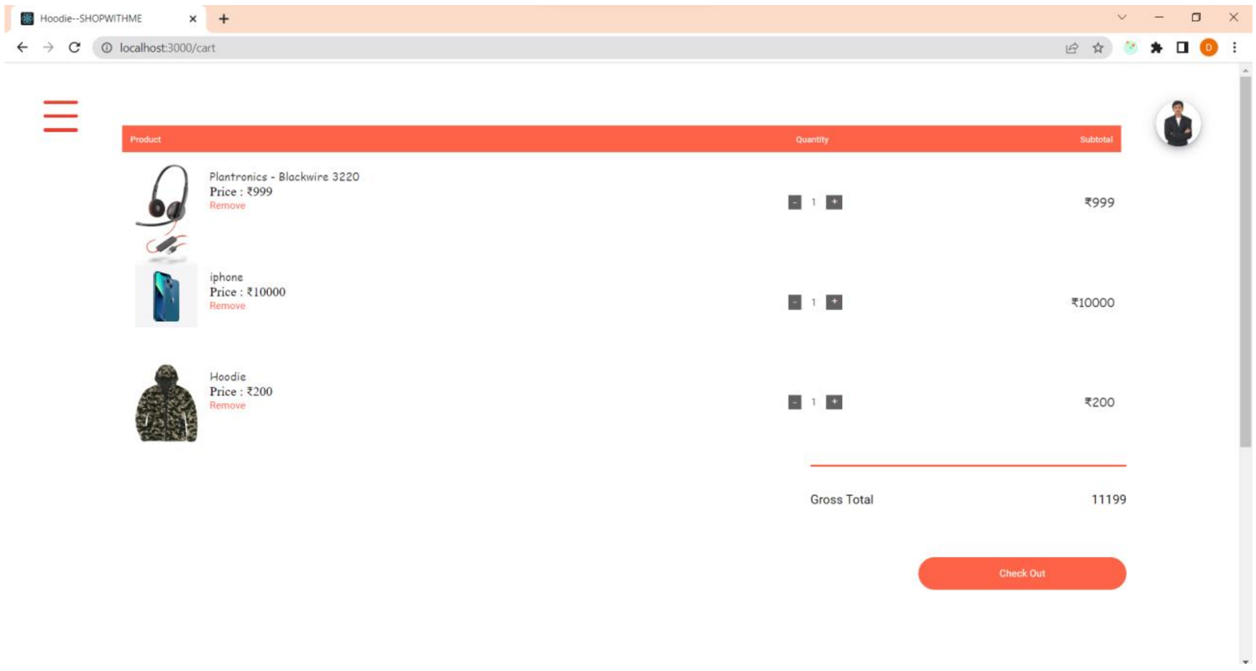
₹999

- 1 + Add to Cart

Status:InStock

Description:
Simple Design: Microphone Headset Allows You To Easily Manage Calls From Pc Using Usb-A Type Connection Ideal For: Users Transitioning To A Pc For Telephony, Webinars, Conference Calls, And Other Applications

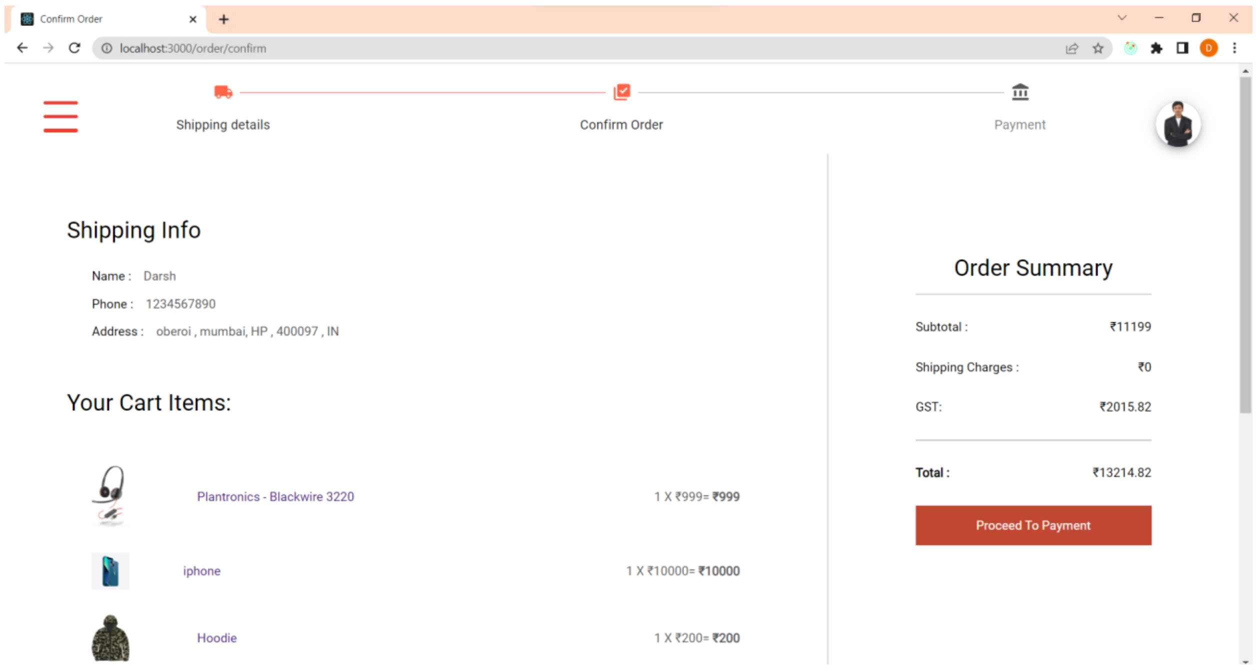
Submit Review



Product	Quantity	Subtotal
Plantronics - Blackwire 3220 Price : ₹999 Remove	- 1 +	₹999
iphone Price : ₹10000 Remove	- 1 +	₹10000
Hoodie Price : ₹200 Remove	- 1 +	₹200

Gross Total 11199

Check Out



Shipping Info

Name : Darsh

Phone : 1234567890

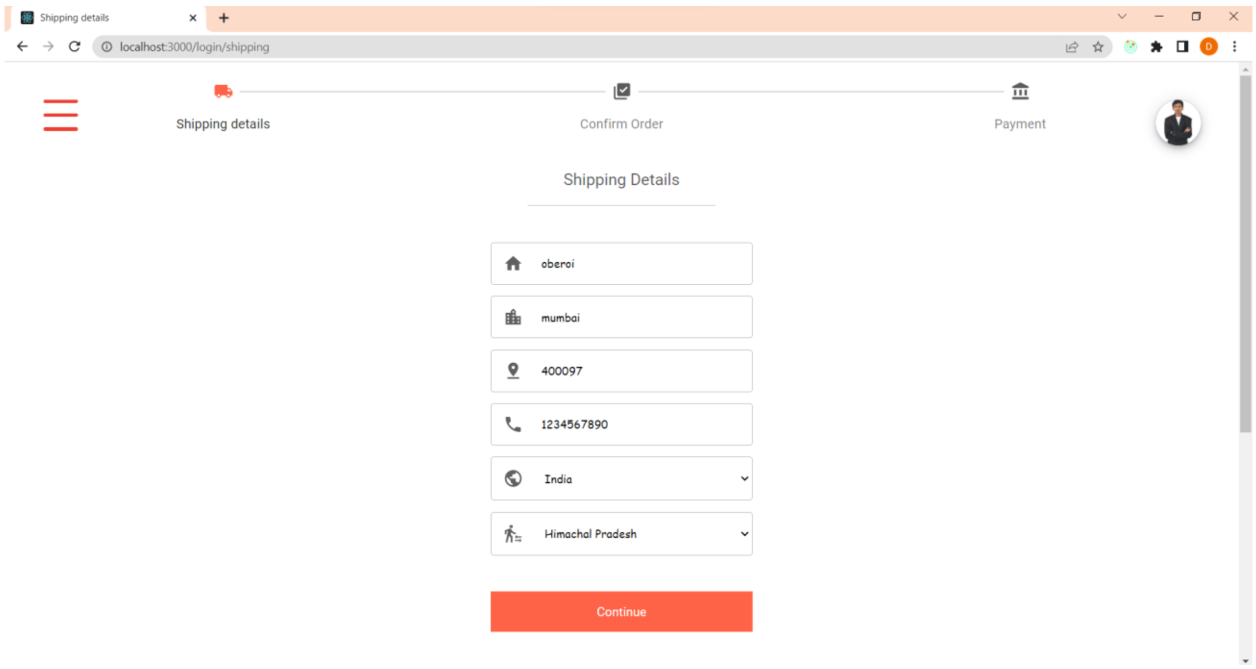
Address : oberoi , mumbai, HP , 400097 , IN

Your Cart Items:

Plantronics - Blackwire 3220	1 X ₹999= ₹999
iphone	1 X ₹10000= ₹10000
Hoodie	1 X ₹200= ₹200

Order Summary

Subtotal :	₹11199
Shipping Charges :	₹0
GST:	₹2015.82
Total :	₹13214.82



Shipping Details

oberoi

mumbai

400097

1234567890

India

Himachal Pradesh

11. Known Issues

- **Payment Gateway**: Some users may experience issues with the Stripe/PayPal integration due to incorrect configuration of API keys or network errors.

- **Product Search**: The search functionality for products is limited and may not be as advanced as desired (e.g., filtering by multiple categories).
- **Mobile Responsiveness**: Minor UI issues may occur on older devices or browsers.

12. Future Enhancements

- **Real-Time Updates**: Implement real-time product availability updates using WebSockets.
- **Advanced Search**: Enhance the search functionality with multi-filtering capabilities.
- **Admin Analytics**: Improve the admin panel with advanced sales and customer analytics.
- **Subscription-Based Model**: Add subscription options for recurring orders or premium memberships.

This comprehensive report outlines the architecture, features, and development process for **Shopsy**, an e-commerce application built using the MERN stack. The platform is designed to be scalable, user-friendly, and secure, providing both customers and administrators with the tools needed for a successful online store.