WEB BASED STOREFRONT

A PROJECT REPORT

Submitted by:

Abhik Gupta (09514902021)

John P Varghese (04714902021)

Yatin Hooda (02914902021)

Nishchay Chandok (35914902021)

In partial fulfillment for the award of the degree

of

BACHELOR OF COMPUTER APPLICATION

MAHARAJA SURAJMAL INSTITUTE

C4, JANAKPURI, NEW DELHI – 110058

MAY 2024

DECLARATION

We affirm that this submission is the result of our individual effort, and we attest that, to the best of our knowledge and belief, it does not incorporate any material previously published or authored by another individual. Furthermore, we confirm that this work has not been utilized in whole or in part for the completion of any other academic degree or diploma from any university or educational institution. Any instances where external contributions have been acknowledged are duly cited within the text.

Name: Abhik Gupta
Enrollment Number: 09514902021
Signature:
Date:

Name: Yatin Hooda
Enrollment Number: 02914902021
Signature:
Date:

Name: John P Varghese
Enrollment Number: 04714902021
Signature:
Date:

Name: Nishchay Chandok
Enrollment Number: 35914902021
Signature:
Date:

BONAFIDE CERTIFICATE

I certify that the project report titled **"WEB BASED STOREFRONT"** is a bonafide, authentic and genuine work and hard work of the team comprising **“Abhik Gupta, John P Varghese, Yatin Hooda, and Nishchay Chandok”**. This team diligently conducted the project under my supervision, and I can attest to the integrity and originality of their work.

SUPERVISOR

Mr. Sundeep Kumar

Assistant professor

Dept. of Computer Application

SIGNATURE OF SUPERVISOR

SELF CERTIFICATE

This is to certify that the dissertation/project report entitled **"WEB BASED STOREFRONT"** is done by me is an authentic work carried out for the partial fulfilment of the requirements for the award of the degree of Bachelor of Computer Applications under the guidance of **"Mr. Sundeep Kumar"**. The matter embodied in this project work has not been submitted earlier for award of any degree or diploma to the best of my knowledge and belief.

ABHIK GUPTA

09514902021

(TEAM LEADER)

ACKNOWLEDGEMENT

Certified We would like to take this opportunity to acknowledge everyone who has helped us in every stage of this project.

We extend our heartfelt appreciation to all those who contributed to the successful realization of this project. A special acknowledgment goes to **Mr. Sundeep Kumar** from the **Department of Computer Science** for his invaluable guidance and insightful suggestions, which played a pivotal role in bringing this project to fruition. We are sincerely grateful for his unwavering support.

Furthermore, our sincere thanks go to our esteemed **Director, Dr. Harish Singh**, whose provision of essential facilities significantly contributed to the seamless completion of this project. We express our gratitude for his support and encouragement throughout the entire process.

ABHIK GUPTA

JOHN P VARGHESE

YATIN HOODA

NISHCHAY CHANDOK

BCA 5A (MORNING)

TABLE OF CONTENTS

CHAPTER NO.	TITLE	PAGE NUMBER
1.	Synopsis 1. Objective 2. Scope 3. What is Vendr? 4. Why did we choose Vendr?	1 - 3
2.	Requirements • Hardware Requirements • Software Requirements	4-5
3.	Abstract of the project.	6
4.	Working of the application	7 - 10
5.	Tech Stack being used • Frontend • Backend	11 - 24
6.	SRS • Functional Requirements • Interface Requirements • Performance Requirements • Design Constraints • Non-Functional Attributes • Schedule and Plan	25 - 33

7.	Software Design • Level-0 DFD of project • Level-1 DFD of project • Use Case Diagram of project • ER Diagram of database structure	34 - 38
8.	Team Description	39
9.	Coding and Implementation	40 - 348
10.	Output Screenshots	349 - 370
11.	Version Control and Collaboration	371 - 374
12.	Conclusion	375
13.	Future Scope	376 – 377
14.	References	378

SYNOPSIS

OBJECTIVE

The objective of the Vendr project is to develop a robust and user-friendly e-commerce platform that facilitates seamless online shopping experiences. By leveraging modern technologies like the MERN stack (MongoDB, Express, React, and Node.js), Vendr aims to provide users with a secure and intuitive interface for browsing, selecting, and purchasing products. The platform also integrates essential features such as user authentication, secure payments via SquareUp, and real-time order tracking to ensure a comprehensive and satisfying shopping journey.

SCOPE

The scope of Vendr encompasses both user-facing and administrative functionalities to cover the full spectrum of e-commerce operations. For users, Vendr offers pages for login, signup, product browsing, cart management, checkout, and order tracking. The platform also includes password recovery, payment processing, and user profile management. On the administrative side, Vendr features an admin panel with capabilities for managing products, processing orders, and handling user queries. The use of MongoDB Atlas for cloud-based data storage ensures scalability and reliability, while deployment on Vercel ensures high availability and performance.

SUMMARY

Vendr is a feature-rich e-commerce application designed to streamline the online shopping experience for both buyers and sellers. Leveraging the power of the MERN stack, Vendr offers a responsive and scalable platform that caters to the diverse needs of modern e-commerce. With secure authentication mechanisms, robust payment processing, and efficient email functionality, Vendr prioritizes user security and satisfaction. Whether it's browsing through a vast array of products, making secure transactions, or managing orders with ease, Vendr provides a comprehensive solution for all e-commerce requirements. By harnessing cutting-edge technologies and adhering to best practices, Vendr sets a new standard for online retail platforms, promising a seamless and enjoyable shopping experience for users worldwide.

WHAT IS VENDR?

Vendr is a dynamic and fully functional e-commerce application crafted to revolutionize the online shopping experience. At its core, Vendr is a sophisticated platform that connects buyers and sellers in a seamless digital marketplace. Built using the MERN (MongoDB, Express.js, React.js, Node.js) stack, Vendr boasts a modern and intuitive user interface powered by React and Redux, ensuring a smooth and responsive browsing experience. With Express.js handling API requests and MongoDB Atlas providing a secure cloud-based database solution, Vendr offers a robust infrastructure that supports the entire e commerce process, from product discovery to order fulfilment. By integrating SquareUp for secure payment processing, Nodemailer for efficient email communication, and JWT with passport for authentication, Vendr prioritizes user security and satisfaction, setting a new standard for online retail platforms.

WHY DID WE CHOOSE VENDR?

Vendr was chosen as a project for its potential to address the growing demand for innovative e-commerce solutions. In today's digital age, online shopping has become an integral part of daily life for millions worldwide. However, existing e-commerce platforms often lack the customization, security, and scalability needed to meet the evolving needs of both buyers and sellers. By developing Vendr, we aimed to fill this gap by creating a comprehensive e-commerce solution that leverages cutting-edge technologies and best practices. Vendr's modular architecture and extensive feature set make it an ideal choice for businesses looking to establish or enhance their online presence. Moreover, by open-sourcing Vendr, we encourage collaboration and contribution from the developer community, fostering innovation and continuous improvement in the realm of e-commerce technology. Ultimately, choosing Vendr as a project aligns with our commitment to delivering impactful and scalable solutions that empower businesses and elevate the online shopping experience for consumers worldwide.

REQUIREMENTS

HARDWARE REQUIREMENTS

1. Servers with adequate CPU, RAM, and storage resources to accommodate anticipated traffic and data storage needs.
2. Reliable network infrastructure including routers, switches, and firewalls for seamless communication.
3. Consideration of a load balancer for distributing incoming requests across multiple servers, particularly for scalability.
4. Compatibility with hardware associated with integrated payment gateways like SquareUp, such as card readers or terminals.

SOFTWARE REQUIREMENTS

1. Compatible operating system (e.g., Linux, Windows Server) installed on servers.
2. Node.js and npm (Node Package Manager) for running the backend application.
3. MongoDB for database management, either installed locally or hosted on MongoDB Atlas.
4. Express.js for building backend APIs and handling HTTP requests.
5. React and Redux libraries for frontend development, ensuring interactive user interfaces and efficient state management.
6. Integration of SquareUp SDK for secure payment transactions.
7. Nodemailer for handling email functionalities such as sending order confirmations and notifications.
8. Authentication managed using JWT (JSON Web Tokens) and Passport.js.
9. Deployment facilitated by tools like Vercel or Docker for production environments.

10. Implementation of monitoring and logging tools for tracking application performance and troubleshooting issues effectively.
11. This combination of hardware and software provides a robust infrastructure capable of supporting the complex requirements of modern e-commerce application development, ensuring that the team can develop, test, and deploy efficiently and effectively.

ABSTRACT

Embarking on the Vendr project has been an exhilarating journey into the heart of e-commerce innovation. Vendr stands as a testament to our dedication to crafting a cutting-edge platform that redefines the online shopping experience. Leveraging the power of the MERN (MongoDB, Express.js, React.js, Node.js) stack, Vendr represents a fusion of modern technology and user-centric design, aimed at empowering both buyers and sellers in the digital marketplace.

At its core, Vendr embodies a comprehensive suite of features meticulously crafted to meet the diverse needs of our users. With a strong emphasis on scalability, security, and seamless functionality, Vendr offers an intuitive platform for browsing, purchasing, and managing products effortlessly. From secure user authentication to efficient order processing, every aspect of Vendr is designed to enhance the user experience and drive business growth.

As we continue to refine and expand Vendr, our vision extends beyond mere transactions to fostering meaningful connections and experiences. Future enhancements such as multi-vendor support, social commerce integration, and personalized recommendations exemplify our commitment to staying at the forefront of e-commerce innovation. With Vendr, we are not just building a platform; we are shaping the future of online retail, one innovative feature at a time.

WORKING OF THE APPLICATION

1. Vendr prioritizes users and works on a User Centric mode.
2. Users can start with a smooth sign-up process where they are needed to provide essential details like email and password on the Sign-Up Page.
3. Upon successful registration, users are automatically logged in. Returning users can log in through the Login Page, where they enter their email and password for secure login using JWT and PassportJS for end-to-end encryption and authentication.
4. The user interface is intuitive after logging in.
5. Users are greeted with a product listing page showcasing a variety of products across categories and brands. For a large number of products, the application presents a paginated view.
6. Users can sort results based on price and ratings.
7. Users can also categorize items based on product category or brand.
8. Two additional pages are accessible through the Navbar: "About Us" and "Contact Us." The "About Us" page is a static page with a brief application description. The "Contact Us" page has a form with a single input field for users to enter their queries to be sent to admins.
9. To purchase a product, users click on the desired item, which opens the product description page.
10. The product description page offers a detailed view, including highlights, description, price, ratings, multiple images, and available sizes and colors.

11. Users can simply choose a color and size and click "Add to Cart" to add the product to their cart.
12. After adding products, users can access the cart through the cart button in the Navbar for a final review before checkout.
13. On the cart page, users can modify the quantity of an item or even remove it entirely from the cart.
14. Once users finalize their cart, they can proceed to checkout by clicking the checkout button.
15. The checkout page offers a form at the top to add a delivery address if not already provided. Once an address is added, users can select it from their saved addresses and choose a payment method.
16. Two payment methods are available: Cash and Card. Cash payments confirm the order immediately. Card payments open a dedicated payment gateway to complete the transaction and confirm the order upon successful payment.
17. Upon successful order placement, an invoice with the product list, subtotal, and individual product prices is sent to the email address provided during checkout.
18. Once the order is confirmed, users can view a detailed view of the order from the "My Orders" page accessible through the User Icon at the top right. This page displays the order, subtotal, total number and description of products ordered, along with the order status.

19. The User Icon at the top right also provides access to the "My Profile" page where users can view their saved email, phone number, and addresses, and even edit or remove this data.
20. Users can log out of the application by clicking the User Icon at the top right and then selecting "Sign Out."
21. The application also provides an "Admin Panel" accessible through an Admin account login.
22. Upon logging in with an Admin account, the admin can perform everything a regular user can, such as purchasing items, viewing the "My Profile" page, or viewing the "My Orders" page. However, the Admin Panel has three additional options: "Admin Products," "Admin Orders," and "Admin Queries."
23. The "Admin Products" page displays the entire product listing with an "Edit Product" button for each product. Clicking this button opens a page form where the admin can edit product details. The "Admin Products" page also has an "Add Product" button at the top, which opens a form page for the Admin to add a new product.
24. The "Admin Orders" page displays all orders created by various users, along with each order's ID, purchased items, payment method, delivery address, payment status, and order status. The Admin can also update the Order status and Payment status of any order from this page.

25. The "Admin Queries" page displays all queries submitted by regular users through the "Contact Us" section.
26. If a user or admin forgets their password while logging in, they can simply click the "Forgot Password Button" on the Login Page. Here, they can enter their email address, and an email will be sent with a link that brings the user back to the application to set a new password, which ultimately changes the user's password upon submission.

TECH STACK USED FOR DEVELOPMENT

PROJECT STRUCTURE:

1. **Client-Side (Frontend):**
    a. Developed using React, React Router DOM, and various UI libraries like Tailwind for enhanced functionality and user experience.
    b. Used React Redux to handle states and a store to manage the application's frontend organized and feature-rich.

2. **Server-Side (Backend):**
    a. Built on Node.js and Express, facilitating the creation of a scalable and efficient server.
    b. Utilizes MongoDB as the database, with Mongoose for data modeling and interactions.
    c. Employs Babel for ECMAScript compatibility and provides additional middleware for features like compression, CORS, and validation.

3. **Note:**

   a. The technology stack combines the strengths of React and Redux for the dynamic frontend, Node.js and Express for the robust backend, MongoDB for flexible and scalable data storage, and various libraries and tools to enhance functionality, security, and development efficiency. This stack ensures a modern, efficient, and scalable architecture for the Vendr project.

FRONTEND DEPENDENCIES (CLIENT):

1. **@headlessui/react**
   - Version: ^1.7.18
   - Description: @headlessui/react is a library that provides accessible and composable React components for building user interfaces without styling constraints.

2. **@heroicons/react**
   - Version: ^2.1.3
   - Description: @heroicons/react offers a set of customizable React components for incorporating Heroicons, a collection of free SVG icons, into your React applications.

3. **@reduxjs/toolkit**
   - Version: ^1.9.7
   - Description: @reduxjs/toolkit is a package that simplifies Redux usage by providing utilities for writing Redux logic, including reducers, actions, and middleware, in a more efficient and ergonomic manner.

4. **@stripe/react-stripe-js**

- Version: ^2.7.0
- Description: @stripe/react-stripe-js facilitates integration with the Stripe payment platform in React applications, allowing for secure and seamless payment processing.

5. **@stripe/stripe-js**

- Version: ^3.3.0
- Description: @stripe/stripe-js is the Stripe JavaScript library for handling client-side interactions with Stripe, such as creating payment methods and handling card validations.

6. **@tailwindcss/aspect-ratio**

- Version: ^0.4.2
- Description: @tailwindcss/aspect-ratio is a Tailwind CSS plugin that provides utilities for creating responsive aspect-ratio boxes, allowing for consistent sizing of elements based on aspect ratios.

7. **@tailwindcss/forms**

- Version: ^0.5.7
- Description: @tailwindcss/forms is a Tailwind CSS plugin that provides styles for form elements, such as input fields, checkboxes, and radio buttons, ensuring consistent and customizable form styling.

8. **@testing-library/jest-dom**

- Version: ^5.17.0
- Description: @testing-library/jest-dom offers Jest matchers for asserting expectations on DOM elements, making it easier to write and maintain test cases for React components.

9. **@testing-library/react**

- Version: ^13.4.0
- Description: @testing-library/react provides utilities for testing React components, including rendering components, interacting with them, and querying the DOM for testing purposes.

10. @testing-library/user-event

- Version: ^14.5.2
- Description: @testing-library/user-event offers utilities for simulating user interactions, such as clicking, typing, and navigating, in testing environments, allowing for comprehensive testing of user interactions in React components.

11. json-server

- Version: 0.17.0
- Description: json-server is a lightweight server that serves JSON data over HTTP, making it useful for mocking APIs and simulating backend services during development and testing.

12. react

- Version: ^18.2.0
- Description: React is a JavaScript library for building user interfaces, providing a declarative and component-based approach to building interactive web applications.

13. react-dom

- Version: ^18.2.0
- Description: react-dom is a package that provides DOM-specific methods for working with React components, enabling rendering of React applications in web browsers.

14. react-hook-form

- Version: ^7.51.3
- Description: react-hook-form is a library for building performant and flexible forms in React applications, offering hooks-based form validation and management capabilities.

15. react-loader-spinner

- Version: ^6.1.6
- Description: react-loader-spinner provides customizable loading spinners for React applications, offering visual feedback to users during asynchronous operations and data loading processes.

16. react-redux

- Version: ^8.1.3
- Description: react-redux is the official React bindings for Redux, enabling integration of Redux state management with React components, allowing for efficient and predictable management of application state.

17. react-router-dom

- Version: ^6.22.3
- Description: react-router-dom provides routing and navigation functionalities for React applications, allowing for declarative and dynamic routing of components based on URL paths.

18. react-scripts

- Version: 5.0.1
- Description: react-scripts is a set of scripts and configuration files for bootstrapping and running React applications, including development servers, build scripts, and testing utilities.

19. react-square-web-payments-sdk

- Version: ^3.2.1
- Description: react-square-web-payments-sdk facilitates integration with Square's Web Payments SDK in React applications, enabling secure and streamlined payment processing using Square's payment platform.

20. react-toastify

- Version: ^10.0.5
- Description: react-toastify provides toast notification capabilities for React applications, offering customizable and user-friendly notifications for displaying messages, alerts, and feedback to users.

21. web-vitals

- Version: ^2.1.4
- Description: web-vitals is a library for measuring and reporting essential performance metrics of web pages, including metrics such as First Contentful Paint (FCP), Largest Contentful Paint (LCP), and Cumulative Layout Shift (CLS).

BACKEND DEPENDENCIES (SERVER):

1. **cookie-parser**
   - Version: ^1.4.6
   - Description: cookie-parser is a middleware for parsing cookies in Express applications, enabling easy access to cookie data in request objects.

2. **cors**
   - Version: ^2.8.5
   - Description: cors is a middleware for enabling Cross-Origin Resource Sharing (CORS) in Express applications, allowing for controlled access to resources from different origins.

3. **dotenv**
   - Version: ^16.4.5
   - Description: dotenv is a module for loading environment variables from a .env file into Node.js applications, providing a convenient way to manage configuration settings.

4. **express**

- Version: ^4.19.2
- Description: express is a fast, unopinionated, and minimalist web framework for Node.js, providing a robust set of features for building web applications and APIs.

5. **express-session**

- Version: ^1.18.0
- Description: express-session is a middleware for managing session data in Express applications, enabling the creation and management of user sessions with session-based authentication.

6. **jsonwebtoken**

- Version: ^9.0.2
- Description: jsonwebtoken is a library for generating and verifying JSON Web Tokens (JWT) in Node.js applications, facilitating secure authentication and authorization mechanisms.

7. **mongoose**

- Version: ^8.3.2
- Description: mongoose is an Object Data Modeling (ODM) library for MongoDB and Node.js, providing a schema-based solution for modeling application data and interacting with MongoDB databases.

8. **nodemailer**

- Version: ^6.9.13
- Description: nodemailer is a module for sending email messages from Node.js applications, offering a simple and efficient API for configuring and sending emails via SMTP or other transport methods.

9. **nodemon**

- Version: ^3.1.0
- Description: nodemon is a utility for automatically restarting Node.js applications when file changes are detected, providing a convenient development workflow for server-side development.

10. passport

- Version: ^0.7.0
- Description: passport is an authentication middleware for Node.js applications, offering a flexible and modular approach to implementing authentication strategies, such as OAuth, JWT, and local authentication.

11. passport-jwt

- Version: ^4.0.1
- Description: passport-jwt is a Passport strategy for authenticating with JSON Web Tokens (JWT) in Node.js applications, enabling secure and stateless authentication mechanisms using JWT tokens.

12. passport-local

- Version: ^1.0.0
- Description: passport-local is a Passport strategy for authenticating with a username and password in Node.js applications, providing a simple and customizable approach to implementing local authentication.

13. path

- Version: ^0.12.7
- Description: path is a core module in Node.js for working with file and directory paths, providing utilities for resolving, joining, and manipulating file paths in a platform-independent manner.

14. stripe

- Version: ^15.4.0
- Description: stripe is a Node.js library for interacting with the Stripe API, enabling integration with Stripe's payment platform for processing payments, managing subscriptions, and handling other e-commerce functionalities.

SOFTWARE REQUIREMENT SPECIFICATIONS

(SRS)

Functional Requirements:

1. **User Registration and Authentication**: Users should be able to register for an account securely, providing essential details such as email address and password. The system should authenticate users upon login, ensuring only authorized access to user accounts.

2. **Product Management**: The platform should support comprehensive product management functionalities for administrators, allowing them to add, edit, and delete products. Users should be able to browse and search for products based on various criteria, such as category, price range, and brand.

3. **Shopping Cart and Checkout**: Users should be able to add products to their shopping cart, review items, adjust quantities, and proceed to checkout seamlessly. The checkout process should support multiple payment methods, including credit/debit cards, digital wallets, and cash on delivery.

4. **Order Management**: The system should facilitate efficient order management for both users and administrators. Users should be able to track order statuses, view order history, and receive email notifications for order updates. Administrators should have tools to manage orders, update statuses, and handle customer inquiries.

5. **User Profile Management**: Users should have access to a profile page where they can view and edit their personal information, such as email address, phone number, and delivery addresses. Administrators should also have access to user profiles for administrative purposes.

6. **Admin Panel**: The system should provide administrators with a dedicated admin panel for managing various aspects of the platform, including product listings, orders, and user accounts. The admin panel should offer functionalities such as product CRUD operations, order tracking, and user management.

Interface Requirements:

1. **User-Friendly Interface**: The user interface should be intuitive and easy to navigate, providing a seamless shopping experience for users of all levels of technical proficiency.

2. **Responsive Design**: The application should be accessible across various devices and screen sizes, including desktops, laptops, tablets, and smartphones. The interface should adapt dynamically to different screen resolutions and orientations.

3. **Clear Product Presentation**: Product listings should include clear and concise information, including product images, descriptions, prices, and availability. Users should be able to easily browse and filter products based on their preferences.

4. **Shopping Cart and Checkout Process**: The shopping cart interface should display a summary of selected items, with options to adjust quantities and remove items as needed. The checkout process should be streamlined and include clear prompts for entering delivery information and selecting payment methods.

5. **Order Tracking**: Users should be able to track the status of their orders through a dedicated interface, providing real-time updates on order processing, shipping, and delivery.

6. **Admin Panel Interface**: The admin panel interface should provide administrators with access to a range of management tools, including product management, order processing, and user account management. The interface should be organized and user-friendly, allowing administrators to perform tasks efficiently.

Performance Requirements:

1. **Response Time**: The system should respond to user interactions, such as browsing products, adding items to the cart, and completing orders, within acceptable time frames. The response time for each operation should be optimized to ensure a smooth and seamless user experience.

2. **Scalability**: Vendr should be able to handle a large volume of concurrent users and transactions without experiencing performance degradation. The system should scale

horizontally to accommodate increased traffic during peak periods, such as seasonal sales or promotional events.

3. **Reliability**: The system should be highly reliable, with minimal downtime and service interruptions. Measures should be in place to ensure continuous availability, including redundant servers, load balancing, and automated failover mechanisms.

4. **Database Performance**: The database should be optimized for performance, with efficient indexing, query optimization, and data caching strategies implemented to minimize response times for database operations.

5. **Page Load Speed**: Web pages should load quickly, with minimal latency and page load times. Optimizations such as code minification, image compression, and caching should be implemented to reduce page load times and improve overall performance.

6. **Transaction Processing**: Transaction processing should be fast and efficient, with orders processed promptly and accurately. Payment processing should be secure and reliable, with transactions completed in a timely manner and confirmation emails sent to users promptly upon order completion.

Design Constraints:

1. **Technology Stack Limitations**: Vendr is constrained by the technology stack chosen for development, including the MERN (MongoDB, Express.js, React.js, Node.js) framework. Any design decisions must align with the capabilities and limitations of these technologies.

2. **Third-Party Integration Compatibility**: Vendr relies on various third-party services and APIs for functionalities such as payment processing, email notifications, and authentication. Design decisions must consider the compatibility and integration requirements of these services.

3. **Security and Compliance Requirements**: Vendr must adhere to industry standards and best practices for security and compliance, including data encryption, secure payment processing, and user authentication. Design decisions must prioritize security and ensure compliance with regulations such as GDPR and PCI DSS.

4. **Scalability and Performance**: The design of Vendr must support scalability and performance requirements, allowing the platform to handle increasing traffic and user demand over time. Design decisions must consider scalability challenges such as database optimization, caching strategies, and load balancing.

5. **User Experience Considerations**: Vendr's design must prioritize user experience and accessibility, ensuring that the platform is intuitive, easy to navigate, and responsive

across different devices and screen sizes. Design decisions must consider usability testing, user feedback, and accessibility standards to enhance the overall user experience.

6. **Maintenance and Upgradability**: The design of Vendr should facilitate maintenance and future upgrades, allowing for seamless updates and enhancements to the platform. Design decisions must consider modularity, code maintainability, and version control practices to support ongoing development and improvements.

Non-functional Attributes:

1. **Security**: Vendr prioritizes robust security measures to protect user data, sensitive information, and transactions. Implementation of encryption protocols, secure authentication mechanisms, and adherence to industry security standards ensures data confidentiality and integrity.

2. **Performance**: Vendr is designed for optimal performance, with fast response times, minimal latency, and high availability. Through efficient code optimization, database tuning, and server infrastructure scaling, Vendr delivers a seamless and responsive user experience, even under heavy loads.

3. **Scalability**: Vendr's architecture is built to scale horizontally, accommodating increasing user traffic and data volumes without compromising performance or

reliability. With flexible deployment options and scalable infrastructure components, Vendr adapts effortlessly to evolving business needs and growth demands.

4. **Reliability**: Vendr guarantees high reliability and uptime, minimizing service disruptions and downtime. Redundant systems, failover mechanisms, and automated recovery processes ensure continuous availability, enabling uninterrupted access to the platform for users and administrators.

5. **Accessibility**: Vendr is committed to accessibility, ensuring that the platform is usable and navigable by users of all abilities. Compliance with accessibility standards, such as WCAG (Web Content Accessibility Guidelines), and inclusive design practices ensure equal access to information and functionality for all users.

6. **Maintainability**: Vendr's codebase is structured for maintainability, with clean, modular, and well-documented code that facilitates ongoing development, debugging, and enhancements. Version control, coding standards, and documentation practices ensure ease of maintenance and future scalability.

Project Plan:

1. **Requirements Gathering**: The project plan begins with a comprehensive requirement gathering phase, where stakeholders' needs and expectations are identified, documented, and prioritized. This phase involves conducting user interviews, analyzing competitor platforms, and defining the scope and objectives of the project.

2. **System Design**: Following requirements gathering, the project moves into the system design phase, where the architectural design and technical specifications of Vendr are defined. This phase includes designing the database schema, outlining the user interface wireframes, and selecting the appropriate technologies and frameworks for development.

3. **Development**: With the system design in place, development efforts commence to build the various components of Vendr. This phase involves front-end development using React.js and Redux, back-end development using Node.js and Express.js, database development using MongoDB, and integration of third-party services such as payment gateways and email providers.

4. **Testing**: Concurrent with development, the testing phase ensures the quality and functionality of Vendr through rigorous testing methodologies. This phase includes unit testing, integration testing, and end-to-end testing to identify and resolve any bugs, errors, or issues before deployment.

5. **Deployment**: Once development and testing are complete, Vendr is deployed to a production environment using platforms such as Vercel or AWS. Deployment involves configuring servers, setting up databases, and ensuring proper security measures are in place to protect user data and ensure system stability.

6. **Training and Documentation**: As Vendr nears completion, training sessions are conducted to familiarize administrators with the platform's features and functionalities. Additionally, comprehensive documentation is provided, including user manuals, technical guides, and troubleshooting resources, to support users and administrators post-launch.

Project Schedule:

- **Month 1 - Requirement Analysis, Planning, Design and Prototyping**:
  - Detailed requirements gathering and project planning.
  - Create design prototypes, finalize architecture, and set up the development environment.
- **Month 2 - Development of backend and frontend individually**:
  - Backend implementation with Node.js and Express, frontend implementation with React and Redux.
- **Month 3 - Integration and Testing**:
  - Integrate all components, perform thorough testing (unit, integration, and user acceptance testing).
- **Month 4 - Deployment and Launch**:
  - Deploy the application to Vercel, perform final testing, and launch the application. Post-launch monitoring and initial bug fixes.

SOFTWARE DESIGN

Note: The next 4 pages display the following software design diagrams

1. Level-0 Data Flow Diagram of the project
2. Level-1 Data Flow Diagram of the project
3. Use Case Diagram of the project
4. ER Diagram of the database structure

SquareUp Payments Gateway
ADMIN
Nodemailer API
VENDR
MongoDB Atlas
USER
Request for payment gateway
Provide payments gateway and process payments
Sign up / Login
Provide order and product details
Add product or Edit product / order status
Request for all required data, and edit/create/delete requests
Provide all data on request
Request for mailing service
Send a mail
Sign up / Login
Fill up cart and checkout
Process payments and confirm order

LEVEL 0 DFD

LEVEL 1 DFD
ADMIN
Provide existing products, brands and categories details
Add new / Edit existing products, brands and categories
Edit existing order status
Display all user's orders
ADMIN ORDERS FUNCTIONALITY
ADMIN QUERIES PAGE
Edit existing order status
Provide all orders
View all queries
Provide all queries
Fetch user specific orders
ORDERS DATABASE
QUERIES DATABASE
Add new / Edit existing products, brands and categories
ADMIN PRODUCTS FUNCTIONALITY
Provide user specific orders
Add a query
Provide existing products, brands and categories details
PRODUCTS, BRANDS CATEGORIES DATABASE
Provide products, brands and categories
Fetch products, brands and categories
PRODUCT LISTING FUNCTIONALITY
MY ORDERS PAGE
CONTACT US FUNCTIONALITY
Display product list and product details
View existing orders
Edit quantity or remove item
CART FUNCTIONALITY
Provide items in cart
View product list or detail of a product
USER
Submit a query
Display cart items
CART DATABASE
Add / Remove products to cart
Provide items in cart
CHECKOUT FUNCTIONALITY
View profile or add / edit existing addresses
ROLE = USER
Edit quantity or remove item
BANK
CHECK AUTH FUNCTION
ROLE = ADMIN
MY PROFILE PAGE
Process payments
Authenticate payment and deduct balance
SQUARE-UP PAYMENTS GATEWAY
Logged in user object
Successful checkout
Request for user bank details
LOGIN / SIGNUP FUNCTIONALITY
Provide login / signup credentials
Add / edit user's address
Provide user login details
Request for user's login detail
END USER
GMAIL
NODEMAILER E-MAILING SYSTEM
Provide user specific details
USER DATABASE
Send invoice to mail
Generated invoice
Edit existing addresses or add new address
Provide user's saved addresses

USE CASE DIAGRAM

QUERY
User email
ID
Query
RAISES
M
1
USER
Role
ID
Email
Password
HAS
1
M
ADDRESS
Name
Email
City
Phone number
Locality
House number
State
Pincode
CREATES
1
M
ORDER
User ID
Shipping address ID
Payment mode
ID
Total amount
Total items
Order status
Payment status
CONTAINS
1
M
PRODUCT
ID
Title
Brand
Price
Discount Percentage
Discounted price
Stock
Brand
Category
Description
Images
Available colors
Available sizes
CONTAINS
M
1
CART
ID
Product ID
User ID
Quantity
Color
Size

MEET THE TEAM

Presenting our dynamic team: Abhik Gupta leads with full-stack expertise, Nishchay Chandok weaves frontend magic, John P Varghese shines in backend brilliance, and Yatin Hooda brings database expertise. Together, we craft unforgettable experiences.

ABHIK GUPTA

Full Stack Web Developer

Leads the project as a full-stack web developer, playing a key role in developing both frontend and backend components using React and Node.js. Responsibilities include seamless integration and design contributions to UI and backend structure.

JOHN P VARGHESE

Back end Web Developer

Has played a pivotal role in crafting all API calls using Express. Additionally, he is responsible for setting up the actual server of the project using Node.js. His efforts also encompass the integration of the backend with the frontend.

YATIN HOODA

Database Engineer

Has made substantial contributions to database management. His role involves efficiently managing and incorporating all data into the actual database for testing purposes. He also has made major efforts in the actual development of backend for the project.

NISHCHAY CHANDOK

Front End Web Developer

Has been instrumental and really effective in the creation of the actual React app that is both effective and good looking, taking charge of a significant portion of the frontend development.

CODING AND IMPLEMENTATION

VENDR PROJECT

SERVER

```
    Directory: C:\~VENDR\server


Mode                LastWriteTime         Length Name
----                -------------         ------ ----
d-----        18-05-2024     00:26                build
d-----        18-05-2024     00:26                controller
d-----        18-05-2024     00:26                model
d-----        18-05-2024     00:27                node_modules
d-----        18-05-2024     00:27                routes
d-----        18-05-2024     00:27                services
-a----        10-05-2024     15:15            430 .env
-a----        09-05-2024     00:02             19 .gitignore
-a----        16-05-2024     15:32           7216 index.js
-a----        10-05-2024     14:51          60697 package-lock.json
-a----        10-05-2024     14:51            822 package.json
-a----        09-05-2024     01:59            329 vercel.json
```

package.json

```
{
  "name": "vendr",
  "version": "1.0.0",
  "description": "A full stack e commerce application",
  "main": "index.js",
  "scripts": {
    "start": "node index.js",
    "dev": "nodemon index.js",
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "keywords": [
    "shop",
    "buy",
    "e-commerce",
    "digital",
    "marketplace",
    "web"
  ],
```

```
  "author": "Abhik Gupta",
  "license": "ISC",
  "dependencies": {
    "cookie-parser": "^1.4.6",
    "cors": "^2.8.5",
    "dotenv": "^16.4.5",
    "express": "^4.19.2",
    "express-session": "^1.18.0",
    "jsonwebtoken": "^9.0.2",
    "mongoose": "^8.3.2",
    "nodemailer": "^6.9.13",
    "nodemon": "^3.1.0",
    "passport": "^0.7.0",
    "passport-jwt": "^4.0.1",
    "passport-local": "^1.0.0",
    "path": "^0.12.7",
    "stripe": "^15.4.0"
  }
}
```

.gitignore

```
/node_modules
.env
```

index.js

```
const express = require('express');
const server = express();
const mongoose = require('mongoose');
const cors = require('cors');

require('dotenv').config();
const path = require('path');

var session = require('express-session');
```

```
var passport = require('passport');
const LocalStrategy = require('passport-local').Strategy;
const crypto = require('crypto');
const JwtStrategy = require('passport-jwt').Strategy;
const ExtractJwt = require('passport-jwt').ExtractJwt;
const jwt = require('jsonwebtoken');
const cookieParser = require('cookie-parser');

const productRouter = require('./routes/Products');
const brandRouter = require('./routes/Brands');
const categoryRoutes = require('./routes/Categories');
const userRoutes = require('./routes/Users');
const authRoutes = require('./routes/Auth');
const cartRoutes = require('./routes/Cart');
const orderRoutes = require('./routes/Orders');
const queryRoutes = require('./routes/Queries');
const { User } = require('./model/User');
const { isAuth, sanitizeUser, cookieExtractor } =
require('./services/common');
// const { Order } = require('./model/Order');


// WEBHOOK FOR STRIPE PAYMENTS
// const endpointSecret = process.env.WEBHOOK_ENDPOINT_SECRET;
// server.post('/webhook', express.raw({ type: 'application/json' }), async
(request, response) => {
//     const sig = request.headers['stripe-signature'];

//     let event;

//     try {
//         event = stripe.webhooks.constructEvent(request.body, sig,
endpointSecret);
//     } catch (err) {
//         response.status(400).send(`Webhook Error: ${err.message}`);
//         return;
//     }

//     switch (event.type) {
//         case 'payment_intent.succeeded':
//             const paymentIntentSucceeded = event.data.object;
```

```
//              console.log({paymentIntentSucceeded});
//              const order = await
Order.findById(paymentIntentSucceeded.metadata.orderId);
//              order.paymentStatus = 'received';
//              await order.save();

//              break;
//          default:
//              console.log(`Unhandled event type ${event.type}`);
//      }

//      response.send();
// });


// JWT Token Authentication
const opts = {}
opts.jwtFromRequest = cookieExtractor;
opts.secretOrKey = process.env.JWT_SECRET_KEY;


// Using express session middleware
server.use(session({
    secret: process.env.SESSION_SECRET,
    resave: false,
    saveUninitialized: false
}));
server.use(passport.authenticate('session'));


// Initializing passportJS
// server.use(passport.initialize());
// server.use(passport.session());


// Passport middleware for LocalStrategy
passport.use(
    'local',
    new LocalStrategy({
        usernameField: 'email',
        passwordField: 'password'
```

```
    },
        async function (email, password, done) {
            try {
                const user = await User.findOne({ email: email }).exec();

                if (!user) {
                    console.log("~ No user found with provided email!");
                    return done(null, false, { message: 'User not found'
});
                }

                crypto.pbkdf2(
                    password,
                    user.salt,
                    310000,
                    32,
                    'sha256',
                    async function (err, hashedPassword) {
                        if (!crypto.timingSafeEqual(user.password,
hashedPassword)) {
                            console.log("~ Invalid login credentials!");
                            return done(null, false, { message: 'Invalid
credentials' });
                        }
                        else {
                            const token = jwt.sign(sanitizeUser(user),
process.env.JWT_SECRET_KEY);
                            console.log("~ Logged in a user!");
                            console.log("~ User token:", token);
                            return done(null, { id: user.id, role:
user.role, token: token });
                        }
                    }
                );
            }
            catch (err) {
                return done(err);
            }
        }
    )
);
```

```
// Passport middleware for JWT
passport.use(
    'jwt',
    new JwtStrategy(opts, async function (jwt_payload, done) {

        try {
            const user = await User.findById(jwt_payload.id);
            if (user) {
                return done(null, sanitizeUser(user));
            } else {
                return done(null, false);
            }
        }
        catch (err) {
            return done(err, false);
        }

    })
);


// Serialized and Deserialized User
passport.serializeUser(function (user, cb) {
    console.log('SERIALIZE', user);
    process.nextTick(function () {
        return cb(null, { id: user.id, role: user.role });
    });
});
passport.deserializeUser(function (user, cb) {
    console.log('DESERIALIZE', user);
    process.nextTick(function () {
        return cb(null, user);
    });
});


// Express Middlewares
server.use(express.static(path.resolve(__dirname, 'build')));
server.use(express.json());
```

```
server.use(cookieParser());
server.use(cors({
    exposedHeaders: ['X-Total-Count']
}));


// Routes
server.use('/products', isAuth(), productRouter.routes);
server.use('/brands', isAuth(), brandRouter.routes);
server.use('/categories', isAuth(), categoryRoutes.routes);
server.use('/users', isAuth(), userRoutes.routes);
server.use('/auth', authRoutes.routes);
server.use('/cart', isAuth(), cartRoutes.routes);
server.use('/orders', isAuth(), orderRoutes.routes);
server.use('/queries', isAuth(), queryRoutes.routes);
server.get('*', (req, res) => res.sendFile(path.resolve('build',
'index.html')));

// STRIPE PAYMENTS
// const stripe = require("stripe")(process.env.STRIPE_SERVER_KEY);
// server.post("/create-payment-intent", async (req, res) => {
//     const { totalAmount, orderId } = req.body;

//     const paymentIntent = await stripe.paymentIntents.create({
//         amount: totalAmount * 100,
//         currency: "inr",
//         automatic_payment_methods: {
//             enabled: true,
//         },
//         metadata: {
//             orderId
//         },
//         // payment_method: 'pm_card_visa'
//     });

//     res.send({
//         clientSecret: paymentIntent.client_secret,
//     });
// });

// STRIPE PAYMENT METHOD
```

```
// const paymentMethodDomain = await stripe.paymentMethodDomains.create(
//     {
//         domain_name: 'https://vendr-deployment.vercel.app',
//     },
//     {
//         stripeAccount: '{{CONNECTED_ACCOUNT_ID}}',
//     }
// );

// DB CONNECT
async function dbConnect() {
    await mongoose.connect(process.env.MONGO_DB_URL);
    console.log(`-- DATABASE CONNECTED SUCCESFULLY  --`);
}
dbConnect().catch(err => console.log(err));


// Default route
server.get('/', (req, res) => {
    res.json({ status: 'API IS WORKING :)' });
});


// Listening at a port
server.listen(process.env.PORT, () => {
    console.log(`-- SERVER STARTED AT ${process.env.PORT} PORT SUCCESSFULLY
--`);
});
```

vercel.json

```
{
    "builds": [
        {
            "src": "index.js",
            "use": "@vercel/node"
        },
        {
            "src": "build/**",
```

```
            "use": "@vercel/static"
        }
    ],
    "rewrites": [
        {
            "source": "/(.*)",
            "destination": "index.js"
        }
    ]
}
```

```
    Directory: C:\~VENDR\server\controller


Mode                LastWriteTime         Length Name
----                -------------         ------ ----
-a----        17-05-2024     18:58           4315 Auth.js
-a----        26-04-2024     00:38            642 Brand.js
-a----        30-04-2024     21:43           1579 Cart.js
-a----        26-04-2024     00:39            682 Category.js
-a----        16-05-2024     15:32           2982 Order.js
-a----        16-05-2024     15:32           2692 Product.js
-a----        16-05-2024     15:11            646 Query.js
-a----        01-05-2024     02:17            756 User.js
```

Auth.js

```
const { User } = require("../model/User");
const crypto = require('crypto');
const { sanitizeUser, sendMail } = require("../services/common");
const jwt = require('jsonwebtoken');

exports.createUser = async (req, res) => {
    const SECRET_KEY = 'TOP_SECRET';
```

```
    try {
        const salt = crypto.randomBytes(16);
        crypto.pbkdf2(
            req.body.password,
            salt,
            310000,
            32,
            'sha256',
            async function (err, hashedPassword) {
                const user = new User({ ...req.body, password:
hashedPassword, salt: salt });

                const document = await user.save();

                req.login(sanitizeUser(document), (err) => {
                    if (err) {
                        res.status(400).json(err);
                    }
                    else {
                        const token = jwt.sign(sanitizeUser(document),
SECRET_KEY);
                        console.log("~ Created a user!");
                        res
                            .cookie('jwt', token, { expires: new
Date(Date.now() + 3600000), httpOnly: true })
                            .status(201)
                            .json(token);
                    }
                });
            }
        );
```

```
    }
    catch (err) {
        res.status(400).json(err);
    }
};

exports.loginUser = async (req, res) => {
    const user = req.user;
    res
        .cookie('jwt', req.user.token, { expires: new Date(Date.now() +
3600000), httpOnly: true })
        .status(200)
        .json({ id: user.id, role: user.role });
};

exports.logoutUser = async (req, res) => {
    res
        .cookie('jwt', null, { expires: new Date(Date.now()), httpOnly:
true })
        .sendStatus(200);
};


exports.checkAuth = async (req, res) => {
    if (req.user) {
        res.json(req.user);
    }
    else {
        res.sendStatus(401);
    }
};
```

```
exports.resetPasswordRequest = async (req, res) => {
    const email = req.body.email;
    const user = await User.findOne({ email: email });

    if (user) {
        const token = crypto.randomBytes(48).toString('hex');
        user.resetPasswordToken = token;
        await user.save();

        const resetPageLink = "https://vendr-deployment.vercel.app/reset-
password?token=" + token + "&email=" + email;
        // const resetPageLink = "http://localhost:3000/reset-
password?token=" + token + "&email=" + email;
        const subject = "Reset your Vendr password";
        const text = "Reset your Vendr password";
        const html = `<p>Click <a href="${resetPageLink}">here</a> to reset
your password</p>`;

        if (email) {
            const response = await sendMail({
                to: email,
                subject: subject,
                text: text,
                html: html
            });

            res.json(response);
        }
        else {
            res.sendStatus(400);
```

```
        }
    }
    else {
        res.sendStatus(400);
    }
};

exports.resetPassword = async (req, res) => {
    const { email, token, password } = req.body;
    const user = await User.findOne({ email: email, resetPasswordToken:
token });

    if (user) {
        const salt = crypto.randomBytes(16);
        crypto.pbkdf2(
            req.body.password,
            salt,
            310000,
            32,
            'sha256',
            async function (err, hashedPassword) {
                user.password = hashedPassword;
                user.salt = salt;
                await user.save();

                const subject = "Vendr password reset successful!";
                const text = "Vendr password reset successful!";
                const html = `<p>Vendr password reset successful!`;

                if (email) {
                    const response = await sendMail({
```

```
                        to: email,
                        subject: subject,
                        text: text,
                        html: html
                    });

                    res.json(response);
                }
                else {
                    res.sendStatus(400);
                }
            });
    }
    else {
        res.sendStatus(400);
    }
};
```

Brand.js

```
const { Brand } = require("../model/Brand");

exports.fetchAllBrands = async (req, res) => {
    try {
        let brands = await Brand.find({}).exec();
        console.log("~ Fetched all brands!");
        res.status(200).json(brands);
    }
    catch (err) {
        res.status(400).json(err);
```

```
    }
};

exports.createBrand = (req, res) => {
    const brand = new Brand(req.body);

    brand.save()
        .then(savedDocument => {
            console.log("~ Created a brand!");
            res.status(201).json(savedDocument);
        })
        .catch(err => {
            res.status(400).json(err);
        });
};
```

Cart.js

```
const { Cart } = require("../model/Cart");

exports.fetchCartByUser = async (req, res) => {
    const { id } = req.user;
    try {
        const cartItems = await Cart.find({ user: id
}).populate('user').populate('product');
        console.log("~ Fetched an user's cart!");
        res.status(200).json(cartItems);
    }
    catch (err) {
        res.status(400).json(err);
```

```
    }
}

exports.addToCart = async (req, res) => {
    const { id } = req.user;
    const cart = new Cart({ ...req.body, user: id });

    try {
        const document = await cart.save();
        const savedDocument = await document.populate('product');
        console.log("~ Added an item to cart!");
        res.status(201).json(savedDocument);
    }
    catch (err) {
        res.status(400).json(err);
    }
}

exports.deleteFromCart = async (req, res) => {
    const cartId = req.params.id;

    try {
        const removedCartItem = await
Cart.findByIdAndDelete(cartId).exec();
        console.log("~ Removed an item from cart!");
        res.status(200).json(removedCartItem);
    }
    catch (err) {
        res.status(400).json(err);
    }
}
```

```
exports.updateCart = async (req, res) => {
    const cartId = req.params.id;

    try {
        const cart = await Cart.findByIdAndUpdate(cartId, req.body, { new:
true }).exec();
        const savedDocument = await cart.populate('product');
        console.log("~ Updated the cart!");
        res.status(200).json(savedDocument);
    }
    catch (err) {
        res.status(400).json(err);
    }
}
```

Category.js

```
const { Category } = require("../model/Category");

exports.fetchAllCategories = async (req, res) => {
    try {
        let categories = await Category.find({}).exec();
        console.log("~ Fetched all categories!");
        res.status(200).json(categories);
    }
    catch (err) {
        res.status(400).json(err);
    }
};
```

```
exports.createCategory = (req, res) => {
    const category = new Category(req.body);

    category.save()
        .then(savedDocument => {
            console.log("~ Created a category!");
            res.status(201).json(savedDocument);
        })
        .catch(err => {
            res.status(400).json(err);
        });
};
```

Order.js

```
const { Order } = require("../model/Order");
const { sendMail, invoiceTemplate } = require("../services/common");
const { Product } = require("../model/Product");

exports.fetchOrdersByUser = async (req, res) => {
    const { id } = req.user;
    try {
        const orders = await Order.find({ user: id });
        console.log("~ Fetched an user's orders!");
        res.status(200).json(orders);
    }
    catch (err) {
        res.status(400).json(err);
    }
```

```
}

exports.createOrder = async (req, res) => {
    const order = new Order(req.body);

    if(req.body.selectedPaymentMode === 'card') {
        order.paymentStatus = 'received';
    }
    else if (req.body.selectedPaymentMode === 'cash') {
        order.paymentStatus = 'pending';
    }

    for(let item of order.items) {
        const product = await Product.findOne({_id: item.product.id});
        product.$inc('stock', -1*item.quantity);
        await product.save();
    }

    order.save()
        .then(savedDocument => {
            sendMail({
                to: order.selectedAddress.email,
                subject: "Vendr - Order Invoice",
                text: `Vendr - Order Invoice - #${order.id}`,
                html: invoiceTemplate(order)
            });
            console.log("~ Created an order!");
            res.status(201).json(savedDocument);
        })
        .catch(err => {
            res.status(400).json(err);
```

```
        });
}

exports.deleteOrder = async (req, res) => {
    const orderId = req.params.id;

    try {
        const removedOrder = await Order.findByIdAndDelete(orderId).exec();
        console.log("~ Removed an order!");
        res.status(200).json(removedOrder);
    }
    catch (err) {
        res.status(400).json(err);
    }
}

exports.updateOrder = async (req, res) => {
    const orderId = req.params.id;

    try {
        const order = await Order.findByIdAndUpdate(orderId, req.body, {
new: true }).exec();
        console.log("~ Updated an order!");
        res.status(200).json(order);
    }
    catch (err) {
        res.status(400).json(err);
    }
}

exports.fetchALlOrders = async (req, res) => {
```

```
    let query = Order.find({ deleted: { $ne: true } });
    let totalOrdersQuery = Order.find({ deleted: { $ne: true } });

    if (req.query._sort && req.query._order) {
        query = query.sort({ [req.query._sort]: req.query._order });
    }
    if (req.query._page && req.query._limit) {
        const pageSize = req.query._limit;
        const page = req.query._page;
        query = query.skip(pageSize * (page - 1)).limit(pageSize);
    }

    const totalDocs = await totalOrdersQuery.count().exec();

    try {
        const docs = await query.exec();
        res.set('X-Total-Count', totalDocs);
        console.log("~ Fetched all orders!");
        res.status(200).json(docs);
    }
    catch (err) {
        res.status(400).json(err);
    }
};
```

Product.js

```
const { Product } = require("../model/Product");

exports.createProduct = (req, res) => {
```

```
    const product = new Product(req.body);
    product.discountedPrice = Math.round(product.price * (1 -
product.discountPercentage / 100));

    product.save()
        .then(savedDocument => {
            console.log("~ Created a product!");
            res.status(201).json(savedDocument);
        })
        .catch(err => {
            res.status(400).json(err);
        });
};

exports.fetchAllProducts = async (req, res) => {
    let condition = {};
    if(!req.query.admin) {
        condition.deleted = { $ne: true };
    }

    let query = Product.find(condition);
    let totalProductsQuery = Product.find(condition);

    if (req.query.category) {
        query = query.find({ category: {$in: req.query.category.split(',')}
});
        totalProductsQuery = totalProductsQuery.find({ category: {$in:
req.query.category.split(',')} });
    }
    if (req.query.brand) {
        query = query.find({ brand: {$in: req.query.brand.split(',')} });
```

```
        totalProductsQuery = totalProductsQuery.find({ brand: {$in:
req.query.brand.split(',')} });
    }
    if (req.query._sort && req.query._order) {
        query = query.sort({ [req.query._sort]: req.query._order });
    }
    if (req.query._page && req.query._limit) {
        const pageSize = req.query._limit;
        const page = req.query._page;
        query = query.skip(pageSize * (page - 1)).limit(pageSize);
    }

    const totalDocs = await totalProductsQuery.count().exec();

    try {
        const docs = await query.exec();
        res.set('X-Total-Count', totalDocs);
        console.log("~ Fetched all products!");
        res.status(200).json(docs);
    }
    catch (err) {
        res.status(400).json(err);
    }
};

exports.fetchProductById = async (req, res) => {
    const productId = req.params.id;

    try {
        const product = await Product.findById(productId).exec();
        console.log("~ Fetched a product by ID!");
```

```
        res.status(200).json(product);
    }
    catch (err) {
        res.status(400).json(err);
    }
}

exports.updateProduct = async (req, res) => {
    const productId = req.params.id;

    try {
        const product = await Product.findByIdAndUpdate(productId,
req.body, { new: true }).exec();
        product.discountedPrice = Math.round(product.price * (1 -
product.discountPercentage / 100));
        const updatedProduct = await product.save();
        console.log("~ Updated a product!");
        res.status(200).json(updatedProduct);
    }
    catch (err) {
        res.status(400).json(err);
    }
}
```

Query.js

```
const { Query } = require("../model/Query");


exports.fetchAllQueries = async (req, res) => {
    try {
        let queries = await Query.find({}).exec();
        console.log("~ Fetched all queries!");
        res.status(200).json(queries);
    }
    catch (err) {
        res.status(400).json(err);
    }
};


exports.createQuery = (req, res) => {
    const query = new Query(req.body);


    query.save()
        .then(savedDocument => {
            console.log("~ Created a query!");
            res.status(201).json(savedDocument);
        })
        .catch(err => {
            res.status(400).json(err);
        });
};
```

User.js

```
const { User } = require("../model/User");


exports.fetchUserById = async (req, res) => {
    const { id } = req.user;


    try {
        const user = await User.findById(id).exec();
        console.log("~ Fetched a user by ID!");
        res.status(200).json({id: user.id, addresses: user.addresses,
email: user.email, role: user.role});
    }
    catch (err) {
        res.status(400).json(err);
    }
}


exports.updateUser = async (req, res) => {
    const userId = req.params.id;


    try {
        const user = await User.findByIdAndUpdate(userId, req.body, { new:
true }).exec();
        console.log("~ Updated a user!");
        res.status(200).json(user);
    }
    catch (err) {
        res.status(400).json(err);
    }
}
```

```
    Directory: C:\~VENDR\server\model


Mode                LastWriteTime         Length Name
----                -------------         ------ ----
-a----        26-04-2024     17:07            635 Brand.js
-a----        12-05-2024     15:06            828 Cart.js
-a----        26-04-2024     17:07            665 Category.js
-a----        11-05-2024     16:58           1315 Order.js
-a----        13-05-2024     00:57           2025 Product.js
-a----        16-05-2024     14:49            646 Query.js
-a----        11-05-2024     16:58            902 User.js
```

Brand.js

```
const mongoose = require('mongoose');
const { Schema } = mongoose;

const brandSchema = new Schema({
    value: { type: String, required: [true, 'Please provide value of
brand'], unique: true },
    label: { type: String, required: [true, 'Please provide label of
brand'], unique: true },
});

const virtual = brandSchema.virtual('id');
virtual.get(function () {
    return this._id;
});
brandSchema.set('toJSON', {
    virtuals: true,
    versionKey: false,
    transform: function (doc, ret) {
        delete ret._id;
```

```
    }
})

const BrandModel = mongoose.model('Brand', brandSchema);
exports.Brand = BrandModel;
```

Category.js

```
const mongoose = require('mongoose');
const { Schema } = mongoose;

const categorySchema = new Schema({
    value: { type: String, required: [true, 'Please provide value of
category'], unique: true },
    label: { type: String, required: [true, 'Please provide label of
category'], unique: true },
});

const virtual = categorySchema.virtual('id');
virtual.get(function () {
    return this._id;
});
categorySchema.set('toJSON', {
    virtuals: true,
    versionKey: false,
    transform: function (doc, ret) {
        delete ret._id;
    }
})
```

```
const CategoryModel = mongoose.model('Category', categorySchema);
exports.Category = CategoryModel;
```

Cart.js

```
const mongoose = require('mongoose');
const { Schema } = mongoose;

const cartSchema = new Schema({
    product: { type: Schema.Types.ObjectId, ref: 'Product', required:
[true, 'Please provide a product ID'] },
    user: { type: Schema.Types.ObjectId, ref: 'User', required: [true,
'Please provide an user ID'] },
    quantity: { type: Number, required: [true, 'Please provide quantity of
the product'] },
    color: { type: Schema.Types.Mixed },
    size: { type: Schema.Types.Mixed }
});

const virtual = cartSchema.virtual('id');
virtual.get(function () {
    return this._id;
});
cartSchema.set('toJSON', {
    virtuals: true,
    versionKey: false,
    transform: function (doc, ret) {
        delete ret._id;
    }
})
```

```
const CartModel = mongoose.model('Cart', cartSchema);
exports.Cart = CartModel;
```

Order.js

```
const mongoose = require('mongoose');
const { Schema } = mongoose;

const orderSchema = new Schema({
    items: { type: [Schema.Types.Mixed], required: [true, 'Please provide
items for the order'] },
    totalAmount: { type: Number, required: [true, 'Please provide total
amount for the order'] },
    totalItems: { type: Number, required: [true, 'Please provide total
number of items for the order'] },
    user: { type: Schema.Types.ObjectId, ref: 'User', required: [true,
'Provide a user ID'] },
    selectedPaymentMode: { type: String, required: [true, 'Please provide a
payment mode for the order'] },
    selectedAddress: { type: Schema.Types.Mixed, required: [true, 'Please
provide an address for the order'] },
    status: { type: String, required: [true, 'Please provide a status for
the order'], default: 'pending' },
    paymentStatus: { type: String, required: [true, 'Please provide a
status for the order'], default: 'pending' }
}, {timestamps: true});

const virtual = orderSchema.virtual('id');
virtual.get(function () {
```

```
    return this._id;
});
orderSchema.set('toJSON', {
    virtuals: true,
    versionKey: false,
    transform: function (doc, ret) {
        delete ret._id;
    }
})

const OrderModel = mongoose.model('Order', orderSchema);
exports.Order = OrderModel;
```

Product.js

```
const mongoose = require('mongoose');
const { Schema } = mongoose;

const productSchema = new Schema({
    title: { type: String, required: [true, 'Please provide a title for the
product'], unique: true },
    brand: { type: String, required: [true, 'Please provide a brand for the
product'] },
    category: { type: String, required: [true, 'Please provide a category
for the product'] },
    description: { type: String, required: [true, 'Please provide a
description for the product'] },
    price: { type: Number, min: [0, 'Price of product cannot be less than
0'], required: [true, 'Please provide a price for the product'], max:
[100000, 'Price of product cannot be greater than 100,000'] },
```

```
    discountPercentage: { type: Number, min: [0, 'Discount percentage of
product cannot be less than 0'], max: [90, 'Discount percentage of product
cannot be gearter than 90'] },
    discountedPrice: { type: Number },
    rating: { type: Number, min: [0, 'Rating of product cannot be less than
0'], max: [5, 'Rating of product cannot be greater than 5'], default: 0 },
    stock: { type: Number, min: [0, 'Stock of product cannot be less than
0'], default: 0 },
    thumbnail: { type: String, required: [true, 'Provide a thumbnail of the
product'] },
    images: [String],
    colors: { type: [Schema.Types.Mixed] },
    sizes: { type: [Schema.Types.Mixed] },
    highlights: { type: [String] },
    deleted: { type: Boolean, default: false }
});

const virtualId = productSchema.virtual('id');
virtualId.get(function () {
    return this._id;
});
productSchema.set('toJSON', {
    virtuals: true,
    versionKey: false,
    transform: function (doc, ret) {
        delete ret._id;
    }
})

const ProductModel = mongoose.model('Product', productSchema);
exports.Product = ProductModel;
```

Query.js

```
const mongoose = require('mongoose');
const { Schema } = mongoose;

const querySchema = new Schema({
    userEmail: { type: String, required: [true, "Please provide user's
email address"] },
    query: { type: String, required: [true, 'Please provide a query from
the user'] },
}, { timestamps: true });

const virtual = querySchema.virtual('id');
virtual.get(function () {
    return this._id;
});
querySchema.set('toJSON', {
    virtuals: true,
    versionKey: false,
    transform: function (doc, ret) {
        delete ret._id;
    }
})
const QueryModel = mongoose.model('Query', querySchema);
exports.Query = QueryModel;
```

User.js

```
const mongoose = require('mongoose');
const { Schema } = mongoose;
```

```
const userSchema = new Schema({
    email: { type: String, required: [true, 'Please provide email of the
user'], unique: true },
    password: { type: Buffer, required: [true, 'Please provide brand of the
user'] },
    role: { type: String, required: [true, 'Please provide role of the
user'], default: 'user' },
    addresses: { type: [Schema.Types.Mixed] },
    name: { type: String },
    salt: { type: Buffer },
    resetPasswordToken: { type: String, default: '' }
}, {timestamps: true});

const virtual = userSchema.virtual('id');
virtual.get(function () {
    return this._id;
});
userSchema.set('toJSON', {
    virtuals: true,
    versionKey: false,
    transform: function (doc, ret) {
        delete ret._id;
    }
})

const UserModel = mongoose.model('User', userSchema);
exports.User = UserModel;
```

```
    Directory: C:\~VENDR\server\routes


Mode                LastWriteTime         Length Name
----                -------------         ------ ----
-a----        11-05-2024     16:02            572 Auth.js
-a----        25-04-2024     17:17            243 Brands.js
-a----        26-04-2024     19:38            336 Cart.js
-a----        25-04-2024     17:17            260 Categories.js
-a----        01-05-2024     02:29            392 Orders.js
-a----        26-04-2024     00:54            357 Products.js
-a----        15-05-2024     20:56            245 Queries.js
-a----        30-04-2024     21:36            245 Users.js
```

Auth.js

```
const express = require('express');
const { createUser, loginUser, checkAuth, resetPasswordRequest,
resetPassword, logoutUser } = require('../controller/Auth');
const passport = require('passport');

const router = express.Router();

router
    .post('/signup', createUser)
    .post('/login', passport.authenticate('local'), loginUser)
    .get('/check', passport.authenticate('jwt'), checkAuth)
    .get('/logout', logoutUser)
    .post('/reset-password-request', resetPasswordRequest)
    .post('/reset-password', resetPassword);

exports.routes = router;
```

Brands.js

```
const express = require('express');
const { fetchAllBrands, createBrand } = require('../controller/Brand');

const router = express.Router();

router
    .get('/', fetchAllBrands)
    .post('/', createBrand);

exports.routes = router;
```

Cart.js

```
const express = require('express');
const { addToCart, fetchCartByUser, deleteFromCart, updateCart } =
require('../controller/Cart');

const router = express.Router();

router
    .post('/', addToCart)
    .get('/', fetchCartByUser)
    .delete('/:id', deleteFromCart)
    .patch('/:id', updateCart)

exports.routes = router;
```

Categories.js

```
const express = require('express');
const { fetchAllCategories, createCategory } =
require('../controller/Category');

const router = express.Router();

router
    .get('/', fetchAllCategories)
    .post('/', createCategory);

exports.routes = router;
```

Orders.js

```
const express = require('express');
const { createOrder, fetchOrdersByUser, deleteOrder, updateOrder,
fetchALlOrders } = require('../controller/Order');

const router = express.Router();

router
    .post('/', createOrder)
    .get('/own', fetchOrdersByUser)
    .get('/', fetchALlOrders)
    .delete('/:id', deleteOrder)
    .patch('/:id', updateOrder);

exports.routes = router;
```

Products.js

```
const express = require('express');
const { createProduct, fetchAllProducts, fetchProductById, updateProduct }
= require('../controller/Product');

const router = express.Router();

router
    .post('/', createProduct)
    .get('/', fetchAllProducts)
    .get('/:id', fetchProductById)
    .patch('/:id', updateProduct);

exports.routes = router;
```

Queries.js

```
const express = require('express');
const { fetchAllQueries, createQuery } = require('../controller/Query');

const router = express.Router();

router
    .get('/', fetchAllQueries)
    .post('/', createQuery);

exports.routes = router;
```

Users.js

```
const express = require('express');
const { fetchUserById, updateUser } = require('../controller/User');

const router = express.Router();

router
    .get('/own', fetchUserById)
    .patch('/:id', updateUser);

exports.routes = router;
```

```
    Directory: C:\~VENDR\server\services


Mode                LastWriteTime         Length Name
----                -------------         ------ ----
-a----        16-05-2024     15:30          12683 common.js
```

common.js

```
const passport = require("passport");
const nodemailer = require("nodemailer");
const crypto = require('crypto');

exports.isAuth = (req, res, done) => {
    return passport.authenticate('jwt');
}
```

```
exports.sanitizeUser = (user) => {
    return {
        id: user.id,
        role: user.role
    }
}

exports.cookieExtractor = function (req) {
    var token = null;
    if (req && req.cookies) {
        token = req.cookies['jwt'];
    }
    return token;
};

// EMAIL SYSTEM
const transporter = nodemailer.createTransport({
    host: "smtp.gmail.com",
    port: 587,
    secure: false, // Use `true` for port 465, `false` for all other ports
    auth: {
        user: "abhikinreallife@gmail.com",
        pass: process.env.MAIL_PASSWORD,
    },
});

exports.sendMail = async function ({ to, subject, text, html }) {
    const info = await transporter.sendMail({
        from: '"Vendr" <abhikinreallife@gmail.com>',
        to: to,
        subject: subject,
```

```
        text: text,
        html: html
    });

    return info;
}



exports.invoiceTemplate = function (order) {
    return (
        `
        <!DOCTYPE html>
        <html>

        <head>
            <meta charset="utf-8">
            <meta http-equiv="x-ua-compatible" content="ie=edge">
            <title>Email Receipt</title>
            <meta name="viewport" content="width=device-width, initial-
scale=1">
            <style type="text/css">
                @media screen {
                    @font-face {
                        font-family: 'Source Sans Pro';
                        font-style: normal;
                        font-weight: 400;
                        src: local('Source Sans Pro Regular'),
local('SourceSansPro-Regular'),
url(https://fonts.gstatic.com/s/sourcesanspro/v10/ODelI1aHBYDBqgeIAH2zlBM0Y
zuT7MdOe03otPbuUS0.woff) format('woff');
                    }
```

```
                    @font-face {
                        font-family: 'Source Sans Pro';
                        font-style: normal;
                        font-weight: 700;
                        src: local('Source Sans Pro Bold'),
local('SourceSansPro-Bold'),
url(https://fonts.gstatic.com/s/sourcesanspro/v10/toadOcfmlt9b38dHJxOBGFkQc
6VGVFSmCnC_l7QZG60.woff) format('woff');
                    }
                }

                body,
                table,
                td,
                a {
                    -ms-text-size-adjust: 100%;
                    -webkit-text-size-adjust: 100%;
                }

                table,
                td {
                    mso-table-rspace: 0pt;
                    mso-table-lspace: 0pt;
                }

                img {
                    -ms-interpolation-mode: bicubic;
                }

                a[x-apple-data-detectors] {
```

```
    font-family: inherit !important;
    font-size: inherit !important;
    font-weight: inherit !important;
    line-height: inherit !important;
    color: inherit !important;
    text-decoration: none !important;
}

div[style*="margin: 16px 0;"] {
    margin: 0 !important;
}

body {
    width: 100% !important;
    height: 100% !important;
    padding: 0 !important;
    margin: 0 !important;
}

table {
    border-collapse: collapse !important;
}

a {
    color: #1a82e2;
}

img {
    height: auto;
    line-height: 100%;
    text-decoration: none;
```

```
                    border: 0;
                    outline: none;
                }
            </style>
        </head>

        <body style="background-color: #D2C7BA;">
            <div class="preheader" style="display: none; max-width: 0; max-height: 0; overflow: hidden; font-size: 1px; line-height: 1px; color: #fff; opacity: 0;">
                A preheader is the short summary text that follows the subject line when an email is viewed in the inbox.
            </div>
            <table border="0" cellpadding="0" cellspacing="0" width="100%">
                <tr>
                    <td align="center" bgcolor="#D2C7BA">
                        <table border="0" cellpadding="0" cellspacing="0" width="100%" style="max-width: 600px;">
                            <tr>
                                <td align="center" valign="top" style="padding: 36px 24px;">
                                    <a href="https://vendr-deployment.vercel.app/" target="_blank" style="display: inline-block;">
                                        <img src="https://raw.githubusercontent.com/abhik2207/Vendr-Development/main/Possible-Favicons/Final/favicon.png" alt="Logo" border="0" width="48" style="display: block; width: 48px; max-width: 48px; min-width: 48px;">
                                    </a>
                                </td>
                            </tr>
```

```
                        </table>
                    </td>
                </tr>
                <tr>
                    <td align="center" bgcolor="#D2C7BA">
                        <table border="0" cellpadding="0" cellspacing="0"
width="100%" style="max-width: 600px;">
                            <tr>
                                <td align="left" bgcolor="#ffffff"
style="padding: 36px 24px 0; font-family: 'Source Sans Pro', Helvetica,
Arial, sans-serif; border-top: 3px solid #d4dadf;">
                                    <h1 style="margin: 0; font-size: 32px;
font-weight: 700; letter-spacing: -1px; line-height: 48px;">Thank you for
your order!</h1>
                                </td>
                            </tr>
                        </table>
                    </td>
                </tr>
                <tr>
                    <td align="center" bgcolor="#D2C7BA">
                        <table border="0" cellpadding="0" cellspacing="0"
width="100%" style="max-width: 600px;">
                            <tr>
                                <td align="left" bgcolor="#ffffff"
style="padding: 24px; font-family: 'Source Sans Pro', Helvetica, Arial,
sans-serif; font-size: 16px; line-height: 24px;">
                                    <p style="margin: 0;">Here is a summary
of your recent order from Vendr :)</p>
                                    <p style="margin: 0;">Order ID :
${order.id}</p>
```

```
                                </td>
                            </tr>
                            <tr>
                                <td align="left" bgcolor="#ffffff"
                                    style="padding: 24px; font-family:
'Source Sans Pro', Helvetica, Arial, sans-serif; font-size: 16px; line-
height: 24px;">
                                    <table border="0" cellpadding="0"
cellspacing="0" width="100%">
                                        <tr>
                                            <td align="left"
bgcolor="#D2C7BA" width="60%" style="padding: 12px;font-family: 'Source
Sans Pro', Helvetica, Arial, sans-serif; font-size: 16px; line-height:
24px;"><strong>Items</strong></td>
                                            <td align="left"
bgcolor="#D2C7BA" width="20%" style="padding: 12px;font-family: 'Source
Sans Pro', Helvetica, Arial, sans-serif; font-size: 16px; line-height:
24px;"><strong>Quantity</strong></td>
                                            <td align="left"
bgcolor="#D2C7BA" width="20%" style="padding: 12px;font-family: 'Source
Sans Pro', Helvetica, Arial, sans-serif; font-size: 16px; line-height:
24px;"><strong>Price</strong></td>
                                        </tr>

                                        ${order.items.map(item => `<tr>
                                                <td align="left"
width="60%" style="padding: 6px 12px;font-family: 'Source Sans Pro',
Helvetica, Arial, sans-serif; font-size: 16px; line-height:
24px;">${item.product.title} (${item.product.brand[0].toUpperCase() +
item.product.brand.slice(1)})</td>
```

```
                                                <td align="left"
width="20%" style="padding: 6px 12px;font-family: 'Source Sans Pro',
Helvetica, Arial, sans-serif; font-size: 16px; line-height:
24px;">${item.quantity}</td>
                                                <td align="left"
width="20%" style="padding: 6px 12px;font-family: 'Source Sans Pro',
Helvetica, Arial, sans-serif; font-size: 16px; line-height:
24px;">$${Math.round(item.product.price * (1-
item.product.discountPercentage/100))}</td>
                                            </tr>`
                                        )}

                                        <tr>
                                            <td align="left" width="60%"
style="padding: 12px; font-family: 'Source Sans Pro', Helvetica, Arial,
sans-serif; font-size: 16px; line-height: 24px; border-top: 2px dashed
#D2C7BA; border-bottom: 2px dashed #D2C7BA;"><strong>Total</strong></td>
                                            <td align="left" width="20%"
style="padding: 12px; font-family: 'Source Sans Pro', Helvetica, Arial,
sans-serif; font-size: 16px; line-height: 24px; border-top: 2px dashed
#D2C7BA; border-bottom: 2px dashed
#D2C7BA;"><strong>${order.totalItems}</td>
                                            <td align="left" width="20%"
style="padding: 12px; font-family: 'Source Sans Pro', Helvetica, Arial,
sans-serif; font-size: 16px; line-height: 24px; border-top: 2px dashed
#D2C7BA; border-bottom: 2px dashed
#D2C7BA;"><strong>$${order.totalAmount}</strong></td>
                                        </tr>
                                    </table>
                                </td>
                            </tr>
```

```
                        </table>
                    </td>
                </tr>
                <tr>
                    <td align="center" bgcolor="#D2C7BA" valign="top"
width="100%">
                        <table align="center" bgcolor="#ffffff" border="0"
cellpadding="0" cellspacing="0" width="100%" style="max-width: 600px;">
                            <tr>
                                <td align="center" valign="top"
style="font-size: 0; border-bottom: 3px solid #d4dadf">
                                    <div style="display: inline-block;
width: 100%; max-width: 50%; min-width: 240px; vertical-align: top;">
                                        <table align="left" border="0"
cellpadding="0" cellspacing="0" width="100%" style="max-width: 300px;">
                                            <tr>
                                                <td align="left"
valign="top"
                                                    style="padding-bottom:
36px; padding-left: 36px; font-family: 'Source Sans Pro', Helvetica, Arial,
sans-serif; font-size: 16px; line-height: 24px;">
                                                    <p><strong>Delivery
Address</strong></p>

<p>${order.selectedAddress.name}</p>
                                                    <p>Phone:
${order.selectedAddress.phone}</p>

<p>${order.selectedAddress.houseNumber}, ${order.selectedAddress.locality},
${order.selectedAddress.city}</p>
```

```
<p>${order.selectedAddress.city} - ${order.selectedAddress.pinCode}</p>
                                                    </td>
                                                </tr>
                                            </table>
                                        </div>
                                        <div style="display: inline-block;
width: 100%; max-width: 50%; min-width: 240px; vertical-align: top;">
                                            <table align="left" border="0"
cellpadding="0" cellspacing="0" width="100%" style="max-width: 300px;">
                                                <tr>
                                                </tr>
                                            </table>
                                        </div>
                                    </td>
                                </tr>
                            </table>
                        </td>
                    </tr>
                </table>
            </body>


            </html>
            `
        )
    }
```

CLIENT

```
    Directory: C:\~VENDR\client


Mode                LastWriteTime         Length Name
----                -------------         ------ ----
d-----        18-05-2024     00:25                build
d-----        18-05-2024     00:26                node_modules
d-----        18-05-2024     00:26                public
d-----        18-05-2024     00:26                src
-a----        26-04-2024     01:30             13 .gitignore
-a----        25-04-2024     04:15         107393 data.json
-a----        10-05-2024     01:39         779378 package-lock.json
-a----        10-05-2024     01:39           1404 package.json
-a----        07-04-2024     16:08           2219 README.md
-a----        10-04-2024     21:54            326 tailwind.config.js
```

package.json

```
{
  "name": "vendr",
  "version": "0.1.0",
  "proxy": "http://localhost:8080",
  "private": true,
  "dependencies": {
    "@headlessui/react": "^1.7.18",
    "@heroicons/react": "^2.1.3",
    "@reduxjs/toolkit": "^1.9.7",
    "@stripe/react-stripe-js": "^2.7.0",
    "@stripe/stripe-js": "^3.3.0",
    "@tailwindcss/aspect-ratio": "^0.4.2",
    "@tailwindcss/forms": "^0.5.7",
    "@testing-library/jest-dom": "^5.17.0",
    "@testing-library/react": "^13.4.0",
    "@testing-library/user-event": "^14.5.2",
```

```
  "json-server": "0.17.0",
  "react": "^18.2.0",
  "react-dom": "^18.2.0",
  "react-hook-form": "^7.51.3",
  "react-loader-spinner": "^6.1.6",
  "react-redux": "^8.1.3",
  "react-router-dom": "^6.22.3",
  "react-scripts": "5.0.1",
  "react-square-web-payments-sdk": "^3.2.1",
  "react-toastify": "^10.0.5",
  "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"
    ]
  },
  "devDependencies": {
    "tailwindcss": "^3.4.3"
  }
}
```

tailwind.config.js

```
/** @type {import('tailwindcss').Config} */
module.exports = {
  content: [
    "./src/**/*.{js,jsx,ts,tsx}"
  ],
  theme: {
    extend: {
      gridTemplateRows: {
        '[auto,auto,1fr]': 'auto auto 1fr',
      }
    }
  },
  plugins: [
    require('@tailwindcss/aspect-ratio'),
    require('@tailwindcss/forms'),
  ],
}
```

.gitignore

```
/node_modules
```

```
    Directory: C:\~VENDR\client\public


Mode                LastWriteTime         Length Name
----                -------------         ------ ----
-a----        25-04-2024     01:16          20068 cart-icon.png
-a----        07-04-2024     16:08           3585 favicon.ico
-a----        25-04-2024     01:15          24793 favicon.png
-a----        25-04-2024     01:21           1768 index.html
-a----        07-04-2024     16:08           4153 logo192.png
-a----        07-04-2024     16:08          12066 logo512.png
-a----        07-04-2024     16:08            492 manifest.json
-a----        25-04-2024     01:16          34138 navbar-icon.png
-a----        07-04-2024     16:08             57 robots.txt
-a----        15-05-2024     00:54          20177 user.png
```

index.html

```
<!DOCTYPE html>
<html class="h-full bg-gray-100" lang="en">
  <head>
    <meta charset="utf-8" />
    <link rel="icon" href="./favicon.png" />
    <meta name="viewport" content="width=device-width, initial-scale=1" />
    <meta name="theme-color" content="#000000" />
    <meta
      name="description"
      content="Web site created using create-react-app"
    />
    <link rel="apple-touch-icon" href="%PUBLIC_URL%/logo192.png" />
```

```
    <link rel="manifest" href="%PUBLIC_URL%/manifest.json" />
    <title>VENDR - Shop Smart & Easy</title>
  </head>
  <body class="h-full">
    <noscript>You need to enable JavaScript to run this app.</noscript>
    <div id="root"></div>
  </body>
</html>
```

```
    Directory: C:\~VENDR\client\src


Mode                LastWriteTime         Length Name
----                -------------         ------ ----
d-----        18-05-2024     00:26                app
d-----        18-05-2024     00:26                features
d-----        18-05-2024     00:26                pages
-a----        09-04-2024     15:39            115 App.css
-a----        16-05-2024     15:19           4611 App.js
-a----        07-04-2024     16:08            372 App.test.js
-a----        07-04-2024     16:16            399 index.css
-a----        25-04-2024     02:54            686 index.js
-a----        07-04-2024     16:08           1122 logo.svg
-a----        10-05-2024     01:38            298 Payment.css
-a----        07-04-2024     16:08            362 reportWebVitals.js
-a----        07-04-2024     16:08            255 setupTests.js
-a----        01-05-2024     15:44           2544 Stripe.css
```

index.js

```
import React from 'react';
import { createRoot } from 'react-dom/client';
import { Provider } from 'react-redux';
import { store } from './app/store';
```

```
import App from './App';
import reportWebVitals from './reportWebVitals';
import './index.css';

const container = document.getElementById('root');
const root = createRoot(container);

root.render(
  <React.StrictMode>
    <Provider store={store}>
      <App />
    </Provider>
  </React.StrictMode>
);
reportWebVitals();
```

App.js

```
import React, { useEffect } from 'react';
import './App.css';

import Homepage from './pages/Homepage';
import LoginPage from './pages/LoginPage';
import SignupPage from './pages/SignupPage';

import { createBrowserRouter, RouterProvider } from "react-router-dom";
import CartPage from './pages/CartPage';
import CheckoutPage from './pages/CheckoutPage';
import ProductDetailPage from './pages/ProductDetailPage';
import Protected from './features/auth/components/Protected';
```

```
import PageNotFound from './pages/PageNotFound';
import { fetchItemsByUserIdAsync } from './features/cart/cartSlice';
import { useDispatch, useSelector } from 'react-redux';
import { checkAuthAsync, selectLoggedInUser, selectUserChecked } from
'./features/auth/authSlice';
import OrderSuccessPage from './pages/OrderSuccessPage';
import UserOrdersPage from './pages/UserOrdersPage';
import UserProfilePage from './pages/UserProfilePage';
import { fetchLoggedInUserAsync } from './features/user/userSlice';
import Logout from './features/auth/components/Logout';
import ForgotPasswordPage from './pages/ForgotPasswordPage';
import ProtectedAdmin from './features/auth/components/ProtectedAdmin';
import AdminHomepage from './pages/AdminHomepage';
import AdminProductDetailPage from './pages/AdminProductDetailPage';
import AdminProductFormPage from './pages/AdminProductFormPage';
import AdminOrdersPage from './pages/AdminOrdersPage';

import { ToastContainer } from 'react-toastify';
import 'react-toastify/dist/ReactToastify.css';
import StripeCheckout from './pages/StripeCheckout';
import PaymentPage from './pages/PaymentPage';
import ResetPasswordPage from './pages/ResetPasswordPage';
import AboutUsPage from './pages/AboutUsPage';
import ContactUsPage from './pages/ContactUsPage';
import AdminQueryPage from './pages/AdminQueryPage';

const router = createBrowserRouter([
  {
    path: "/",
    element: <Protected>
      <Homepage />
```

```
  </Protected>,
},
{
  path: "/admin",
  element: <ProtectedAdmin>
    <AdminHomepage />
  </ProtectedAdmin>,
},
{
  path: "/signup",
  element: <SignupPage />,
},
{
  path: "/login",
  element: <LoginPage />,
},
{
  path: "/about",
  element: <Protected>
    <AboutUsPage />
  </Protected>,
},
{
  path: "/contact",
  element: <Protected>
    <ContactUsPage />
  </Protected>,
},
{
  path: "/cart",
  element: <Protected>
```

```
    <CartPage />
  </Protected>,
},
{
  path: "/checkout",
  element: <Protected>
    <CheckoutPage />
  </Protected>,
},
{
  path: "/product-detail/:id",
  element: <Protected>
    <ProductDetailPage />
  </Protected>,
},
{
  path: "/admin/product-detail/:id",
  element: <ProtectedAdmin>
    <AdminProductDetailPage />
  </ProtectedAdmin>,
},
{
  path: "/admin/product-form",
  element: <ProtectedAdmin>
    <AdminProductFormPage />
  </ProtectedAdmin>,
},
{
  path: "/admin/queries",
  element: <ProtectedAdmin>
    <AdminQueryPage />
```

```
  </ProtectedAdmin>,
},
{
  path: "/admin/product-form/edit/:id",
  element: <ProtectedAdmin>
    <AdminProductFormPage />
  </ProtectedAdmin>,
},
{
  path: "/admin/orders",
  element: <ProtectedAdmin>
    <AdminOrdersPage />
  </ProtectedAdmin>,
},
{
  path: "/order-success/:id",
  element: <Protected>
    <OrderSuccessPage />
  </Protected>
},
{
  path: "/stripe-checkout",
  element: <Protected>
    <StripeCheckout />
  </Protected>
},
{
  path: "/payment-gateway",
  element: <Protected>
    <PaymentPage />
  </Protected>
```

```
  },
  {
    path: "/my-orders",
    element: <Protected>
      <UserOrdersPage />
    </Protected>
  },
  {
    path: "/profile",
    element: <Protected>
      <UserProfilePage />
    </Protected>
  },
  {
    path: "/logout",
    element: <Logout />
  },
  {
    path: "/forgot-password",
    element: <ForgotPasswordPage />
  },
  {
    path: "/reset-password",
    element: <ResetPasswordPage />
  },
  {
    path: "*",
    element: <PageNotFound />
  }
]);
```

```
function App() {
  const dispatch = useDispatch();
  const user = useSelector(selectLoggedInUser);
  const userChecked = useSelector(selectUserChecked);

  useEffect(() => {
    dispatch(checkAuthAsync());
  }, [dispatch]);

  useEffect(() => {
    if (user) {
      dispatch(fetchItemsByUserIdAsync());
      dispatch(fetchLoggedInUserAsync());
    }
  }, [dispatch, user]);

  return (
    <div id="main-body">
      {userChecked &&
        <RouterProvider router={router} />
      }
      <ToastContainer />
    </div>
  );
}

export default App;
```

```
    Directory: C:\~VENDR\client\src\app


Mode                LastWriteTime         Length Name
----                -------------         ------ ----
-a----        28-04-2024     21:43             32 constants.js
-a----        15-05-2024     21:20            598 store.js
```

constants.js

```
export const ITEMS_PER_PAGE = 6;
```

store.js

```
import { configureStore } from '@reduxjs/toolkit';
import productReducer from '../features/product/productSlice';
import authReducer from '../features/auth/authSlice';
import cartReducer from '../features/cart/cartSlice';
import orderReducer from '../features/order/orderSlice';
import userReducer from '../features/user/userSlice';
import queryReducer from '../features/query/querySlice';
export const store = configureStore({
  reducer: {
    product: productReducer,
    auth: authReducer,
    cart: cartReducer,
    order: orderReducer,
    user: userReducer,
    query: queryReducer
  },
});
```

```
    Directory: C:\~VENDR\client\src\features


Mode                LastWriteTime         Length Name
----                -------------         ------ ----
d-----        18-05-2024     00:26                admin
d-----        18-05-2024     00:26                auth
d-----        18-05-2024     00:26                cart
d-----        18-05-2024     00:26                common
d-----        18-05-2024     00:26                counter
d-----        18-05-2024     00:26                navbar
d-----        18-05-2024     00:26                order
d-----        18-05-2024     00:26                product
d-----        18-05-2024     00:26                query
d-----        18-05-2024     00:26                user
```

AdminOrders.js

```
import React, { useEffect, useState } from 'react';
import { useDispatch, useSelector } from 'react-redux';
import { ITEMS_PER_PAGE } from '../../../app/constants';
import { fetchAllOrdersAsync, selectAllOrders, selectTotalOrders,
updateOrderAsync } from '../../order/orderSlice';
import { EyeIcon, PencilIcon, ArrowUpIcon, ArrowDownIcon } from
'@heroicons/react/24/outline';
```

```
import Pagination from '../../common/Pagination';

function AdminOrders() {
    const [page, setPage] = useState(1);
    const dispatch = useDispatch();

    const orders = useSelector(selectAllOrders);
    const totalOrders = useSelector(selectTotalOrders);

    const [sort, setSort] = useState({});

    const [editableOrderId, setEditableOrderId] = useState(-1);

    const handlePage = (page) => {
        setPage(page);
    }

    const handleShow = (order) => {
        console.log(order);
    }

    const handleOrderStatus = (e, order) => {
        const updatedOrder = { ...order, status: e.target.value };
        dispatch(updateOrderAsync(updatedOrder));
        setEditableOrderId(-1);
    }

    const handlePaymentStatus = (e, order) => {
        const updatedOrder = { ...order, paymentStatus: e.target.value };
        dispatch(updateOrderAsync(updatedOrder));
        setEditableOrderId(-1);
```

```
}

const handleEdit = (order) => {
    setEditableOrderId(order.id);
}

const chooseColor = (status) => {
    if (status === 'pending') {
        return 'bg-gray-200 text-gray-600';
    }
    else if (status === 'received') {
        return 'bg-green-200 text-green-600';
    }
    else if (status === 'confirmed') {
        return 'bg-yellow-200 text-yellow-600';
    }
    else if (status === 'dispatched') {
        return 'bg-purple-200 text-purple-600';
    }
    else if (status === 'shipped') {
        return 'bg-blue-200 text-blue-600';
    }
    else if (status === 'delivered') {
        return 'bg-green-200 text-green-600';
    }
    else if (status === 'cancelled') {
        return 'bg-red-200 text-red-600';
    }
    else {
        return 'bg-gray-200 text-gray-600';
    }
```

```
    }

    const handleSort = (sortOption) => {
        const sort = { _sort: sortOption.sort, _order: sortOption.order };
        setSort(sort);
    }

    useEffect(() => {
        const pagination = { _page: page, _limit: ITEMS_PER_PAGE };
        console.log({ pagination });
        dispatch(fetchAllOrdersAsync({ sort, pagination }));
    }, [dispatch, page, sort]);

    return (
        <div>
            <div className="overflow-x-auto mt-2">
                <div className="bg-gray-100 flex items-center justify-
center font-sans overflow-hidden rounded-lg">
                    <div className="w-full">
                        <div className="bg-white shadow-md">
                            <table className="w-full table-auto">
                                <thead>
                                    <tr className="bg-gray-200 text-gray-
600 uppercase text-sm leading-normal">
                                        <th className="py-3 px-3 text-left
cursor-pointer flex items-center" onClick={(e) => handleSort({ sort: 'id',
order: sort?._order === 'asc' ? 'desc' : 'asc' })}>Order ID {
                                            sort._sort === 'id' && (
                                                sort._order === 'desc' ?
                                                    <ArrowDownIcon
className='w-5 h-5 text-green-600 mb-1 ml-1' /> :
```

```
                                                    <ArrowUpIcon
className='w-5 h-5 text-red-600 mb-1 ml-1' />
                                                )}
                                            </th>
                                            <th className="py-3 px-3 text-
left">Items</th>
                                            <th className="py-3 px-3 text-
center cursor-pointer flex items-center" onClick={(e) => handleSort({ sort:
'totalAmount', order: sort?._order === 'asc' ? 'desc' : 'asc' })}>Total
Amount {
                                                sort._sort === 'totalAmount' &&
(
                                                    sort._order === 'desc' ?
                                                    <ArrowDownIcon
className='w-5 h-5 text-green-600 mb-1 ml-1' /> :
                                                    <ArrowUpIcon
className='w-5 h-5 text-red-600 mb-1 ml-1' />
                                                )}
                                            </th>
                                            <th className="py-3 px-3 text-
center">Payment Mode</th>
                                            <th className="py-3 px-3 text-
center">Payment Status</th>
                                            <th className="py-3 px-3 text-
center">Order Time</th>
                                            <th className="py-3 px-3 text-
center">Last Updated</th>
                                            <th className="py-3 px-3 text-
center">Shipping address</th>
                                            <th className="py-3 px-3 text-
center">Order Status</th>
```

```
                                        <th className="py-3 px-3 text-
center">Actions</th>
                                    </tr>
                                </thead>
                                <tbody className="text-gray-600 text-sm
font-light">
                                    {orders.map((order, index) => (
                                        <tr key={index} className="border-b
border-gray-200 hover:bg-gray-100">
                                            <td className="py-3 px-3 text-
left whitespace-nowrap">
                                                <div className="flex items-
center">
                                                    <span className="font-
medium text-md">#{order.id.slice(-4)}</span>
                                                </div>
                                            </td>
                                            <td className="py-3 px-3 text-
left">
                                                {order.items.map((item,
index) => (
                                                    <div key={index}
className="flex items-center py-1">
                                                        <div className="mr-
2">
                                                            <img

className="w-8 h-8 rounded-full"

src={item.product.thumbnail}
```

```
alt={item.product.title}
                                                                    />
                                                                </div>
                                                                <span
className='font-medium'>{item.product.title} -
${item.product.discountedPrice} - ({item.quantity})</span>
                                                            </div>
                                                        ))}
                                                    </td>
                                                    <td className="py-3 px-3 text-
center">
                                                        <div className="flex items-
center justify-center">
                                                            <p className='text-md
text-green-500 font-bold'>${order.totalAmount}</p>
                                                        </div>
                                                    </td>
                                                    <td className="py-3 px-3 text-
center">
                                                        <div className="flex items-
center justify-center">
                                                            <p className='text-md
text-blue-500 font-bold'>{order.selectedPaymentMode[0].toUpperCase() +
order.selectedPaymentMode.slice(1)}</p>
                                                        </div>
                                                    </td>
                                                    <td className="py-3 px-3 text-
center">
                                                        {order.id ===
editableOrderId ?
```

```
                                                        (
                                                            <select
name="paymentStatus" id="paymentStatus" onChange={e =>
handlePaymentStatus(e, order)} value={order.paymentStatus}>
                                                                <option
value="pending">Pending</option>
                                                                <option
value="received">Received</option>
                                                            </select>
                                                        ) :
                                                        (
                                                            <span
className={`${chooseColor(order.paymentStatus)} font-medium py-1 px-3
rounded-full text-xs`}>

{order.paymentStatus[0].toUpperCase() + order.paymentStatus.slice(1)}
                                                            </span>
                                                        )}
                                                </td>
                                                <td className="py-3 px-3 text-
center">
                                                    <div className="flex items-
center justify-center">
                                                        <p className='text-md
text-green-500 font-bold'>{new Date(order.createdAt).toLocaleString()}</p>
                                                    </div>
                                                </td>
                                                <td className="py-3 px-3 text-
center">
                                                    <div className="flex items-
center justify-center">
```

```
                                                    <p className='text-md
text-blue-500 font-bold'>{new Date(order.updatedAt).toLocaleString()}</p>
                                                </div>
                                            </td>
                                            <td className="py-3 px-3 text-
center">
                                                <div className="flex items-
center justify-center flex-col">
                                                    <p className='text-sm
font-bold'>{order.selectedAddress.name}</p>
                                                    <p className='text-sm
font-normal'>{order.selectedAddress.houseNumber} -
{order.selectedAddress.locality}</p>
                                                    <p className='text-sm
font-normal'>{order.selectedAddress.city} - {order.selectedAddress.state} -
{order.selectedAddress.pinCode}</p>
                                                </div>
                                            </td>
                                            <td className="py-3 px-3 text-
center">
                                                {order.id ===
editableOrderId ?
                                                    (
                                                        <select
name="status" id="status" onChange={e => handleOrderStatus(e, order)}
value={order.status}>
                                                            <option
value="pending">Pending</option>
                                                            <option
value="confirmed">Confirmed</option>
```

```
                                                                <option
value="dispatched">Dispatched</option>
                                                                <option
value="shipped">Shipped</option>
                                                                <option
value="delivered">Delivered</option>
                                                                <option
value="cancelled">Cancelled</option>
                                                            </select>
                                                        ) :
                                                        (
                                                            <span
className={`${chooseColor(order.status)} font-medium py-1 px-3 rounded-full
text-xs`}>

{order.status[0].toUpperCase() + order.status.slice(1)}
                                                            </span>
                                                        )}
                                                </td>
                                                <td className="py-3 px-3 text-
center">
                                                    <div className="flex item-
center justify-center">
                                                        <div className="w-8 mr-
2 transform hover:text-purple-500 hover:scale-120">
                                                            <button onClick={e
=> handleShow(order)} className='cursor-pointer'>
                                                                <EyeIcon
className='w-6 h-6' />
                                                            </button>
                                                        </div>
```

```
                                                        <div className="w-8 mr-
2 transform hover:text-purple-500 hover:scale-120">
                                                            <button onClick={e
=> handleEdit(order)} className='cursor-pointer'>
                                                                <PencilIcon
className='w-6 h-6' />
                                                            </button>
                                                        </div>
                                                    </div>
                                                </td>
                                            </tr>
                                        ))}
                                    </tbody>
                                </table>
                            </div>
                        </div>
                    </div>

                    <Pagination page={page} setPage={setPage}
handlePage={handlePage} totalItems={totalOrders} />
                </div>
            </div>
        )
}

export default AdminOrders;
```

AdminProductDetail.js

```
import React, { useEffect, useState } from 'react';
import { StarIcon } from '@heroicons/react/20/solid';
import { RadioGroup } from '@headlessui/react';
import { useDispatch, useSelector } from 'react-redux';
import { fetchProductByIdAsync, selectProductById, selectProductStatus }
from '../../product/productSlice';
import { useParams } from 'react-router-dom';
import { addToCartAsync, selectItems } from '../../cart/cartSlice';
import { toast } from 'react-toastify';
import { MutatingDots } from 'react-loader-spinner';

function classNames(...classes) {
    return classes.filter(Boolean).join(' ');
}

export default function AdminProductDetail() {
    const [selectedColor, setSelectedColor] = useState({});
    const [selectedSize, setSelectedSize] = useState({});

    const product = useSelector(selectProductById);
    const dispatch = useDispatch();

    const params = useParams();
    useEffect(() => {
        dispatch(fetchProductByIdAsync(params.id));
    }, [dispatch, params.id]);

    const loadingStatus = useSelector(selectProductStatus);
    const cartItems = useSelector(selectItems);
```

```
    const handleCart = (e) => {
        e.preventDefault();
        if (cartItems.findIndex((item) => item.product.id === product.id) <
0) {
            const newItem = { product: product.id, quantity: 1 };
            if (selectedColor) {
                newItem.color = selectedColor;
            }
            if (selectedSize) {
                newItem.size = selectedSize;
            }

            dispatch(addToCartAsync(newItem));
        }
        else {
            toast.info('Item already exists in cart!', {
                position: "bottom-right",
                autoClose: 2000,
                hideProgressBar: false,
                closeOnClick: true,
                pauseOnHover: true,
                draggable: true,
                progress: undefined,
                theme: "colored"
            });
        }
    }

    return (
        <div>
```

```
            <div className="bg-white">


                {loadingStatus === 'loading' &&
                    <div className='w-full flex justify-center items-
center'>
                        <MutatingDots
                            visible={true}
                            height="100"
                            width="100"
                            color="#4464f2"
                            secondaryColor="#4464f2"
                            radius="12.5"
                            ariaLabel="mutating-dots-loading"
                            wrapperStyle={{}}
                            wrapperClass=""
                        />'
                    </div>}


                {product &&
                    (<div className="pt-6">
                        <nav aria-label="Breadcrumb">
                            <ol className="mx-auto flex max-w-2xl items-
center space-x-2 px-4 sm:px-6 lg:max-w-7xl lg:px-8">
                                {product.breadcrumbs &&
product.breadcrumbs.map((breadcrumb) => (
                                    <li key={breadcrumb.id}>
                                        <div className="flex items-center">
                                            <a href={breadcrumb.href}
className="mr-2 text-sm font-medium text-gray-900">
                                                {breadcrumb.name}
                                            </a>
```

```
                                                <svg
                                                    width={16}
                                                    height={20}
                                                    viewBox="0 0 16 20"
                                                    fill="currentColor"
                                                    aria-hidden="true"
                                                    className="h-5 w-4 text-
gray-300"
                                                >
                                                    <path d="M5.697 4.34L8.98
16.532h1.327L7.025 4.341H5.697z" />
                                                </svg>
                                            </div>
                                        </li>
                                    ))}
                                    <li className="text-sm">
                                        <a href={product.href} aria-
current="page" className="font-medium text-gray-500 hover:text-gray-600">
                                            {product.brand}
                                        </a>
                                    </li>
                                </ol>
                            </nav>

                            <div className="mx-auto mt-6 max-w-2xl sm:px-6
lg:grid lg:max-w-7xl lg:grid-cols-3 lg:gap-x-8 lg:px-8">
                                <div className="aspect-h-4 aspect-w-3 hidden
overflow-hidden rounded-lg lg:block">
                                    <img
                                        src={product.images[0]}
                                        alt={product.title}
```

```
                                    className="h-full w-full object-cover
object-center"
                                />
                            </div>
                            <div className="hidden lg:grid lg:grid-cols-1
lg:gap-y-8">
                                <div className="aspect-h-2 aspect-w-3
overflow-hidden rounded-lg">
                                    <img
                                        src={product.images[1]}
                                        alt={product.title}
                                        className="h-full w-full object-
cover object-center"
                                    />
                                </div>
                                <div className="aspect-h-2 aspect-w-3
overflow-hidden rounded-lg">
                                    <img
                                        src={product.images[2]}
                                        alt={product.title}
                                        className="h-full w-full object-
cover object-center"
                                    />
                                </div>
                            </div>
                            <div className="aspect-h-5 aspect-w-4
lg:aspect-h-4 lg:aspect-w-3 sm:overflow-hidden sm:rounded-lg">
                                <img
                                    src={product.images[3]}
                                    alt={product.title}
```

```
                                    className="h-full w-full object-cover
object-center"
                                />
                            </div>
                        </div>

                        <div className="mx-auto max-w-2xl px-4 pb-16 pt-10
sm:px-6 lg:grid lg:max-w-7xl lg:grid-cols-3 lg:grid-rows-[auto,auto,1fr]
lg:gap-x-8 lg:px-8 lg:pb-24 lg:pt-16">
                            <div className="lg:col-span-2 lg:border-r
lg:border-gray-200 lg:pr-8">
                                <h1 className="text-2xl font-bold tracking-
tight text-gray-900 sm:text-3xl">{product.title}</h1>
                            </div>

                            <div className="mt-4 lg:row-span-3 lg:mt-0">
                                <h2 className="sr-only">Product
information</h2>
                                <p className="text-3xl tracking-tight text-
gray-900">${product.discountedPrice}</p>

                                <div className="mt-6">
                                    <h3 className="sr-only">Reviews</h3>
                                    <div className="flex items-center">
                                        <div className="flex items-center">
                                            {[0, 1, 2, 3, 4].map((rating)
=> (
                                                <StarIcon
                                                    key={rating}
                                                    className={classNames(
```

```
                                                        product.rating >
rating ? 'text-gray-900' : 'text-gray-200',
                                                        'h-5 w-5 flex-
shrink-0'
                                                    )}
                                                    aria-hidden="true"
                                                />
                                            ))}
                                        </div>
                                        <p className="sr-
only">{product.rating} out of 5 stars</p>
                                    </div>
                                </div>

                                <form className="mt-10">
                                    {product.colors &&
product.colors.length > 0 && <div>
                                        <h3 className="text-sm font-medium
text-gray-900">Color</h3>

                                        <RadioGroup value={selectedColor}
onChange={setSelectedColor} className="mt-4">
                                            <RadioGroup.Label
className="sr-only">Choose a color</RadioGroup.Label>
                                            <div className="flex items-
center space-x-3">
                                                {product.colors.map((color)
=> (
                                                    <RadioGroup.Option
                                                        key={color.name}
                                                        value={color}
```

```
                                                        className={({
active, checked }) =>
                                                            classNames(

color.selectedClass,
                                                                active &&
checked ? 'ring ring-offset-1' : '',
                                                                !active &&
checked ? 'ring-2' : '',
                                                                'relative -
m-0.5 flex cursor-pointer items-center justify-center rounded-full p-0.5
focus:outline-none'
                                                            )
                                                        }
                                                    >
                                                        <RadioGroup.Label
as="span" className="sr-only">
                                                            {color.name}
                                                        </RadioGroup.Label>
                                                        <span
                                                            aria-
hidden="true"

className={classNames(

color.class,
                                                                'h-8 w-8
rounded-full border border-black border-opacity-10'
                                                            )}
                                                        />
                                                    </RadioGroup.Option>
```

```
                                                    ))}
                                                </div>
                                            </RadioGroup>
                                        </div>}

                                        {product.sizes && product.sizes.length
> 0 && <div className="mt-10">
                                            <div className="flex items-center
justify-between">
                                                <h3 className="text-sm font-
medium text-gray-900">Size</h3>
                                                <a href="/" className="text-sm
font-medium text-indigo-600 hover:text-indigo-500">
                                                    Size guide
                                                </a>
                                            </div>

                                            <RadioGroup value={selectedSize}
onChange={setSelectedSize} className="mt-4">
                                                <RadioGroup.Label
className="sr-only">Choose a size</RadioGroup.Label>
                                                <div className="grid grid-cols-
4 gap-4 sm:grid-cols-8 lg:grid-cols-4">
                                                    {product.sizes.map((size)
=> (
                                                        <RadioGroup.Option
                                                            key={size.name}
                                                            value={size}

disabled={!size.inStock}
```

```
                                                                    className={({
active }) =>
                                                                        classNames(

size.inStock
                                                                                    ?
'cursor-pointer bg-white text-gray-900 shadow-sm'
                                                                                    :
'cursor-not-allowed bg-gray-50 text-gray-200',
                                                                            active ?
'ring-2 ring-indigo-500' : '',
                                                                            'group
relative flex items-center justify-center rounded-md border py-3 px-4 text-
sm font-medium uppercase hover:bg-gray-50 focus:outline-none sm:flex-1
sm:py-6'
                                                                        )
                                                                    }
                                                                >
                                                                    {({ active, checked
}) => (
                                                                        <>

<RadioGroup.Label as="span">{size.name}</RadioGroup.Label>

{size.inStock ? (
                                                                                    <span

className={classNames(

active ? 'border' : 'border-2',
```

```
checked ? 'border-indigo-500' : 'border-transparent',

'pointer-events-none absolute -inset-px rounded-md'
                                                                    )}

aria-hidden="true"
                                                                    />
                                                                ) : (
                                                                    <span

aria-hidden="true"

className="pointer-events-none absolute -inset-px rounded-md border-2
border-gray-200"
                                                                    >

<svg

className="absolute inset-0 h-full w-full stroke-2 text-gray-200"

viewBox="0 0 100 100"

preserveAspectRatio="none"

stroke="currentColor"
                                                                    >

<line x1={0} y1={100} x2={100} y2={0} vectorEffect="non-scaling-stroke" />

</svg>
```

```
                                                                                    </span>
                                                                                )}
                                                                            </>
                                                                        )}
                                                                    </RadioGroup.Option>
                                                                ))}
                                                            </div>
                                                        </RadioGroup>
                                                    </div>}

                                                    <button
                                                        onClick={handleCart}
                                                        type="submit"
                                                        className="mt-10 flex w-full items-
center justify-center rounded-md border border-transparent bg-indigo-600
px-8 py-3 text-base font-medium text-white hover:bg-indigo-700
focus:outline-none focus:ring-2 focus:ring-indigo-500 focus:ring-offset-2"
                                                    >
                                                        Add to Cart
                                                    </button>
                                                </form>
                                            </div>

                                            <div className="py-10 lg:col-span-2 lg:col-
start-1 lg:border-r lg:border-gray-200 lg:pb-16 lg:pr-8 lg:pt-6">
                                                <div>
                                                    <h3 className="sr-
only">Description</h3>

                                                    <div className="space-y-6">
```

```
                                        <p className="text-base text-gray-
900">{product.description}</p>
                                    </div>
                                </div>

                                {product.highlights &&
product.highlights.length > 0 && <div className="mt-10">
                                    <h3 className="text-sm font-medium
text-gray-900">Highlights</h3>
                                    <div className="mt-4">
                                        <ul className="list-disc space-y-2
pl-4 text-sm">

{product.highlights.map((highlight) => (
                                                <li key={highlight}
className="text-gray-400">
                                                    <span className="text-
gray-600">{highlight}</span>
                                                </li>
                                            ))}
                                        </ul>
                                    </div>
                                </div>}

                                <div className="mt-10">
                                    <h2 className="text-sm font-medium
text-gray-900">Details</h2>

                                    <div className="mt-4 space-y-6">
                                        <p className="text-sm text-gray-
600">{product.description}</p>
```

```
                                    </div>
                                </div>
                            </div>
                        </div>
                    </div>)
                }
            </div>
        </div>
    )
}
```

AdminProductList.js

```
import React, { useState, Fragment, useEffect } from 'react';
import { useDispatch, useSelector } from 'react-redux';

import { Dialog, Disclosure, Menu, Transition } from '@headlessui/react';
import { XMarkIcon } from '@heroicons/react/24/outline';
import { ChevronDownIcon, FunnelIcon, MinusIcon, PlusIcon, Squares2X2Icon }
from '@heroicons/react/20/solid';
import { ChevronLeftIcon, ChevronRightIcon, StarIcon } from
'@heroicons/react/20/solid';

import { selectAllProducts, fetchProductsByFilterAsync, selectTotalItems,
selectBrands, selectCategories, fetchAllCategoriesAsync,
fetchAllBrandsAsync } from '../../product/productSlice';
import { ITEMS_PER_PAGE } from '../../../app/constants';

import { Link } from 'react-router-dom';
```

```
const sortOptions = [
  { name: 'Best Rating', sort: 'rating', order: 'desc', current: false },
  { name: 'Price: Low to High', sort: 'discountedPrice', order: 'asc',
current: false },
  { name: 'Price: High to Low', sort: 'discountedPrice', order: 'desc',
current: false }
];

function classNames(...classes) {
  return classes.filter(Boolean).join(' ');
}



export default function AdminProductList() {
  const [mobileFiltersOpen, setMobileFiltersOpen] = useState(false);
  const [filter, setFilter] = useState({});
  const [sort, setSort] = useState({});
  const [page, setPage] = useState(1);

  const dispatch = useDispatch();
  const products = useSelector(selectAllProducts);
  const brands = useSelector(selectBrands);
  const categories = useSelector(selectCategories);
  const totalItems = useSelector(selectTotalItems);

  const filters = [
    {
      id: 'category',
      name: 'Category',
      options: categories,
    },
```

```
    {
      id: 'brand',
      name: 'Brand',
      options: brands,
    }
  ];

  const handleFilter = (e, section, option) => {
    const newFilter = { ...filter };
    if (e.target.checked) {
      if (newFilter[section.id]) {
        newFilter[section.id].push(option.value);
      }
      else {
        newFilter[section.id] = [option.value];
      }
    }
    else {
      const index = newFilter[section.id].findIndex(el => el ===
option.value);
      newFilter[section.id].splice(index, 1);
    }
    setFilter(newFilter);
  }

  const handleSort = (e, option) => {
    const newSort = { _sort: option.sort, _order: option.order };
    setSort(newSort);
  }

  const handlePage = (page) => {
```

```
    setPage(page);
  }

  useEffect(() => {
    const pagination = { _page: page, _limit: ITEMS_PER_PAGE };
    dispatch(fetchProductsByFilterAsync({ filter, sort, pagination, admin:
true }));
  }, [dispatch, filter, sort, page]);

  useEffect(() => {
    setPage(1);
  }, [totalItems, filter, sort]);

  useEffect(() => {
    dispatch(fetchAllBrandsAsync());
    dispatch(fetchAllCategoriesAsync());
  }, [dispatch]);



  return (
    <div>
      <div className="bg-white rounded-lg">
        <div>
          <MobileFilter mobileFiltersOpen={mobileFiltersOpen}
setMobileFiltersOpen={setMobileFiltersOpen} handleFilter={handleFilter}
filters={filters} />

          <main className="mx-auto max-w-7xl px-4 sm:px-6 lg:px-8">
            <div className="flex items-baseline justify-between border-b
border-gray-200 pb-6 pt-24">
```

```
              <h1 className="text-4xl font-bold tracking-tight text-gray-
900">All Products</h1>

              <div className="flex items-center">
                <Menu as="div" className="relative inline-block text-left">
                  <div>
                    <Menu.Button className="group inline-flex justify-
center text-sm font-medium text-gray-700 hover:text-gray-900">
                      Sort
                      <ChevronDownIcon
                        className="-mr-1 ml-1 h-5 w-5 flex-shrink-0 text-
gray-400 group-hover:text-gray-500"
                        aria-hidden="true"
                      />
                    </Menu.Button>
                  </div>

                  <Transition
                    as={Fragment}
                    enter="transition ease-out duration-100"
                    enterFrom="transform opacity-0 scale-95"
                    enterTo="transform opacity-100 scale-100"
                    leave="transition ease-in duration-75"
                    leaveFrom="transform opacity-100 scale-100"
                    leaveTo="transform opacity-0 scale-95"
                  >
                    <Menu.Items className="absolute right-0 z-10 mt-2 w-40
origin-top-right rounded-md bg-white shadow-2xl ring-1 ring-black ring-
opacity-5 focus:outline-none">
                      <div className="py-1">
                        {sortOptions.map((option) => (
```

```
                        <Menu.Item key={option.name}>
                          {({ active }) => (
                            <p
                              onClick={e => handleSort(e, option)}
                              className={classNames(
                                option.current ? 'font-medium text-gray-
900' : 'text-gray-500',
                                active ? 'bg-gray-100' : '',
                                'block px-4 py-2 text-sm'
                              )}
                            >
                              {option.name}
                            </p>
                          )}
                        </Menu.Item>
                      ))}
                    </div>
                  </Menu.Items>
                </Transition>
              </Menu>

              <button type="button" className="-m-2 ml-5 p-2 text-gray-
400 hover:text-gray-500 sm:ml-7">
                <span className="sr-only">View grid</span>
                <Squares2X2Icon className="h-5 w-5" aria-hidden="true" />
              </button>
              <button
                type="button"
                className="-m-2 ml-4 p-2 text-gray-400 hover:text-gray-
500 sm:ml-6 lg:hidden"
                onClick={() => setMobileFiltersOpen(true)}
```

```
                >
                  <span className="sr-only">Filters</span>
                  <FunnelIcon className="h-5 w-5" aria-hidden="true" />
                </button>
              </div>
            </div>

            <section aria-labelledby="products-heading" className="pb-24
pt-6">
              <h2 id="products-heading" className="sr-only">
                Products
              </h2>

              <div className="grid grid-cols-1 gap-x-8 gap-y-10 lg:grid-
cols-4">
                <DesktopFilter handleFilter={handleFilter}
filters={filters} />

                <div className="lg:col-span-3">
                  <div className='px-8'>
                    <Link to='/admin/product-form' type="button"
className="rounded-md w-full bg-green-600 px-3 py-2 my-2 text-sm font-
semibold text-white shadow-sm hover:bg-green-500 focus-visible:outline
focus-visible:outline-2 focus-visible:outline-offset-2 focus-
visible:outline-green-600">
                      + Add new product
                    </Link>
                  </div>
                  <ProductGrid products={products} />
                </div>
              </div>
```

```
            </section>

            <Pagination page={page} setPage={setPage}
handlePage={handlePage} totalItems={totalItems} />
          </main>
        </div>
      </div>
    </div>
  );
}


function MobileFilter({ mobileFiltersOpen, setMobileFiltersOpen,
handleFilter, filters }) {
  return (
    <Transition.Root show={mobileFiltersOpen} as={Fragment}>
      <Dialog as="div" className="relative z-40 lg:hidden"
onClose={setMobileFiltersOpen}>
        <Transition.Child
          as={Fragment}
          enter="transition-opacity ease-linear duration-300"
          enterFrom="opacity-0"
          enterTo="opacity-100"
          leave="transition-opacity ease-linear duration-300"
          leaveFrom="opacity-100"
          leaveTo="opacity-0"
        >
          <div className="fixed inset-0 bg-black bg-opacity-25" />
        </Transition.Child>

        <div className="fixed inset-0 z-40 flex">
```

```
          <Transition.Child
            as={Fragment}
            enter="transition ease-in-out duration-300 transform"
            enterFrom="translate-x-full"
            enterTo="translate-x-0"
            leave="transition ease-in-out duration-300 transform"
            leaveFrom="translate-x-0"
            leaveTo="translate-x-full"
          >
            <Dialog.Panel className="relative ml-auto flex h-full w-full
max-w-xs flex-col overflow-y-auto bg-white py-4 pb-12 shadow-xl">
              <div className="flex items-center justify-between px-4">
                <h2 className="text-lg font-medium text-gray-
900">Filters</h2>
                <button
                  type="button"
                  className="-mr-2 flex h-10 w-10 items-center justify-
center rounded-md bg-white p-2 text-gray-400"
                  onClick={() => setMobileFiltersOpen(false)}
                >
                  <span className="sr-only">Close menu</span>
                  <XMarkIcon className="h-6 w-6" aria-hidden="true" />
                </button>
              </div>

              <form className="mt-4 border-t border-gray-200">
                {filters.map((section) => (
                  <Disclosure as="div" key={section.id} className="border-t
border-gray-200 px-4 py-6">
                    {({ open }) => (
                      <>
```

```
                        <h3 className="-mx-2 -my-3 flow-root">
                          <Disclosure.Button className="flex w-full items-
center justify-between bg-white px-2 py-3 text-gray-400 hover:text-gray-
500">
                            <span className="font-medium text-gray-
900">{section.name}</span>
                            <span className="ml-6 flex items-center">
                              {open ? (
                                <MinusIcon className="h-5 w-5" aria-
hidden="true" />
                              ) : (
                                <PlusIcon className="h-5 w-5" aria-
hidden="true" />
                              )}
                            </span>
                          </Disclosure.Button>
                        </h3>
                        <Disclosure.Panel className="pt-6">
                          <div className="space-y-6">
                            {section.options.map((option, optionIdx) => (
                              <div key={option.value} className="flex
items-center">
                                <input
                                  id={`filter-mobile-${section.id}-
${optionIdx}`}
                                  name={`${section.id}[]`}
                                  defaultValue={option.value}
                                  type="checkbox"
                                  defaultChecked={option.checked}
                                  onChange={e => handleFilter(e, section,
option)}
```

```
                                  className="h-4 w-4 rounded border-gray-
300 text-indigo-600 focus:ring-indigo-500"
                                />
                                <label
                                  htmlFor={`filter-mobile-${section.id}-
${optionIdx}`}
                                  className="ml-3 min-w-0 flex-1 text-gray-
500"
                                >
                                  {option.label[0].toUpperCase() +
option.label.slice(1)}
                                </label>
                              </div>
                            ))}
                          </div>
                        </Disclosure.Panel>
                      </>
                    )}
                  </Disclosure>
                ))}
              </form>
            </Dialog.Panel>
          </Transition.Child>
        </div>
      </Dialog>
    </Transition.Root>
  );
}


function DesktopFilter({ handleFilter, filters }) {
```

```
  return (
    <form className="hidden lg:block">
      {filters.map((section) => (
        <Disclosure as="div" key={section.id} className="border-b border-
gray-200 py-6">
          {({ open }) => (
            <>
              <h3 className="-my-3 flow-root">
                <Disclosure.Button className="flex w-full items-center
justify-between bg-white py-3 text-sm text-gray-400 hover:text-gray-500">
                  <span className="font-medium text-gray-
900">{section.name}</span>
                  <span className="ml-6 flex items-center">
                    {open ? (
                      <MinusIcon className="h-5 w-5" aria-hidden="true" />
                    ) : (
                      <PlusIcon className="h-5 w-5" aria-hidden="true" />
                    )}
                  </span>
                </Disclosure.Button>
              </h3>
              <Disclosure.Panel className="pt-6">
                <div className="space-y-4">
                  {section.options.map((option, optionIdx) => (
                    <div key={option.value} className="flex items-center">
                      <input
                        id={`filter-${section.id}-${optionIdx}`}
                        name={`${section.id}[]`}
                        defaultValue={option.value}
                        type="checkbox"
                        defaultChecked={option.checked}
```

```
                        onChange={e => handleFilter(e, section, option)}
                        className="h-4 w-4 rounded border-gray-300 text-
indigo-600 focus:ring-indigo-500"
                      />
                      <label
                        htmlFor={`filter-${section.id}-${optionIdx}`}
                        className="ml-3 text-sm text-gray-600"
                      >
                        {option.label[0].toUpperCase() +
option.label.slice(1)}
                      </label>
                    </div>
                  ))}
                </div>
              </Disclosure.Panel>
            </>
          )}
        </Disclosure>
      ))}
    </form>
  );
}



function ProductGrid({ products }) {
  return (
    <div className="bg-white">
      <div className="mx-auto max-w-2xl px-4 py-0 sm:px-6 sm:py-0 lg:max-w-
7xl lg:px-8">
        <div className="mt-6 grid grid-cols-1 gap-x-6 gap-y-10 sm:grid-
cols-2 lg:grid-cols-3 xl:gap-x-8">
```

```
          {products.map((product) => (
            <div key={product.id}>
              <Link to={`/product-detail/${product.id}`}>
                <div className="group relative border border-1 mb-3 border-
gray-400 p-3 rounded-md">
                  <div className="aspect-h-1 aspect-w-1 w-full overflow-
hidden rounded-md bg-gray-200 lg:aspect-none group-hover:opacity-75 lg:h-
60">
                    <img
                      src={product.thumbnail}
                      alt={product.title}
                      className="h-full w-full object-cover object-center
lg:h-full lg:w-full"
                    />
                  </div>
                  <div className="mt-4 flex justify-between">
                    <div>
                      <h3 className="text-sm text-gray-700">
                        <div href={product.thumbnail}>
                          <span aria-hidden="true" className="absolute
inset-0" />
                          {product.title}
                        </div>
                      </h3>
                      <p className="mt-1 text-sm text-gray-500 flex items-
center">
                        <StarIcon className='w-5 h-5 inline' />
                        <span className='mx-1 mt-
1'>{product.rating}/5</span>
                      </p>
                    </div>
```

```
                    <div className='flex flex-col items-center'>
                      <p className="text-md font-medium text-gray-
900">${product.discountedPrice}</p>
                      <p className="text-xs font-normal text-gray-400 line-
through">${product.price}</p>
                    </div>
                  </div>
                  {product.deleted && <div>
                    <p className='text-sm text-red-500 mt-2'>Product
deleted</p>
                  </div>}
                  {product.stock <= 0 && <div>
                    <p className='text-sm text-red-500 mt-2'>Out of
stock</p>
                  </div>}
                </div>
              </Link>
              <div className='mb-2'>
                <Link to={`/admin/product-form/edit/${product.id}`}
type="button" className="rounded-md bg-indigo-600 px-3 py-2 text-sm font-
semibold text-white shadow-sm hover:bg-indigo-500 focus-visible:outline
focus-visible:outline-2 focus-visible:outline-offset-2 focus-
visible:outline-indigo-600">
                  Edit product
                </Link>
              </div>
            </div>
          ))}
        </div>
      </div>
    </div>
```

```
  );
}


function Pagination({ page, setPage, handlePage, totalItems }) {
  const totalPages = Math.ceil(totalItems / ITEMS_PER_PAGE);
  return (
    <div className="flex items-center justify-between border-t border-gray-
200 bg-white px-4 py-3 sm:px-6">
      <div className="flex flex-1 justify-between sm:hidden">
        <div
          onClick={(e) => handlePage(page > 1 ? page - 1 : page)}
          className="relative inline-flex items-center rounded-md border
border-gray-300 bg-white px-4 py-2 text-sm font-medium text-gray-700
hover:bg-gray-50"
        >
          Previous
        </div>
        <div
          onClick={(e) => handlePage(totalPages > page ? page + 1 : page)}
          className="relative ml-3 inline-flex items-center rounded-md
border border-gray-300 bg-white px-4 py-2 text-sm font-medium text-gray-700
hover:bg-gray-50"
        >
          Next
        </div>
      </div>
      <div className="hidden sm:flex sm:flex-1 sm:items-center sm:justify-
between">
        <div>
          <p className="text-sm text-gray-700">
```

```
            Showing{' '}
            <span className="font-medium">{(page - 1) * ITEMS_PER_PAGE +
1}</span>{' '}
            to{' '}
            <span className="font-medium">{page * ITEMS_PER_PAGE >
totalItems ? totalItems : page * ITEMS_PER_PAGE}</span>{' '}
            of{' '}
            <span className="font-medium">{totalItems}</span>{' '}
            results
          </p>
        </div>
        <div>
          <nav className="isolate inline-flex -space-x-px rounded-md
shadow-sm" aria-label="Pagination">
            <div
              onClick={(e) => handlePage(page > 1 ? page - 1 : page)}
              className="relative inline-flex items-center rounded-l-md px-
2 py-2 text-gray-400 ring-1 ring-inset ring-gray-300 hover:bg-gray-50
focus:z-20 focus:outline-offset-0 cursor-pointer"
            >
              <span className="sr-only">Previous</span>
              <ChevronLeftIcon className="h-5 w-5" aria-hidden="true" />
            </div>



            {Array.from({ length: totalPages }).map((el, index) => (
              <div
                key={index}
                onClick={e => handlePage(index + 1)}
                aria-current="page"
```

```
                className={`relative z-10 inline-flex items-center ${page
=== index + 1 ? 'bg-indigo-600 text-white' : 'bg-white text-gray-500'} px-4
py-2 text-sm font-semibold focus:z-20 focus-visible:outline focus-
visible:outline-2 focus-visible:outline-offset-2 focus-visible:outline-
indigo-600 cursor-pointer`}
              >
                {index + 1}
              </div>
            )
            )}



            <div
              onClick={(e) => handlePage(totalPages > page ? page + 1 :
page)}
              className="relative inline-flex items-center rounded-r-md px-
2 py-2 text-gray-400 ring-1 ring-inset ring-gray-300 hover:bg-gray-50
focus:z-20 focus:outline-offset-0 cursor-pointer"
            >
              <span className="sr-only">Next</span>
              <ChevronRightIcon className="h-5 w-5" aria-hidden="true" />
            </div>
          </nav>
        </div>
      </div>
    </div>
  );
}
```

AdminQuery.js

```
import React, { useEffect } from 'react';
import { useDispatch, useSelector } from 'react-redux';
import { fetchAllQueriesAsync, selectAllQueries } from
'../../query/querySlice';

function AdminQuery() {
    const dispatch = useDispatch();
    const queries = useSelector(selectAllQueries);

    useEffect(() => {
        dispatch(fetchAllQueriesAsync());
    }, [dispatch]);

    return (
        <div>
            <div className="overflow-x-auto mt-2">
                <div className="bg-gray-100 flex items-center justify-
center font-sans overflow-hidden rounded-lg">
                    <div className="w-full">
                        <div className="bg-white shadow-md">
                            <table className="w-full table-auto">
                                <thead>
                                    <tr className="bg-gray-200 text-gray-
600 uppercase text-sm leading-normal">
                                        <th className="py-3 px-3 text-left
w-1/4">Email address</th>
                                        <th className="py-3 px-3 text-left
w-2/4">Query</th>
```

```
                                        <th className="py-3 px-3 text-right
w-1/4">Sent at</th>
                                    </tr>
                                </thead>
                                <tbody className="text-gray-600 text-sm
font-light">
                                    {queries.map((queryIterator, index) =>
(
                                        <tr key={index} className="border-b
border-gray-200 hover:bg-gray-100">
                                            <td className="py-3 px-3 text-
left whitespace-nowrap w-1/4">
                                                <div className="flex items-
center">
                                                    <span className="font-
medium text-md text-green-500">{queryIterator.userEmail}</span>
                                                </div>
                                            </td>
                                            <td className="py-3 px-3 text-
left w-2/4">
                                                <div className="flex items-
center">
                                                    <span className="font-
normal text-md">{queryIterator.query}</span>
                                                </div>
                                            </td>
                                            <td className="py-3 px-3 text-
left w-1/4">
                                                <div className="flex items-
center justify-end">
```

```
                                                        <span className="font-
medium text-md text-blue-600">{new
Date(queryIterator.createdAt).toLocaleString()}</span>
                                                        </div>
                                                    </td>
                                                </tr>
                                            ))}
                                        </tbody>
                                    </table>
                                </div>
                            </div>
                        </div>
                    </div>
                </div>
    )
}
export default AdminQuery;
```

ProductForm.js

```
import React, { useEffect, useState } from 'react';
import { clearSelectedProduct, createProductAsync, fetchProductByIdAsync,
selectBrands, selectCategories, selectProductById, updateProductAsync }
from '../../product/productSlice';
import { useDispatch, useSelector } from 'react-redux';
import { useForm } from 'react-hook-form';
import { useParams } from 'react-router-dom';
import { useNavigate } from "react-router-dom";
import Modal from '../../common/Modal';
```

```
function ProductForm() {
    const categories = useSelector(selectCategories);
    const brands = useSelector(selectBrands);

    const dispatch = useDispatch();

    const { register, handleSubmit, formState: { errors }, setValue, reset
} = useForm();
    console.log(errors);

    const navigate = useNavigate();

    const params = useParams();
    const selectedProduct = useSelector(selectProductById);

    const [openModal, setOpenModal] = useState(null);

    const colors = [
        { name: 'White', class: 'bg-white', selectedClass: 'ring-gray-400',
id: 'white' },
        { name: 'Gray', class: 'bg-gray-200', selectedClass: 'ring-gray-
400', id: 'gray' },
        { name: 'Black', class: 'bg-gray-900', selectedClass: 'ring-gray-
900', id: 'black' },
        { name: 'Red', class: 'bg-red-500', selectedClass: 'ring-red-900',
id: 'red' },
        { name: 'Blue', class: 'bg-blue-500', selectedClass: 'ring-blue-
900', id: 'blue' },
        { name: '-', class: 'bg-white', selectedClass: 'ring-gray-400', id:
'std' }
    ];
```

```
    const sizes = [
        { name: 'XXS', inStock: true, id: 'xxs' },
        { name: 'XS', inStock: true, id: 'xs' },
        { name: 'S', inStock: true, id: 'sm' },
        { name: 'M', inStock: true, id: 'md' },
        { name: 'L', inStock: true, id: 'lg' },
        { name: 'XL', inStock: true, id: 'xl' },
        { name: '2XL', inStock: true, id: 'xxl' },
        { name: '-', inStock: true, id: 'std' }
    ];
    useEffect(() => {
        if (params.id) {
            dispatch(fetchProductByIdAsync(params.id));
        }
        else {
            dispatch(clearSelectedProduct());
        }
    }, [params.id, dispatch]);

    useEffect(() => {
        if (selectedProduct && params.id) {
            setValue('title', selectedProduct.title);
            setValue('description', selectedProduct.description);
            setValue('price', selectedProduct.price);
            setValue('discountPercentage',
selectedProduct.discountPercentage);
            setValue('stock', selectedProduct.stock);
            setValue('brand', selectedProduct.brand);
            setValue('category', selectedProduct.category);
            setValue('thumbnail', selectedProduct.thumbnail);
```

```
            setValue('image1', selectedProduct.images[0]);
            setValue('image2', selectedProduct.images[1]);
            setValue('image3', selectedProduct.images[2]);
            setValue('image4', selectedProduct.images[3]);
            setValue('highlight1', selectedProduct.highlights[0]);
            setValue('highlight2', selectedProduct.highlights[1]);
            setValue('highlight3', selectedProduct.highlights[2]);
            setValue('highlight4', selectedProduct.highlights[3]);
            setValue('sizes', selectedProduct.sizes.map(size => size.id));
            setValue('colors', selectedProduct.colors.map(color =>
color.id));
        }
    }, [selectedProduct, params.id, setValue]);

    const handleDelete = () => {
        const product = { ...selectedProduct };
        product.deleted = true;
        dispatch(updateProductAsync(product));
    }

    return (
        <div>
            {brands && categories && <form noValidate action="#"
method="POST" onSubmit={handleSubmit((data) => {
                const product = { ...data };
                product.images = [product.image1, product.image2,
product.image3, product.image4];
                product.highlights = [product.highlight1,
product.highlight2, product.highlight3, product.highlight4];
                delete product['image1'];
                delete product['image2'];
```

```
                delete product['image3'];
                delete product['image4'];
                product.colors = product.colors.map(color =>
colors.find(clr => clr.id === color));
                product.sizes = product.sizes.map(size => sizes.find(sz =>
sz.id === size));
                product.price = +product.price;
                product.stock = +product.stock;
                product.discountPercentage = +product.discountPercentage;

                if (params.id) {
                    product.id = params.id;
                    product.rating = selectedProduct.rating;
                    dispatch(updateProductAsync(product));
                    reset();
                    navigate('/admin');
                }
                else {
                    product.rating = 0;
                    dispatch(createProductAsync(product));
                    reset();
                    navigate('/admin');
                }
            })}>
                <div className="space-y-12 bg-white p-10 rounded-lg">
                    <div className="border-b border-gray-900/10 pb-12">
                        <h2 className="text-2xl font-semibold leading-7
text-gray-900">Add a new Product</h2>
                        <p className="text-sm leading-6 text-gray-600">
                            Make sure to give detailed information about
the new product.
```

```
                        </p>

                        {selectedProduct && selectedProduct.deleted ? <h2
className="text-red-500 text-lg mt-6 font-bold">This product is
deleted</h2> : <h2 className="text-green-500 text-lg mt-6 font-bold">This
product exists</h2>}

                        <div className="mt-10 grid grid-cols-1 gap-x-6 gap-
y-8 sm:grid-cols-6">
                            <div className="sm:col-span-6">
                                <label htmlFor="title" className="block
text-md font-medium leading-6 text-gray-900">
                                    Product title
                                </label>
                                <div className="mt-2">
                                    <div className="flex rounded-md shadow-
sm ring-1 ring-inset ring-gray-300 focus-within:ring-2 focus-within:ring-
inset focus-within:ring-indigo-600">
                                        <input
                                            type="text"
                                            {...register('title', {
required: 'Title is required' })}
                                            id="title"
                                            className="block flex-1 border-
0 bg-transparent py-1 px-3 text-gray-900 placeholder:text-gray-400
focus:ring-0 sm:text-sm sm:leading-6"
                                        />
                                    </div>
                                </div>
                            </div>
```

```
                            <div className="col-span-full">
                                <label htmlFor="description"
className="block text-md font-medium leading-6 text-gray-900">
                                    Description
                                </label>
                                <div className="mt-2">
                                    <textarea
                                        id="description"
                                        {...register('description', {
required: 'Description is required' })}
                                        rows={3}
                                        className="block w-full rounded-md
border-0 py-1.5 text-gray-900 shadow-sm ring-1 ring-inset ring-gray-300
placeholder:text-gray-400 focus:ring-2 focus:ring-inset focus:ring-indigo-
600 sm:text-sm sm:leading-6"
                                        defaultValue={''}
                                    />
                                </div>
                                <p className="mt-3 text-sm leading-6 text-
gray-600">Write a few lines about your product.</p>
                            </div>

                            <div className="sm:col-span-6">
                                <label htmlFor="brand" className="block
text-md font-medium leading-6 text-gray-900">
                                    Product brand
                                </label>
                                <div className="mt-2">
                                    <select
                                        {...register('brand', { required:
'Brand is required' })}
```

```
                                        id="brand"
                                    >
                                        <option value="choose">Choose
brand</option>
                                        {brands.map((brand, index) => (
                                            <option key={index}
value={brand.value}>{brand.label}</option>
                                        ))}
                                    </select>
                                </div>
                            </div>

                            <div className="sm:col-span-6">
                                <label htmlFor="colors" className="block
text-md font-medium leading-6 text-gray-900">
                                    Product Colors
                                </label>
                                <div className="mt-2">
                                        {colors.map((color) => (
                                            <div className='mr-6 inline-
flex items-center gap-1' key={color.id}>
                                                <input
                                                    className='cursor-
pointer'
                                                    {...register('colors')}
                                                    type='checkbox'
                                                    value={color.id}
                                                    /> {color.name}
                                            </div>
                                        ))}
                                </div>
```

```
                            </div>

                            <div className="sm:col-span-6">
                                <label htmlFor="sizes" className="block
text-md font-medium leading-6 text-gray-900">
                                    Product Sizes
                                </label>
                                <div className="mt-2">
                                        {sizes.map((size) => (
                                            <div className='mr-6 inline-
flex items-center gap-1' key={size.id}>
                                                <input
                                                    className='cursor-
pointer'
                                                    {...register('sizes')}
                                                    type='checkbox'
                                                    value={size.id}
                                                    /> {size.name}
                                            </div>
                                        ))}
                                </div>
                            </div>

                            <div className="sm:col-span-6">
                                <label htmlFor="category" className="block
w-full text-md font-medium leading-6 text-gray-900">
                                    Product category
                                </label>
                                <div className="mt-2">
                                    <select
```

```
                                        {...register('category', {
required: 'Category is required' })}
                                        id="category"
                                    >
                                        <option value="choose">Choose
category</option>
                                        {categories.map((category, index)
=> (
                                            <option key={index}
value={category.value}>{category.label}</option>
                                        ))}
                                    </select>
                                </div>
                            </div>

                            <div className="sm:col-span-2">
                                <label htmlFor="price" className="block
text-md font-medium leading-6 text-gray-900">
                                    Price
                                </label>
                                <div className="mt-2">
                                    <div className="flex rounded-md shadow-
sm ring-1 ring-inset ring-gray-300 focus-within:ring-2 focus-within:ring-
inset focus-within:ring-indigo-600">
                                        <input
                                            type="number"
                                            {...register('price', {
required: 'Price is required', min: 1, max: 100000 })}
                                            id="price"
```

```
                                            className="block flex-1 border-
0 bg-transparent py-1 px-3 text-gray-900 placeholder:text-gray-400
focus:ring-0 sm:text-sm sm:leading-6"
                                        />
                                    </div>
                                </div>
                            </div>

                            <div className="sm:col-span-2">
                                <label htmlFor="discountPercentage"
className="block text-md font-medium leading-6 text-gray-900">
                                    Discount Percentage
                                </label>
                                <div className="mt-2">
                                    <div className="flex rounded-md shadow-
sm ring-1 ring-inset ring-gray-300 focus-within:ring-2 focus-within:ring-
inset focus-within:ring-indigo-600">
                                        <input
                                            type="number"

{...register('discountPercentage', { required: 'Discount Percentage is
required', min: 0, max: 100 })}
                                            id="discountPercentage"
                                            className="block flex-1 border-
0 bg-transparent py-1 px-3 text-gray-900 placeholder:text-gray-400
focus:ring-0 sm:text-sm sm:leading-6"
                                        />
                                    </div>
                                </div>
                            </div>
```

```
                            <div className="sm:col-span-2">
                                <label htmlFor="stock" className="block
text-md font-medium leading-6 text-gray-900">
                                    Stock
                                </label>
                                <div className="mt-2">
                                    <div className="flex rounded-md shadow-
sm ring-1 ring-inset ring-gray-300 focus-within:ring-2 focus-within:ring-
inset focus-within:ring-indigo-600">
                                        <input
                                            type="number"
                                            {...register('stock', {
required: 'Stock is required', min: 0 })}
                                            id="stock"
                                            className="block flex-1 border-
0 bg-transparent py-1 px-3 text-gray-900 placeholder:text-gray-400
focus:ring-0 sm:text-sm sm:leading-6"
                                        />
                                    </div>
                                </div>
                            </div>

                            <div className="sm:col-span-6">
                                <label htmlFor="thumbnail" className="block
text-md font-medium leading-6 text-gray-900">
                                    Product thumbnail
                                </label>
                                <div className="mt-2">
                                    <div className="flex rounded-md shadow-
sm ring-1 ring-inset ring-gray-300 focus-within:ring-2 focus-within:ring-
inset focus-within:ring-indigo-600">
```

```
                                        <input
                                            type="text"
                                            {...register('thumbnail', {
required: 'Thumbnail is required' })}
                                            id="thumbnail"
                                            className="block flex-1 border-
0 bg-transparent py-1 px-3 text-gray-900 placeholder:text-gray-400
focus:ring-0 sm:text-sm sm:leading-6"
                                        />
                                    </div>
                                </div>
                            </div>

                            <div className="sm:col-span-3">
                                <label htmlFor="image1" className="block
text-md font-medium leading-6 text-gray-900">
                                    Image 1
                                </label>
                                <div className="mt-2">
                                    <div className="flex rounded-md shadow-
sm ring-1 ring-inset ring-gray-300 focus-within:ring-2 focus-within:ring-
inset focus-within:ring-indigo-600">
                                        <input
                                            type="text"
                                            {...register('image1', {
required: 'Image 1 is required' })}
                                            id="image1"
                                            className="block flex-1 border-
0 bg-transparent py-1 px-3 text-gray-900 placeholder:text-gray-400
focus:ring-0 sm:text-sm sm:leading-6"
                                        />
```

```
                                    </div>
                                </div>
                            </div>
                            <div className="sm:col-span-3">
                                <label htmlFor="image2" className="block
text-md font-medium leading-6 text-gray-900">
                                    Image 2
                                </label>
                                <div className="mt-2">
                                    <div className="flex rounded-md shadow-
sm ring-1 ring-inset ring-gray-300 focus-within:ring-2 focus-within:ring-
inset focus-within:ring-indigo-600">
                                        <input
                                            type="text"
                                            {...register('image2', {
required: 'Image 2 is required' })}
                                            id="image2"
                                            className="block flex-1 border-
0 bg-transparent py-1 px-3 text-gray-900 placeholder:text-gray-400
focus:ring-0 sm:text-sm sm:leading-6"
                                        />
                                    </div>
                                </div>
                            </div>
                            <div className="sm:col-span-3">
                                <label htmlFor="image3" className="block
text-md font-medium leading-6 text-gray-900">
                                    Image 3
                                </label>
                                <div className="mt-2">
```

```
                                    <div className="flex rounded-md shadow-
sm ring-1 ring-inset ring-gray-300 focus-within:ring-2 focus-within:ring-
inset focus-within:ring-indigo-600">
                                        <input
                                            type="text"
                                            {...register('image3', {
required: 'Image 3 is required' })}
                                            id="image3"
                                            className="block flex-1 border-
0 bg-transparent py-1 px-3 text-gray-900 placeholder:text-gray-400
focus:ring-0 sm:text-sm sm:leading-6"
                                        />
                                    </div>
                                </div>
                            </div>
                            <div className="sm:col-span-3">
                                <label htmlFor="image4" className="block
text-md font-medium leading-6 text-gray-900">
                                    Image 4
                                </label>
                                <div className="mt-2">
                                    <div className="flex rounded-md shadow-
sm ring-1 ring-inset ring-gray-300 focus-within:ring-2 focus-within:ring-
inset focus-within:ring-indigo-600">
                                        <input
                                            type="text"
                                            {...register('image4', {
required: 'Image 4 is required' })}
                                            id="image4"
```

```
                                            className="block flex-1 border-
0 bg-transparent py-1 px-3 text-gray-900 placeholder:text-gray-400
focus:ring-0 sm:text-sm sm:leading-6"
                                        />
                                    </div>
                                </div>
                            </div>




                            <div className="sm:col-span-3">
                                <label htmlFor="highlight1"
className="block text-md font-medium leading-6 text-gray-900">
                                    Highlight 1
                                </label>
                                <div className="mt-2">
                                    <div className="flex rounded-md shadow-
sm ring-1 ring-inset ring-gray-300 focus-within:ring-2 focus-within:ring-
inset focus-within:ring-indigo-600">
                                        <input
                                            type="text"
                                            {...register('highlight1')}
                                            id="highlight1"
                                            className="block flex-1 border-
0 bg-transparent py-1 px-3 text-gray-900 placeholder:text-gray-400
focus:ring-0 sm:text-sm sm:leading-6"
                                        />
                                    </div>
                                </div>
                            </div>
```

```
                            <div className="sm:col-span-3">
                                <label htmlFor="highlight2"
className="block text-md font-medium leading-6 text-gray-900">
                                    Highlight 2
                                </label>
                                <div className="mt-2">
                                    <div className="flex rounded-md shadow-
sm ring-1 ring-inset ring-gray-300 focus-within:ring-2 focus-within:ring-
inset focus-within:ring-indigo-600">
                                        <input
                                            type="text"
                                            {...register('highlight2')}
                                            id="highlight2"
                                            className="block flex-1 border-
0 bg-transparent py-1 px-3 text-gray-900 placeholder:text-gray-400
focus:ring-0 sm:text-sm sm:leading-6"
                                        />
                                    </div>
                                </div>
                            </div>
                            <div className="sm:col-span-3">
                                <label htmlFor="highlight3"
className="block text-md font-medium leading-6 text-gray-900">
                                    Highlight 3
                                </label>
                                <div className="mt-2">
                                    <div className="flex rounded-md shadow-
sm ring-1 ring-inset ring-gray-300 focus-within:ring-2 focus-within:ring-
inset focus-within:ring-indigo-600">
                                        <input
                                            type="text"
```

```
                                                {...register('highlight3')}
                                                id="highlight3"
                                                className="block flex-1 border-
0 bg-transparent py-1 px-3 text-gray-900 placeholder:text-gray-400
focus:ring-0 sm:text-sm sm:leading-6"
                                            />
                                        </div>
                                    </div>
                                </div>
                                <div className="sm:col-span-3">
                                    <label htmlFor="highlight4"
className="block text-md font-medium leading-6 text-gray-900">
                                        Highlight 4
                                    </label>
                                    <div className="mt-2">
                                        <div className="flex rounded-md shadow-
sm ring-1 ring-inset ring-gray-300 focus-within:ring-2 focus-within:ring-
inset focus-within:ring-indigo-600">
                                            <input
                                                type="text"
                                                {...register('highlight4')}
                                                id="highlight4"
                                                className="block flex-1 border-
0 bg-transparent py-1 px-3 text-gray-900 placeholder:text-gray-400
focus:ring-0 sm:text-sm sm:leading-6"
                                            />
                                        </div>
                                    </div>
                                </div>
```

```
                        </div>
                    </div>

                    <div className="border-b border-gray-900/10 pb-12">
                        <h2 className="text-base font-semibold leading-7
text-gray-900">Extras</h2>
                        <p className="mt-1 text-sm leading-6 text-gray-
600">
                            Extra options here.
                        </p>

                        <div className="mt-10 space-y-10">
                            <fieldset>
                                <legend className="text-sm font-semibold
leading-6 text-gray-900">By Email</legend>
                                <div className="mt-6 space-y-6">
                                    <div className="relative flex gap-x-3">
                                        <div className="flex h-6 items-
center">
                                            <input
                                                id="comments"
                                                name="comments"
                                                type="checkbox"
                                                className="h-4 w-4 rounded
border-gray-300 text-indigo-600 focus:ring-indigo-600"
                                            />
                                        </div>
                                        <div className="text-sm leading-6">
```

```
                                            <label htmlFor="comments"
className="font-medium text-gray-900">
                                                Comments
                                            </label>
                                            <p className="text-gray-
500">Get notified when someones posts a comment on a posting.</p>
                                        </div>
                                    </div>
                                    <div className="relative flex gap-x-3">
                                        <div className="flex h-6 items-
center">
                                            <input
                                                id="candidates"
                                                name="candidates"
                                                type="checkbox"
                                                className="h-4 w-4 rounded
border-gray-300 text-indigo-600 focus:ring-indigo-600"
                                            />
                                        </div>
                                        <div className="text-sm leading-6">
                                            <label htmlFor="candidates"
className="font-medium text-gray-900">
                                                Candidates
                                            </label>
                                            <p className="text-gray-
500">Get notified when a candidate applies for a job.</p>
                                        </div>
                                    </div>
                                    <div className="relative flex gap-x-3">
                                        <div className="flex h-6 items-
center">
```

```
                                                <input
                                                    id="offers"
                                                    name="offers"
                                                    type="checkbox"
                                                    className="h-4 w-4 rounded
border-gray-300 text-indigo-600 focus:ring-indigo-600"
                                                />
                                            </div>
                                            <div className="text-sm leading-6">
                                                <label htmlFor="offers"
className="font-medium text-gray-900">
                                                    Offers
                                                </label>
                                                <p className="text-gray-
500">Get notified when a candidate accepts or rejects an offer.</p>
                                            </div>
                                        </div>
                                    </div>
                                </fieldset>
                                <fieldset>
                                    <legend className="text-sm font-semibold
leading-6 text-gray-900">Push Notifications</legend>
                                    <p className="mt-1 text-sm leading-6 text-
gray-600">These are delivered via SMS to your mobile phone.</p>
                                    <div className="mt-6 space-y-6">
                                        <div className="flex items-center gap-
x-3">
                                            <input
                                                id="push-everything"
                                                name="push-notifications"
                                                type="radio"
```

```
                                            className="h-4 w-4 border-gray-
300 text-indigo-600 focus:ring-indigo-600"
                                        />
                                        <label htmlFor="push-everything"
className="block text-sm font-medium leading-6 text-gray-900">
                                            Everything
                                        </label>
                                    </div>
                                    <div className="flex items-center gap-
x-3">
                                        <input
                                            id="push-email"
                                            name="push-notifications"
                                            type="radio"
                                            className="h-4 w-4 border-gray-
300 text-indigo-600 focus:ring-indigo-600"
                                        />
                                        <label htmlFor="push-email"
className="block text-sm font-medium leading-6 text-gray-900">
                                            Same as email
                                        </label>
                                    </div>
                                    <div className="flex items-center gap-
x-3">
                                        <input
                                            id="push-nothing"
                                            name="push-notifications"
                                            type="radio"
                                            className="h-4 w-4 border-gray-
300 text-indigo-600 focus:ring-indigo-600"
                                        />
```

```
                                        <label htmlFor="push-nothing"
className="block text-sm font-medium leading-6 text-gray-900">
                                            No push notifications
                                        </label>
                                    </div>
                                </div>
                            </fieldset>
                        </div>
                    </div>
                </div>

                <div className="mt-6 flex items-center justify-end gap-x-
6">
                    <button type="button" className="text-sm font-semibold
leading-6 text-gray-900" onClick={() => {
                        navigate('/admin');
                    }}>
                        Cancel
                    </button>
                    {selectedProduct && !selectedProduct.deleted && <button
                        onClick={(e) => {
                            e.preventDefault();
                            setOpenModal(true);
                        }}
                        className="rounded-md bg-red-600 px-3 py-2 text-sm
font-semibold text-white shadow-sm hover:bg-red-500 focus-visible:outline
focus-visible:outline-2 focus-visible:outline-offset-2 focus-
visible:outline-red-600"
                    >
                        Delete
                    </button>}
```

```
                    {selectedProduct && <Modal
                        title={'Delete product'}
                        message={`Are you sure you want to delete
${selectedProduct.title}?`}
                        dangerOption='Delete'
                        cancelOption='Cancel'
                        dangerAction={handleDelete}
                        cancelAction={(e) => setOpenModal(null)}
                        showModal={openModal}
                    />}

                    <button
                        type="submit"
                        className="rounded-md bg-indigo-600 px-3 py-2 text-
sm font-semibold text-white shadow-sm hover:bg-indigo-500 focus-
visible:outline focus-visible:outline-2 focus-visible:outline-offset-2
focus-visible:outline-indigo-600"
                    >
                        Save
                    </button>
                </div>
            </form>}
        </div>
    )
}

export default ProductForm;
```

ForgotPassword.js

```
import React from 'react';
import { useForm } from "react-hook-form";
import { Link } from "react-router-dom";
import { useDispatch, useSelector } from 'react-redux';
import { resetPasswordRequestAsync, selectAuthError, selectMailSent } from
'../authSlice';

export default function ForgotPassword() {
    const { register, handleSubmit, formState: { errors } } = useForm();
    const dispatch = useDispatch();
    const mailSent = useSelector(selectMailSent);

    const error = useSelector(selectAuthError);

    return (
```

```
        <>
            <div className='h-screen w-full bg-white'>
                <div className="flex min-h-full flex-col justify-center px-
6 py-12 lg:px-8">
                    <div className="sm:mx-auto sm:w-full sm:max-w-sm">
                        <img className="mx-auto h-10 w-auto" src="/navbar-
icon.png" alt="Your Company" />
                        <h2 className="mt-10 text-center text-2xl font-bold
leading-9 tracking-tight text-gray-900">Enter your email address to reset
password</h2>
                    </div>

                    <div className="mt-10 sm:mx-auto sm:w-full sm:max-w-
sm">
                        <form noValidate className="space-y-6" action="#"
method="POST" onSubmit={handleSubmit((data) => {

dispatch(resetPasswordRequestAsync(data.email));
                        })}>
                            <div>
                                <label htmlFor="email" className="block
text-sm font-medium leading-6 text-gray-900">Email address</label>
                                <div className="mt-2">
                                    <input
                                        id="email"
                                        {...register('email',
                                            {
                                                required: "Please enter
your email address",
                                                // eslint-disable-next-line
```

```
                                                    pattern: { value: /\b[\w\.-
]+@[\w\.-]+\.\w{2,4}\b/gi, message: 'Please enter a valid email address' }
                                                }
                                            )}
                                            type="email"
                                            className="block w-full rounded-md
border-0 py-1.5 text-gray-900 shadow-sm ring-1 ring-inset ring-gray-300
placeholder:text-gray-400 focus:ring-2 focus:ring-inset focus:ring-indigo-
600 sm:text-sm sm:leading-6"
                                        />
                                        {errors.email && <p className='text-
red-500 text-xs'>{errors.email.message}</p>}
                                        {mailSent && <p className='text-green-
500 text-xs'>Mail sent successfully</p>}
                                        {error && <p className='text-green-500
text-xs'>Mail sent successfully</p>}
                                    </div>
                                </div>

                                <div>
                                    <button type="submit" className="flex w-
full justify-center rounded-md bg-indigo-600 px-3 py-1.5 text-sm font-
semibold leading-6 text-white shadow-sm hover:bg-indigo-500 focus-
visible:outline focus-visible:outline-2 focus-visible:outline-offset-2
focus-visible:outline-indigo-600">Send email</button>
                                </div>
                            </form>

                            <p className="mt-10 text-center text-sm text-gray-
500">
                                Want to login instead?{' '}
```

```
                            <Link to="/login" className="font-semibold
leading-6 text-indigo-600 hover:text-indigo-500">Log in</Link>
                        </p>
                    </div>
                </div>
            </div>
        </>
    );
}
```

Login.js

```
import React from 'react';
import { useDispatch, useSelector } from 'react-redux';
import { useForm } from "react-hook-form";
import { Link, Navigate } from "react-router-dom";
import { loginUserAsync, selectAuthError, selectLoggedInUser } from
'../authSlice';

export default function Login() {
    const dispatch = useDispatch();

    const { register, handleSubmit, formState: { errors } } = useForm();
    console.log({ errors });

    const error = useSelector(selectAuthError);
    const user = useSelector(selectLoggedInUser);

    return (
        <>
```

```
            {user && <Navigate to='/' replace={true}></Navigate>}
            <div className='h-screen w-full bg-white'>
                <div className="flex min-h-full flex-col justify-center px-
6 py-12 lg:px-8">
                    <div className="sm:mx-auto sm:w-full sm:max-w-sm">
                        <img className="mx-auto h-16 w-auto" src="/navbar-
icon.png" alt="Your Company" />
                        <h2 className="mt-2 text-center text-2xl font-bold
leading-9 tracking-tight text-gray-900">Sign in to your account</h2>
                    </div>

                    <div className="mt-10 sm:mx-auto sm:w-full sm:max-w-
sm">
                        <form noValidate className="space-y-6" action="#"
method="POST" onSubmit={handleSubmit((data) => {
                            dispatch(loginUserAsync({ email: data.email,
password: data.password }));
                        })}>
                            <div>
                                <label htmlFor="email" className="block
text-sm font-medium leading-6 text-gray-900">Email address</label>
                                <div className="mt-2">
                                    <input
                                        id="email"
                                        {...register('email',
                                            {
                                                required: "Please enter
your email address",
                                                // eslint-disable-next-line
                                                pattern: { value: /\b[\w\.-
]+@[\w\.-]+\.\w{2,4}\b/gi, message: 'Please enter a valid email address' }
```

```
                                        }
                                    )}
                                    type="email"
                                    className="block w-full rounded-md
border-0 py-1.5 text-gray-900 shadow-sm ring-1 ring-inset ring-gray-300
placeholder:text-gray-400 focus:ring-2 focus:ring-inset focus:ring-indigo-
600 sm:text-sm sm:leading-6"
                                />
                                {errors.email && <p className='text-
red-500 text-xs'>{errors.email.message}</p>}
                            </div>
                        </div>

                        <div>
                            <div className="flex items-center justify-
between">
                                <label htmlFor="password"
className="block text-sm font-medium leading-6 text-gray-
900">Password</label>
                                <div className="text-sm">
                                    <Link to="/forgot-password"
className="font-semibold text-indigo-600 hover:text-indigo-500">Forgot
password?</Link>
                                </div>
                            </div>
                            <div className="mt-2">
                                <input
                                    id="password"
                                    {...register('password',
                                        {
```

```
                                                required: "Please enter
your password"
                                            }
                                        )}
                                        type="password"
                                        className="block w-full rounded-md
border-0 py-1.5 text-gray-900 shadow-sm ring-1 ring-inset ring-gray-300
placeholder:text-gray-400 focus:ring-2 focus:ring-inset focus:ring-indigo-
600 sm:text-sm sm:leading-6"
                                    />
                                    {errors.password && <p className='text-
red-500 text-xs'>{errors.email.message}</p>}
                                </div>
                                {error && <p className='text-red-500 text-
xs'>{error || error.message}</p>}
                            </div>

                            <div>
                                <button type="submit" className="flex w-
full justify-center rounded-md bg-indigo-600 px-3 py-1.5 text-sm font-
semibold leading-6 text-white shadow-sm hover:bg-indigo-500 focus-
visible:outline focus-visible:outline-2 focus-visible:outline-offset-2
focus-visible:outline-indigo-600">Log in</button>
                            </div>
                        </form>

                        <p className="mt-10 text-center text-sm text-gray-
500">
                            Does not have an account?{' '}
                            <Link to="/signup" className="font-semibold
leading-6 text-indigo-600 hover:text-indigo-500">Sign up</Link>
```

```
                        </p>
                    </div>
                </div>
            </div>
        </>
    );
}
```

Logout.js

```
import React, { useEffect } from 'react'
import { useDispatch, useSelector } from 'react-redux';
import { selectLoggedInUser, signOutAsync } from '../authSlice';
import { Navigate } from 'react-router-dom';

export default function Logout() {
    const dispatch = useDispatch();
    const user = useSelector(selectLoggedInUser);

    useEffect(() => {
        dispatch(signOutAsync());
    }, [dispatch]);

    return (
        <>
            {!user && <Navigate to='/login' replace={true} />}
        </>
    )
}
```

Protected.js

```
import { useSelector } from 'react-redux';
import { selectLoggedInUser } from '../authSlice';
import { Navigate } from 'react-router-dom';
export default function Protected({ children }) {
    const user = useSelector(selectLoggedInUser);
    if(!user) {
        return <Navigate to='/login' replace={true} />
    }
    return children;
}
```

ProtectedAdmin.js

```
import { useSelector } from 'react-redux';
import { selectLoggedInUser } from '../authSlice';
import { Navigate } from 'react-router-dom';
import { selectUserInfo } from '../../user/userSlice';
export default function ProtectedAdmin({ children }) {
    const user = useSelector(selectLoggedInUser);
    const userInfo = useSelector(selectUserInfo)
    if(!user) {
        return <Navigate to='/login' replace={true} />
    }
    if(user && userInfo && userInfo.role !== 'admin') {
        return <Navigate to='/' replace={true} />
    }
    return children;
}
```

ResetPassword.js

```
import React from 'react';
import { useForm } from "react-hook-form";
import { Link } from "react-router-dom";
import { useDispatch, useSelector } from 'react-redux';
import { resetPasswordAsync, selectAuthError, selectPasswordReset } from
'../authSlice';

export default function ResetPassword() {
    const { register, handleSubmit, formState: { errors } } = useForm();
    const dispatch = useDispatch();
    const passwordReset = useSelector(selectPasswordReset);

    const query = new URLSearchParams(window.location.search);
    const token = query.get('token');
    const email = query.get('email');

    const error = useSelector(selectAuthError);

    return (
        <>
            {(email && token) ? <div className='h-screen w-full bg-white'>
                <div className="flex min-h-full flex-col justify-center px-
6 py-12 lg:px-8">
                    <div className="sm:mx-auto sm:w-full sm:max-w-sm">
                        <img className="mx-auto h-10 w-auto" src="/navbar-
icon.png" alt="Your Company" />
                        <h2 className="mt-10 text-center text-2xl font-bold
leading-9 tracking-tight text-gray-900">Reset your password</h2>
                    </div>
```

```
                    <div className="mt-10 sm:mx-auto sm:w-full sm:max-w-
sm">
                        <form noValidate className="space-y-6" action="#"
method="POST" onSubmit={handleSubmit((data) => {
                            dispatch(resetPasswordAsync({email, token,
password: data.password}));
                        })}>
                            <div>
                                <div className="flex items-center justify-
between">
                                    <label htmlFor="password"
className="block text-sm font-medium leading-6 text-gray-900">New
password</label>
                                </div>
                                <div className="mt-2">
                                    <input
                                        id="password"
                                        {...register('password',
                                            {
                                                required: "Please enter a
password",
                                                pattern: { value:
/^(?=.*\d)(?=.*[a-z])(?=.*[A-Z])(?=.*[a-zA-Z]).{8,}$/gm, message: '->
Password must contain at least 8 characters\n-> Password must contain at
least 1 uppercase letter, 1 lowercase letter, and 1 number\nPassword may or
may not contain special characters' }
                                            }
                                        )}
                                        type="password"
```

```
                                        className="block w-full rounded-md
border-0 py-1.5 text-gray-900 shadow-sm ring-1 ring-inset ring-gray-300
placeholder:text-gray-400 focus:ring-2 focus:ring-inset focus:ring-indigo-
600 sm:text-sm sm:leading-6"
                                    />
                                    {errors.password && <p className='text-
red-500 text-xs'>{errors.password.message}</p>}
                                </div>
                            </div>


                            <div>
                                <div className="flex items-center justify-
between">
                                    <label htmlFor="password"
className="block text-sm font-medium leading-6 text-gray-900">Confirm new
password</label>
                                </div>
                                <div className="mt-2">
                                    <input
                                        id="confirmPassword"
                                        {...register('confirmPassword',
                                            {
                                                required: "Please re-enter
enter the password",
                                                validate: (value,
formValues) => value === formValues.password || 'Both passwords do not
match'
                                            }
                                        )}
                                        type="password"
```

```
                                        className="block w-full rounded-md
border-0 py-1.5 text-gray-900 shadow-sm ring-1 ring-inset ring-gray-300
placeholder:text-gray-400 focus:ring-2 focus:ring-inset focus:ring-indigo-
600 sm:text-sm sm:leading-6"
                                    />
                                    {errors.confirmPassword && <p
className='text-red-500 text-xs'>{errors.confirmPassword.message}</p>}
                                    {passwordReset && <p className='text-
green-500 text-xs'>Password reset successfully</p>}
                                    {error && <p className='text-red-500
text-xs'>{error}</p>}
                                </div>
                            </div>

                            <div>
                                <button type="submit" className="flex w-
full justify-center rounded-md bg-indigo-600 px-3 py-1.5 text-sm font-
semibold leading-6 text-white shadow-sm hover:bg-indigo-500 focus-
visible:outline focus-visible:outline-2 focus-visible:outline-offset-2
focus-visible:outline-indigo-600">Reset password</button>
                            </div>
                        </form>

                        <p className="mt-10 text-center text-sm text-gray-
500">
                            Want to login instead?{' '}
                            <Link to="/login" className="font-semibold
leading-6 text-indigo-600 hover:text-indigo-500">Log in</Link>
                        </p>
                    </div>
                </div>
```

```
            </div> :
            <p>Link was expired</p>}
        </>
    );
}
```

Signup.js

```
import React from 'react';
import { useDispatch, useSelector } from "react-redux";
import { useForm } from "react-hook-form";
import { createUserAsync, selectLoggedInUser } from '../authSlice';
import { Link, Navigate } from "react-router-dom";

export default function Signup() {
    const dispatch = useDispatch();

    const { register, handleSubmit, formState: { errors } } = useForm();
    console.log({ errors });

    const user = useSelector(selectLoggedInUser);

    return (
        <>
            {user && <Navigate to='/' replace={true}></Navigate>}
            <div className='h-screen w-full bg-white'>
                <div className="flex min-h-full flex-col justify-center px-
6 py-12 lg:px-8">
                    <div className="sm:mx-auto sm:w-full sm:max-w-sm">
```

```
                        <img className="mx-auto h-16 w-auto" src="/navbar-
icon.png" alt="Your Company" />
                        <h2 className="mt-2 text-center text-2xl font-bold
leading-9 tracking-tight text-gray-900">Create an account</h2>
                    </div>

                    <div className="mt-10 sm:mx-auto sm:w-full sm:max-w-
sm">
                        <form noValidate className="space-y-6" action="#"
method="POST" onSubmit={handleSubmit((data) => {
                            dispatch(createUserAsync({ email: data.email,
password: data.password, addresses: [], role: 'user' }));
                        })}>
                            <div>
                                <label htmlFor="email" className="block
text-sm font-medium leading-6 text-gray-900">Email address</label>
                                <div className="mt-2">
                                    <input
                                        id="email"
                                        {...register('email',
                                            {
                                                required: "Please enter an
email address",
                                                // eslint-disable-next-line
                                                pattern: { value: /\b[\w\.-
]+@[\w\.-]+\.\w{2,4}\b/gi, message: 'Please enter a valid email address' }
                                            }
                                        )}
                                        type="email"
                                        className="block w-full rounded-md
border-0 py-1.5 text-gray-900 shadow-sm ring-1 ring-inset ring-gray-300
```

```
placeholder:text-gray-400 focus:ring-2 focus:ring-inset focus:ring-indigo-
600 sm:text-sm sm:leading-6"
                                    />
                                    {errors.email && <p className='text-
red-500 text-xs'>{errors.email.message}</p>}
                                </div>
                            </div>


                            <div>
                                <div className="flex items-center justify-
between">
                                    <label htmlFor="password"
className="block text-sm font-medium leading-6 text-gray-
900">Password</label>
                                </div>
                                <div className="mt-2">
                                    <input
                                        id="password"
                                        {...register('password',
                                            {
                                                required: "Please enter a
password",
                                                pattern: { value:
/^(?=.*\d)(?=.*[a-z])(?=.*[A-Z])(?=.*[a-zA-Z]).{8,}$/gm, message: '->
Password must contain at least 8 characters\n-> Password must contain at
least 1 uppercase letter, 1 lowercase letter, and 1 number\nPassword may or
may not contain special characters' }
                                            }
                                        )}
                                        type="password"
```

```
                                        className="block w-full rounded-md
border-0 py-1.5 text-gray-900 shadow-sm ring-1 ring-inset ring-gray-300
placeholder:text-gray-400 focus:ring-2 focus:ring-inset focus:ring-indigo-
600 sm:text-sm sm:leading-6"
                                    />
                                    {errors.password && <p className='text-
red-500 text-xs'>{errors.password.message}</p>}
                                </div>
                            </div>


                            <div>
                                <div className="flex items-center justify-
between">
                                    <label htmlFor="password"
className="block text-sm font-medium leading-6 text-gray-900">Confirm
Password</label>
                                </div>
                                <div className="mt-2">
                                    <input
                                        id="confirmPassword"
                                        {...register('confirmPassword',
                                            {
                                                required: "Please re-enter
enter the password",
                                                validate: (value,
formValues) => value === formValues.password || 'Both passwords do not
match'
                                            }
                                        )}
                                        type="password"
```

```
                                        className="block w-full rounded-md
border-0 py-1.5 text-gray-900 shadow-sm ring-1 ring-inset ring-gray-300
placeholder:text-gray-400 focus:ring-2 focus:ring-inset focus:ring-indigo-
600 sm:text-sm sm:leading-6"
                                    />
                                    {errors.confirmPassword && <p
className='text-red-500 text-xs'>{errors.confirmPassword.message}</p>}
                                </div>
                            </div>

                            <div>
                                <button type="submit" className="flex w-
full justify-center rounded-md bg-indigo-600 px-3 py-1.5 text-sm font-
semibold leading-6 text-white shadow-sm hover:bg-indigo-500 focus-
visible:outline focus-visible:outline-2 focus-visible:outline-offset-2
focus-visible:outline-indigo-600">Sign Up</button>
                            </div>
                        </form>

                        <p className="mt-10 text-center text-sm text-gray-
500">
                            Already have an account?{' '}
                            <Link to="/login" className="font-semibold
leading-6 text-indigo-600 hover:text-indigo-500">Log in</Link>
                        </p>
                    </div>
                </div>
            </div>
        </>
    );
}
```

authAPI.js

```
export function createUser(userData) {
  return new Promise(async (resolve) => {
    const response = await fetch('/auth/signup', {
      method: 'POST',
      body: JSON.stringify(userData),
      headers: { 'content-type': 'application/json' }
    });
    const data = await response.json();
    resolve({ data });
  });
}

export function loginUser(loginInfo) {
  return new Promise(async (resolve, reject) => {
    try {
      const response = await fetch('/auth/login', {
        method: 'POST',
        body: JSON.stringify(loginInfo),
        headers: { 'content-type': 'application/json' }
      });

      if(response.ok) {
        const data = await response.json();
        resolve({ data });
      }
      else {
        const error = await response.text();
        reject(error);
      }
```

```
    }
    catch (error) {
      reject(error);
    }
  });
}

export function checkAuth() {
  return new Promise(async (resolve, reject) => {
    try {
      const response = await fetch('/auth/check');

      if(response.ok) {
        const data = await response.json();
        resolve({ data });
      }
      else {
        const error = await response.text();
        reject(error);
      }
    }
    catch (error) {
      reject(error);
    }
  });
}

export function signOut() {
  return new Promise(async (resolve, reject) => {
    try {
      const response = await fetch('/auth/logout');
```

```
      if(response.ok) {
        resolve({ data: 'success' });
      }
      else {
        const error = await response.text();
        reject(error);
      }
    }
    catch (error) {
      reject(error);
    }
  });
}

export function resetPasswordRequest(email) {
  return new Promise(async (resolve, reject) => {
    try {
      const response = await fetch('/auth/reset-password-request', {
        method: 'POST',
        body: JSON.stringify({email}),
        headers: { 'content-type': 'application/json' }
      });

      if(response.ok) {
        const data = await response.json();
        resolve({ data });
      }
      else {
        const error = await response.text();
        reject(error);
```

```
      }
    }
    catch (error) {
      reject(error);
    }
  });
}

export function resetPassword(data) {
  return new Promise(async (resolve, reject) => {
    try {
      const response = await fetch('/auth/reset-password', {
        method: 'POST',
        body: JSON.stringify(data),
        headers: { 'content-type': 'application/json' }
      });

      if(response.ok) {
        const data = await response.json();
        resolve({ data });
      }
      else {
        const error = await response.text();
        reject(error);
      }
    }
    catch (error) {
      reject(error);
    }
  });
}
```

authSlice.js

```
import { createAsyncThunk, createSlice } from '@reduxjs/toolkit';
import { loginUser, createUser, signOut, checkAuth, resetPasswordRequest,
resetPassword } from './authAPI';
import { toast } from 'react-toastify';

const initialState = {
  loggedInUserToken: null,
  status: 'idle',
  error: null,
  userChecked: false,
  mailSent: false,
  passwordReset: false
};

export const createUserAsync = createAsyncThunk(
  'user/createUser',
  async (userData) => {
    const response = await createUser(userData);
    if (response) {
      toast.success('Signed up successfully!', {
        position: "bottom-left",
        autoClose: 2000,
        hideProgressBar: false,
        closeOnClick: true,
        pauseOnHover: true,
        draggable: true,
        progress: undefined,
        theme: "colored"
      });
```

```
    }
    return response.data;
  }
);

export const loginUserAsync = createAsyncThunk(
  'user/loginUser',
  async (loginInfo, { rejectWithValue }) => {
    try {
      const response = await loginUser(loginInfo);
      if (response) {
        toast.success('Logged in successfully!', {
          position: "bottom-left",
          autoClose: 2000,
          hideProgressBar: false,
          closeOnClick: true,
          pauseOnHover: true,
          draggable: true,
          progress: undefined,
          theme: "colored"
        });
      }
      return response.data;
    }
    catch (error) {
      console.log(error);
      if (error) {
        toast.error('Invalid credentials!', {
          position: "bottom-left",
          autoClose: 2000,
          hideProgressBar: false,
```

```
          closeOnClick: true,
          pauseOnHover: true,
          draggable: true,
          progress: undefined,
          theme: "colored"
        });
      }
      return rejectWithValue(error);
    }
  }
);

export const checkAuthAsync = createAsyncThunk(
  'user/checkAuth',
  async () => {
    try {
      const response = await checkAuth();
      return response.data;
    }
    catch (error) {
      console.log(error);
    }
  }
);

export const resetPasswordRequestAsync = createAsyncThunk(
  'user/resetPasswordRequest',
  async (email, { rejectWithValue }) => {
    try {
      const response = await resetPasswordRequest(email);
      if (response) {
```

```
        toast.success('Mail sent successfully!', {
          position: "bottom-left",
          autoClose: 2000,
          hideProgressBar: false,
          closeOnClick: true,
          pauseOnHover: true,
          draggable: true,
          progress: undefined,
          theme: "colored"
        });
      }
      return response.data;
    }
    catch (error) {
      console.log(error);
      if (error) {
        toast.error('An unknown error occurred!', {
          position: "bottom-left",
          autoClose: 2000,
          hideProgressBar: false,
          closeOnClick: true,
          pauseOnHover: true,
          draggable: true,
          progress: undefined,
          theme: "colored"
        });
      }
      return rejectWithValue(error);
    }
  }
);
```

```
export const resetPasswordAsync = createAsyncThunk(
  'user/resetPassword',
  async (data, { rejectWithValue }) => {
    try {
      const response = await resetPassword(data);
      if (response) {
        toast.success('Password reset successfully!', {
          position: "bottom-left",
          autoClose: 2000,
          hideProgressBar: false,
          closeOnClick: true,
          pauseOnHover: true,
          draggable: true,
          progress: undefined,
          theme: "colored"
        });
      }
      return response.data;
    }
    catch (error) {
      console.log(error);
      if (error) {
        toast.error('An unknown error occurred!', {
          position: "bottom-left",
          autoClose: 2000,
          hideProgressBar: false,
          closeOnClick: true,
          pauseOnHover: true,
          draggable: true,
          progress: undefined,
```

```
          theme: "colored"
        });
      }
      return rejectWithValue(error);
    }
  }
);

export const signOutAsync = createAsyncThunk(
  'user/signOut',
  async () => {
    const response = await signOut();
    toast.success('Logged out successfully!', {
      position: "bottom-left",
      autoClose: 2000,
      hideProgressBar: false,
      closeOnClick: true,
      pauseOnHover: true,
      draggable: true,
      progress: undefined,
      theme: "colored"
    });
    return response.data;
  }
);

export const authSlice = createSlice({
  name: 'user',
  initialState,
  reducers: {
    increment: (state) => {
```

```
    state.value += 1;
  }
},
extraReducers: (builder) => {
  builder
    .addCase(createUserAsync.pending, (state) => {
      state.status = 'loading';
    })
    .addCase(createUserAsync.fulfilled, (state, action) => {
      state.status = 'idle';
      state.loggedInUserToken = action.payload;
    })
    .addCase(loginUserAsync.pending, (state) => {
      state.status = 'loading';
    })
    .addCase(loginUserAsync.fulfilled, (state, action) => {
      state.status = 'idle';
      state.loggedInUserToken = action.payload;
    })
    .addCase(loginUserAsync.rejected, (state, action) => {
      state.status = 'idle';
      state.error = action.payload;
    })
    .addCase(signOutAsync.pending, (state) => {
      state.status = 'loading';
    })
    .addCase(signOutAsync.fulfilled, (state, action) => {
      state.status = 'idle';
      state.loggedInUserToken = null;
    })
    .addCase(checkAuthAsync.pending, (state) => {
```

```
        state.status = 'loading';
      })
      .addCase(checkAuthAsync.fulfilled, (state, action) => {
        state.status = 'idle';
        state.loggedInUserToken = action.payload;
        state.userChecked = true;
      })
      .addCase(checkAuthAsync.rejected, (state, action) => {
        state.status = 'idle';
        state.userChecked = true;
      })
      .addCase(resetPasswordRequestAsync.pending, (state) => {
        state.status = 'loading';
      })
      .addCase(resetPasswordRequestAsync.fulfilled, (state, action) => {
        state.status = 'idle';
        state.mailSent = true;
      })
      .addCase(resetPasswordAsync.pending, (state) => {
        state.status = 'loading';
      })
      .addCase(resetPasswordAsync.fulfilled, (state, action) => {
        state.status = 'idle';
        state.passwordReset = true;
      })
      .addCase(resetPasswordAsync.rejected, (state, action) => {
        state.status = 'idle';
        state.error = action.payload;
      });
  },
});
```

```
export const { increment } = authSlice.actions;

export const selectLoggedInUser = (state) => state.auth.loggedInUserToken;
export const selectAuthError = (state) => state.auth.error;
export const selectUserChecked = (state) => state.auth.userChecked;
export const selectMailSent = (state) => state.auth.mailSent;
export const selectPasswordReset = (state) => state.auth.passwordReset;

export default authSlice.reducer;
```

Cart.js

```
import React, { useState } from 'react';
import { useDispatch, useSelector } from 'react-redux';
import { Link, Navigate } from 'react-router-dom';
import { deleteItemFromCartAsync, selectCartItemsStatus, selectCartLoaded,
selectItems, updateCartAsync } from './cartSlice';
import { MutatingDots } from 'react-loader-spinner';
import Modal from '../common/Modal';

export default function Cart() {
  const dispatch = useDispatch();
```

```
  // eslint-disable-next-line no-unused-vars
  const [open, setOpen] = useState(true);
  const [openModal, setOpenModal] = useState(null);

  const loadingStatus = useSelector(selectCartItemsStatus);
  const cartLoaded = useSelector(selectCartLoaded);

  const items = useSelector(selectItems);
  const totalAmount = items.reduce((amount, item) => {
    let itemPrice = item.product.discountedPrice;
    return itemPrice * item.quantity + amount;
  }, 0);
  const totalItems = items.reduce((total, item) => item.quantity + total,
0);

  const handleQuantity = (e, item) => {
    dispatch(updateCartAsync({ id: item.id, quantity: e.target.value }));
  }

  const handleRemove = (e, itemId) => {
    dispatch(deleteItemFromCartAsync(itemId));
  }

  return (
    <>
      {!items.length && cartLoaded && <Navigate to='/' replace={true} />}
      <div className='mx-auto my-24 max-w-7xl px-4 sm:px-6 lg:px-8 bg-white
rounded-lg'>

        {loadingStatus === 'loading' &&
```

```
          <div className='w-full flex justify-center items-center'>
            <MutatingDots
              visible={true}
              height="100"
              width="100"
              color="#4464f2"
              secondaryColor="#4464f2"
              radius="12.5"
              ariaLabel="mutating-dots-loading"
              wrapperStyle={{}}
              wrapperClass=""
            />'
          </div>}

        <h2 className='text-4xl font-bold tracking-tight text-gray-900 mb-5
pt-5 text-center'>Cart</h2>
        <div className="border-t border-gray-200 px-4 py-6 sm:px-6">
          <div className="flow-root">
            <ul className="-my-6 divide-y divide-gray-200">
              {items.map((item) => (
                <li key={item.id} className="flex py-6">
                  <div className="h-24 w-24 flex-shrink-0 overflow-hidden
rounded-md border border-gray-200">
                    <img
                      src={item.product.thumbnail}
                      alt={item.product.title}
                      className="h-full w-full object-cover object-center"
                    />
                  </div>

                  <div className="ml-4 flex flex-1 flex-col">
```

```
                    <div>
                      <div className="flex justify-between text-base font-
medium text-gray-900">
                        <h3>
                          <a
href={item.product.thumbnail}>{item.product.title}</a>
                        </h3>
                        <p className="ml-
4">${item.product.discountedPrice}</p>
                      </div>
                      <p className="mt-1 text-sm text-gray-
500">{item.product.brand[0].toUpperCase() +
item.product.brand.slice(1)}</p>
                    </div>
                    <div className="flex flex-1 items-end justify-between
text-sm">
                      <div className="text-gray-500">
                        <label htmlFor="quantity" className="inline mr-5
text-sm font-medium leading-6 text-gray-900">Qty {item.quantity}</label>
                        <select name="quantity" id="quantity"
className='text-sm' onChange={(e) => handleQuantity(e, item)}
value={item.quantity}>
                          <option value="1">1</option>
                          <option value="2">2</option>
                          <option value="3">3</option>
                          <option value="4">4</option>
                          <option value="5">5</option>
                        </select>
                      </div>

                      <div className="flex">
```

```
                        <Modal
                          title={'Remove item'}
                          message={`Are you sure you want to remove
${item.product.title} from the cart?`}
                          dangerOption='Remove'
                          cancelOption='Cancel'
                          dangerAction={(e) => handleRemove(e, item.id)}
                          cancelAction={(e) => setOpenModal(-1)}
                          showModal={openModal === item.id}
                        />

                        <button
                          onClick={(e) => setOpenModal(item.id)}
                          type="button"
                          className="font-medium text-indigo-400
hover:text-indigo-600"
                        >
                          Remove
                        </button>
                      </div>
                    </div>
                  </div>
                </li>
              ))}
            </ul>
          </div>
        </div>

        <div className="border-t border-gray-200 px-4 py-6 sm:px-6">
          <div className="flex justify-between text-base font-medium text-
gray-900 my-2">
```

```
            <p>Subtotal</p>
            <p>${totalAmount}</p>
          </div>
          <div className="flex justify-between text-base font-medium text-
gray-900 my-2">
            <p>Total Items in Cart</p>
            <p>{totalItems}</p>
          </div>
          <p className="mt-0.5 text-sm text-gray-500">Shipping and taxes
calculated at checkout.</p>
          <div className="mt-6">
            <Link to="/checkout"
              href="/"
              className="flex items-center justify-center rounded-md border
border-transparent bg-indigo-600 px-6 py-3 text-base font-medium text-white
shadow-sm hover:bg-indigo-700"
            >
              Checkout
            </Link>
          </div>
          <div className="mt-6 flex justify-center text-center text-sm
text-gray-500">
            <p>
              or{' '}
              <Link to="/">
                <button
                  type="button"
                  className="font-medium text-indigo-600 hover:text-indigo-
500"
                  onClick={() => setOpen(false)}
                >
```

```
                  Continue Shopping
                  <span aria-hidden="true"> &rarr;</span>
                </button>
              </Link>
            </p>
          </div>
        </div>
      </div>
    </>
  );
}
```

cartAPI.js

```
export function addToCart(item) {
  return new Promise(async (resolve) => {
    const response = await fetch('/cart', {
      method: 'POST',
      body: JSON.stringify(item),
      headers: { 'content-type': 'application/json' }
    });
    const data = await response.json();
    resolve({ data });
  });
}

export function fetchItemsByUserId() {
  return new Promise(async (resolve) => {
    const response = await fetch('/cart');
    const data = await response.json();
```

```
    resolve({ data });
  });
}

export function updateCart(update) {
  return new Promise(async (resolve) => {
    const response = await fetch('/cart/' + update.id, {
      method: 'PATCH',
      body: JSON.stringify(update),
      headers: { 'content-type': 'application/json' }
    });
    const data = await response.json();
    resolve({ data });
  });
}

export function deleteItemFromCart(itemId) {
  return new Promise(async (resolve) => {
    const response = await fetch('/cart/' + itemId, {
      method: 'DELETE',
      headers: { 'content-type': 'application/json' }
    });
    const data = await response.json();
    console.log(data);
    resolve({ data: { id: itemId } });
  });
}

export async function resetCart() {
  return new Promise(async (resolve) => {
    const response = await fetchItemsByUserId();
```

```
    const items = response.data;
    for (let item of items) {
      await deleteItemFromCart(item.id);
    }
    resolve({status: 'success'});
  });
}
```

cartSlice.js

```
import { createAsyncThunk, createSlice } from '@reduxjs/toolkit';
import { addToCart, deleteItemFromCart, fetchItemsByUserId, resetCart,
updateCart } from './cartAPI';
import { toast } from 'react-toastify';

const initialState = {
  status: 'idle',
  items: [],
  cartLoaded: false
};

export const addToCartAsync = createAsyncThunk(
  'cart/addToCart',
  async (item) => {
    const response = await addToCart(item);
    toast.success('Item added to cart!', {
      position: "bottom-right",
      autoClose: 2000,
      hideProgressBar: false,
      closeOnClick: true,
```

```
      pauseOnHover: true,
      draggable: true,
      progress: undefined,
      theme: "colored"
    });
    return response.data;
  }
);

export const fetchItemsByUserIdAsync = createAsyncThunk(
  'cart/fetchItemsByUserId',
  async () => {
    const response = await fetchItemsByUserId();
    return response.data;
  }
);

export const updateCartAsync = createAsyncThunk(
  'cart/updateCart',
  async (userId) => {
    const response = await updateCart(userId);
    return response.data;
  }
);

export const deleteItemFromCartAsync = createAsyncThunk(
  'cart/deleteItemFromCart',
  async (itemId) => {
    const response = await deleteItemFromCart(itemId);
    toast.info(`Item removed from cart`, {
      position: "bottom-right",
```

```
      autoClose: 2000,
      hideProgressBar: false,
      closeOnClick: true,
      pauseOnHover: true,
      draggable: true,
      progress: undefined,
      theme: "colored"
    });
    return response.data;
  }
);

export const resetCartAsync = createAsyncThunk(
  'cart/resetCart',
  async () => {
    const response = await resetCart();
    return response.data;
  }
);

export const cartSlice = createSlice({
  name: 'cart',
  initialState,
  reducers: {
    increment: (state) => {
      state.value += 1;
    }
  },
  extraReducers: (builder) => {
    builder
      .addCase(addToCartAsync.pending, (state) => {
```

```
        state.status = 'loading';
      })
      .addCase(addToCartAsync.fulfilled, (state, action) => {
        state.status = 'idle';
        state.items.push(action.payload);
      })
      .addCase(fetchItemsByUserIdAsync.pending, (state) => {
        state.status = 'loading';
      })
      .addCase(fetchItemsByUserIdAsync.fulfilled, (state, action) => {
        state.status = 'idle';
        state.items = action.payload;
        state.cartLoaded = true;
      })
      .addCase(fetchItemsByUserIdAsync.rejected, (state, action) => {
        state.status = 'idle';
        state.cartLoaded = true;
      })
      .addCase(updateCartAsync.pending, (state) => {
        state.status = 'loading';
      })
      .addCase(updateCartAsync.fulfilled, (state, action) => {
        state.status = 'idle';
        const index = state.items.findIndex(item => item.id ===
action.payload.id);
        state.items[index] = action.payload;
      })
      .addCase(deleteItemFromCartAsync.pending, (state) => {
        state.status = 'loading';
      })
      .addCase(deleteItemFromCartAsync.fulfilled, (state, action) => {
```

```
        state.status = 'idle';
        const index = state.items.findIndex(item => item.id ===
action.payload.id);
        state.items.splice(index, 1);
      })
      .addCase(resetCartAsync.pending, (state) => {
        state.status = 'loading';
      })
      .addCase(resetCartAsync.fulfilled, (state, action) => {
        state.status = 'idle';
        state.items = [];
      });
  },
});

export const { increment } = cartSlice.actions;

export const selectItems = (state) => state.cart.items;
export const selectCartItemsStatus = (state) => state.cart.status;
export const selectCartLoaded = (state) => state.cart.cartLoaded;

export default cartSlice.reducer;
```

Footer.js

```
import React from 'react';


function Footer() {
    return (
        <div>
            <div className=" bg-gray-900">
                <div className="max-w-2xl mx-auto text-white py-10">
                    <div className="text-center">
                        <h3 className="text-3xl mb-3"> Vendr </h3>
                        <p> Shop Smart, Shop Easy: Your Ultimate Ecommerce
Destination. </p>


                    </div>
                    <div className="mt-12 flex flex-col md:flex-row
md:justify-between items-center text-sm text-gray-400">
                        <p className="order-2 md:order-1 mt-8 md:mt-0">
                            {" "}
                            © Abhik Gupta, 2024.{" "}
                        </p>
                        <div className="order-1 md:order-2">
                            <span className="px-2">About us</span>
                            <span className="px-2 border-l">Contact
us</span>
                            <span className="px-2 border-l">Privacy
Policy</span>
                        </div>
                    </div>
                </div>
            </div>
```

```
        </div>
    )
}
export default Footer;
```

Modal.js

```
import React, { useRef, useState, Fragment, useEffect } from 'react';
import { Dialog, Transition } from '@headlessui/react';
import { ExclamationTriangleIcon } from '@heroicons/react/24/outline';

function Modal({ title, message, dangerOption, cancelOption, dangerAction,
showModal, cancelAction }) {
    const [open, setOpen] = useState(false)
    const cancelButtonRef = useRef(null);

    const handleDanger = () => {
        setOpen(false);
        dangerAction();
    }

    const handleCancel = () => {
        setOpen(false);
        cancelAction();
    }

    useEffect(() => {
        if (showModal) {
            setOpen(true);
        }
```

```
        else {
            setOpen(false);
        }
    }, [showModal]);

    return (
        <div>
            <Transition.Root show={open} as={Fragment}>
                <Dialog as="div" className="relative z-10"
initialFocus={cancelButtonRef} onClose={setOpen}>
                    <Transition.Child
                        as={Fragment}
                        enter="ease-out duration-300"
                        enterFrom="opacity-0"
                        enterTo="opacity-100"
                        leave="ease-in duration-200"
                        leaveFrom="opacity-100"
                        leaveTo="opacity-0"
                    >
                        <div className="fixed inset-0 bg-gray-500 bg-
opacity-75 transition-opacity" />
                    </Transition.Child>

                    <div className="fixed inset-0 z-10 w-screen overflow-y-
auto">
                        <div className="flex min-h-full items-end justify-
center p-4 text-center sm:items-center sm:p-0">
                            <Transition.Child
                                as={Fragment}
                                enter="ease-out duration-300"
```

```
                                enterFrom="opacity-0 translate-y-4
sm:translate-y-0 sm:scale-95"
                                enterTo="opacity-100 translate-y-0
sm:scale-100"
                                leave="ease-in duration-200"
                                leaveFrom="opacity-100 translate-y-0
sm:scale-100"
                                leaveTo="opacity-0 translate-y-4
sm:translate-y-0 sm:scale-95"
                            >
                                <Dialog.Panel className="relative transform
overflow-hidden rounded-lg bg-white text-left shadow-xl transition-all
sm:my-8 sm:w-full sm:max-w-lg">
                                    <div className="bg-white px-4 pb-4 pt-5
sm:p-6 sm:pb-4">
                                        <div className="sm:flex sm:items-
start">
                                            <div className="mx-auto flex h-
12 w-12 flex-shrink-0 items-center justify-center rounded-full bg-red-100
sm:mx-0 sm:h-10 sm:w-10">
                                                <ExclamationTriangleIcon
className="h-6 w-6 text-red-600" aria-hidden="true" />
                                            </div>
                                            <div className="mt-3 text-
center sm:ml-4 sm:mt-0 sm:text-left">
                                                <Dialog.Title as="h3"
className="text-base font-semibold leading-6 text-gray-900">
                                                    {title}
                                                </Dialog.Title>
                                                <div className="mt-2">
```

```
                                                    <p className="text-sm
text-gray-500">
                                                        {message}
                                                    </p>
                                                </div>
                                            </div>
                                        </div>
                                    </div>
                                    <div className="bg-gray-50 px-4 py-3
sm:flex sm:flex-row-reverse sm:px-6">
                                        <button
                                            type="button"
                                            className="inline-flex w-full
justify-center rounded-md bg-red-600 px-3 py-2 text-sm font-semibold text-
white shadow-sm hover:bg-red-500 sm:ml-3 sm:w-auto"
                                            onClick={handleDanger}
                                        >
                                            {dangerOption}
                                        </button>
                                        <button
                                            type="button"
                                            className="mt-3 inline-flex w-
full justify-center rounded-md bg-white px-3 py-2 text-sm font-semibold
text-gray-900 shadow-sm ring-1 ring-inset ring-gray-300 hover:bg-gray-50
sm:mt-0 sm:w-auto"
                                            onClick={handleCancel}
                                            ref={cancelButtonRef}
                                        >
                                            {cancelOption}
                                        </button>
                                    </div>
```

```
                                </Dialog.Panel>
                            </Transition.Child>
                        </div>
                    </div>
                </Dialog>
            </Transition.Root>
        </div>
    )
}
export default Modal;
```

Pagination.js

```
import { ChevronLeftIcon, ChevronRightIcon } from
"@heroicons/react/24/outline";
import { ITEMS_PER_PAGE } from "../../app/constants";

export default function Pagination({ page, setPage, handlePage, totalItems
}) {
    const totalPages = Math.ceil(totalItems / ITEMS_PER_PAGE);
    return (
        <div className="flex items-center justify-between border-t border-
gray-200 bg-white px-4 py-3 sm:px-6">
            <div className="flex flex-1 justify-between sm:hidden">
                <div
                    onClick={(e) => handlePage(page > 1 ? page - 1 : page)}
                    className="relative inline-flex items-center rounded-md
border border-gray-300 bg-white px-4 py-2 text-sm font-medium text-gray-700
hover:bg-gray-50"
```

```
                >
                    Previous
                </div>
                <div
                    onClick={(e) => handlePage(page < totalPages ? page + 1
: page)}
                    className="relative ml-3 inline-flex items-center
rounded-md border border-gray-300 bg-white px-4 py-2 text-sm font-medium
text-gray-700 hover:bg-gray-50"
                >
                    Next
                </div>
            </div>
            <div className="hidden sm:flex sm:flex-1 sm:items-center
sm:justify-between">
                <div>
                    <p className="text-sm text-gray-700">
                        Showing{' '}
                        <span className="font-medium">{(page - 1) *
ITEMS_PER_PAGE + 1}</span>{' '}
                        to{' '}
                        <span className="font-medium">{page *
ITEMS_PER_PAGE > totalItems ? totalItems : page * ITEMS_PER_PAGE}</span>{'
'}
                        of{' '}
                        <span className="font-medium">{totalItems}</span>{'
'}
                        results
                    </p>
                </div>
                <div>
```

```
                    <nav className="isolate inline-flex -space-x-px
rounded-md shadow-sm" aria-label="Pagination">
                        <div
                            onClick={(e) => handlePage(page > 1 ? page - 1
: page)}
                            className="relative inline-flex items-center
rounded-l-md px-2 py-2 text-gray-400 ring-1 ring-inset ring-gray-300
hover:bg-gray-50 focus:z-20 focus:outline-offset-0 cursor-pointer"
                        >
                            <span className="sr-only">Previous</span>
                            <ChevronLeftIcon className="h-5 w-5" aria-
hidden="true" />
                        </div>

                        {Array.from({ length: totalPages }).map((el, index)
=> (
                            <div
                                key={index}
                                onClick={e => handlePage(index + 1)}
                                aria-current="page"
                                className={`relative z-10 inline-flex
items-center ${page === index + 1 ? 'bg-indigo-600 text-white' : 'bg-white
text-gray-500'} px-4 py-2 text-sm font-semibold focus:z-20 focus-
visible:outline focus-visible:outline-2 focus-visible:outline-offset-2
focus-visible:outline-indigo-600 cursor-pointer border-2`}
                            >
                                {index + 1}
                            </div>
                        )
                        )}
```

```
                        <div
                            onClick={(e) => handlePage(totalPages > page ?
page + 1 : page)}
                            className="relative inline-flex items-center
rounded-r-md px-2 py-2 text-gray-400 ring-1 ring-inset ring-gray-300
hover:bg-gray-50 focus:z-20 focus:outline-offset-0 cursor-pointer"
                        >
                            <span className="sr-only">Next</span>
                            <ChevronRightIcon className="h-5 w-5" aria-
hidden="true" />
                        </div>
                    </nav>
                </div>
            </div>
        </div>
    );
}
```

Navbar.js

```
import React from 'react';
import { Fragment } from 'react';
import { Disclosure, Menu, Transition } from '@headlessui/react';
import { Bars3Icon, XMarkIcon } from '@heroicons/react/24/outline';

import { Link } from "react-router-dom";
```

```
import { useSelector } from 'react-redux';
import { selectItems } from '../cart/cartSlice';
import { selectUserInfo } from '../user/userSlice';

const navigation = [
    { name: 'Products', link: '/', current: true, user: true },
    { name: 'About Us', link: '/about', current: false, user: true },
    { name: 'Contact Us', link: '/contact', current: false, user: true },
    { name: 'Admin Products', link: '/admin', current: false, admin: true
},
    { name: 'Admin Orders', link: '/admin/orders', current: false, admin:
true },
    { name: 'Admin Queries', link: '/admin/queries', current: false, admin:
true }
]
const userNavigation = [
    { name: 'My Profile', link: '/profile' },
    { name: 'My Orders', link: '/my-orders' },
    { name: 'Sign out', link: '/logout' }
]

function classNames(...classes) {
    return classes.filter(Boolean).join(' ')
}

function Navbar({ children }) {
    const items = useSelector(selectItems);
    const userInfo = useSelector(selectUserInfo);

    return (
        <div>
```

```
            {userInfo && <div className="min-h-full">
                <Disclosure as="nav" className="bg-[#111827]">
                    {({ open }) => (
                        <>
                            <div className="mx-auto max-w-7xl px-4 sm:px-6
lg:px-8">
                                <div className="flex h-16 items-center
justify-between">
                                    <div className="flex items-center">
                                        <div className="flex-shrink-0">
                                            <Link to="/">
                                                <img
                                                    className="h-10 w-10"
                                                    src="/navbar-icon.png"
                                                    alt="Your Company"
                                                />
                                            </Link>
                                        </div>
                                        <div className="hidden md:block">
                                            <div className="ml-10 flex
items-baseline space-x-4">
                                                {navigation.map((item) =>
                                                    item[userInfo.role] ? (
                                                        <Link
                                                            key={item.name}
                                                            to={item.link}

className={classNames(

item.current
```

```
                                                                    ? 'bg-
white text-black'
                                                                    :
'text-gray-300 hover:bg-gray-700 hover:text-white',
                                                                        'rounded-md
px-3 py-2 text-sm font-medium'
                                                                )}
                                                                aria-
current={item.current ? 'page' : undefined}
                                                            >
                                                                {item.name}
                                                            </Link>
                                                        ) : null
                                                    )}
                                                </div>
                                            </div>
                                        </div>
                                        <div className="hidden md:block">
                                            <div className="ml-4 flex items-
center md:ml-6">
                                                <Link to="/cart">
                                                    <button
                                                        type="button"
                                                        className="relative
rounded-full bg-gray-800 p-1 text-gray-400 hover:text-white focus:outline-
none focus:ring-2 focus:ring-transparent focus:ring-offset-2 focus:ring-
offset-gray-800"
                                                    >
                                                        <span
className="absolute -inset-1.5" />
```

```
                                                        <span className="sr-
only">View notifications</span>
                                                        {items.length > 0 &&
<span className="inline-flex items-center justify-center gap-2 rounded-md
bg-blue-100 px-3 py-1 font-medium text-[#1F2937] ring-1 ring-inset ring-
green-600/20 outline-none border-none">
                                                            <img src="/cart-
icon.png" className="h-8 w-8" aria-hidden="true" alt="Navbar Image" />
                                                            <p className='text-
md font-bold -mt-3 -ml-1 py-1'>{items.length}</p>
                                                        </span>}
                                                    </button>
                                                </Link>


                                                <Menu as="div"
className="relative ml-3">
                                                    <div>
                                                        <Menu.Button
className="relative flex max-w-xs items-center rounded-full bg-gray-800
text-sm focus:outline-none focus:ring-2 focus:ring-white focus:ring-offset-
2 focus:ring-offset-gray-800">
                                                            <span
className="absolute -inset-1.5" />
                                                            <span
className="sr-only">Open user menu</span>
                                                            <img className="h-
10 w-10 rounded-full" src='/user.png' alt="" />
                                                        </Menu.Button>
                                                    </div>
                                                    <Transition
                                                        as={Fragment}
```

```
                                                    enter="transition ease-
out duration-100"
                                                    enterFrom="transform
opacity-0 scale-95"
                                                    enterTo="transform
opacity-100 scale-100"
                                                    leave="transition ease-
in duration-75"
                                                    leaveFrom="transform
opacity-100 scale-100"
                                                    leaveTo="transform
opacity-0 scale-95"
                                                >
                                                    <Menu.Items
className="absolute right-0 z-10 mt-2 w-48 origin-top-right rounded-md bg-
white py-1 shadow-lg ring-1 ring-black ring-opacity-5 focus:outline-none">

{userNavigation.map((item) => (
                                                            <Menu.Item
key={item.name}>
                                                                {({ active
}) => (
                                                                    <Link

to={item.link}

className={classNames(

active ? 'bg-gray-100' : '',

'block px-4 py-2 text-sm text-gray-700'
```

```
                                                                                )}
                                                                            >

{item.name}
                                                                            </Link>
                                                                        )}
                                                                    </Menu.Item>
                                                                ))}
                                                            </Menu.Items>
                                                        </Transition>
                                                    </Menu>
                                                </div>
                                            </div>
                                            <div className="-mr-2 flex md:hidden">
                                                <Disclosure.Button
className="relative inline-flex items-center justify-center rounded-md bg-
gray-800 p-2 text-gray-400 hover:bg-gray-700 hover:text-white
focus:outline-none focus:ring-2 focus:ring-white focus:ring-offset-2
focus:ring-offset-gray-800">
                                                    <span className="absolute -
inset-0.5" />
                                                    <span className="sr-only">Open
main menu</span>
                                                    {open ? (
                                                        <XMarkIcon className="block
h-6 w-6" aria-hidden="true" />
                                                    ) : (
                                                        <Bars3Icon className="block
h-6 w-6" aria-hidden="true" />
                                                    )}
                                                </Disclosure.Button>
```

```
                                    </div>
                                </div>
                            </div>

                            <Disclosure.Panel className="md:hidden">
                                <div className="space-y-1 px-2 pb-3 pt-2
sm:px-3">
                                    {navigation.map((item) =>
                                        item[userInfo.role] ? (
                                            <Link
                                                key={item.name}
                                                to={item.link}
                                                className={classNames(
                                                    item.current
                                                        ? 'bg-white text-
black'
                                                        : 'text-gray-300
hover:bg-gray-700 hover:text-white',
                                                    'rounded-md px-3 py-2
text-sm font-medium'
                                                )}
                                                aria-current={item.current
? 'page' : undefined}
                                            >
                                                {item.name}
                                            </Link>
                                        ) : null
                                    )}
                                </div>
                                <div className="border-t border-gray-700
pb-3 pt-4">
```

```
                                    <div className="flex items-center px-
5">
                                        <div className="flex-shrink-0">
                                            <img className="h-12 w-12
rounded-full" src='/user.png' alt="" />
                                        </div>
                                        <div className="ml-3">
                                            <div className="text-base font-
medium leading-none text-white">{userInfo.name}</div>
                                            <div className="text-sm font-
medium leading-none text-gray-400">{userInfo.email}</div>
                                        </div>
                                        <Link to="/cart">
                                            <button
                                                type="button"
                                                className="relative ml-auto
flex-shrink-0 rounded-full bg-gray-800 p-1 text-gray-400 hover:text-white
focus:outline-none focus:ring-2 focus:ring-transparent focus:ring-offset-2
focus:ring-offset-gray-800"
                                            >
                                                <span className="absolute -
inset-1.5" />
                                                <span className="sr-
only">View notifications</span>
                                                {items.length > 0 && <span
className="inline-flex items-center justify-center gap-2 rounded-md bg-
[#b0d2ff] px-2 py-1 font-medium text-[#1F2937] ring-1 ring-inset ring-
green-600/20 outline-none border-none mx-3">
                                                    <img src="/cart-
icon.png" className="h-8 w-8" aria-hidden="true" alt="Navbar Image" />
```

```
                                                    <p className='text-md
font-bold -mt-3 -ml-1 py-1'>{items.length}</p>
                                                        </span>}
                                                    </button>
                                                </Link>
                                            </div>
                                            <div className="mt-3 space-y-1 px-12">
                                                {userNavigation.map((item) => (
                                                    <Disclosure.Button
                                                        key={item.name}
                                                        className="block rounded-md
px-3 py-2 text-base font-medium text-gray-400 border-2 border-gray-400 w-
full my-5"
                                                    >
                                                        <Link
                                                            to={item.link}
                                                            className={classNames(
                                                                'block px-4 py-2
text-sm text-gray-200'
                                                            )}
                                                        >
                                                            {item.name}
                                                        </Link>
                                                    </Disclosure.Button>
                                                ))}
                                            </div>
                                        </div>
                                    </Disclosure.Panel>
                                </>
                            )}
                        </Disclosure>
```

```
                <header className="bg-white shadow">
                    <div className="mx-auto max-w-7xl px-4 py-3 sm:px-6
lg:px-8">
                        <h1 className="text-3xl font-bold tracking-tight
text-gray-900">Vendr</h1>
                    </div>
                </header>
                <main>
                    <div className="mx-auto max-w-7xl py-6 sm:px-6 lg:px-
8">{children}</div>
                </main>
            </div>}
        </div>
    )
}
export default Navbar;
```

Order.js

```
import React from 'react';
export default function Order() {
  return (<div id='order'></div>);
}
```

orderAPI.js

```
export function createOrder(order) {
  return new Promise(async (resolve) => {
    const response = await fetch('/orders', {
      method: 'POST',
      body: JSON.stringify(order),
      headers: { 'content-type': 'application/json' }
    });
    const data = await response.json();
    resolve({ data });
  });
}

export function fetchAllOrders(sort, pagination) {
  let queryString = '';

  for (let key in sort) {
    queryString += `${key}=${sort[key]}&`;
  }

  for (let key in pagination) {
    queryString += `${key}=${pagination[key]}&`;
  }



  return new Promise(async (resolve) => {
    const response = await fetch('/orders?' + queryString);
    const dataJSON = await response.json();
    const totalOrders = await response.headers.get('X-Total-Count');
```

```
    resolve({ data: { orders: dataJSON, totalOrders: +totalOrders } });
  });
}

export function updateOrder(update) {
  return new Promise(async (resolve) => {
    const response = await fetch('/orders/' + update.id, {
      method: 'PATCH',
      body: JSON.stringify(update),
      headers: { 'content-type': 'application/json' }
    });
    const data = await response.json();
    resolve({ data });
  });
}
```

orderSlice.js

```
import { createAsyncThunk, createSlice } from '@reduxjs/toolkit';
import { createOrder, fetchAllOrders, updateOrder } from './orderAPI';
import { toast } from 'react-toastify';

const initialState = {
  orders: [],
  status: 'idle',
  currentOrder: null,
  totalOrders: 0
};

export const createOrderAsync = createAsyncThunk(
```

```
  'order/createOrder',
  async (order) => {
    const response = await createOrder(order);
    return response.data;
  }
);

export const fetchAllOrdersAsync = createAsyncThunk(
  'order/fetchAllOrders',
  async ({ sort, pagination }) => {
    const response = await fetchAllOrders(sort, pagination);
    return response.data;
  }
);

export const updateOrderAsync = createAsyncThunk(
  'order/updateOrder',
  async (update) => {
    const response = await updateOrder(update);
    toast.info('Order status updated!', {
      position: "bottom-right",
      autoClose: 2000,
      hideProgressBar: false,
      closeOnClick: true,
      pauseOnHover: true,
      draggable: true,
      progress: undefined,
      theme: "colored"
    });
    return response.data;
  }
```

```
);

export const orderSlice = createSlice({
  name: 'order',
  initialState,
  reducers: {
    increment: (state) => {
      state.value += 1;
    },
    resetOrder: (state) => {
      state.currentOrder = null;
    }
  },
  extraReducers: (builder) => {
    builder
      .addCase(createOrderAsync.pending, (state) => {
        state.status = 'loading';
      })
      .addCase(createOrderAsync.fulfilled, (state, action) => {
        state.status = 'idle';
        state.orders.push(action.payload);
        state.currentOrder = action.payload;
      })
      .addCase(fetchAllOrdersAsync.pending, (state) => {
        state.status = 'loading';
      })
      .addCase(fetchAllOrdersAsync.fulfilled, (state, action) => {
        state.status = 'idle';
        state.orders = action.payload.orders;
        state.totalOrders = action.payload.totalOrders;
      })
```

```
      .addCase(updateOrderAsync.pending, (state) => {
        state.status = 'loading';
      })
      .addCase(updateOrderAsync.fulfilled, (state, action) => {
        state.status = 'idle';
        const index = state.orders.findIndex(order => order.id ===
action.payload.id);
        state.orders[index] = action.payload;
      });
  },
});

export const { increment, resetOrder } = orderSlice.actions;

export const selectCurrentOrder = (state) => state.order.currentOrder;
export const selectAllOrders = (state) => state.order.orders;
export const selectTotalOrders = (state) => state.order.totalOrders;
export const selectOrderStatus = (state) => state.order.status;

export default orderSlice.reducer;
```

ProductDetail.js

```
import React, { useEffect, useState } from 'react';


import { StarIcon } from '@heroicons/react/20/solid';
import { RadioGroup } from '@headlessui/react';
import { useDispatch, useSelector } from 'react-redux';
import { fetchProductByIdAsync, selectProductById, selectProductStatus }
from '../productSlice';
import { useParams } from 'react-router-dom';
import { addToCartAsync, selectItems } from '../../cart/cartSlice';

import { toast } from 'react-toastify';
import { MutatingDots } from 'react-loader-spinner';



function classNames(...classes) {
    return classes.filter(Boolean).join(' ');
}



export default function ProductDetail() {
    const [selectedColor, setSelectedColor] = useState({});
    const [selectedSize, setSelectedSize] = useState({});

    const product = useSelector(selectProductById);
    const dispatch = useDispatch();

    const cartItems = useSelector(selectItems);
    const loadingStatus = useSelector(selectProductStatus);
```

```
    const params = useParams();
    useEffect(() => {
        dispatch(fetchProductByIdAsync(params.id));
    }, [dispatch, params.id]);

    const handleCart = (e) => {
        e.preventDefault();
        if (cartItems.findIndex((item) => item.product.id === product.id) <
0) {
            const newItem = { product: product.id, quantity: 1 };

            if(selectedColor) {
                newItem.color = selectedColor;
            }
            if(selectedSize) {
                newItem.size = selectedSize;
            }

            dispatch(addToCartAsync(newItem));
        }
        else {
            toast.info('Item already exists in cart!', {
                position: "bottom-right",
                autoClose: 2000,
                hideProgressBar: false,
                closeOnClick: true,
                pauseOnHover: true,
                draggable: true,
                progress: undefined,
                theme: "colored"
            });
```

```
        }
    }

    return (
        <div>
            <div className="bg-white">

                {loadingStatus === 'loading' &&
                    <div className='w-full flex justify-center items-
center'>
                        <MutatingDots
                            visible={true}
                            height="100"
                            width="100"
                            color="#4464f2"
                            secondaryColor="#4464f2"
                            radius="12.5"
                            ariaLabel="mutating-dots-loading"
                            wrapperStyle={{}}
                            wrapperClass=""
                        />'
                    </div>}

                {product && loadingStatus===='idle' &&
                    (<div className="pt-6">
                        <nav aria-label="Breadcrumb">
                            <ol className="mx-auto flex max-w-2xl items-
center space-x-2 px-4 sm:px-6 lg:max-w-7xl lg:px-8">
                                {product.breadcrumbs &&
product.breadcrumbs.map((breadcrumb) => (
                                    <li key={breadcrumb.id}>
```

```
                                        <div className="flex items-center">
                                            <a href={breadcrumb.href}
className="mr-2 text-sm font-medium text-gray-900">
                                                {breadcrumb.name}
                                            </a>
                                            <svg
                                                width={16}
                                                height={20}
                                                viewBox="0 0 16 20"
                                                fill="currentColor"
                                                aria-hidden="true"
                                                className="h-5 w-4 text-
gray-300"
                                            >
                                                <path d="M5.697 4.34L8.98
16.532h1.327L7.025 4.341H5.697z" />
                                            </svg>
                                        </div>
                                    </li>
                                ))}
                                <li className="text-sm">
                                    <a href={product.href} aria-
current="page" className="font-medium text-gray-500 hover:text-gray-600">
                                        {product.brand}
                                    </a>
                                </li>
                            </ol>
                        </nav>

                        <div className="mx-auto mt-6 max-w-2xl sm:px-6
lg:grid lg:max-w-7xl lg:grid-cols-3 lg:gap-x-8 lg:px-8">
```

```
                            <div className="aspect-h-4 aspect-w-3 hidden
overflow-hidden rounded-lg lg:block">
                                <img
                                    src={product.images[0]}
                                    alt={product.title}
                                    className="h-full w-full object-cover
object-center"
                                />
                            </div>
                            <div className="hidden lg:grid lg:grid-cols-1
lg:gap-y-8">
                                <div className="aspect-h-2 aspect-w-3
overflow-hidden rounded-lg">
                                    <img
                                        src={product.images[1]}
                                        alt={product.title}
                                        className="h-full w-full object-
cover object-center"
                                    />
                                </div>
                                <div className="aspect-h-2 aspect-w-3
overflow-hidden rounded-lg">
                                    <img
                                        src={product.images[2]}
                                        alt={product.title}
                                        className="h-full w-full object-
cover object-center"
                                    />
                                </div>
                            </div>
```

```
                            <div className="aspect-h-5 aspect-w-4
lg:aspect-h-4 lg:aspect-w-3 sm:overflow-hidden sm:rounded-lg">
                                <img
                                    src={product.images[3]}
                                    alt={product.title}
                                    className="h-full w-full object-cover
object-center"
                                />
                            </div>
                        </div>

                        <div className="mx-auto max-w-2xl px-4 pb-16 pt-10
sm:px-6 lg:grid lg:max-w-7xl lg:grid-cols-3 lg:grid-rows-[auto,auto,1fr]
lg:gap-x-8 lg:px-8 lg:pb-24 lg:pt-16">
                            <div className="lg:col-span-2 lg:border-r
lg:border-gray-200 lg:pr-8">
                                <h1 className="text-2xl font-bold tracking-
tight text-gray-900 sm:text-3xl">{product.title}</h1>
                            </div>

                            <div className="mt-4 lg:row-span-3 lg:mt-0">
                                <h2 className="sr-only">Product
information</h2>
                                <p className="text-3xl tracking-tight font-
bold text-gray-900">${product.discountedPrice}</p>
                                <p className="text-1xl line-through font-
semibold tracking-tight text-gray-400">${product.price}</p>

                                {/* Reviews */}
                                <div className="mt-6">
                                    <h3 className="sr-only">Reviews</h3>
```

```
                                    <div className="flex items-center">
                                        <div className="flex items-center">
                                            {[0, 1, 2, 3, 4].map((rating)
=> (
                                                <StarIcon
                                                    key={rating}
                                                    className={classNames(
                                                        product.rating >
rating ? 'text-gray-900' : 'text-gray-200',
                                                        'h-5 w-5 flex-
shrink-0'
                                                    )}
                                                    aria-hidden="true"
                                                />
                                            ))}
                                        </div>
                                        <p className="sr-
only">{product.rating} out of 5 stars</p>
                                    </div>
                                </div>

                                <form className="mt-10">
                                    {product.colors &&
product.colors.length > 0 && <div>
                                        <h3 className="text-sm font-medium
text-gray-900">Color</h3>

                                        <RadioGroup value={selectedColor}
onChange={setSelectedColor} className="mt-4">
                                            <RadioGroup.Label
className="sr-only">Choose a color</RadioGroup.Label>
```

```
                                            <div className="flex items-
center space-x-3">
                                                {product.colors.map((color)
=> (
                                                    <RadioGroup.Option
                                                        key={color.name}
                                                        value={color}
                                                        className={({
active, checked }) =>
                                                            classNames(

color.selectedClass,
                                                                active &&
checked ? 'ring ring-offset-1' : '',
                                                                !active &&
checked ? 'ring-2' : '',
                                                                'relative -
m-0.5 flex cursor-pointer items-center justify-center rounded-full p-0.5
focus:outline-none'
                                                            )
                                                        }
                                                    >
                                                        <RadioGroup.Label
as="span" className="sr-only">
                                                            {color.name}
                                                        </RadioGroup.Label>
                                                        <span
                                                            aria-
hidden="true"

className={classNames(
```

```
color.class,
                                                                    'h-8 w-8
rounded-full border border-black border-opacity-10'
                                                                )}
                                                            />
                                                        </RadioGroup.Option>
                                                    ))}
                                                </div>
                                            </RadioGroup>
                                        </div>}

                                        {product.sizes && product.sizes.length
> 0 && <div className="mt-10">
                                            <div className="flex items-center
justify-between">
                                                <h3 className="text-sm font-
medium text-gray-900">Size</h3>
                                                <a href="/" className="text-sm
font-medium text-indigo-600 hover:text-indigo-500">
                                                    Size guide
                                                </a>
                                            </div>

                                            <RadioGroup value={selectedSize}
onChange={setSelectedSize} className="mt-4">
                                                <RadioGroup.Label
className="sr-only">Choose a size</RadioGroup.Label>
                                                <div className="grid grid-cols-
4 gap-4 sm:grid-cols-8 lg:grid-cols-4">
```

```
                                                    {product.sizes.map((size)
=> (
                                                        <RadioGroup.Option
                                                            key={size.name}
                                                            value={size}

disabled={!size.inStock}
                                                            className={({
active }) =>
                                                                classNames(

size.inStock
                                                                        ?
'cursor-pointer bg-white text-gray-900 shadow-sm'
                                                                        :
'cursor-not-allowed bg-gray-50 text-gray-200',
                                                                    active ?
'ring-2 ring-indigo-500' : '',
                                                                    'group
relative flex items-center justify-center rounded-md border py-3 px-4 text-
sm font-medium uppercase hover:bg-gray-50 focus:outline-none sm:flex-1
sm:py-6'
                                                                )
                                                            }
                                                        >
                                                            {({ active, checked
}) => (
                                                                <>

<RadioGroup.Label as="span">{size.name}</RadioGroup.Label>
```

```
{size.inStock ? (
                                                                    <span
className={classNames(
active ? 'border' : 'border-2',
checked ? 'border-indigo-500' : 'border-transparent',
'pointer-events-none absolute -inset-px rounded-md'
                                                                        )}
aria-hidden="true"
                                                                    />
                                                                ) : (
                                                                    <span
aria-hidden="true"
className="pointer-events-none absolute -inset-px rounded-md border-2
border-gray-200"
                                                                    >
<svg
className="absolute inset-0 h-full w-full stroke-2 text-gray-200"
viewBox="0 0 100 100"
preserveAspectRatio="none"
```

```
stroke="currentColor"
                                                                                >

<line x1={0} y1={100} x2={100} y2={0} vectorEffect="non-scaling-stroke" />

</svg>
                                                                                    </span>
                                                                                )}
                                                                            </>
                                                                        )}
                                                                    </RadioGroup.Option>
                                                                ))}
                                                            </div>
                                                        </RadioGroup>
                                                    </div>}

                                                    <button
                                                        onClick={handleCart}
                                                        type="submit"
                                                        className="mt-10 flex w-full items-
center justify-center rounded-md border border-transparent bg-indigo-600
px-8 py-3 text-base font-medium text-white hover:bg-indigo-700
focus:outline-none focus:ring-2 focus:ring-indigo-500 focus:ring-offset-2"
                                                    >
                                                        Add to Cart
                                                    </button>
                                                </form>
                                            </div>
```

```
                            <div className="py-10 lg:col-span-2 lg:col-
start-1 lg:border-r lg:border-gray-200 lg:pb-16 lg:pr-8 lg:pt-6">
                                <div>
                                    <h3 className="sr-
only">Description</h3>

                                    <div className="space-y-6">
                                        <p className="text-base text-gray-
900">{product.description}</p>
                                    </div>
                                </div>

                                {product.highlights &&
product.highlights.length > 0 && <div className="mt-10">
                                    <h3 className="text-sm font-medium
text-gray-900">Highlights</h3>
                                    <div className="mt-4">
                                        <ul className="list-disc space-y-2
pl-4 text-sm">

{product.highlights.map((highlight) => (
                                                <li key={highlight}
className="text-gray-400">
                                                    <span className="text-
gray-600">{highlight}</span>
                                                </li>
                                            ))}
                                        </ul>
                                    </div>
                                </div>}
```

```
                                <div className="mt-10">
                                    <h2 className="text-sm font-medium
text-gray-900">Details</h2>

                                    <div className="mt-4 space-y-6">
                                        <p className="text-sm text-gray-
600">{product.description}</p>
                                    </div>
                                </div>
                            </div>
                        </div>
                    </div>)
                }
            </div>
        </div>
    )
}
```

ProductList.js

```
import React, { useState, Fragment, useEffect } from 'react';
import { useDispatch, useSelector } from 'react-redux';

import { Dialog, Disclosure, Menu, Transition } from '@headlessui/react';
import { XMarkIcon } from '@heroicons/react/24/outline';
import { ChevronDownIcon, FunnelIcon, MinusIcon, PlusIcon, Squares2X2Icon }
from '@heroicons/react/20/solid';
import { StarIcon } from '@heroicons/react/20/solid';
```

```
import { selectAllProducts, fetchProductsByFilterAsync, selectTotalItems,
selectBrands, selectCategories, fetchAllCategoriesAsync,
fetchAllBrandsAsync, selectProductStatus } from '../productSlice';
import { ITEMS_PER_PAGE } from '../../../app/constants';

import { Link } from 'react-router-dom';
import Pagination from '../../common/Pagination';
import { MutatingDots } from 'react-loader-spinner';

const sortOptions = [
  { name: 'Best Rating', sort: 'rating', order: 'desc', current: false },
  { name: 'Price: Low to High', sort: 'discountedPrice', order: 'asc',
current: false },
  { name: 'Price: High to Low', sort: 'discountedPrice', order: 'desc',
current: false }
];

function classNames(...classes) {
  return classes.filter(Boolean).join(' ');
}


export default function ProductList() {
  const [mobileFiltersOpen, setMobileFiltersOpen] = useState(false);
  const [filter, setFilter] = useState({});
  const [sort, setSort] = useState({});
  const [page, setPage] = useState(1);

  const dispatch = useDispatch();
  const products = useSelector(selectAllProducts);
  const brands = useSelector(selectBrands);
```

```
  const categories = useSelector(selectCategories);
  const totalItems = useSelector(selectTotalItems);

  const loadingStatus = useSelector(selectProductStatus);

  const filters = [
    {
      id: 'category',
      name: 'Category',
      options: categories,
    },
    {
      id: 'brand',
      name: 'Brand',
      options: brands,
    }
  ];

  const handleFilter = (e, section, option) => {
    const newFilter = { ...filter };
    if (e.target.checked) {
      if (newFilter[section.id]) {
        newFilter[section.id].push(option.value);
      }
      else {
        newFilter[section.id] = [option.value];
      }
    }
    else {
      const index = newFilter[section.id].findIndex(el => el ===
option.value);
```

```
    newFilter[section.id].splice(index, 1);
  }
  setFilter(newFilter);
}

const handleSort = (e, option) => {
  const newSort = { _sort: option.sort, _order: option.order };
  setSort(newSort);
}

const handlePage = (page) => {
  setPage(page);
}

useEffect(() => {
  const pagination = { _page: page, _limit: ITEMS_PER_PAGE };
  dispatch(fetchProductsByFilterAsync({ filter, sort, pagination }));
}, [dispatch, filter, sort, page]);

useEffect(() => {
  setPage(1);
}, [totalItems, sort]);

useEffect(() => {
  dispatch(fetchAllBrandsAsync());
  dispatch(fetchAllCategoriesAsync());
}, [dispatch]);


return (
  <div>
```

```
      <div className="bg-white rounded-lg">
        <div>
          <MobileFilter mobileFiltersOpen={mobileFiltersOpen}
setMobileFiltersOpen={setMobileFiltersOpen} handleFilter={handleFilter}
filters={filters} />

          <main className="mx-auto max-w-7xl px-4 sm:px-6 lg:px-8">
            <div className="flex items-baseline justify-between border-b
border-gray-200 pb-6 pt-24">
              <h1 className="text-4xl font-bold tracking-tight text-gray-
900">All Products</h1>

              <div className="flex items-center">
                <Menu as="div" className="relative inline-block text-left">
                  <div>
                    <Menu.Button className="group inline-flex justify-
center text-sm font-medium text-gray-700 hover:text-gray-900">
                      Sort
                      <ChevronDownIcon
                        className="-mr-1 ml-1 h-5 w-5 flex-shrink-0 text-
gray-400 group-hover:text-gray-500"
                        aria-hidden="true"
                      />
                    </Menu.Button>
                  </div>

                  <Transition
                    as={Fragment}
                    enter="transition ease-out duration-100"
                    enterFrom="transform opacity-0 scale-95"
                    enterTo="transform opacity-100 scale-100"
```

```
                    leave="transition ease-in duration-75"
                    leaveFrom="transform opacity-100 scale-100"
                    leaveTo="transform opacity-0 scale-95"
                  >
                    <Menu.Items className="absolute right-0 z-10 mt-2 w-40
origin-top-right rounded-md bg-white shadow-2xl ring-1 ring-black ring-
opacity-5 focus:outline-none">
                      <div className="py-1">
                        {sortOptions.map((option) => (
                          <Menu.Item key={option.name}>
                            {({ active }) => (
                              <p
                                onClick={e => handleSort(e, option)}
                                className={classNames(
                                  option.current ? 'font-medium text-gray-
900' : 'text-gray-500',
                                  active ? 'bg-gray-100' : '',
                                  'block px-4 py-2 text-sm'
                                )}
                              >
                                {option.name}
                              </p>
                            )}
                          </Menu.Item>
                        ))}
                      </div>
                    </Menu.Items>
                  </Transition>
                </Menu>
```

```
                <button type="button" className="-m-2 ml-5 p-2 text-gray-
400 hover:text-gray-500 sm:ml-7">
                  <span className="sr-only">View grid</span>
                  <Squares2X2Icon className="h-5 w-5" aria-hidden="true" />
                </button>
                <button
                  type="button"
                  className="-m-2 ml-4 p-2 text-gray-400 hover:text-gray-
500 sm:ml-6 lg:hidden"
                  onClick={() => setMobileFiltersOpen(true)}
                >
                  <span className="sr-only">Filters</span>
                  <FunnelIcon className="h-5 w-5" aria-hidden="true" />
                </button>
              </div>
            </div>

            <section aria-labelledby="products-heading" className="pb-24
pt-6">
              <h2 id="products-heading" className="sr-only">
                Products
              </h2>

              <div className="grid grid-cols-1 gap-x-8 gap-y-10 lg:grid-
cols-4">
                <DesktopFilter handleFilter={handleFilter}
filters={filters} />

                <div className="lg:col-span-3">
                  <ProductGrid products={products}
loadingStatus={loadingStatus} />
```

```
                </div>
              </div>
            </section>

            <Pagination page={page} setPage={setPage}
handlePage={handlePage} totalItems={totalItems} />
          </main>
        </div>
      </div>
    </div>
  );
}


function MobileFilter({ mobileFiltersOpen, setMobileFiltersOpen,
handleFilter, filters }) {
  return (
    <Transition.Root show={mobileFiltersOpen} as={Fragment}>
      <Dialog as="div" className="relative z-40 lg:hidden"
onClose={setMobileFiltersOpen}>
        <Transition.Child
          as={Fragment}
          enter="transition-opacity ease-linear duration-300"
          enterFrom="opacity-0"
          enterTo="opacity-100"
          leave="transition-opacity ease-linear duration-300"
          leaveFrom="opacity-100"
          leaveTo="opacity-0"
        >
          <div className="fixed inset-0 bg-black bg-opacity-25" />
        </Transition.Child>
```

```
        <div className="fixed inset-0 z-40 flex">
          <Transition.Child
            as={Fragment}
            enter="transition ease-in-out duration-300 transform"
            enterFrom="translate-x-full"
            enterTo="translate-x-0"
            leave="transition ease-in-out duration-300 transform"
            leaveFrom="translate-x-0"
            leaveTo="translate-x-full"
          >
            <Dialog.Panel className="relative ml-auto flex h-full w-full
max-w-xs flex-col overflow-y-auto bg-white py-4 pb-12 shadow-xl">
              <div className="flex items-center justify-between px-4">
                <h2 className="text-lg font-medium text-gray-
900">Filters</h2>
                <button
                  type="button"
                  className="-mr-2 flex h-10 w-10 items-center justify-
center rounded-md bg-white p-2 text-gray-400"
                  onClick={() => setMobileFiltersOpen(false)}
                >
                  <span className="sr-only">Close menu</span>
                  <XMarkIcon className="h-6 w-6" aria-hidden="true" />
                </button>
              </div>

              <form className="mt-4 border-t border-gray-200">
                {filters.map((section) => (
                  <Disclosure as="div" key={section.id} className="border-t
border-gray-200 px-4 py-6">
```

```
                    {({ open }) => (
                      <>
                        <h3 className="-mx-2 -my-3 flow-root">
                          <Disclosure.Button className="flex w-full items-
center justify-between bg-white px-2 py-3 text-gray-400 hover:text-gray-
500">
                            <span className="font-medium text-gray-
900">{section.name}</span>
                            <span className="ml-6 flex items-center">
                              {open ? (
                                <MinusIcon className="h-5 w-5" aria-
hidden="true" />
                              ) : (
                                <PlusIcon className="h-5 w-5" aria-
hidden="true" />
                              )}
                            </span>
                          </Disclosure.Button>
                        </h3>
                        <Disclosure.Panel className="pt-6">
                          <div className="space-y-6">
                            {section.options.map((option, optionIdx) => (
                              <div key={option.value} className="flex
items-center">
                                <input
                                  id={`filter-mobile-${section.id}-
${optionIdx}`}
                                  name={`${section.id}[]`}
                                  defaultValue={option.value}
                                  type="checkbox"
                                  defaultChecked={option.checked}
```

```
                                  onChange={e => handleFilter(e, section,
option)}
                                  className="h-4 w-4 rounded border-gray-
300 text-indigo-600 focus:ring-indigo-500"
                                />
                                <label
                                  htmlFor={`filter-mobile-${section.id}-
${optionIdx}`}
                                  className="ml-3 min-w-0 flex-1 text-gray-
500"
                                >
                                  {option.label[0].toUpperCase() +
option.label.slice(1)}
                                </label>
                              </div>
                            ))}
                          </div>
                        </Disclosure.Panel>
                      </>
                    )}
                  </Disclosure>
                ))}
              </form>
            </Dialog.Panel>
          </Transition.Child>
        </div>
      </Dialog>
    </Transition.Root>
  );
}
```

```
function DesktopFilter({ handleFilter, filters }) {
  return (
    <form className="hidden lg:block">
      {filters.map((section) => (
        <Disclosure as="div" key={section.id} className="border-b border-
gray-200 py-6">
          {({ open }) => (
            <>
              <h3 className="-my-3 flow-root">
                <Disclosure.Button className="flex w-full items-center
justify-between bg-white py-3 text-sm text-gray-400 hover:text-gray-500">
                  <span className="font-medium text-gray-
900">{section.name}</span>
                  <span className="ml-6 flex items-center">
                    {open ? (
                      <MinusIcon className="h-5 w-5" aria-hidden="true" />
                    ) : (
                      <PlusIcon className="h-5 w-5" aria-hidden="true" />
                    )}
                  </span>
                </Disclosure.Button>
              </h3>
              <Disclosure.Panel className="pt-6">
                <div className="space-y-4">
                  {section.options.map((option, optionIdx) => (
                    <div key={option.value} className="flex items-center">
                      <input
                        id={`filter-${section.id}-${optionIdx}`}
                        name={`${section.id}[]`}
                        defaultValue={option.value}
                        type="checkbox"
```

```
                        defaultChecked={option.checked}
                        onChange={e => handleFilter(e, section, option)}
                        className="h-4 w-4 rounded border-gray-300 text-
indigo-600 focus:ring-indigo-500"
                      />
                      <label
                        htmlFor={`filter-${section.id}-${optionIdx}`}
                        className="ml-3 text-sm text-gray-600"
                      >
                        {option.label[0].toUpperCase() +
option.label.slice(1)}
                      </label>
                    </div>
                  ))}
                </div>
              </Disclosure.Panel>
            </>
          )}
        </Disclosure>
      ))}
    </form>
  );
}

function ProductGrid({ products, loadingStatus }) {
  return (
    <div className="bg-white">

      {loadingStatus === 'loading' &&
        <div className='w-full flex justify-center items-center'>
          <MutatingDots
```

```
            visible={true}
            height="100"
            width="100"
            color="#4464f2"
            secondaryColor="#4464f2"
            radius="12.5"
            ariaLabel="mutating-dots-loading"
            wrapperStyle={{}}
            wrapperClass=""
          />'
        </div>}

      {loadingStatus === 'idle' && <div className="mx-auto max-w-2xl px-4
py-0 sm:px-6 sm:py-0 lg:max-w-7xl lg:px-8">
        <div className="mt-6 grid grid-cols-1 gap-x-6 gap-y-10 sm:grid-
cols-2 lg:grid-cols-3 xl:gap-x-8">
          {products.map((product) => (
            <Link to={`/product-detail/${product.id}`} key={product.id}>
              <div className="group relative border border-1 border-gray-
400 p-3 rounded-md">
                <div className="aspect-h-1 aspect-w-1 w-full overflow-
hidden rounded-md bg-gray-200 lg:aspect-none group-hover:opacity-75 lg:h-
60">
                  <img
                    src={product.thumbnail}
                    alt={product.title}
                    className="h-full w-full object-cover object-center
lg:h-full lg:w-full"
                  />
                </div>
                <div className="mt-4 flex justify-between">
```

```
                  <div>
                    <h3 className="text-sm text-gray-700">
                      <div href={product.thumbnail}>
                        <span aria-hidden="true" className="absolute inset-
0" />
                        {product.title}
                      </div>
                    </h3>
                    <p className="mt-1 text-sm text-gray-500 flex items-
center">
                      <StarIcon className='w-5 h-5 inline' />
                      <span className='mx-1 mt-1'>{product.rating}/5</span>
                    </p>
                  </div>
                  <div className='flex flex-col items-center'>
                    <p className="text-md font-medium text-gray-
900">${product.discountedPrice}</p>
                    <p className="text-xs font-normal text-gray-400 line-
through">${product.price}</p>
                  </div>
                </div>
                {product.deleted && <div>
                  <p className='text-sm text-red-500 mt-2'>Product
deleted</p>
                </div>}
                {product.stock <= 0 && <div>
                  <p className='text-sm text-red-500 mt-2'>Out of stock</p>
                </div>}
              </div>
            </Link>
          ))}
```

```
        </div>
      </div>}
    </div>
  );
}
```

productAPI.js

```
export function fetchProductById(id) {
  return new Promise(async (resolve) => {
    const response = await fetch('/products/' + id);
    const data = await response.json();
    resolve({ data });
  });
}

export function createProduct(product) {
  return new Promise(async (resolve) => {
    const response = await fetch('/products/', {
      method: 'POST',
      body: JSON.stringify(product),
      headers: { 'content-type': 'application/json' }
    });
    const data = await response.json();
    resolve({ data });
  });
}

export function updateProduct(update) {
  return new Promise(async (resolve) => {
```

```
    const response = await fetch('/products/' + update.id, {
      method: 'PATCH',
      body: JSON.stringify(update),
      headers: { 'content-type': 'application/json' }
    });
    const data = await response.json();
    resolve({ data });
  });
}

export function fetchProductsByFilter(filter, sort, pagination, admin) {
  let queryString = '';

  for (let key in filter) {
    const categoryValues = filter[key];
    if (categoryValues.length) {
      queryString += `${key}=${categoryValues}&`;
    }
  }

  for (let key in sort) {
    queryString += `${key}=${sort[key]}&`;
  }

  for (let key in pagination) {
    queryString += `${key}=${pagination[key]}&`;
  }

  if (admin) {
    queryString += "admin=true";
  }
```

```
  return new Promise(async (resolve) => {
    const response = await fetch('/products?' + queryString);
    const dataJSON = await response.json();

    const totalItems = await response.headers.get('X-Total-Count');
    resolve({ data: { products: dataJSON, totalItems: +totalItems } });
  });
}

export function fetchAllCategories() {
  return new Promise(async (resolve) => {
    const response = await fetch('/categories');
    const data = await response.json();
    resolve({ data });
  });
}

export function fetchAllBrands() {
  return new Promise(async (resolve) => {
    const response = await fetch('/brands');
    const data = await response.json();
    resolve({ data });
  });
}
```

productSlice.js

```
import { createAsyncThunk, createSlice } from '@reduxjs/toolkit';
import { toast } from 'react-toastify';
```

```
import { createProduct, fetchAllBrands, fetchAllCategories,
fetchProductById, fetchProductsByFilter, updateProduct } from
'./productAPI';

const initialState = {
  products: [],
  brands: [],
  categories: [],
  status: 'idle',
  totalItems: 0,
  selectedProduct: null
};

export const fetchProductByIdAsync = createAsyncThunk(
  'product/fetchProductById',
  async (id) => {
    const response = await fetchProductById(id);
    return response.data;
  }
);

export const createProductAsync = createAsyncThunk(
  'product/createProduct',
  async (product) => {
    const response = await createProduct(product);
    toast.success('New item added successfully!', {
      position: "bottom-right",
      autoClose: 2000,
      hideProgressBar: false,
      closeOnClick: true,
      pauseOnHover: true,
```

```
      draggable: true,
      progress: undefined,
      theme: "colored"
    });
    return response.data;
  }
);

export const updateProductAsync = createAsyncThunk(
  'product/updateProduct',
  async (update) => {
    const response = await updateProduct(update);
    toast.success('Item updated successfully!', {
      position: "bottom-right",
      autoClose: 2000,
      hideProgressBar: false,
      closeOnClick: true,
      pauseOnHover: true,
      draggable: true,
      progress: undefined,
      theme: "colored"
    });
    return response.data;
  }
);

export const fetchProductsByFilterAsync = createAsyncThunk(
  'product/fetchProductsByFilter',
  async ({ filter, sort, pagination, admin }) => {
    const response = await fetchProductsByFilter(filter, sort, pagination,
admin);
```

```
    return response.data;
  }
);

export const fetchAllBrandsAsync = createAsyncThunk(
  'product/fetchAllBrands',
  async () => {
    const response = await fetchAllBrands();
    return response.data;
  }
);

export const fetchAllCategoriesAsync = createAsyncThunk(
  'product/fetchAllCategories',
  async () => {
    const response = await fetchAllCategories();
    return response.data;
  }
);

export const productSlice = createSlice({
  name: 'product',
  initialState,
  reducers: {
    increment: (state) => {
      state.value += 1;
    },
    clearSelectedProduct: (state) => {
      state.selectedProduct = null;
    }
  },
```

```
extraReducers: (builder) => {
  builder
    .addCase(fetchProductsByFilterAsync.pending, (state) => {
      state.status = 'loading';
    })
    .addCase(fetchProductsByFilterAsync.fulfilled, (state, action) => {
      state.status = 'idle';
      state.products = action.payload.products;
      state.totalItems = action.payload.totalItems;
    })
    .addCase(fetchAllBrandsAsync.pending, (state) => {
      state.status = 'loading';
    })
    .addCase(fetchAllBrandsAsync.fulfilled, (state, action) => {
      state.status = 'idle';
      state.brands = action.payload;
    })
    .addCase(fetchAllCategoriesAsync.pending, (state) => {
      state.status = 'loading';
    })
    .addCase(fetchAllCategoriesAsync.fulfilled, (state, action) => {
      state.status = 'idle';
      state.categories = action.payload;
    })
    .addCase(fetchProductByIdAsync.pending, (state) => {
      state.status = 'loading';
    })
    .addCase(fetchProductByIdAsync.fulfilled, (state, action) => {
      state.status = 'idle';
      state.selectedProduct = action.payload;
    })
```

```
      .addCase(createProductAsync.pending, (state) => {
        state.status = 'loading';
      })
      .addCase(createProductAsync.fulfilled, (state, action) => {
        state.status = 'idle';
        state.products.push(action.payload);
      })
      .addCase(updateProductAsync.pending, (state) => {
        state.status = 'loading';
      })
      .addCase(updateProductAsync.fulfilled, (state, action) => {
        state.status = 'idle';
        const index = state.products.findIndex(product => product.id ===
action.payload.id);
        state.products[index] = action.payload;
        state.selectedProduct = action.payload;
      });
  },
});

export const { increment, clearSelectedProduct } = productSlice.actions;

export const selectAllProducts = (state) => state.product.products;
export const selectProductById = (state) => state.product.selectedProduct;
export const selectBrands = (state) => state.product.brands;
export const selectCategories = (state) => state.product.categories;
export const selectTotalItems = (state) => state.product.totalItems;
export const selectProductStatus = (state) => state.product.status;

export default productSlice.reducer;
```

ContactUs.js

```
import React from 'react';
import { useForm } from 'react-hook-form';
import { useDispatch, useSelector } from 'react-redux';
import { Link } from 'react-router-dom';
import { selectUserInfo } from '../user/userSlice';
import { createQueryAsync } from './querySlice';

function ContactUs() {
    // eslint-disable-next-line no-unused-vars
    const { register, handleSubmit, reset, formState: { errors } } =
useForm();
    const userInfo = useSelector(selectUserInfo);
    const dispatch = useDispatch();

    return (
        <form noValidate className='bg-white p-6 rounded-lg'
onSubmit={handleSubmit((data) => {
            const queryObject = { userEmail: userInfo.email, query:
data.query };
            dispatch(createQueryAsync(queryObject));
            reset();
        })}>
```

```
            <div className="space-y-12">
                <div className="border-b border-gray-900/10 pb-12">
                    <h2 className="text-base font-semibold leading-7 text-
gray-900">Tell us what is your issue</h2>
                    <p className="mt-1 text-sm leading-6 text-gray-600">
                        We will try to reach out to you as soon as
possible.
                    </p>

                    <div className="mt-10 grid grid-cols-1 gap-x-6 gap-y-8
sm:grid-cols-6">
                        <div className="col-span-full">
                            <label htmlFor="query" className="block text-sm
font-medium leading-6 text-gray-900">
                                Your query
                            </label>
                            <div className="mt-2">
                                <textarea
                                    id="query"
                                    {...register('query', { required:
'Please enter your issue' })}
                                    rows={5}
                                    className="block w-full rounded-md
border-0 py-1.5 text-gray-900 shadow-sm ring-1 ring-inset ring-gray-300
placeholder:text-gray-400 focus:ring-2 focus:ring-inset focus:ring-indigo-
600 sm:text-sm sm:leading-6"
                                    defaultValue={''}
                                    placeholder='Enter your issue here'
                                />
                            </div>
                        </div>
```

```
                    </div>
                </div>
            </div>

            <div className="mt-6 flex items-center justify-end gap-x-6">
                <Link to='/' type="button" className="text-sm font-semibold
leading-6 text-gray-900">
                    Cancel
                </Link>
                <button
                    type="submit"
                    className="rounded-md bg-indigo-600 px-3 py-2 text-sm
font-semibold text-white shadow-sm hover:bg-indigo-500 focus-
visible:outline focus-visible:outline-2 focus-visible:outline-offset-2
focus-visible:outline-indigo-600"
                >
                    Submit
                </button>
            </div>
        </form>
    )
}

export default ContactUs;
```

queryAPI.js

```
export function fetchAllQueries() {
  return new Promise(async (resolve) => {
    const response = await fetch('/queries');
```

```
    const data = await response.json();
    resolve({ data });
  });
}

export function createQuery(pagination) {
  return new Promise(async (resolve) => {
    const response = await fetch('/queries', {
      method: 'POST',
      body: JSON.stringify(pagination),
      headers: { 'content-type': 'application/json' }
    });
    const data = await response.json();
    resolve({ data });
  });
}
```

querySlice.js

```
import { createAsyncThunk, createSlice } from '@reduxjs/toolkit';
import { createQuery, fetchAllQueries } from './queryAPI';
import { toast } from 'react-toastify';

const initialState = {
  value: 0,
  status: 'idle',
  queries: []
};

export const createQueryAsync = createAsyncThunk(
```

```
  'query/createQuery',
  async (queryObject) => {
    const response = await createQuery(queryObject);
    toast.success('Query sent successfully!', {
      position: "bottom-right",
      autoClose: 2000,
      hideProgressBar: false,
      closeOnClick: true,
      pauseOnHover: true,
      draggable: true,
      progress: undefined,
      theme: "colored"
    });
    return response.data;
  }
);

export const fetchAllQueriesAsync = createAsyncThunk(
  'query/fetchAllQueries',
  async () => {
    const response = await fetchAllQueries();
    return response.data;
  }
);

export const querySlice = createSlice({
  name: 'query',
  initialState,
  reducers: {
    increment: (state) => {
      state.value += 1;
```

```
    }
  },
  extraReducers: (builder) => {
    builder
      .addCase(createQueryAsync.pending, (state) => {
        state.status = 'loading';
      })
      .addCase(createQueryAsync.fulfilled, (state, action) => {
        state.status = 'idle';
        state.queries.push(action.payload);
      })
      .addCase(fetchAllQueriesAsync.pending, (state) => {
        state.status = 'loading';
      })
      .addCase(fetchAllQueriesAsync.fulfilled, (state, action) => {
        state.status = 'idle';
        state.queries = action.payload;
      });
  },
});

export const { increment } = querySlice.actions;
export const selectAllQueries = (state) => state.query.queries;
export const selectTotalQueries = (state) => state.query.queries.length;

export default querySlice.reducer;
```

UserOrders.js

```
import React, { useEffect } from 'react';
import { useSelector, useDispatch } from 'react-redux';
import { fetchLoggedInUserOrdersAsync, selectUserOrders, selectUserStatus }
from '../userSlice';
import { MutatingDots } from 'react-loader-spinner';

export default function UserOrders() {
    const dispatch = useDispatch();
    const orders = useSelector(selectUserOrders);

    const loadingStatus = useSelector(selectUserStatus);

    useEffect(() => {
        dispatch(fetchLoggedInUserOrdersAsync());
    }, [dispatch]);

    return (
        <div>
            <div className='flex flex-col w-full gap-5 py-5 px-2'>
```

```
                {loadingStatus === 'loading' &&
                    <div className='w-full flex justify-center items-
center'>
                        <MutatingDots
                            visible={true}
                            height="100"
                            width="100"
                            color="#4464f2"
                            secondaryColor="#4464f2"
                            radius="12.5"
                            ariaLabel="mutating-dots-loading"
                            wrapperStyle={{}}
                            wrapperClass=""
                        />'
                    </div>
                }

                {orders && orders.map((order, index) => (
                    <div key={index} className='mx-auto w-full max-w-7xl
px-4 sm:px-6 lg:px-8 bg-white rounded-lg'>
                        <h2 className='text-4xl font-bold tracking-tight
text-gray-900 pt-5 text-left'>Order number - #{order.id}</h2>
                        <h4 className='text-xl font-bold tracking-tight
text-red-600 mb-5 text-left'>Order status - {order.status[0].toUpperCase()
+ order.status.slice(1)}</h4>
                        <div className="border-t border-gray-200 py-6
sm:px-0">
                            <div className="flow-root">
                                <ul className="-my-6 divide-y divide-gray-
200">
                                    {order.items.map((item) => (
```

```
                                        <li key={item.id} className="flex
py-6">
                                            <div className="h-24 w-24 flex-
shrink-0 overflow-hidden rounded-md border border-gray-200">
                                                <img

src={item.product.thumbnail}

alt={item.product.title}
                                                    className="h-full w-
full object-cover object-center"
                                                />
                                            </div>

                                            <div className="ml-4 flex flex-
1 flex-col">
                                                <div>
                                                    <div className="flex
justify-between text-base font-medium text-gray-900">
                                                        <h3>
                                                            <a
href={item.product.thumbnail}>{item.product.title}</a>
                                                        </h3>
                                                        <p className="ml-
4">${item.product.discountedPrice}</p>
                                                    </div>
                                                    <p className="mt-1
text-sm text-gray-500">{item.product.brand[0].toUpperCase() +
item.product.brand.slice(1)}</p>

                                                </div>
```

```
                                                <div className="flex flex-1
items-end justify-between text-sm">
                                                    <div className="text-
gray-500">
                                                        <label
htmlFor="quantity" className="inline text-sm font-medium leading-6 text-
gray-900">Qty: {item.quantity}</label>
                                                    </div>
                                                    <div className="text-
gray-500">
                                                        <label
htmlFor="quantity" className="inline text-sm font-medium leading-6 text-
gray-900">{item.color && `Color: ${item.color.name}`}</label>
                                                    </div>
                                                    <div className="text-
gray-500">
                                                        <label
htmlFor="quantity" className="inline text-sm font-medium leading-6 text-
gray-900">{item.size && `Size: ${item.size.name}`}</label>
                                                    </div>
                                                </div>
                                            </div>
                                        </li>
                                    ))}
                                </ul>
                            </div>
                        </div>

                        <div className="border-t border-gray-200 py-2
sm:px-0">
```

```
                            <div className="flex justify-between text-base
font-medium text-gray-900 my-1">
                                <p className='text-xl font-bold tracking-
tight text-green-600'>Subtotal</p>
                                <p className='text-xl font-bold tracking-
tight text-green-600'>${order.totalAmount}</p>
                            </div>
                            <div className="flex justify-between text-base
font-medium text-gray-900 my-1">
                                <p className='text-lg font-bold tracking-
tight text-green-600'>Total Items in Cart</p>
                                <p className='text-lg font-bold tracking-
tight text-green-600'>{order.totalItems}</p>
                            </div>
                        </div>
                        <h4 className='border-t border-gray-200 text-xl
font-bold tracking-tight text-blue-600 mt-2 pt-3 text-left'>Shipping
Address</h4>
                        <div className="flex justify-between gap-x-6 py-1">
                            <div className="flex min-w-0 gap-x-4">
                                <div className="min-w-0 flex-auto">
                                    <p className="text-md font-semibold
leading-6 text-gray-900">{order.selectedAddress.name}</p>
                                    <p className="mt-1 truncate text-xs
leading-5 text-gray-500">Phone: {order.selectedAddress.phone}</p>
                                    <p className="mt-1 truncate text-xs
leading-5 text-gray-500">Email: {order.selectedAddress.email}</p>
                                </div>
                            </div>
                            <div className="hidden shrink-0 sm:flex
sm:flex-col sm:items-end">
```

```
                                <p className="text-sm leading-6 text-gray-
700">{order.selectedAddress.houseNumber} -
{order.selectedAddress.locality}</p>
                                <p className="text-sm leading-6 text-gray-
500">{order.selectedAddress.city}</p>
                                <p className="text-sm leading-6 text-gray-
500">{order.selectedAddress.state} - {order.selectedAddress.pinCode}</p>
                            </div>
                        </div>
                        <h4 className='border-t border-gray-200 text-xl
font-bold tracking-tight text-blue-600 mt-2 pt-3 text-left'>Payment
Mode</h4>
                        <h4 className='text-md font-semibold leading-6
text-gray-900 mb-5'>Payment via {order.selectedPaymentMode[0].toUpperCase()
+ order.selectedPaymentMode.slice(1)}</h4>
                    </div>
                ))}
            </div>
        </div>
    );
}
```

UserProfile.js

```
import React, { useState } from 'react';
import { useDispatch, useSelector } from 'react-redux';
import { selectUserInfo, selectUserStatus, updateUserAsync } from
'../userSlice';
import { useForm } from 'react-hook-form';
```

```
import { toast } from 'react-toastify';
import { MutatingDots } from 'react-loader-spinner';

import Modal from '../../common/Modal';

export default function UserProfile() {
    const dispatch = useDispatch();
    const userInfo = useSelector(selectUserInfo);

    const loadingStatus = useSelector(selectUserStatus);

    // eslint-disable-next-line no-unused-vars
    const { register, handleSubmit, reset, setValue, formState: { errors }
} = useForm();

    const [selectedEditIndex, setSelectedEditIndex] = useState(-1);
    const [showAddAddressForm, setShowAddAddressForm] = useState(false);

    const [openModal, setOpenModal] = useState(null);

    const handleEdit = (addressUpdate, index) => {
        const newUser = { ...userInfo, addresses: [...userInfo.addresses]
};
        newUser.addresses.splice(index, 1, addressUpdate);
        dispatch(updateUserAsync(newUser));
        setSelectedEditIndex(-1);

        toast.success('Address edited successfully!', {
            position: "bottom-right",
            autoClose: 2000,
            hideProgressBar: false,
```

```
            closeOnClick: true,
            pauseOnHover: true,
            draggable: true,
            progress: undefined,
            theme: "colored"
        });
    }

    const handleRemove = (e, index) => {
        const newUser = { ...userInfo, addresses: [...userInfo.addresses]
};
        newUser.addresses.splice(index, 1);
        dispatch(updateUserAsync(newUser));

        toast.success('Address removed successfully!', {
            position: "bottom-right",
            autoClose: 2000,
            hideProgressBar: false,
            closeOnClick: true,
            pauseOnHover: true,
            draggable: true,
            progress: undefined,
            theme: "colored"
        });
    }

    const handleEditForm = (index) => {
        setSelectedEditIndex(index);
        setShowAddAddressForm(false);
        const address = userInfo.addresses[index];
        setValue('name', address.name);
```

```
        setValue('email', address.email);
        setValue('phone', address.phone);
        setValue('houseNumber', address.houseNumber);
        setValue('locality', address.locality);
        setValue('city', address.city);
        setValue('state', address.state);
        setValue('pinCode', address.pinCode);
    }

    const handleAdd = (address) => {
        const newUser = { ...userInfo, addresses: [...userInfo.addresses,
address] };
        dispatch(updateUserAsync(newUser));
        setShowAddAddressForm(false);

        toast.success('Address added successfully!', {
            position: "bottom-right",
            autoClose: 2000,
            hideProgressBar: false,
            closeOnClick: true,
            pauseOnHover: true,
            draggable: true,
            progress: undefined,
            theme: "colored"
        });
    }

    return (
        <div>
            <div className='flex flex-col min-h-screen w-full gap-5 py-5
px-2'>
```

```
                {loadingStatus === 'loading' &&
                    <div className='w-full flex justify-center items-
center'>
                        <MutatingDots
                            visible={true}
                            height="100"
                            width="100"
                            color="#4464f2"
                            secondaryColor="#4464f2"
                            radius="12.5"
                            ariaLabel="mutating-dots-loading"
                            wrapperStyle={{}}
                            wrapperClass=""
                        />'
                    </div>}

                <div className='mx-auto w-full max-w-7xl px-4 sm:px-6
lg:px-8 bg-white rounded-lg'>
                    <h2 className='text-4xl font-bold tracking-tight text-
gray-900 pt-5 mb-5 text-left'>Name: {userInfo.name ? userInfo.name :
'GUEST'}</h2>
                    <h4 className='text-xl font-bold tracking-tight text-
red-600 mb-0 text-left'>Email address: {userInfo.email}</h4>
                    <h4 className='border-b border-gray-200 text-xl font-
bold tracking-tight text-red-600 mb-2 pb-4 text-left'>Phone number:
{userInfo.addresses.length > 0 ? `${userInfo.addresses[0].phone}` : 'Not
added yet'}</h4>
                    {userInfo.role === 'admin' && <h4 className='border-b
border-gray-200 text-xl font-bold tracking-tight text-green-600 mb-2 pb-4
text-left'>Role: {userInfo.role}</h4>}
```

```
                    <button
                        type="button"
                        onClick={() => {
                            setShowAddAddressForm(true);
                            setSelectedEditIndex(-1);
                        }}
                        className="my-6 rounded-md bg-green-600 px-3 py-2
text-sm font-semibold text-white shadow-sm hover:bg-green-500 focus-
visible:outline focus-visible:outline-2 focus-visible:outline-offset-2
focus-visible:outline-green-600"
                    >
                        Add a new address
                    </button>
                    {showAddAddressForm && <form noValidate className='bg-
white px-5 my-6 py-6 rounded-lg' onSubmit={handleSubmit((data) => {
                        handleAdd(data);
                        reset();
                    })}>
                        <div className="space-y-12">
                            <div className="border-b border-gray-900/10 pb-
12">
                                <h2 className="text-3xl font-semibold
leading-7 text-gray-900">Edit your existing address</h2>
                                <p className="mt-1 text-sm leading-6 text-
gray-600">Change the existing values and click on save.</p>

                                <div className="mt-10 grid grid-cols-1 gap-
x-6 gap-y-8 sm:grid-cols-6">
                                    <div className="sm:col-span-6">
```

```
                                        <label htmlFor="name"
className="block text-sm font-medium leading-6 text-gray-900">
                                            Full name
                                        </label>
                                        <div className="mt-2">
                                            <input
                                                type="text"
                                                {...register('name', {
required: 'Please enter your name' })}
                                                id="name"
                                                className="block w-full
rounded-md border-0 py-1.5 text-gray-900 shadow-sm ring-1 ring-inset ring-
gray-300 placeholder:text-gray-400 focus:ring-2 focus:ring-inset
focus:ring-indigo-600 sm:text-sm sm:leading-6"
                                            />
                                        </div>
                                    </div>

                                    <div className="sm:col-span-3">
                                        <label htmlFor="email"
className="block text-sm font-medium leading-6 text-gray-900">
                                            Email address
                                        </label>
                                        <div className="mt-2">
                                            <input
                                                id="email"
                                                {...register('email', {
required: 'Please enter your email address' })}
                                                type="email"
                                                className="block w-full
rounded-md border-0 py-1.5 text-gray-900 shadow-sm ring-1 ring-inset ring-
```

```
gray-300 placeholder:text-gray-400 focus:ring-2 focus:ring-inset
focus:ring-indigo-600 sm:text-sm sm:leading-6"
                                            />
                                        </div>
                                    </div>

                                    <div className="sm:col-span-3">
                                        <label htmlFor="phone"
className="block text-sm font-medium leading-6 text-gray-900">
                                            Phone number
                                        </label>
                                        <div className="mt-2">
                                            <input
                                                id="phone"
                                                {...register('phone', {
required: 'Please enter your email address' })}
                                                type="tel"
                                                className="block w-full
rounded-md border-0 py-1.5 text-gray-900 shadow-sm ring-1 ring-inset ring-
gray-300 placeholder:text-gray-400 focus:ring-2 focus:ring-inset
focus:ring-indigo-600 sm:text-sm sm:leading-6"
                                            />
                                        </div>
                                    </div>

                                    <div className="sm:col-span-3">
                                        <label htmlFor="houseNumber"
className="block text-sm font-medium leading-6 text-gray-900">
                                            House Number
                                        </label>
                                        <div className="mt-2">
```

```
                                                <input
                                                    type="text"
                                                    {...register('houseNumber',
{ required: 'Please enter your house number' })}
                                                    id="houseNumber"
                                                    className="block w-full
rounded-md border-0 py-1.5 text-gray-900 shadow-sm ring-1 ring-inset ring-
gray-300 placeholder:text-gray-400 focus:ring-2 focus:ring-inset
focus:ring-indigo-600 sm:text-sm sm:leading-6"
                                                />
                                            </div>
                                        </div>

                                        <div className="sm:col-span-3">
                                            <label htmlFor="locality"
className="block text-sm font-medium leading-6 text-gray-900">
                                                Locality
                                            </label>
                                            <div className="mt-2">
                                                <input
                                                    type="text"
                                                    {...register('locality', {
required: 'Please enter your Locality' })}
                                                    id="locality"
                                                    className="block w-full
rounded-md border-0 py-1.5 text-gray-900 shadow-sm ring-1 ring-inset ring-
gray-300 placeholder:text-gray-400 focus:ring-2 focus:ring-inset
focus:ring-indigo-600 sm:text-sm sm:leading-6"
                                                />
                                            </div>
                                        </div>
```

```
                                    <div className="sm:col-span-2 sm:col-
start-1">
                                        <label htmlFor="city"
className="block text-sm font-medium leading-6 text-gray-900">
                                            City
                                        </label>
                                        <div className="mt-2">
                                            <input
                                                type="text"
                                                {...register('city', {
required: 'Please enter your country' })}
                                                id="city"
                                                className="block w-full
rounded-md border-0 py-1.5 text-gray-900 shadow-sm ring-1 ring-inset ring-
gray-300 placeholder:text-gray-400 focus:ring-2 focus:ring-inset
focus:ring-indigo-600 sm:text-sm sm:leading-6"
                                            />
                                        </div>
                                    </div>

                                    <div className="sm:col-span-2">
                                        <label htmlFor="state"
className="block text-sm font-medium leading-6 text-gray-900">
                                            State / Province
                                        </label>
                                        <div className="mt-2">
                                            <input
                                                type="text"
                                                {...register('state', {
required: 'Please enter your state' })}
```

```
                                                id="state"
                                                className="block w-full
rounded-md border-0 py-1.5 text-gray-900 shadow-sm ring-1 ring-inset ring-
gray-300 placeholder:text-gray-400 focus:ring-2 focus:ring-inset
focus:ring-indigo-600 sm:text-sm sm:leading-6"
                                            />
                                        </div>
                                    </div>

                                    <div className="sm:col-span-2">
                                        <label htmlFor="pinCode"
className="block text-sm font-medium leading-6 text-gray-900">
                                            PIN / Postal code
                                        </label>
                                        <div className="mt-2">
                                            <input
                                                type="text"
                                                {...register('pinCode', {
required: 'Please enter your PIN code' })}
                                                id="pinCode"
                                                className="block w-full
rounded-md border-0 py-1.5 text-gray-900 shadow-sm ring-1 ring-inset ring-
gray-300 placeholder:text-gray-400 focus:ring-2 focus:ring-inset
focus:ring-indigo-600 sm:text-sm sm:leading-6"
                                            />
                                        </div>
                                    </div>
                                </div>
                            </div>
```

```
                            <div className="mt-5 flex items-center justify-
end gap-x-6">
                                <button
                                    type="button"
                                    onClick={(e) => {
                                        setShowAddAddressForm(false);
                                        setSelectedEditIndex(-1);
                                        reset();
                                    }}
                                    className="text-sm font-semibold
leading-6 text-gray-900 px-3 py-2 rounded-md bg-gray-100 hover:bg-gray-200"
                                >
                                    Cancel
                                </button>
                                <button
                                    type="submit"
                                    className="rounded-md bg-green-600 px-3
py-2 text-sm font-semibold text-white shadow-sm hover:bg-green-500 focus-
visible:outline focus-visible:outline-2 focus-visible:outline-offset-2
focus-visible:outline-green-600"
                                >
                                    Add
                                </button>
                            </div>
                        </div>
                    </form>}

                    {userInfo.addresses.length > 0 && <h4
className='border-t border-gray-200 text-xl font-bold tracking-tight text-
blue-600 mt-2 pt-3 text-left'>Saved Addresses</h4>}
                    {userInfo.addresses.map((address, index) =>
```

```
                        <div key={index}>
                            {selectedEditIndex === index && <form
noValidate className='bg-white px-5 my-6 py-6 rounded-lg'
onSubmit={handleSubmit((data) => {
                                handleEdit(data, index);
                                reset();
                            })}>
                                <div className="space-y-12">
                                    <div className="border-b border-gray-
900/10 pb-12">
                                        <h2 className="text-3xl font-
semibold leading-7 text-gray-900">Edit your existing address</h2>
                                        <p className="mt-1 text-sm leading-
6 text-gray-600">Change the existing values and click on save.</p>

                                        <div className="mt-10 grid grid-
cols-1 gap-x-6 gap-y-8 sm:grid-cols-6">
                                            <div className="sm:col-span-6">
                                                <label htmlFor="name"
className="block text-sm font-medium leading-6 text-gray-900">
                                                    Full name
                                                </label>
                                                <div className="mt-2">
                                                    <input
                                                        type="text"

{...register('name', { required: 'Please enter your name' })}
                                                        id="name"
                                                        className="block w-
full rounded-md border-0 py-1.5 text-gray-900 shadow-sm ring-1 ring-inset
```

```
ring-gray-300 placeholder:text-gray-400 focus:ring-2 focus:ring-inset
focus:ring-indigo-600 sm:text-sm sm:leading-6"
                                                        />
                                                    </div>
                                                </div>

                                                <div className="sm:col-span-3">
                                                    <label htmlFor="email"
className="block text-sm font-medium leading-6 text-gray-900">
                                                        Email address
                                                    </label>
                                                    <div className="mt-2">
                                                        <input
                                                            id="email"

{...register('email', { required: 'Please enter your email address' })}
                                                            type="email"
                                                            className="block w-
full rounded-md border-0 py-1.5 text-gray-900 shadow-sm ring-1 ring-inset
ring-gray-300 placeholder:text-gray-400 focus:ring-2 focus:ring-inset
focus:ring-indigo-600 sm:text-sm sm:leading-6"
                                                        />
                                                    </div>
                                                </div>

                                                <div className="sm:col-span-3">
                                                    <label htmlFor="phone"
className="block text-sm font-medium leading-6 text-gray-900">
                                                        Phone number
                                                    </label>
                                                    <div className="mt-2">
```

```
                                                        <input
                                                            id="phone"

{...register('phone', { required: 'Please enter your email address' })}
                                                            type="tel"
                                                            className="block w-
full rounded-md border-0 py-1.5 text-gray-900 shadow-sm ring-1 ring-inset
ring-gray-300 placeholder:text-gray-400 focus:ring-2 focus:ring-inset
focus:ring-indigo-600 sm:text-sm sm:leading-6"
                                                        />
                                                    </div>
                                                </div>

                                                <div className="sm:col-span-3">
                                                    <label
htmlFor="houseNumber" className="block text-sm font-medium leading-6 text-
gray-900">
                                                        House Number
                                                    </label>
                                                    <div className="mt-2">
                                                        <input
                                                            type="text"

{...register('houseNumber', { required: 'Please enter your house number'
})}
                                                            id="houseNumber"
                                                            className="block w-
full rounded-md border-0 py-1.5 text-gray-900 shadow-sm ring-1 ring-inset
ring-gray-300 placeholder:text-gray-400 focus:ring-2 focus:ring-inset
focus:ring-indigo-600 sm:text-sm sm:leading-6"
                                                        />
```

```
                                                    </div>
                                                </div>

                                                <div className="sm:col-span-3">
                                                    <label htmlFor="locality"
className="block text-sm font-medium leading-6 text-gray-900">
                                                        Locality
                                                    </label>
                                                    <div className="mt-2">
                                                        <input
                                                            type="text"

{...register('locality', { required: 'Please enter your Locality' })}
                                                            id="locality"
                                                            className="block w-
full rounded-md border-0 py-1.5 text-gray-900 shadow-sm ring-1 ring-inset
ring-gray-300 placeholder:text-gray-400 focus:ring-2 focus:ring-inset
focus:ring-indigo-600 sm:text-sm sm:leading-6"
                                                        />
                                                    </div>
                                                </div>

                                                <div className="sm:col-span-2
sm:col-start-1">
                                                    <label htmlFor="city"
className="block text-sm font-medium leading-6 text-gray-900">
                                                        City
                                                    </label>
                                                    <div className="mt-2">
                                                        <input
                                                            type="text"
```

```
{...register('city', { required: 'Please enter your country' })}
                                                            id="city"
                                                            className="block w-
full rounded-md border-0 py-1.5 text-gray-900 shadow-sm ring-1 ring-inset
ring-gray-300 placeholder:text-gray-400 focus:ring-2 focus:ring-inset
focus:ring-indigo-600 sm:text-sm sm:leading-6"
                                                                />
                                                        </div>
                                                    </div>

                                                    <div className="sm:col-span-2">
                                                        <label htmlFor="state"
className="block text-sm font-medium leading-6 text-gray-900">
                                                                State / Province
                                                        </label>
                                                        <div className="mt-2">
                                                            <input
                                                                type="text"

{...register('state', { required: 'Please enter your state' })}
                                                            id="state"
                                                            className="block w-
full rounded-md border-0 py-1.5 text-gray-900 shadow-sm ring-1 ring-inset
ring-gray-300 placeholder:text-gray-400 focus:ring-2 focus:ring-inset
focus:ring-indigo-600 sm:text-sm sm:leading-6"
                                                                />
                                                        </div>
                                                    </div>

                                                    <div className="sm:col-span-2">
```

```
                                                <label htmlFor="pinCode"
className="block text-sm font-medium leading-6 text-gray-900">
                                                    PIN / Postal code
                                                </label>
                                                <div className="mt-2">
                                                    <input
                                                        type="text"

{...register('pinCode', { required: 'Please enter your PIN code' })}
                                                        id="pinCode"
                                                        className="block w-
full rounded-md border-0 py-1.5 text-gray-900 shadow-sm ring-1 ring-inset
ring-gray-300 placeholder:text-gray-400 focus:ring-2 focus:ring-inset
focus:ring-indigo-600 sm:text-sm sm:leading-6"
                                                    />
                                                </div>
                                            </div>
                                        </div>
                                    </div>

                                    <div className="mt-5 flex items-center
justify-end gap-x-6">
                                        <button
                                            type="button"
                                            onClick={(e) => {
                                                setSelectedEditIndex(-1)

setShowAddAddressForm(false);
                                                reset();
                                            }}
```

```
                                            className="text-sm font-
semibold leading-6 text-gray-900 px-3 py-2 rounded-md bg-gray-100 hover:bg-
gray-200"
                                        >
                                            Cancel
                                        </button>
                                        <button
                                            type="submit"
                                            className="rounded-md bg-
indigo-600 px-3 py-2 text-sm font-semibold text-white shadow-sm hover:bg-
indigo-500 focus-visible:outline focus-visible:outline-2 focus-
visible:outline-offset-2 focus-visible:outline-indigo-600"
                                        >
                                            Save
                                        </button>
                                    </div>
                                </div>
                            </form>}
                            <div key={index} className="flex justify-
between gap-x-6 py-5">
                                <div className="flex min-w-0 gap-x-4">
                                    <div className="min-w-0 flex-auto">
                                        <p className="text-md font-semibold
leading-6 text-gray-900">{address.name}</p>
                                        <p className="mt-1 truncate text-xs
leading-5 text-gray-500">Phone: {address.phone}</p>
                                        <p className="mt-1 truncate text-xs
leading-5 text-gray-500">Email: {address.email}</p>
                                    </div>
                                </div>
```

```
                                <div className="hidden shrink-0 sm:flex
sm:flex-col sm:items-end w-1/3">
                                    <p className="text-sm leading-6 text-
gray-700">{address.houseNumber} - {address.locality}</p>
                                    <p className="text-sm leading-6 text-
gray-500">{address.city}</p>
                                    <p className="text-sm leading-6 text-
gray-500">{address.state} - {address.pinCode}</p>
                                </div>
                                <div className="hidden shrink-0 sm:flex
sm:flex-col sm:items-end">
                                    <button
                                        onClick={(e) =>
handleEditForm(index)}
                                        type="button"
                                        className="font-medium text-indigo-
400 hover:text-indigo-600"
                                    >
                                        Edit
                                    </button>
                                    <Modal
                                        title={'Remove address'}
                                        message={`Are you sure you want to
remove this address?`}
                                        dangerOption='Remove'
                                        cancelOption='Cancel'
                                        dangerAction={(e) =>
handleRemove(e, index)}
                                        cancelAction={(e) => setOpenModal(-
1)}
                                        showModal={openModal === index}
```

```
                                    />
                                    <button
                                        onClick={(e) =>
setOpenModal(index)}
                                        type="button"
                                        className="font-medium text-indigo-
400 hover:text-indigo-600"
                                    >
                                        Remove
                                    </button>
                                </div>
                            </div>
                        </div>
                    )}
                </div>
            </div>
        </div>
    );
}
```

userAPI.js

```
export function fetchLoggedInUserOrders() {
  return new Promise(async (resolve) => {
    const response = await fetch('/orders/own');
    const data = await response.json();
    resolve({ data });
  });
}
```

```
export function fetchLoggedInUser() {
  return new Promise(async (resolve) => {
    const response = await fetch('/users/own');
    const data = await response.json();
    resolve({ data });
  });
}

export function updateUser(update) {
  return new Promise(async (resolve) => {
    const response = await fetch('/users/' + update.id, {
      method: 'PATCH',
      body: JSON.stringify(update),
      headers: { 'content-type': 'application/json' }
    });
    const data = await response.json();
    resolve({ data });
  });
}
```

userSlice.js

```
import { createAsyncThunk, createSlice } from '@reduxjs/toolkit';
import { fetchLoggedInUser, fetchLoggedInUserOrders, updateUser } from
'./userAPI';

const initialState = {
  status: 'idle',
  userInfo: null
};
```

```
export const fetchLoggedInUserOrdersAsync = createAsyncThunk(
  'user/fetchLoggedInUserOrders',
  async () => {
    const response = await fetchLoggedInUserOrders();
    return response.data;
  }
);

export const fetchLoggedInUserAsync = createAsyncThunk(
  'user/fetchLoggedInUser',
  async () => {
    const response = await fetchLoggedInUser();
    return response.data;
  }
);

export const updateUserAsync = createAsyncThunk(
  'user/updateUser',
  async (update) => {
    const response = await updateUser(update);
    return response.data;
  }
);

export const userSlice = createSlice({
  name: 'user',
  initialState,
  reducers: {
    increment: (state) => {
      state.value += 1;
```

```
    }
  },
  extraReducers: (builder) => {
    builder
      .addCase(fetchLoggedInUserOrdersAsync.pending, (state) => {
        state.status = 'loading';
      })
      .addCase(fetchLoggedInUserOrdersAsync.fulfilled, (state, action) => {
        state.status = 'idle';
        state.userInfo.orders = action.payload;
      })
      .addCase(updateUserAsync.pending, (state) => {
        state.status = 'loading';
      })
      .addCase(updateUserAsync.fulfilled, (state, action) => {
        state.status = 'idle';
        state.userInfo = action.payload;
      })
      .addCase(fetchLoggedInUserAsync.pending, (state) => {
        state.status = 'loading';
      })
      .addCase(fetchLoggedInUserAsync.fulfilled, (state, action) => {
        state.status = 'idle';
        state.userInfo = action.payload;
      });
  },
});

export const { increment } = userSlice.actions;

export const selectUserOrders = (state) => state.user.userInfo.orders;
```

```
export const selectUserInfo = (state) => state.user.userInfo;

export const selectUserStatus = (state) => state.user.status;

export default userSlice.reducer;
```

```
    Directory: C:\~VENDR\client\src\pages


Mode                LastWriteTime         Length Name
----                -------------         ------ ----
-a----        15-05-2024     02:27          11601 AboutUsPage.js
-a----        22-04-2024     23:56            357 AdminHomepage.js
-a----        24-04-2024     01:39            346 AdminOrdersPage.js
-a----        22-04-2024     23:55            381 AdminProductDetailPage.js
-a----        23-04-2024     01:01            356 AdminProductFormPage.js
-a----        15-05-2024     21:29            341 AdminQueryPage.js
-a----        26-04-2024     19:44            253 CartPage.js
-a----        16-05-2024     15:19           2779 CheckoutForm.js
-a----        16-05-2024     15:36          28389 CheckoutPage.js
-a----        15-05-2024     21:01            325 ContactUsPage.js
-a----        21-04-2024     18:36            262 ForgotPasswordPage.js
-a----        25-04-2024     01:38            407 Homepage.js
-a----        09-04-2024     00:52            217 LoginPage.js
-a----        01-05-2024     01:30           1992 OrderSuccessPage.js
-a----        19-04-2024     01:36           1400 PageNotFound.js
-a----        10-05-2024     01:38           2938 PaymentPage.js
-a----        11-04-2024     13:41            358 ProductDetailPage.js
-a----        11-05-2024     02:01            257 ResetPasswordPage.js
-a----        09-04-2024     01:07            222 SignupPage.js
-a----        09-05-2024     02:26           1473 StripeCheckout.js
-a----        21-04-2024     14:25            340 UserOrdersPage.js
-a----        21-04-2024     14:25            345 UserProfilePage.js
```

AboutUsPage.js

```
import React from 'react'
import Navbar from '../features/navbar/Navbar'
import { Link } from 'react-router-dom'

function AboutUsPage() {
    return (
        <div>
            <Navbar>
                <div className="bg-white relative   flex items-center  n
justify-center overflow-hidden z-50 ">
                    <div className="relative mx-auto h-full px-4  pb-20
md:pb-10 sm:max-w-xl md:max-w-full md:px-24 lg:max-w-screen-xl lg:px-8">
                        <div className="flex flex-col items-center justify-
between lg:flex-row py-16">
                            <div className=" relative ">
                                <div className=" absolute top-0 -left-48 z-
0  opacity-50 ">
                                    <img
                                        src="https://placehold.co/200x100"
                                        className="w-36 z-0 h-full object-
fill fill-y text-y"
                                        alt=""
                                    />
                                </div>
                                <div className="lg:max-w-xl lg:pr-5
relative z-40">
                                    <p className="flex text-sm uppercase
text-g1  ">About Us</p>
```

```
                                    <h2 className="mb-6 max-w-lg text-5xl
font-light leading-snug tracking-tight text-g1 sm:text-7xl sm:leading-
snug">
                                        Shop Smart & Easy
                                        <span className="my-1 inline-block
border-b-8 border-g4 bg-white px-4 font-bold text-g4 animate__animated
animate__flash">
                                            - VENDR
                                        </span>
                                    </h2>
                                    <p className="text-base text-gray-700">
                                        The primary objective of "Vendr," a
web-based storefront application, is to provide a seamless, efficient, and
scalable online shopping platform. By leveraging the MERN framework, Vendr
aims to offer both individual users and businesses a robust environment for
displaying and managing an extensive range of products. The application
will integrate advanced functionalities like secure payment transactions
through Stripe, comprehensive product management, and a personalized
shopping experience, with an emphasis on high performance and user-friendly
interfaces.<br/><br/>
                                    </p>
                                    <p className="text-base text-gray-700">
                                    "Vendr" is not just a product of
technology; it's the culmination of relentless dedication and teamwork.
Crafted by the hands and minds of Abhik Gupta, John P Varghese, Yatin
Hooda, and Nishchay Chandok, this project embodies the spirit of
collaboration and shared vision. Their collective efforts have been the
driving force behind the realization of this ambitious endeavor. With each
member contributing their unique skills and expertise, "Vendr" has been
shaped into a sophisticated and innovative e-commerce solution, poised to
make its mark in the digital marketplace.
```

```
                                    </p>
                                    <div className="mt-10 flex flex-col
items-center md:flex-row">
                                        <Link
                                            to="/"
                                            className="mb-3 inline-flex h-
12 w-full items-center justify-center rounded bg-[#F7B030] px-6 font-medium
tracking-wide text-black hover:text-white shadow-md transition hover:bg-
[#0096FF] focus:outline-none md:mr-4 md:mb-0 md:w-auto"
                                        >
                                            Visit our website
                                        </Link>
                                        <a

href="https://github.com/abhik2207/Vendr-Development"
                                            target="_blank"
                                            rel="noreferrer"
                                            aria-label=""
                                            className="group inline-flex
items-center font-semibold text-g1"
                                        >
                                            Watch how we built this
                                            <svg

xmlns="http://www.w3.org/2000/svg"
                                                className="ml-4 h-6 w-6
transition-transform group-hover:translate-x-2"
                                                fill="none"
                                                viewBox="0 0 24 24"
                                                stroke="currentColor"
                                                strokeWidth={2}
```

```
                                            >
                                                <path
                                                    strokeLinecap="round"
                                                    strokeLinejoin="round"
                                                    d="M17 8l4 4m0 0l-4
4m4-4H3"
                                                />
                                            </svg>
                                        </a>
                                    </div>
                                </div>
                            </div>
                            <div className="relative hidden lg:ml-32
lg:block lg:w-1/2">
                                <svg
                                    xmlns="http://www.w3.org/2000/svg"
                                    className="my-6 mx-auto h-10 w-10
animate-bounce rounded-full bg-white p-2 lg:hidden"
                                    fill="none"
                                    viewBox="0 0 24 24"
                                    stroke="currentColor"
                                    strokeWidth={2}
                                >
                                    <path
                                        strokeLinecap="round"
                                        strokeLinejoin="round"
                                        d="M16 17l-4 4m0 0l-4-4m4 4V3"
                                    />
                                </svg>
                                <div className="abg-orange-400 mx-auto w-
fit overflow-hidden rounded-[6rem] rounded-br-none rounded-tl-none">
```

```
                                <img src="/favicon.png" alt="" />
                            </div>
                        </div>
                    </div>
                </div>
                <div className="hidden text-9xl varien absolute top-6
left-1/4 text-g/10 z-40    ">
                    About Us
                </div>
                <div className=" absolute -bottom-24 left-10 z-0
opacity-10 ">
                    <svg
                        width="800px"
                        height="800px"
                        viewBox="0 0 24 24"
                        className="w-96 z-0 h-full object-fill fill-
gray-300 text-gray-300"
                        fill="none"
                        xmlns="http://www.w3.org/2000/svg"
                    >
                        <path
                            fillRule="evenodd"
                            clipRule="evenodd"
                            d="M12 6C12 5.44772 11.5523 5 11 5C10.4477
5 10 5.44772 10 6V16C10 16.5523 10.4477 17 11 17C11.5523 17 12 16.5523 12
16V6ZM9 9C9 8.44772 8.55228 8 8 8C7.44772 8 7 8.44772 7 9V16C7 16.5523
7.44772 17 8 17C8.55228 17 9 16.5523 9 16V9ZM15 9C15 8.44772 14.5523 8 14
8C13.4477 8 13 8.44772 13 9V16C13 16.5523 13.4477 17 14 17C14.5523 17 15
16.5523 15 16V9ZM18 13C18 12.4477 17.5523 12 17 12C16.4477 12 16 12.4477 16
13V16C16 16.5523 16.4477 17 17 17C17.5523 17 18 16.5523 18 16V13ZM6 15C6
14.4477 5.55228 14 5 14C4.44772 14 4 14.4477 4 15V16C4 16.5523 4.44772 17 5
```

```
17C5.55228 17 6 16.5523 6 16V15ZM21 15C21 14.4477 20.5523 14 20 14C19.4477
14 19 14.4477 19 15V16C19 16.5523 19.4477 17 20 17C20.5523 17 21 16.5523 21
16V15ZM4 18C3.44772 18 3 18.4477 3 19C3 19.5523 3.44772 20 4 20H21C21.5523
20 22 19.5523 22 19C22 18.4477 21.5523 18 21 18H4Z"
                            />
                        </svg>
                    </div>
                    <div className=" absolute -bottom-0 left-3/4 z-0
opacity-10 ">
                        <svg
                            width="800px"
                            height="800px"
                            viewBox="0 0 256 256"
                            xmlns="http://www.w3.org/2000/svg"
                            className="w-48 z-0  h-full -rotate-90 object-
fill fill-red-300 text-red-300"
                        >
                            <path
                                d="M32 225h12.993A4.004 4.004 0 0 0 49
220.997V138.01c0-4.976.724-5.04 1.614-.16l12.167 66.708c.397 2.177 2.516
3.942 4.713 3.942h8.512a3.937 3.937 0 0 0 3.947-4S79 127.5 80 129s14.488
52.67 14.488 52.67c.559 2.115 2.8 3.83 5.008 3.83h8.008a3.993 3.993 0 0 0
3.996-3.995v-43.506c0-4.97 1.82-5.412 4.079-.965l10.608 20.895c1.001 1.972
3.604 3.571 5.806 3.571h9.514a3.999 3.999 0 0 0 3.993-4.001v-19.49c0-4.975
2.751-6.074 6.155-2.443l6.111 6.518c1.51 1.61 4.528 2.916 6.734
2.916h7c2.21 0 5.567-.855 7.52-1.92l9.46-5.16c1.944-1.06 5.309-1.92 7.524-
1.92h23.992a4.002 4.002 0 0 0 4.004-3.992v-7.516a3.996 3.996 0 0 0-4.004-
3.992h-23.992c-2.211 0-5.601.823-7.564 1.834l-4.932 2.54c-4.423 2.279-
12.028 3.858-16.993 3.527l2.97.198c-4.962-.33-10.942-4.12-13.356-8.467l-
11.19-20.14c-1.07-1.929-3.733-3.492-5.939-3.492h-7c-2.21 0-4 1.794-4
4.001v19.49c0 4.975-1.14 5.138-2.542.382l-12.827-43.535c-.625-2.12-2.92-
```

```
3.838-5.127-3.838h-8.008c-2.207 0-3.916 1.784-3.817 4.005l1.92 42.998c.221
4.969-.489 5.068-1.585.224l-15.13-66.825c-.488-2.155-2.681-3.902-4.878-
3.902h-8.512a3.937 3.937 0 0 0-3.947 4s.953 77-.047 75.5-13.937-92.072-
13.937-92.072C49.252 34.758 47.21 33 45 33H31.999"
                                fillRule="evenodd"
                            />
                        </svg>
                    </div>
                    <div className=" absolute top-10 left-3/4 z-0  opacity-
10 ">
                        <svg
                            fill="#000000"
                            width="800px"
                            height="800px"
                            viewBox="0 0 256 256"
                            xmlns="http://www.w3.org/2000/svg"
                            className="w-96 z-0  h-full    object-fill
fill-blue-300 text-blue-300"
                        >
                            <path
                                d="M230.704 99.2a4.004 4.004 0 0 0-4.01-
3.995h-50.981c-2.215 0-5.212-1.327-6.693-2.964L155.289 77.08c-17.795-19.65-
41.628-16.256-53.234 7.58l-38.736 79.557C60.42 170.172 52.705 175 46.077
175H29.359a3.996 3.996 0 0 0-3.994 3.995v10.01A4 4 0 0 0 29.372
193h24.7c8.835 0 19.208-6.395 23.174-14.293l43.645-86.914c3.964-7.894
12.233-9.228 18.473-2.974l17.184 17.219c3.123 3.13 9.242 5.667 13.647
5.667H226.7a4.005 4.005 0 0 0 4.004-3.994v-8.512z"
                                fillRule="evenodd"
                            />
                        </svg>
                    </div>
```

```
                </div>
            </Navbar>
        </div>
    )
}

export default AboutUsPage;
```

AdminHomepage.js

```
import React from 'react';
import Navbar from '../features/navbar/Navbar';
import AdminProductList from
'../features/admin/components/AdminProductList';

function AdminHomepage() {
    return (
        <div>
            <Navbar>
                <AdminProductList />
            </Navbar>
        </div>
    )
}
export default AdminHomepage;
```

AdminOrdersPage.js

```
import React from 'react';
import Navbar from '../features/navbar/Navbar';
import AdminOrders from '../features/admin/components/AdminOrders';

function AdminOrdersPage() {
    return (
        <div>
            <Navbar>
                <AdminOrders />
            </Navbar>
        </div>
    )
}
export default AdminOrdersPage;
```

AdminProductDetail.js

```
import React from 'react';
import Navbar from '../features/navbar/Navbar';
import AdminProductDetail from
'../features/admin/components/AdminProductDetail';

function AdminProductDetailPage() {
    return (
        <div>
            <Navbar>
                <AdminProductDetail />
            </Navbar>
```

```
        </div>
    )
}
export default AdminProductDetailPage;
```

AdminProductFormPage.js

```
import React from 'react';
import Navbar from '../features/navbar/Navbar';
import ProductForm from '../features/admin/components/ProductForm';

function AdminProductFormPage() {
    return (
        <div>
            <Navbar>
                <ProductForm />
            </Navbar>
        </div>
    )
}
export default AdminProductFormPage;
```

AdminQueryPage.js

```
import React from 'react';
import Navbar from '../features/navbar/Navbar';
import AdminQuery from '../features/admin/components/AdminQuery';

function AdminQueryPage() {
```

```
    return (
        <div>
            <Navbar>
                <AdminQuery />
            </Navbar>
        </div>
    )
}
export default AdminQueryPage;
```

CartPage.js

```
import React from 'react';
import Cart from '../features/cart/Cart';

function CartPage() {
    return (
        <div className='w-full min-h-screen overflow-hidden p-6'>
            <Cart />
        </div>
    )
}
export default CartPage;
```

CheckoutPage.js

```
import React, { useState } from 'react';
import { useDispatch, useSelector } from 'react-redux';
import { Link, Navigate } from 'react-router-dom';
```

```
import { deleteItemFromCartAsync, selectItems, updateCartAsync } from
'../features/cart/cartSlice';
import { useForm } from 'react-hook-form';
import { updateUserAsync } from '../features/user/userSlice';
import { createOrderAsync, selectCurrentOrder, selectOrderStatus } from
'../features/order/orderSlice';
import { selectUserInfo } from '../features/user/userSlice';

import Modal from '../features/common/Modal';
import { MutatingDots } from 'react-loader-spinner';

function CheckoutPage() {
    const dispatch = useDispatch();

    // eslint-disable-next-line no-unused-vars
    const [open, setOpen] = useState(true);
    const [openModal, setOpenModal] = useState(null);

    const currentOrder = useSelector(selectCurrentOrder);
    const loadingStatus = useSelector(selectOrderStatus);

    const items = useSelector(selectItems);
    const totalAmount = items.reduce((amount, item) => {
        let itemPrice = item.product.discountedPrice;
        return itemPrice * item.quantity + amount;
    }, 0);
    const totalItems = items.reduce((total, item) => item.quantity + total,
0);

    const handleQuantity = (e, item) => {
```

```
        dispatch(updateCartAsync({ id: item.id, quantity: +e.target.value
}));
    }
    const handleRemove = (e, itemId) => {
        dispatch(deleteItemFromCartAsync(itemId));
    }

    // eslint-disable-next-line no-unused-vars
    const { register, handleSubmit, reset, formState: { errors } } =
useForm();
    const userInfo = useSelector(selectUserInfo);

    const [selectedAddress, setSelectedAddress] = useState(null);
    const [selectedPaymentMethod, setSelectedPaymentMethod] =
useState('cash');

    const handleAddress = (e) => {
        setSelectedAddress(userInfo.addresses[e.target.value]);
    }
    const handlePaymentMethod = (e) => {
        setSelectedPaymentMethod(e.target.value);
    }

    const handleOrder = () => {
        const order = {
            items: items,
            totalAmount: totalAmount,
            totalItems: totalItems,
            user: userInfo.id,
            selectedPaymentMode: selectedPaymentMethod,
            selectedAddress: selectedAddress,
```

```
            status: 'pending'
        };
        dispatch(createOrderAsync(order));
    }

    return (
        <>
            {!items.length && <Navigate to='/' replace={true} />}
            {currentOrder && currentOrder.selectedPaymentMode === 'cash' &&
<Navigate to={`/order-success/${currentOrder.id}`} replace={true} />}
            {/* {currentOrder && currentOrder.selectedPaymentMode===='card'
&& <Navigate to={`/order-success/${currentOrder.id}`} replace={true} />}
*/}
            {/* {currentOrder && currentOrder.selectedPaymentMode===='card'
&& <Navigate to={`/stripe-checkout`} replace={true} />} */}
            {currentOrder && currentOrder.selectedPaymentMode === 'card' &&
<Navigate to={`/payment-gateway`} replace={true} />}

            {userInfo && <div className="mx-auto max-w-7xl px-4 sm:px-6
lg:px-8">
                {loadingStatus === 'loading' &&
                    <div className='w-full flex justify-center items-
center'>
                        <MutatingDots
                            visible={true}
                            height="100"
                            width="100"
                            color="#4464f2"
                            secondaryColor="#4464f2"
                            radius="12.5"
                            ariaLabel="mutating-dots-loading"
```

```
                            wrapperStyle={{}}
                            wrapperClass=""
                        />'
                    </div>}
                {loadingStatus === 'idle' && <div className="grid grid-
cols-1 gap-x-8 gap-y-10 lg:grid-cols-5">
                    <div className='lg:col-span-3'>
                        <form noValidate className='bg-white px-5 my-6 py-6
rounded-lg' onSubmit={handleSubmit((data) => {
                            dispatch(updateUserAsync({ ...userInfo,
addresses: [...userInfo.addresses, data] }));
                            reset();
                        })}>
                            <div className="space-y-12">
                                <div className="border-b border-gray-900/10
pb-12">
                                    <h2 className="text-3xl font-semibold
leading-7 text-gray-900">Personal Information</h2>
                                    <p className="mt-1 text-sm leading-6
text-gray-600">Use a permanent address where you can receive mail.</p>

                                    <div className="mt-10 grid grid-cols-1
gap-x-6 gap-y-8 sm:grid-cols-6">
                                        <div className="sm:col-span-6">
                                            <label htmlFor="name"
className="block text-sm font-medium leading-6 text-gray-900">
                                                Full name
                                            </label>
                                            <div className="mt-2">
                                                <input
                                                    type="text"
```

```
                                                    {...register('name', {
required: 'Please enter your name' })}
                                                    id="name"
                                                    className="block w-full
rounded-md border-0 py-1.5 text-gray-900 shadow-sm ring-1 ring-inset ring-
gray-300 placeholder:text-gray-400 focus:ring-2 focus:ring-inset
focus:ring-indigo-600 sm:text-sm sm:leading-6"
                                                    />
                                                </div>
                                            </div>

                                            <div className="sm:col-span-3">
                                                <label htmlFor="email"
className="block text-sm font-medium leading-6 text-gray-900">
                                                    Email address
                                                </label>
                                                <div className="mt-2">
                                                    <input
                                                        id="email"
                                                        {...register('email', {
required: 'Please enter your email address' })}
                                                        type="email"
                                                        className="block w-full
rounded-md border-0 py-1.5 text-gray-900 shadow-sm ring-1 ring-inset ring-
gray-300 placeholder:text-gray-400 focus:ring-2 focus:ring-inset
focus:ring-indigo-600 sm:text-sm sm:leading-6"
                                                    />
                                                </div>
                                            </div>

                                            <div className="sm:col-span-3">
```

```
                                            <label htmlFor="phone"
className="block text-sm font-medium leading-6 text-gray-900">
                                                Phone number
                                            </label>
                                            <div className="mt-2">
                                                <input
                                                    id="phone"
                                                    {...register('phone', {
required: 'Please enter your email address' })}
                                                    type="tel"
                                                    className="block w-full
rounded-md border-0 py-1.5 text-gray-900 shadow-sm ring-1 ring-inset ring-
gray-300 placeholder:text-gray-400 focus:ring-2 focus:ring-inset
focus:ring-indigo-600 sm:text-sm sm:leading-6"
                                                />
                                            </div>
                                        </div>

                                        <div className="sm:col-span-3">
                                            <label htmlFor="houseNumber"
className="block text-sm font-medium leading-6 text-gray-900">
                                                House Number
                                            </label>
                                            <div className="mt-2">
                                                <input
                                                    type="text"

{...register('houseNumber', { required: 'Please enter your house number'
})}
                                                    id="houseNumber"
```

```
                                                        className="block w-full
rounded-md border-0 py-1.5 text-gray-900 shadow-sm ring-1 ring-inset ring-
gray-300 placeholder:text-gray-400 focus:ring-2 focus:ring-inset
focus:ring-indigo-600 sm:text-sm sm:leading-6"
                                                    />
                                                </div>
                                            </div>

                                            <div className="sm:col-span-3">
                                                <label htmlFor="locality"
className="block text-sm font-medium leading-6 text-gray-900">
                                                    Locality
                                                </label>
                                                <div className="mt-2">
                                                    <input
                                                        type="text"

{...register('locality', { required: 'Please enter your Locality' })}
                                                        id="locality"
                                                        className="block w-full
rounded-md border-0 py-1.5 text-gray-900 shadow-sm ring-1 ring-inset ring-
gray-300 placeholder:text-gray-400 focus:ring-2 focus:ring-inset
focus:ring-indigo-600 sm:text-sm sm:leading-6"
                                                    />
                                                </div>
                                            </div>

                                            <div className="sm:col-span-2
sm:col-start-1">
                                                <label htmlFor="city"
className="block text-sm font-medium leading-6 text-gray-900">
```

```
                                                    City
                                                </label>
                                                <div className="mt-2">
                                                    <input
                                                        type="text"
                                                        {...register('city', {
required: 'Please enter your country' })}
                                                        id="city"
                                                        className="block w-full
rounded-md border-0 py-1.5 text-gray-900 shadow-sm ring-1 ring-inset ring-
gray-300 placeholder:text-gray-400 focus:ring-2 focus:ring-inset
focus:ring-indigo-600 sm:text-sm sm:leading-6"
                                                    />
                                                </div>
                                            </div>

                                            <div className="sm:col-span-2">
                                                <label htmlFor="state"
className="block text-sm font-medium leading-6 text-gray-900">
                                                    State / Province
                                                </label>
                                                <div className="mt-2">
                                                    <input
                                                        type="text"
                                                        {...register('state', {
required: 'Please enter your state' })}
                                                        id="state"
                                                        className="block w-full
rounded-md border-0 py-1.5 text-gray-900 shadow-sm ring-1 ring-inset ring-
gray-300 placeholder:text-gray-400 focus:ring-2 focus:ring-inset
focus:ring-indigo-600 sm:text-sm sm:leading-6"
```

```
                                                    />
                                                </div>
                                            </div>

                                            <div className="sm:col-span-2">
                                                <label htmlFor="pinCode"
className="block text-sm font-medium leading-6 text-gray-900">
                                                    PIN / Postal code
                                                </label>
                                                <div className="mt-2">
                                                    <input
                                                        type="text"
                                                        {...register('pinCode',
{ required: 'Please enter your PIN code' })}
                                                        id="pinCode"
                                                        className="block w-full
rounded-md border-0 py-1.5 text-gray-900 shadow-sm ring-1 ring-inset ring-
gray-300 placeholder:text-gray-400 focus:ring-2 focus:ring-inset
focus:ring-indigo-600 sm:text-sm sm:leading-6"
                                                    />
                                                </div>
                                            </div>
                                        </div>
                                    </div>

                                    <div className="mt-5 flex items-center
justify-end gap-x-6">
                                        <button type="button" className="text-
sm font-semibold leading-6 text-gray-900 px-3 py-2 rounded-md bg-gray-100">
                                            Reset
                                        </button>
```

```
                                    <button
                                        type="submit"
                                        className="rounded-md bg-indigo-600
px-3 py-2 text-sm font-semibold text-white shadow-sm hover:bg-indigo-500
focus-visible:outline focus-visible:outline-2 focus-visible:outline-offset-
2 focus-visible:outline-indigo-600"
                                    >
                                        Add address
                                    </button>
                                </div>

                                <div className="border-b border-gray-900/10
pb-12">
                                    <h2 className="text-base font-semibold
leading-7 text-gray-900">Address</h2>
                                    <p className="mt-1 text-sm leading-6
text-gray-600">
                                        Choose from existing saved
addresses.
                                    </p>

                                    <ul className="divide-y divide-gray-
300">
                                        {userInfo.addresses.map((address,
index) => (
                                            <li key={index} className="flex
justify-between gap-x-6 py-5">
                                                <div className="flex min-w-
0 gap-x-4">
                                                    <input
                                                        id={address.city}
```

```
                                                                    name="address"
                                                                    type="radio"

onChange={handleAddress}
                                                                    value={index}
                                                                    className="h-4 w-4
border-gray-400 text-indigo-600 focus:ring-indigo-600 mt-1 cursor-pointer"
                                                                />
                                                                <div className="min-w-0
flex-auto">
                                                                    <p className="text-
sm font-semibold leading-6 text-gray-900">{address.name}</p>
                                                                    <p className="mt-1
truncate text-xs leading-5 text-gray-500">Phone: {address.phone}</p>
                                                                    <p className="mt-1
truncate text-xs leading-5 text-gray-500">Email: {address.email}</p>
                                                                </div>
                                                            </div>
                                                            <div className="hidden
shrink-0 sm:flex sm:flex-col sm:items-end">
                                                                <p className="text-sm
leading-6 text-gray-700">{address.houseNumber} - {address.locality}</p>
                                                                <p className="text-sm
leading-6 text-gray-500">{address.city}</p>
                                                                <p className="text-sm
leading-6 text-gray-500">{address.state} - {address.pinCode}</p>
                                                            </div>
                                                        </li>
                                                    ))}
                                                </ul>
```

```
                                    <div className="mt-10 space-y-10">
                                        <fieldset>
                                            <legend className="text-sm
font-semibold leading-6 text-gray-900">Payment Method</legend>
                                            <p className="mt-1 text-sm
leading-6 text-gray-600">Choose how you would like to pay.</p>
                                            <div className="mt-6 space-y-
6">
                                                <div className="flex items-
center gap-x-3">
                                                    <input
                                                        id="cash"
                                                        name="payment"
                                                        type="radio"

onChange={handlePaymentMethod}
                                                        value='cash'

checked={selectedPaymentMethod === 'cash'}
                                                        className="h-4 w-4
border-gray-400 text-indigo-600 focus:ring-indigo-600 cursor-pointer"
                                                    />
                                                    <label htmlFor="cash"
className="block text-sm font-medium leading-6 text-gray-900">
                                                        Cash Payment
                                                    </label>
                                                </div>
                                                <div className="flex items-
center gap-x-3">
                                                    <input
                                                        id="card"
```

```
                                                                    name="payment"
                                                                    type="radio"

onChange={handlePaymentMethod}
                                                                    value='card'

checked={selectedPaymentMethod === 'card'}
                                                                    className="h-4 w-4
border-gray-400 text-indigo-600 focus:ring-indigo-600 cursor-pointer"
                                                                />
                                                                <label htmlFor="card"
className="block text-sm font-medium leading-6 text-gray-900">
                                                                    Card Payment
                                                                </label>
                                                            </div>
                                                        </div>
                                                    </fieldset>
                                                </div>
                                            </div>
                                        </div>



                                    </form>
                                </div>
                                <div className='lg:col-span-2'>
                                    <div className='mx-auto m-6 max-w-7xl px-4 sm:px-0
lg:px-0 bg-white rounded-lg'>
                                        <h2 className='text-3xl font-bold tracking-
tight text-gray-900 pt-5 text-left mx-4'>Cart</h2>
                                        <p className="mt-1 text-xs leading-6 text-gray-
600 px-4 mb-5">Review your cart for one last time</p>
```

```
                            <div className="border-t border-gray-200 px-4
py-6 sm:px-6">
                                <div className="flow-root">
                                    <ul className="-my-6 divide-y divide-
gray-200">
                                        {items.map((item) => (
                                            <li key={item.id}
className="flex py-6">
                                                <div className="h-24 w-24
flex-shrink-0 overflow-hidden rounded-md border border-gray-200">
                                                    <img

src={item.product.thumbnail}

alt={item.product.title}
                                                        className="h-full
w-full object-cover object-center"
                                                    />
                                                </div>

                                                <div className="ml-4 flex
flex-1 flex-col">
                                                    <div>
                                                        <div
className="flex justify-between text-base font-medium text-gray-900">
                                                            <h3>
                                                                <a
href={item.product.thumbnail}>{item.product.title}</a>
                                                            </h3>
                                                            <p
className="ml-4">${item.product.discountedPrice}</p>
```

```
                                                                    </div>
                                                                    <p className="mt-1
text-sm text-gray-500">{item.product.brand}</p>
                                                                </div>
                                                                <div className="flex
flex-1 items-end justify-between text-sm">
                                                                    <div
className="text-gray-500">
                                                                        <label
htmlFor="quantity" className="inline mr-5 text-sm font-medium leading-6
text-gray-900">Qty {item.quantity}</label>
                                                                        <select
name="quantity" id="quantity" className='text-sm' onChange={(e) =>
handleQuantity(e, item)} value={item.quantity}>
                                                                            <option
value="1">1</option>
                                                                            <option
value="2">2</option>
                                                                            <option
value="3">3</option>
                                                                            <option
value="4">4</option>
                                                                            <option
value="5">5</option>
                                                                        </select>
                                                                    </div>


                                                                    <div
className="flex">
                                                                        <Modal
```

```
title={'Remove item'}

message={`Are you sure you want to remove ${item.title} from the cart?`}

dangerOption='Remove'

cancelOption='Cancel'

dangerAction={(e) => handleRemove(e, item.id)}

cancelAction={(e) => setOpenModal(-1)}

showModal={openModal === item.id}
                                                                    />

                                                                    <button

onClick={(e) => setOpenModal(item.id)}

type="button"

className="font-medium text-indigo-400 hover:text-indigo-600"
                                                                    >
                                                                        Remove
                                                                    </button>
                                                                </div>
                                                            </div>
                                                        </div>
                                                    </li>
                                                ))}
```

```
                                    </ul>
                                </div>
                            </div>

                            <div className="border-t border-gray-200 px-4
py-6 sm:px-6">
                                <div className="flex justify-between text-
base font-medium text-gray-900 my-2">
                                    <p>Subtotal</p>
                                    <p>${totalAmount}</p>
                                </div>
                                <div className="flex justify-between text-
base font-medium text-gray-900 my-2">
                                    <p>Total Items in Cart</p>
                                    <p>{totalItems}</p>
                                </div>
                                <p className="mt-0.5 text-sm text-gray-
500">Shipping and taxes calculated at checkout.</p>
                                <div className="mt-6">
                                    <div
                                        onClick={handleOrder}
                                        className="flex items-center
justify-center rounded-md border border-transparent bg-indigo-600 px-6 py-3
text-base font-medium text-white shadow-sm hover:bg-indigo-700 cursor-
pointer"
                                    >
                                        Confirm Order
                                    </div>
                                </div>
                                <div className="mt-6 flex justify-center
text-center text-sm text-gray-500">
```

```
                                        <p>
                                            or{' '}
                                            <Link to="/">
                                                <button
                                                    type="button"
                                                    className="font-medium
text-indigo-600 hover:text-indigo-500"
                                                    onClick={() =>
setOpen(false)}
                                                >
                                                    Continue Shopping
                                                    <span aria-hidden="true">
&rarr;</span>
                                                </button>
                                            </Link>
                                        </p>
                                    </div>
                                </div>
                            </div>
                        </div>}
                    </div>}
                </>
            )
        }

        export default CheckoutPage;
```

ContactUsPage.js

```
import React from 'react';
import Navbar from '../features/navbar/Navbar';
import ContactUs from '../features/query/ContactUs';

function ContactUsPage() {
    return (
        <div>
            <Navbar>
                <ContactUs />
            </Navbar>
        </div>
    )
}
export default ContactUsPage;
```

ForgotPasswordPage.js

```
import React from 'react';
import ForgotPassword from '../features/auth/components/ForgotPassword';

function ForgotPasswordPage() {
    return (
        <div>
            <ForgotPassword />
        </div>
    )
}
export default ForgotPasswordPage;
```

Homepage.js

```
import React from 'react';
import Navbar from '../features/navbar/Navbar';
import ProductList from '../features/product/components/ProductList';
import Footer from '../features/common/Footer';

function Homepage() {
    return (
        <div>
            <Navbar>
                <ProductList />
            </Navbar>
            <Footer />
        </div>
    )
}
export default Homepage;
```

LoginPage.js

```
import React from 'react';
import Login from '../features/auth/components/Login';
function LoginPage() {
    return (
        <div>
            <Login />
        </div>
    )
}
export default LoginPage;
```

OrderSuccessPage.js

```
import React, { useEffect } from 'react';
import { useDispatch } from 'react-redux';
import { Link, Navigate, useParams } from 'react-router-dom';
import { resetCartAsync } from '../features/cart/cartSlice';
import { resetOrder } from '../features/order/orderSlice';

function OrderSuccessPage() {
    const params = useParams();
    const dispatch = useDispatch();



    useEffect(() => {
        dispatch(resetCartAsync());
        dispatch(resetOrder());
    }, [dispatch]);

    return (
        <>
            {!params.id && <Navigate to='/' replace={true} />}
            <div className='grid min-h-screen place-items-center bg-white
px-6 py-24 sm:py-32 lg:px-8'>
                <div className="text-center">
                    <p className="text-base font-semibold text-indigo-
600">Order placed successfully!</p>
                    <h1 className="mt-4 text-3xl font-bold tracking-tight
text-gray-900 sm:text-5xl">Order Number : #{params?.id}</h1>
                    <p className="mt-6 text-base leading-7 text-gray-
600">You can check your order in My Account - My orders</p>
```

```
                    <div className="mt-10 flex items-center justify-center
gap-x-6">
                        <Link
                            to="/"
                            className="rounded-md bg-indigo-600 px-3.5 py-
2.5 text-sm font-semibold text-white shadow-sm hover:bg-indigo-500 focus-
visible:outline focus-visible:outline-2 focus-visible:outline-offset-2
focus-visible:outline-indigo-600"
                        >
                            Go back home
                        </Link>
                        <a href="https://github.com/abhik2207/Vendr-
Development" className="text-sm font-semibold text-gray-900">
                            Contact support <span aria-
hidden="true">&rarr;</span>
                        </a>
                    </div>
                </div>
            </div>
        </>
    )
}
export default OrderSuccessPage;
```

PageNotFound.js

```
import React from 'react';
import { Link } from 'react-router-dom';

function PageNotFound() {
```

```
    return (
        <div className='grid min-h-screen place-items-center bg-white px-6
py-24 sm:py-32 lg:px-8'>
            <div className="text-center">
                <p className="text-base font-semibold text-indigo-
600">404</p>
                <h1 className="mt-4 text-3xl font-bold tracking-tight text-
gray-900 sm:text-5xl">Page not found</h1>
                <p className="mt-6 text-base leading-7 text-gray-
600">Sorry, we couldn’t find the page you’re looking for.</p>
                <div className="mt-10 flex items-center justify-center gap-
x-6">
                    <Link to="/"
                        className="rounded-md bg-indigo-600 px-3.5 py-2.5
text-sm font-semibold text-white shadow-sm hover:bg-indigo-500 focus-
visible:outline focus-visible:outline-2 focus-visible:outline-offset-2
focus-visible:outline-indigo-600"
                    >
                        Go back home
                    </Link>
                    <a href="https://github.com/abhik2207/Vendr-
Development" className="text-sm font-semibold text-gray-900">
                        Contact support <span aria-
hidden="true">&rarr;</span>
                    </a>
                </div>
            </div>
        </div>
    )
}
export default PageNotFound;
```

PaymentPage.js

```
import React, { useState } from 'react';
import { CreditCard, PaymentForm } from 'react-square-web-payments-sdk';
import { Navigate } from "react-router-dom";
import { useSelector } from 'react-redux';
import { selectCurrentOrder } from '../features/order/orderSlice';

import '../Payment.css';

function PaymentPage() {
    const [paymentSuccessToken, setPaymentSuccessToken] = useState('');
    const currentOrder = useSelector(selectCurrentOrder);

    return (
        <div id='payment-page'>
            {paymentSuccessToken !== '' && <Navigate to={`/order-
success/${currentOrder.id}`} replace={true}></Navigate>}

            <h1>Pay with your Card</h1>
            <div id="payment-form">
                <PaymentForm
                    /**
                     * Identifies the calling form with a verified
application ID generated from
                     * the Square Application Dashboard.
                     */
                    applicationId="sandbox-sq0idb-dxbQH8c9ntE_7QV6pyF8og"
                    /**
                     * Invoked when payment form receives the result of a
tokenize generation
```

```
                     * request. The result will be a valid credit card or
wallet token, or an error.
                     */
                    cardTokenizeResponseReceived={async (token, buyer) => {
                        const myToken = await token;
                        setPaymentSuccessToken(myToken);
                        console.log({ token, buyer });
                    }}
                    /**
                     * This function enable the Strong Customer
Authentication (SCA) flow
                     *
                     * We strongly recommend use this function to verify
the buyer and reduce
                     * the chance of fraudulent transactions.
                     */
                    createVerificationDetails={() => ({
                        amount: (`${currentOrder.totalAmount}`),
                        /* collected from the buyer */
                        billingContact: {
                            addressLines: ['123 Main Street', 'Apartment
1'],
                            familyName: 'Doe',
                            givenName: 'John',
                            countryCode: 'GB',
                            city: 'London',
                        },
                        currencyCode: 'INR',
                        intent: 'CHARGE',
                    })}
                    /**
```

```
                     * Identifies the location of the merchant that is
taking the payment.
                     * Obtained from the Square Application Dashboard -
Locations tab.
                     */
                    locationId="LVN8DKK0HRF0S"
                >
                    <CreditCard />
                </PaymentForm>
            </div>
        </div>
    )
}
export default PaymentPage;
```

ProductDetailPage.js

```
import React from 'react';
import Navbar from '../features/navbar/Navbar';
import ProductDetail from '../features/product/components/ProductDetail';
function ProductDetailPage() {
    return (
        <div>
            <Navbar>
                <ProductDetail />
            </Navbar>
        </div>
    )
}
export default ProductDetailPage;
```

ResetPasswordPage.js

```
import React from 'react';
import ResetPassword from '../features/auth/components/ResetPassword';

function ResetPasswordPage() {
    return (
        <div>
            <ResetPassword />
        </div>
    )
}
export default ResetPasswordPage;
```

SignupPage.js

```
import React from 'react';
import Signup from '../features/auth/components/Signup';

function SignupPage() {
    return (
        <div>
            <Signup />
        </div>
    )
}
export default SignupPage;
```

UserOrdersPage.js

```
import React from 'react';
import Navbar from '../features/navbar/Navbar';
import UserOrders from '../features/user/components/UserOrders';
function UserOrdersPage() {
    return (
        <div>
            <Navbar>
                <UserOrders />
            </Navbar>
        </div>
    )
}
export default UserOrdersPage;
```

UserProfilePage.js

```
import React from 'react';
import Navbar from '../features/navbar/Navbar';
import UserProfile from '../features/user/components/UserProfile';
function UserProfilePage() {
    return (
        <div>
            <Navbar> <UserProfile /> </Navbar>
        </div>
    )
}
export default UserProfilePage;
```

OUTPUT SCREENSHOTS

Below, we have included comprehensive screenshots that illustrate the various functionalities and user interfaces of our project. These screenshots cover major pages such as the Login page, Signup page, Forgot Password page, Cart page, Checkout page, Payments Gateway page, Order Success page, About Us page, Contact Us page, My Profile page, and My Orders page. Additionally, the screenshots showcase administrative functionalities with the Admin Products page, Admin Queries page, and Admin Orders page. Important screenshots like the invoice received via email and the reset link received via email for password recovery are also provided to give a complete overview of the user and admin experiences within our application.

Login Page

VENDR - Shop Smart & Easy
vendr-deployment.vercel.app/login
Sign in to your account
Email address
Forgot password?
Password
Log in
Does not have an account? Sign up

Signup Page

VENDR - Shop Smart & Easy
vendr-deployment.vercel.app/signup
Create an account
Email address
Password
Confirm Password
Sign Up
Already have an account? Log in

Home Page

VENDR - Shop Smart & Easy
vendr-deployment.vercel.app
Vendr
Products
About Us
Contact Us
All Products
Sort
Category
Brand
Casual Tshirt
$16
0/5
Men Slim Fit Jeans
$35
0/5
iPhone 15 Pro Max
$1647
0/5
Galaxy S24+ Ultra
Xiaomi 14 Ultra

Product Detail Page

vendr-deployment – Deployme
VENDR - Shop Smart & Easy
vendr-deployment.vercel.app/product-detail/664093f3e1af8d4e0bd72305
Casual Tshirt
100% Original Products. Pay on delivery might be available. Easy 14 days returns and exchanges!
Highlights
100% Cotton
Machine Wash
Vibrant Colors
Comfortable
Details
100% Original Products. Pay on delivery might be available. Easy 14 days returns and exchanges!
$16
$18
Color
Size
Size guide
S
M
L
XL
Add to Cart

Cart Page

vendr-deployment – Deployme
VENDR - Shop Smart & Easy
vendr-deployment.vercel.app/cart
Cart
Dime 3 Pro
Skullcandy
Qty 1
1
$30
Remove
Casual Tshirt
Roadster
Qty 1
1
$16
Remove
Subtotal
$46
Total Items in Cart
2
Shipping and taxes calculated at checkout.
Checkout
or Continue Shopping →

Checkout Page

vendr-deployment – Deployme
VENDR - Shop Smart & Easy
Data | Cloud: MongoDB Cloud
vendr-deployment.vercel.app/checkout
House Number
Locality
City
State / Province
PIN / Postal code
Reset
Add address
Address
Choose from existing saved addresses.
Abhik Gupta (Home)
Phone: 7042053577
Email: abhikgupta01@gmail.com
RZ-29, 2nd Floor - Hanspark, West Sagarpur
New Delhi
Delhi - 110046
Abhik Gupta (Office)
Phone: 1234567890
Email: abhikgupta01@gmail.com
Building 11 - Spartan Commercial Block
Bangalore
Karnataka - 200101
Payment Method
Choose how you would like to pay.
Cash Payment
Card Payment
Casual Tshirt
$16
roadster
Qty 1
1
Remove
Subtotal
$46
Total Items in Cart
2
Shipping and taxes calculated at checkout.
Confirm Order
or Continue Shopping →

Payment Gateway Page

Order Success Page

About us Page

Contact us Page

My profile Page

My orders Page

Invoice of the order received on Gmail

Forgot password Page

Password reset link sent over Gmail

Password reset Page

Admin products page to edit and add products

Admin orders page to view all orders and edit order status

Admin queries page to view all queries by users

EMAIL ADDRESS	QUERY	SENT AT
abhikgupta01@gmail.com	venenatis urna cursus eget nunc scelerisque viverra mauris in aliquam!	5/16/2024, 2:49:18 PM
abhikgupta01@gmail.com	augue interdum velit euismod in pellentesque massa placerat duis ultricies lacus sed turpis tincidunt id aliquet risus feugiat in ante metus dictum at tempor commodo ullamcorper a lacus vestibulum sed...	5/16/2024, 2:49:52 PM
yatin@gmail.com	augue interdum velit euismod in pellentesque massa placerat duis ultricies lacus sed turpis tincidunt id aliquet risus feugiat in ante metus dictum at tempor commodo ullamcorper a lacus vestibulum sed!	5/16/2024, 2:50:18 PM
yatin@gmail.com	bibendum est ultricies integer quis auctor elit sed vulputate mi sit amet mauris commodo quis imperdiet massa tincidunt nunc pulvinar sapien et ligula ullamcorper malesuada proin libero nunc consequat interdum varius sit amet mattis vulputate enim nulla aliquet porttitor lacus luctus accumsan tortor posuere ac ut consequat semper viverra nam libero justo laoreet sit amet cursus sit amet dictum sit?	5/16/2024, 2:50:28 PM
abhikgupta01@gmail.com	lobortis elementum nibh tellus molestie nunc non blandit massa enim nec dui nunc mattis enim ut tellus elementum sagittis vitae!	5/16/2024, 3:03:05 PM
abhikgupta01@gmail.com	in cursus turpis massa tincidunt dui ut ornare lectus sit amet est placerat in egestas erat imperdiet sed euismod nisi.	5/16/2024, 3:03:13 PM
yatin@gmail.com	magna eget est lorem ipsum dolor sit amet consectetur adipiscing elit pellentesque habitant morbi tristique senectus et netus et malesuada...	5/16/2024, 3:03:43 PM
yatin@gmail.com	et netus et malesuada fames ac turpis egestas integer eget.	5/16/2024, 3:03:55 PM
yatin@gmail.com	Test query	5/16/2024, 3:13:34 PM
yatin@gmail.com	hello	5/16/2024, 3:44:59 PM
abhikgupta01@gmail.com	hello world	5/17/2024, 6:55:13 PM

VERSION CONTROL AND COLLABORATION

The development journey of Vendr, a comprehensive e-commerce application, reflects the collaborative efforts of Abhik Gupta, Yatin Hooda, John P. Varghese, and Nishchay Chandok. With Git and GitHub serving as the backbone for collaboration and version control, the team navigated through **112 commits** across **35 major versions**, each marking a significant milestone in the application's evolution.

Development phases

- Versions 1.0 - 18.0: Frontend development using React, Redux, and mock data with json-server and MongoDB Compass.
- Versions 19.0 - 23.0: Backend API development and integration with frontend.
- Versions 24.0 - 26.0: Implementation of authentication and authorization using JWT and Passport JS, with password hashing using Bcrypt.
- Versions 27.0: Initiated Stripe integration for payments gateway, but faced internal errors leading to failure.
- Versions 28.0 - 28.3: Deployment of application over Vercel and migration of database to MongoDB Atlas.
- Version 29.0: Integration of Squareup Payments SDK for payments instead of Stripe.
- Version 30.0: Addition of mailing functionality using Nodemailer for forgot password feature.
- Versions 31.0 - 31.3: Bug fixes.

- Version 32.0: Added email functionality for invoice of products bought.
- Versions 33.0 - 33.5: Continued bug fixes.
- Version 34.0: Added final two pages, About Us and Contact Us.
- Version 35.0: Finalized version and documented the project.

Our github repository

Note: The next 2 pages show our Github repository and the collaborators list of the Github repository.

abhik2207/Vendr-Development
github.com/abhik2207/Vendr-Development
abhik2207 / Vendr-Development
Type / to search
Code
Issues
Pull requests
Actions
Projects
Wiki
Security 2
Insights
Settings
Vendr-Development Public
Unpin
Unwatch 1
Fork 0
Star 0
main
1 Branch
0 Tags
Go to file
Add file
Code
abhik2207 Final report of the project
1df7785 · 27 minutes ago
94 Commits
Extra-Resourcs Initial setup 3 weeks ago
Possible-Favicons Favicons 3 weeks ago
Version 1.0 Initial Setup last month
Version 10.0 Version 10.0 - User Auth API last month
Version 11.0 Version 11.0 - Add to Cart last month
Version 11.1 Version 11.1 - Cart Page last month
Version 12.0 Version 12.0 - Checkout Page last month
Version 12.1 Checkout Page and Order API last month
Version 13.0 Version 13.0 - Order Success and My Products Pages last month
Version 14.0 Version 14.0 - User Profile 3 weeks ago
Version 15.0 Version 15.0 - Logout and Forgot Password 3 weeks ago
Version 16.0 Version 16.0 - Admin Panel Pages 3 weeks ago
About
Vendr is a full stack E-commerce application built using the MERN framework. I am creating this project as my major subject under my undergraduate degree.
Readme
Activity
0 stars
1 watching
0 forks
Releases
No releases published
Create a new release
Packages
No packages published
Publish your first package
Languages

Manage access
github.com/abhik2207/Vendr-Development/settings/access
abhik2207 / Vendr-Development
Type / to search
Code
Issues
Pull requests
Actions
Projects
Wiki
Security 2
Insights
Settings
General
Access
Collaborators
Moderation options
Code and automation
Branches
Tags
Rules
Actions
Webhooks
Environments
Codespaces
Pages
Security
Code security and analysis
Deploy keys
Secrets and variables
Integrations
GitHub Apps
Email notifications
Who has access
PUBLIC REPOSITORY
This repository is public and visible to anyone.
Manage
DIRECT ACCESS
4 have access to this repository. 4 collaborators.
Manage access
Add people
Select all
Type
Find a collaborator...
developer-abhik
Collaborator
nishchaychandok
Collaborator
John Varghese
Sladeslayer • Collaborator
YatinHooda
Collaborator
Get team access controls and discussions for your contributors in an organization.
NEW Private repos and unlimited members are free.
Create an organization

CONCLUSION

Throughout Vendr's development, our team has navigated challenges, embraced innovation, and remained committed to delivering a premier e-commerce solution. Reflecting on our milestones, lessons learned, and the transformative impact Vendr aims to make in the digital marketplace, we see a project born from collective effort, dedication, and passion. Vendr stands as a testament to our pursuit of excellence and our drive to exceed customer expectations. With its user-centric design, seamless functionality, and commitment to security, Vendr sets a new standard for online shopping, empowering users to explore diverse products, streamline order management, and receive personalized recommendations.

As we embark on Vendr's next phase, we remain dedicated to innovation, continuous improvement, and customer satisfaction. Our vision extends beyond current capabilities, with plans to implement advanced features, expand into new markets, and embrace emerging technologies. With an unwavering commitment to pushing boundaries and exceeding expectations, Vendr is poised to lead in the evolving world of e-commerce, empowering users, delighting customers, and shaping the future of digital commerce.

FUTURE SCOPE

In our ongoing pursuit of excellence, Vendr continuously explores avenues for growth and improvement. Looking ahead, we envision a future brimming with innovation and enhanced user experiences. With an unwavering commitment to staying at the forefront of e-commerce trends, we are excited to unveil our ambitious roadmap for future development. Below, we outline our forthcoming enhancements and features, designed to elevate Vendr to new heights of functionality, accessibility, and customer satisfaction.

1. **AI-Powered Recommendations**: Implement artificial intelligence to offer personalized product recommendations based on user behavior, past purchases, and browsing history. This could increase sales and improve customer satisfaction by making the shopping experience more relevant and tailored to individual preferences.
2. **Multi-vendor Marketplace**: Expanding Vendr to support multiple vendors can significantly broaden its scope. This would allow various sellers to register on the platform, list their products, and manage their inventory independently, creating a diverse marketplace for users.
3. **Augmented Reality (AR) Integration**: Develop AR capabilities to allow customers to visualize products in their own environment before making a purchase. This could be particularly useful for items like furniture and home decor, enhancing consumer confidence in their buying decisions.
4. **Expansion to International Markets**: Adapt the platform for international use by incorporating multi language support, multi-currency pricing, and local payment

gateways. This expansion would broaden the customer base and increase revenue opportunities.

5. **Mobile Application Development**: Create a dedicated mobile app for Vendr to provide users with a more convenient, fast, and accessible way to shop and manage their accounts from their smartphones, potentially increasing user engagement and sales.
6. **Blockchain for Enhanced Security**: Explore the use of blockchain technology to enhance transaction security and transparency, especially for high-value transactions and to protect against fraud in payment processes and supply chain documentation.
7. **Voice Commerce Integration**: Integrate voice-activated commerce capabilities, allowing users to search for products, place orders, and manage their accounts through voice assistants like Amazon Alexa, Google Assistant, or Apple Siri. This feature could tap into the growing market of users who rely on smart home devices.
8. **Enhanced Analytics and Insights**: Continuously improving analytics capabilities to gather actionable insights into user behavior, sales trends, and market dynamics can inform strategic decision-making and drive business growth. Implementing advanced analytics tools and predictive analytics can unlock valuable insights for optimizing marketing strategies, product offerings, and user experience.
9. **Dynamic Rating System**: In the future, Vendr aims to implement a dynamic rating system to enhance user engagement and provide more accurate product feedback. Currently, the rating system is static, but the plan is to introduce features such as user reviews, star ratings, and real-time feedback mechanisms. This dynamic rating system will enable users to share their experiences, opinions, and insights about products, fostering a vibrant community-driven environment and aiding other shoppers in making informed purchasing decisions.

REFERENCES

1. MongoDB: https://www.mongodb.com/docs/
2. React.js: https://legacy.reactjs.org/docs/getting-started.html
3. Node.js: https://nodejs.org/docs/latest/api/
4. Express.js: https://expressjs.com/
5. Redux JS: https://redux.js.org/
6. Stack Overflow: https://stackoverflow.com/
7. ChatGPT: https://chatgpt.com/?oai-dm=1
8. Gemini: https://gemini.google.com/app
9. JavaScript: https://developer.mozilla.org/en-US/docs/Web/JavaScript
10. React toastify: https://fkhadra.github.io/react-toastify/installation/
11. React icons: https://react-icons.github.io/react-icons/
12. Tailwind CSS: https://tailwindcss.com/
13. Mongoose JS: https://mongoosejs.com/
14. React Router DOM: https://reactrouter.com/en/main