**Full Stack Development with MERN**

**1. Introduction**

**Project Title:** MERN Stack E-commerce Website  
**Team Members:**

* Surya SJ [au412721205050]
* Ranjith R [au412721205038]
* Vignesh A [au412721205057]
* Shylus Clinton Geoge S [au412721205045]
* Juswanth V [au412721205023]

**2. Project Overview**

**Purpose:**  
This project aims to create a fully functional e-commerce web application that allows users to browse products, add items to their cart, and complete purchases.

**Features:**

* User authentication and authorization
* Product browsing and search functionality
* Shopping cart management
* Checkout and payment integration
* Admin dashboard for product and order management

**3. Architecture**

**Frontend:**

The frontend is developed using **React** and styled with **MUI (Material-UI)** and **Tailwind CSS**. State management is handled by **Redux Toolkit**, and forms are managed with **React Hook Form**.

**Backend:**

The backend is built with **Node.js** and **Express.js**. It handles RESTful API routes for user authentication, product management, and order processing. Middleware such as **cors**, **cookie-parser**, and **morgan** is used to handle cross-origin requests, cookies, and HTTP logging, respectively.

**Database:**

The project uses **MongoDB** for data storage, with **Mongoose** as the ODM. The database schema includes collections for users, products, orders, and cart items.

**4. Setup Instructions**

**Prerequisites:**

* **Node.js** (v14 or higher)
* **MongoDB** (local or cloud instance)
* **Nodemon** (for development)

**Installation:**

1. Clone the repository:

git clone [repository-url]

1. Navigate to the frontend directory and install dependencies:
2. cd frontend

npm install

1. Navigate to the backend directory and install dependencies:
2. cd backend

npm install

1. Set up environment variables:
   * Create a .env file in the backend directory with the following:
   * MONGO\_URI=[Your MongoDB connection string]
   * JWT\_SECRET=[Your secret key]

PORT=[Port number]

**5. Folder Structure**

**Client (Frontend):**

frontend/

|-- public/

|-- src/

|-- components/

|-- pages/

|-- redux/

|-- styles/

|-- App.js

|-- index.js

**Server (Backend):**

backend/

|-- config/

|-- controllers/

|-- middleware/

|-- models/

|-- routes/

|-- seed/

|-- index.js

**6. Running the Application**

**Frontend:**

cd frontend

npm start

**Backend:**

cd backend

npm start

For development:

npm run dev

**7. API Documentation**

**User Endpoints:**

* **POST** /api/auth/register
  + Request: { "username": "", "password": "" }
  + Response: { "message": "User registered successfully" }
* **POST** /api/auth/login
  + Request: { "username": "", "password": "" }
  + Response: { "token": "JWT\_TOKEN" }

**Product Endpoints:**

* **GET** /api/products
* **POST** /api/products
  + Request: Admin authorization required.

**8. Authentication**

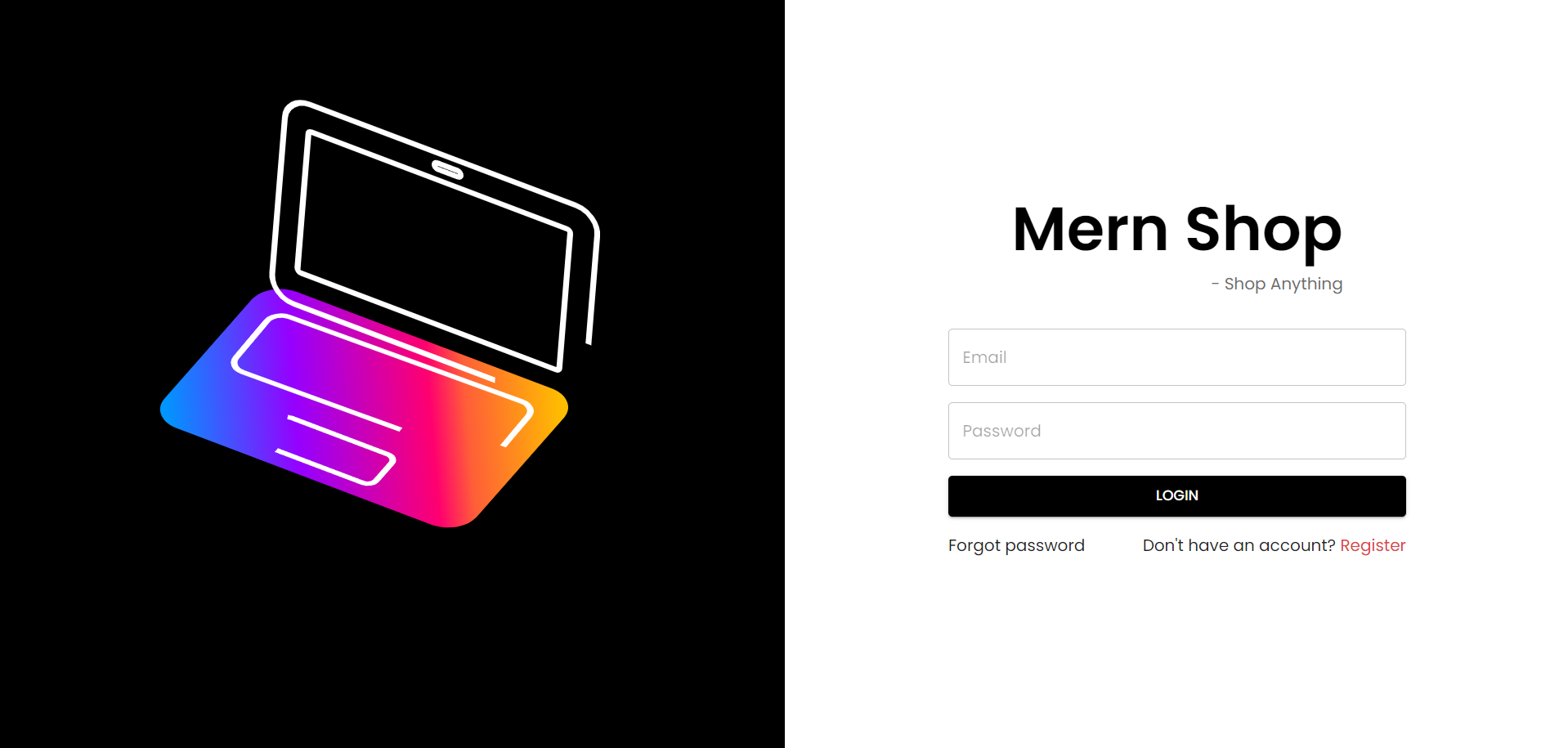
**Method:**  
JWT tokens are used for user authentication. Tokens are generated upon successful login and stored in HTTP-only cookies for security.

**Authorization:**  
Role-based access is implemented to distinguish between admin and regular users.

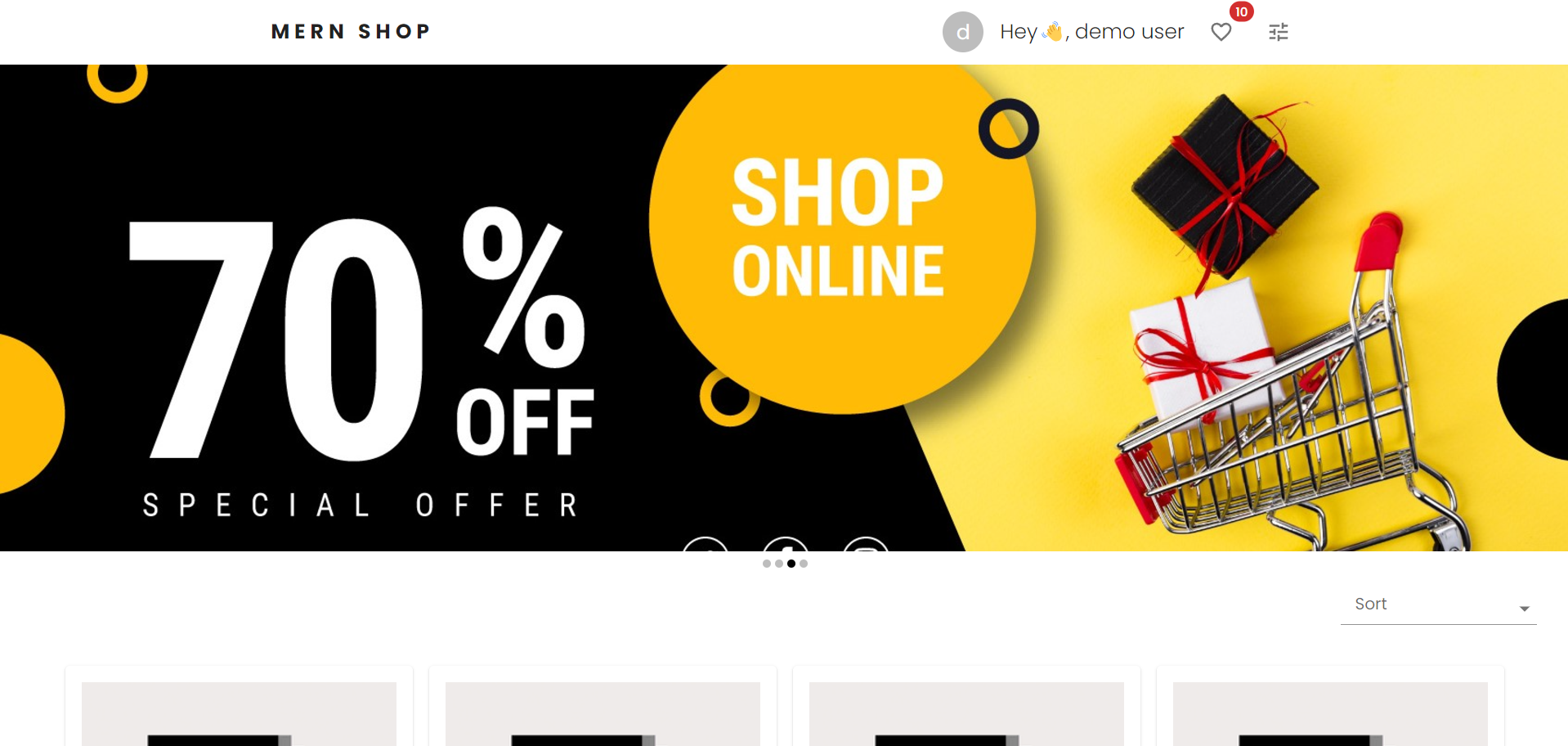
**9. User Interface**

Screenshots showcasing:

* Home page



* Product listing

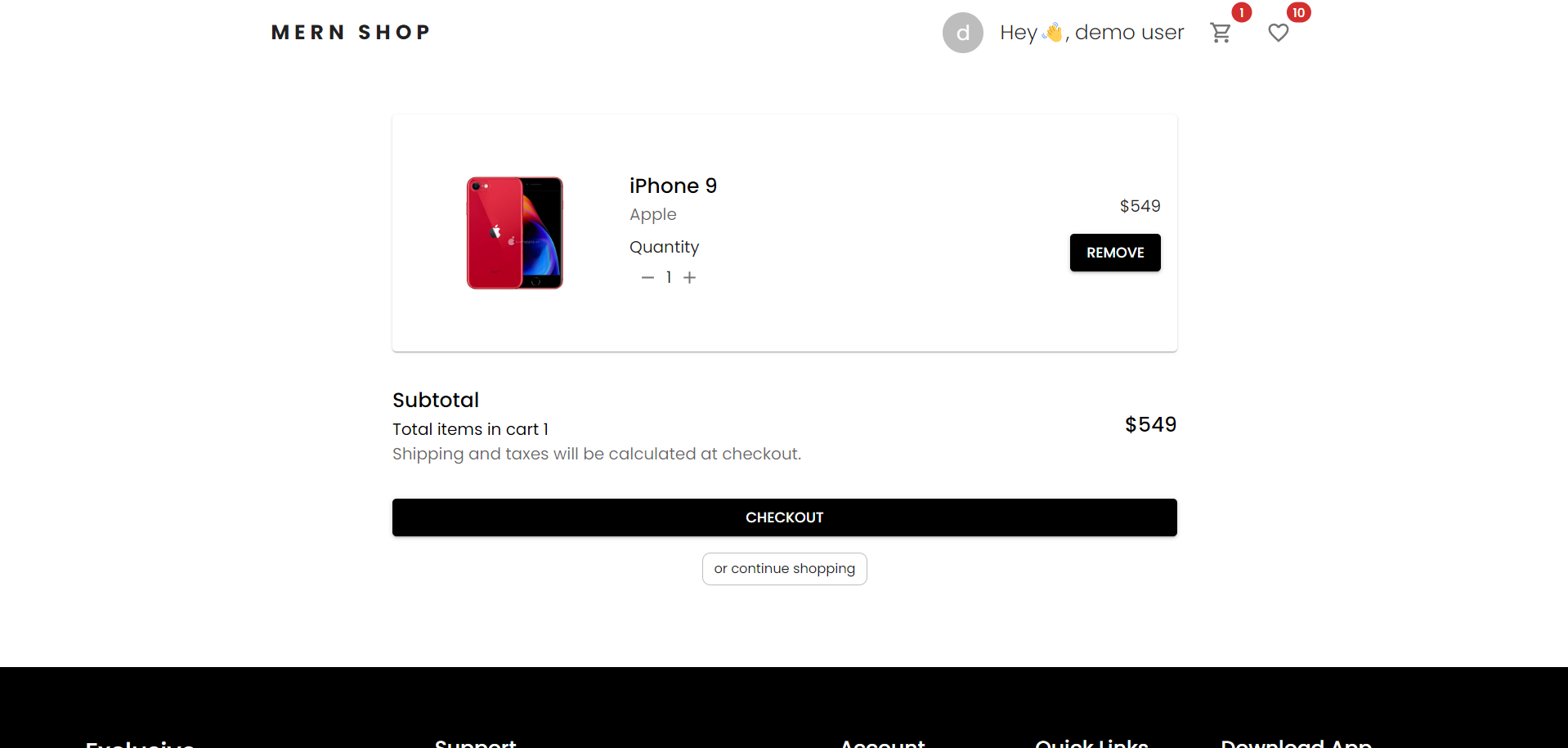


* Product detail

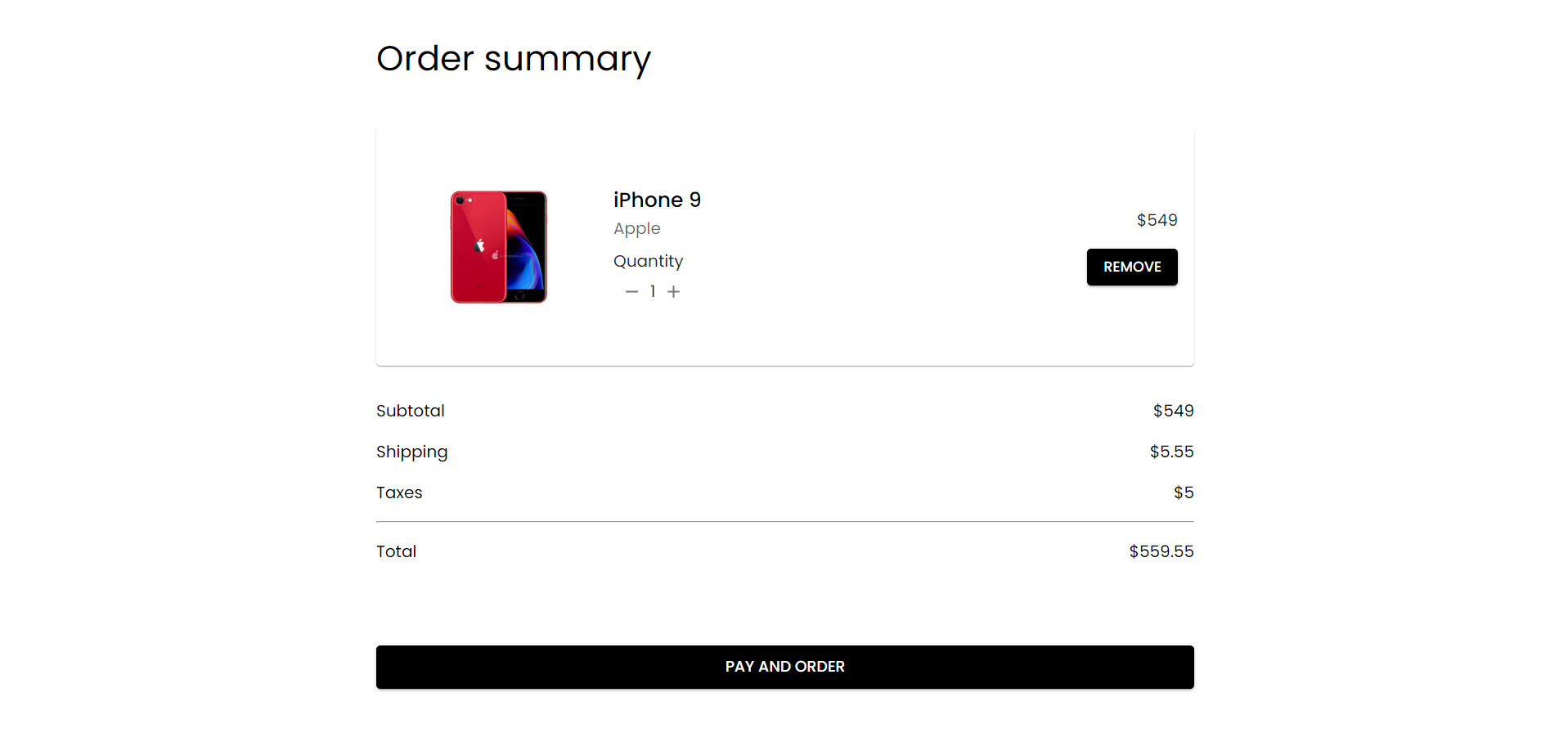
A red cell phone with a black screen

Description automatically generated

* Shopping cart



* User profile and checkout



**10. Testing**

**Strategy:**  
Unit testing is done using **Jest** and **React Testing Library**. The backend is tested using **Mocha** and **Chai**.

**11. Screenshots or Demo**

**https://drive.google.com/file/d/1\_ke7nE8PBl1KEy2jZfRf-BPfP8QtynPa/view?usp=sharing**

**12. Known Issues**

* Certain API endpoints may have performance issues under heavy load.
* Minor UI inconsistencies in mobile view.

**13. Future Enhancements**

* Integrate payment gateway (e.g., Stripe, PayPal)
* Add product reviews and ratings
* Implement push notifications for order status updates