

Hair body store Sajid Shaikh store

Okay we will make the same app but the domain and field will be different and it will be the

Full stack Hairs Body supplement products weight loss and everything buy affiliate links attached to each product store app

Okay, I am providing you the entire project structure and the code inside each and every file. You have to deeply understand this project and because I am providing you the main reason behind why I am providing you this is because I want you to analyze and thoroughly analyze and learn this project so that you can scale this project for me. Add new features, add more code, add new functionality, add new files into it. Suggest me code, analyze and more features for free including free tools available on the internet and all the tools so that this app can be very better from this MVP.

No errors understand detailed in the intelliJ idea ultimate use the same tech stack in this project first go to the <https://github.com/sajidbaba1/Full-Stack-Sajid-Health-Store-.git> make it origin then make the branch main and add first commit and give me onwards whenever you will add the new feature and when we will complete one small feature then we will add a commit that that particular commit the of the feature with proper commit message give me three lines ad that time to copy and paste like the add and commit and push to main build this app step by step feature by feature same as the pdf code of reference app provided we will make this app more advanced and step by step and feature by feature so there will be no errors

I am providing the pdf containing each and every file of the reference project and the project structure so you will be understand where each file and role of each file were

You have to make this project in keeping this in mind you will be using the java for the project maven build tool and everything same give me postman curl command to test at relevant place so till the time of end of project i will be having the collection document in the postman and make it as much errorless as possible before starting the code do all the unit testing and integration testing by your own before starting giving me the project for the copy pasting so i will be zero errors in the process keep all the common errors and all the possible errors i may get so you to avoid those errors you have do all the testing and all before actually giving me the most precise and best possible code for that feature in this world and keep in mind you have remember all the history of this project from the start you will be start doing this project so it will super easy fro you to actually do the scaling of code and increase the features in future at each code you have to give the comments of detailed explanation of that code block and by keeping mind if provide that code file to any other ai bot it can understand and easily explain me again so you have to maek such comments in it alot while writing the code of the each file explain me what we have done what is this feature why and how we have did the code so till the end of this project I will get a notes of you how you made the project so i can read and revision of it keep documenting the flow and progress and make me very easy for copy paste understand and keep going

At the end give me the readme file and and also make the required env file gitignore and the dockerfile in the project and keep updating it as the project progresses so i can errorlessly create image container and the push to docker hub easily and deploy it at before starting the project imagine and tell me as a architect how you have designed this project why you chose mysql etc stuff and all trade off and this tech stack and all system design like game keep updating the git actions file give detailed comments in each code file so it will be super and detailed and easy to just looking and reading i will able to understand give me readme file at the end and before project give me super detailed notion document as well it will be super detailed and useful and it will contain all the necessary stuff and also how to setup file and all all that such files you can add more files like these we will be putting them in the asset folder and in asset folder there will be folder called as the images in which all the pexel images will be there

Give me step by step project from this text create each and every file of the project in the intellij idea ultimate you have to give me each and every step in the process like even changing the directory before pasting or creating the file such detail and also tell me all the steps like changing the branch before making commit such details even tell me while creating the next js project for example tell me what option to choose javascript or the type script and alias etc and for example for creating the sprinngboot project tell me what should be group artifact build tool project name java version such details give me all of these things while making this project so develop the project step by step step in this manner

read each and every line do what is insturcted

Ecommerce Multi Vendor

Tools

- intelliJ idea
- Vs code
- node js
- my sql

Technology

- spring boot
- mysql
- spring security
- java mail sender
- jwt
- react
- typescript
- redux toolkit
- mui
- tailwind css
- react chart
- formik
- yup
- react router dom

- axios
- Payment Gateway: Razorpay (for indian student) Stripe (for international student)

Features

Customer Features

1. **Chatbot for Queries**
 - A chatbot allows users to ask questions about:
 - **Order History:** View past orders.
 - **Cart:** Inquire about items in their cart.
 - **Product Details:** Ask for detailed information about products.
2. **Product Management**
 - **Fetch Product List:** Users can browse through available products.
 - **Filter & Sort:** Filter products by categories, price, etc., and sort them by price.
 - **Pagination:** Display products in multiple pages to improve performance and user experience.
 - **Product Details:** View detailed information about a specific product.
3. **Cart Management**
 - **Add Item to Cart:** Add products to the shopping cart.
 - **Update Cart Item:** Modify item quantities or remove items from the cart.
4. **Checkout Process**
 - **Apply Coupon:** Users can apply discount coupons to their cart.
 - **Add New Address:** Add and manage shipping addresses during checkout.
 - **Payment Gateways:** Checkout using payment options like Razorpay or Stripe.
5. **Order History**
 - **View Past Orders:** Users can see a list of their previous purchases and order details.
 - cancel order
6. **User Account Management**
 - Manage personal details, view order history, and track account settings.
7. **Review & Rating**
 - Write Review
8. wishlist
 - add and remove product from wishlist

Seller Features

1. **Seller Dashboard**

- Earning Graph (Today, last 7 days, last 12 month), and seller Report
 - 2. **Seller Reports**
 - **Total Sales:** View the total number of products sold.
 - **Total Earnings:** Track overall earnings from sales.
 - **Refunds & Cancellations:** Monitor refunds and canceled orders.
 - 3. **Product Management**
 - **Create Products:** Add new products to the store.
 - **Orders Management:** View and manage customer orders.
 - 4. **Payment & Transactions**
 - **Track Payments:** Monitor incoming payments for orders.
 - **Transaction History:** Detailed history of all transactions.
 - 5. **Seller Account Management**
 - **Profile Management:** Update and manage seller profiles.
-

Admin Features

1. **Admin Dashboard**
2. **Seller Management**
 - Handle all sellers, including approval, and suspension.
3. **Coupon Management**
 - **Create, Edit, Delete Coupons:** Manage discount codes available for customers.
4. **Home Page Management**
 - Update and customize the home page through the admin panel.
5. **Deal Management**
 - Create and manage promotional deals and offers on products.

Entity Relationships

1. **User**
 - One-to-Many with **Address:** A user can have multiple addresses.
 - Many-to-Many with **Coupon:** Users can use multiple coupons, and a coupon can be used by multiple users.
 - One-to-Many with **Cart:** Each user has one cart.
 - One-to-Many with **Order:** A user can have multiple orders.
 - One-to-Many with **Review:** A user can leave multiple reviews.
 - One-to-Many with **Transaction:** A user can have multiple transactions.

- One-to-One with **Wishlist**: Each user has one wishlist.
2. **Address**
- Many-to-One with **User**: An address belongs to one user.
 - Many-to-One with **Order**: An order has one shipping address.
3. **Cart**
- One-to-One with **User**: Each user has one cart.
 - One-to-Many with **CartItem**: A cart can contain multiple cart items.
4. **CartItem**
- Many-to-One with **Cart**: A cart item belongs to one cart.
 - Many-to-One with **Product**: A cart item refers to one product.
5. **Product**
- Many-to-One with **Category**: A product belongs to one category.
 - Many-to-One with **Seller**: A product is sold by one seller.
 - One-to-Many with **Review**: A product can have multiple reviews.
6. **Category**
- Many-to-One with **Category**: A category can have a parent category (for subcategories).
7. **Coupon**
- Many-to-Many with **User**: A coupon can be used by multiple users.
8. **Order**
- Many-to-One with **User**: An order belongs to one user.
 - One-to-Many with **OrderItem**: An order can have multiple order items.
 - Many-to-One with **Address**: An order has one shipping address.
9. **OrderItem**
- Many-to-One with **Order**: An order item belongs to one order.
 - Many-to-One with **Product**: An order item refers to one product.
10. **PaymentOrder**
- Many-to-One with **User**: A payment order belongs to one user.
 - One-to-Many with **Order**: A payment order can include multiple orders.
11. **Seller**
- One-to-One with **Address**: A seller has one pickup address.
 - One-to-Many with **Product**: A seller can sell multiple products.
 - One-to-Many with **Transaction**: A seller can be involved in multiple transactions.
12. **Transaction**
- Many-to-One with **User**: A transaction is associated with one user.
 - Many-to-One with **Seller**: A transaction is associated with one seller.
 - One-to-One with **Order**: A transaction corresponds to one order.
13. **Review**
- Many-to-One with **Product**: A review is for one product.
 - Many-to-One with **User**: A review is written by one user.
14. **Wishlist**
- One-to-One with **User**: Each user has one wishlist.
 - Many-to-Many with **Product**: A wishlist can contain multiple products.
15. **VerificationCode**

- One-to-One with **User**: A verification code can be associated with one user.
- One-to-One with **Seller**: A verification code can be associated with one seller.

16. **SellerReport**

- One-to-One with **Seller**: A report corresponds to one seller.

Connect Frontend With Backend

User Interaction

- User makes a request (e.g., button click) in the React component.

API Call

- React component sends a request to the backend API.

Backend Processing

- Backend processes the request and retrieves data from the database or other source.

Response from API

- Backend sends the requested data back to the React component as a response.

Store Data in Local State

- React component receives the response and stores the data in local state (using useState or similar).

Render Data

- React component renders the data in the UI.

Project Step

- set up mysql database
- create first api
- create entity model
- spring security (login, signup)
- service and impl
- controller
- test all api end points

Okay we will make the same app but the domain and field will be different and it will be the

Full stack Hairs Body supplement products weight loss and everything buy affiliate links attached to each product store app

Okay, I am providing you the entire project structure and the code inside each and every file. You have to deeply understand this project and because I am providing you the main reason behind why I am providing you this is because I want you to analyze and thoroughly analyze and learn this project so that you can scale this project for me. Add new features, add more code, add new functionality, add new files into it. Suggest me code, analyze and more features for free including free tools available on the internet and all the tools so that this app can be very better from this MVP.

No errors understand detailed in the intelliJ idea ultimate use the same tech stack in this project first go to the <https://github.com/sajidbaba1/Full-Stack-Sajid-Health-Store-.git> make it origin then make the branch main and add first commit and give me onwards whenever you will add the new feature and when we will complete one small feature then we will add a commit that that particular commit the of the feature with proper commit message give me three lines ad that time to copy and paste like the add and commit and push to main build this app step by step feature by feature same as the pdf code of reference app provided we will make this app more advanced and step by step and feature by feature so there will be no errors

I am providing the pdf containing each and every file of the reference project and the project structure so you will be understand where each file and role of each file were

You have to make this project in keeping this in mind you will be using the java for the project maven build tool and everything same give me postman curl command to test at relevant place so till the time of end of project i will be having the collection document in the postman and make it as much errorless as possible before starting the code do all the unit testing and integration testing by your own before starting giving me the project for the copy pasting so i will be zero errors in the process keep all the common errors and all the possible errors i may get so you to avoid those errors you have do all the testing and all before actually giving me the most precise and best possible code for that feature in this world and keep in mind you have remember all the history of this project from the start you will be start doing this project so it will super easy fro you to actually do the scaling of code and increase the features in future at each code you have to give the comments of detailed explanation of that code block and by keeping mind if provide that code file to any other ai bot it can understand and easily explain me again so you have to maek such comments in it alot while writing the code of the each file explain me what we have done what is this feature why and how we have did the code so till the end of this

project I will get a notes of you how you made the project so i can read and revision of it keep documenting the flow and progress and make me very easy for copy paste understand and keep going

At the end give me the readme file and and also make the required env file gitignore and the dockerfile in the project and keep updating it as the project progresses so i can errorlessly create image container and the push to docker hub easily and deploy it at before starting the project imagine and tell me as a architect how you have designed this project why you chose mysql etc stuff and all trade off and this tech stack and all system design like game keep updating the git actions file give detailed comments in each code file so it will be super and detailed and easy to just looking and reading i will able to understand give me readme file at the end and before project give me super detailed notion document as well it will be super detailed and useful and it will contain all the necessary stuff and also how to setup file and all all that such files you can add more files like these we will be putting them in the asset folder and in asset folder there will be folder called as the images in which all the pexel images will be there

Give me step by step project from this text create each and every file of the project in the intellij idea ultimate you have to give me each and every step in the process like even changing the directory before pasting or creating the file such detail and also tell me all the steps like changing the branch before making commit such details even tell me while creating the next js project for example tell me what option to choose javascript or the type script and alias etc and for example for creating the sprinngboot project tell me what should be group artifact build tool project name java version such details give me all of these things while making this project so develop the project step by step step in this manner

read each and every line do what is insturcted

Ecommerce Multi Vendor

Tools

- intelliJ idea
- Vs code
- node js
- my sql

Technology

- spring boot
- mysql
- spring security
- java mail sender
- jwt
- react
- typescript
- redux toolkit
- mui
- tailwind css
- react chart
- formik

- yup
- react router dom
- axios
- Payment Gateway: Razorpay (for indian student) Stripe (for international student)

Features

Customer Features

1. **Chatbot for Queries**
 - A chatbot allows users to ask questions about:
 - **Order History:** View past orders.
 - **Cart:** Inquire about items in their cart.
 - **Product Details:** Ask for detailed information about products.
2. **Product Management**
 - **Fetch Product List:** Users can browse through available products.
 - **Filter & Sort:** Filter products by categories, price, etc., and sort them by price.
 - **Pagination:** Display products in multiple pages to improve performance and user experience.
 - **Product Details:** View detailed information about a specific product.
3. **Cart Management**
 - **Add Item to Cart:** Add products to the shopping cart.
 - **Update Cart Item:** Modify item quantities or remove items from the cart.
4. **Checkout Process**
 - **Apply Coupon:** Users can apply discount coupons to their cart.
 - **Add New Address:** Add and manage shipping addresses during checkout.
 - **Payment Gateways:** Checkout using payment options like Razorpay or Stripe.
5. **Order History**
 - **View Past Orders:** Users can see a list of their previous purchases and order details.
 - cancel order
6. **User Account Management**
 - Manage personal details, view order history, and track account settings.
7. **Review & Rating**
 - Write Review
8. **wishlist**
 - add and remove product from wishlist

Seller Features

1. **Seller Dashboard**
 - Earning Graph (Today, last 7 days, last 12 month), and seller Report
 2. **Seller Reports**
 - **Total Sales:** View the total number of products sold.
 - **Total Earnings:** Track overall earnings from sales.
 - **Refunds & Cancellations:** Monitor refunds and canceled orders.
 3. **Product Management**
 - **Create Products:** Add new products to the store.
 - **Orders Management:** View and manage customer orders.
 4. **Payment & Transactions**
 - **Track Payments:** Monitor incoming payments for orders.
 - **Transaction History:** Detailed history of all transactions.
 5. **Seller Account Management**
 - **Profile Management:** Update and manage seller profiles.
-

Admin Features

1. **Admin Dashboard**
2. **Seller Management**
 - Handle all sellers, including approval, and suspension.
3. **Coupon Management**
 - **Create, Edit, Delete Coupons:** Manage discount codes available for customers.
4. **Home Page Management**
 - Update and customize the home page through the admin panel.
5. **Deal Management**
 - Create and manage promotional deals and offers on products.

Entity Relationships

1. **User**
 - One-to-Many with **Address:** A user can have multiple addresses.
 - Many-to-Many with **Coupon:** Users can use multiple coupons, and a coupon can be used by multiple users.
 - One-to-Many with **Cart:** Each user has one cart.

- One-to-Many with **Order**: A user can have multiple orders.
 - One-to-Many with **Review**: A user can leave multiple reviews.
 - One-to-Many with **Transaction**: A user can have multiple transactions.
 - One-to-One with **Wishlist**: Each user has one wishlist.
2. **Address**
- Many-to-One with **User**: An address belongs to one user.
 - Many-to-One with **Order**: An order has one shipping address.
3. **Cart**
- One-to-One with **User**: Each user has one cart.
 - One-to-Many with **CartItem**: A cart can contain multiple cart items.
4. **CartItem**
- Many-to-One with **Cart**: A cart item belongs to one cart.
 - Many-to-One with **Product**: A cart item refers to one product.
5. **Product**
- Many-to-One with **Category**: A product belongs to one category.
 - Many-to-One with **Seller**: A product is sold by one seller.
 - One-to-Many with **Review**: A product can have multiple reviews.
6. **Category**
- Many-to-One with **Category**: A category can have a parent category (for subcategories).
7. **Coupon**
- Many-to-Many with **User**: A coupon can be used by multiple users.
8. **Order**
- Many-to-One with **User**: An order belongs to one user.
 - One-to-Many with **OrderItem**: An order can have multiple order items.
 - Many-to-One with **Address**: An order has one shipping address.
9. **OrderItem**
- Many-to-One with **Order**: An order item belongs to one order.
 - Many-to-One with **Product**: An order item refers to one product.
10. **PaymentOrder**
- Many-to-One with **User**: A payment order belongs to one user.
 - One-to-Many with **Order**: A payment order can include multiple orders.
11. **Seller**
- One-to-One with **Address**: A seller has one pickup address.
 - One-to-Many with **Product**: A seller can sell multiple products.
 - One-to-Many with **Transaction**: A seller can be involved in multiple transactions.
12. **Transaction**
- Many-to-One with **User**: A transaction is associated with one user.
 - Many-to-One with **Seller**: A transaction is associated with one seller.
 - One-to-One with **Order**: A transaction corresponds to one order.
13. **Review**
- Many-to-One with **Product**: A review is for one product.
 - Many-to-One with **User**: A review is written by one user.
14. **Wishlist**

- One-to-One with **User**: Each user has one wishlist.
- Many-to-Many with **Product**: A wishlist can contain multiple products.

15. **VerificationCode**

- One-to-One with **User**: A verification code can be associated with one user.
- One-to-One with **Seller**: A verification code can be associated with one seller.

16. **SellerReport**

- One-to-One with **Seller**: A report corresponds to one seller.

Connect Frontend With Backend

User Interaction

- User makes a request (e.g., button click) in the React component.

API Call

- React component sends a request to the backend API.

Backend Processing

- Backend processes the request and retrieves data from the database or other source.

Response from API

- Backend sends the requested data back to the React component as a response.

Store Data in Local State

- React component receives the response and stores the data in local state (using useState or similar).

Render Data

- React component renders the data in the UI.

Project Step

- set up mysql database
- create first api
- create entity model
- spring security (login, signup)
- service and impl
- controller
- test all api end points

Complete Project Structure with File Descriptions

Here's a comprehensive breakdown of the project structure, including all files and their purposes:

Root Structure

```
Ecommerce-Multi-Vendor-Project/
├── .git/                      # Main project directory
├── .idea/                     # IntelliJ IDEA configuration
├── assets/                    # Static assets (images, icons, etc.)
│   └── ... (various asset files) and images
└── source code/
    ├── backend-spring-boot/    # Main source code directory
    │   ├── frontend-vite/      # Backend implementation
    │   └── frontend-vite/      # Frontend implementation
```

Backend Structure (Spring Boot)

```
backend-spring-boot/
├── .gitignore                 # Git ignore configuration
├── .mvn/                      # Maven wrapper files
├── Dockerfile                  # Docker configuration for backend
├── mvnw                        # Maven wrapper (Unix)
├── mvnw.cmd                    # Maven wrapper (Windows)
├── pom.xml                     # Maven project configuration
└── src/
    └── main/
        ├── java/com/zosh/      # Main application package
        │   ├── EcommerceMultiVendorApplication.java # Main application class
        │   └── ai/               # AI/ML related functionality
```

```
    |   |   └── GeminiService.java           # Integration with
Gemini AI
    |   |   └── ... (other AI service implementations)
    |
    |   ├── config/                      # Spring configuration classes
    |   |   ├── JwtAuthenticationFilter.java      # JWT authentication
    |   |   ├── SecurityConfig.java            # Security configuration
    |   |   ├── WebConfig.java               # Web configuration
    |   |   └── WebMvcConfig.java          # MVC configuration
    |
    |   ├── controller/                 # REST API controllers
    |   |   ├── AdminController.java        # Admin operations
    |   |   ├── AuthController.java       # Authentication
    |
    | endpoints
    |   |   └── CartController.java        # Shopping cart
    |
    | operations
    |   |   └── OrderController.java      # Order management
    |   |   └── ProductController.java    # Product CRUD
    |
    | operations
    |   |   └── ReviewController.java     # Product reviews
    |   |   └── ... (other controller files)
    |
    |   ├── domain/                     # Domain models
    |   |   ├── Address.java             # User address
    |   |   ├── Cart.java               # Shopping cart
    |   |   ├── Order.java              # Order details
    |   |   ├── Product.java            # Product details
    |   |   └── ... (other domain models)
    |
    |   ├── dto/                        # Data Transfer Objects
    |   |   ├── AuthResponse.java        # Authentication
    |
    response
    |   |   └── ProductDto.java          # Product data transfer
    |   |   └── ... (other DTOs)
    |
    |   ├── exception/                # Custom exceptions
    |   |   ├── ProductException.java    # Product-related errors
    |   |   ├── UserException.java      # User-related errors
    |   |   └── ... (other exceptions)
    |
    |   ├── mapper/                   # Object mappers
    |   |   ├── ProductMapper.java      # Product mapping
```

```
|   |   └ ... (other mappers)
|   |
|   └ model/           # Data models
|       ├── Address.java          # Address model
|       ├── CartItem.java         # Cart item model
|       ├── Category.java         # Category model
|       ├── Order.java            # Order model
|       ├── Product.java          # Product model
|       └ ... (other model classes)
|
|   └ repository/        # Data access layer
|       ├── CartRepository.java    # Cart data access
|       ├── OrderRepository.java   # Order data access
|       ├── ProductRepository.java # Product data access
|       └ ... (other repositories)
|
|   └ request/          # Request objects
|       ├── AddItemRequest.java    # Add item to cart
|       ├── CreateProductRequest.java # Create product
|       └ ... (other request objects)
|
|   └ response/         # Response objects
|       ├── ApiResponse.java      # Generic API response
|       └ ... (other response objects)
|
|   └ service/          # Business logic
|       └ impl/                  # Service
|
implementations
|   |   └ ... (other implementations)
|   |
|   └ CartService.java          # Cart service interface
|   └ OrderService.java          # Order service
|
interface
|   |   └ ProductService.java      # Product service
|
interface
|   |   └ ... (other service interfaces)
|
|   └ utils/          # Utility classes
|       └ JwtUtils.java          # JWT utilities
```

```

└── resources/          # Application resources
    ├── application.properties      # Main config
    ├── application-dev.properties # Dev environment config
    ├── application-prod.properties # Production config
    └── ... (other resource files)

```

Frontend Structure (Vite + React + TypeScript)

```

frontend-vite/
├── .dockerignore           # Docker ignore file
├── .gitignore               # Git ignore configuration
├── Dockerfile                # Frontend Docker configuration
├── README.md                 # Project documentation
├── index.html                # Main HTML entry point
├── nginx.conf                # Nginx configuration
├── package.json              # NPM dependencies and scripts
├── public/                   # Static files
|   └── ... (static assets)
├── src/
|   ├── App.css                # Global styles
|   ├── App.tsx                # Root React component
|   ├── Config/
|   |   └── config.ts          # Application configuration
|   |       # API endpoints and settings
|
|   ├── Redux Toolkit/          # State management
|   |   ├── features/            # Feature slices
|   |   |   ├── auth/             # Authentication state
|   |   |   ├── cart/             # Shopping cart state
|   |   |   ├── order/            # Order state
|   |   |   ├── product/           # Product state
|   |   |   └── user/             # User state
|   |   ├── hooks.ts            # Redux hooks
|   |   └── store.ts            # Redux store
|
|   ├── Theme/
|   |   └── index.ts            # Styling themes
|   |       # Theme configuration
|
|   ├── admin/
|   |   ├── components/          # Admin panel components
|   |   ├── pages/                # Admin page components
|   |   └── ... (other admin files)
|
|   └── admin seller/          # Admin seller interface

```

```
    |   |   └ ... (admin seller files)

    |   └ assets/           # Static assets
    |       └ ... (images, icons, etc.)

    └ customer/          # Customer-facing components
        └ components/      # Reusable customer components
            └ Cart/          # Cart components
            └ Checkout/        # Checkout components
            └ Common/          # Common components
            └ Footer/          # Footer component
            └ Header/          # Header component
            └ Products/         # Product components
            └ ... (other component folders)

        └ pages/             # Customer pages
            └ Account/        # User account pages
            └ Auth/            # Authentication pages
            └ Cart/            # Shopping cart pages
            └ Checkout/         # Checkout process
            └ Home/            # Home page
            └ Orders/           # Order history
            └ Products/          # Product listing and details
            └ ... (other page folders)

        └ util/              # Utility functions

    └ data/                # Mock data and constants
        └ categories.ts      # Product categories
        └ ... (other data files)

    └ routes/              # Application routing
        └ AdminRoutes.tsx    # Admin routes
        └ AppRoutes.tsx       # Main application routes
        └ index.tsx           # Router configuration

    └ seller/              # Seller dashboard
        └ components/        # Seller components
        └ pages/              # Seller pages
            └ Dashboard/        # Seller dashboard
            └ Orders/            # Order management
            └ Products/           # Product management
```

```
    |   |   └ ... (other seller pages)
    |   └ ... (other seller files)

    └ types/                      # TypeScript type definitions
        ├── cart.ts                # Cart types
        ├── order.ts                # Order types
        ├── product.ts              # Product types
        └ ... (other type files)

    └ util/                       # Utility functions
        ├── api.ts                 # API client
        ├── formatters.ts          # Data formatting
        └ ... (other utilities)

    └ index.css                   # Global styles
    └ main.tsx                    # Application entry point
    └ vite-env.d.ts               # Vite type definitions

    └ tailwind.config.js          # Tailwind CSS configuration
    └ tsconfig.json               # TypeScript configuration
    └ tsconfig.node.json          # TypeScript Node.js configuration
    └ vite.config.ts              # Vite configuration
```

Key Configuration Files

Backend Configuration

- pom.xml - Maven project configuration with all dependencies
- application.properties - Main Spring Boot configuration
- Dockerfile - Container configuration for backend

Frontend Configuration

- package.json - Lists all dependencies and scripts
- vite.config.ts - Vite build configuration
- tailwind.config.js - Tailwind CSS configuration
- tsconfig.json - TypeScript configuration

Key Components

Backend

- Controllers: Handle HTTP requests and responses
- Services: Implement business logic
- Repositories: Data access layer
- Models/DTOs: Data structures and transfer objects

- Security: Authentication and authorization

Frontend

- Components: Reusable UI elements
- Pages: Main views/screens
- Redux: State management
- Routes: Navigation and routing
- API: Service layer for backend communication

This structure follows modern web application architecture with clear separation of concerns, making it maintainable and scalable for a multi-vendor e-commerce platform. The backend is organized following Spring Boot best practices, while the frontend uses a feature-based folder structure with Redux for state management.

Feedback submitted

Code files starts from here

```
for backend (spring boot)

=> open in IntelliJ idea

change db credential in application.properties file

=> username

=> password

=> and create database

=> change razorpay api & secret

=> give your gmail and app password

=> give your gemini api key

=> now run app

for frontend (React)

=> open in vs code
=> do npm i
```

```
=> npm run dev
```

```
then signup
```

```
https://watery-lunaria-74f.notion.site/Ecommerce-Multi-Vendor-10ae63b763e080269  
26bc280116a3738?pvs=74
```

```
# Build stage
FROM maven:3.8.4-openjdk-17 AS build
WORKDIR /workspace/app

# Copy the Maven project files
COPY pom.xml .
COPY src src/

# Build the application
RUN mvn clean package -DskipTests

# Production stage
FROM openjdk:17-jdk-slim
WORKDIR /app

# Copy the JAR file from the build stage
COPY --from=build /workspace/app/target/*.jar app.jar

# Add a volume to store logs
VOLUME /tmp

# Expose the application port
EXPOSE 5454

# Set environment variables with defaults
ENV SPRING_PROFILES_ACTIVE=prod \
    DB_HOST=db \
    DB_PORT=3306 \
    DB_NAME=ecommerce_multi_vendor \
    DB_USERNAME=user \
    DB_PASSWORD=password \
    SPRING_DATASOURCE_URL=jdbc:mysql://db:3306/ecommerce_multi_vendor?useSSL=false&
serverTimezone=UTC&allowPublicKeyRetrieval=true \
    SPRING_DATASOURCE_USERNAME=user \
    SPRING_DATASOURCE_PASSWORD=password

# Run the Spring Boot application
ENTRYPOINT ["java", "-jar", "app.jar"]

# Health check
```

```
HEALTHCHECK --interval=30s --timeout=3s \
  CMD curl -f http://localhost:5454/actuator/health || exit 1
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/4.0.0
  https://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <parent>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-parent</artifactId>
    <version>3.3.2</version>
    <relativePath/> <!-- lookup parent from repository -->
  </parent>
  <groupId>com.zosh</groupId>
  <artifactId>Ecommerce-multi-vendor</artifactId>
  <version>0.0.1-SNAPSHOT</version>
  <name>Ecommerce-multi-vendor</name>
  <description>Ecommerce multi vendor</description>
  <properties>
    <java.version>17</java.version>
    <spring-ai.version>1.0.0-M1</spring-ai.version>
  </properties>
  <dependencyManagement>
    <dependencies>
      <dependency>
        <groupId>org.springframework.ai</groupId>
        <artifactId>spring-ai-bom</artifactId>
        <version>${spring-ai.version}</version>
        <type>pom</type>
        <scope>import</scope>
      </dependency>
    </dependencies>
  </dependencyManagement>

  <dependencies>
    <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-data-jpa</artifactId>
    </dependency>
    <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-security</artifactId>
    </dependency>
    <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-web</artifactId>
    </dependency>
  </dependencies>

```

```
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-devtools</artifactId>
    <scope>runtime</scope>
    <optional>true</optional>
</dependency>
<dependency>
    <groupId>com.mysql</groupId>
    <artifactId>mysql-connector-j</artifactId>
    <scope>runtime</scope>
</dependency>
<dependency>
    <groupId>org.projectlombok</groupId>
    <artifactId>lombok</artifactId>
    <optional>true</optional>
</dependency>
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-test</artifactId>
    <scope>test</scope>
</dependency>
<dependency>
    <groupId>org.springframework.security</groupId>
    <artifactId>spring-security-test</artifactId>
    <scope>test</scope>
</dependency>

<dependency>
    <groupId>io.jsonwebtoken</groupId>
    <artifactId>jjwt-api</artifactId>
    <version>0.11.1</version>
</dependency>
<dependency>
    <groupId>io.jsonwebtoken</groupId>
    <artifactId>jjwt-impl</artifactId>
    <version>0.11.1</version>
    <scope>runtime</scope>
</dependency>
<dependency>
    <groupId>io.jsonwebtoken</groupId>
    <artifactId>jjwt-jackson</artifactId>
    <version>0.11.1</version>
    <scope>runtime</scope>
</dependency>

<dependency>
    <groupId>com.razorpay</groupId>
    <artifactId>razorpay-java</artifactId>
```

```
        <version>1.4.3</version>
    </dependency>

    <dependency>
        <groupId>com.stripe</groupId>
        <artifactId>stripe-java</artifactId>
        <version>20.62.0</version>
    </dependency>

    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-mail</artifactId>
    </dependency>

    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-validation</artifactId>
    </dependency>

    <dependency>
        <groupId>com.jayway.jsonpath</groupId>
        <artifactId>json-path</artifactId>
    </dependency>

</dependencies>

<build>
    <plugins>
        <plugin>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-maven-plugin</artifactId>
            <configuration>
                <excludes>
                    <exclude>
                        <groupId>org.projectlombok</groupId>
                        <artifactId>lombok</artifactId>
                    </exclude>
                </excludes>
            </configuration>
        </plugin>
    </plugins>
</build>
<repositories>
    <repository>
        <id>spring-milestones</id>
        <name>Spring Milestones</name>
        <url>https://repo.spring.io/milestone</url>
        <snapshots>
```

```

        <enabled>false</enabled>
    </snapshots>
</repository>
</repositories>
</project>
server.port=5454

#db specific properties
#spring.datasource.url=jdbc:mysql://sql12.freemysqlhosting.net:3306/sql12732253
#spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
#spring.datasource.username=sql12732253
#spring.datasource.password=qVJTshEk4b

spring.datasource.url=jdbc:mysql://${DB_HOST:localhost}:${DB_PORT:3306}/${DB_NAME:ecommerce_multi_vendor}
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
spring.datasource.username=${DB_USERNAME:root}
spring.datasource.password=${DB_PASSWORD:sajidsai}

#ORM s/w specific properties
#spring.jpa.properties.hibernate.hbm2ddl.auto=update
spring.jpa.hibernate.ddl-auto=update
spring.jpa.show-sql=true

razorpay.api.key=provide your razorpay api key
razorpay.api.secret=provide your razorpay api secret

stripe.api.key=stripe api key

spring.mail.host=smtp.gmail.com
spring.mail.port=587
spring.mail.username=ss2727303@gmail.com
spring.mail.password=vzjr xyav iloc tacj
#if needed search on youtube how to generate app password in gmail
spring.mail.properties.mail.smtp.auth=true
spring.mail.properties.mail.smtp.starttls.enable=true

spring.devtools.livereload.enabled=true

#ai
gemini.api.key=A1zaSyA6wy9waR2CM-izC6z5myWtrMn0RyGnyE8

package com.zosh;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

```

```
public class EcommerceMultiVendorApplication {

    public static void main(String[] args) {
        SpringApplication.run(EcommerceMultiVendorApplication.class, args);
    }

}

package com.zosh.ai.controllers;

import com.zosh.ai.services.AiChatBotService;
import com.zosh.model.User;
import com.zosh.request.Prompt;
import com.zosh.response.ApiResponse;
import com.zosh.service.UserService;
import lombok.RequiredArgsConstructor;
import org.json.JSONArray;
import org.json.JSONObject;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.*;

@RequiredArgsConstructor
@RestController
@RequestMapping("/ai/chat")
public class AiChatBotController {

    private final AiChatBotService aiChatBotService;
    private final UserService userService;

    @PostMapping()
    public ResponseEntity<ApiResponse> generate(
        @RequestBody Prompt prompt,
        @RequestParam(required = false) Long userId,
        @RequestParam(required = false) Long productId,
        @RequestHeader(required = false, name = "Authorization") String jwt)
throws Exception {

        String message = prompt.getPrompt();
        if (productId != null) {
            message = "the product id is " + productId +", " + message ;
        }

        User user=new User();
        if(jwt!=null)
            user=userService.findUserProfileByJwt(jwt);

        //        Long userId;
        //        if(user!=null){
        //            userId=user.getId();
        //        }
    }
}
```

```
        ApiResponse apiResponse =
aiChatBotService.aiChatBot(message,productId,user.getId());

        return ResponseEntity.ok(apiResponse);

    }

}

package com.zosh.ai.controllers;

import com.zosh.response.ApiResponse;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.bind.annotation.RestController;

import java.util.Map;

@RestController
@RequestMapping("/ai")
public class AiHomeController {

    @GetMapping()
    public ResponseEntity<ApiResponse> AiHome(){
        ApiResponse response = new ApiResponse();
        response.setMessage("welcome to ai world");
        return new ResponseEntity<>(response, HttpStatus.OK);
    }

}
package com.zosh.ai.controllers;

import com.zosh.ai.services.AiProductService;
import com.zosh.response.ApiResponse;
import org.json.JSONArray;
import org.json.JSONObject;
import org.springframework.web.bind.annotation.*;

@RestController
@RequestMapping("/ai")
public class AiProductController {

    private final AiProductService productService;

    public AiProductController(AiProductService productService) {
```

```
        this.productService = productService;
    }

    @PostMapping("/chat/demo")
    public ApiResponse generate(@RequestParam(
        value = "message",
        defaultValue = "Tell me a joke") String message) throws Exception {

        String ans = productService.simpleChat(message);

        JSONObject messageJson = new JSONObject(ans);

        // Extract the text from the JSON
        JSONArray candidates = messageJson.getJSONArray("candidates");
        JSONObject content =
        candidates.getJSONObject(0).getJSONObject("content");
        JSONArray parts = content.getJSONArray("parts");
        String text = parts.getJSONObject(0).getString("text");

        ApiResponse apiResponse = new ApiResponse();
        apiResponse.setMessage(text);

        return apiResponse;
    }
}

package com.zosh.ai.services;

import com.zosh.exception.ProductException;
import com.zosh.response.ApiResponse;

public interface AiChatBotService {

    ApiResponse aiChatBot(String prompt, Long productId, Long userId) throws
ProductException;
}

```

```
package com.zosh.ai.services;

import com.zosh.exception.ProductException;
import com.zosh.mapper.OrderMapper;
import com.zosh.mapper.ProductMapper;
import com.zosh.model.Cart;
import com.zosh.model.Order;
import com.zosh.model.Product;
import com.zosh.model.User;
import com.zosh.repository.CartRepository;
import com.zosh.repository.OrderRepository;
import com.zosh.repository.ProductRepository;
import com.zosh.repository.UserRepository;
```

```
import com.zosh.response.ApiResponse;
import com.zosh.response.FunctionResponse;
import lombok.RequiredArgsConstructor;
import org.json.JSONArray;
import org.json.JSONObject;
import org.springframework.http.HttpEntity;
import org.springframework.http.HttpHeaders;
import org.springframework.http.MediaType;
import org.springframework.http.ResponseEntity;
import org.springframework.stereotype.Service;
import org.springframework.web.client.RestTemplate;

import java.util.List;
import java.util.Map;

@Service
@RequiredArgsConstructor
public class AiChatBotServiceImpl implements AiChatBotService {

    String GEMINI_API_KEY = "AIzaSyDp-jeRRqqbr08scpInlp9rLEL_Nqv5Zuo";

    private final CartRepository cartRepository;

    private final OrderRepository orderRepository;

    private final ProductRepository productRepository;
    private final UserRepository userRepository;

    private JSONArray createFunctionDeclarations() {
        return new JSONArray()
            .put(new JSONObject()
                .put("name", "getUserCart")
                .put("description", "Retrieve the user's cart details")
                .put("parameters", new JSONObject()
                    .put("type", "OBJECT")
                    .put("properties", new JSONObject()
                        .put("cart", new JSONObject()
                            .put("type", "STRING")
                            .put("description", "Cart
Details, like total item in cart, cart item, remove item from cart, cart Id"))
                    )
                )
                .put("required", new JSONArray()
                    .put("cart")
                )
            )
    }
}
```

```

        .put(new JSONObject()
            .put("name", "getUsersOrder")
            .put("description", "Retrieve the user's order details")
            .put("parameters", new JSONObject()
                .put("type", "OBJECT")
                .put("properties", new JSONObject()
                    .put("order", new JSONObject()
                        .put("type", "STRING")
                        .put("description", "Order
Details, order, total order, current order, delivered order, pending order,
current order, canceled order"))
                    )
                )
            .put("required", new JSONArray()
                .put("order")
            )
        )
    )
    .put(new JSONObject()
        .put("name", "getProductDetails")
        .put("description", "Retrieve product details")
        .put("parameters", new JSONObject()
            .put("type", "OBJECT")
            .put("properties", new JSONObject()
                .put("product", new JSONObject()
                    .put("type", "STRING")
                    .put("description", "The Product
Details like, Product title, product id, product color, product size, selling
price, mrp price, rating extra..."))
                )
            )
        .put("required", new JSONArray()
            .put("product")
        )
    )
}
);

private FunctionResponse processFunctionCall(JSONObject functionCall,
    Long productId,
    Long userId
) throws ProductException {
    String functionName = functionCall.getString("name");
    JSONObject args = functionCall.getJSONObject("args");

    FunctionResponse res = new FunctionResponse();
    res.setFunctionName(functionName);
    User user=userRepository.findById(userId).orElse(null);

```

```

        switch (functionName) {
            case "getUserCart":
//                Long userId = Long.parseLong(args.getString("userId"));
                Cart cart = cartRepository.findById(userId);
                System.out.println("cart: " + cart.getId());
                res.setUserCart(cart);
                break;
            case "getUsersOrder":
//                Long orderId = Long.parseLong(args.getString("orderId"));
                List<Order> orders = orderRepository.findByUserId(userId);
                res.setOrderHistory(OrderMapper.toOrderHistory(orders, user));
                System.out.println("order history: " +
OrderMapper.toOrderHistory(orders, user));
                break;
            case "getProductDetails":
//                Long productId = Long.parseLong(args.getString("productId"));
                Product product =
productRepository.findById(productId).orElseThrow(
                    () -> new ProductException("product not found")
                );
                res.setProduct(product);
                break;
            default:
                throw new IllegalArgumentException("Unsupported function: " +
functionName);
        }
        return res;
    }

    public FunctionResponse getFunctionResponse(String prompt, Long productId,
Long userId) throws ProductException {
    String GEMINI_API_URL =
"https://generativelanguage.googleapis.com/v1beta/models/gemini-pro:generateContent?key=" + GEMINI_API_KEY;

    JSONObject requestBodyJson = new JSONObject()
        .put("contents", new JSONArray()
            .put(new JSONObject()
                .put("parts", new JSONArray()
                    .put(new JSONObject()
                        .put("text", prompt)
                    )
                )
            )
        )
        .put("tools", new JSONArray()
            .put(new JSONObject()

```

```

        .put("functionDeclarations",
createFunctionDeclarations() )
    )
);

HttpHeaders headers = new HttpHeaders();

headers.setContentType(org.springframework.http.MediaType.APPLICATION_JSON);

HttpEntity<String> requestEntity = new
HttpEntity<>(requestBodyJson.toString(), headers);

RestTemplate restTemplate = new RestTemplate();
 ResponseEntity<String> response =
restTemplate.postForEntity(GEMINI_API_URL, requestEntity, String.class);

String responseBody = response.getBody();
JSONObject jsonObject = new JSONObject(responseBody);

System.out.println("functionResponse: " + responseBody);
JSONArray candidates = jsonObject.getJSONArray("candidates");
JSONObject firstCandidate = candidates.getJSONObject(0);
JSONObject content = firstCandidate.getJSONObject("content");
JSONArray parts = content.getJSONArray("parts");
JSONObject firstPart = parts.getJSONObject(0);
JSONObject functionCall = firstPart.getJSONObject("functionCall");

return processFunctionCall(functionCall, productId, userId);
}

@Override
public ApiResponse aiChatBot(String prompt, Long productId, Long userId)
throws ProductException {
    String GEMINI_API_URL =
"https://generativelanguage.googleapis.com/v1beta/models/gemini-pro:generateContent?key=" + GEMINI_API_KEY;

    System.out.println("----- " + prompt);

    FunctionResponse functionResponse = getFunctionResponse(prompt,
productId, userId);
    System.out.println("----- " + functionResponse);

    HttpHeaders headers = new HttpHeaders();
    headers.setContentType(MediaType.APPLICATION_JSON);

    // Construct the request body

```

```

String body = new JSONObject()
    .put("contents", new JSONArray()
        .put(new JSONObject()
            .put("role", "user")
            .put("parts", new JSONArray()
                .put(new JSONObject()
                    .put("text", prompt)
                )
            )
        )
    )
    .put(new JSONObject()
        .put("role", "model")
        .put("parts", new JSONArray()
            .put(new JSONObject()
                .put("functionCall", new
JSONObject()
                    .put("name",
functionResponse.getFunctionName())
                    .put("args", new
JSONObject()
                        .put("cart",
functionResponse.getUserCart() !=null?functionResponse.getUserCart().getUser():n
ull)
                        .put("order",
functionResponse.getOrderHistory() !=null? functionResponse.getOrderHistory()
:null )
                        .put("product",
functionResponse.getProduct() !=null?ProductMapper.toProductDto(functionResponse
.getProduct()):null)
                    )
                )
            )
        )
    )
    .put(new JSONObject()
        .put("role", "function")
        .put("parts", new JSONArray()
            .put(new JSONObject()
                .put("functionResponse", new
JSONObject()
                    .put("name",
functionResponse.getFunctionName())
                    .put("response", new
JSONObject()
                        .put("name",
functionResponse.getFunctionName())
                        .put("content",
functionResponse)
                )
            )
        )
    )
}

```

```

        )
    )
)
)
.put("tools", new JSONArray()
    .put(new JSONObject()
        .put("functionDeclarations",
createFunctionDeclarations())
    )
)
.toString();

// Make the API request
HttpEntity<String> request = new HttpEntity<>(body, headers);
RestTemplate restTemplate = new RestTemplate();
ResponseEntity<String> response =
restTemplate.postForEntity(GEMINI_API_URL, request, String.class);

// Process the response
String responseBody = response.getBody();
JSONObject jsonObject = new JSONObject(responseBody);

// Extract the first candidate
JSONArray candidates = jsonObject.getJSONArray("candidates");
JSONObject firstCandidate = candidates.getJSONObject(0);

// Extract the text
JSONObject content = firstCandidate.getJSONObject("content");
JSONArray parts = content.getJSONArray("parts");
JSONObject firstPart = parts.getJSONObject(0);
String text = firstPart.getString("text");

// Prepare and return the API response
ApiResponse res = new ApiResponse();
res.setMessage(text);
return res;

}
}

package com.zosh.ai.services;

public interface AiProductService {

    String simpleChat(String prompt);

}

}

```

```
import com.zosh.response.ApiResponse;
import com.zosh.response.FunctionResponse;
import org.json.JSONArray;
import org.json.JSONObject;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.beans.factory.annotation.Value;

import com.jayway.jsonpath.JsonPath;
import com.jayway.jsonpath.ReadContext;

import org.springframework.http.*;
import org.springframework.stereotype.Service;
import org.springframework.web.client.RestTemplate;

import java.util.Map;

@Service
public class AiProductServiceImpl implements AiProductService {

    @Value("${gemini.api.key}")
    private static String API_KEY;

    @Override
    public String simpleChat(String prompt) {
        return "";
    }
}

package com.zosh.ai.services;

public interface ProductDetailsBotService {

    String productDetailsChatBot(String prompt);
}

package com.zosh.ai.services;

import org.springframework.stereotype.Service;

@Service
public class ProductDetailsBotServiceImpl implements ProductDetailsBotService{



    @Override
    public String productDetailsChatBot(String prompt) {
        return "";
    }
}

package com.zosh.config;
```

```
import java.lang.reflect.Array;
import java.util.Arrays;
import java.util.Collections;

import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import
org.springframework.security.config.annotation.web.builders.HttpSecurity;
import
org.springframework.security.config.annotation.web.configuration.EnableWebSecurity;
import org.springframework.security.config.http.SessionCreationPolicy;
import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;
import org.springframework.security.crypto.password.PasswordEncoder;
import org.springframework.security.web.SecurityFilterChain;
import
org.springframework.security.web.authentication.www.BasicAuthenticationFilter;
import org.springframework.web.client.RestTemplate;
import org.springframework.web.cors.CorsConfiguration;
import org.springframework.web.cors.CorsConfigurationSource;

import jakarta.servlet.http.HttpServletRequest;

@Configuration
@EnableWebSecurity
public class AppConfig {

    @Bean
    SecurityFilterChain securityFilterChain(HttpSecurity http) throws Exception
    {

        http.sessionManagement(management ->
management.sessionCreationPolicy(SessionCreationPolicy.STATELESS)
            .authorizeHttpRequests(Authorize -> Authorize
        //
        .requestMatchers("/api/admin/**").hasAnyRole("SHOP_OWNER", "ADMIN")
            .requestMatchers("/api/**").authenticated()

        .requestMatchers("/api/products/*/reviews").permitAll()
            .anyRequest().permitAll()
        )
            .addFilterBefore(new JwtTokenValidator(),
BasicAuthenticationFilter.class)
            .csrf(csrf -> csrf.disable())
            .cors(cors ->
cors.configurationSource(corsConfigurationSource())));
    }
}
```

```

        return http.build();

    }

    // CORS Configuration
    private CorsConfigurationSource corsConfigurationSource() {
        return new CorsConfigurationSource() {
            @Override
            public CorsConfiguration getCorsConfiguration(HttpServletRequest request) {
                CorsConfiguration cfg = new CorsConfiguration();
                cfg.setAllowedOrigins(Arrays.asList(
                    "https://zosh-bazaar-zosh.vercel.app",
                    "http://localhost:3000",
                    "http://localhost:5174"));
                cfg.setAllowedMethods(Collections.singletonList("*"));
                cfg.setAllowCredentials(true);
                cfg.setAllowedHeaders(Collections.singletonList("*"));
                cfg.setExposedHeaders(Arrays.asList("Authorization"));
                cfg.setMaxAge(3600L);
                return cfg;
            }
        };
    }

    @Bean
    PasswordEncoder passwordEncoder() {
        return new BCryptPasswordEncoder();
    }

    @Bean
    public RestTemplate restTemplate() {
        return new RestTemplate();
    }
}

package com.zosh.config;

public class JwtConstant {

    public static final String
SECRET_KEY="wpembytrwcvnryxksdbqwjebruyGHyudqgwveytrtrCSnwfoesarjbwe";
    public static final String JWT_HEADER="Authorization";

}

package com.zosh.config;

import java.util.Collection;
import java.util.Date;
import java.util.HashSet;

```

```
import java.util.Set;

import javax.crypto.SecretKey;

import org.springframework.security.core.Authentication;
import org.springframework.security.core.GrantedAuthority;
import org.springframework.stereotype.Service;

import io.jsonwebtoken.Claims;
import io.jsonwebtoken.Jwts;
import io.jsonwebtoken.security.Keys;

@Service
public class JwtProvider {

    private SecretKey key=Keys.hmacShaKeyFor(JwtConstant.SECRET_KEY.getBytes());

    public String generateToken(Authentication auth) {
        Collection<? extends GrantedAuthority> authorities =
auth.getAuthorities();
        String roles = populateAuthorities(authorities);

        String jwt=Jwts.builder()
            .setIssuedAt(new Date())
            .setExpiration(new Date(new Date().getTime() +86400000))
            .claim("email",auth.getName())
            .claim("authorities", roles)
            .signWith(key)
            .compact();
        return jwt;
    }

    public String getEmailFromJwtToken(String jwt) {
        jwt=jwt.substring(7);

        Claims
claims=Jwts.parserBuilder().setSigningKey(key).build().parseClaimsJws(jwt).getBody();
        String email=String.valueOf(claims.get("email"));

        return email;
    }

    public String populateAuthorities(Collection<? extends GrantedAuthority>
collection) {
        Set<String> auths=new HashSet<>();

        for(GrantedAuthority authority:collection) {
```

```
        auths.add(authority.getAuthority());
    }
    return String.join(", ", auths);
}

}

package com.zosh.config;

import java.io.IOException;
import java.util.List;

import javax.crypto.SecretKey;

import org.springframework.security.authentication.BadCredentialsException;
import org.springframework.security.authentication.UsernamePasswordAuthenticationToken;
;
import org.springframework.security.core.Authentication;
import org.springframework.security.core.GrantedAuthority;
import org.springframework.security.core.authority.AuthorityUtils;
import org.springframework.security.core.context.SecurityContextHolder;
import org.springframework.web.filter.OncePerRequestFilter;

import io.jsonwebtoken.Claims;
import io.jsonwebtoken.Jwts;
import io.jsonwebtoken.security.Keys;

import jakarta.servlet.FilterChain;
import jakarta.servlet.ServletException;

import jakarta.servlet.http.HttpServletRequest;
import jakarta.servlet.http.HttpServletResponse;

public class JwtTokenValidator extends OncePerRequestFilter {

    @Override
    protected void doFilterInternal(HttpServletRequest request,
HttpServletResponse response, FilterChain filterChain)
        throws ServletException, IOException {
    String jwt = request.getHeader(JwtConstant.JWT_HEADER);

    if(jwt!=null) {
        jwt=jwt.substring(7);

        try {

```

```

        SecretKey key=
Keys.hmacShaKeyFor(JwtConstant.SECRET_KEY.getBytes());

        Claims
claims=JwtTokenBuilder().setSigningKey(key).build().parseClaimsJws(jwt).getBody();

        String email=String.valueOf(claims.get("email"));

        String authorities=String.valueOf(claims.get("authorities"));

        System.out.println("authorities ----- "+authorities);

        List<GrantedAuthority>
auths=AuthorityUtils.commaSeparatedStringToAuthorityList(authorities);
        Authentication authentication=new
UsernamePasswordAuthenticationToken(email,null, auths);

SecurityContextHolder.getContext().setAuthentication(authentication);

    } catch (Exception e) {
        throw new BadCredentialsException("invalid token...");
    }
}

filterChain.doFilter(request, response);

}

}

package com.zosh.controller;

import com.zosh.domain.AccountStatus;
import com.zosh.exception.SellerException;
import com.zosh.model.HomeCategory;
import com.zosh.model.Seller;
import com.zosh.service.HomeCategoryService;
import com.zosh.service.SellerService;
import lombok.RequiredArgsConstructor;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.*;

import java.util.List;

@RestController
@RequestMapping("/admin")
@RequiredArgsConstructor
public class AdminController {

```

```

private final SellerService sellerService;
private final HomeCategoryService homeCategoryService;

@PatchMapping("/seller/{id}/status/{status}")
public ResponseEntity<Seller> updateSellerStatus(
    @PathVariable Long id,
    @PathVariable AccountStatus status) throws SellerException {

    Seller updatedSeller =
sellerService.updateSellerAccountStatus(id,status);
    return ResponseEntity.ok(updatedSeller);

}

@GetMapping("/home-category")
public ResponseEntity<List<HomeCategory>> getHomeCategory(
) throws Exception {

    List<HomeCategory> categories=homeCategoryService.getAllCategories();
    return ResponseEntity.ok(categories);

}

@PatchMapping("/home-category/{id}")
public ResponseEntity<HomeCategory> updateHomeCategory(
    @PathVariable Long id,
    @RequestBody HomeCategory homeCategory) throws Exception {

    HomeCategory
updatedCategory=homeCategoryService.updateCategory(homeCategory,id);
    return ResponseEntity.ok(updatedCategory);

}
}

package com.zosh.controller;

import com.zosh.exception.UserException;
import com.zosh.model.Cart;
import com.zosh.model.Coupon;
import com.zosh.model.User;
import com.zosh.service.CartService;
import com.zosh.service.CouponService;
import com.zosh.service.UserService;
import lombok.RequiredArgsConstructor;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.*;

```

```
import java.util.List;

@RestController
@RequestMapping("/api/coupons")
@RequiredArgsConstructor
public class AdminCouponController {

    private final CouponService couponService;
    private final UserService userService;
    private final CartService cartService;

    @PostMapping("/apply")
    public ResponseEntity<Cart> applyCoupon(
        @RequestParam String apply,
        @RequestParam String code,
        @RequestParam double orderValue,
        @RequestHeader("Authorization")
        ) String jwt
    ) throws Exception {
        User user=userService.findUserProfileByJwt(jwt);
        Cart cart;

        if(apply.equals("true")){
            cart = couponService.applyCoupon(code,orderValue,user);
        }
        else {
            cart = couponService.removeCoupon(code,user);
        }

        return ResponseEntity.ok(cart);
    }

    // Admin operations

    @PostMapping("/admin/create")
    public ResponseEntity<Coupon> createCoupon(@RequestBody Coupon coupon) {
        Coupon createdCoupon = couponService.createCoupon(coupon);
        return ResponseEntity.ok(createdCoupon);
    }

    @DeleteMapping("/admin/delete/{id}")
    public ResponseEntity<?> deleteCoupon(@PathVariable Long id) {
        couponService.deleteCoupon(id);
        return ResponseEntity.ok("Coupon deleted successfully");
    }

    @GetMapping("/admin/all")
```

```
public ResponseEntity<List<Coupon>> getAllCoupons() {
    List<Coupon> coupons = couponService.getAllCoupons();
    return ResponseEntity.ok(coupons);
}
}

package com.zosh.controller;

import com.zosh.domain.USER_ROLE;
import com.zosh.exception.SellerException;
import com.zosh.exception.UserException;
import com.zosh.model.*;
import com.zosh.repository.VerificationCodeRepository;
import com.zosh.request.ResetPasswordRequest;
import com.zosh.request.SignupRequest;
import com.zosh.service.AuthService;
import jakarta.mail.MessagingException;
import lombok.RequiredArgsConstructor;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;

import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.bind.annotation.RestController;

import com.zosh.request.LoginRequest;
import com.zosh.response.ApiResponse;
import com.zosh.response.AuthResponse;

import jakarta.validation.Valid;

@RestController
@RequestMapping("/auth")
@RequiredArgsConstructor
public class AuthController {

    private final AuthService authService;

    @PostMapping("/sent/login-signup-otp")
    public ResponseEntity<ApiResponse> sentLoginOtp(
        @RequestBody VerificationCode req) throws MessagingException,
UserException {

        authService.sentLoginOtp(req.getEmail());
    }
}
```

```
        ApiResponse res = new ApiResponse();
        res.setMessage("otp sent");
        return new ResponseEntity<>(res, HttpStatus.CREATED);
    }

    @PostMapping("/signup")
    public ResponseEntity<AuthResponse> createUserHandler(
        @Valid
        @RequestBody SignupRequest req)
        throws SellerException {

        String token = authService.createUser(req);
        AuthResponse authResponse = new AuthResponse();
        authResponse.setJwt(token);
        authResponse.setMessage("Register Success");
        authResponse.setRole(USER_ROLE.ROLE_CUSTOMER);

        return new ResponseEntity<>(authResponse, HttpStatus.OK);
    }

    @PostMapping("/signin")
    public ResponseEntity<AuthResponse> signin(@RequestBody LoginRequest
loginRequest) throws SellerException {

        AuthResponse authResponse = authService.signin(loginRequest);
        return new ResponseEntity<>(authResponse, HttpStatus.OK);
    }

}

package com.zosh.controller;

import com.zosh.exception.CartItemException;
import com.zosh.exception.ProductException;
import com.zosh.exception.UserException;
import com.zosh.model.Cart;
import com.zosh.model.CartItem;
import com.zosh.model.Product;
import com.zosh.model.User;
import com.zosh.request.AddItemRequest;
import com.zosh.response.ApiResponse;
import com.zosh.service.CartItemService;
import com.zosh.service.CartService;
import com.zosh.service.ProductService;
```

```
import com.zosh.service.UserService;
import lombok.RequiredArgsConstructor;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.*;

@RestController
@RequestMapping("/api/cart")
@RequiredArgsConstructor
public class CartController {

    private final CartService cartService;
    private final UserService userService;
    private final ProductService productService;
    private final CartItemService cartItemService;

    @GetMapping
    public ResponseEntity<Cart>
    findUserCartHandler(@RequestHeader("Authorization") String jwt) throws
    UserException{

        User user=userService.findUserProfileByJwt(jwt);

        Cart cart=cartService.findUserCart(user);

        System.out.println("cart - "+cart.getUser().getEmail());

        return new ResponseEntity<Cart>(cart,HttpStatus.OK);
    }

    @PutMapping("/add")
    public ResponseEntity<CartItem> addItemToCart(@RequestBody AddItemRequest
req,
                                                @RequestHeader("Authorization") String
jwt) throws UserException, ProductException{

        User user=userService.findUserProfileByJwt(jwt);
        Product product=productService.findProductById(req.getProductId());

        CartItem item = cartService.addCartItem(user,
            product,
            req.getSize(),
            req.getQuantity());

        return new ResponseEntity<>(item,HttpStatus.ACCEPTED);
    }
}
```

```

    }

    @DeleteMapping("/item/{cartItemId}")
    public ResponseEntity<ApiResponse> deleteCartItemHandler(
        @PathVariable Long cartItemId,
        @RequestHeader("Authorization") String jwt)
        throws CartItemException, UserException{

        User user=userService.findUserProfileByJwt(jwt);
        cartItemService.removeCartItem(user.getId(), cartItemId);

        ApiResponse res=new ApiResponse("Item Remove From Cart",true);

        return new ResponseEntity<ApiResponse>(res,HttpStatus.ACCEPTED);
    }

    @PutMapping("/item/{cartItemId}")
    public ResponseEntity<CartItem> updateCartItemHandler(
        @PathVariable Long cartItemId,
        @RequestBody CartItem cartItem,
        @RequestHeader("Authorization") String jwt)
        throws CartItemException, UserException{

        User user=userService.findUserProfileByJwt(jwt);

        CartItem updatedCartItem = null;
        if(cartItem.getQuantity()>0){
            updatedCartItem=cartItemService.updateCartItem(user.getId(),
                cartItemId, cartItem);
        }

        return new ResponseEntity<>(updatedCartItem,HttpStatus.ACCEPTED);
    }

}
package com.zosh.controller;

import com.zosh.exception.CartItemException;
import com.zosh.exception.UserException;

import com.zosh.model.CartItem;
import com.zosh.model.User;
import com.zosh.response.ApiResponse;
import com.zosh.service.CartItemService;

```

```
import com.zosh.service.UserService;

import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.*;

@RestController
@RequestMapping("/api/cart_items")
public class CartItemController {

    private CartItemService cartItemService;
    private UserService userService;

    public CartItemController(CartItemService cartItemService, UserService
userService) {
        this.cartItemService=cartItemService;
        this.userService=userService;
    }

}

package com.zosh.controller;

import com.zosh.model.Home;
import com.zosh.model.HomeCategory;
import com.zosh.service.HomeCategoryService;
import com.zosh.service.HomeService;
import lombok.RequiredArgsConstructor;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RestController;

import java.util.List;

@RestController
@RequiredArgsConstructor
public class CustomerController {
    private final HomeCategoryService homeCategoryService;
    private final HomeService homeService;

    @GetMapping("/home-page")
    public ResponseEntity<Home> getHomePageData() {
//        Home homepageData = homeService.getHomePageData();
//        return new ResponseEntity<>(homepageData, HttpStatus.ACCEPTED);
        return null;
    }
}
```

```
@PostMapping("/home/categories")
public ResponseEntity<Home> createHomeCategories(
    @RequestBody List<HomeCategory> homeCategories
) {
    List<HomeCategory> categories =
homeCategoryService.createCategories(homeCategories);
    Home home=homeService.createHomePageData(categories);
    return new ResponseEntity<>(home, HttpStatus.ACCEPTED);
}
}
package com.zosh.controller;

import com.zosh.model.Deal;
import com.zosh.model.HomeCategory;
import com.zosh.response.ApiResponse;
import com.zosh.service.DealService;
import lombok.RequiredArgsConstructor;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.*;

import java.util.List;

@RestController
@RequiredArgsConstructor
@RequestMapping("/admin/deals")
public class DealController {
    private final DealService dealService;

    @PostMapping
    public ResponseEntity<Deal> createDeals(
        @RequestBody Deal deals
    ) {
        Deal createdDeals=dealService.createDeal(deals);

        return new ResponseEntity<>(createdDeals, HttpStatus.ACCEPTED);
    }

    @GetMapping
    public ResponseEntity<List<Deal>> getDeals(
    ) {
        List<Deal> deals=dealService.getDeals();

        return new ResponseEntity<>(deals, HttpStatus.ACCEPTED);
    }

    @PatchMapping("/{id}")

```

```

public ResponseEntity<Deal> updateDeal(
    @PathVariable Long id,
    @RequestBody Deal deal) throws Exception {

    Deal updatedDeal=dealService.updateDeal(deal,id);
    return ResponseEntity.ok(updatedDeal);

}

@DeleteMapping("/{id}")
public ResponseEntity<ApiResponse> deleteDeals(
    @PathVariable Long id
) throws Exception {
    dealService.deleteDeal(id);

    ApiResponse apiResponse=new ApiResponse();
    apiResponse.setMessage("Deal deleted");

    return new ResponseEntity<>(apiResponse, HttpStatus.ACCEPTED);
}

}

package com.zosh.controller;

import com.zosh.model.Home;
import com.zosh.response.ApiResponse;
import com.zosh.service.HomeService;
import lombok.RequiredArgsConstructor;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RestController;

import java.util.List;

@RestController
@RequiredArgsConstructor
public class HomeController {

    private final HomeService homeService;

    @GetMapping
    public ResponseEntity<ApiResponse> home() {
        ApiResponse apiResponse = new ApiResponse();
        apiResponse.setMessage("Ecommerce multi vendor system");
        return new ResponseEntity<>(apiResponse, HttpStatus.ACCEPTED);
    }
}

```

```
}
```

```
package com.zosh.controller;
```

```
import com.razorpay.PaymentLink;
```

```
import com.razorpay.RazorpayException;
```

```
import com.stripe.exception.StripeException;
```

```
import com.zosh.domain.PaymentMethod;
```

```
import com.zosh.exception.OrderException;
```

```
import com.zosh.exception.SellerException;
```

```
import com.zosh.exception.UserException;
```

```
import com.zosh.model.*;
```

```
import com.zosh.repository.PaymentOrderRepository;
```

```
import com.zosh.response.PaymentLinkResponse;
```

```
import com.zosh.service.*;
```

```
import lombok.RequiredArgsConstructor;
```

```
import org.springframework.beans.factory.annotation.Autowired;
```

```
import org.springframework.http.HttpStatus;
```

```
import org.springframework.http.ResponseEntity;
```

```
import org.springframework.web.bind.annotation.*;
```

```
import java.util.List;
```

```
import java.util.Set;
```

```
@RestController
```

```
@RequestMapping("/api/orders")
```

```
@RequiredArgsConstructor
```

```
public class OrderController {
```

```
    private final OrderService orderService;
```

```
    private final UserService userService;
```

```
    private final OrderItemService orderItemService;
```

```
    private final CartService cartService;
```

```
    private final PaymentService paymentService;
```

```
    private final PaymentOrderRepository paymentOrderRepository;
```

```
    private final SellerReportService sellerReportService;
```

```
    private final SellerService sellerService;
```

```
    @PostMapping()
```

```
    public ResponseEntity<PaymentLinkResponse> createOrderHandler(
```

```
        @RequestBody Address spippingAddress,
```

```
        @RequestParam PaymentMethod paymentMethod,
```

```
        @RequestHeader("Authorization")String jwt)
```

```
        throws UserException, RazorpayException, StripeException {
```

```

        User user=userService.findUserProfileByJwt(jwt);
        Cart cart=cartService.findUserCart(user);
        Set<Order> orders =orderService.createOrder(user, spippingAddress,cart);

        PaymentOrder paymentOrder=paymentService.createOrder(user,orders);

        PaymentLinkResponse res = new PaymentLinkResponse();

        if(paymentMethod.equals(PaymentMethod.RAZORPAY)) {
            PaymentLink payment=paymentService.createRazorpayPaymentLink(user,
                paymentOrder.getAmount(),
                paymentOrder.getId());
            String paymentUrl=payment.get("short_url");
            String paymentUrlId=payment.get("id");

            res.setPayment_link_url(paymentUrl);
//            res.setPayment_link_id(paymentUrlId);
            paymentOrder.setPaymentLinkId(paymentUrlId);
            paymentOrderRepository.save(paymentOrder);
        }
        else{
            String paymentUrl=paymentService.createStripePaymentLink(user,
                paymentOrder.getAmount(),
                paymentOrder.getId());
            res.setPayment_link_url(paymentUrl);
        }
        return new ResponseEntity<>(res,HttpStatus.OK);
    }

    @GetMapping("/user")
    public ResponseEntity< List<Order>> usersOrderHistoryHandler(
        @RequestHeader("Authorization")
        String jwt) throws UserException{

        User user=userService.findUserProfileByJwt(jwt);
        List<Order> orders=orderService.usersOrderHistory(user.getId());
        return new ResponseEntity<>(orders,HttpStatus.ACCEPTED);
    }

    @GetMapping("/{orderId}")
    public ResponseEntity< Order> getOrderById(@PathVariable Long orderId,
@RequestHeader("Authorization")
    String jwt) throws OrderException, UserException{

        User user = userService.findUserProfileByJwt(jwt);
        Order orders=orderService.findOrderById(orderId);
        return new ResponseEntity<>(orders,HttpStatus.ACCEPTED);
    }

```

```

    }

    @GetMapping("/item/{orderItemId}")
    public ResponseEntity<OrderItem> getOrderItemById(
        @PathVariable Long orderItemId, @RequestHeader("Authorization")
    String jwt) throws Exception {
        System.out.println("----- controller -----");
        User user = userService.findUserProfileByJwt(jwt);
        OrderItem orderItem=orderItemService.getOrderItemById(orderItemId);
        return new ResponseEntity<>(orderItem,HttpStatus.ACCEPTED);
    }

    @PutMapping("/{orderId}/cancel")
    public ResponseEntity<Order> cancelOrder(
        @PathVariable Long orderId,
        @RequestHeader("Authorization") String jwt
    ) throws UserException, OrderException, SellerException {
        User user=userService.findUserProfileByJwt(jwt);
        Order order=orderService.cancelOrder(orderId,user);

        Seller seller= sellerService.getSellerById(order.getSellerId());
        SellerReport report=sellerReportService.getSellerReport(seller);

        report.setCanceledOrders(report.getCanceledOrders() +1);

        report.setTotalRefunds(report.getTotalRefunds() +order.getTotalSellingPrice());
        sellerReportService.updateSellerReport(report);

        return ResponseEntity.ok(order);
    }

}

package com.zosh.controller;

import com.razorpay.RazorpayException;
import com.stripe.exception.StripeException;
import com.zosh.domain.PaymentMethod;
import com.zosh.exception.UserException;
import com.zosh.model.*;
import com.zosh.repository.CartItemRepository;
import com.zosh.repository.CartRepository;
import com.zosh.response.ApiResponse;
import com.zosh.response.PaymentLinkResponse;
import com.zosh.service.*;
import lombok.RequiredArgsConstructor;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.*;

```

```
import java.util.ArrayList;
import java.util.HashSet;
import java.util.List;
import java.util.Set;

@RestController
@RequiredArgsConstructor
public class PaymentController {

    private final UserService userService;
    private final PaymentService paymentService;
    private final TransactionService transactionService;
    private final SellerReportService sellerReportService;
    private final SellerService sellerService;
    private final CartRepository cartRepository;
    private final CartItemRepository cartItemRepository;

    @PostMapping("/api/payment/{paymentMethod}/order/{orderId}")
    public ResponseEntity<PaymentLinkResponse> paymentHandler(
        @PathVariable PaymentMethod paymentMethod,
        @PathVariable Long orderId,
        @RequestHeader("Authorization") String jwt) throws Exception {

        User user = userService.findUserProfileByJwt(jwt);

        PaymentLinkResponse paymentResponse;

        PaymentOrder order= paymentService.getPaymentOrderById(orderId);

        // if(paymentMethod.equals(PaymentMethod.RAZORPAY)) {
        //     paymentResponse=paymentService.createRazorpayPaymentLink(user,
        //         order.getAmount(),
        //         order.getId());
        // }
        // else{
        //     paymentResponse=paymentService.createStripePaymentLink(user,
        //         order.getAmount(),
        //         order.getId());
        // }

        return new ResponseEntity<>(null, HttpStatus.CREATED);
    }

    @GetMapping("/api/payment/{paymentId}")
    public ResponseEntity<ApiResponse> paymentSuccessHandler(
        @PathVariable String paymentId,
```

```

    @RequestParam String paymentLinkId,
    @RequestHeader("Authorization") String jwt) throws Exception {

    User user = userService.findUserProfileByJwt(jwt);

    PaymentLinkResponse paymentResponse;

    PaymentOrder paymentOrder= paymentService
        .getPaymentOrderByPaymentId(paymentLinkId);

    boolean paymentSuccess = paymentService.ProceedPaymentOrder(
        paymentOrder,
        paymentId,
        paymentLinkId
    );
    if(paymentSuccess){
        for(Order order:paymentOrder.getOrders()){
            transactionService.createTransaction(order);
            Seller seller=sellerService.getSellerById(order.getSellerId());
            SellerReport report=sellerReportService.getSellerReport(seller);
            report.setTotalOrders(report.getTotalOrders()+1);

            report.setTotalEarnings(report.getTotalEarnings() +order.getTotalSellingPrice());
        }

        report.setTotalSales(report.getTotalSales() +order.getOrderItems().size());
        sellerReportService.updateSellerReport(report);
    }
    Cart cart=cartRepository.findById(user.getId());
    cart.setCouponPrice(0);
    cart.setCouponCode(null);
//    Set<CartItem> items=cart.getCartItems();
//    cartItemRepository.deleteAll(items);
//    cart.setCartItems(new HashSet<>());
//    cartRepository.save(cart);

}

ApiResponse res = new ApiResponse();
res.setMessage("Payment successful");
res.setStatus(true);

return new ResponseEntity<>(res, HttpStatus.CREATED);
}
}

package com.zosh.controller;

import com.zosh.exception.*;
import com.zosh.model.*;

```

```
import com.zosh.request.CreateProductRequest;
import com.zosh.service.*;
import lombok.RequiredArgsConstructor;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.data.domain.Page;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.*;

import java.util.List;

@RestController
@RequestMapping("/products")
@RequiredArgsConstructor
public class ProductController {

    private final ProductService productService;
    private final UserService userService;
    private final SellerService sellerService;

    @GetMapping("/{productId}")
    public ResponseEntity<Product> getProductById(@PathVariable Long productId)
            throws ProductException {
        Product product = productService.findProductById(productId);
        return new ResponseEntity<>(product, HttpStatus.OK);
    }

    @GetMapping("/search")
    public ResponseEntity<List<Product>> searchProduct(@RequestParam(required =
false) String query) {
        List<Product> products = productService.searchProduct(query);
        return new ResponseEntity<>(products, HttpStatus.OK);
    }

    @GetMapping
    public ResponseEntity<Page<Product>> getAllProducts(@RequestParam(required =
false) String category,
                                                       @RequestParam(required =
false) String brand,
                                                       @RequestParam(required =
false) String color,
```

```

        @RequestParam(required =
false) String size,
        @RequestParam(required =
false) Integer minPrice,
        @RequestParam(required =
false) Integer maxPrice,
        @RequestParam(required =
false) Integer minDiscount,
        @RequestParam(required =
false) String sort,
        @RequestParam(required =
false) String stock,

@RequestParam(defaultValue = "0") Integer pageNumber) {
    System.out.println("color p ----- "+pageNumber);
    return new ResponseEntity<>(
        productService.getAllProduct(category, brand,
            color, size, minPrice,
            maxPrice, minDiscount, sort,
            stock, pageNumber), HttpStatus.OK);
}
}

package com.zosh.controller;

import com.zosh.dto.RevenueChart;
import com.zosh.exception.SellerException;
import com.zosh.model.Seller;
import com.zosh.service.RevenueService;
import com.zosh.service.SellerService;
import lombok.RequiredArgsConstructor;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.*;

import java.util.List;
import java.util.Map;

@RestController
@RequiredArgsConstructor
@RequestMapping("/api/seller/revenue/chart")
public class RevenueController {
    private final RevenueService revenueService;
    private final SellerService sellerService;

    @GetMapping()
    public ResponseEntity<List<RevenueChart>> getRevenueChart(
        @RequestParam(defaultValue = "today") String type,
        @RequestHeader("Authorization") String jwt) throws SellerException {
        Seller seller = sellerService.getSellerProfile(jwt);
        List<RevenueChart> revenue = revenueService

```

```
        .getRevenueChartByType(type, seller.getId());
    return ResponseEntity.ok(revenue);
}

}

package com.zosh.controller;

import com.zosh.exception.ProductException;
import com.zosh.exception.NotFoundException;
import com.zosh.exception.UserException;
import com.zosh.model.Product;
import com.zosh.model.Review;
import com.zosh.model.User;
import com.zosh.request.CreateReviewRequest;
import com.zosh.response.ApiResponse;
import com.zosh.service.ProductService;
import com.zosh.service.ReviewService;
import com.zosh.service.UserService;
import lombok.RequiredArgsConstructor;
import org.springframework.data.web.config.EnableSpringDataWebSupport;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.*;

import javax.naming.AuthenticationException;
import java.util.List;

@RestController
@RequiredArgsConstructor
@RequestMapping("/api")
public class ReviewController {

    private final ReviewService reviewService;
    private final UserService userService;
    private final ProductService productService;

    @GetMapping("/products/{productId}/reviews")
    public ResponseEntity<List<Review>> getReviewsByProductId(
            @PathVariable Long productId) {

        List<Review> reviews = reviewService.getReviewsByProductId(productId);
        return ResponseEntity.ok(reviews);
    }

    @PostMapping("/products/{productId}/reviews")
    public ResponseEntity<Review> writeReview(
            @RequestBody CreateReviewRequest req,
            @PathVariable Long productId,
```

```
    @RequestHeader("Authorization") String jwt) throws UserException,
ProductException {

    User user = userService.findUserProfileByJwt(jwt);
    Product product = productService.findProductById(productId);

    Review review = reviewService.createReview(
        req, user, product
    );
    return ResponseEntity.ok(review);

}

@RequestMapping("/reviews/{reviewId}")
public ResponseEntity<Review> updateReview(
    @RequestBody CreateReviewRequest req,
    @PathVariable Long reviewId,
    @RequestHeader("Authorization") String jwt)
throws UserException,
ReviewNotFoundException, AuthenticationException {

    User user = userService.findUserProfileByJwt(jwt);

    Review review = reviewService.updateReview(
        reviewId,
        req.getReviewText(),
        req.getReviewRating(),
        user.getId()
    );
    return ResponseEntity.ok(review);
}

@DeleteMapping("/reviews/{reviewId}")
public ResponseEntity<ApiResponse> deleteReview(
    @PathVariable Long reviewId,
    @RequestHeader("Authorization") String jwt) throws UserException,
ReviewNotFoundException, AuthenticationException {

    User user = userService.findUserProfileByJwt(jwt);

    reviewService.deleteReview(reviewId, user.getId());
    ApiResponse res = new ApiResponse();
    res.setMessage("Review deleted successfully");
    res.setStatus(true);

    return ResponseEntity.ok(res);
}
```

```
}

package com.zosh.controller;

import com.zosh.config.JwtProvider;
import com.zosh.domain.AccountStatus;
import com.zosh.domain.USER_ROLE;
import com.zosh.exception.SellerException;
import com.zosh.model.Seller;
import com.zosh.model.SellerReport;
import com.zosh.model.VerificationCode;
import com.zosh.repository.VerificationCodeRepository;
import com.zosh.response.ApiResponse;
import com.zosh.response.AuthResponse;
import com.zosh.service.*;
import com.zosh.service.impl.CustomeUserServiceImplementation;
import com.zosh.utils.OtpUtils;
import jakarta.mail.MessagingException;
import lombok.RequiredArgsConstructor;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.security.authentication.BadCredentialsException;
import org.springframework.security.authentication.UsernamePasswordAuthenticationToken;
;
import org.springframework.security.core.Authentication;
import org.springframework.security.core.GrantedAuthority;
import org.springframework.security.core.context.SecurityContextHolder;
import org.springframework.security.core.userdetails.UserDetails;
import org.springframework.web.bind.annotation.*;

import java.util.Collection;
import java.util.List;

@RestController
@RequestMapping("/sellers")
@RequiredArgsConstructor
public class SellerController {

    private final SellerService sellerService;
    private final SellerReportService sellerReportService;
    private final EmailService emailService;
    private final VerificationCodeRepository verificationCodeRepository;
    private final VerificationService verificationService;
    private final JwtProvider jwtProvider;
    private final CustomeUserServiceImplementation
customeUserServiceImplementation;
```

```

    @PostMapping("/sent/login-top")
    public ResponseEntity<ApiResponse> sentLoginOtp(@RequestBody
VerificationCode req) throws MessagingException, SellerException {
        Seller seller = sellerService.getSellerByEmail(req.getEmail());

        String otp = OtpUtils.generateOTP();
        VerificationCode verificationCode =
verificationService.createVerificationCode(otp, req.getEmail());

        String subject = "Zosh Bazaar Login Otp";
        String text = "your login otp is - ";
        emailService.sendVerificationOtpEmail(req.getEmail(),
verificationCode.getOtp(), subject, text);

        ApiResponse res = new ApiResponse();
        res.setMessage("otp sent");
        return new ResponseEntity<>(res, HttpStatus.CREATED);
    }

    @PostMapping("/verify/login-top")
    public ResponseEntity<AuthResponse> verifyLoginOtp(@RequestBody
VerificationCode req) throws MessagingException, SellerException {
//        Seller savedSeller = sellerService.createSeller(seller);

        String otp = req.getOtp();
        String email = req.getEmail();
        VerificationCode verificationCode =
verificationCodeRepository.findByEmail(email);

        if (verificationCode == null || !verificationCode.getOtp().equals(otp))
{
            throw new SellerException("wrong otp...");
        }

        Authentication authentication = authenticate(req.getEmail());
        SecurityContextHolder.getContext().setAuthentication(authentication);

        String token = jwtProvider.generateToken(authentication);
        AuthResponse authResponse = new AuthResponse();

        authResponse.setMessage("Login Success");
        authResponse.setJwt(token);
        Collection<? extends GrantedAuthority> authorities =
authentication.getAuthorities();

        String roleName = authorities.isEmpty() ? null :
authorities.iterator().next().getAuthority();
    }
}

```

```

        authResponse.setRole(USER_ROLE.valueOf(roleName));

        return new ResponseEntity<AuthResponse>(authResponse, HttpStatus.OK);
    }

    private Authentication authenticate(String username) {
        UserDetails userDetailsService =
customUserServiceImpl.loadUserByUsername("seller_" + username);

        System.out.println("sign in userDetails - " + userDetailsService);

        if (userDetailsService == null) {
            System.out.println("sign in userDetailsService - null " + userDetailsService);
            throw new BadCredentialsException("Invalid username or password");
        }

        return new UsernamePasswordAuthenticationToken(userDetailsService, null,
userDetailsService.getAuthorities());
    }

    @PatchMapping("/verify/{otp}")
    public ResponseEntity<Seller> verifySellerEmail(@PathVariable String otp)
throws SellerException {

        VerificationCode verificationCode =
verificationCodeRepository.findByOtp(otp);

        if (verificationCode == null || !verificationCode.getOtp().equals(otp))
{
            throw new SellerException("wrong otp...");
        }

        Seller seller = sellerService.verifyEmail(verificationCode.getEmail(),
otp);

        return new ResponseEntity<>(seller, HttpStatus.OK);
    }

    @PostMapping
    public ResponseEntity<Seller> createSeller(@RequestBody Seller seller)
throws SellerException, MessagingException {
        Seller savedSeller = sellerService.createSeller(seller);

        String otp = OtpUtils.generateOTP();
    }
}

```

```

        VerificationCode verificationCode =
verificationService.createVerificationCode(otp, seller.getEmail());

        String subject = "Zosh Bazaar Email Verification Code";
        String text = "Welcome to Zosh Bazaar, verify your account using this
link ";
        String frontend_url = "http://localhost:3000/verify-seller/";
        emailService.sendVerificationOtpEmail(seller.getEmail(),
verificationCode.getOtp(), subject, text + frontend_url);
        return new ResponseEntity<>(savedSeller, HttpStatus.CREATED);
    }

    @GetMapping("/{id}")
    public ResponseEntity<Seller> getSellerById(@PathVariable Long id) throws
SellerException {
    Seller seller = sellerService.getSellerById(id);
    return new ResponseEntity<>(seller, HttpStatus.OK);
}

    @GetMapping("/profile")
    public ResponseEntity<Seller> getSellerByJwt(
        @RequestHeader("Authorization") String jwt) throws SellerException {
    String email = jwtProvider.getEmailFromJwtToken(jwt);
    Seller seller = sellerService.getSellerByEmail(email);
    return new ResponseEntity<>(seller, HttpStatus.OK);
}

    @GetMapping("/report")
    public ResponseEntity<SellerReport> getSellerReport(
        @RequestHeader("Authorization") String jwt) throws SellerException {
    String email = jwtProvider.getEmailFromJwtToken(jwt);
    Seller seller = sellerService.getSellerByEmail(email);
    SellerReport report = sellerReportService.getSellerReport(seller);
    return new ResponseEntity<>(report, HttpStatus.OK);
}

    @GetMapping
    public ResponseEntity<List<Seller>> getAllSellers(
        @RequestParam(required = false) AccountStatus status) {
    List<Seller> sellers = sellerService.getAllSellers(status);
    return ResponseEntity.ok(sellers);
}

    @PatchMapping()
    public ResponseEntity<Seller> updateSeller(
        @RequestHeader("Authorization") String jwt, @RequestBody Seller
seller) throws SellerException {

    Seller profile = sellerService.getSellerProfile(jwt);
}

```

```
        Seller updatedSeller = sellerService.updateSeller(profile.getId(),
seller);
        return ResponseEntity.ok(updatedSeller);

    }

    @DeleteMapping("/{id}")
    public ResponseEntity<Void> deleteSeller(@PathVariable Long id) throws
SellerException {

        sellerService.deleteSeller(id);
        return ResponseEntity.noContent().build();

    }
}
package com.zosh.controller;

import com.zosh.domain.OrderStatus;
import com.zosh.exception.OrderException;
import com.zosh.exception.SellerException;
import com.zosh.model.Order;
import com.zosh.model.Seller;
import com.zosh.response.ApiResponse;
import com.zosh.service.OrderService;
import com.zosh.service.SellerService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.*;

import java.util.List;

@RestController
@RequestMapping("/seller/orders")
public class SellerOrderController {

    private final OrderService orderService;

    private final SellerService sellerService;

    @Autowired
    public SellerOrderController(OrderService orderService,
                                SellerService sellerService) {
        this.orderService=orderService;

        this.sellerService = sellerService;
    }
}
```

```

    @GetMapping()
    public ResponseEntity<List<Order>> getAllOrdersHandler(
        @RequestHeader("Authorization") String jwt
    ) throws SellerException {
        Seller seller=sellerService.getSellerProfile(jwt);
        List<Order> orders=orderService.getShopsOrders(seller.getId());
        return new ResponseEntity<>(orders, HttpStatus.ACCEPTED);
    }

    @PatchMapping("/{orderId}/status/{orderStatus}")
    public ResponseEntity<Order> updateOrderHandler(
        @RequestHeader("Authorization") String jwt,
        @PathVariable Long orderId,
        @PathVariable OrderStatus orderStatus
    ) throws OrderException {
        Order orders=orderService.updateOrderStatus(orderId,orderStatus);
        return new ResponseEntity<>(orders,HttpStatus.ACCEPTED);
    }

    @DeleteMapping("/{orderId}/delete")
    public ResponseEntity<ApiResponse> deleteOrderHandler(@PathVariable Long orderId,
    @RequestHeader("Authorization") String jwt) throws OrderException{
        orderService.deleteOrder(orderId);
        ApiResponse res=new ApiResponse("Order Deleted Successfully",true);
        System.out.println("delete method working....");
        return new ResponseEntity<>(res,HttpStatus.ACCEPTED);
    }
}

package com.zosh.controller;

import com.zosh.exception.CategoryNotFoundException;
import com.zosh.exception.ProductException;
import com.zosh.exception.SellerException;
import com.zosh.exception.UserException;
import com.zosh.model.Product;
import com.zosh.model.Seller;
import com.zosh.request.CreateProductRequest;
import com.zosh.service.ProductService;
import com.zosh.service.SellerService;
import com.zosh.service.UserService;
import lombok.RequiredArgsConstructor;

```

```
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.*;

import java.util.List;

@RestController
@RequestMapping("/sellers/product")
@RequiredArgsConstructor
public class SellerProductController {

    private final ProductService productService;
    private final SellerService sellerService;
    private final UserService userService;

    @GetMapping()
    public ResponseEntity<List<Product>> getProductBySellerId(
        @RequestHeader("Authorization") String jwt) throws ProductException,
SellerException {

        Seller seller=sellerService.getSellerProfile(jwt);

        List<Product> products =
productService.getProductBySellerId(seller.getId());
        return new ResponseEntity<>(products, HttpStatus.OK);

    }

    @PostMapping()
    public ResponseEntity<Product> createProduct(
        @RequestBody CreateProductRequest request,

        @RequestHeader("Authorization")String jwt)
        throws UserException,
        ProductException, CategoryNotFoundException, SellerException {

        Seller seller=sellerService.getSellerProfile(jwt);

        Product product = productService.createProduct(request, seller);
        return new ResponseEntity<>(product, HttpStatus.CREATED);

    }

    @DeleteMapping("/{productId}")
    public ResponseEntity<Void> deleteProduct(@PathVariable Long productId) {
        try {
            productService.deleteProduct(productId);
        }
    }
}
```

```

        return new ResponseEntity<>(HttpStatus.OK);
    } catch (ProductException e) {
        return new ResponseEntity<>(HttpStatus.NOT_FOUND);
    }
}

@PatchMapping("/{productId}")
public ResponseEntity<Product> updateProduct(@PathVariable Long productId,
@RequestParam Product product) {
    try {
        Product updatedProduct = productService.updateProduct(productId,
product);
        return new ResponseEntity<>(updatedProduct, HttpStatus.OK);
    } catch (ProductException e) {
        return new ResponseEntity<>(HttpStatus.NOT_FOUND);
    }
}

@PatchMapping("/{productId}/stock")
public ResponseEntity<Product> updateProductStock(@PathVariable Long productId) {
    try {
        Product updatedProduct =
productService.updateProductStock(productId);
        return new ResponseEntity<>(updatedProduct, HttpStatus.OK);
    } catch (ProductException e) {
        return new ResponseEntity<>(HttpStatus.NOT_FOUND);
    }
}

}

package com.zosh.controller;

import com.zosh.exception.SellerException;
import com.zosh.model.Order;
import com.zosh.model.Seller;
import com.zosh.model.Transaction;
import com.zosh.service.SellerService;
import com.zosh.service.TransactionService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.*;

import java.util.List;

@RestController
@RequestMapping("/api/transactions")
public class TransactionController {

```

```

private final TransactionService transactionService;
private final SellerService sellerService;

@Autowired
public TransactionController(TransactionService transactionService,
SellerService sellerService) {
    this.transactionService = transactionService;
    this.sellerService = sellerService;
}

@PostMapping
public ResponseEntity<Transaction> createTransaction(@RequestBody Order
order) {
    Transaction transaction = transactionService.createTransaction(order);
    return ResponseEntity.ok(transaction);
}

@GetMapping("/seller")
public ResponseEntity<List<Transaction>> getTransactionBySeller(
    @RequestHeader("Authorization") String jwt) throws SellerException {
    Seller seller=sellerService.getSellerProfile(jwt);

    List<Transaction> transactions =
transactionService.getTransactionBySeller(seller);
    return ResponseEntity.ok(transactions);
}

@GetMapping
public ResponseEntity<List<Transaction>> getAllTransactions() {
    List<Transaction> transactions =
transactionService.getAllTransactions();
    return ResponseEntity.ok(transactions);
}
}

package com.zosh.controller;

import com.zosh.model.User;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RequestHeader;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

import com.zosh.exception.UserException;

import com.zosh.service.UserService;

```

```
@RestController
@RequestMapping("/api/users")
public class UserController {

    private final UserService userService;

    public UserController(UserService userService) {
        this.userService=userService;
    }

    @GetMapping("/profile")
    public ResponseEntity<User> getUserProfileHandler(
        @RequestHeader("Authorization") String jwt) throws UserException{

        System.out.println("/api/users/profile");
        User user=userService.findUserProfileByJwt(jwt);
        return new ResponseEntity<>(user,HttpStatus.ACCEPTED);
    }

}

package com.zosh.controller;

import com.zosh.exception.ProductException;
import com.zosh.exception.UserException;
import com.zosh.exception.WishlistNotFoundException;
import com.zosh.model.Product;
import com.zosh.model.User;
import com.zosh.model.Wishlist;
import com.zosh.response.ApiResponse;
import com.zosh.service.ProductService;
import com.zosh.service.UserService;
import com.zosh.service.WishlistService;
import lombok.RequiredArgsConstructor;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.*;

@RestController
@RequestMapping("/api/wishlist")
@RequiredArgsConstructor
public class WishlistController {

    private final WishlistService wishlistService;
    private final ProductService productService;
    private final UserService userService;

    @PostMapping("/create")
```

```

public ResponseEntity<Wishlist> createWishlist(@RequestBody User user) {
    Wishlist wishlist = wishlistService.createWishlist(user);
    return ResponseEntity.ok(wishlist);
}

@GetMapping()
public ResponseEntity<Wishlist> getWishlistByUserId(
    @RequestHeader("Authorization") String jwt) throws UserException {

    User user = userService.findUserProfileByJwt(jwt);
    Wishlist wishlist = wishlistService.getWishlistByUserId(user);
    return ResponseEntity.ok(wishlist);
}

@PostMapping("/add-product/{productId}")
public ResponseEntity<Wishlist> addProductToWishlist(
    @PathVariable Long productId,
    @RequestHeader("Authorization") String jwt) throws
WishlistNotFoundException, ProductException, UserException {

    Product product = productService.findProductById(productId);
    User user=userService.findUserProfileByJwt(jwt);
    Wishlist updatedWishlist = wishlistService.addProductToWishlist(
        user,
        product
    );
    return ResponseEntity.ok(updatedWishlist);
}

}

package com.zosh.domain;

public enum AccountStatus {
    PENDING_VERIFICATION, // Account is created but not yet verified
    ACTIVE, // Account is active and in good standing
    SUSPENDED, // Account is temporarily suspended, possibly due to
    violations
    DEACTIVATED, // Account is deactivated, user may have chosen to
deactivate it
    BANNED, // Account is permanently banned due to severe
    violations
    CLOSED // Account is permanently closed, possibly at user
    request
}

```

```
package com.zosh.domain;

public enum HomeCategorySection {
    ELECTRIC_CATEGORIES,
    GRID,
    SHOP_BY_CATEGORIES,
    DEALS
}
package com.zosh.domain;

public enum OrderStatus {
    PENDING,
    PLACED,
    CONFIRMED,
    SHIPPED,
    DELIVERED,
    CANCELLED
}
package com.zosh.domain;

public enum PaymentMethod {
    RAZORPAY,
    STRIPE
}
package com.zosh.domain;

public enum PaymentOrderStatus {
    PENDING, SUCCESS, FAILED
}
package com.zosh.domain;

public enum PaymentStatus {

    PENDING,
    PROCESSING,
    COMPLETED,
    FAILED
}
package com.zosh.domain;

public enum PayoutsStatus {
    PENDING,
    SUCCESS
}
package com.zosh.domain;

public enum USER_ROLE {
```

```
    ROLE_CUSTOMER,
    ROLE_SELLER,
    ROLE_ADMIN

}

package com.zosh.dto;

import com.zosh.domain.OrderStatus;
import com.zosh.domain.PaymentStatus;
import com.zosh.model.Address;
import com.zosh.model.OrderItem;
import com.zosh.model.PaymentDetails;
import com.zosh.model.User;
import jakarta.persistence.*;
import lombok.Data;

import java.time.LocalDate;
import java.time.LocalDateTime;
import java.util.ArrayList;
import java.util.List;

@Data
public class OrderDto {

    private Long id;

    private String orderId;

    private UserDto user;

    private Long sellerId;

    private List<OrderItemDto> orderItems = new ArrayList<>();

    private Address shippingAddress;

    private PaymentDetails paymentDetails=new PaymentDetails();

    private double totalMrpPrice;

    private Integer totalSellingPrice;

    private Integer discount;

    private OrderStatus orderStatus;

    private int totalItem;
```

```
private PaymentStatus paymentStatus=PaymentStatus.PENDING;

private LocalDateTime orderDate = LocalDateTime.now();
private LocalDateTime deliverDate = orderDate.plusDays(7);

}

package com.zosh.dto;

import com.zosh.model.User;
import lombok.Data;

import java.util.List;

@Data
public class OrderHistory {
    private Long id;
    private UserDto user;
    private List<OrderDto> currentOrders;
    private int totalOrders;
    private int cancelledOrders;
    private int completedOrders;
    private int pendingOrders;
}
package com.zosh.dto;

import com.fasterxml.jackson.annotation.JsonIgnore;
import com.zosh.model.Order;
import com.zosh.model.Product;
import jakarta.persistence.GeneratedValue;
import jakarta.persistence.GenerationType;
import jakarta.persistence.Id;
import jakarta.persistence.ManyToOne;
import lombok.Data;

@Data
public class OrderItemDto {

    private Long id;

    private ProductDto product;

    private String size;

    private int quantity;

    private Integer mrpPrice;

    private Integer sellingPrice;
```

```
    private Long userId;

}

package com.zosh.dto;

import com.zosh.model.Category;
import com.zosh.model.Review;
import com.zosh.model.Seller;
import jakarta.persistence.*;
import lombok.*;

import java.time.LocalDateTime;
import java.util.ArrayList;
import java.util.List;

@Data
public class ProductDto {

    private Long id;

    private String title;

    private String description;

    private int mrpPrice;

    private int sellingPrice;

    private int discountPercent;

    private int quantity;

    private String color;

    private List<String> images = new ArrayList<>();

    private int numRatings;

    private LocalDateTime createdAt;

    private String Sizes;

}

package com.zosh.dto;

import lombok.Data;
```

```
@Data
public class RevenueChart {
    private String date;
    private double revenue;
}
package com.zosh.dto;

import lombok.Data;

@Data
public class UserDto {
    private Long id;
    private String fullName;
    private String email;
    private String profilePicture;
}
package com.zosh.exception;

public class CartItemException extends Exception {

    public CartItemException(String message) {
        super(message);
        // TODO Auto-generated constructor stub
    }

}
package com.zosh.exception;

public class CategoryNotFoundException extends Exception {
    public CategoryNotFoundException(String categoryNotFound) {
        super(categoryNotFound);
    }
}
package com.zosh.exception;

public class CouponNotValidException extends Exception {
    public CouponNotValidException(String message) {
        super(message);
    }
}
package com.zosh.exception;

import java.time.LocalDateTime;

public class ErrorDetails {

    private String error;
    private String details;
    private LocalDateTime timestamp;
```

```
public ErrorDetails() {
    // TODO Auto-generated constructor stub
}

public ErrorDetails(String error, String details, LocalDateTime timestamp) {
    super();
    this.error = error;
    this.details = details;
    this.timestamp = timestamp;
}

public String getError() {
    return error;
}

public void setError(String error) {
    this.error = error;
}

public String getDetails() {
    return details;
}

public void setDetails(String details) {
    this.details = details;
}

public LocalDateTime getTimestamp() {
    return timestamp;
}

public void setTimestamp(LocalDateTime timestamp) {
    this.timestamp = timestamp;
}

}

package com.zosh.exception;

import java.net.http.HttpHeaders;
import java.time.LocalDateTime;
import java.util.LinkedHashMap;
import java.util.Map;

import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.MethodArgumentNotValidException;
import org.springframework.web.bind.annotation.ControllerAdvice;
import org.springframework.web.bind.annotation.ExceptionHandler;
```

```
import org.springframework.web.context.request.WebRequest;
import org.springframework.web.servlet.NoHandlerFoundException;

@ControllerAdvice
public class GlobleException {

    @ExceptionHandler(UserException.class)
    public ResponseEntity<ErrorDetails> UserExceptionHandler(UserException ue,
    WebRequest req) {

        ErrorDetails err= new
ErrorDetails(ue.getMessage(),req.getDescription(false),LocalDateTime.now());

        return new ResponseEntity<ErrorDetails>(err,HttpStatus.BAD_REQUEST);

    }

    @ExceptionHandler(ProductException.class)
    public ResponseEntity<ErrorDetails> ProductExceptionHandler(ProductException
ue, WebRequest req) {

        ErrorDetails err= new
ErrorDetails(ue.getMessage(),req.getDescription(false),LocalDateTime.now());

        return new ResponseEntity<ErrorDetails>(err,HttpStatus.BAD_REQUEST);

    }

    @ExceptionHandler(CartItemException.class)
    public ResponseEntity<ErrorDetails>
CartItemExceptionHandler(CartItemException ue, WebRequest req) {

        ErrorDetails err= new
ErrorDetails(ue.getMessage(),req.getDescription(false),LocalDateTime.now());

        return new ResponseEntity<ErrorDetails>(err,HttpStatus.BAD_REQUEST);

    }

    @ExceptionHandler(OrderException.class)
    public ResponseEntity<ErrorDetails> OrderExceptionHandler(OrderException ue,
    WebRequest req) {

        ErrorDetails err= new ErrorDetails(ue.getMessage(),
            req.getDescription(false),LocalDateTime.now());
    }
}
```

```

        return new ResponseEntity<ErrorDetails>(err, HttpStatus.BAD_REQUEST);

    }

    @ExceptionHandler(SellerException.class)
    public ResponseEntity<ErrorDetails> handleSellerException(SellerException
ex, WebRequest req) {
    ErrorDetails err= new ErrorDetails(ex.getMessage(),
        req.getDescription(false),
        LocalDateTime.now());

    return new ResponseEntity<ErrorDetails>(err, HttpStatus.BAD_REQUEST);
}

    @ExceptionHandler(CouponNotValidException.class)
    public ResponseEntity<ErrorDetails> CouponNotValidExceptionHandler(
        CouponNotValidException ue,
        WebRequest req) {

    ErrorDetails err= new ErrorDetails(
        ue.getMessage(),
        req.getDescription(false),
        LocalDateTime.now()
    );

    return new ResponseEntity<>(err, HttpStatus.BAD_REQUEST);
}

    @ExceptionHandler(MethodArgumentNotValidException.class)
    public ResponseEntity<ErrorDetails>
methodArgumentNotValidExceptionHandler(MethodArgumentNotValidException me) {
    ErrorDetails err=new
ErrorDetails(me.getBindingResult().getFieldError().getDefaultMessage(), "validat
ion error", LocalDateTime.now());
    return new ResponseEntity<ErrorDetails>(err, HttpStatus.BAD_REQUEST);
}

    @ExceptionHandler(NoHandlerFoundException.class)
    public ResponseEntity<Object>
handleNoHandlerFoundException(NoHandlerFoundException ex, HttpHeaders headers,
HttpStatus status, WebRequest request) {
    Map<String, Object> body = new LinkedHashMap<>();
    body.put("message", "Endpoint not found");

    return new ResponseEntity<>(body, HttpStatus.NOT_FOUND);
}

```

```
@ExceptionHandler(RuntimeException.class)
public ResponseEntity<String> handleRuntimeException(RuntimeException ex) {
    return new ResponseEntity<>(ex.getMessage() ,
HttpStatus.INTERNAL_SERVER_ERROR);
}

@ExceptionHandler(Exception.class)
public ResponseEntity<ErrorDetails> otherExceptionHandler(Exception e,
WebRequest req) {
    ErrorDetails error=new
ErrorDetails(e.getMessage(),req.getDescription(false),LocalDateTime.now());
    return new ResponseEntity<ErrorDetails>(error,HttpStatus.ACCEPTED);
}

}
package com.zosh.exception;

public class OrderException extends Exception {

    public OrderException(String message) {
        super(message);
    }
}

}
package com.zosh.exception;

public class ProductException extends Exception{

    public ProductException(String message) {
        super(message);
    }
}

}
package com.zosh.exception;

public class ReviewNotFoundException extends Exception {
    public ReviewNotFoundException(String message) {
        super(message);
    }
}

}
package com.zosh.exception;

public class SellerException extends Exception {

    public SellerException(String message) {
        super(message);
    }
}
```



```

        orderItemDto.setId(orderItem.getId());

        orderItemDto.setProduct(ProductMapper.toProductDto(orderItem.getProduct()));
        orderItemDto.setSize(orderItem.getSize());
        orderItemDto.setQuantity(orderItem.getQuantity());
        orderItemDto.setMrpPrice(orderItem.getMrpPrice());
        orderItemDto.setSellingPrice(orderItem.getSellingPrice());
//        orderItemDto.setUserId(orderItem.getUser().getId());

        return orderItemDto;
    }

    // Maps OrderItemDto to OrderItem
    public static OrderItem toOrderItem(OrderItemDto orderItemDto) {
        if (orderItemDto == null) {
            return null;
        }

        OrderItem orderItem = new OrderItem();
        orderItem.setId(orderItemDto.getId());
//
        orderItem.setProduct(ProductMapper.toProductDto(orderItemDto.getProduct()));
        orderItem.setSize(orderItemDto.getSize());
        orderItem.setQuantity(orderItemDto.getQuantity());
        orderItem.setMrpPrice(orderItemDto.getMrpPrice());
        orderItem.setSellingPrice(orderItemDto.getSellingPrice());

        // Assuming that the User is fetched separately and set here
        return orderItem;
    }

    // Maps Order to OrderDto
    public static OrderDto toOrderDto(Order order) {
        if (order == null) {
            return null;
        }

        OrderDto orderDto = new OrderDto();
        orderDto.setId(order.getId());
        orderDto.setOrderId(order.getOrderId());
        orderDto.setUser(UserMapper.toUserDto(order.getUser()));
        orderDto.setSellerId(order.getSellerId());

        orderDto.setOrderItems(order.getOrderItems().stream().map(OrderMapper::toOrderItemDto).collect(Collectors.toList()));
        orderDto.setShippingAddress(order.getShippingAddress());
        orderDto.setPaymentDetails(order.getPaymentDetails());
        orderDto.setTotalMrpPrice(order.getTotalMrpPrice());
        orderDto.setTotalSellingPrice(order.getTotalSellingPrice());
    }
}

```

```

        orderDto.setDiscount(order.getDiscount());
        orderDto.setOrderStatus(order.getOrderStatus());
        orderDto.setTotalItem(order.getTotalItem());
        orderDto.setPaymentStatus(order.getPaymentStatus());
        orderDto.setOrderDate(order.getOrderDate());
        orderDto.setDeliverDate(order.getDeliverDate());

        return orderDto;
    }

    // Maps OrderDto to Order
    public static Order toOrder(OrderDto orderDto) {
        if (orderDto == null) {
            return null;
        }

        Order order = new Order();
        order.setId(orderDto.getId());
        order.setOrderId(orderDto.getOrderId());
//        order.setUser(UserMapper.toUser(orderDto.getUser()));
        order.setSellerId(orderDto.getSellerId());
//
        order.setOrderItems(orderDto.getOrderItems().stream().map(OrderMapper::toOrderItem)
            .collect(Collectors.toList()));
        order.setShippingAddress(orderDto.getShippingAddress());
        order.setPaymentDetails(orderDto.getPaymentDetails());
        order.setTotalMrpPrice(orderDto.getTotalMrpPrice());
        order.setTotalSellingPrice(orderDto.getTotalSellingPrice());
        order.setDiscount(orderDto.getDiscount());
        order.setOrderStatus(orderDto.getOrderStatus());
        order.setTotalItem(orderDto.getTotalItem());
        order.setPaymentStatus(orderDto.getPaymentStatus());
        order.setOrderDate(orderDto.getOrderDate());
        order.setDeliverDate(orderDto.getDeliverDate());

        return order;
    }

    // Maps OrderHistory to OrderHistoryDto
    public static OrderHistory toOrderHistory(List<Order> orders, User user) {
        if (orders == null || user == null) {
            return null;
        }

        OrderHistory orderHistory = new OrderHistory();

        // Set user
        orderHistory.setUser(UserMapper.toUserDto(user));

```

```

        // Filter current orders (those that are not DELIVERED or CANCELLED)
        List<OrderDto> currentOrders = orders.stream()
            .filter(order -> order.getOrderStatus() != OrderStatus.DELIVERED
&& order.getOrderStatus() != OrderStatus.CANCELLED)
            .map(OrderMapper::toOrderDto)
            .collect(Collectors.toList());

        orderHistory.setCurrentOrders(currentOrders);

        // Set total orders
        orderHistory.setTotalOrders(orders.size());

        // Set cancelled orders
        int cancelledOrders = (int) orders.stream()
            .filter(order -> order.getOrderStatus() ==
OrderStatus.CANCELLED)
            .count();
        orderHistory.setCancelledOrders(cancelledOrders);

        // Set completed orders (those that are DELIVERED)
        int completedOrders = (int) orders.stream()
            .filter(order -> order.getOrderStatus() ==
OrderStatus.DELIVERED)
            .count();
        orderHistory.setCompletedOrders(completedOrders);

        return orderHistory;
    }

}

package com.zosh.mapper;

import com.zosh.dto.ProductDto;
import com.zosh.model.Product;

public class ProductMapper {

    public static ProductDto toProductDto(Product product) {
        ProductDto productDto = new ProductDto();
        productDto.setId(product.getId());
        productDto.setTitle(product.getTitle());
        productDto.setDescription(product.getDescription());
        productDto.setMrpPrice(product.getMrpPrice());
        productDto.setSellingPrice(product.getSellingPrice());
        productDto.setDiscountPercent(product.getDiscountPercent());
        productDto.setQuantity(product.getQuantity());
        productDto.setColor(product.getColor());
    }
}

```

```
        productDto.setImages(product.getImages());
        productDto.setNumRatings(product.getNumRatings());
        productDto.setCreatedAt(product.getCreatedAt());
        productDto.setSizes(product.getSizes());

        return productDto;
    }
    public Product mapToEntity(ProductDto productDto) {
        return null;
    }
}
package com.zosh.mapper;

import com.zosh.dto.OrderDto;
import com.zosh.dto.OrderItemDto;
import com.zosh.dto.UserDto;
import com.zosh.model.Order;
import com.zosh.model.OrderItem;
import com.zosh.model.User;

public class UserMapper {

    public static UserDto toUserDto(User user) {
        UserDto userDto = new UserDto();
        userDto.setId(user.getId());
        userDto.setFullName(user.getFullName());
        userDto.setEmail(user.getEmail());
        return userDto;
    }

}
package com.zosh.model;

import jakarta.persistence.Entity;
import jakarta.persistence.GeneratedValue;
import jakarta.persistence.GenerationType;
import jakarta.persistence.Id;
import lombok.*;


@Entity
@Getter
@Setter
@AllArgsConstructor
@NoArgsConstructor
@EqualsAndHashCode
public class Address {

    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
```

```
    private Long id;

    private String name;

    private String locality;

    private String address;

    private String city;

    private String state;

    private String pinCode;

    private String mobile;

}

package com.zosh.model;

import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;

@Data
@NoArgsConstructor
@AllArgsConstructor
public class BankDetails {

    private String accountNumber;
    private String accountHolderName;
//    private String bankName;
    private String ifscCode;

}
package com.zosh.model;

import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;
```

```
@Data
@NoArgsConstructor
@AllArgsConstructor
public class BusinessDetails {

    private String businessName;
    private String businessEmail;
    private String businessMobile;
    private String businessAddress;
    private String logo;
    private String banner;

}

package com.zosh.model;

import java.util.HashSet;
import java.util.Set;

import jakarta.persistence.CascadeType;
import jakarta.persistence.Column;
import jakarta.persistence.Entity;
import jakarta.persistence.FetchType;
import jakarta.persistence.GeneratedValue;
import jakarta.persistence.GenerationType;
import jakarta.persistence.Id;
import jakarta.persistence.JoinColumn;
import jakarta.persistence.OneToMany;
import jakarta.persistence.OneToOne;
import lombok.*;

@Entity
@Setter
@Getter
@NoArgsConstructor
@AllArgsConstructor
@Data
public class Cart {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    @OneToOne(fetch = FetchType.LAZY)
    private User user;

    @OneToMany(mappedBy = "cart", cascade = CascadeType.ALL, orphanRemoval =
true)
    private Set<CartItem> cartItems = new HashSet<>();
}
```

```
    private double totalSellingPrice;

    private int totalItem;

    private int totalMrpPrice;

    private int discount;

    private String couponCode;
    private int couponPrice;

}

package com.zosh.model;

import java.util.Objects;

import com.fasterxml.jackson.annotation.JsonIgnore;

import jakarta.persistence.Entity;
import jakarta.persistence.GeneratedValue;
import jakarta.persistence.GenerationType;
import jakarta.persistence.Id;
import jakarta.persistence.ManyToOne;
import lombok.*;


@Setter
@Getter
@NoArgsConstructor
@AllArgsConstructor
@Entity
public class CartItem {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    @JsonIgnore
    @ManyToOne
    private Cart cart;

    @ManyToOne
    private Product product;

    private String size;

    private int quantity;

    private Integer mrpPrice;
```

```
    private Integer sellingPrice;

    private Long userId;

}

package com.zosh.model;

import jakarta.persistence.*;
import jakarta.validation.constraints.NotNull;
import jakarta.validation.constraints.Size;
import lombok.Data;

@Entity
@Table(name = "categories")
@Data
public class Category {

    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private Long id;

    private String name;

    @NotNull
    @Column(unique = true)
    private String categoryId;

    @ManyToOne
    private Category parentCategory;

    @NotNull
    private Integer level;

}

package com.zosh.model;

import com.zosh.model.User;
import jakarta.persistence.*;
import lombok.Getter;
import lombok.NoArgsConstructor;
import lombok.Setter;
```

```
import java.time.LocalDate;
import java.util.HashSet;
import java.util.Set;

@Entity
@Getters
@Setter
@NoArgsConstructor
public class Coupon {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    @Column(unique = true, nullable = false)
    private String code;

    private double discountPercentage;

    private LocalDate validityStartDate;

    private LocalDate validityEndDate;

    private double minimumOrderValue;

    private boolean isActive=true;

    @ManyToMany(mappedBy = "usedCoupons")
    private Set<User> usedByUsers=new HashSet<>();

}

package com.zosh.model;

import jakarta.persistence.*;
import lombok.AllArgsConstructorConstructor;
import lombok.Getter;
import lombok.NoArgsConstructor;
import lombok.Setter;

@Entity
@Getter
@Setter
@AllArgsConstructorConstructor
@NoArgsConstructor
public class Deal {
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private Long id;
    private Integer discount;
```

```
@OneToOne
private HomeCategory category;

}

package com.zosh.model;

import jakarta.persistence.*;
import lombok.AllArgsConstructor;
import lombok.Getter;
import lombok.NoArgsConstructor;
import lombok.Setter;

import java.util.List;

@Getter
@Setter
@NoArgsConstructor
public class Home {

    private List<HomeCategory> grid;

    private List<HomeCategory> shopByCategories;

    private List<HomeCategory> electricCategories;

    private List<HomeCategory> dealCategories;

    private List<Deal> deals;

}
package com.zosh.model;

import com.zosh.domain.HomeCategorySection;
import jakarta.persistence.Entity;
import jakarta.persistence.GeneratedValue;
import jakarta.persistence.GenerationType;
import jakarta.persistence.Id;
import lombok.AllArgsConstructor;
import lombok.Getter;
import lombok.NoArgsConstructor;
import lombok.Setter;

@Entity
@Getter
@Setter
```

```
@NoArgsConstructor  
@AllArgsConstructor  
public class HomeCategory {  
    @Id  
    @GeneratedValue(strategy = GenerationType.AUTO)  
    private Long id;  
  
    private String name;  
    private String image;  
    private String categoryId;  
    private HomeCategorySection section;  
}  
package com.zosh.model;  
  
import java.util.Date;  
  
import jakarta.persistence.Entity;  
import jakarta.persistence.GeneratedValue;  
import jakarta.persistence.GenerationType;  
import jakarta.persistence.Id;  
import jakarta.persistence.JoinColumn;  
import jakarta.persistence.ManyToOne;  
import jakarta.persistence.Temporal;  
import jakarta.persistence.TemporalType;  
import lombok.Data;  
  
@Entity  
@Data  
public class Notification {  
    @Id  
    @GeneratedValue(strategy = GenerationType.IDENTITY)  
    private Long id;  
  
    @ManyToOne  
    @JoinColumn(name = "recipient_id")  
    private User customer;  
  
    private String message;  
  
    @Temporal(TemporalType.TIMESTAMP)  
    private Date sentAt;  
  
    private boolean readStatus;  
  
}  
package com.zosh.model;
```

```
import com.zosh.domain.OrderStatus;
import com.zosh.domain.PaymentStatus;
import jakarta.persistence.CascadeType;
import jakarta.persistence.Column;
import jakarta.persistence.Embedded;
import jakarta.persistence.Entity;
import jakarta.persistence.GeneratedValue;
import jakarta.persistence.GenerationType;
import jakarta.persistence.Id;
import jakarta.persistence.ManyToOne;
import jakarta.persistence.OneToMany;
import jakarta.persistence.OneToOne;
import jakarta.persistence.Table;
import lombok.*;

import java.time.LocalDate;
import java.time.LocalDateTime;
import java.util.ArrayList;
import java.util.List;

@Entity
@Table(name = "orders")
@Getter
@Setter
@NoArgsConstructor
@AllArgsConstructor
public class Order {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    private String orderId;

    @ManyToOne
    private User user;

    private Long sellerId;

    @OneToMany(mappedBy = "order", cascade = CascadeType.ALL, orphanRemoval =
true)
    private List<OrderItem> orderItems = new ArrayList<>();

    @ManyToOne
    private Address shippingAddress;
```

```
@Embedded
private PaymentDetails paymentDetails=new PaymentDetails();

private double totalMrpPrice;

private Integer totalSellingPrice;

private Integer discount;

private OrderStatus orderStatus;

private int totalItem;

private PaymentStatus paymentStatus=PaymentStatus.PENDING;

private LocalDateTime orderDate = LocalDateTime.now();
private LocalDateTime deliverDate = orderDate.plusDays(7);

}

package com.zosh.model;

import java.time.LocalDateTime;
import java.util.Objects;

import com.fasterxml.jackson.annotation.JsonIgnore;

import jakarta.persistence.Entity;
import jakarta.persistence.GeneratedValue;
import jakarta.persistence.GenerationType;
import jakarta.persistence.Id;
import jakarta.persistence.ManyToOne;
import lombok.*;

@Getter
@Setter
@NoArgsConstructor
@AllArgsConstructor
@Entity
public class OrderItem {

@Id
@GeneratedValue(strategy = GenerationType.IDENTITY)
private Long id;

@JsonIgnore
@ManyToOne
private Order order;
```

```
@ManyToOne
private Product product;

private String size;

private int quantity;

private Integer mrpPrice;

private Integer sellingPrice;

private Long userId;

}

package com.zosh.model;

import java.util.Date;

import jakarta.persistence.Entity;
import jakarta.persistence.FetchType;
import jakarta.persistence.GeneratedValue;
import jakarta.persistence.GenerationType;
import jakarta.persistence.Id;
import jakarta.persistence.JoinColumn;
import jakarta.persistence.ManyToOne;
import jakarta.persistence.OneToOne;
import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;
import lombok.NonNull;
import lombok.RequiredArgsConstructor;

@Entity
@Data
@NoArgsConstructor
@AllArgsConstructor
@RequiredArgsConstructor
@AllArgsConstructor
public class PasswordResetToken {

    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Id
    private Integer id;

    private @NonNull String token;

    @ManyToOne(targetEntity = User.class, fetch = FetchType.EAGER)
    private @NonNull User user;
```

```
    private @NonNull Date expiryDate;

    public boolean isExpired() {
        return expiryDate.before(new Date());
    }

}

package com.zosh.model;

import com.zosh.domain.PaymentStatus;
import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;

@Data
@NoArgsConstructor
@AllArgsConstructor
public class PaymentDetails {

    private String paymentId;
    private String razorpayPaymentLinkId;
    private String razorpayPaymentLinkReferenceId;
    private String razorpayPaymentLinkStatus;
    private String razorpayPaymentId;
    private PaymentStatus status;

}

package com.zosh.model;

import java.time.LocalDate;

import jakarta.persistence.Column;

public class PaymentInformation {

    @Column(name = "cardholder_name")
    private String cardholderName;

    @Column(name = "card_number")
    private String cardNumber;

    @Column(name = "expiration_date")
    private LocalDate expirationDate;

    @Column(name = "cvv")
    private String cvv;
```

```
// getters and setters
}

package com.zosh.model;

import com.zosh.domain.PaymentMethod;
import com.zosh.domain.PaymentOrderStatus;
import jakarta.persistence.*;
import lombok.AllArgsConstructorConstructor;
import lombok.Data;
import lombok.NoArgsConstructorConstructor;

import java.util.ArrayList;
import java.util.HashSet;
import java.util.List;
import java.util.Set;

@Entity
@AllArgsConstructorConstructor
@NoArgsConstructorConstructor
@Data
public class PaymentOrder {

    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private Long id;

    private Long amount;

    private PaymentOrderStatus status = PaymentOrderStatus.PENDING;

    private PaymentMethod paymentMethod;

    private String paymentLinkId;

    @ManyToOne
    private User user;

    @OneToMany
    private Set<Order> orders = new HashSet<>();
}

package com.zosh.model;

import com.zosh.domain.PayoutsStatus;
import jakarta.persistence.*;
import lombok.AllArgsConstructorConstructor;
```

```
import lombok.Getter;
import lombok.NoArgsConstructor;
import lombok.Setter;

import java.time.LocalDateTime;
import java.util.ArrayList;
import java.util.List;

@NoArgsConstructor
@AllArgsConstructor
@Getter
@Setter
@Entity
public class Payouts {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    @OneToMany
    private List<Transaction> transactions = new ArrayList<>();

    @ManyToOne
    private Seller seller;

    private Long amount;

    private PayoutsStatus status = PayoutsStatus.PENDING;

    private LocalDateTime date=LocalDateTime.now();
}

package com.zosh.model;

import java.time.LocalDateTime;
import java.util.ArrayList;
import java.util.HashSet;
import java.util.List;
import java.util.Objects;
import java.util.Set;

import com.fasterxml.jackson.annotation.JsonBackReference;
import com.fasterxml.jackson.annotation.JsonManagedReference;
import jakarta.persistence.*;

import lombok.*;

@Entity
@Getter
@Setter
```

```
@NoArgsConstructor  
@AllArgsConstructor  
@EqualsAndHashCode  
  
public class Product {  
  
    @Id  
    @GeneratedValue(strategy = GenerationType.AUTO)  
    private Long id;  
  
    private String title;  
  
    private String description;  
  
    private int mrpPrice;  
  
    private int sellingPrice;  
  
    private int discountPercent;  
  
    private int quantity;  
  
    private String color;  
  
    @ElementCollection  
    private List<String> images =new ArrayList<>();  
  
    private int numRatings;  
  
    @ManyToOne  
    private Category category;  
  
    @ManyToOne  
    private Seller seller;  
  
    private LocalDateTime createdAt;  
  
//    @ElementCollection  
    private String Sizes;  
  
    @OneToMany(mappedBy = "product",cascade = CascadeType.ALL,orphanRemoval = true)  
    private List<Review> reviews = new ArrayList<>();  
  
    private boolean in_stock = true;  
}  
package com.zosh.model;
```

```
import com.fasterxml.jackson.annotation.JsonIgnore;
import jakarta.persistence.*;
import lombok.AllArgsConstructor;
import lombok.Getter;
import lombok.NoArgsConstructor;
import lombok.Setter;

import java.time.LocalDateTime;
import java.util.List;

@Getter
@Setter
@NoArgsConstructor
@AllArgsConstructor
@Entity
public class Review {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    @Column(nullable = false)
    private String reviewText;

    @Column(nullable = false)
    private double rating;

    @ElementCollection
    private List<String> productImages;

    @JsonIgnore
    @ManyToOne
    @JoinColumn(name = "product_id", nullable = false)
    private Product product;

    @ManyToOne
    @JoinColumn(name = "user_id", nullable = false)
    private User user;

    @Column(nullable = false)
    private LocalDateTime createdAt=LocalDateTime.now();

}

package com.zosh.model;

import com.zosh.domain.AccountStatus;
import com.zosh.domain.USER_ROLE;
import jakarta.persistence.*;
import lombok.AllArgsConstructor;
```

```
import lombok.Data;
import lombok.NoArgsConstructor;

@Entity
@Data
@NoArgsConstructor
@AllArgsConstructor
public class Seller {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    private String sellerName;

    private String mobile;

    @Column(unique = true, nullable = false)
    private String email;
    private String password;

    @Embedded
    private BusinessDetails businessDetails = new BusinessDetails();

    @Embedded
    private BankDetails bankDetails = new BankDetails();

    @OneToOne(cascade = CascadeType.ALL)
    private Address pickupAddress=new Address();

    private String GSTIN;

    private USER_ROLE role;

    private boolean isEmailVerified=false;

    private AccountStatus accountStatus = AccountStatus.PENDING_VERIFICATION;

}

package com.zosh.model;

import jakarta.persistence.*;
import lombok.AllArgsConstructor;
import lombok.Getter;
import lombok.NoArgsConstructor;
import lombok.Setter;

import java.math.BigDecimal;
```

```
import java.time.LocalDate;

@Entity
@NoArgsConstructor
@AllArgsConstructor
@Getter
@Setter
public class SellerReport {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    @OneToOne
    private Seller seller;

    private Long totalEarnings = 0L;

    private Long totalSales= 0L;

    private Long totalRefunds= 0L;

    private Long totalTax = 0L;

    private Long netEarnings = 0L;

    private Integer totalOrders= 0;

    private Integer canceledOrders=0;

    private Integer totalTransactions=0;

}

package com.zosh.model;

import jakarta.persistence.*;
import lombok.AllArgsConstructor;
import lombok.Getter;
import lombok.NoArgsConstructor;
import lombok.Setter;

import java.time.LocalDateTime;
import java.time.LocalTime;

@NoArgsConstructor
@AllArgsConstructor
@Getter
@Setter
```

```
@Entity
public class Transaction {

    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private Long id;

    @ManyToOne
    private User customer;

    @OneToOne
    private Order order;

    @ManyToOne
    private Seller seller;

    private LocalDateTime date= LocalDateTime.now();
}

package com.zosh.model;

import com.fasterxml.jackson.annotation.JsonIgnore;
import com.fasterxml.jackson.annotation.JsonProperty;
import com.zosh.domain.USER_ROLE;
import jakarta.persistence.*;
import lombok.Data;

import java.util.HashSet;
import java.util.List;
import java.util.Set;

@Entity
@Data
public class User {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    @Column(nullable = false)
    @JsonProperty(access = JsonProperty.Access.WRITE_ONLY)
    private String password;

    @Column(nullable = false)
    private String email;

    @Column(nullable = false)
    private String fullName;

    private String mobile;
```

```
private USER_ROLE role;

@OneToMany
private Set<Address> addresses=new HashSet<>();

@JsonIgnore
@ManyToMany
@JoinTable(
    name = "user_coupons",
    inverseJoinColumns = @JoinColumn(name = "coupon_id")
)
private Set<Coupon> usedCoupons=new HashSet<>();

}

package com.zosh.model;

import jakarta.persistence.*;
import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;

@Entity
@NoArgsConstructor
@AllArgsConstructor
@Data
public class VerificationCode {

    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private Long id;

    private String otp;

    private String email;

    @OneToOne
    private User user;

    @OneToOne
    private Seller seller;

}

package com.zosh.model;

import jakarta.persistence.*;
import lombok.AllArgsConstructor;
import lombok.Getter;
import lombok.NoArgsConstructor;
import lombok.Setter;
```

```
import java.util.HashSet;
import java.util.Set;

@Entity
@Getters
@Setters
@AllArgsConstructor
@NoArgsConstructor
public class Wishlist {

    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private Long id;

    @OneToOne
    private User user;

    @ManyToMany
    private Set<Product> products=new HashSet<>();
}

package com.zosh.repository;

import org.springframework.data.jpa.repository.JpaRepository;

import com.zosh.model.Address;

public interface AddressRepository extends JpaRepository<Address, Long> {

}

package com.zosh.repository;

import com.zosh.model.Cart;
import com.zosh.model.Product;
import com.zosh.model.User;
import org.springframework.data.jpa.repository.JpaRepository;

import com.zosh.model.CartItem;

public interface CartItemRepository extends JpaRepository<CartItem, Long> {

    CartItem findByCartAndProductAndSize(Cart cart, Product product, String
size);

}

package com.zosh.repository;
```

```
import java.util.Optional;

import org.springframework.data.jpa.repository.JpaRepository;

import com.zosh.model.Cart;

public interface CartRepository extends JpaRepository<Cart, Long> {

    Cart findByUserId(Long userId);
}

package com.zosh.repository;

import java.util.List;

import org.springframework.data.jpa.repository.JpaRepository;

import com.zosh.model.Category;

public interface CategoryRepository extends JpaRepository<Category, Long> {

    Category findByCategoryId(String categoryId);

    List<Category>findByLevel(Integer level);

}

package com.zosh.repository;

import com.zosh.model.Coupon;
import org.springframework.data.jpa.repository.JpaRepository;

public interface CouponRepository extends JpaRepository<Coupon, Long> {
    Coupon findByCode(String couponCode);
}

package com.zosh.repository;

import com.zosh.model.Deal;
import org.springframework.data.jpa.repository.JpaRepository;

public interface DealRepository extends JpaRepository<Deal, Long> {

}

package com.zosh.repository;

import com.zosh.model.HomeCategory;
import org.springframework.data.jpa.repository.JpaRepository;

public interface HomeCategoryRepository extends
JpaRepository<HomeCategory, Long> {
}
```

```
package com.zosh.repository;

import java.util.List;

import org.springframework.data.jpa.repository.JpaRepository;

import com.zosh.model.Notification;

public interface NotificationRepository extends JpaRepository<Notification,
Long> {

}

package com.zosh.repository;

import org.springframework.data.jpa.repository.JpaRepository;

import com.zosh.model.OrderItem;

public interface OrderItemRepository extends JpaRepository<OrderItem, Long> {

}

package com.zosh.repository;

import java.time.LocalDateTime;
import java.util.Date;
import java.util.List;
import java.util.Optional;

import com.zosh.domain.OrderStatus;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.data.jpa.repository.Query;
import org.springframework.data.repository.query.Param;

import com.zosh.model.Order;
import com.zosh.model.User;

public interface OrderRepository extends JpaRepository<Order, Long> {

    List<Order> findByUserId(Long userId);
    List<Order> findBySellerIdOrderByOrderDateDesc(Long sellerId);
    List<Order> findBySellerIdAndOrderDateBetween(Long sellerId, LocalDateTime
startDate, LocalDateTime endDate);

}
package com.zosh.repository;

import org.springframework.data.jpa.repository.JpaRepository;
```

```
import com.zosh.model.PasswordResetToken;

public interface PasswordResetTokenRepository extends
JpaRepository<PasswordResetToken, Integer> {
    PasswordResetToken findByToken(String token);
}
package com.zosh.repository;

import com.zosh.model.PaymentOrder;
import org.springframework.data.jpa.repository.JpaRepository;

public interface PaymentOrderRepository extends
JpaRepository<PaymentOrder, Long> {

    PaymentOrder findByPaymentLinkId(String paymentId);
}
package com.zosh.repository;

import com.zosh.domain.PayoutsStatus;
import com.zosh.model.Payouts;
import org.springframework.data.jpa.repository.JpaRepository;

import java.util.List;

public interface PayoutsRepository extends JpaRepository<Payouts, Long> {

    List<Payouts> findPayoutsBySellerId(Long sellerId);
    List<Payouts> findAllByStatus(PayoutsStatus status);
}
package com.zosh.repository;

import com.zosh.model.Category;
import com.zosh.model.Product;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.data.jpa.repository.JpaSpecificationExecutor;
import org.springframework.data.jpa.repository.Query;
import org.springframework.data.repository.query.Param;

import java.util.List;

public interface ProductRepository extends JpaRepository<Product, Long> ,
JpaSpecificationExecutor<Product> {

    List<Product> findBySellerId(Long shopId);

    @Query("SELECT p FROM Product p WHERE (:query IS NULL OR LOWER(p.title) " +
           "LIKE LOWER(CONCAT('%', :query, '%')))) " +

```

```

        "OR (:query IS NULL OR LOWER(p.category.name) " +
        "LIKE LOWER(CONCAT('%', :query, '%')))" +
        "OR (:query IS NULL OR LOWER(p.category.categoryId) " +
        "LIKE LOWER(CONCAT('%', :query, '%')))"
    )
    List<Product> searchProduct(@Param("query") String query);

}

package com.zosh.repository;

import com.zosh.model.Product;
import com.zosh.model.Review;
import com.zosh.model.User;
import org.springframework.data.jpa.repository.JpaRepository;

import java.util.List;

public interface ReviewRepository extends JpaRepository<Review, Long> {
    List<Review> findReviewsByUserId(Long userId);
    List<Review> findReviewsByProductId(Long productId);
}
package com.zosh.repository;

import com.zosh.model.SellerReport;
import org.springframework.data.jpa.repository.JpaRepository;

public interface SellerReportRepository extends
JpaRepository<SellerReport, Long> {
    SellerReport findBySellerId(Long sellerId);
}
package com.zosh.repository;

import com.zosh.domain.AccountStatus;
import com.zosh.model.Seller;
import org.springframework.data.jpa.repository.JpaRepository;

import java.util.List;

public interface SellerRepository extends JpaRepository<Seller, Long> {

    Seller findByEmail(String email);
    List<Seller> findByAccountStatus(AccountStatus status);
}
package com.zosh.repository;

import com.zosh.model.Transaction;
import org.springframework.data.jpa.repository.JpaRepository;

```

```
import java.util.List;

public interface TransactionRepository extends JpaRepository<Transaction, Integer> {

    List<Transaction> findBySellerId(Long sellerId);
}
package com.zosh.repository;

import java.util.List;
import java.util.Optional;

import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.data.jpa.repository.Query;

import com.zosh.model.User;

public interface UserRepository extends JpaRepository<User, Long> {

    public User findByEmail(String username);

}
package com.zosh.repository;

import com.zosh.model.VerificationCode;
import org.springframework.data.jpa.repository.JpaRepository;

public interface VerificationCodeRepository extends JpaRepository<VerificationCode, Long> {
    VerificationCode findByEmail(String email);
    VerificationCode findByOtp(String otp);
}
package com.zosh.repository;

import com.zosh.model.Wishlist;
import org.springframework.data.jpa.repository.JpaRepository;

public interface WishlistRepository extends JpaRepository<Wishlist, Long> {
    Wishlist findByUserId(Long userId);
}
package com.zosh.request;

import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;

@Data
@AllArgsConstructor
@NoArgsConstructor
```

```
@NoArgsConstructor
@AllArgsConstructor
public class AddItemRequest {

    private Long productId;
    private String size;
    private int quantity;
    private Integer price;

}

package com.zosh.request;

import lombok.Data;

@Data
public class CreateCategoryRequest {

    private String parentCategoryId;
    private int level;
    private String name;
    private String categoryId;
}
package com.zosh.request;

import lombok.Data;

@Data
public class CreateHomeCategories {
    private String categoryId;
    private String name;
    private String image;
}
package com.zosh.request;

import java.util.ArrayList;
import java.util.HashSet;
import java.util.List;
import java.util.Set;

import com.zosh.model.Category;
import jakarta.persistence.Column;
import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;

@Data
```

```
@NoArgsConstructor
@AllArgsConstructor
public class CreateProductRequest {

    private String title;

    @Column(length = 2000)
    private String description;

    private int mrpPrice;

    private int sellingPrice;

    private String brand;

    private String color;

    @Column(length = 5000)
    private List<String> images;

    private String category;
    private String category2;
    private String category3;

    private String sizes;

}

package com.zosh.request;

import lombok.Data;

import java.util.List;

@Data
public class CreateReviewRequest {
    private String reviewText;
    private double reviewRating;
    private List<String> productImages;
}
package com.zosh.request;

public class DeleteProductRequest {

//    private Long

}

package com.zosh.request;

import lombok.AllArgsConstructor;
```

```
import lombok.Data;
import lombok.NoArgsConstructor;

@Data
@NoArgsConstructor
@AllArgsConstructor
public class LoginRequest {

    private String email;
    private String password;
    private String otp;

}

package com.zosh.request;

import lombok.Data;
import lombok.Getter;

@Data
public class Prompt {
    private String prompt;
}
package com.zosh.request;

import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;

@Data
@NoArgsConstructor
@AllArgsConstructor
public class RatingRequest {

    private Long productId;
    private double rating;

}

package com.zosh.request;

import lombok.Data;

@Data
public class ResetPasswordRequest {

    private String password;
    private String token;
```

```
}

package com.zosh.request;

import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;

@Data
@NoArgsConstructor
@AllArgsConstructor
public class ReviewRequest {

    private Long productId;
    private String review;

}

package com.zosh.request;

import lombok.Data;
import lombok.NoArgsConstructor;

@Data
@NoArgsConstructor
public class SignupRequest {
    private String fullName;
    private String email;
    private String otp;
}
package com.zosh.response;

import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;

@Data
@NoArgsConstructor
@AllArgsConstructor
public class ApiResponse {

    private String message;
    private boolean status;

}
```

```
package com.zosh.response;

import com.zosh.domain.USER_ROLE;
import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;

@Data
@NoArgsConstructor
@AllArgsConstructor
public class AuthResponse {

    private String jwt;

    private boolean status;

    private String message;

    private USER_ROLE role;
}

package com.zosh.response;

import com.zosh.dto.OrderHistory;
import com.zosh.model.Cart;
import com.zosh.model.Order;
import com.zosh.model.Product;
import lombok.*;

@Data
@NoArgsConstructor
@AllArgsConstructor
public class FunctionResponse {
    private String functionName;
    private Cart userCart;
    private OrderHistory orderHistory;
    private Product product;
}

package com.zosh.response;

import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;

@Data
@NoArgsConstructor
@AllArgsConstructor
public class PaymentLinkResponse {

    private String payment_link_url;
```

```
    private String payment_link_id;

}

package com.zosh.service;

import com.zosh.exception.SellerException;
import com.zosh.exception.UserException;
import com.zosh.request.LoginRequest;
import com.zosh.request.ResetPasswordRequest;
import com.zosh.request.SignupRequest;
import com.zosh.response.ApiResponse;
import com.zosh.response.AuthResponse;
import jakarta.mail.MessagingException;

public interface AuthService {

    void sentLoginOtp(String email) throws UserException, MessagingException;
    String createUser(SignupRequest req) throws SellerException;
    AuthResponse signin(LoginRequest req) throws SellerException;

}
package com.zosh.service;

import com.zosh.exception.CartItemException;
import com.zosh.exception.UserException;
import com.zosh.model.Cart;
import com.zosh.model.CartItem;
import com.zosh.model.Product;

public interface CartItemService {

    public CartItem updateCartItem(Long userId, Long id, CartItem cartItem)
throws CartItemException, UserException;

    public void removeCartItem(Long userId, Long cartItemId) throws
CartItemException, UserException;

    public CartItem findCartItemById(Long cartItemId) throws CartItemException;

}
package com.zosh.service;

import com.zosh.exception.ProductException;
import com.zosh.model.Cart;
import com.zosh.model.CartItem;
import com.zosh.model.Product;
import com.zosh.model.User;
```

```
import com.zosh.request.AddItemRequest;

public interface CartService {

    public CartItem addCartItem(User user,
                                Product product,
                                String size,
                                int quantity) throws ProductException;

    public Cart findUserCart(User user);

}

package com.zosh.service;

import com.zosh.model.Cart;
import com.zosh.model.Coupon;
import com.zosh.model.User;

import java.util.List;
import java.util.Optional;

public interface CouponService {
    Cart applyCoupon(String code, double orderValue, User user) throws
Exception;
    Cart removeCoupon(String code, User user) throws Exception;
    Coupon createCoupon(Coupon coupon);
    void deleteCoupon(Long couponId);
    List<Coupon> getAllCoupons();

    Coupon getCouponById(Long couponId);
}
package com.zosh.service;

import com.zosh.domain.USER_ROLE;
import com.zosh.model.User;
import com.zosh.repository.UserRepository;
import lombok.RequiredArgsConstructor;
import org.springframework.boot.CommandLineRunner;
import org.springframework.security.crypto.password.PasswordEncoder;
import org.springframework.stereotype.Component;

@Component
@RequiredArgsConstructor
public class DataInitializationComponent implements CommandLineRunner {

    private final UserRepository userRepository;
    private final PasswordEncoder passwordEncoder;
```

```

@Override
public void run(String... args) {
    initializeAdminUser();
}

private void initializeAdminUser() {
    String adminUsername = "ss2727303@gmail.com";

    if (userRepository.findByEmail(adminUsername)==null) {
        User adminUser = new User();

        adminUser.setPassword(passwordEncoder.encode("sajidsai"));
        adminUser.setFullName("Sajid");
        adminUser.setEmail(adminUsername);
        adminUser.setRole(USER_ROLE.ROLE_ADMIN);

        User admin=userRepository.save(adminUser);
    }
}

package com.zosh.service;

import com.zosh.model.Deal;

import java.util.List;

public interface DealService {
    Deal createDeal(Deal deal);
//    List<Deal> createDeals(List<Deal> deals);
    List<Deal> getDeals();
    Deal updateDeal(Deal deal,Long id) throws Exception;
    void deleteDeal(Long id) throws Exception;
}

package com.zosh.service;

import jakarta.mail.MessagingException;
import jakarta.mail.internet.MimeMessage;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.mail.MailException;
import org.springframework.mail.MailSendException;
import org.springframework.mail.javamail.JavaMailSender;
import org.springframework.mail.javamail.MimeMessageHelper;
import org.springframework.stereotype.Service;

@Service

```

```
public class EmailService {

    @Autowired
    private JavaMailSender javaMailSender;

    public void sendVerificationOtpEmail(String userEmail, String otp, String subject, String text) throws MessagingException, MailSendException {

        try {
            MimeMessage mimeMessage = javaMailSender.createMimeMessage();
            MimeMessageHelper helper = new MimeMessageHelper(mimeMessage,
"utf-8");

            helper.setSubject(subject);
            helper.setText(text+otp, true);
            helper.setTo(userEmail);
            javaMailSender.send(mimeMessage);
        } catch (MailException e) {
            throw new MailSendException("Failed to send email");
        }
    }
}

package com.zosh.service;

import com.zosh.model.HomeCategory;
import java.util.List;

public interface HomeCategoryService {
    HomeCategory createCategory(HomeCategory categories);
    List<HomeCategory> createCategories(List<HomeCategory> categories);
    List<HomeCategory> getAllCategories();
    HomeCategory updateCategory(HomeCategory categories, Long id) throws Exception;
}
package com.zosh.service;

import com.zosh.model.Home;
import com.zosh.model.HomeCategory;

import java.util.List;

public interface HomeService {

    Home creatHomePageData(List<HomeCategory> categories);
```

```
}

package com.zosh.service;

import com.zosh.exception.OrderException;
import com.zosh.model.OrderItem;
import com.zosh.model.Product;

public interface OrderItemService {

    OrderItem getOrderItemById(Long id) throws Exception;

}

package com.zosh.service;

import com.zosh.domain.OrderStatus;
import com.zosh.exception.OrderException;
import com.zosh.model.*;
import com.zosh.model.Order;

import java.util.List;
import java.util.Set;

public interface OrderService {

    public Set<Order> createOrder(User user, Address shippingAddress, Cart cart);

    public Order findOrderById(Long orderId) throws OrderException;

    public List<Order> usersOrderHistory(Long userId);

    public List<Order>getShopsOrders(Long sellerId);

    public Order updateOrderStatus(Long orderId,
                                  OrderStatus orderStatus)
        throws OrderException;

    public void deleteOrder(Long orderId) throws OrderException;

    Order cancelOrder(Long orderId,User user) throws OrderException;

}

package com.zosh.service;

import com.razorpay.PaymentLink;
import com.razorpay.RazorpayException;
```

```
import com.stripe.exception.StripeException;
import com.zosh.domain.PaymentMethod;
import com.zosh.model.Order;
import com.zosh.model.PaymentOrder;
import com.zosh.model.User;
import com.zosh.response.PaymentLinkResponse;

import java.util.List;
import java.util.Set;

public interface PaymentService {

    PaymentOrder createOrder(User user,
                            Set<Order> orders);

    PaymentOrder getPaymentOrderById(Long id) throws Exception;

    PaymentOrder getPaymentOrderByPaymentId(String paymentId) throws Exception;

    Boolean ProceedPaymentOrder (PaymentOrder paymentOrder,
                                String paymentId, String paymentLinkId) throws
RazorpayException;

    PaymentLink createRazorpayPaymentLink(User user,
                                         Long Amount,
                                         Long orderId) throws
RazorpayException;

    String createStripePaymentLink(User user, Long Amount,
                                         Long orderId) throws
StripeException;
}

package com.zosh.service;

import com.zosh.exception.ProductException;
import com.zosh.model.Product;
import com.zosh.model.Seller;
import com.zosh.request.CreateProductRequest;
import org.springframework.data.domain.Page;

import java.util.List;

public interface ProductService {
    public Product createProduct(CreateProductRequest req,
                                Seller seller) throws ProductException;

    public void deleteProduct(Long productId) throws ProductException;
```

```
    public Product updateProduct(Long productId, Product product) throws
ProductException;
    public Product updateProductStock(Long productId) throws ProductException;

    public Product findProductById(Long id) throws ProductException;

    public List<Product> searchProduct(String query);

    public Page<Product> getAllProduct(String category,
                                         String brand,
                                         String colors,
                                         String sizes,
                                         Integer minPrice,
                                         Integer maxPrice,
                                         Integer minDiscount,
                                         String sort,
                                         String stock,
                                         Integer pageNumber);

    public List<Product> recentlyAddedProduct();
    List<Product> getProductsBySellerId(Long sellerId);
}

package com.zosh.service;

import com.zosh.dto.RevenueChart;

import java.util.List;
import java.util.Map;

public interface RevenueService {
    List<RevenueChart> getDailyRevenueForChart(int days, Long sellerId);
    List<RevenueChart> getMonthlyRevenueForChart(int months, Long sellerId);
    List<RevenueChart> getYearlyRevenueForChart(int years, Long sellerId);
    List<RevenueChart> getHourlyRevenueForChart(Long sellerId);
    List<RevenueChart> getRevenueChartByType(String type, Long sellerId);
}
package com.zosh.service;

import com.zosh.exception.ReviewNotFoundException;
import com.zosh.model.Product;
import com.zosh.model.Review;
import com.zosh.model.User;
import com.zosh.request.CreateReviewRequest;

import javax.naming.AuthenticationException;
import java.util.List;
```

```
public interface ReviewService {

    Review createReview(CreateReviewRequest req,
                       User user,
                       Product product);

    List<Review> getReviewsByProductId(Long productId);

    Review updateReview(Long reviewId,
                        String reviewText,
                        double rating,
                        Long userId) throws ReviewNotFoundException,
AuthenticationException;

    void deleteReview(Long reviewId, Long userId) throws
ReviewNotFoundException, AuthenticationException;

}

package com.zosh.service;

import com.zosh.model.Seller;
import com.zosh.model.SellerReport;
import java.util.List;
import java.util.Optional;

public interface SellerReportService {
    SellerReport getSellerReport(Seller seller);
    SellerReport updateSellerReport( SellerReport sellerReport);

}
package com.zosh.service;

import com.zosh.domain.AccountStatus;
import com.zosh.exception.SellerException;
import com.zosh.model.Seller;
import java.util.List;
import java.util.Optional;

public interface SellerService {
    Seller getSellerProfile(String jwt) throws SellerException;
    Seller createSeller(Seller seller) throws SellerException;
    Seller getSellerById(Long id) throws SellerException;
    Seller getSellerByEmail(String email) throws SellerException;
    List<Seller> getAllSellers(AccountStatus status);
    Seller updateSeller(Long id, Seller seller) throws SellerException;
    void deleteSeller(Long id) throws SellerException;
    Seller verifyEmail(String email, String otp) throws SellerException;
```

```
    Seller updateSellerAccountStatus(Long sellerId, AccountStatus status) throws
SellerException;
}
package com.zosh.service;

import com.zosh.model.Order;
import com.zosh.model.Seller;
import com.zosh.model.Transaction;
import com.zosh.model.User;

import java.util.List;

public interface TransactionService {

    Transaction createTransaction(Order order);
    List<Transaction> getTransactionBySeller(Seller seller);
    List<Transaction> getAllTransactions();
}
package com.zosh.service;

import java.util.List;

import com.zosh.exception.UserException;
import com.zosh.model.User;

public interface UserService {

    public User findUserProfileByJwt(String jwt) throws UserException;

    public User findUserByEmail(String email) throws UserException;

}
package com.zosh.service;

import com.zosh.model.VerificationCode;

public interface VerificationService {

    VerificationCode createVerificationCode(String otp, String email);
}
package com.zosh.service;

import com.zosh.exception.WishlistNotFoundException;
import com.zosh.model.Product;
import com.zosh.model.User;
import com.zosh.model.Wishlist;
```

```
import java.util.Optional;

public interface WishlistService {

    Wishlist createWishlist(User user);

    Wishlist getWishlistByUserId(User user);

    Wishlist addProductToWishlist(User user, Product product) throws
WishlistNotFoundException;

}

package com.zosh.service.impl;

import com.zosh.config.JwtProvider;
import com.zosh.domain.USER_ROLE;
import com.zosh.exception.SellerException;
import com.zosh.exception.UserException;
import com.zosh.model.Cart;
import com.zosh.model.PasswordResetToken;
import com.zosh.model.User;
import com.zosh.model.VerificationCode;
import com.zosh.repository.CartRepository;
import com.zosh.repository.UserRepository;
import com.zosh.repository.VerificationCodeRepository;
import com.zosh.request.LoginRequest;
import com.zosh.request.ResetPasswordRequest;
import com.zosh.request.SignupRequest;
import com.zosh.response.ApiResponse;
import com.zosh.response.AuthResponse;
import com.zosh.service.AuthService;
import com.zosh.service.EmailService;
import com.zosh.service.UserService;
import com.zosh.utils.OtpUtils;
import jakarta.mail.MessagingException;
import lombok.RequiredArgsConstructor;
import org.springframework.security.authentication.BadCredentialsException;
import
org.springframework.security.authentication.UsernamePasswordAuthenticationToken
;
import org.springframework.security.core.Authentication;
import org.springframework.security.core.GrantedAuthority;
import org.springframework.security.core.authority.SimpleGrantedAuthority;
import org.springframework.security.core.context.SecurityContextHolder;
import org.springframework.security.core.userdetails.UserDetails;
import org.springframework.security.crypto.password.PasswordEncoder;
import org.springframework.stereotype.Service;
```

```
import java.util.ArrayList;
import java.util.Collection;
import java.util.List;

@Service
@RequiredArgsConstructor
public class AuthServiceImpl implements AuthService {

    private final UserService userService;

    private final VerificationCodeRepository verificationCodeRepository;
    private final EmailService emailService;
    private final PasswordEncoder passwordEncoder;
    private final UserRepository userRepository;

    private final JwtProvider jwtProvider;
    private final CustomeUserServiceImplementation customUserDetails;
    private final CartRepository cartRepository;

    @Override
    public void sentLoginOtp(String email) throws UserException,
    MessagingException {

        String SIGNING_PREFIX = "signing_";

        if (email.startsWith(SIGNING_PREFIX)) {
            email = email.substring(SIGNING_PREFIX.length());
            userService.findUserByEmail(email);
        }

        VerificationCode isExist = verificationCodeRepository
            .findByEmail(email);

        if (isExist != null) {
            verificationCodeRepository.delete(isExist);
        }

        String otp = OtpUtils.generateOTP();

        VerificationCode verificationCode = new VerificationCode();
        verificationCode.setOtp(otp);
        verificationCode.setEmail(email);
        verificationCodeRepository.save(verificationCode);

        String subject = "Zosh Bazaar Login/Signup Otp";
        String text = "your login otp is - ";
    }
}
```

```

        emailService.sendVerificationOtpEmail(email, otp, subject, text);
    }

    @Override
    public String createUser(SignupRequest req) throws SellerException {

        String email = req.getEmail();

        String fullName = req.getFullName();

        String otp = req.getOtp();

        VerificationCode verificationCode =
        verificationCodeRepository.findByEmail(email);

        if (verificationCode == null || !verificationCode.getOtp().equals(otp))
        {
            throw new SellerException("wrong otp...");
        }

        User user = userRepository.findByEmail(email);

        if (user == null) {

            User createdUser = new User();
            createdUser.setEmail(email);
            createdUser.setFullName(fullName);
            createdUser.setRole(USER_ROLE.ROLE_CUSTOMER);
            createdUser.setMobile("9083476123");
            createdUser.setPassword(passwordEncoder.encode(otp));

            System.out.println(createdUser);

            user = userRepository.save(createdUser);

            Cart cart = new Cart();
            cart.setUser(user);
            cartRepository.save(cart);
        }

        List<GrantedAuthority> authorities = new ArrayList<>();

        authorities.add(new SimpleGrantedAuthority(
            USER_ROLE.ROLE_CUSTOMER.toString()));

        Authentication authentication = new UsernamePasswordAuthenticationToken(
            email, null, authorities);
    }
}

```

```
        SecurityContextHolder.getContext().setAuthentication(authentication);

        return jwtProvider.generateToken(authentication);
    }

    @Override
    public AuthResponse signin(LoginRequest req) throws SellerException {

        String username = req.getEmail();
        String otp = req.getOtp();

        System.out.println(username + " ----- " + otp);

        Authentication authentication = authenticate(username, otp);
        SecurityContextHolder.getContext().setAuthentication(authentication);

        String token = jwtProvider.generateToken(authentication);
        AuthResponse authResponse = new AuthResponse();

        authResponse.setMessage("Login Success");
        authResponse.setJwt(token);
        Collection<? extends GrantedAuthority> authorities =
authentication.getAuthorities();

        String roleName = authorities.isEmpty() ? null :
authorities.iterator().next().getAuthority();

        authResponse.setRole(USER_ROLE.valueOf(roleName));

        return authResponse;
    }

    private Authentication authenticate(String username, String otp) throws
SellerException {
    UserDetails userDetailsService =
customUserDetailsService.loadUserByUsername(username);

    System.out.println("sign in userDetailsService - " + userDetailsService);

    if (userDetailsService == null) {
        System.out.println("sign in userDetailsService - null ");
        throw new BadCredentialsException("Invalid username or password");
    }
}
```

```

        VerificationCode verificationCode =
verificationCodeRepository.findByEmail(username);

        if (verificationCode == null || !verificationCode.getOtp().equals(otp) )
{
            throw new SellerException("wrong otp...");
}
        return new UsernamePasswordAuthenticationToken(userDetails, null,
userDetails.getAuthorities());
    }
}
package com.zosh.service.impl;

import com.zosh.exception.CartItemException;
import com.zosh.exception.UserException;

import com.zosh.model.Cart;
import com.zosh.model.CartItem;
import com.zosh.model.Product;
import com.zosh.model.User;
import com.zosh.repository.CartItemRepository;
import com.zosh.service.CartItemService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import java.util.Optional;

@Service
public class CartItemServiceImplementation implements CartItemService {

    private CartItemRepository cartItemRepository;

    @Autowired
    public CartItemServiceImplementation(CartItemRepository cartItemRepository)
{
    this.cartItemRepository=cartItemRepository;
}

    @Override
    public CartItem updateCartItem(Long userId,
                                  Long id, CartItem cartItem)
throws CartItemException, UserException {

    CartItem item=findCartItemById(id);
    User cartItemUser=item.getCart().getUser();
}

```

```

        if(cartItemUser.getId().equals(userId)) {

            item.setQuantity(cartItem.getQuantity());
            item.setMrpPrice(item.getQuantity()*item.getProduct().getMrpPrice());

            item.setSellingPrice(item.getQuantity()*item.getProduct().getSellingPrice());

            return cartItemRepository.save(item);

        }
        else {
            throw new CartItemException("You can't update another users
cart_item");
        }
    }

@Override
public void removeCartItem(Long userId,Long cartItemId)
    throws CartItemException,
    UserException {

    System.out.println("userId- "+userId+" cartItemId "+cartItemId);

    CartItem cartItem=findCartItemById(cartItemId);

    User cartItemUser=cartItem.getCart().getUser();

    if(cartItemUser.getId().equals(userId)) {
        cartItemRepository.deleteById(cartItem.getId());
    }
    else {
        throw new UserException("you can't remove another users item");
    }
}

@Override
public CartItem findCartItemById(Long cartItemId) throws CartItemException {
    Optional<CartItem> opt=cartItemRepository.findById(cartItemId);

    if(opt.isPresent()) {
        return opt.get();
    }
    throw new CartItemException("cartItem not found with id : "+cartItemId);
}

```

```
}
```

```
package com.zosh.service.impl;
```

```
import com.zosh.exception.ProductException;
```

```
import com.zosh.model.Cart;
```

```
import com.zosh.model.CartItem;
```

```
import com.zosh.model.Product;
```

```
import com.zosh.model.User;
```

```
import com.zosh.repository.CartItemRepository;
```

```
import com.zosh.repository.CartRepository;
```

```
import com.zosh.service.CartItemService;
```

```
import com.zosh.service.CartService;
```

```
import com.zosh.service.ProductService;
```

```
import lombok.RequiredArgsConstructor;
```

```
import org.springframework.stereotype.Service;
```

```
@Service
```

```
@RequiredArgsConstructor
```

```
public class CartServiceImplementation implements CartService {
```

```
    private final CartRepository cartRepository;
```

```
    private final CartItemRepository cartItemRepository;
```

```
    private final ProductService productService;
```

```
    public Cart findUserCart(User user) {
```

```
        Cart cart = cartRepository.findById(user.getId());
```

```
        // Create a new cart if one doesn't exist for the user
```

```
        if (cart == null) {
```

```
            cart = new Cart();
```

```
            cart.setUser(user);
```

```
            cart.setTotalMrpPrice(0);
```

```
            cart.setTotalSellingPrice(0);
```

```
            cart.setDiscount(0);
```

```
            cart.setTotalItem(0);
```

```
            cart.setCouponPrice(0);
```

```
            cart = cartRepository.save(cart);
```

```
        }
```

```
        // Only process cart items if they exist
```

```
        if (cart.getCartItems() != null && !cart.getCartItems().isEmpty()) {
```

```
            int totalPrice = 0;
```

```
            int totalDiscountedPrice = 0;
```

```
            int totalItem = 0;
```

```
            for (CartItem cartItem : cart.getCartItems()) {
```

```

        totalPrice += cartItem.getMrpPrice();
        totalDiscountedPrice += cartItem.getSellingPrice();
        totalItem += cartItem.getQuantity();
    }

    cart.setTotalMrpPrice(totalPrice);
    cart.setTotalItem(cart.getCartItems().size());
    cart.setTotalSellingPrice(totalDiscountedPrice -
cart.getCouponPrice());
    cart.setDiscount(calculateDiscountPercentage(totalPrice,
totalDiscountedPrice));
    cart.setTotalItem(totalItem);

    return cartRepository.save(cart);
}

return cart;
}

public static int calculateDiscountPercentage(double mrpPrice, double
sellingPrice) {
    if (mrpPrice <= 0) {
        return 0;
    }
    double discount = mrpPrice - sellingPrice;
    double discountPercentage = (discount / mrpPrice) * 100;
    return (int) discountPercentage;
}

@Override
public CartItem addCartItem(User user,
                            Product product,
                            String size,
                            int quantity
                           ) throws ProductException {
    Cart cart=findUserCart(user);

    CartItem isPresent=cartItemRepository.findByCartAndProductAndSize(
        cart, product, size);

    if(isPresent == null) {
        CartItem cartItem = new CartItem();
        cartItem.setProduct(product);

        cartItem.setQuantity(quantity);
        cartItem.setUserId(user.getId());

        int totalPrice=quantity*product.getSellingPrice();
        cartItem.setSellingPrice(totalPrice);
    }
}

```

```
        cartItem.setMrpPrice(quantity*product.getMrpPrice());
        cartItem.setSize(size);

        cart.getCartItems().add(cartItem);
        cartItem.setCart(cart);

        return cartItemRepository.save(cartItem);
    }

    return isPresent;
}

}

package com.zosh.service.impl;

import com.zosh.exception.CouponNotValidException;
import com.zosh.model.Cart;
import com.zosh.model.Coupon;
import com.zosh.model.User;
import com.zosh.repository.CartRepository;
import com.zosh.repository.CouponRepository;
import com.zosh.repository.UserRepository;
import com.zosh.service.CouponService;
import lombok.RequiredArgsConstructor;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.security.access.prepost.PreAuthorize;
import org.springframework.stereotype.Service;

import java.time.LocalDate;
import java.util.List;
import java.util.Optional;

@Service
@RequiredArgsConstructor
public class CouponServiceImpl implements CouponService {

    private final CouponRepository couponRepository;
    private final UserRepository userRepository;
    private final CartRepository cartRepository;

    @Override
    public Cart applyCoupon(String code,
                           double orderValue,
                           User user)
        throws Exception {
        Coupon coupon = couponRepository.findByCode(code);
        Cart cart = cartRepository.findByUserId(user.getId());
        cartItem.setCart(cart);
        cartItem.setMrpPrice(quantity*product.getMrpPrice());
        cartItem.setSize(size);

        cart.getCartItems().add(cartItem);
        cartItem.setCart(cart);

        return cartItemRepository.save(cartItem);
    }

    return isPresent;
}

}

package com.zosh.service.impl;

import com.zosh.exception.CouponNotValidException;
import com.zosh.model.Cart;
import com.zosh.model.Coupon;
import com.zosh.model.User;
import com.zosh.repository.CartRepository;
import com.zosh.repository.CouponRepository;
import com.zosh.repository.UserRepository;
import com.zosh.service.CouponService;
import lombok.RequiredArgsConstructor;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.security.access.prepost.PreAuthorize;
import org.springframework.stereotype.Service;

import java.time.LocalDate;
import java.util.List;
import java.util.Optional;

@Service
@RequiredArgsConstructor
public class CouponServiceImpl implements CouponService {

    private final CouponRepository couponRepository;
    private final UserRepository userRepository;
    private final CartRepository cartRepository;

    @Override
    public Cart applyCoupon(String code,
                           double orderValue,
                           User user)
        throws Exception {
        Coupon coupon = couponRepository.findByCode(code);
        Cart cart = cartRepository.findByUserId(user.getId());
        cartItem.setCart(cart);
        cartItem.setMrpPrice(quantity*product.getMrpPrice());
        cartItem.setSize(size);

        cart.getCartItems().add(cartItem);
        cartItem.setCart(cart);

        return cartItemRepository.save(cartItem);
    }

    return isPresent;
}

}
```

```

        if (coupon==null) {
            throw new CouponNotValidException("coupon not found");
        }
        if(user.getUsedCoupons().contains(coupon)) {
            throw new CouponNotValidException("coupon already used");
        }
        if(orderValue <= coupon.getMinimumOrderValue()){
            throw new CouponNotValidException("valid for minimum order value
"+coupon.getMinimumOrderValue());
        }

        if (
            coupon.isActive() &&
            LocalDate.now().isAfter(coupon.getValidityStartDate()) &&
            LocalDate.now().isBefore(coupon.getValidityEndDate())
        ) {

            user.getUsedCoupons().add(coupon);
            userRepository.save(user);

            double discountedPrice = Math.round((cart.getTotalSellingPrice() *
coupon.getDiscountPercentage()) / 100);
            cart.setTotalSellingPrice(cart.getTotalSellingPrice() -
discountedPrice);
            cart.setCouponCode(code);
            cart.setCouponPrice((int) discountedPrice);
            return cartRepository.save(cart);
        }
        return cart;
    }
    throw new CouponNotValidException("coupon not valid...");
}

@Override
public Cart removeCoupon(String code, User user) throws Exception {
    Coupon coupon = couponRepository.findByCode(code);

    if(coupon==null){
        throw new Exception("coupon not found...");
    }
    user.getUsedCoupons().remove(coupon);

    Cart cart = cartRepository.findById(user.getId());
//    double discountedPrice = (cart.getTotalSellingPrice() *
coupon.getDiscountPercentage()) / 100;
//    cart.setTotalSellingPrice(cart.getTotalSellingPrice() +
discountedPrice);
}

```

```
cart.setTotalSellingPrice(cart.getTotalSellingPrice() + cart.getCouponPrice());
    cart.setCouponCode(null);
    cart.setCouponPrice(0);
    return cartRepository.save(cart);

}

@Override
@PreAuthorize("hasRole('ADMIN')")
public Coupon createCoupon(Coupon coupon) {
    return couponRepository.save(coupon);
}

@Override
@PreAuthorize("hasRole('ADMIN')")
public void deleteCoupon(Long couponId) {
    couponRepository.deleteById(couponId);
}

@Override
@PreAuthorize("hasRole('ADMIN')")
public List<Coupon> getAllCoupons() {
    return couponRepository.findAll();
}

@Override
public Coupon getCouponById(Long couponId) {
//    return couponRepository.findById(couponId).orElseThrow(new
Exception("coupon not found"));
    return null;
}
}

package com.zosh.service.impl;

import java.util.ArrayList;
import java.util.List;

import com.zosh.domain.USER_ROLE;
import com.zosh.model.Seller;
import com.zosh.repository.SellerRepository;
import org.springframework.security.core.GrantedAuthority;
import org.springframework.security.core.authority.SimpleGrantedAuthority;
import org.springframework.security.core.userdetails.UserDetails;
import org.springframework.security.core.userdetails.UserDetailsService;
import org.springframework.security.core.userdetails.UsernameNotFoundException;
import org.springframework.stereotype.Service;

import com.zosh.model.User;
```

```
import com.zosh.repository.UserRepository;

@Service
public class CustomeUserServiceImplementation implements UserDetailsService {

    private final UserRepository userRepository;
    private final SellerRepository sellerRepository;
    private static final String SELLER_PREFIX = "seller_";

    public CustomeUserServiceImplementation(UserRepository userRepository,
SellerRepository sellerRepository) {
        this.userRepository = userRepository;
        this.sellerRepository = sellerRepository;
    }

    @Override
    public UserDetails loadUserByUsername(String username) throws
UsernameNotFoundException {
        if (username.startsWith(SELLER_PREFIX)) {
            // Remove prefix to get the actual username/email
            String actualUsername = username.substring(SELLER_PREFIX.length());
            Seller seller = sellerRepository.findByEmail(actualUsername);
            if (seller != null) {
                return buildUserDetails(seller.getEmail(), seller.getPassword(),
seller.getRole());
            }
        } else {
            User user = userRepository.findByEmail(username);
            if (user != null) {
                return buildUserDetails(user.getEmail(), user.getPassword(),
user.getRole());
            }
        }
        throw new UsernameNotFoundException("User or Seller not found with email
- " + username);
    }

    private UserDetails buildUserDetails(String email, String password,
USER_ROLE role) {
        if (role == null) role = USER_ROLE.ROLE_CUSTOMER;

        List<GrantedAuthority> authorities = new ArrayList<>();
        authorities.add(new SimpleGrantedAuthority(role.toString()));

        return new org.springframework.security.core.userdetails.User(email,
password, authorities);
    }
}
```

```
package com.zosh.service.impl;

import com.zosh.model.Deal;
import com.zosh.model.Home;
import com.zosh.model.HomeCategory;
import com.zosh.repository.DealRepository;
import com.zosh.repository.HomeCategoryRepository;
import com.zosh.service.DealService;
import lombok.RequiredArgsConstructor;
import org.springframework.stereotype.Service;

import java.util.List;

@Service
@RequiredArgsConstructor
public class DealServiceImpl implements DealService {
    private final DealRepository dealRepository;
    private final HomeCategoryRepository homeCategoryRepository;

    @Override
    public Deal createDeal(Deal deal) {
        HomeCategory category = homeCategoryRepository
            .findById(deal.getCategory().getId()).orElse(null);
        Deal newDeal = new Deal();
        newDeal.setCategory(category);
        newDeal.setDiscount(deal.getDiscount());
        return dealRepository.save(newDeal);
    }
    //
    //    @Override
    //    public List<Deal> createDeals(List<Deal> deals) {
    //        if(dealRepository.findAll().isEmpty()){
    //            return dealRepository.saveAll(deals);
    //        }
    //        else return dealRepository.findAll();
    //    }
    //

    @Override
    public List<Deal> getDeals() {
        return dealRepository.findAll();
    }

    @Override
    public Deal updateDeal(Deal deal, Long id) throws Exception {
        Deal existingDeal = dealRepository.findById(id).orElse(null);
```

```

    HomeCategory
category=homeCategoryRepository.findById(deal.getCategory().getId()).orElse(null);

    if(existingDeal!=null) {
        if(deal.getDiscount()!=null) {
            existingDeal.setDiscount(deal.getDiscount());
        }
        if(category!=null) {
            existingDeal.setCategory(category);
        }
        return dealRepository.save(existingDeal);
    }
    throw new Exception("Deal not found");
}

@Override
public void deleteDeal(Long id) throws Exception {
    Deal deal = dealRepository.findById(id).orElse(null);

    if (deal != null) {

        dealRepository.delete(deal);
    }
}

}

package com.zosh.service.impl;

import com.zosh.model.HomeCategory;
import com.zosh.repository.HomeCategoryRepository;
import com.zosh.service.HomeCategoryService;
import lombok.RequiredArgsConstructor;
import org.springframework.stereotype.Service;

import java.util.List;

@Service
@RequiredArgsConstructor
public class HomeCategoryServiceImpl implements HomeCategoryService {

    private final HomeCategoryRepository homeCategoryRepository;

    @Override
    public HomeCategory createCategory(HomeCategory categories) {

        return homeCategoryRepository.save(categories);
    }
}

```

```

    }

    @Override
    public List<HomeCategory> createCategories(List<HomeCategory> categories) {
        if (homeCategoryRepository.findAll().isEmpty()) {
            return homeCategoryRepository.saveAll(categories);
        }
        return homeCategoryRepository.findAll();
    }

    @Override
    public List<HomeCategory> getAllCategories() {
        return homeCategoryRepository.findAll();
    }

    @Override
    public HomeCategory updateCategory(HomeCategory category, Long id) throws
Exception {
        HomeCategory existingCategory = homeCategoryRepository.findById(id)
            .orElseThrow(() -> new Exception("Category not found"));
        if(category.getImage() !=null){
            existingCategory.setImage(category.getImage());
        }
        if(category.getCategoryId() !=null){
            existingCategory.setCategoryId(category.getCategoryId());
        }
        return homeCategoryRepository.save(existingCategory);
    }

}

package com.zosh.service.impl;

import com.zosh.domain.HomeCategorySection;
import com.zosh.model.Deal;
import com.zosh.model.Home;
import com.zosh.model.HomeCategory;
import com.zosh.repository.DealRepository;
import com.zosh.service.HomeService;
import lombok.RequiredArgsConstructor;
import org.springframework.stereotype.Service;

import java.util.ArrayList;
import java.util.List;
import java.util.stream.Collectors;

@RequiredArgsConstructor

```

```

@Service
public class HomeServiceImpl implements HomeService {

    private final DealRepository dealRepository;

    @Override
    public Home createHomePageData(List<HomeCategory> allCategories) {

        List<HomeCategory> gridCategories = allCategories.stream()
            .filter(category ->
                category.getSection() == HomeCategorySection.GRID)
            .collect(Collectors.toList());

        List<HomeCategory> shopByCategories = allCategories.stream()
            .filter(category ->
                category.getSection() ==
        HomeCategorySection.SHOP_BY_CATEGORIES)
            .collect(Collectors.toList());

        List<HomeCategory> electricCategories = allCategories.stream()
            .filter(category ->
                category.getSection() ==
        HomeCategorySection.ELECTRIC_CATEGORIES)
            .collect(Collectors.toList());

        List<HomeCategory> dealCategories = allCategories.stream()
            .filter(category -> category.getSection() ==
        HomeCategorySection.DEALS)
            .toList();

        List<Deal> createdDeals = new ArrayList<>();

        if (dealRepository.findAll().isEmpty()) {
            List<Deal> deals = allCategories.stream()
                .filter(category -> category.getSection() ==
        HomeCategorySection.DEALS)
                .map(category -> new Deal(null, 10, category)) // Assuming
a discount of 10 for each deal
                    .collect(Collectors.toList());
            createdDeals = dealRepository.saveAll(deals);
        } else createdDeals = dealRepository.findAll();

        Home home = new Home();
        home.setGrid(gridCategories);
        home.setShopByCategories(shopByCategories);
    }
}

```

```
        home.setElectricCategories(electricCategories);
        home.setDeals(createdDeals);
        home.setDealCategories(dealCategories);

        return home;
    }

}

package com.zosh.service.impl;

import com.zosh.exception.OrderException;
import com.zosh.model.OrderItem;
import com.zosh.repository.OrderItemRepository;
import com.zosh.service.OrderItemService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import java.util.Optional;

@Service
public class OrderItemServiceImpl implements OrderItemService {

    private final OrderItemRepository orderItemRepository;

    @Autowired
    public OrderItemServiceImpl(OrderItemRepository orderItemRepository) {
        this.orderItemRepository = orderItemRepository;
    }

    @Override
    public OrderItem getOrderItemById(Long id) throws Exception {

        System.out.println("----- "+id);
        Optional<OrderItem> orderItem = orderItemRepository.findById(id);
        if(orderItem.isPresent()){
            return orderItem.get();
        }
        throw new OrderException("Order item not found");
    }
}

package com.zosh.service.impl;

import com.zosh.domain.OrderStatus;
import com.zosh.domain.PaymentStatus;
import com.zosh.exception.OrderException;
import com.zosh.model.*;
```

```

import com.zosh.repository.AddressRepository;
import com.zosh.repository.OrderItemRepository;
import com.zosh.repository.OrderRepository;
import com.zosh.repository.UserRepository;

import com.zosh.service.CartService;
import com.zosh.service.OrderItemService;
import com.zosh.service.OrderService;
import lombok.RequiredArgsConstructor;
import org.springframework.stereotype.Service;

import java.util.*;
import java.util.stream.Collectors;

@Service
@RequiredArgsConstructor
public class OrderServiceImplementation implements OrderService {

    private final OrderRepository orderRepository;
    private final CartService cartService;
    private final AddressRepository addressRepository;
    private final UserRepository userRepository;
    private final OrderItemService orderItemService;
    private final OrderItemRepository orderItemRepository;

    @Override
    public Set<Order> createOrder(User user, Address shippAddress, Cart cart) {

        //    shippAddress.setUser(user);
        if (!user.getAddresses().contains(shippAddress)) {
            user.getAddresses().add(shippAddress);
        }

        Address address= addressRepository.save(shippAddress);

        Map<Long, List<CartItem>> itemsBySeller = cart.getCartItems().stream()
            .collect(Collectors.groupingBy(item ->
                item.getProduct().getSeller().getId()));

        Set<Order> orders=new HashSet<>();

        for(Map.Entry<Long, List<CartItem>> entry:itemsBySeller.entrySet()) {
            Long sellerId=entry.getKey();
            List<CartItem> cartItems=entry.getValue();

```

```

        int totalOrderPrice = cartItems.stream()
            .mapToInt(CartItem::getSellingPrice).sum();
        int
totalItem=cartItems.stream().mapToInt(CartItem::getQuantity).sum();

        Order createdOrder=new Order();
        createdOrder.setUser(user);
        createdOrder.setSellerId(sellerId);
        createdOrder.setTotalMrpPrice(totalOrderPrice);
        createdOrder.setTotalSellingPrice(totalOrderPrice);
        createdOrder.setTotalItem(totalItem);
        createdOrder.setShippingAddress(address);
        createdOrder.setOrderStatus(OrderStatus.PENDING);
        createdOrder.getPaymentDetails().setStatus(PaymentStatus.PENDING);

        Order savedOrder=orderRepository.save(createdOrder);
        orders.add(savedOrder);

List<OrderItem> orderItems=new ArrayList<>();

for(CartItem item: cartItems) {
    OrderItem orderItem=new OrderItem();

    orderItem.setOrder(savedOrder);
    orderItem.setMrpPrice(item.getMrpPrice());
    orderItem.setProduct(item.getProduct());
    orderItem.setQuantity(item.getQuantity());
    orderItem.setSize(item.getSize());
    orderItem.setUserId(item.getUserId());
    orderItem.setSellingPrice(item.getSellingPrice());

    savedOrder.getOrderItems().add(orderItem);

    OrderItem createdOrderItem=orderItemRepository.save(orderItem);

    orderItems.add(createdOrderItem);
}

return orders;
}

@Override
public Order findOrderById(Long orderId) throws OrderException {
    Optional<Order> opt=orderRepository.findById(orderId);
}

```

```

        if(opt.isPresent()) {
            return opt.get();
        }
        throw new OrderException("order not exist with id "+orderId);
    }

@Override
public List<Order> usersOrderHistory(Long userId) {

    return orderRepository.findByUserId(userId);
}

@Override
public List<Order> getShopsOrders(Long sellerId) {

    return orderRepository.findBySellerIdOrderByOrderDateDesc(sellerId);
}

@Override
public Order updateOrderStatus(Long orderId, OrderStatus orderStatus)
    throws OrderException {
    Order order=findOrderById(orderId);
    order.setOrderStatus(orderStatus);
    return orderRepository.save(order);
}

@Override
public void deleteOrder(Long orderId) throws OrderException {
    Order order = findOrderById(orderId);

    orderRepository.deleteById(orderId);

}

@Override
public Order cancelOrder(Long orderId, User user) throws OrderException {
    Order order=this.findOrderById(orderId);
    if(user.getId()!=order.getUser().getId()){
        throw new OrderException("you can't perform this action "+orderId);
    }
    order.setOrderStatus(OrderStatus.CANCELLED);

    return orderRepository.save(order);
}

}
package com.zosh.service.impl;

```

```
import com.razorpay.Payment;
import com.razorpay.PaymentLink;
import com.razorpay.RazorpayClient;
import com.razorpay.RazorpayException;
import com.stripe.Stripe;
import com.stripe.exception.StripeException;
import com.stripe.model.checkout.Session;
import com.stripe.param.checkout.SessionCreateParams;
import com.zosh.domain.PaymentOrderStatus;
import com.zosh.domain.PaymentStatus;
import com.zosh.model.Order;
import com.zosh.model.PaymentOrder;
import com.zosh.model.User;
import com.zosh.repository.CartRepository;
import com.zosh.repository.OrderRepository;
import com.zosh.repository.PaymentOrderRepository;
import com.zosh.service.PaymentService;
import lombok.RequiredArgsConstructor;
import org.json.JSONObject;
import org.springframework.beans.factory.annotation.Value;
import org.springframework.stereotype.Service;

import java.util.Optional;
import java.util.Set;

@Service
@RequiredArgsConstructor
public class PaymentServiceImpl implements PaymentService {

    @Value("${stripe.api.key}")
    private String stripeSecretKey;

    @Value("${razorpay.api.key}")
    private String apiKey;

    @Value("${razorpay.api.secret}")
    private String apiSecret;

    private final PaymentOrderRepository paymentOrderRepository;
    private final OrderRepository orderRepository;
    private final CartRepository cartRepository;

    @Override
    public PaymentOrder createOrder(User user, Set<Order> orders) {
```

```

        Long
amount=orders.stream().mapToLong(Order::getTotalSellingPrice).sum();
        int
couponPrice=cartRepository.findByUserId(user.getId()).getCouponPrice();

        PaymentOrder order=new PaymentOrder();
        order.setUser(user);
        order.setAmount(amount-couponPrice);
        order.setOrders(orders);

        return paymentOrderRepository.save(order);
    }

@Override
public PaymentOrder getPaymentOrderById(Long id) throws Exception {
    Optional<PaymentOrder>
optionalPaymentOrder=paymentOrderRepository.findById(id);
    if(optionalPaymentOrder.isEmpty()){
        throw new Exception("payment order not found with id "+id);
    }
    return optionalPaymentOrder.get();
}

@Override
public PaymentOrder getPaymentOrderByPaymentLinkId(String paymentLinkId) throws
Exception {
    PaymentOrder paymentOrder = paymentOrderRepository
        .findByPaymentLinkId(paymentLinkId);

    if(paymentOrder==null){
        throw new Exception("payment order not found with id
"+paymentLinkId);
    }
    return paymentOrder;
}

@Override
public Boolean ProceedPaymentOrder(PaymentOrder paymentOrder,
                                    String paymentId,
                                    String paymentLinkId) throws
RazorpayException {

    if(paymentOrder.getStatus().equals(PaymentOrderStatus.PENDING)) {

        RazorpayClient razorpay = new RazorpayClient(apiKey, apiSecret);
        Payment payment = razorpay.payments.fetch(paymentId);

        Integer amount = payment.get("amount");

```

```

        String status = payment.get("status");

        if(status.equals("captured")){
            // System.out.println("payment ===== captured");
            Set<Order> orders=paymentOrder.getOrders();
            for(Order order:orders){
                order.setPaymentStatus(PaymentStatus.COMPLETED);
                orderRepository.save(order);
            }
            paymentOrder.setStatus(PaymentOrderStatus.SUCCESS);
            paymentOrderRepository.save(paymentOrder);

            return true;
        }
        paymentOrder.setStatus(PaymentOrderStatus.FAILED);
        paymentOrderRepository.save(paymentOrder);
        return false;
    }

    return false;
}

@Override
public PaymentLink createRazorpayPaymentLink(User user,
                                               Long Amount,
                                               Long orderId
)
throws RazorpayException {

    Long amount = Amount * 100;

    try {
        // Instantiate a Razorpay client with your key ID and secret
        RazorpayClient razorpay = new RazorpayClient(apiKey, apiSecret);

        JSONObject paymentLinkRequest = new JSONObject();
        paymentLinkRequest.put("amount",amount);
        paymentLinkRequest.put("currency","INR");

        // Create a JSON object with the customer details
        JSONObject customer = new JSONObject();
        customer.put("name",user.getFullName());

        customer.put("email",user.getEmail());
        paymentLinkRequest.put("customer",customer);

        // Create a JSON object with the notification settings

```

```

        JSONObject notify = new JSONObject();
        notify.put("email",true);
        paymentLinkRequest.put("notify",notify);

        // Set the reminder settings
        paymentLinkRequest.put("reminder_enable",true);

        // Set the callback URL and method

paymentLinkRequest.put("callback_url","http://localhost:3000/payment-success/"+orderId);
        paymentLinkRequest.put("callback_method","get");

        PaymentLink payment =
razorpay.paymentLink.create(paymentLinkRequest);

        String paymentLinkUrl = payment.get("short_url");
        String paymentLinkId = payment.get("id");



System.out.println("payment ----- "+payment);

        return payment;

    } catch (RazorpayException e) {

        System.out.println("Error creating payment link: " +
e.getMessage());
        throw new RazorpayException(e.getMessage());
    }
}

@Override
public String createStripePaymentLink(User user, Long amount,Long orderId)
throws StripeException {
    Stripe.apiKey = stripeSecretKey;

    SessionCreateParams params = SessionCreateParams.builder()

.addPaymentMethodType(SessionCreateParams.PaymentMethodType.CARD)
        .setMode(SessionCreateParams.Mode.PAYMENT)
        .setSuccessUrl("http://localhost:3000/payment-success/"+orderId)
        .setCancelUrl("http://localhost:3000/payment/cancel")
        .addLineItem(SessionCreateParams.LineItem.builder()
            .setQuantity(1L)

.setPriceData(SessionCreateParams.LineItem.PriceData.builder()
            .setCurrency("usd")

```

```

        .setUnitAmount(amount*100)
        .setProductData(SessionCreateParams
            .LineItem
            .PriceData
            .ProductData
            .builder()
            .setName("Top up wallet")
            .build()
        )
        .build()
    )
    .build();
}

Session session = Session.create(params);

System.out.println("session _____ " + session);

// PaymentLinkResponse res = new PaymentLinkResponse();
// res.setPayment_link_url(session.getUrl());

return session.getUrl();
}
}
package com.zosh.service.impl;

import com.zosh.exception.ProductException;
import com.zosh.model.Category;
import com.zosh.model.Product;
import com.zosh.model.Seller;
import com.zosh.repository.CategoryRepository;
import com.zosh.repository.ProductRepository;
import com.zosh.request.CreateProductRequest;
import com.zosh.service.ProductService;
import jakarta.persistence.criteria.Join;
import jakarta.persistence.criteria.Predicate;
import lombok.RequiredArgsConstructor;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.data.domain.Page;
import org.springframework.data.domain.PageRequest;
import org.springframework.data.domain.Pageable;
import org.springframework.data.domain.Sort;
import org.springframework.data.jpa.domain.Specification;
import org.springframework.stereotype.Service;

import java.time.LocalDateTime;
import java.util.ArrayList;
import java.util.List;

@Service
@RequiredArgsConstructor

```

```
public class ProductServiceImpl implements ProductService {

    private final ProductRepository productRepository;

    private final CategoryRepository categoryRepository;

    @Override
    public Product createProduct(CreateProductRequest req,
                                 Seller seller
        ) throws ProductException {

        int discountPercentage = calculateDiscountPercentage(req.getMrpPrice(),
        req.getSellingPrice());

        Category category1=categoryRepository.findById(req.getCategory());
        if(category1==null){
            Category category=new Category();
            category.setCategoryId(req.getCategory());
            category.setLevel(1);
            category.setName(req.getCategory().replace("_", " "));
            category1=categoryRepository.save(category);
        }

        Category category2=categoryRepository.findById(req.getCategory2());
        if(category2==null){
            Category category=new Category();
            category.setCategoryId(req.getCategory2());
            category.setLevel(2);
            category.setParentCategory(category1);
            category.setName(req.getCategory2().replace("_", " "));
            category2=categoryRepository.save(category);
        }

        Category category3=categoryRepository.findById(req.getCategory3());
        if(category3==null){
            Category category=new Category();
            category.setCategoryId(req.getCategory3());
            category.setLevel(3);
            category.setParentCategory(category2);
            category.setName(req.getCategory3().replace("_", " "));
            category3=categoryRepository.save(category);
        }
    }
}
```

```

Product product=new Product();

product.setSeller(seller);
product.setCategory(category3);
product.setTitle(req.getTitle());
product.setColor(req.getColor());
product.setDescription(req.getDescription());
product.setDiscountPercent(discountPercentage);
product.setSellingPrice(req.getSellingPrice());
product.setImages(req.getImages());
product.setMrpPrice(req.getMrpPrice());
product.setSizes(req.getSizes());
product.setCreatedAt(LocalDateTime.now());

return productRepository.save(product);
}

public static int calculateDiscountPercentage(double mrpPrice, double sellingPrice) {
    if (mrpPrice <= 0) {
        throw new IllegalArgumentException("Actual price must be greater than zero.");
    }
    double discount = mrpPrice - sellingPrice;
    double discountPercentage = (discount / mrpPrice) * 100;
    return (int) discountPercentage;
}

@Override
public void deleteProduct(Long productId) throws ProductException {
    Product product=findProductById(productId);
    productRepository.delete(product);

}

@Override
public Product updateProduct(Long productId, Product product) throws ProductException {
    productRepository.findById(productId);
    product.setId(productId);
    return productRepository.save(product);
}

@Override
public Product updateProductStock(Long productId) throws ProductException {
    Product product = this.findProductById(productId);
    product.setIn_stock(!product.isIn_stock());
    return productRepository.save(product);
}

```

```
}

@Override
public Product findProductById(Long id) throws ProductException {
    return productRepository.findById(id)
        .orElseThrow(() -> new ProductException("product not found"));
}

@Override
public List<Product> searchProduct(String query) {
    return productRepository.searchProduct(query);
}

@Override
public Page<Product> getAllProduct(String category,
                                      String brand,
                                      String color,
                                      String size,
                                      Integer minPrice,
                                      Integer maxPrice,
                                      Integer minDiscount,
                                      String sort,
                                      String stock,
                                      Integer pageNumber) {
    Specification<Product> spec = (root, query, criteriaBuilder) -> {
        List<Predicate> predicates = new ArrayList<>();

        if (category != null) {
            Join<Product, Category> categoryJoin = root.join("category");
            //
            predicates.add(criteriaBuilder.equal(categoryJoin.get("categoryId"),
                category));
            //
            predicates.add(criteriaBuilder.equal(categoryJoin.get("parentCategory").get("ca
                tegoryId"), category));
            Predicate categoryPredicate = criteriaBuilder.or(
                criteriaBuilder.equal(categoryJoin.get("categoryId"),
                    category), // Match categoryId
                criteriaBuilder.equal(categoryJoin.get("parentCategory").get("categoryId"),
                    category) // Match parentCategory.categoryId
            );
            predicates.add(categoryPredicate);
        }
    }
}
```

```

        if (color != null && !color.isEmpty()) {
            System.out.println("color "+color);
            predicates.add(criteriaBuilder.equal(root.get("color"), color));
        }

        // Filter by size (single value)
        if (size != null && !size.isEmpty()) {
            predicates.add(criteriaBuilder.equal(root.get("size"), size));
        }

        if (minPrice != null) {

predicates.add(criteriaBuilder.greaterThanOrEqualTo(root.get("sellingPrice"),
        minPrice));
    }

        if (maxPrice != null) {

predicates.add(criteriaBuilder.lessThanOrEqualTo(root.get("sellingPrice"),
        maxPrice));
    }

        if (minDiscount != null) {

predicates.add(criteriaBuilder.greaterThanOrEqualTo(root.get("discountPercent"))
,
        minDiscount));
    }

        if (stock != null) {
            predicates.add(criteriaBuilder.equal(root.get("stock"), stock));
        }

        return criteriaBuilder.and(predicates.toArray(new Predicate[0]));
};

Pageable pageable;
if (sort != null && !sort.isEmpty()) {
    pageable = switch (sort) {
        case "price_low" ->
            PageRequest.of(pageNumber != null ? pageNumber : 0, 10,
Sort.by("sellingPrice").ascending());
        case "price_high" ->
            PageRequest.of(pageNumber != null ? pageNumber : 0, 10,
Sort.by("sellingPrice").descending());
        default -> PageRequest.of(pageNumber != null ? pageNumber : 0,
10, Sort.unsorted());
    };
} else {
}

```

```
        pageable = PageRequest.of(pageNumber != null ? pageNumber : 0, 10,
Sort.unsorted());
    }

    return productRepository.findAll(spec, pageable);
}

@Override
public List<Product> recentlyAddedProduct() {
    return List.of();
}

@Override
public List<Product> getProductBySellerId(Long sellerId) {
    return productRepository.findBySellerId(sellerId);
}
}

package com.zosh.service.impl;

import com.zosh.dto.RevenueChart;
import com.zosh.model.Order;
import com.zosh.repository.OrderRepository;
import com.zosh.service.RevenueService;
import lombok.RequiredArgsConstructor;
import org.springframework.stereotype.Service;

import java.time.LocalDate;
import java.time.LocalDateTime;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

@Service
@RequiredArgsConstructor
public class RevenueServiceImpl implements RevenueService {

    private final OrderRepository orderRepository;

    // Get daily revenue data for the past X days
    public List<RevenueChart> getDailyRevenueForChart(int days, Long sellerId) {
        List<RevenueChart> revenueData = new ArrayList<>();
        LocalDate currentDate = LocalDate.now();

        for (int i = days - 1; i >= 0; i--) {
            LocalDate date = currentDate.minusDays(i);
            double dailyRevenue = orderRepository
```

```

        .findBySellerIdAndOrderDateBetween(sellerId, date.atStartOfDay(),
date.plusDays(1).atStartOfDay())
            .stream()
            .mapToDouble(Order::getTotalSellingPrice)
            .sum();

        RevenueChart revenueChart=new RevenueChart();
        revenueChart.setRevenue(dailyRevenue);
        revenueChart.setDate(date.toString());

        revenueData.add(revenueChart);
    }

    return revenueData;
}

// Get monthly revenue data for the past X months
public List<RevenueChart> getMonthlyRevenueForChart(int months, Long
sellerId) {
    List<RevenueChart> revenueData = new ArrayList<>();
    LocalDate currentDate = LocalDate.now();

    for (int i = months - 1; i >= 0; i--) {
        LocalDate date = currentDate.minusMonths(i);
        LocalDate startOfMonth = date.withDayOfMonth(1);
        LocalDate startOfNextMonth = startOfMonth.plusMonths(1);

        double monthlyRevenue = orderRepository

        .findBySellerIdAndOrderDateBetween(sellerId, startOfMonth.atStartOfDay(),
startOfNextMonth.atStartOfDay())
            .stream()
            .mapToDouble(Order::getTotalSellingPrice)
            .sum();

        RevenueChart revenueChart=new RevenueChart();
        revenueChart.setRevenue(monthlyRevenue);
        revenueChart.setDate( date.getYear() + "-" + String.format("%02d",
date.getMonthValue()));

        revenueData.add(revenueChart);
    }

    return revenueData;
}

// Get yearly revenue data for the past X years

```

```

public List<RevenueChart> getYearlyRevenueForChart(int years, Long sellerId)
{
    List<RevenueChart> revenueData = new ArrayList<>();
    LocalDate currentDate = LocalDate.now();

    for (int i = years - 1; i >= 0; i--) {
        LocalDate startOfYear = currentDate.minusYears(i).withDayOfYear(1);
        LocalDate startOfNextYear = startOfYear.plusYears(1);

        double yearlyRevenue = orderRepository

            .findBySellerIdAndOrderDateBetween(sellerId, startOfYear.atStartOfDay(),
            startOfNextYear.atStartOfDay())
                .stream()
                .mapToDouble(Order::getTotalSellingPrice)
                .sum();

        RevenueChart revenueChart=new RevenueChart();
        revenueChart.setRevenue(yearlyRevenue);
        revenueChart.setDate(String.valueOf(startOfYear.getYear()));
        revenueData.add(revenueChart);
    }

    return revenueData;
}

@Override
public List<RevenueChart> getHourlyRevenueForChart(Long sellerId) {
    List<RevenueChart> revenueData = new ArrayList<>();

    // Get the current date
    LocalDate currentDate = LocalDate.now();

    // Define the start of the day (12 AM, or 00:00) and loop through 24
    hours
    LocalDateTime startOfDay = currentDate.atStartOfDay(); // 12 AM (00:00)
    of the current day

    // Loop through each hour of the day from 12 AM to 11:59 PM
    for (int i = 0; i < 24; i++) {
        LocalDateTime startOfHour = startOfDay.plusHours(i);
        LocalDateTime startOfNextHour = startOfHour.plusHours(1); // Next
        hour boundary

        // Calculate revenue for this hour (from startOfHour to
        startOfNextHour)
        double hourlyRevenue = orderRepository
            .findBySellerIdAndOrderDateBetween(sellerId, startOfHour,
            startOfNextHour)

```

```

        .stream()
        .mapToDouble(Order::getTotalSellingPrice)
        .sum();

        // Prepare the data for this hour
        RevenueChart revenueChart = new RevenueChart();
        revenueChart.setRevenue(hourlyRevenue);
        revenueChart.setDate(startOfHour.getHour() + ":00"); // Format as
"HH:00" (e.g., "00:00", "01:00")

        // Add the hourly revenue data to the list
        revenueData.add(revenueChart);
    }

    return revenueData;
}

@Override
public List<RevenueChart> getRevenueChartByType(String type, Long sellerId) {
    if(type.equals("monthly")){
        return this.getMonthlyRevenueForChart(12, sellerId);
    }
    else if(type.equals("daily")){
        return this.getDailyRevenueForChart(30, sellerId);
    }
    else return this.getHourlyRevenueForChart(sellerId);
}

}
package com.zosh.service.impl;

import com.zosh.exception.ReviewNotFoundException;
import com.zosh.model.Product;
import com.zosh.model.Review;
import com.zosh.model.User;
import com.zosh.repository.ReviewRepository;
import com.zosh.request.CreateReviewRequest;
import com.zosh.service.ReviewService;
import lombok.RequiredArgsConstructor;
import org.springframework.stereotype.Service;

import javax.naming.AuthenticationException;
import java.util.List;

@Service
@RequiredArgsConstructor
public class ReviewServiceImpl implements ReviewService {

    private final ReviewRepository reviewRepository;

```

```
    @Override
    public Review createReview(CreateReviewRequest req,
                               User user,
                               Product product) {
        Review newReview = new Review();

        newReview.setReviewText(req.getReviewText());
        newReview.setRating(req.getReviewRating());
        newReview.setProductImages(req.getProductImages());
        newReview.setUser(user);
        newReview.setProduct(product);

        product.getReviews().add(newReview);

        return reviewRepository.save(newReview);
    }

    @Override
    public List<Review> getReviewsByProductId(Long productId) {
        return reviewRepository.findReviewsByProductId(productId);
    }

    @Override
    public Review updateReview(Long reviewId,
                               String reviewText,
                               double rating,
                               Long userId) throws ReviewNotFoundException,
    AuthenticationException {
        Review review=reviewRepository.findById(reviewId)
            .orElseThrow(()-> new ReviewNotFoundException("Review Not
found"));

        if(review.getUser().getId()!=userId){
            throw new AuthenticationException("You do not have permission to
delete this review");
        }

        review.setReviewText(reviewText);
        review.setRating(rating);
        return reviewRepository.save(review);
    }

    @Override
    public void deleteReview(Long reviewId,Long userId) throws
ReviewNotFoundException,
    AuthenticationException {
```

```

        Review review=reviewRepository.findById(reviewId)
                .orElseThrow(() -> new ReviewNotFoundException("Review Not
found"));
        if(review.getUser().getId()!=userId){
            throw new AuthenticationException("You do not have permission to
delete this review");
        }
        reviewRepository.delete(review);
    }

}

package com.zosh.service.impl;

import com.zosh.model.Seller;
import com.zosh.model.SellerReport;
import com.zosh.repository.SellerReportRepository;
import com.zosh.service.SellerReportService;
import lombok.RequiredArgsConstructor;
import org.springframework.stereotype.Service;

import java.util.List;
import java.util.Optional;

@Service
@RequiredArgsConstructor
public class SellerReportServiceImpl implements SellerReportService {

    private final SellerReportRepository sellerReportRepository;

    @Override
    public SellerReport getSellerReport(Seller seller) {
        SellerReport report =
sellerReportRepository.findBySellerId(seller.getId());
        if(report == null){
            SellerReport newReport = new SellerReport();
            newReport.setSeller(seller);
            return sellerReportRepository.save(newReport);
        }
        return report;
    }

    @Override
    public SellerReport updateSellerReport(SellerReport sellerReport) {

        return sellerReportRepository.save(sellerReport);
    }
}

```



```

        newSeller.setEmail(seller.getEmail());
        newSeller.setPickupAddress(savedAddress);
        newSeller.setSellerName(seller.getSellerName());
        newSeller.setGSTIN(seller.getGSTIN());
        newSeller.setRole(USER_ROLE.ROLE_SELLER);
        newSeller.setMobile(seller.getMobile());

        newSeller.setPassword(passwordEncoder.encode(seller.getPassword()));
        newSeller.setBankDetails(seller.getBankDetails());
        newSeller.setBusinessDetails(seller.getBusinessDetails());

        System.out.println(newSeller);
        return sellerRepository.save(newSeller);
    }

    @Override
    public Seller getSellerById(Long id) throws SellerException {
        Optional<Seller> optionalSeller = sellerRepository.findById(id);
        if (optionalSeller.isPresent()) {
            return optionalSeller.get();
        }
        throw new SellerException("Seller not found");
    }

    @Override
    public Seller getSellerByEmail(String email) throws SellerException {
        Seller seller = sellerRepository.findByEmail(email);
        if (seller != null) {
            return seller;
        }
        throw new SellerException("Seller not found");
    }

    @Override
    public List<Seller> getAllSellers(AccountStatus status) {
        return sellerRepository.findByAccountStatus(status);
    }

    @Override
    public Seller updateSeller(Long id, Seller seller) throws SellerException {
        Seller existingSeller = sellerRepository.findById(id)
            .orElseThrow(() ->
                new SellerException("Seller not found with id " + id));

        if (seller.getSellerName() != null) {
            existingSeller.setSellerName(seller.getSellerName());
        }
        if (seller.getMobile() != null) {

```

```
        existingSeller.setMobile(seller.getMobile());
    }
    if (seller.getEmail() != null) {
        existingSeller.setEmail(seller.getEmail());
    }

    if (seller.getBusinessDetails() != null
        && seller.getBusinessDetails().getBusinessName() != null
    ) {

        existingSeller.getBusinessDetails().setBusinessName(
            seller.getBusinessDetails().getBusinessName()
        );
    }

    if (seller.getBankDetails() != null
        && seller.getBankDetails().getAccountHolderName() != null
        && seller.getBankDetails().getIfscCode() != null
        && seller.getBankDetails().getAccountNumber() != null
    ) {

        existingSeller.getBankDetails().setAccountHolderName(
            seller.getBankDetails().getAccountHolderName()
        );
        existingSeller.getBankDetails().setAccountNumber(
            seller.getBankDetails().getAccountNumber()
        );
        existingSeller.getBankDetails().setIfscCode(
            seller.getBankDetails().getIfscCode()
        );
    }

    if (seller.getPickupAddress() != null
        && seller.getPickupAddress().getAddress() != null
        && seller.getPickupAddress().getMobile() != null
        && seller.getPickupAddress().getCity() != null
        && seller.getPickupAddress().getState() != null
    ) {
        existingSeller.getPickupAddress()
            .setAddress(seller.getPickupAddress().getAddress());
    }

    existingSeller.getPickupAddress().setCity(seller.getPickupAddress().getCity());

    existingSeller.getPickupAddress().setState(seller.getPickupAddress().getState());
}

existingSeller.getPickupAddress().setMobile(seller.getPickupAddress().getMobile());
```

```

existingSeller.getPickupAddress().setPinCode(seller.getPickupAddress().getPinCode());
}
if (seller.getGSTIN() != null) {
    existingSeller.setGSTIN(seller.getGSTIN());
}

return sellerRepository.save(existingSeller);
}

@Override
public void deleteSeller(Long id) throws SellerException {
    if (sellerRepository.existsById(id)) {
        sellerRepository.deleteById(id);
    } else {
        throw new SellerException("Seller not found with id " + id);
    }
}

@Override
public Seller verifyEmail(String email, String otp) throws SellerException {
    Seller seller = this.getSellerByEmail(email);
    seller.setEmailVerified(true);
    return sellerRepository.save(seller);
}

@Override
public Seller updateSellerAccountStatus(Long sellerId, AccountStatus status)
throws SellerException {
    Seller seller = this.getSellerById(sellerId);
    seller.setAccountStatus(status);
    return sellerRepository.save(seller);
}
}

package com.zosh.service.impl;

import com.zosh.model.Order;
import com.zosh.model.Seller;
import com.zosh.model.Transaction;
import com.zosh.repository.SellerRepository;
import com.zosh.repository.TransactionRepository;
import com.zosh.service.TransactionService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import java.util.List;

```

```
@Service
public class TransactionServiceImpl implements TransactionService {

    private final TransactionRepository transactionRepository;
    private final SellerRepository sellerRepository;

    @Autowired
    public TransactionServiceImpl(TransactionRepository transactionRepository,
                                  SellerRepository sellerRepository
    ) {
        this.transactionRepository = transactionRepository;
        this.sellerRepository = sellerRepository;
    }

    @Override
    public Transaction createTransaction(Order order) {
        Seller seller = sellerRepository.findById(order.getSellerId()).get();
        Transaction transaction = new Transaction();
        transaction.setCustomer(order.getUser());
        transaction.setOrder(order);
        transaction.setSeller(seller);
        return transactionRepository.save(transaction);
    }

    @Override
    public List<Transaction> getTransactionBySeller(Seller seller) {
        return transactionRepository.findBySellerId(seller.getId());
    }

    @Override
    public List<Transaction> getAllTransactions() {
        return transactionRepository.findAll();
    }

}
package com.zosh.service.impl;

import java.util.Calendar;
import java.util.Date;
import java.util.List;
import java.util.UUID;

import com.zosh.exception.UserException;
import com.zosh.service.UserService;
import org.springframework.mail.SimpleMailMessage;
import org.springframework.mail.javamail.JavaMailSender;
import org.springframework.security.crypto.password.PasswordEncoder;
```

```
import org.springframework.stereotype.Service;

import com.zosh.config.JwtProvider;
import com.zosh.model.PasswordResetToken;
import com.zosh.model.User;
import com.zosh.repository.PasswordResetTokenRepository;
import com.zosh.repository.UserRepository;

@Service
public class UserServiceImplementation implements UserService {

    private UserRepository userRepository;
    private JwtProvider jwtProvider;
    private PasswordEncoder passwordEncoder;
    private PasswordResetTokenRepository passwordResetTokenRepository;
    private JavaMailSender javaMailSender;

    public UserServiceImplementation(
        UserRepository userRepository,
        JwtProvider jwtProvider,
        PasswordEncoder passwordEncoder,
        PasswordResetTokenRepository passwordResetTokenRepository,
        JavaMailSender javaMailSender) {

        this.userRepository=userRepository;
        this.jwtProvider=jwtProvider;
        this.passwordEncoder=passwordEncoder;
        this.passwordResetTokenRepository=passwordResetTokenRepository;
        this.javaMailSender=javaMailSender;
    }

    @Override
    public User findUserProfileByJwt(String jwt) throws UserException {
        String email=jwtProvider.getEmailFromJwtToken(jwt);

        User user = userRepository.findByEmail(email);

        if(user==null) {
            throw new UserException("user not exist with email "+email);
        }
        return user;
    }
}
```

```
    @Override
    public User findUserByEmail(String username) throws UserException {

        User user=userRepository.findByEmail(username);

        if(user!=null) {

            return user;
        }

        throw new UserException("user not exist with username "+username);
    }

}

package com.zosh.service.impl;

import com.zosh.model.VerificationCode;
import com.zosh.repository.VerificationCodeRepository;
import com.zosh.service.VerificationService;
import org.springframework.stereotype.Service;

@Service
public class VerificationServiceImpl implements VerificationService {

    private final VerificationCodeRepository verificationCodeRepository;

    VerificationServiceImpl(VerificationCodeRepository verificationCodeRepository) {

        this.verificationCodeRepository = verificationCodeRepository;
    }

    @Override
    public VerificationCode createVerificationCode(String otp, String email) {
        VerificationCode isExist=verificationCodeRepository.findByEmail(email);

        if(isExist!=null) {
            verificationCodeRepository.delete(isExist);
        }

        VerificationCode verificationCode=new VerificationCode();
        verificationCode.setOtp(otp);
        verificationCode.setEmail(email);

        return verificationCodeRepository.save(verificationCode);
    }
}
```

```
}

package com.zosh.service.impl;

import com.zosh.exception.WishlistNotFoundException;
import com.zosh.model.Product;
import com.zosh.model.User;
import com.zosh.model.Wishlist;
import com.zosh.repository.WishlistRepository;
import com.zosh.service.WishlistService;
import lombok.RequiredArgsConstructor;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import java.util.Optional;

@Service
@RequiredArgsConstructor
public class WishlistServiceImpl implements WishlistService {

    private final WishlistRepository wishlistRepository;

    @Override
    public Wishlist createWishlist(User user) {
        Wishlist wishlist = new Wishlist();
        wishlist.setUser(user);
        return wishlistRepository.save(wishlist);
    }

    @Override
    public Wishlist getWishlistById(User user) {
        Wishlist wishlist = wishlistRepository.findById(user.getId());
        if (wishlist == null) {
            wishlist = this.createWishlist(user);
        }
        return wishlist;
    }

    @Override
    public Wishlist addProductToWishlist(User user, Product product) throws
WishlistNotFoundException {
        Wishlist wishlist = this.getWishlistById(user);
        if(wishlist.getProducts().contains(product)){
            wishlist.getProducts().remove(product);
        }
        else wishlist.getProducts().add(product);

        return wishlistRepository.save(wishlist);
    }
}
```

```
}

package com.zosh.utils;

import java.util.Random;

public class OtpUtils {

    public static String generateOTP() {

        int otpLength = 6;

        Random random = new Random();

        StringBuilder otp = new StringBuilder(otpLength);

        for (int i = 0; i < otpLength; i++) {
            otp.append(random.nextInt(10));

        }

        return otp.toString();
    }
}

# Build stage
FROM node:18-alpine as build

# Set working directory
WORKDIR /app

# Copy only package files first for better caching
COPY package*.json ./
COPY pnpm-lock.yaml ./

# Install pnpm
RUN npm install -g pnpm

# Install dependencies
RUN pnpm install --frozen-lockfile

# Copy the rest of the application code
COPY . .

# Build the application
```

```
RUN pnpm run build

# Production stage
FROM nginx:alpine

# Copy built assets from build stage
COPY --from=build /app/dist /usr/share/nginx/html

# Copy nginx config
COPY nginx.conf /etc/nginx/conf.d/default.conf

# Expose port 80
EXPOSE 80

# Environment variables
ENV NODE_ENV=production

# Start nginx
CMD ["nginx", "-g", "daemon off;"]

# Health check
HEALTHCHECK --interval=30s --timeout=3s \
  CMD wget --no-verbose --tries=1 --spider http://localhost:80/ || exit 1
<!doctype html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <link rel="icon" type="image/svg+xml" href="/vite.svg" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Sajid Bazaar</title>

    <link
      href="https://fonts.googleapis.com/css2?family=Open+Sans:ital,wght@0,300..800;1
      ,300..800&family=Pacifico&display=swap"
      rel="stylesheet"
    />
  </head>
  <body>
    <div id="root"></div>
    <script type="module" src="/src/main.tsx"></script>
  </body>
</html>
server {
  listen 80;
  server_name localhost;

  location / {
    root /usr/share/nginx/html;
```

```
    index index.html;
    try_files $uri $uri/ /index.html;
}

# Proxy API requests to the backend
location /api/ {
    proxy_pass http://backend:5454/;
    proxy_http_version 1.1;
    proxy_set_header Upgrade $http_upgrade;
    proxy_set_header Connection 'upgrade';
    proxy_set_header Host $host;
    proxy_cache_bypass $http_upgrade;
}
}

"version": "1.10.0",
{
  "name": "axios",
  "version": "1.10.0",
  "description": "Promise based HTTP client for the browser and node.js",
  "main": "index.js",
  "exports": {
    ".": {
      "types": {
        "require": "./index.d.cts",
        "default": "./index.d.ts"
      },
      "react-native": {
        "require": "./dist/browser/axios.cjs",
        "default": "./dist/esm/axios.js"
      },
      "browser": {
        "require": "./dist/browser/axios.cjs",
        "default": "./index.js"
      },
      "default": {
        "require": "./dist/node/axios.cjs",
        "default": "./index.js"
      }
    },
    "./lib/adapters/http.js": "./lib/adapters/http.js",
    "./lib/adapters/xhr.js": "./lib/adapters/xhr.js",
    "./unsafe/*": "./lib/*",
    "./unsafe/core/settle.js": "./lib/core/settle.js",
    "./unsafe/core/buildFullPath.js": "./lib/core/buildFullPath.js",
    "./unsafe/helpers/isAbsoluteURL.js": "./lib/helpers/isAbsoluteURL.js",
    "./unsafe/helpers/buildURL.js": "./lib/helpers/buildURL.js",
    "./unsafe/helpers/combineURLs.js": "./lib/helpers/combineURLs.js",
    "./unsafe/adapters/http.js": "./lib/adapters/http.js",
    "./unsafe/adapters/xhr.js": "./lib/adapters/xhr.js",
  }
}
```

```
    "./unsafe/utils.js": "./lib/utils.js",
    "./package.json": "./package.json"
  },
  "type": "module",
  "types": "index.d.ts",
  "scripts": {
    "test": "npm run test:eslint && npm run test:mocha && npm run test:karma && npm run test:dtSLint && npm run test:exports",
    "test:eslint": "node bin/ssl_hotfix.js eslint lib/**/*.js",
    "test:dtSLint": "dtSLint --localTs node_modules/typescript/lib",
    "test:mocha": "node bin/ssl_hotfix.js mocha test/unit/**/*.js --timeout 30000 --exit",
    "test:exports": "node bin/ssl_hotfix.js mocha test/module/test.js --timeout 30000 --exit",
    "test:karma": "node bin/ssl_hotfix.js cross-env LISTEN_ADDR=: karma start karma.conf.cjs --single-run",
    "test:karma:firefox": "node bin/ssl_hotfix.js cross-env LISTEN_ADDR=: Browsers=Firefox karma start karma.conf.cjs --single-run",
    "test:karma:server": "node bin/ssl_hotfix.js cross-env karma start karma.conf.cjs",
    "test:build:version": "node ./bin/check-build-version.js",
    "start": "node ./sandbox/server.js",
    "preversion": "gulp version",
    "version": "npm run build && git add dist && git add package.json",
    "prepublishOnly": "npm run test:build:version",
    "postpublish": "git push && git push --tags",
    "build": "gulp clear && cross-env NODE_ENV=production rollup -c -m",
    "examples": "node ./examples/server.js",
    "coveralls": "cat coverage/lcov.info | ./node_modules/coveralls/bin/coveralls.js",
    "fix": "eslint --fix lib/**/*.js",
    "prepare": "husky install && npm run prepare:hooks",
    "prepare:hooks": "npx husky set .husky/commit-msg \"npx commitlint --edit $1\",
      \"release:dry\": \"release-it --dry-run --no-npm\",
      \"release:info\": \"release-it --release-version\",
      \"release:beta:no-npm\": \"release-it --preRelease=beta --no-npm\",
      \"release:beta\": \"release-it --preRelease=beta\",
      \"release:no-npm\": \"release-it --no-npm\",
      \"release:changelog:fix\": \"node ./bin/injectContributorsList.js && git add CHANGELOG.md\",
      \"release\": \"release-it\""
  },
  "repository": {
    "type": "git",
    "url": "https://github.com/axios/axios.git"
  },
  "keywords": [
    "xhr",
    "Promise",
    "Promise/A+",
    "ES6 Promises"
  ]
}
```



```
"Jay (https://github.com/jasonsaayman)",
"Emily Morehouse (https://github.com/emilyemorehouse)",
"Rubén Norte (https://github.com/rubennorte)",
"Justin Beckwith (https://github.com/JustinBeckwith)",
"Martti Laine (https://github.com/codeclown)",
"Xianming Zhong (https://github.com/chinesedfan)",
"Rikki Gibson (https://github.com/RikkiGibson)",
"Remco Haszing (https://github.com/remcohaszing)",
"Yasu Flores (https://github.com/yasuf)",
"Ben Carp (https://github.com/carpben)"

],
"sideEffects": false,
"release-it": {
  "git": {
    "commitMessage": "chore(release): v${version}",
    "push": true,
    "commit": true,
    "tag": true,
    "requireCommits": false,
    "requireCleanWorkingDir": false
  },
  "github": {
    "release": true,
    "draft": true
  },
  "npm": {
    "publish": false,
    "ignoreVersion": false
  },
  "plugins": {
    "@release-it/conventional-changelog": {
      "preset": "angular",
      "infile": "CHANGELOG.md",
      "header": "# Changelog"
    }
  },
  "hooks": {
    "before:init": "npm test",
    "after:bump": "gulp version --bump ${version} && npm run build && npm run test:build:version && git add ./dist && git add ./package-lock.json",
    "before:release": "npm run release:changelog:fix",
    "after:release": "echo Successfully released ${name} v${version} to ${repo.repository}."
  }
},
"commitlint": {
  "rules": {
    "header-max-length": [
      2,
```

```

    "always",
    130
  ]
},
"extends": [
  "@commitlint/config-conventional"
]
}
}

export default {
  plugins: {
    tailwindcss: {},
    autoprefixer: {}
  }
}
# React + TypeScript + Vite

```

This template provides a minimal setup to get React working in Vite with HMR and some ESLint rules.

Currently, two official plugins are available:

- [\[@vitejs/plugin-react\]](https://github.com/vitejs/vite-plugin-react) (<https://github.com/vitejs/vite-plugin-react/blob/main/packages/plugin-react>) uses [Babel] (<https://babeljs.io/>) for Fast Refresh
- [\[@vitejs/plugin-react-swc\]](https://github.com/vitejs/vite-plugin-react-swc) (<https://github.com/vitejs/vite-plugin-react/blob/main/packages/plugin-react-swc>) uses [SWC] (<https://swc.rs/>) for Fast Refresh

Expanding the ESLint configuration

If you are developing a production application, we recommend updating the configuration to enable type-aware lint rules:

```

```js
export default tseslint.config([
 globalIgnores(['dist']),
 {
 files: ['**/*.{ts,tsx}'],
 extends: [
 // Other configs...

 // Remove tseslint.configs.recommended and replace with this
 ...tseslint.configs.recommendedTypeChecked,
 // Alternatively, use this for stricter rules
 ...tseslint.configs.strictTypeChecked,
 // Optionally, add this for stylistic rules
 ...tseslint.configs.stylisticTypeChecked,
]
 }
])
```

```

```
// Other configs...
],
languageOptions: {
  parserOptions: {
    project: ['./tsconfig.node.json', './tsconfig.app.json'],
    tsconfigRootDir: import.meta.dirname,
  },
  // other options...
},
),
```
You can also install
[eslint-plugin-react-x] (https://github.com/Rellcx/eslint-react/tree/main/packages/plugins/eslint-plugin-react-x) and
[eslint-plugin-react-dom] (https://github.com/Rellcx/eslint-react/tree/main/packages/plugins/eslint-plugin-react-dom) for React-specific lint rules:
```
js
// eslint.config.js
import reactX from 'eslint-plugin-react-x'
import reactDom from 'eslint-plugin-react-dom'

export default tseslint.config([
  globalIgnores(['dist']),
  {
    files: ['**/*.{ts,tsx}'],
    extends: [
      // Other configs...
      // Enable lint rules for React
      reactX.configs['recommended-typescript'],
      // Enable lint rules for React DOM
      reactDom.configs.recommended,
    ],
    languageOptions: {
      parserOptions: {
        project: ['./tsconfig.node.json', './tsconfig.app.json'],
        tsconfigRootDir: import.meta.dirname,
      },
      // other options...
    },
  },
])
```
/** @type {import('tailwindcss').Config} */
export default {
 content: [
 "./index.html",

```

```
 "./src/**/*.{js,ts,jsx,tsx}",
],
theme: {
 extend: {},
},
plugins: [],
}

.App {
 font-family: "Open Sans", sans-serif;
 font-optical-sizing: auto;
 font-weight: 300;
 font-style: normal;
 font-variation-settings: "wdth" 100;
}
import './App.css';
import { ThemeProvider } from '@emotion/react';
import customeTheme from './Theme/customeTheme';
import { Button } from '@mui/material';
import Navbar from './customer/components/Navbar/Navbar';
import Home from './customer/pages/Home/Home';
import Footer from './customer/components/Footer/Footer';
import Products from './customer/pages/Products/Products';
import { Route, Routes, useNavigate } from 'react-router-dom';

import SellerDashboard from './seller/pages/SellerDashboard/SellerDashboard';
import CustomerRoutes from './routes/CustomerRoutes';
import AdminDashboard from './admin/pages/Dashboard/Dashboard';
import SellerAccountVerification from
'./seller/pages/SellerAccountVerification';
import SellerAccountVerified from './seller/pages/SellerAccountVerified';
import { useAppDispatch, useAppSelector } from './Redux Toolkit/Store';
import { useEffect } from 'react';
import { fetchSellerProfile } from './Redux Toolkit/Seller/sellerSlice';
import BecomeSeller from './customer/pages/BecomeSeller/BecomeSeller';
import AdminLoginForm from './admin/pages/Auth/AdminLogin';
import AdminAuth from './admin/pages/Auth/AdminAuth';
import { fetchUserProfile } from './Redux Toolkit/Customer/UserSlice';
import { createHomeCategories, fetchHomePageData } from './Redux
Toolkit/Customer/Customer/AsyncThunk';
import { homeCategories } from './data/homeCategories';
import Mobile from './data/Products/mobile';

function App() {
 const dispatch = useAppDispatch()
 // Use scoped selectors
 const jwt = useAppSelector(state => state.auth.jwt);
 const sellerJwt = useAppSelector(state => state.sellerAuth.jwt);
 const user = useAppSelector(state => state.user.user);
```

```

const sellers = useAppSelector(state => state.sellers);
const navigate = useNavigate();

useEffect(() => {
 const storedJwt = localStorage.getItem("jwt");
 if (storedJwt) {
 // Only fetch user profile if we don't have user data
 if (!user) {
 dispatch(fetchUserProfile({ jwt: storedJwt, navigate }));
 }

 // Only fetch seller profile if user is a seller and we have a seller JWT
 if (sellerJwt && user?.role === 'ROLE_SELLER') {
 dispatch(fetchSellerProfile(storedJwt));
 }
 }
}, [jwt, sellerJwt, user, dispatch, navigate]);

useEffect(() => {
 dispatch(createHomeCategories(homeCategories))
 // dispatch(fetchHomePageData())
}, [dispatch])

return (
 <ThemeProvider theme={customTheme}>
 <div className='App' >

 <Routes>
 {sellers?.profile && <Route path='/seller/*' element={<SellerDashboard />} />}
 {user?.role === "ROLE_ADMIN" && <Route path='/admin/*' element={<AdminDashboard />} />}
 <Route path='/verify-seller/:otp' element={<SellerAccountVerification />} />
 <Route path='/seller-account-verified' element={<SellerAccountVerified />} />
 <Route path='/become-seller' element={<BecomeSeller />} />
 <Route path='/admin-login' element={<AdminAuth />} />

 <Route path='/dummy' element={<Mobile />} />

 <Route path='*' element={<CustomerRoutes />} />

 </Routes>
 {/* <Footer/> */}
 </div>
)

```

```
 </ThemeProvider>
);
}

export default App;
/* @import "tailwindcss"; */

@tailwind base;
@tailwind components;
@tailwind utilities;
import { StrictMode } from 'react'
import { createRoot } from 'react-dom/client'
import './index.css'
import App from './App.tsx'
import { Provider } from 'react-redux'
import store from './Redux Toolkit/Store.ts'
import { BrowserRouter } from 'react-router-dom'

createRoot(document.getElementById('root')!).render(
<StrictMode>
 <Provider store={store}>
 <BrowserRouter>
 <App />
 </BrowserRouter>

 </Provider>

</StrictMode>,
)
import * as React from "react";
import DrawerList from "../../admin_seller/components/drawerList/DrawerList";
import DashboardIcon from '@mui/icons-material/Dashboard';
import LocalOfferIcon from '@mui/icons-material/LocalOffer';
import AddIcon from '@mui/icons-material/Add';
import HomeIcon from '@mui/icons-material/Home';
import ElectricBoltIcon from '@mui/icons-material/ElectricBolt';
import IntegrationInstructionsIcon from
'@mui/icons-material/IntegrationInstructions';
import { Category } from "@mui/icons-material";
import AccountBoxIcon from '@mui/icons-material/AccountBox';
import LogoutIcon from '@mui/icons-material/Logout';

const menu = [
{
 name: "Dashboard",
 path: "/admin",
 icon: <DashboardIcon className="text-primary-color" />,
 activeIcon: <DashboardIcon className="text-white" />,
```

```

 },
 {
 name: "Coupons",
 path: "/admin/coupon",
 icon: <IntegrationInstructionsIcon className="text-primary-color" />,
 activeIcon: <IntegrationInstructionsIcon className="text-white" />,
 },
 {
 name: "Add New Coupon",
 path: "/admin/add-coupon",
 icon: <AddIcon className="text-primary-color" />,
 activeIcon: <AddIcon className="text-white" />,
 },
 {
 name: "Home Page",
 path: "/admin/home-grid",
 icon: <HomeIcon className="text-primary-color" />,
 activeIcon: <HomeIcon className="text-white" />,
 },
 {
 name: "Electronics Category",
 path: "/admin/electronics-category",
 icon: <ElectricBoltIcon className="text-primary-color" />,
 activeIcon: <ElectricBoltIcon className="text-white" />,
 },
 {
 name: "Shop By Category",
 path: "/admin/shop-by-category",
 icon: <Category className="text-primary-color" />,
 activeIcon: <Category className="text-white" />,
 },
 {
 name: "Deals",
 path: "/admin/deals",
 icon: <LocalOfferIcon className="text-primary-color" />,
 activeIcon: <LocalOfferIcon className="text-white" />,
 },
],
}

const menu2 = [
 {
 name: "Account",
 path: "/seller/account",
 icon: <AccountBoxIcon className="text-primary-color" />,
 activeIcon: <AccountBoxIcon className="text-white" />,
 },
 {

```

```

 name: "Logout",
 path: "/",
 icon: <LogoutIcon className="text-primary-color" />,
 activeIcon: <LogoutIcon className="text-white" />,
 },
]

interface DrawerListProps{
 toggleDrawer?:any;
}

const AdminDrawerList = ({ toggleDrawer }: DrawerListProps) => {

 return (
 <>
 <DrawerList toggleDrawer={toggleDrawer} menu={menu} menu2={menu2}>
 </>
) ;
};

export default AdminDrawerList;
import React from 'react'
import AdminLoginForm from './AdminLogin'

const AdminAuth = () => {
 return (
 <div className='flex justify-center items-center h-screen'>
 <div className='max-w-4xl border rounded-md px-5 py-20'>
 <AdminLoginForm/>
 </div>
 </div>
)
}

export default AdminAuth
/* eslint-disable @typescript-eslint/no-explicit-any */
import { Button, CircularProgress, TextField } from '@mui/material'
import { useEffect, useState } from 'react'
import { useFormik } from 'formik';
import { useDispatch, useSelector } from '../../../../../Redux Toolkit/Store';
import { useNavigate } from 'react-router-dom';
import { sendLoginSignupOtp, signin } from '../../../../../Redux Toolkit/Customer/AuthSlice';
import OTPInput from '../../../../../customer/components/OtpField/OTPInput';

const AdminLoginForm = () => {

 const navigate = useNavigate();

```

```

const [otp, setOtp] = useState("");
const [isOtpSent, setIsOtpSent] = useState(false)
const [timer, setTimer] = useState<number>(30); // Timer state
const [isTimerActive, setIsTimerActive] = useState<boolean>(false);
const dispatch = useAppDispatch();
const { auth } = useAppSelector(store => store)

const formik = useFormik({
 initialValues: {
 email: '',
 otp: ''
 },
 onSubmit: (values: any) => {
 // Handle form submission
 dispatch(signin({ email: values.email, otp, navigate }));
 console.log('Form data:', values);
 }
});

const handleOtpChange = (otp: any) => {
 setOtp(otp);
};

const handleResendOTP = () => {
 // Implement OTP resend logic
 dispatch(sendLoginSignupOtp({ email: "signing_" + formik.values.email }));
 console.log('Resend OTP');
 setTimer(30);
 setIsTimerActive(true);
};

const handleSentOtp = () => {
 setIsOtpSent(true);
 handleResendOTP();
};

const handleLogin = () => {
 formik.handleSubmit()
};

useEffect(() => {
 let interval: any;

 if (isTimerActive) {

```

```

interval = setInterval(() => {
 setTimer(prev => {
 if (prev === 1) {
 clearInterval(interval);
 setIsTimerActive(false);
 return 30; // Reset timer for next OTP request
 }
 return prev - 1;
 });
}, 1000);
}

return () => {
 if (interval) clearInterval(interval);
},
[isTimerActive]);

return (
<div>
 <h1 className='text-center font-bold text-xl text-primary-color pb-8'>Login</h1>
 <form className="space-y-5">

 <TextField
 fullWidth
 name="email"
 label="Enter Your Email"
 value={formik.values.email}
 onChange={formik.handleChange}
 onBlur={formik.handleBlur}
 error={formik.touched.email && Boolean(formik.errors.email)}
 helperText={formik.touched.email ? formik.errors.email as string : undefined}>
 />

 {auth.otpSent && <div className="space-y-2">
 <p className="font-medium text-sm">
 * Enter OTP sent to your mobile number
 </p>
 <OTPIInput
 length={6}
 onChange={handleOtpChange}
 error={false}
 />
 <p className="text-xs space-x-2">
 {isTimerActive ? (
 Resend OTP in {timer} seconds
) : (
 OTP Expired
)}
 </p>
 </div>
 </form>
</div>

```

```

) : (
 <>
 Didn't receive OTP?{" "}
 <span
 onClick={handleResendOTP}
 className="text-teal-600 cursor-pointer
 hover:text-teal-800 font-semibold"
 >
 Resend OTP

 </>
) }
</p>
{ formik.touched.otp && formik.errors.otp &&
<p>{formik.errors.otp as string}</p>
</div>

{auth.otpSent && <div>
 <Button disabled={auth.loading} onClick={handleLogin}
 fullWidth variant='contained' sx={{ py: "11px" }}>{{
 auth.loading ? <CircularProgress />:
 "Login"}</Button>
 </div>

 {!auth.otpSent && <Button
 disabled={auth.loading}
 fullWidth
 variant='contained'
 onClick={handleSentOtp}
 sx={{ py: "11px" }}>{{
 auth.loading ? <CircularProgress />: "sent
otp"</Button>
 }
}

</form>

</div>
)
}

export default AdminLoginForm
import React, { useEffect } from 'react'
import CouponTable from './CouponTable'
import { useDispatch } from '../../../../../Redux Toolkit/Store'
import { fetchAllCoupons } from '../../../../../Redux Toolkit/Admin/AdminCouponSlice'

```

```

const Coupon = () => {
 const dispatch = useAppDispatch()
 useEffect(() => {
 dispatch(fetchAllCoupons(localStorage.getItem("jwt") || ""))
 }, [])
 return (
 <div>
 <CouponTable />
 </div>
)
}

export default Coupon
import * as React from 'react';
import Table from '@mui/material/Table';
import TableBody from '@mui/material/TableBody';
import TableCell, { TableCellClasses } from '@mui/material/TableCell';
import TableContainer from '@mui/material/TableContainer';
import TableHead from '@mui/material/TableHead';
import TableRow from '@mui/material/TableRow';
import Paper from '@mui/material/Paper';
import { Button, IconButton, Menu, MenuItem, Select, styled } from
'@mui/material';
import { useAppDispatch, useAppSelector } from '../../../../../Redux Toolkit/Store';
import type { Coupon } from '../../../../../types/couponTypes';
import DeleteOutlineIcon from '@mui/icons-material/DeleteOutline';
import { deleteCoupon } from '../../../../../Redux Toolkit/Admin/AdminCouponSlice';

const StyledTableCell = styled(TableCell)(({ theme }) => ({
 [`&.${tableCellClasses.head}`]: {
 backgroundColor: theme.palette.common.black,
 color: theme.palette.common.white,
 },
 [`&.${tableCellClasses.body}`]: {
 fontSize: 14,
 },
}));

const StyledTableRow = styled(TableRow)(({ theme }) => ({
 '&:nth-of-type(odd)': {
 backgroundColor: theme.palette.action.hover,
 },
 // hide last border
 '&:last-child td, &:last-child th': {
 border: 0,
 },
}));

const accountStatuses = [

```

```

 { status: 'ACTIVE', title: 'Active', description: 'Account is active and in
good standing' },
 { status: 'PENDING_VERIFICATION', title: 'Pending Verification',
description: 'Account is created but not yet verified' },
 { status: 'SUSPENDED', title: 'Suspended', description: 'Account is
temporarily suspended, possibly due to violations' },
 { status: 'DEACTIVATED', title: 'Deactivated', description: 'Account is
deactivated, user may have chosen to deactivate it' },
 { status: 'BANNED', title: 'Banned', description: 'Account is permanently
banned due to severe violations' },
 { status: 'CLOSED', title: 'Closed', description: 'Account is permanently
closed, possibly at user request' }
];

export default function CouponTable() {
 const [page, setPage] = React.useState(0);
 const [status, setStatus] = React.useState(accountStatuses[0].status)
 const { sellers, adminCoupon } = useAppSelector(store => store)
 const dispatch = useAppDispatch();

 const handleDeleteCoupon = (id:number) => {
 dispatch(deleteCoupon({ id, jwt: localStorage.getItem("jwt") || "" }))
 }

 return (
 <>
 <div className='pb-5 w-60'>
 <Select
 id="demo-simple-select"
 value={status}
 color='primary'
 className='text-primary-color'
 >
 {accountStatuses.map((status) =>
 <MenuItem
value={status.status}>{status.title}</MenuItem>)}
 </Select>
 </div>

 <TableContainer component={Paper}>
 <Table sx={{ minWidth: 700 }} aria-label="customized table">
 <TableHead>
 <TableRow>
 <StyledTableCell>Coupon Code</StyledTableCell>
 <StyledTableCell >Start Date</StyledTableCell>
 <StyledTableCell >End Date</StyledTableCell>
 <StyledTableCell >Min Order Value</StyledTableCell>
 <StyledTableCell >Discount %</StyledTableCell>

```

```

 <StyledTableCell
align="right">Status</StyledTableCell>
 <StyledTableCell
align="right">Delete</StyledTableCell>
 </TableRow>
 </TableHead>
 <TableBody>
 {adminCoupon.coupons?.map((coupon: Coupon) => (
 <StyledTableRow key={coupon.id}>
 <StyledTableCell component="th" scope="row">
 {coupon.code}
 </StyledTableCell>
 <StyledTableCell
>{coupon.validityStartDate}</StyledTableCell>
 <StyledTableCell
>{coupon.validityEndDate}</StyledTableCell>
 <StyledTableCell
>{coupon.minimumOrderValue}</StyledTableCell>
 <StyledTableCell
>{coupon.discountPercentage}</StyledTableCell>
 <StyledTableCell align="right">{coupon.active ? "Active" : "Deactive"}</StyledTableCell>
 <StyledTableCell align="right">
 <IconButton onClick={() =>
 handleDeleteCoupon(coupon.id)}>
 <DeleteOutlineIcon
className='text-red-700 cursor-pointer' />
 </IconButton>
 </StyledTableCell>
 </StyledTableRow>
))}
 </TableBody>
 </Table>
</TableContainer>
</>
);
}
import React, { useEffect, useState } from "react";
import { useFormik } from "formik";
import * as Yup from "yup";
import {
 TextField,
 Button,
 Box,
 Grid,
 Alert,
 Snackbar,
 CircularProgress,
} from "@mui/material";

```

```
import { DatePicker } from "@mui/x-date-pickers/DatePicker";
import { LocalizationProvider } from "@mui/x-date-pickers/LocalizationProvider";
import { AdapterDayjs } from "@mui/x-date-pickers/AdapterDayjs";
import dayjs, { Dayjs } from "dayjs";
import { useDispatch, useSelector } from "../../../../../Redux Toolkit/Store";
import { createCoupon } from "../../../../../Redux Toolkit/Admin/AdminCouponSlice";

interface CouponFormValues {
 code: string;
 discountPercentage: number;
 validityStartDate: Dayjs | null;
 validityEndDate: Dayjs | null;
 minimumOrderValue: number;
}

const CouponForm: React.FC = () => {
 const dispatch = useDispatch();
 const { coupon, adminCoupon } = useSelector((store) => store);
 const [snackbarOpen, setOpenSnackbar] = useState(false);

 const formik = useFormik<CouponFormValues>({
 initialValues: {
 code: "",
 discountPercentage: 0,
 validityStartDate: null,
 validityEndDate: null,
 minimumOrderValue: 0,
 },
 validationSchema: Yup.object({
 code: Yup.string()
 .required("Coupon code is required")
 .min(3, "Code should be at least 3 characters")
 .max(20, "Code should be at most 20 characters"),
 discountPercentage: Yup.number()
 .required("Discount percentage is required")
 .min(1, "Discount should be at least 1%")
 .max(100, "Discount cannot exceed 100%"),
 validityStartDate: Yup.date()
 .nullable()
 .required("Start date is required")
 .typeError("Invalid date"),
 validityEndDate: Yup.date()
 .nullable()
 .required("End date is required")
 .typeError("Invalid date")
 .min(
 Yup.ref("validityStartDate"),
 "End date cannot be before start date"
)
 })
 });
}
```

```

),
 minimumOrderValue: Yup.number()
 .required("Minimum order value is required")
 .min(1, "Minimum order value should be at least 1"),
) ,
 onSubmit: (values) => {
 const formattedValues = {
 ...values,
 validityStartDate: values.validityStartDate
 ? values.validityStartDate.toISOString()
 : null,
 validityEndDate: values.validityEndDate
 ? values.validityEndDate.toISOString()
 : null,
 };
 console.log("Form Values:", formattedValues);
 dispatch(
 createCoupon({
 coupon: formattedValues,
 jwt: localStorage.getItem("jwt") || "",
 })
);
 // Submit form values to the backend
},
)) ;

const handleCloseSnackbar = () => {
 setOpenSnackbar(false);
};

useEffect(() => {
 if (adminCoupon.couponCreated) {
 setOpenSnackbar(true);
 }
}, [adminCoupon.couponCreated]);

return (
<div className="max-w-3xl">
 <LocalizationProvider dateAdapter={AdapterDayjs}>
 <Box component="form" onSubmit={formik.handleSubmit} sx={{ mt: 3 }}>
 <Grid container spacing={2}>
 <Grid item xs={12} sm={6}>
 <TextField
 fullWidth
 id="code"
 name="code"
 label="Coupon Code"
 value={formik.values.code}
 onChange={formik.handleChange}

```

```
 onBlur={formik.handleBlur}
 error={formik.touched.code && Boolean(formik.errors.code) }
 helperText={formik.touched.code && formik.errors.code}
 margin="normal"
 />
</Grid>
<Grid item xs={12} sm={6}>
 <TextField
 fullWidth
 id="discountPercentage"
 name="discountPercentage"
 label="Discount Percentage"
 type="number"
 value={formik.values.discountPercentage}
 onChange={formik.handleChange}
 onBlur={formik.handleBlur}
 error={
 formik.touched.discountPercentage &&
 Boolean(formik.errors.discountPercentage)
 }
 helperText={
 formik.touched.discountPercentage &&
 formik.errors.discountPercentage
 }
 margin="normal"
 />
</Grid>
<Grid item xs={12} sm={6}>
 <DatePicker
 sx={{ width: "100%" }}
 label="Validity Start Date"
 value={formik.values.validityStartDate}
 onChange={(date) =>
 formik.setFieldValue("validityStartDate", date)
 }
 />
</Grid>
<Grid item xs={12} sm={6}>
 <DatePicker
 sx={{ width: "100%" }}
 label="Validity End Date"
 value={formik.values.validityEndDate}
 onChange={(date) =>
 formik.setFieldValue("validityEndDate", date)
 }
 />
</Grid>
<Grid item xs={12}>
 <TextField
```

```
 fullWidth
 id="minimumOrderValue"
 name="minimumOrderValue"
 label="Minimum Order Value"
 type="number"
 value={formik.values.minimumOrderValue}
 onChange={formik.handleChange}
 onBlur={formik.handleBlur}
 error={
 formik.touched.minimumOrderValue &&
 Boolean(formik.errors.minimumOrderValue)
 }
 helperText={
 formik.touched.minimumOrderValue &&
 formik.errors.minimumOrderValue
 }
 margin="normal"
 />
 </Grid>
 <Grid item xs={12}>
 <Button
 color="primary"
 variant="contained"
 type="submit"
 sx={{ mt: 2 }}
 fullWidth
 disabled={adminCoupon.loading}
 >
 {adminCoupon.loading ? (
 <CircularProgress
 size="small"
 sx={{ width: "27px", height: "27px" }}
 />
) : (
 "create coupon"
)}
 </Button>
 </Grid>
 </Box>
</LocalizationProvider>
<Snackbar
 anchorOrigin={{ vertical: "top", horizontal: "right" }}
 open={snackbarOpen}
 autoHideDuration={6000}
 onClose={handleCloseSnackbar}
>
 <Alert
 onClose={handleCloseSnackbar}
```

```

 severity={adminCoupon.error ? "error" : "success"}
 variant="filled"
 sx={{ width: "100%" }}
 >
 {adminCoupon.error ? adminCoupon.error : "Coupon created
successfully"}
 </Alert>
 </Snackbar>
</div>
);
};

export default CouponForm;
import React, { useEffect, useState } from 'react'
import AdminRoutes from '../../../../../routes/AdminRoutes'
// import DrawerList from './DrawerList'
import Navbar from '../../../../../admin_seller/components/navbar/Navbar'
import AdminDrawerList from '../../../../../components/DrawerList'
import { Alert, Snackbar } from '@mui/material'
import { useSelector } from '../../../../../Redux Toolkit/Store'

const AdminDashboard = () => {
 const { deal, admin } = useSelector(store => store)
 const [snackbarOpen, setOpenSnackbar] = useState(false);

 const handleCloseSnackbar = () => {
 setOpenSnackbar(false);
 }
 useEffect(() => {
 if (deal?.dealCreated || deal?.dealUpdated || deal?.error ||
admin?.categoryUpdated) {
 setOpenSnackbar(true);
 }
 }, [deal?.dealCreated, deal?.dealUpdated, deal?.error,
admin?.categoryUpdated]);
 return (
 <>
 <div className="min-h-screen">
 <Navbar DrawerList={AdminDrawerList} />
 <section className="lg:flex lg:h-[90vh]">
 <div className="hidden lg:block h-full">
 <AdminDrawerList />
 </div>
 <div className="p-10 w-full lg:w-[80%] overflow-y-auto">
 <AdminRoutes />
 </div>
 </section>
 </div>
 </>
);
}

```

```

 <Snackbar
 anchorOrigin={{ vertical: "top", horizontal: "right" }}
 open={snackbarOpen} autoHideDuration={6000}
 onClose={handleCloseSnackbar}
 >
 <Alert
 onClose={handleCloseSnackbar}
 severity={deal?.error ? "error" : "success"}
 variant="filled"
 sx={{ width: '100%' }}
 >
 {deal?.error ? deal.error : deal?.dealCreated ? "Deal created
successfully" : deal?.dealUpdated ? "Deal updated successfully" :
admin?.categoryUpdated ? "Category updated successfully" : ""}
 </Alert>
 </Snackbar>
</>

)
}

export default AdminDashboard
import { Box, Button, FormControl, FormHelperText, InputLabel, MenuItem,
Select, TextField, Typography } from '@mui/material'
import { useFormik } from 'formik';
import React from 'react'
import { useDispatch, useSelector } from '../../../../../Redux Toolkit/Store';
import { createDeal } from '../../../../../Redux Toolkit/Admin/DealSlice';

const CreateDealForm = () => {
 const { homePage } = useSelector(store => store);
 const dispatch = useDispatch()
 const formik = useFormik({
 initialValues: {
 discount: 0,
 category: '',
 },
 // validationSchema: validationSchema,
 onSubmit: (values) => {

 // console.log("Form Data -- :", values);
 dispatch(createDeal({
 discount: values.discount, category: {
 id: values.category
 }
 }))
 }
 })
}

```

```

 },
 });
console.log("----- ",homePage.homePageData?.dealCategories)
return (
<Box
 component="form"
 onSubmit={formik.handleSubmit}
 sx={{ maxWidth: 500, margin: "auto", padding: 3 }}
 className="space-y-6"
>
<Typography className='text-center' variant="h4" gutterBottom>
 Create Deal
</Typography>

{/* Image Field */}
<TextField
 fullWidth
 id="discount"
 name="discount"
 label="Discount"
 type="number"
 value={formik.values.discount}
 onChange={formik.handleChange}
 onBlur={formik.handleBlur}
 error={formik.touched.discount && Boolean(formik.errors.discount)}
 helperText={formik.touched.discount && formik.errors.discount}
/>

<FormControl
 fullWidth
 error={formik.touched.category && Boolean(formik.errors.category)}
 required
>
<InputLabel id="category-label">Category</InputLabel>
<Select
 labelId="category-label"
 id="category"
 name="category"
 value={formik.values.category}
 onChange={formik.handleChange}
 label="Category"
>
{homePage.homePageData?.dealCategories.map((item) => (
 <MenuItem value={item.id}>{item.categoryId}</MenuItem>
))
}
</Select>
{formik.touched.category && formik.errors.category && (
 <FormHelperText>{formik.errors.category}</FormHelperText>
)
}

```

```

 </FormControl>

 }

 /* Submit Button */
 <Button
 color="primary"
 variant="contained"
 fullWidth
 type="submit"
 sx={{ py: ".9rem" }}
 >
 Submit
 </Button>
</Box>
)
}

export default CreateDealForm
import { Button } from '@mui/material'
import React, { useState } from 'react'
import DealsTable from './DealsTable'
import DealsCategoryTable from './DealsCategoryTable'
import CreateDealForm from './CreateDealForm'
const tab = [
 { name: "Deals" },
 { name: "Categories" },
 { name: "Create Deal" }
]
const Deal = () => {
 const [activeTab, setActiveTab] = useState(tab[0].name);

 const handleActiveTab = (item: any) => {
 setActiveTab(item.name);
 }
 return (
 <div>
 <div className='>

 <div className='flex gap-4'>
 {tab.map((item) => <Button onClick={() =>
handleActiveTab(item)} variant={activeTab === item.name ? "contained" :
"outlined"}>{item.name}</Button>)}

 </div>
 <div className='mt-5'>
 {activeTab === "Deals" ? <DealsTable /> :
activeTab==="Categories"? <DealsCategoryTable />:<div className='mt-5 border-t flex flex-col justify-center items-center h-[70vh]'>

```

```

 <CreateDealForm/></div>}
 </div>

 </div>
 </div>
)
}

export default Deal
import { useAppSelector } from "../../Redux Toolkit/Store";
import HomeCategoryTable from "./HomeCategoryTable";

function DealsCategoryTable() {
const { homepage } = useAppSelector((store) => store);

return (
<>
 <HomeCategoryTable categories={homepage.homepageData?.dealCategories}/>
</>
);
}

export default DealsCategoryTable
import { Box, IconButton, Modal, Paper, styled, Table, TableBody, TableCell, tableCellClasses, TableContainer, TableHead, TableRow } from '@mui/material'
import React, { useEffect, useState } from 'react'
import { useAppDispatch, useAppSelector } from "../../Redux Toolkit/Store";
import EditIcon from '@mui/icons-material/Edit';
import { deleteDeal, getAllDeals } from "../../Redux Toolkit/Admin/DealSlice";
import UpdateDealForm from './UpdateDealForm';
import DeleteIcon from '@mui/icons-material/Delete';

const StyledTableCell = styled(TableCell)(({ theme }) => ({
[`&.${tableCellClasses.head}`]: {
 backgroundColor: theme.palette.common.black,
 color: theme.palette.common.white,
},
[`&.${tableCellClasses.body}`]: {
 fontSize: 14,
},
}));

const StyledTableRow = styled(TableRow)(({ theme }) => ({
"&:nth-of-type(odd)": {
 backgroundColor: theme.palette.action.hover,
},
})

```

```

// hide last border
"&:last-child td, &:last-child th": {
 border: 0,
},
})};

const style = {
 position: "absolute",
 top: "50%",
 left: "50%",
 transform: "translate(-50%, -50%)",
 width: 400,
 bgcolor: "background.paper",
 boxShadow: 24,
 p: 4,
};

const DealsTable = () => {
 const { homePage, deal } = useAppSelector(store => store)
 const [selectedDealId, setSelectedDealId] = useState<number>();
 const [open, setOpen] = React.useState(false);
 const dispatch = useAppDispatch()

 const handleOpen = (id: number | undefined) => () => {
 setSelectedDealId(id);
 setOpen(true);
 };
 const handleClose = () => setOpen(false);
 const handleDelete = (id: any) => () => {
 dispatch(deleteDeal(id))
 }
 useEffect(() => {
 dispatch(getAllDeals())
 }, [])
 return (
 <>
 <TableContainer component={Paper}>
 <Table sx={{ minWidth: 700 }} aria-label="customized table">
 <TableHead>
 <TableRow>
 <StyledTableCell>No</StyledTableCell>
 <StyledTableCell>image</StyledTableCell>
 <StyledTableCell>category</StyledTableCell>
 <StyledTableCell>Discount</StyledTableCell>
 <StyledTableCell align="right">Edit</StyledTableCell>
 <StyledTableCell align="right">Delete</StyledTableCell>
 </TableRow>
 </Table>
 </TableContainer>
 </>
);
}

export default DealsTable;

```

```

 </TableHead>
 <TableBody>
 {deal.deals.map(
 (deal: any, index) => (
 <StyledTableRow key={deal.id}>
 <StyledTableCell component="th" scope="row">
 {index + 1}
 </StyledTableCell>
 <StyledTableCell component="th" scope="row">
 <img
 className="w-20 rounded-md"
 src={deal.category.image}
 alt=""
 />
 </StyledTableCell>

 <StyledTableCell component="th" scope="row">
 {deal.category.categoryId}
 </StyledTableCell>
 <StyledTableCell component="th" scope="row">
 {deal.discount}%
 </StyledTableCell>

 <StyledTableCell align="right">
 <IconButton onClick={handleOpen(deal.id)}>
 <EditIcon className="text-orange-400 cursor-pointer" />
 </IconButton>
 </StyledTableCell>
 <StyledTableCell align="right">
 <IconButton onClick={handleDelete(deal.id)}>
 <DeleteIcon className="text-red-600 cursor-pointer" />
 </IconButton>
 </StyledTableCell>

 </StyledTableRow>
)
) }
 </TableBody>
 </Table>
 </TableContainer>
 {selectedDealId && <Modal
 open={open}
 onClose={handleClose}
 aria-labelledby="modal-modal-title"
 aria-describedby="modal-modal-description"
 >
 <Box sx={style}>
 <UpdateDealForm id={selectedDealId} />

```

```

 </Box>
 </Modal>}

 </>
)
}

export default DealsTable
import { useAppSelector } from "../../Redux Toolkit/Store";
import HomeCategoryTable from "./HomeCategoryTable";

function ElectronicsTable() {
 const { homepage } = useAppSelector((store) => store);

 return (
 <>
 <HomeCategoryTable
 categories={homepage.homepageData?.electricCategories}>
 </>
);
}
}

export default ElectronicsTable
import { useAppSelector } from "../../Redux Toolkit/Store";
import HomeCategoryTable from "./HomeCategoryTable";

export default function GridTable() {
 const { homepage } = useAppSelector((store) => store);

 return (
 <>
 <HomeCategoryTable categories={homepage.homepageData?.grid}>
 </>
);
}
}

import * as React from "react";
import Table from "@mui/material/Table";
import TableBody from "@mui/material/TableBody";
import TableCell, { TableCellClasses } from "@mui/material/TableCell";
import TableContainer from "@mui/material/TableContainer";
import TableHead from "@mui/material/TableHead";
import TableRow from "@mui/material/TableRow";
import Paper from "@mui/material/Paper";
import { Box, IconButton, Modal, styled } from "@mui/material";
import type { HomeCategory } from "../../types/homeDataTypes";
import EditIcon from "@mui/icons-material/Edit";

```

```

import UpdateHomeCategoryForm from "./UpdateHomeCategoryForm";

const StyledTableCell = styled(TableCell)(({ theme }) => ({
 [`&.${tableCellClasses.head}`]: {
 backgroundColor: theme.palette.common.black,
 color: theme.palette.common.white,
 },
 [`&.${tableCellClasses.body}`]: {
 fontSize: 14,
 },
})) ;

const StyledTableRow = styled(TableRow)(({ theme }) => ({
 "&:nth-of-type(odd)": {
 backgroundColor: theme.palette.action.hover,
 },
 // hide last border
 "&:last-child td, &:last-child th": {
 border: 0,
 },
})) ;

const style = {
 position: "absolute",
 top: "50%",
 left: "50%",
 transform: "translate(-50%, -50%)",
 width: 400,
 bgcolor: "background.paper",
 boxShadow: 24,
 p: 4,
};

function HomeCategoryTable({categories}:{categories:HomeCategory[] | undefined}) {
 const [selectedCategory, setSelectedCategory] =
 React.useState<HomeCategory>();
 const [open, setOpen] = React.useState(false);
 const handleOpen = (category: HomeCategory | undefined) => () => {
 setSelectedCategory(category);
 setOpen(true);
 };
 const handleClose = () => setOpen(false);

 return (
 <>
 <TableContainer component={Paper}>
 <Table sx={{ minWidth: 700 }} aria-label="customized table">

```

```

<TableHead>
 <TableRow>
 <StyledTableCell>No</StyledTableCell>
 <StyledTableCell>Id</StyledTableCell>
 <StyledTableCell>image</StyledTableCell>
 <StyledTableCell align="right">category</StyledTableCell>
 <StyledTableCell align="right">Name</StyledTableCell>
 </TableRow>
</TableHead>
<TableBody>
 {categories?.map(
 (category: HomeCategory, index) => (
 <StyledTableRow key={category.categoryId}>
 <StyledTableCell component="th" scope="row">
 {index + 1}
 </StyledTableCell>
 <StyledTableCell component="th" scope="row">
 {category.id}
 </StyledTableCell>
 <StyledTableCell component="th" scope="row">
 <img
 className="w-20 rounded-md"
 src={category.image}
 alt=""
 />
 </StyledTableCell>
 <StyledTableCell align="right" component="th" scope="row">
 {category.categoryId}
 </StyledTableCell>

 <StyledTableCell align="right">
 <IconButton onClick={handleOpen(category)}>
 <EditIcon className="text-orange-400 cursor-pointer" />
 </IconButton>
 </StyledTableCell>
 </StyledTableRow>
)
) }
</TableBody>
</Table>
</TableContainer>
<Modal
 open={open}
 onClose={handleClose}
 aria-labelledby="modal-modal-title"
 aria-describedby="modal-modal-description"
>
 <Box sx={style}>
 <UpdateHomeCategoryForm

```

```

 category={selectedCategory}
 handleClose={handleClose}
 />
 </Box>
 </Modal>
</>
);
}

export default HomeCategoryTable
import React from 'react'
import HomeCategoryCard from
'../../../../customer/pages/Home/HomeCategory/HomeCategoryCard'
import HomeCategoryTable from './HomeCategoryTable'
import { useAppSelector } from '../../../../Redux Toolkit/Store';

const ShopByCategoryTable = () => {
 const { homePage } = useAppSelector((store) => store);
 return (
 <HomeCategoryTable categories={homePage.homePageData?.shopByCategories}/>
)
}

export default ShopByCategoryTable
// UpdateDealForm.tsx

import React, { useEffect } from "react";
import { useFormik } from "formik";
import * as Yup from "yup";
import {
 TextField,
 Button,
 Typography,
} from "@mui/material";
import { useAppDispatch, useAppSelector } from "../../../../Redux Toolkit/Store";
import { fetchHomeCategories } from "../../../../Redux Toolkit/Admin/AdminSlice";
import { updateDeal } from "../../../../Redux Toolkit/Admin/DealSlice";

// Validation schema using Yup
const validationSchema = Yup.object({
 discount: Yup.number()
 .required("Discount is required")
 .min(0, "Discount must be a positive number")
 .max(100, "Discount cannot exceed 100"),
 // category: Yup.string()
 // .oneOf(Object.values(HomeCategory), 'Invalid category')
 // .required('Category is required'),
}) ;

```

```

// Initial form values
const initialValues = {
 discount: 0,
 category: '',
};

const UpdateDealForm = ({ id }: { id: number }) => {
 const { admin } = useAppSelector((store) => store);
 const dispatch = useAppDispatch();
 const formik = useFormik({
 initialValues,
 validationSchema,
 onSubmit: (values) => {
 console.log("Deal submit", values);
 dispatch(
 updateDeal(
 {
 id,
 deal: {
 discount: values.discount,
 category: { id: Number(values.category) },
 }
 }
)
);
 },
 });

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

 return (
 <form className="space-y-4" onSubmit={formik.handleSubmit}>
 <Typography align="center" variant="h4" gutterBottom>
 Update Deal
 </Typography>

 <TextField
 fullWidth
 id="discount"
 name="discount"
 label="Discount"
 type="number"
 value={formik.values.discount}
 onChange={formik.handleChange}
 onBlur={formik.handleBlur}
 error={formik.touched.discount &&
Boolean(formik.errors.discount)}>

```

```

 helperText={formik.touched.discount && formik.errors.discount}
 />

 <Button
 sx={{ py: ".8rem" }}
 color="primary"
 variant="contained"
 fullWidth
 type="submit"
 >
 Update Deal
 </Button>
 </form>
);
};

export default UpdateDealForm;
import React from "react";
import { useFormik } from "formik";
import * as Yup from "yup";
import {
 Button,
 TextField,
 Typography,
 MenuItem,
 InputLabel,
 FormControl,
 Select,
 Box,
 FormHelperText,
} from "@mui/material";
import { mainCategory } from "../../../../../data/category/mainCategory";
import { menLevelTwo } from "../../../../../data/category/level two/menLevelTwo";
import { womenLevelTwo } from "../../../../../data/category/level two/womenLevelTwo";
import { menLevelThree } from "../../../../../data/category/level
three/menLevelThree";
import { womenLevelThree } from "../../../../../data/category/level
three/womenLevelThree";
import { furnitureLevelThree } from "../../../../../data/category/level
three/furnitureLevelThree";
import { electronicsLevelThree } from "../../../../../data/category/level
three/electronicsLevelThree";
import { furnitureLevelTwo } from "../../../../../data/category/level
two/furnitureLevleTwo";
import { electronicsLevelTwo } from "../../../../../data/category/level
two/electronicsLavelTwo";
import { useDispatch } from "../../../../../Redux Toolkit/Store";
import { updateHomeCategory } from "../../../../../Redux Toolkit/Admin/AdminSlice";
import type { HomeCategory } from "../../../../../types/homeDataTypes";

```

```
// Define validation schema using Yup
const validationSchema = Yup.object({
 image: Yup.string().required("Image is required"),
 category: Yup.string().required("Category is required"),
});

const categoryTwo: { [key: string]: any[] } = {
 men: menLevelTwo,
 women: womenLevelTwo,
 home_furniture: furnitureLevelTwo,
 beauty: [],
 electronics: electronicsLevelTwo,
};

const categoryThree: { [key: string]: any[] } = {
 men: menLevelThree,
 women: womenLevelThree,
 home_furniture: furnitureLevelThree,
 beauty: [],
 electronics: electronicsLevelThree,
};

const UpdateHomeCategoryForm = ({
 category,
 handleClose,
}: {
 category: HomeCategory | undefined;
 handleClose: () => void;
}) => {
 const dispatch = useAppDispatch();
 const formik = useFormik({
 initialValues: {
 image: "",
 category: "",
 category2: "",
 category3: "",
 },
 validationSchema: validationSchema,
 onSubmit: (values) => {
 // const data =
 console.log("Form Data:", values, category);
 if (category?.id) {
 dispatch(
 updateHomeCategory({
 id: category.id,
 data: { image: values.image, categoryId: values.category3 },
 })
);
 }
 }
 });
}
```

```
 handleClose()
 },
});

const childCategory = (category: any, parentCategoryId: any) => {
 return category.filter((child: any) => {
 // console.log("Category", parentCategoryId, child)
 return child.parentCategoryId == parentCategoryId;
 });
};

return (
 <Box
 component="form"
 onSubmit={formik.handleSubmit}
 sx={{ maxWidth: 500, margin: "auto", padding: 3 }}
 className="space-y-6"
 >
 <Typography variant="h4" gutterBottom>
 Update Category
 </Typography>

 {/* Image Field */}
 <TextField
 fullWidth
 id="image"
 name="image"
 label="Image URL"
 value={formik.values.image}
 onChange={formik.handleChange}
 onBlur={formik.handleBlur}
 error={formik.touched.image && Boolean(formik.errors.image)}
 helperText={formik.touched.image && formik.errors.image}
 />

 <FormControl
 fullWidth
 error={formik.touched.category && Boolean(formik.errors.category)}
 required
 >
 <InputLabel id="category-label">Category2</InputLabel>
 <Select
 labelId="category-label"
 id="category"
 name="category"
 value={formik.values.category}
 onChange={formik.handleChange}
 label="Category2"
 >
```

```

 {/* <MenuItem value="">None</MenuItem> */}
 {mainCategory.map((item) =>
 <MenuItem value={item.categoryId}>{item.name}</MenuItem>
)))
 </Select>
 {formik.touched.category && formik.errors.category && (
 <FormHelperText>{formik.errors.category}</FormHelperText>
)}
</FormControl>

<FormControl
 fullWidth
 error={formik.touched.category && Boolean(formik.errors.category)}
 required
>
 <InputLabel id="category2-label">Second Category</InputLabel>
 <Select
 labelId="category2-label"
 id="category2"
 name="category2"
 value={formik.values.category2}
 onChange={formik.handleChange}
 label="Second Category"
 >
 {formik.values.category &&
 categoryTwo[formik.values.category] ?.map((item) => (
 <MenuItem value={item.categoryId}>{item.name}</MenuItem>
)))
 </Select>
 {formik.touched.category && formik.errors.category && (
 <FormHelperText>{formik.errors.category}</FormHelperText>
)}
</FormControl>
<FormControl
 fullWidth
 error={formik.touched.category && Boolean(formik.errors.category)}
 required
>
 <InputLabel id="category-label">Third Category</InputLabel>
 <Select
 labelId="category-label"
 id="category"
 name="category3"
 value={formik.values.category3}
 onChange={formik.handleChange}
 label="Third Category"
 >
 <MenuItem value="">
 None

```

```

 </MenuItem>
 {formik.values.category2 &&
 childCategory(
 categoryThree[formik.values.category],
 formik.values.category2
)?.map((item: any) => (
 <MenuItem value={item.categoryId}>{item.name}</MenuItem>
)));
 </Select>
 {formik.touched.category && formik.errors.category && (
 <FormHelperText>{formik.errors.category}</FormHelperText>
)}
 </FormControl>

 {/* Submit Button */}
 <Button
 color="primary"
 variant="contained"
 fullWidth
 type="submit"
 sx={{ py: ".9rem" }}
 >
 Submit
 </Button>
 </Box>
);
};

export default UpdateHomeCategoryForm;
import * as React from 'react';
import Table from '@mui/material/Table';
import TableBody from '@mui/material/TableBody';
import TableCell, { tableCellClasses } from '@mui/material/TableCell';
import TableContainer from '@mui/material/TableContainer';
import TableHead from '@mui/material/TableHead';
import TableRow from '@mui/material/TableRow';
import Paper from '@mui/material/Paper';
import { Button, FormControl, Menu, MenuItem, Select, styled } from '@mui/material';
import { useDispatch, useSelector } from '../../../../../Redux Toolkit/Store';
import { fetchSellers, updateSellerAccountStatus } from '../../../../../Redux Toolkit/Seller/sellerSlice';

function createData(
 name: string,
 calories: number,
 fat: number,
 carbs: number,
 protein: number,

```

```

) {
 return { name, calories, fat, carbs, protein };
}

const StyledTableCell = styled(TableCell)(({ theme }) => ({
 [`&.${tableCellClasses.head}`]: {
 backgroundColor: theme.palette.common.black,
 color: theme.palette.common.white,
 },
 [`&.${tableCellClasses.body}`]: {
 fontSize: 14,
 },
}));

const StyledTableRow = styled(TableRow)(({ theme }) => ({
 '&:nth-of-type(odd)': {
 backgroundColor: theme.palette.action.hover,
 },
 // hide last border
 '&:last-child td, &:last-child th': {
 border: 0,
 },
}));

const accountStatuses = [
 { status: 'PENDING_VERIFICATION', title: 'Pending Verification',
description: 'Account is created but not yet verified' },
 { status: 'ACTIVE', title: 'Active', description: 'Account is active and in
good standing' },
 { status: 'SUSPENDED', title: 'Suspended', description: 'Account is
temporarily suspended, possibly due to violations' },
 { status: 'DEACTIVATED', title: 'Deactivated', description: 'Account is
deactivated, user may have chosen to deactivate it' },
 { status: 'BANNED', title: 'Banned', description: 'Account is permanently
banned due to severe violations' },
 { status: 'CLOSED', title: 'Closed', description: 'Account is permanently
closed, possibly at user request' }
];

export default function SellersTable() {
 const [page, setPage] = React.useState(0);
 const [accountStatus, setAccountStatus] = React.useState("ACTIVE")
 const { sellers } = useAppSelector(store => store)

 const dispatch = useAppDispatch();
}

```

```

 React.useEffect(() => {
 dispatch(fetchSellers(accountStatus))
 }, [accountStatus])

 const handleAccountStatusChange = (event: any) => {
 setAccountStatus(event.target.value as string);
 }

 const handleUpdateSellerAccountStatus = (id: number, status: string) => {
 dispatch(updateSellerAccountStatus({ id, status }))
 }

 const [anchorEl, setAnchorEl] = React.useState<{ [key: number]: HTMLElement | null }>({});

 const handleClick = (event: React.MouseEvent<HTMLButtonElement>, sellerId: any) => {
 setAnchorEl((prev) => ({ ...prev, [sellerId]: event.currentTarget }));
 };

 const handleClose = (sellerId: number) => {
 setAnchorEl((prev) => ({ ...prev, [sellerId]: null }));
 };

 return (
 <>
 <div className='pb-5 w-60'>
 <FormControl color='primary' fullWidth>
 <Select
 // labelId="demo-simple-select-label"
 id="demo-simple-select"
 value={accountStatus}
 onChange={handleAccountStatusChange}
 color='primary'
 className='text-primary-color'

 >
 {accountStatuses.map((status) =>
 <MenuItem
 value={status.status}>{status.title}</MenuItem>)
 }
 </Select>
 </FormControl>
 </div>

 <TableContainer component={Paper}>
 <Table sx={{ minWidth: 700 }} aria-label="customized table">
 <TableHead>
 <TableRow>
 <StyledTableCell>Seller Name</StyledTableCell>
 <StyledTableCell >Email</StyledTableCell>

```

```

 <StyledTableCell>Mobile</StyledTableCell>
 <StyledTableCell>GSTIN</StyledTableCell>
 <StyledTableCell>Business Name</StyledTableCell>
 <StyledTableCell align="right">Account
Status</StyledTableCell>
 <StyledTableCell align="right">Change
Status</StyledTableCell>
 </TableRow>
</TableHead>
<TableBody>
 {sellers.sellers?.map((seller) => (
 <StyledTableRow key={seller.sellerName}>
 <StyledTableCell component="th" scope="row">
 {seller.sellerName}
 </StyledTableCell>
 <StyledTableCell>
>{seller.email}</StyledTableCell>
 <StyledTableCell>
>{seller.mobile}</StyledTableCell>
 <StyledTableCell>
>{seller.gstin}</StyledTableCell>
 <StyledTableCell>
>{seller.businessDetails?.businessName}</StyledTableCell>
 <StyledTableCell>
align="right">{seller.accountStatus}</StyledTableCell>
 <StyledTableCell align="right">
 <Button
 id={"basic-button" + seller.id}
 onClick={(e) => handleClick(e,
seller.id)}
 >
 Change Status
 </Button>
 <Menu
 id={"basic-menus" + seller.id}
 anchorEl={anchorEl[seller.id || 1]}
 open={Boolean(anchorEl[seller.id || 1])}
 onClose={()=>handleClose(seller.id || 1)}
 >
 {accountStatuses.map((status) =>
 <MenuItem onClick={() =>
 handleUpdateSellerAccountStatus(
 seller.id || 1, status.status)
 } value={status.status}>{status.title}</MenuItem>)
 </Menu>

```

```

 </StyledTableCell>

 </StyledTableRow>
))}
</TableBody>
</Table>
</TableContainer>
</>

);

}

.activeTab{
 background-color: var(--primary-color);
}

import * as React from "react";
import Divider from "@mui/material/Divider";
import ListItemIcon from "@mui/material(ListItemIcon";
importListItemText from "@mui/material(ListItemText";
import { useLocation, useNavigate } from "react-router-dom";
import { useDispatch } from "../../Redux Toolkit/Store";
import { performLogout } from "../../Redux Toolkit/Customer/AuthSlice";

export interface Menu{
 name: string;
 path: string;
 icon: React.ReactElement<any>;
 activeIcon: React.ReactElement<any>;
}

interface DrawerListProps{
 toggleDrawer?:any;
 menu:Menu[];
 menu2:Menu[];
}

const DrawerList = ({ toggleDrawer,menu,menu2 }: DrawerListProps) => {

 const dispatch = useDispatch()

 const location = useLocation();
 const navigate = useNavigate();

 const handleLogout = () => {
 dispatch(performLogout())
 }

 const handleClick = (item: any)=>() => {

```

```

 if (item.name === "Logout") {
 handleLogout()

 }
 navigate(item.path);
 if(toggleDrawer) toggleDrawer(false)();
 }
 return (
 <div className="h-full">
 <div
 className="flex flex-col justify-between h-full w-[300px] border-r py-5"
 >
 <div>
 <div className="space-y-2">
 {menu.map((item, index) => (
 <div key={item.name}
 onClick={handleClick(item)}
 className="pr-9 cursor-pointer">
 <p className={`${item.path === location.pathname ? "bg-primary-color text-white" : "text-primary-color"} flex items-center px-5 py-3 rounded-r-full`}>
 <ListIcon>{location.pathname === item.path ? item.activeIcon : item.icon}</ListIcon>
 <ListItemText primary={item.name} />
 </p>
 </div>
))}
 </div>
 <div className="space-y-4">
 <Divider />
 <div className="space-y-2">
 {menu2.map((item, index) => (
 <div onClick={handleClick(item)} className="pr-9 cursor-pointer" key={item.name}>
 <p className={`${item.path === location.pathname ? "bg-primary-color text-white" : "text-primary-color"} flex items-center px-5 py-3 rounded-r-full`}>
 <ListIcon>{location.pathname === item.path ? item.activeIcon : item.icon}</ListIcon>
 <ListItemText primary={item.name} />
 </p>
 </div>
))}
 </div>
 </div>
 </div>
 </div>
)
)

```

```
 </div>
);
};

export default DrawerList;
import React from 'react'
import MenuIcon from '@mui/icons-material/Menu';
import { Drawer, IconButton } from '@mui/material';
import { useNavigate } from 'react-router-dom';

const Navbar = ({DrawerList}:any) => {
 const navigate = useNavigate()
 const [open, setOpen] = React.useState(false);

 const toggleDrawer = (newOpen: any)=>() => {
 setOpen(newOpen);
 };

 return (
 <div className='h-[10vh] flex items-center px-5 border-b'>
 <div className='flex items-center gap-3 '>
 <IconButton onClick={toggleDrawer(true)} color='primary'>
 <MenuIcon color='primary' />
 </IconButton>

 <h1 onClick={() => navigate("/")} className='logo text-xl cursor-pointer'>Sajid Bazzar</h1>
 </div>

 <Drawer open={open} onClose={toggleDrawer(false)}>
 <DrawerList toggleDrawer={toggleDrawer} />
 </Drawer>
 </div>
)
}

export default Navbar
import axios from 'axios';

export const API_URL = "http://localhost:5454";
export const DEPLOYED_URL = "https://sajid-bazzar-backend.onrender.com"
// change api

export const api = axios.create({
 baseURL: API_URL,
 headers: {
 'Content-Type': 'application/json',
 }
}
```

```

},
});

import React from 'react'

const Footer = () => {
 return (
 <footer className="mt-20 p-20 bg-gray-800 text-white py-4">
 <div className="container mx-auto px-4">
 <div className="flex flex-wrap justify-between items-center">
 <div className="text-center md:text-left">
 <h5 className="text-lg font-semibold">Sajid Bazzar</h5>
 <p className="text-sm mt-2">
 © {new Date().getFullYear()} Sajid Bazzar. All rights reserved.
 </p>
 </div>
 <div className="text-center mt-4 md:mt-0">
 <ul className="flex justify-center space-x-4">
 Home
 About
 Services
 Contact

 </div>
 </div>
 </div>
 </footer>
)
}

export default Footer
import React from 'react'
import { menLevelThree } from '../../../../../data/category/level
three/menLevelThree'
import { menLevelTwo } from '../../../../../data/category/level two/menLevelTwo'
import { womenLevelThree } from '../../../../../data/category/level
three/womenLevelThree'
import { womenLevelTwo } from '../../../../../data/category/level two/womenLevelTwo'
import { useNavigate } from 'react-router-dom'
import { Box } from '@mui/material'
import { electronicsLevelTwo } from '../../../../../data/category/level
two/electronicsLevelTwo'
import { furnitureLevelTwo } from '../../../../../data/category/level
two/furnitureLevleTwo'
import { furnitureLevelThree } from '../../../../../data/category/level
three/furnitureLevelThree'
import { electronicsLevelThree } from '../../../../../data/category/level
three/electronicsLevelThree'

```

```

const categoryTwo: { [key: string]: any[] } = {
 men: menLevelTwo,
 women: womenLevelTwo,
 electronics:electronicsLevelTwo,
 home_furniture:furnitureLevelTwo,
}

const categoryThree: { [key: string]: any[] } = {
 men: menLevelThree,
 women: womenLevelThree,
 electronics:electronicsLevelThree,
 home_furniture:furnitureLevelThree,
}

const CategorySheet = ({ selectedCategory, toggleDrawer, setShowSheet }: any) =>
{
 const navigate=useNavigate()

 const childCategory = (category: any, parentCategoryId: any) => {
 return category.filter((child: any) => {
 // console.log("Category", parentCategoryId, child)
 return child.parentCategoryId == parentCategoryId
 })
 }

 const handleCategoryClick = (category:string) => {
 if(toggleDrawer){
 toggleDrawer(false)()
 }
 if(setShowSheet){
 setShowSheet(false)
 }

 navigate("/products/"+category)
 }
 return (
 <Box className='bg-white shadow-lg lg:h-[500px] overflow-y-auto'>
 <div className=' flex text-sm flex-wrap'>
 {categoryTwo[selectedCategory]?.map((item: any,index) =>
 <div key={item.name} className={`${`p-8 lg:w-[20%]`}`}
${index%2==0?"bg-slate-50":"bg-white"}>

```

```

 <p className='text-[#00927c] mb-5
font-semibold'>{item.name}</p>

 <ul className='space-y-3'>
 {childCategory(categoryThree[selectedCategory],
item.categoryId)?.map((item: any) => <div key={item.name}>
 <li
 onClick={()=>handleCategoryClick(item.categoryId)}
 className='hover:text-[#00927c] cursor-pointer'>
 {item.name}

 </div>) }

 </div>
</Box>
)
}

export default CategorySheet
import { Box, Divider, List, ListItem, ListItemButton, ListItemText } from
'@mui/material'
import React, { useState } from 'react'
import { mainCategory } from '../../../../../data/category/mainCategory'
import CategorySheet from './CategorySheet';

const DrawerList = ({toggleDrawer}:any) => {
 const [selectedCategory, setSelectedCategory]=useState("");
 return (
 <Box sx={{ width: 250 }} role="presentation"
 // onClick={toggleDrawer(false)}
 >
 <List>
 <ListItem>
 <ListItemButton>
 <ListItemText primary={<h1 className='logo text-2xl
text-[#00927c]'>Sajid Bazaar</h1>} />
 </ListItemButton>
 </ListItem>
 <Divider />
{mainCategory.map((item) => <ListItem key={item.name} disablePadding>

```

```
 <ListItemButton onClick={()=>setSelectedCategory(item.categoryId)}>
 <ListItemText primary={item.name} />
 </ListItemButton>
 </ListItem>
)}

</List>

{selectedCategory && <div
 // onMouseLeave={() => setShowSheet(false)}
 // onMouseEnter={() => setShowSheet(true)}
 className='categorySheet absolute top-[4.41rem] left-0 right-0
h-[400px]'>
 <CategorySheet toggleDrawer={toggleDrawer}
selectedCategory={selectedCategory}/>
</div>}

</Box>
)
}

export default DrawerList
.logo{
 font-family: "Pacifico", cursive;
 font-weight: 400;
 font-style: normal;
}
.category{
 font-weight: 400;
}

.blur-bg {
 backdrop-filter: blur(5px);
 -webkit-backdrop-filter: blur(10px); /* For Safari */
}

import {
Avatar,
Badge,
Box,
Button,
Drawer,
IconButton,
useMediaQuery,
useTheme,
} from "@mui/material";
import React, { useEffect, useState } from "react";
import "./Navbar.css";
```

```
import AddShoppingCartIcon from "@mui/icons-material/AddShoppingCart";
import StorefrontIcon from "@mui/icons-material/Storefront";
import SearchIcon from "@mui/icons-material/Search";
import MenuIcon from "@mui/icons-material/Menu";
import { mainCategory } from "../../../../../data/category/mainCategory";
import CategorySheet from "./CategorySheet";
import DrawerList from "./DrawerList";
import { useNavigate, useSearchParams } from "react-router-dom";
import AccountCircleIcon from "@mui/icons-material/AccountCircle";
import { useDispatch, useSelector } from "../../../../../Redux Toolkit/Store";
import { FavoriteBorder } from "@mui/icons-material";

const Navbar = () => {
 const [showSheet, setShowSheet] = useState(false);
 const [selectedCategory, setSelectedCategory] = useState("men");
 const theme = useTheme();
 const isLarge = useMediaQuery(theme.breakpoints.up("lg"));
 const dispatch = useDispatch();
 const { user, auth, cart, sellers } = useSelector((store) => store);
 const navigate = useNavigate();

 const [open, setOpen] = React.useState(false);

 const toggleDrawer = (newOpen: boolean) => () => {
 setOpen(newOpen);
 };

 const becomeSellerClick = () => {
 if (sellers.profile?.id) {
 navigate("/seller")
 } else navigate("/become-seller")
 }

 return (
 <Box
 sx={{ zIndex: 2 }}
 className="sticky top-0 left-0 right-0 bg-white blur-bg bg-opacity-80"
 >
 <div className="flex items-center justify-between px-5 lg:px-20 h-[70px] border-b">
 <div className="flex items-center gap-9">
 <div className="flex items-center gap-2">
 {!isLarge && (
 <SearchIcon />
 <input type="text" placeholder="Search" />
 <AddShoppingCartIcon />
)}
 <div>
 <StorefrontIcon />
 <CategorySheet />
 </div>
 </div>
 <div>
 <MenuIcon />
 <DrawerList />
 </div>
 </div>
 <div>
 <AccountCircleIcon />
 <UserIcon />
 </div>
 </div>

```

```

 <IconButton onClick={() => toggleDrawer(true)} />
 <MenuIcon className="text-gray-700" sx={{ fontSize: 29 }} />
 </IconButton>
) }
<h1
 onClick={() => navigate("/") }
 className="logo cursor-pointer text-lg md:text-2xl
text-[#00927c]"
>
 Sajid Bazaar
</h1>
</div>

{isLarge && (
<ul
 className="flex items-center font-medium text-gray-800 "
>
 {mainCategory.map((item) => (
<li
 onMouseLeave={() => {
 // setSelectedCategory("")
 setShowSheet(false);
 } }
 onMouseEnter={() => {
 setSelectedCategory(item.categoryId);
 setShowSheet(true);
 } }
 className="mainCategory hover:text-[#00927c] cursor-pointer
hover:border-b-2 h-[70px] px-4 border-[#00927c] flex items-center"
>
 {item.name}

)) }

) }
</div>

<div className="flex gap-1 lg:gap-6 items-center">
 <IconButton onClick={()=>navigate("/search-products")}>
 <SearchIcon className="text-gray-700" sx={{ fontSize: 29 }} />
 </IconButton>

{user.user ? (
<Button
 onClick={() => navigate("/account/orders") }
 className="flex items-center gap-2"
>
 <Avatar

```

```

 sx={{ width: 29, height: 29 }}

src="https://cdn.pixabay.com/photo/2015/04/15/09/28/head-723540_640.jpg"
 //
src="https://www.tanishq.co.in/dw/image/v2/BKCK_PRD/on/demandware.static/-/Library-Sites-TanishqSharedLibrary/default/dwc0abe627/homepage/ShopByGender/Woman.j
pg"
 /
 <h1 className="font-semibold hidden lg:block">
 {user.user?.fullName?.split(" ")[0]}
 </h1>
 </Button>
) : (
 <Button
 variant="contained"
 startIcon={<AccountCircleIcon sx={{ fontSize: "12px" }} />}
 onClick={() => navigate("/login")}
 >
 Login
 </Button>
) }

<IconButton onClick={()=>navigate("/wishlist")}>
 <FavoriteBorder sx={{ fontSize: 29 }}
 className="text-gray-700" />
</IconButton>

<IconButton onClick={() => navigate("/cart")}>
 <Badge badgeContent={cart.cart?.cartItems.length} color="primary">
 <AddShoppingCartIcon
 sx={{ fontSize: 29 }}
 className="text-gray-700"
 />
 </Badge>
</IconButton>

{isLarge && (
 <Button
 onClick={becomeSellerClick}
 startIcon={<StorefrontIcon />}
 variant="outlined"
 >
 Become Seller
 </Button>
)
</div>
</div>
<Drawer open={open} onClose={toggleDrawer(false)}>
 {<DrawerList toggleDrawer={toggleDrawer} />}

```

```

 </Drawer>
 {showSheet && selectedCategory && (
 <div
 onMouseLeave={() => setShowSheet(false)}
 onMouseEnter={() => setShowSheet(true)}
 className="categorySheet absolute top-[4.41rem] left-20 right-20"
 >
 <CategorySheet
 setShowSheet={setShowSheet}
 selectedCategory={selectedCategory}
 />
 </div>
)}
 </Box>
);
}

export default Navbar;
// NOTE: Users of the `experimental` builds of React should add a reference
// to 'react/experimental' in their project. See experimental.d.ts's top
comment
// for reference and documentation on how exactly to do it.

/// <reference path="global.d.ts" />

import * as CSS from "csstype";

type NativeAnimationEvent = AnimationEvent;
type NativeClipboardEvent = ClipboardEvent;
type NativeCompositionEvent = CompositionEvent;
type NativeDragEvent = DragEvent;
type NativeFocusEvent = FocusEvent;
type NativeInputEvent = InputEvent;
type NativeKeyboardEvent = KeyboardEvent;
type NativeMouseEvent = MouseEvent;
type NativeTouchEvent = TouchEvent;
type NativePointerEvent = PointerEvent;
type NativeToggleEvent = ToggleEvent;
type NativeTransitionEvent = TransitionEvent;
type NativeUIEvent = UIEvent;
type NativeWheelEvent = WheelEvent;

/**
 * Used to represent DOM API's where users can either pass
 * true or false as a boolean or as its equivalent strings.
 */
type Booleanish = boolean | "true" | "false";

/**

```

```

* @see {@link
https://developer.mozilla.org/en-US/docs/Web/HTML/Attributes/crossorigin MDN}
*/
type CrossOrigin = "anonymous" | "use-credentials" | "" | undefined;

declare const UNDEFINED_VOID_ONLY: unique symbol;

/**
* @internal Use `Awaited<ReactNode>` instead
*/
// Helper type to enable `Awaited<ReactNode>`.
// Must be a copy of the non-thenables of `ReactNode`.
type AwaitedReactNode =
 | React.ReactElement
 | string
 | number
 | bigint
 | Iterable<React.ReactNode>
 | React.ReactPortal
 | boolean
 | null
 | undefined
 | React.DO_NOT_USE_OR_YOU_WILL_BE_FIRED_EXPERIMENTAL_REACT_NODES[
 keyof React.DO_NOT_USE_OR_YOU_WILL_BE_FIRED_EXPERIMENTAL_REACT_NODES
];

/**
* The function returned from an effect passed to {@link React.useEffect useEffect},
* which can be used to clean up the effect when the component unmounts.
*
* @see {@link https://react.dev/reference/react/useEffect React Docs}
*/
type Destructor = () => void | { [UNDEFINED_VOID_ONLY]: never };
type VoidOrUndefinedOnly = void | { [UNDEFINED_VOID_ONLY]: never };

// eslint-disable-next-line @definitelytyped/export-just-namespace
export = React;
export as namespace React;

declare namespace React {
 //
 // React Elements
 // -----
 //

 /**
 * Used to retrieve the possible components which accept a given set of
 * props.
 *

```

```

 * Can be passed no type parameters to get a union of all possible
components
 * and tags.
 *
 * Is a superset of {@link ComponentType}.
 *
 * @template P The props to match against. If not passed, defaults to any.
 * @template Tag An optional tag to match against. If not passed, attempts
to match against all possible tags.
 *
 * @example
 *
 * ````tsx
 * // All components and tags (img, embed etc.)
 * // which accept `src`
 * type SrcComponents = ElementType<{ src: any }>;
 * ````

 *
 * @example
 *
 * ````tsx
 * // All components
 * type AllComponents = ElementType<{}>;
 * ````

 *
 * @example
 *
 * ````tsx
 * // All custom components which match `src`, and tags which
 * // match `src`, narrowed down to just `audio` and `embed`
 * type SrcComponents = ElementType<{ src: any }, 'audio' | 'embed'>;
 * ````

 */
type ElementType<P = any, Tag extends keyof JSX.IntrinsicElements = keyof
JSX.IntrinsicElements> =
 | { [K in Tag]: P extends JSX.IntrinsicElements[K] ? K : never }[Tag]
 | ComponentType<P>;

/**
 * Represents any user-defined component, either as a function or a class.
 *
 * Similar to {@link JSXElementConstructor}, but with extra properties like
 * {@link FunctionComponent.defaultProps defaultProps }.
 *
 * @template P The props the component accepts.
 *
 * @see {@link ComponentClass}
 * @see {@link FunctionComponent}
 */

```

```

type ComponentType<P = {}> = ComponentClass<P> | FunctionComponent<P>;

/**
 * Represents any user-defined component, either as a function or a class.
 *
 * Similar to {@link ComponentType}, but without extra properties like
 * {@link FunctionComponent.defaultProps defaultProps}.
 *
 * @template P The props the component accepts.
 */
type JSXElementConstructor<P> =
 | ((
 props: P,
) => ReactNode | Promise<ReactNode>)
 // constructor signature must match React.Component
 | (new(props: P, context: any) => Component<any, any>);

/**
 * Created by {@link createRef}, or {@link useRef} when passed `null`.
 *
 * @template T The type of the ref's value.
 *
 * @example
 *
 * ```ts
 * const ref = createRef<HTMLDivElement>();
 *
 * ref.current = document.createElement('div'); // Error
 * ```
 */
interface RefObject<T> {
 /**
 * The current value of the ref.
 */
 current: T;
}

interface DO_NOT_USE_OR_YOU_WILL_BE_FIRED_CALLBACK_REF_RETURN_VALUES {
}

/**
 * A callback fired whenever the ref's value changes.
 *
 * @template T The type of the ref's value.
 *
 * @see {@link
https://react.dev/reference/react-dom/components/common#ref-callback React
Docs}
 *
 * @example

```

```

/*
 * `` `tsx
 * <div ref={(node) => console.log(node)} />
 * ` ` `

*/
type RefCallback<T> = {
 bivarianceHack(
 instance: T | null,
): {
 | void
 | () => VoidOrUndefinedOnly
 | DO_NOT_USE_OR_YOU_WILL_BE_FIRED_CALLBACK_REF_RETURN_VALUES[
 keyof DO_NOT_USE_OR_YOU_WILL_BE_FIRED_CALLBACK_REF_RETURN_VALUES
];
 }["bivarianceHack"];
}

/***
 * A union type of all possible shapes for React refs.
 *
 * @see {link RefCallback}
 * @see {link RefObject}
 */
type Ref<T> = RefCallback<T> | RefObject<T | null> | null;
/***
 * @deprecated Use `Ref` instead. String refs are no longer supported.
 * If you're typing a library with support for React versions with string
refs, use `RefAttributes<T>['ref']` instead.
 */
type LegacyRef<T> = Ref<T>;
/***
 * @deprecated Use `ComponentRef<T>` instead
 *
 * Retrieves the type of the 'ref' prop for a given component type or tag
name.
 *
 * @template C The component type.
 *
 * @example
 *
 * `` `tsx
 * type MyComponentRef = React.ElementRef<typeof MyComponent>;
 * ` ` `

 *
 * @example
 *
 * `` `tsx
 * type DivRef = React.ElementRef<'div'>;
 * ` ` `


```

```
/*
type ElementRef<
 C extends
 | ForwardRefExoticComponent<any>
 | { new(props: any, context: any): Component<any> }
 | ((props: any) => ReactNode)
 | keyof JSX.IntrinsicElements,
> = ComponentRef<C>;

type ComponentState = any;

/***
 * A value which uniquely identifies a node among items in an array.
 *
 * @see {@link https://react.dev/learn/rendering-lists#keeping-list-items-in-order-with-key React Docs}
 */
type Key = string | number | bigint;

/***
 * @internal The props any component can receive.
 * You don't have to add this type. All components automatically accept
these props.
 * ````tsx
 * const Component = () => <div />;
 * <Component key="one" />
 * `````
 *
 * WARNING: The implementation of a component will never have access to
these attributes.
 * The following example would be incorrect usage because {@link Component} would
never have access to `key`:
 * ````tsx
 * const Component = (props: React.Attributes) => props.key;
 * `````
 */
interface Attributes {
 key?: Key | null | undefined;
}
/***
 * The props any component accepting refs can receive.
 * Class components, built-in browser components (e.g. `div`) and
forwardRef components can receive refs and automatically accept these props.
 * ````tsx
 * const Component = forwardRef(() => <div />);
 * <Component ref={(current) => console.log(current)} />
 * `````
 */
```

```

 * You only need this type if you manually author the types of props that
need to be compatible with legacy refs.
 * ````tsx
 * interface Props extends React.RefAttributes<HTMLDivElement> {}
 * declare const Component: React.FunctionComponent<Props>;
 * ``
 *
 * Otherwise it's simpler to directly use {@link Ref} since you can safely
use the
 * props type to describe to props that a consumer can pass to the
component
 * as well as describing the props the implementation of a component
"sees".
 * {@link RefAttributes} is generally not safe to describe both consumer
and seen props.
 *
 * ````tsx
 * interface Props extends {
 * ref?: React.Ref<HTMLDivElement> | undefined;
 * }
 * declare const Component: React.FunctionComponent<Props>;
 * ``
 *
 * WARNING: The implementation of a component will not have access to the
same type in versions of React supporting string refs.
 * The following example would be incorrect usage because {@link Component}
would never have access to a `ref` with type `string`
 * ````tsx
 * const Component = (props: React.RefAttributes) => props.ref;
 * ``
 */
interface RefAttributes<T> extends Attributes {
 /**
 * Allows getting a ref to the component instance.
 * Once the component unmounts, React will set `ref.current` to `null`.
 * (or call the ref with `null` if you passed a callback ref).
 *
 * @see {@link
https://react.dev/learn/referencing-values-with-refs#refs-and-the-dom React
Docs}
 */
 ref?: Ref<T> | undefined;
}

/**
 * Represents the built-in attributes available to class components.
*/
interface ClassAttributes<T> extends RefAttributes<T> {
}

```

```
/**
 * Represents a JSX element.
 *
 * Where {@link ReactNode} represents everything that can be rendered,
`ReactElement`
* only represents JSX.
*
* @template P The type of the props object
* @template T The type of the component or tag
*
* @example
*
* ``tsx
* const element: ReactElement = <div />;
* ``
*/
interface ReactElement<
 P = unknown,
 T extends string | JSXElementConstructor<any> = string |
JSXElementConstructor<any>,
> {
 type: T;
 props: P;
 key: string | null;
}

/**
 * @deprecated
 */
interface ReactComponentElement<
 T extends keyof JSX.IntrinsicElements | JSXElementConstructor<any>,
 P = Pick<ComponentProps<T>, Exclude<keyof ComponentProps<T>, "key" |
"ref">>,
> extends ReactElement<P, Exclude<T, number>> {}

/**
 * @deprecated Use `ReactElement<P, React.FunctionComponent<P>>`
 */
interface FunctionComponentElement<P> extends ReactElement<P,
FunctionComponent<P>> {
 /**
 * @deprecated Use `element.props.ref` instead.
 */
 ref?: ("ref" extends keyof P ? P extends { ref?: infer R | undefined } ?
R : never : never) | undefined;
}

/**
```

```
 * @deprecated Use `ReactElement<P, React.ComponentClass<P>>`
 */
type CElement<P, T extends Component<P, ComponentState>> =
ComponentElement<P, T>;
/**
 * @deprecated Use `ReactElement<P, React.ComponentClass<P>>`
 */
interface ComponentElement<P, T extends Component<P, ComponentState>>
extends ReactElement<P, ComponentClass<P>> {
 /**
 * @deprecated Use `element.props.ref` instead.
 */
 ref?: Ref<T> | undefined;
}

/**
 * @deprecated Use {@link ComponentElement} instead.
 */
type ClassicElement<P> = CElement<P, ClassicComponent<P, ComponentState>>;

// string fallback for custom web-components
/**
 * @deprecated Use `ReactElement<P, string>`
 */
interface DOMElement<P extends HTMLAttributes<T> | SVGAttributes<T>, T
extends Element>
 extends ReactElement<P, string>
{
 /**
 * @deprecated Use `element.props.ref` instead.
 */
 ref: Ref<T>;
}

// ReactHTML for ReactHTMLElement
interface ReactHTMLElement<T extends HTMLElement> extends
DetailedReactHTMLElement<AllHTMLAttributes<T>, T> {}

interface DetailedReactHTMLElement<P extends HTMLAttributes<T>, T extends
HTMLElement> extends DOMElement<P, T> {
 type: HTMLElementType;
}

// ReactSVG for ReactSVGELEMENT
interface ReactSVGELEMENT extends DOMELEMENT<SVGAttributes<SVGELEMENT>,
SVGELEMENT> {
 type: SVGELEMENTType;
}
```

```
interface ReactPortal extends ReactElement {
 children: ReactNode;
}

/**
 * Different release channels declare additional types of ReactNode this
particular release channel accepts.
 * App or library types should never augment this interface.
*/
interface DO_NOT_USE_OR_YOU_WILL_BE_FIRED_EXPERIMENTAL_REACT_NODES {}

/**
 * Represents all of the things React can render.
 *
 * Where {@link ReactElement} only represents JSX, `ReactNode` represents
everything that can be rendered.
 *
 * @see {@link https://react-typescript-cheatsheet.netlify.app/docs/react-types/reactnode/|React TypeScript Cheatsheet}
 *
 * @example
 *
 * ````tsx
 * // Typing children
 * type Props = { children: ReactNode }
 *
 * const Component = ({ children }: Props) => <div>{children}</div>
 *
 * <Component>hello</Component>
 * ````

 *
 * @example
 *
 * ````tsx
 * // Typing a custom element
 * type Props = { customElement: ReactNode }
 *
 * const Component = ({ customElement }: Props) =>
<div>{customElement}</div>
 *
 * <Component customElement={<div>hello</div>} />
 * ````
 */
// non-thenables need to be kept in sync with AwaitedReactNode
type ReactNode =
 | ReactElement
 | string
 | number
```

```
| bigint
| Iterable<ReactNode>
| ReactPortal
| boolean
| null
| undefined
| DO_NOT_USE_OR_YOU_WILL_BE_FIRED_EXPERIMENTAL_REACT_NODES [
 keyof DO_NOT_USE_OR_YOU_WILL_BE_FIRED_EXPERIMENTAL_REACT_NODES
]
| Promise<AwaitedReactNode>;
```

//  
// Top Level API  
// -----

```
// DOM Elements
// TODO: generalize this to everything in `keyof ReactHTML`, not just
"input"
function createElement(
 type: "input",
 props?: InputHTMLAttributes<HTMLInputElement> &
ClassAttributes<HTMLInputElement> | null,
 ...children: ReactNode[]
): DetailedReactHTMLElement<InputHTMLAttributes<HTMLInputElement>,
HTMLInputElement>;
function createElement<P extends HTMLAttributes<T>, T extends HTMLElement>(
 type: HTMLElementType,
 props?: ClassAttributes<T> & P | null,
 ...children: ReactNode[]
): DetailedReactHTMLElement<P, T>;
function createElement<P extends SVGAttributes<T>, T extends SVGELEMENT>(
 type: SVGELEMENTType,
 props?: ClassAttributes<T> & P | null,
 ...children: ReactNode[]
): ReactSVGELEMENT;
function createElement<P extends DOMAttributes<T>, T extends Element>(
 type: string,
 props?: ClassAttributes<T> & P | null,
 ...children: ReactNode[]
): DOMElement<P, T>;
// Custom components
```

```
function createElement<P extends {}>(
 type: FunctionComponent<P>,
 props?: Attributes & P | null,
 ...children: ReactNode[]
): FunctionComponentElement<P>;
```

```
function createElement<P extends {}, T extends Component<P, ComponentState>, C extends ComponentClass<P>>(
 type: ClassType<P, T, C>,
 props?: ClassAttributes<T> & P | null,
 ...children: ReactNode[]
): CElement<P, T>;
function createElement<P extends {}>(
 type: FunctionComponent<P> | ComponentClass<P> | string,
 props?: Attributes & P | null,
 ...children: ReactNode[]
): ReactElement<P>;

// DOM Elements
// ReactHTMLElement
function cloneElement<P extends HTMLAttributes<T>, T extends HTMLElement>(
 element: DetailedReactHTMLElement<P, T>,
 props?: P,
 ...children: ReactNode[]
): DetailedReactHTMLElement<P, T>;
// ReactHTMLElement, less specific
function cloneElement<P extends HTMLAttributes<T>, T extends HTMLElement>(
 element: ReactHTMLElement<T>,
 props?: P,
 ...children: ReactNode[]
): ReactHTMLElement<T>;
// SVGELEMENT
function cloneElement<P extends SVGAttributes<T>, T extends SVGELEMENT>(
 element: ReactSVGELEMENT,
 props?: P,
 ...children: ReactNode[]
): ReactSVGELEMENT;
// DOM Element (has to be the last, because type checking stops at first
overload that fits)
function cloneElement<P extends DOMAttributes<T>, T extends Element>(
 element: DOMELEMENT<P, T>,
 props?: DOMAttributes<T> & P,
 ...children: ReactNode[]
): DOMELEMENT<P, T>;

// Custom components
function cloneElement<P>(
 element: FunctionComponentElement<P>,
 props?: Partial<P> & Attributes,
 ...children: ReactNode[]
): FunctionComponentElement<P>;
function cloneElement<P, T extends Component<P, ComponentState>>(
 element: CElement<P, T>,
 props?: Partial<P> & ClassAttributes<T>,
 ...children: ReactNode[]
```

```
: CElement<P, T>;
function cloneElement<P>(
 element: ReactElement<P>,
 props?: Partial<P> & Attributes,
 ...children: ReactNode[]
): ReactElement<P>;

/**
 * Describes the props accepted by a Context {@link Provider}.
 *
 * @template T The type of the value the context provides.
 */
interface ProviderProps<T> {
 value: T;
 children?: ReactNode | undefined;
}

/**
 * Describes the props accepted by a Context {@link Consumer}.
 *
 * @template T The type of the value the context provides.
 */
interface ConsumerProps<T> {
 children: (value: T) => ReactNode;
}

/**
 * An object masquerading as a component. These are created by functions
 * like {@link forwardRef}, {@link memo}, and {@link createContext}.
 *
 * In order to make TypeScript work, we pretend that they are normal
 * components.
 *
 * But they are, in fact, not callable - instead, they are objects which
 * are treated specially by the renderer.
 *
 * @template P The props the component accepts.
 */
interface ExoticComponent<P = {}> {
 (props: P): ReactNode;
 readonly $$typeof: symbol;
}

/**
 * An {@link ExoticComponent} with a `displayName` property applied to it.
 *
 * @template P The props the component accepts.
 */
interface NamedExoticComponent<P = {}> extends ExoticComponent<P> {
```

```
 /**
 * Used in debugging messages. You might want to set it
 * explicitly if you want to display a different name for
 * debugging purposes.
 *
 * @see {@link
https://legacy.reactjs.org/docs/react-component.html#displayname Legacy React
Docs}
 */
 displayName?: string | undefined;
}

/**
 * An {@link ExoticComponent} with a `propTypes` property applied to it.
 *
 * @template P The props the component accepts.
 */
interface ProviderExoticComponent<P> extends ExoticComponent<P> {
}

/**
 * Used to retrieve the type of a context object from a {@link Context}.
 *
 * @template C The context object.
 *
 * @example
 *
 * ```tsx
 * import { createContext } from 'react';
 *
 * const MyContext = createContext({ foo: 'bar' });
 *
 * type ContextType = ContextType<typeof MyContext>;
 * // ContextType = { foo: string }
 * ```
 */
type ContextType<C extends Context<any>> = C extends Context<infer T> ? T : never;

/**
 * Wraps your components to specify the value of this context for all
components inside.
*
* @see {@link https://react.dev/reference/react createContext#provider
React Docs}
*
* @example
*
* ```tsx
```

```

* import { createContext } from 'react';
*
* const ThemeContext = createContext('light');
*
* function App() {
* return (
* <ThemeContext.Provider value="dark">
* <Toolbar />
* </ThemeContext.Provider>
*);
* }
* ````
*/
type Provider<T> = ProviderExoticComponent<ProviderProps<T>>;

/**
 * The old way to read context, before {@link useContext} existed.
 *
 * @see {@link https://react.dev/reference/react/createContext#consumer}
React Docs}
*
* @example
*
* ```.tsx
* import { UserContext } from './user-context';
*
* function Avatar() {
* return (
* <UserContext.Consumer>
* {user => }
* </UserContext.Consumer>
*);
* }
* ````
*/
type Consumer<T> = ExoticComponent<ConsumerProps<T>>;

/**
 * Context lets components pass information deep down without explicitly
* passing props.
*
* Created from {@link createContext}
*
* @see {@link https://react.dev/learn/passing-data-deeply-with-context}
React Docs}
* @see {@link
https://react-typescript-cheatsheet.netlify.app/docs/basic/getting-started/context/ React TypeScript Cheatsheet}
*

```

```

* @example
*
* ````tsx
* import { createContext } from 'react';
*
* const ThemeContext = createContext('light');
* ``
*/
interface Context<T> extends Provider<T> {
 Provider: Provider<T>;
 Consumer: Consumer<T>;
 /**
 * Used in debugging messages. You might want to set it
 * explicitly if you want to display a different name for
 * debugging purposes.
 *
 * @see {@link
https://legacy.reactjs.org/docs/react-component.html#displayname Legacy React
Docs}
 */
 displayName?: string | undefined;
}

/**
 * Lets you create a {@link Context} that components can provide or read.
 *
 * @param defaultValue The value you want the context to have when there is
no matching
 * {@link Provider} in the tree above the component reading the context.
This is meant
 * as a "last resort" fallback.
 *
 * @see {@link https://react.dev/reference/react/createContext#reference
React Docs}
 * @see {@link
https://react-typescript-cheatsheet.netlify.app/docs/basic/getting-started/context/ React TypeScript CheatSheet}
 *
* @example
*
* ````tsx
* import { createContext } from 'react';
*
* const ThemeContext = createContext('light');
* function App() {
* return (
* <ThemeContext value="dark">
* <Toolbar />
* </ThemeContext>

```

```
*);
* }
* ` ```
*/
function createContext<T>(
 // If you thought this should be optional, see
 //
https://github.com/DefinitelyTyped/DefinitelyTyped/pull/24509#issuecomment-382213106
 defaultValue: T,
): Context<T>;

function isValidElement<P>(object: {} | null | undefined): object is ReactElement<P>;
```

```
const Children: {
 map<T, C>(
 children: C | readonly C[],
 fn: (child: C, index: number) => T,
): C extends null | undefined ? C : Array<Exclude<T, boolean | null | undefined>>;
 forEach<C>(children: C | readonly C[], fn: (child: C, index: number) => void): void;
 count(children: any): number;
 only<C>(children: C): C extends any[] ? never : C;
 toArray(children: ReactNode | ReactNode[]): Array<Exclude<ReactNode, boolean | null | undefined>>;
};

export interface FragmentProps {
 children?: React.ReactNode;
}
/***
 * Lets you group elements without a wrapper node.
 *
 * @see {@link https://react.dev/reference/react/Fragment React Docs}
 *
 * @example
 *
 * ````tsx
 * import { Fragment } from 'react';
 *
 * <Fragment>
 * <td>Hello</td>
 * <td>World</td>
 * </Fragment>
 * `````
 *
 * @example
 */
function createFragment(): FragmentProps {
 return {
 children: null,
 };
}
```

```
/*
* ````tsx
* // Using the <></> shorthand syntax:
*
* <>
* <td>Hello</td>
* <td>World</td>
* </>
* ````
*/
const Fragment: ExoticComponent<FragmentProps>;

/**
* Lets you find common bugs in your components early during development.
*
* @see {https://react.dev/reference/react/StrictMode React Docs}
*
* @example
*
* ````tsx
* import { StrictMode } from 'react';
*
* <StrictMode>
* <App />
* </StrictMode>
* ````
*/
const StrictMode: ExoticComponent<{ children?: ReactNode | undefined }>;

/**
* The props accepted by {link Suspense}.
*
* @see {https://react.dev/reference/react/Suspense React Docs}
*
interface SuspenseProps {
 children?: ReactNode | undefined;

 /** A fallback react tree to show when a Suspense child (like
React.lazy) suspends */
 fallback?: ReactNode;

 /**
 * A name for this Suspense boundary for instrumentation purposes.
 * The name will help identify this boundary in React DevTools.
 */
 name?: string | undefined;
}

/**
```

```
* Lets you display a fallback until its children have finished loading.
*
* @see {@link https://react.dev/reference/react/Suspense React Docs}
*
* @example
*
* ````tsx
* import { Suspense } from 'react';
*
* <Suspense fallback=<Loading />>
* <ProfileDetails />
* </Suspense>
* ```
*
* /
const Suspense: ExoticComponent<SuspenseProps>;
const version: string;

/**
 * The callback passed to {@link ProfilerProps.onRender}.
 *
 * @see {@link https://react.dev/reference/react/Profiler#onrender-callback React Docs}
 */
type ProfilerOnRenderCallback = (
 /**
 * The string id prop of the {@link Profiler} tree that has just committed. This lets
 * you identify which part of the tree was committed if you are using multiple
 * profilers.
 *
 * @see {@link https://react.dev/reference/react/Profiler#onrender-callback React Docs}
 */
 id: string,
 /**
 * This lets you know whether the tree has just been mounted for the first time
 * or re-rendered due to a change in props, state, or hooks.
 *
 * @see {@link https://react.dev/reference/react/Profiler#onrender-callback React Docs}
 */
 phase: "mount" | "update" | "nested-update",
 /**
 * The number of milliseconds spent rendering the {@link Profiler} and its descendants
 * for the current update. This indicates how well the subtree makes use of

```

```
 * memoization (e.g. {@link memo} and {@link useMemo}). Ideally this
value should decrease
 * significantly after the initial mount as many of the descendants
will only need to
 * re-render if their specific props change.
 *
 * @see {@link
https://react.dev/reference/react/Profiler#onrender-callback React Docs}
 */
actualDuration: number,
/**
 * The number of milliseconds estimating how much time it would take to
re-render the entire
 * {@link Profiler} subtree without any optimizations. It is calculated
by summing up the most
 * recent render durations of each component in the tree. This value
estimates a worst-case
 * cost of rendering (e.g. the initial mount or a tree with no
memoization). Compare
 * {@link actualDuration} against it to see if memoization is working.
 *
 * @see {@link
https://react.dev/reference/react/Profiler#onrender-callback React Docs}
 */
baseDuration: number,
/**
 * A numeric timestamp for when React began rendering the current
update.
 *
 * @see {@link
https://react.dev/reference/react/Profiler#onrender-callback React Docs}
 */
startTime: number,
/**
 * A numeric timestamp for when React committed the current update.
This value is shared
 * between all profilers in a commit, enabling them to be grouped if
desirable.
 *
 * @see {@link
https://react.dev/reference/react/Profiler#onrender-callback React Docs}
 */
commitTime: number,
) => void;

/**
 * The props accepted by {@link Profiler}.
*
* @see {@link https://react.dev/reference/react/Profiler React Docs}
```

```
 */
interface ProfilerProps {
 children?: ReactNode | undefined;
 id: string;
 onRender: ProfilerOnRenderCallback;
}

/**
 * Lets you measure rendering performance of a React tree programmatically.
 *
 * @see {@link https://react.dev/reference/react/Profiler#onrender-callback}
 * @React Docs
 *
 * @example
 *
 * ````ts
 * <Profiler id="App" onRender={onRender}>
 * <App />
 * </Profiler>
 * ````
 */
const Profiler: ExoticComponent<ProfilerProps>;

// -----
// Component API
// -----

type ReactInstance = Component<any> | Element;

// Base component for plain JS classes
interface Component<P = {}, S = {}, SS = any> extends ComponentLifecycle<P, S, SS> {}
class Component<P, S> {
 /**
 * If set, `this.context` will be set at runtime to the current value
 * of the given Context.
 *
 * @example
 *
 * ````ts
 * type MyContext = number
 * const Ctx = React.createContext<MyContext>()
 *
 * class Foo extends React.Component {
 * static contextType = Ctx
 * context!: React.ContextType<typeof Ctx>
 * render () {
 * return <>My context's value: {this.context}</>;
 * }
 *
```

```

 * }
 * ` ```
 *
 * @see {@link
https://react.dev/reference/react/Component#static-contexttype}
 */
 static contextType?: Context<any> | undefined;

 /**
 * Ignored by React.
 * @deprecated Only kept in types for backwards compatibility. Will be
 removed in a future major release.
 */
 static propTypes?: any;

 /**
 * If using React Context, re-declare this in your class to be the
 * `React.ContextType` of your `static contextType`.
 * Should be used with type annotation or static contextType.
 *
 * @example
 * ``ts
 * static contextType = MyContext
 * // For TS pre-3.7:
 * context!: React.ContextType<typeof MyContext>
 * // For TS 3.7 and above:
 * declare context: React.ContextType<typeof MyContext>
 * ```
 *
 * @see {@link https://react.dev/reference/react/Component#context
React Docs}
 */
 context: unknown;

 // Keep in sync with constructor signature of JSElementConstructor and
 ComponentClass.
 constructor(props: P);
 /**
 * @param props
 * @param context value of the parent {@link


```

```

 // See:
 https://github.com/DefinitelyTyped/DefinitelyTyped/issues/18365#issuecomment-351013257
 // Also, the ` | S` allows intellisense to not be dumbisense
 setState<K extends keyof S>(
 state: ((prevState: Readonly<S>, props: Readonly<P>) => Pick<S, K> | S | null) | (Pick<S, K> | S | null),
 callback?: () => void,
): void;

 forceUpdate(callback?: () => void): void;
 render(): ReactNode;

 readonly props: Readonly<P>;
 state: Readonly<S>;
 }

 class PureComponent<P = {}, S = {}, SS = any> extends Component<P, S, SS> {}

 /**
 * @deprecated Use `ClassicComponent` from `create-react-class`
 *
 * @see {@link https://legacy.reactjs.org/docs/react-without-es6.html Legacy React Docs}
 * @see {@link https://www.npmjs.com/package/create-react-class `create-react-class` on npm}
 */
 interface ClassicComponent<P = {}, S = {}> extends Component<P, S> {
 replaceState(nextState: S, callback?: () => void): void;
 isMounted(): boolean;
 getInitialState?(): S;
 }

 //
 // Class Interfaces
 // -----
 //

 /**
 * Represents the type of a function component. Can optionally receive a type argument that represents the props the component receives.
 *
 * @template P The props the component accepts.
 * @see {@link https://react-typescript-cheatsheet.netlify.app/docs/basic/getting-started/function_components React TypeScript Cheatsheet}
 * @alias for {@link FunctionComponent}
 *
 * @example

```

```

/*
* `` `tsx
* // With props:
* type Props = { name: string }
*
* const MyComponent: FC<Props> = (props) => {
* return <div>{props.name}</div>
* }
* `` `

*
* @example
*
* `` `tsx
* // Without props:
* const MyComponentWithoutProps: FC = () => {
* return <div>MyComponentWithoutProps</div>
* }
* `` `

*/
type FC<P = {}> = FunctionComponent<P>;
```

/\*\*  
*\* Represents the type of a function component. Can optionally  
\* receive a type argument that represents the props the component  
\* accepts.*  
*\*  
\* @template P The props the component accepts.  
\* @see {@link*  
[https://react-typescript-cheatsheet.netlify.app/docs/basic/getting-started/function\\_components](https://react-typescript-cheatsheet.netlify.app/docs/basic/getting-started/function_components) React TypeScript Cheatsheet}  
*\**

```

* @example
*
* `` `tsx
* // With props:
* type Props = { name: string }
*
* const MyComponent: FunctionComponent<Props> = (props) => {
* return <div>{props.name}</div>
* }
* `` `

*
* @example
*
* `` `tsx
* // Without props:
* const MyComponentWithoutProps: FunctionComponent = () => {
* return <div>MyComponentWithoutProps</div>
* }
```

```
* ``
*/
interface FunctionComponent<P = {}> {
 (props: P): ReactNode | Promise<ReactNode>;
 /**
 * Ignored by React.
 * @deprecated Only kept in types for backwards compatibility. Will be
 removed in a future major release.
 */
 propTypes?: any;
 /**
 * Used in debugging messages. You might want to set it
 * explicitly if you want to display a different name for
 * debugging purposes.
 *
 * @see {@link
https://legacy.reactjs.org/docs/react-component.html#displayname Legacy React
Docs}
 *
 * @example
 *
 * ````ts
 *
 * const MyComponent: FC = () => {
 * return <div>Hello!</div>
 * }
 *
 * MyComponent.displayName = 'MyAwesomeComponent'
 * ```
 */
 displayName?: string | undefined;
}

/**
 * The type of the ref received by a {@link ForwardRefRenderFunction}.
 *
 * @see {@link ForwardRefRenderFunction}
 */
// Making T nullable is assuming the refs will be managed by React or the
component impl will write it somewhere else.
// But this isn't necessarily true. We haven't heard complains about it yet
and hopefully `forwardRef` is removed from React before we do.
type ForwardedRef<T> = ((instance: T | null) => void) | RefObject<T | null>
| null;

/**
 * The type of the function passed to {@link forwardRef}. This is
considered different
```

```

 * to a normal {@link FunctionComponent} because it receives an additional
argument,
 *
 * @param props Props passed to the component, if any.
 * @param ref A ref forwarded to the component of type {@link
ForwardedRef}.
 *
 * @template T The type of the forwarded ref.
 * @template P The type of the props the component accepts.
 *
 * @see {@link
https://react-typescript-cheatsheet.netlify.app/docs/basic/getting-started/forward_and_create_ref/ React TypeScript Cheatsheet}
 * @see {@link forwardRef}
 */
interface ForwardRefRenderFunction<T, P = {}> {
 (props: P, ref: ForwardedRef<T>): ReactNode;
 /**
 * Used in debugging messages. You might want to set it
 * explicitly if you want to display a different name for
 * debugging purposes.
 *
 * Will show `ForwardRef(${Component.displayName || Component.name})`
 * in devtools by default, but can be given its own specific name.
 *
 * @see {@link
https://legacy.reactjs.org/docs/react-component.html#displayname Legacy React
Docs}
 */
 displayName?: string | undefined;
 /**
 * Ignored by React.
 * @deprecated Only kept in types for backwards compatibility. Will be
removed in a future major release.
 */
 propTypes?: any;
}

/**
 * Represents a component class in React.
 *
 * @template P The props the component accepts.
 * @template S The internal state of the component.
 */
interface ComponentClass<P = {}, S = ComponentState> extends
StaticLifecycle<P, S> {
 // constructor signature must match React.Component
 new(
 props: P,

```

```
 /**
 * Value of the parent {@link
https://react.dev/reference/react/Component#context_Context} specified
 * in `contextType`.
 */
 context?: any,
): Component<P, S>;
/***
 * Ignored by React.
 * @deprecated Only kept in types for backwards compatibility. Will be
removed in a future major release.
*/
propTypes?: any;
contextType?: Context<any> | undefined;
defaultProps?: Partial<P> | undefined;
/***
 * Used in debugging messages. You might want to set it
 * explicitly if you want to display a different name for
 * debugging purposes.
*
* @see {@link
https://legacy.reactjs.org/docs/react-component.html#displayname_Legacy React Docs
}
displayName?: string | undefined;
}

/***
 * @deprecated Use `ClassicComponentClass` from `create-react-class`
*
* @see {@link https://legacy.reactjs.org/docs/react-without-es6.html_Legacy React Docs}
* @see {@link https://www.npmjs.com/package/create-react-class
`create-react-class` on npm}
*/
interface ClassicComponentClass<P = {}> extends ComponentClass<P> {
 new(props: P): ClassicComponent<P, ComponentState>;
 getDefaultProps?(): P;
}

/***
 * Used in {@link createElement} and {@link createFactory} to represent
 * a class.
*
* An intersection type is used to infer multiple type parameters from
* a single argument, which is useful for many top-level API defs.
* See {@link https://github.com/Microsoft/TypeScript/issues/7234 this
GitHub issue}
* for more info.

```

```


/*
type ClassType<P, T extends Component<P, ComponentState>, C extends
ComponentClass<P>> =
 & C
 & (new(props: P, context: any) => T);

// -----
// Component Specs and Lifecycle
// -----

// This should actually be something like `Lifecycle<P, S> |
DeprecatedLifecycle<P, S>`,
// as React will _not_ call the deprecated lifecycle methods if any of the
new lifecycle
// methods are present.
interface ComponentLifecycle<P, S, SS = any> extends NewLifecycle<P, S, SS>,
DeprecatedLifecycle<P, S> {
 /**
 * Called immediately after a component is mounted. Setting state here
will trigger re-rendering.
 */
 componentDidMount?(): void;
 /**
 * Called to determine whether the change in props and state should
trigger a re-render.
 *
 * `Component` always returns true.
 * `PureComponent` implements a shallow comparison on props and state
and returns true if any
 * props or states have changed.
 *
 * If false is returned, {@link Component.render},
`componentWillUpdate`
 * and `componentDidUpdate` will not be called.
 */
 shouldComponentUpdate?(nextProps: Readonly<P>, nextState: Readonly<S>,
nextContext: any): boolean;
 /**
 * Called immediately before a component is destroyed. Perform any
necessary cleanup in this method, such as
 * cancelled network requests, or cleaning up any DOM elements created
in `componentDidMount`.
 */
 componentWillUnmount?(): void;
 /**
 * Catches exceptions generated in descendant components. Unhandled
exceptions will cause
 * the entire component tree to unmount.
 */
}


```

```

 componentDidCatch?(error: Error, errorInfo: ErrorInfo): void;
 }

 // Unfortunately, we have no way of declaring that the component constructor
 must implement this
 interface StaticLifecycle<P, S> {
 getDerivedStateFromProps?: GetDerivedStateFromProps<P, S> | undefined;
 getDerivedStateFromError?: GetDerivedStateFromError<P, S> | undefined;
 }

 type GetDerivedStateFromProps<P, S> =
 /**
 * Returns an update to a component's state based on its new props and
 old state.
 *
 * Note: its presence prevents any of the deprecated lifecycle methods
 from being invoked
 */
 (nextProps: Readonly<P>, prevState: S) => Partial<S> | null;

 type GetDerivedStateFromError<P, S> =
 /**
 * This lifecycle is invoked after an error has been thrown by a
 descendant component.
 * It receives the error that was thrown as a parameter and should
 return a value to update state.
 *
 * Note: its presence prevents any of the deprecated lifecycle methods
 from being invoked
 */
 (error: any) => Partial<S> | null;

 // This should be "infer SS" but can't use it yet
 interface NewLifecycle<P, S, SS> {
 /**
 * Runs before React applies the result of {@link Component.render
 render} to the document, and
 * returns an object to be given to {@link componentDidUpdate}. Useful
 for saving
 * things such as scroll position before {@link Component.render
 render} causes changes to it.
 *
 * Note: the presence of this method prevents any of the deprecated
 * lifecycle events from running.
 */
 getSnapshotBeforeUpdate?(prevProps: Readonly<P>, prevState:
 Readonly<S>): SS | null;
 /**

```

```
 * Called immediately after updating occurs. Not called for the initial
render.
 *
 * The snapshot is only present if {@link getSnapshotBeforeUpdate} is
present and returns non-null.
 */
 componentDidUpdate?(prevProps: Readonly<P>, prevState: Readonly<S>,
snapshot?: S): void;
}
```

```
interface DeprecatedLifecycle<P, S> {
 /**
 * Called immediately before mounting occurs, and before {@link
Component.render}.
 * Avoid introducing any side-effects or subscriptions in this method.
 *
 * Note: the presence of {@link NewLifecycle.getSnapshotBeforeUpdate
getSnapshotBeforeUpdate}
 * or {@link StaticLifecycle.getDerivedStateFromProps
getDerivedStateFromProps} prevents
 * this from being invoked.
 *
 * @deprecated 16.3, use {@link ComponentLifecycle.componentDidMount
componentDidMount} or the constructor instead; will stop working in React 17
 * @see {@link
https://legacy.reactjs.org/blog/2018/03/27/update-on-async-rendering.html#initializing-state}
 * @see {@link
https://legacy.reactjs.org/blog/2018/03/27/update-on-async-rendering.html#gradual-migration-path}
 */
 componentWillMount?(): void;
 /**
 * Called immediately before mounting occurs, and before {@link
Component.render}.
 * Avoid introducing any side-effects or subscriptions in this method.
 *
 * This method will not stop working in React 17.
 *
 * Note: the presence of {@link NewLifecycle.getSnapshotBeforeUpdate
getSnapshotBeforeUpdate}
 * or {@link StaticLifecycle.getDerivedStateFromProps
getDerivedStateFromProps} prevents
 * this from being invoked.
 *
 * @deprecated 16.3, use {@link ComponentLifecycle.componentDidMount
componentDidMount} or the constructor instead

```

```
 * @see {@link
https://legacy.reactjs.org/blog/2018/03/27/update-on-async-rendering.html#initializing-state}
 * @see {@link
https://legacy.reactjs.org/blog/2018/03/27/update-on-async-rendering.html#gradual-migration-path}
 */
UNSAFE_componentWillMount?(): void;
/***
 * Called when the component may be receiving new props.
 * React may call this even if props have not changed, so be sure to
compare new and existing
 * props if you only want to handle changes.
 *
 * Calling {@link Component.setState} generally does not trigger this
method.
 *
 * Note: the presence of {@link NewLifecycle.getSnapshotBeforeUpdate
getSnapshotBeforeUpdate}
 * or {@link StaticLifecycle.getDerivedStateFromProps
getDerivedStateFromProps} prevents
 * this from being invoked.
 *
 * @deprecated 16.3, use static {@link
StaticLifecycle.getDerivedStateFromProps getDerivedStateFromProps} instead;
will stop working in React 17
 * @see {@link
https://legacy.reactjs.org/blog/2018/03/27/update-on-async-rendering.html#updating-state-based-on-props}
 * @see {@link
https://legacy.reactjs.org/blog/2018/03/27/update-on-async-rendering.html#gradual-migration-path}
 */
componentWillReceiveProps?(nextProps: Readonly<P>, nextContext: any): void;
/***
 * Called when the component may be receiving new props.
 * React may call this even if props have not changed, so be sure to
compare new and existing
 * props if you only want to handle changes.
 *
 * Calling {@link Component.setState} generally does not trigger this
method.
 *
 * This method will not stop working in React 17.
 *
 * Note: the presence of {@link NewLifecycle.getSnapshotBeforeUpdate
getSnapshotBeforeUpdate}
```

```
 * or {@link StaticLifecycle.getDerivedStateFromProps
getDerivedStateFromProps} prevents
 * this from being invoked.
 *
 * @deprecated 16.3, use static {@link
StaticLifecycle.getDerivedStateFromProps getDerivedStateFromProps} instead
 * @see {@link
https://legacy.reactjs.org/blog/2018/03/27/update-on-async-rendering.html#updating-state-based-on-props
 * @see {@link
https://legacy.reactjs.org/blog/2018/03/27/update-on-async-rendering.html#gradual-migration-path
 */
UNSAFE_componentWillReceiveProps?(nextProps: Readonly<P>, nextContext: any): void;
/**
 * Called immediately before rendering when new props or state is
received. Not called for the initial render.
 *
 * Note: You cannot call {@link Component.setState} here.
 *
 * Note: the presence of {@link NewLifecycle.getSnapshotBeforeUpdate
getSnapshotBeforeUpdate}
 * or {@link StaticLifecycle.getDerivedStateFromProps
getDerivedStateFromProps} prevents
 * this from being invoked.
 *
 * @deprecated 16.3, use getSnapshotBeforeUpdate instead; will stop
working in React 17
 * @see {@link
https://legacy.reactjs.org/blog/2018/03/27/update-on-async-rendering.html#reading-dom-properties-before-an-update
 * @see {@link
https://legacy.reactjs.org/blog/2018/03/27/update-on-async-rendering.html#gradual-migration-path
 */
componentWillUpdate?(nextProps: Readonly<P>, nextState: Readonly<S>, nextContext: any): void;
/**
 * Called immediately before rendering when new props or state is
received. Not called for the initial render.
 *
 * Note: You cannot call {@link Component.setState} here.
 *
 * This method will not stop working in React 17.
 *
 * Note: the presence of {@link NewLifecycle.getSnapshotBeforeUpdate
getSnapshotBeforeUpdate}
```

```

 * or {@link StaticLifecycle.getDerivedStateFromProps
getDerivedStateFromProps} prevents
 * this from being invoked.
 *
 * @deprecated 16.3, use getSnapshotBeforeUpdate instead
 * @see {@link
https://legacy.reactjs.org/blog/2018/03/27/update-on-async-rendering.html#reading-dom-properties-before-an-update}
 * @see {@link
https://legacy.reactjs.org/blog/2018/03/27/update-on-async-rendering.html#gradual-migration-path}
 */
 UNSAFE_componentWillUpdate?(nextProps: Readonly<P>, nextState: Readonly<S>, nextContext: any): void;
}

function createRef<T>(): RefObject<T | null>;

/**
 * The type of the component returned from {@link forwardRef}.
 *
 * @template P The props the component accepts, if any.
 *
 * @see {@link ExoticComponent}
 */
interface ForwardRefExoticComponent<P> extends NamedExoticComponent<P> {
 /**
 * Ignored by React.
 * @deprecated Only kept in types for backwards compatibility. Will be removed in a future major release.
 */
 propTypes?: any;
}

/**
 * Lets your component expose a DOM node to a parent component using a ref.
 *
 * @see {@link https://react.dev/reference/react/forwardRef React Docs}
 * @see {@link
https://react-typescript-cheatsheet.netlify.app/docs/basic/getting-started/forward_and_create_ref/ React TypeScript Cheatsheet}
 *
 * @param render See the {@link ForwardRefRenderFunction}.
 *
 * @template T The type of the DOM node.
 * @template P The props the component accepts, if any.
 *
 * @example

```

```

/*
 * `` `tsx
 * interface Props {
 * children?: ReactNode;
 * type: "submit" | "button";
 * }
 *
 * export const FancyButton = forwardRef<HTMLButtonElement, Props>((props,
ref) => (
 <button ref={ref} className="MyClassName" type={props.type}>
 {props.children}
 </button>
)) ;
`` `
*/
function forwardRef<T, P = {}>(
 render: ForwardRefRenderFunction<T, PropsWithoutRef<P>>,
): ForwardRefExoticComponent<PropsWithoutRef<P> & RefAttributes<T>>;

/**
 * Omits the 'ref' attribute from the given props object.
 *
 * @template Props The props object type.
 */
type PropsWithoutRef<Props> =
 // Omit would not be sufficient for this. We'd like to avoid unnecessary
mapping and need a distributive conditional to support unions.
 // see:
https://www.typescriptlang.org/docs/handbook/2/conditional-types.html#distributive-conditional-types
 // https://github.com/Microsoft/TypeScript/issues/28339
 Props extends any ? ("ref" extends keyof Props ? Omit<Props, "ref"> :
Props) : Props;
/**
 * Ensures that the props do not include string ref, which cannot be
forwarded
 * @deprecated Use `Props` directly. `PropsWithRef<Props>` is just an alias
for `Props`
 */
type PropsWithRef<Props> = Props;

type PropsWithChildren<P = unknown> = P & { children?: ReactNode | undefined
};

/**
 * Used to retrieve the props a component accepts. Can either be passed a
string,
 * indicating a DOM element (e.g. 'div', 'span', etc.) or the type of a
React

```

```

* component.
*
* It's usually better to use {@link ComponentPropsWithRef} or {@link ComponentPropsWithoutRef}
* instead of this type, as they let you be explicit about whether or not
to include
* the `ref` prop.
*
* @see {@link
https://react-typescript-cheatsheet.netlify.app/docs/react-types/componentprops/ React TypeScript Cheatsheet}
*
* @example
*
* ````tsx
* // Retrieves the props an 'input' element accepts
* type InputProps = React.ComponentProps<'input'>;
* ````

*
* @example
*
* ````tsx
* const MyComponent = (props: { foo: number, bar: string }) => <div />;
*
* // Retrieves the props 'MyComponent' accepts
* type MyComponentProps = React.ComponentProps<typeof MyComponent>;
* ````

*/
type ComponentProps<T extends keyof JSX.IntrinsicElements | JSXElementConstructor<any>> = T extends
JSXElementConstructor<infer Props> ? Props
: T extends keyof JSX.IntrinsicElements ? JSX.IntrinsicElements[T]
: {};

/**
* Used to retrieve the props a component accepts with its ref. Can either
be
* passed a string, indicating a DOM element (e.g. 'div', 'span', etc.) or
the
* type of a React component.
*
* @see {@link
https://react-typescript-cheatsheet.netlify.app/docs/react-types/componentprops/ React TypeScript Cheatsheet}
*
* @example
*
* ````tsx
* // Retrieves the props an 'input' element accepts

```

```

* type InputProps = React.ComponentPropsWithRef<'input'>;
* ``
*
* @example
*
* ``tsx
* const MyComponent = (props: { foo: number, bar: string }) => <div />;
*
* // Retrieves the props 'MyComponent' accepts
* type MyComponentPropsWithRef = React.ComponentPropsWithRef<typeof
MyComponent>;
* ``
*/
type ComponentPropsWithRef<T extends ElementType> = T extends
JSXElementConstructor<infer Props>
 // If it's a class i.e. newable we're dealing with a class component
 ? T extends abstract new(args: any) => any ? PropsWithoutRef<Props> &
RefAttributes<InstanceType<T>>
 : Props
 : ComponentProps<T>;
/***
* Used to retrieve the props a custom component accepts with its ref.
*
* Unlike {@link ComponentPropsWithRef}, this only works with custom
* components, i.e. components you define yourself. This is to improve
* type-checking performance.
*
* @example
*
* ``tsx
* const MyComponent = (props: { foo: number, bar: string }) => <div />;
*
* // Retrieves the props 'MyComponent' accepts
* type MyComponentPropsWithRef = React.CustomComponentPropsWithRef<typeof
MyComponent>;
* ``
*/
type CustomComponentPropsWithRef<T extends ComponentType> = T extends
JSXElementConstructor<infer Props>
 // If it's a class i.e. newable we're dealing with a class component
 ? T extends abstract new(args: any) => any ? PropsWithoutRef<Props> &
RefAttributes<InstanceType<T>>
 : Props
 : never;
/***
* Used to retrieve the props a component accepts without its ref. Can
either be

```

```

 * passed a string, indicating a DOM element (e.g. 'div', 'span', etc.) or
the
 * type of a React component.
 *
 * @see {@link
https://react-typescript-cheatsheet.netlify.app/docs/react-types/componentprops/ React TypeScript Cheatsheet}
 *
 * @example
 *
 * ````tsx
 * // Retrieves the props an 'input' element accepts
 * type InputProps = React.ComponentPropsWithoutRef<'input'>;
 * ````

 *
 * @example
 *
 * ````tsx
 * const MyComponent = (props: { foo: number, bar: string }) => <div />;
 *
 * // Retrieves the props 'MyComponent' accepts
 * type MyComponentPropsWithoutRef = React.ComponentPropsWithoutRef<typeof
MyComponent>;
 * ````

 */
type ComponentPropsWithoutRef<T extends ElementType> =
PropsWithoutRef<ComponentProps<T>>;

/**
 * Retrieves the type of the 'ref' prop for a given component type or tag
name.
*
* @template C The component type.
*
* @example
*
* ````tsx
* type MyComponentRef = React.ComponentRef<typeof MyComponent>;
* ````

*
* @example
*
* ````tsx
* type DivRef = React.ComponentRef<'div'>;
* ````

*/
type ComponentRef<T extends ElementType> = ComponentPropsWithRef<T> extends
RefAttributes<infer Method> ? Method
 : never;

```

```
// will show `Memo(${Component.displayName || Component.name})` in devtools
by default,
// but can be given its own specific name
type MemoExoticComponent<T extends ComponentType<any>> =
NamedExoticComponent<CustomComponentPropsWithRef<T>> & {
 readonly type: T;
};

/**
 * Lets you skip re-rendering a component when its props are unchanged.
 *
 * @see {@link https://react.dev/reference/react/memo React Docs}
 *
 * @param Component The component to memoize.
 * @param propsAreEqual A function that will be used to determine if the
props have changed.
 *
 * @example
 *
 * ``tsx
 * import { memo } from 'react';
 *
 * const SomeComponent = memo(function SomeComponent(props: { foo: string
}) {
 // ...
);
 //
}
function memo<P extends object>(
 Component: FunctionComponent<P>,
 propsAreEqual?: (prevProps: Readonly<P>, nextProps: Readonly<P>) =>
boolean,
): NamedExoticComponent<P>;
function memo<T extends ComponentType<any>>(
 Component: T,
 propsAreEqual?: (prevProps: Readonly<ComponentProps<T>>, nextProps:
Readonly<ComponentProps<T>>) => boolean,
): MemoExoticComponent<T>;

interface LazyExoticComponent<T extends ComponentType<any>>
 extends ExoticComponent<CustomComponentPropsWithRef<T>>
{
 readonly _result: T;
}

/**
 * Lets you defer loading a component's code until it is rendered for the
first time.
```

```
*
* @see {@link https://react.dev/reference/react/lazy React Docs}
*
* @param load A function that returns a `Promise` or another thenable (a
`Promise`-like object with a
* then method). React will not call `load` until the first time you
attempt to render the returned
* component. After React first calls load, it will wait for it to resolve,
and then render the
* resolved value's `.default` as a React component. Both the returned
`Promise` and the `Promise`'s
* resolved value will be cached, so React will not call load more than
once. If the `Promise` rejects,
* React will throw the rejection reason for the nearest Error Boundary to
handle.
*
* @example
*
* ````tsx
* import { lazy } from 'react';
*
* const MarkdownPreview = lazy(() => import('./MarkdownPreview.js'));
* ````
*/
function lazy<T extends ComponentType<any>>(
 load: () => Promise<{ default: T }>,
): LazyExoticComponent<T>;
// -----
// React Hooks
// -----
/***
 * The instruction passed to a {@link Dispatch} function in {@link useState}
 * to tell React what the next value of the {@link useState} should be.
 *
 * Often found wrapped in {@link Dispatch}.
 *
 * @template S The type of the state.
 *
 * @example
 *
 * ````tsx
 * // This return type correctly represents the type of
 * // `setCount` in the example below.
 * const useCustomState = (): Dispatch<SetStateAction<number>> => {
 * const [count, setCount] = useState(0);
 *
```

```

 * return setCount;
 *
 */
 type SetStateAction<S> = S | ((prevState: S) => S);

 /**
 * A function that can be used to update the state of a {@link useState}
 * or {@link useReducer} hook.
 */
 type Dispatch<A> = (value: A) => void;
 /**
 * A {@link Dispatch} function can sometimes be called without any
 * arguments.
 */
 type DispatchWithoutAction = () => void;
 // Limit the reducer to accept only 0 or 1 action arguments
 // eslint-disable-next-line @definitelytyped/no-single-element-tuple-type
 type AnyActionArg = [] | [any];
 // Get the dispatch type from the reducer arguments (captures optional
 action argument correctly)
 type ActionDispatch<ActionArg extends AnyActionArg> = (...args: ActionArg)
=> void;
 // Unlike redux, the actions _can_ be anything
 type Reducer<S, A> = (prevState: S, action: A) => S;
 // If useReducer accepts a reducer without action, dispatch may be called
 without any parameters.
 type ReducerWithoutAction<S> = (prevState: S) => S;
 // types used to try and prevent the compiler from reducing S
 // to a supertype common with the second argument to useReducer()
 type ReducerState<R extends Reducer<any, any>> = R extends Reducer<infer S,
 any> ? S : never;
 type DependencyList = readonly unknown[];

 // NOTE: callbacks are _only_ allowed to return either void, or a
 destructor.
 type EffectCallback = () => void | Destructor;

 /**
 * @deprecated Use `RefObject` instead.
 */
 interface MutableRefObject<T> {
 current: T;
 }

 // This will technically work if you give a Consumer<T> or Provider<T> but
 it's deprecated and warns
 /**

```

```
* Accepts a context object (the value returned from `React.createContext`)
and returns the current
* context value, as given by the nearest context provider for the given
context.
*
* @version 16.8.0
* @see {@link https://react.dev/reference/react/useContext}
*/
function useContext<T>(context: Context<T> /*, (not public API)
observedBits?: number|boolean */): T;
/**
* Returns a stateful value, and a function to update it.
*
* @version 16.8.0
* @see {@link https://react.dev/reference/react/useState}
*/
function useState<S>(initialState: S | (() => S)): [S,
Dispatch<SetStateAction<S>>];
// convenience overload when first argument is omitted
/**
* Returns a stateful value, and a function to update it.
*
* @version 16.8.0
* @see {@link https://react.dev/reference/react/useState}
*/
function useState<S = undefined>(): [S | undefined,
Dispatch<SetStateAction<S | undefined>>];
/**
* An alternative to `useState`.
*
* `useReducer` is usually preferable to `useState` when you have complex
state logic that involves
* multiple sub-values. It also lets you optimize performance for
components that trigger deep
* updates because you can pass `dispatch` down instead of callbacks.
*
* @version 16.8.0
* @see {@link https://react.dev/reference/react/useReducer}
*/
function useReducer<S, A extends AnyActionArg>(
 reducer: (prevState: S, ...args: A) => S,
 initialState: S,
): [S, ActionDispatch<A>];
/**
* An alternative to `useState`.
*
* `useReducer` is usually preferable to `useState` when you have complex
state logic that involves
```

```
* multiple sub-values. It also lets you optimize performance for
components that trigger deep
* updates because you can pass `dispatch` down instead of callbacks.
*
* @version 16.8.0
* @see {@link https://react.dev/reference/react/useReducer}
*/
function useReducer<S, I, A extends AnyActionArg>(
 reducer: (prevState: S, ...args: A) => S,
 initialArg: I,
 init: (i: I) => S,
): [S, ActionDispatch<A>];
/**
* `useRef` returns a mutable ref object whose `current` property is
initialized to the passed argument
* (`initialValue`). The returned object will persist for the full lifetime
of the component.
*
* Note that `useRef()` is useful for more than the `ref` attribute. It's
handy for keeping any mutable
* value around similar to how you'd use instance fields in classes.
*
* @version 16.8.0
* @see {@link https://react.dev/reference/react/useRef}
*/
function useRef<T>(initialValue: T): RefObject<T>;
// convenience overload for refs given as a ref prop as they typically start
with a null value
/**
* `useRef` returns a mutable ref object whose `current` property is
initialized to the passed argument
* (`initialValue`). The returned object will persist for the full lifetime
of the component.
*
* Note that `useRef()` is useful for more than the `ref` attribute. It's
handy for keeping any mutable
* value around similar to how you'd use instance fields in classes.
*
* @version 16.8.0
* @see {@link https://react.dev/reference/react/useRef}
*/
function useRef<T>(initialValue: T | null): RefObject<T | null>;
// convenience overload for undefined initialValue
/**
* `useRef` returns a mutable ref object whose `current` property is
initialized to the passed argument
* (`initialValue`). The returned object will persist for the full lifetime
of the component.
*
```

```
* Note that `useRef()` is useful for more than the `ref` attribute. It's
handy for keeping any mutable
 * value around similar to how you'd use instance fields in classes.
 *
 * @version 16.8.0
 * @see {@link https://react.dev/reference/react/useRef}
 */
function useRef<T>(initialValue: T | undefined): RefObject<T | undefined>;
/***
 * The signature is identical to `useEffect`, but it fires synchronously
after all DOM mutations.
 * Use this to read layout from the DOM and synchronously re-render.
Updates scheduled inside
 * `useLayoutEffect` will be flushed synchronously, before the browser has
a chance to paint.
 *
 * Prefer the standard `useEffect` when possible to avoid blocking visual
updates.
 *
 * If you're migrating code from a class component, `useLayoutEffect` fires
in the same phase as
 * `componentDidMount` and `componentDidUpdate`.
 *
 * @version 16.8.0
 * @see {@link https://react.dev/reference/react/useLayoutEffect}
 */
function useLayoutEffect(effect: EffectCallback, deps?: DependencyList): void;
/***
 * Accepts a function that contains imperative, possibly effectful code.
 *
 * @param effect Imperative function that can return a cleanup function
 * @param deps If present, effect will only activate if the values in the
list change.
 *
 * @version 16.8.0
 * @see {@link https://react.dev/reference/react/useEffect}
 */
function useEffect(effect: EffectCallback, deps?: DependencyList): void;
// NOTE: this does not accept strings, but this will have to be fixed by
removing strings from type Ref<T>
/***
 * `useImperativeHandle` customizes the instance value that is exposed to
parent components when using
 * `ref`. As always, imperative code using refs should be avoided in most
cases.
 *
 * `useImperativeHandle` should be used with `React.forwardRef`.
 */
```

```

 * @version 16.8.0
 * @see {@link https://react.dev/reference/react/useImperativeHandle}
 */
function useImperativeHandle<T, R extends T>(ref: Ref<T> | undefined, init: () => R, deps?: DependencyList): void;
 // I made 'inputs' required here and in useMemo as there's no point to memoizing without the memoization key
 // useCallback(X) is identical to just using X, useMemo(() => Y) is identical to just using Y.
 /**
 * `useCallback` will return a memoized version of the callback that only changes if one of the `inputs` has changed.
 *
 * @version 16.8.0
 * @see {@link https://react.dev/reference/react/useCallback}
 */
 // A specific function type would not trigger implicit any.
 // See
https://github.com/DefinitelyTyped/DefinitelyTyped/issues/52873#issuecomment-845806435 for a comparison between `Function` and more specific types.
 // eslint-disable-next-line @typescript-eslint/no-unsafe-function-type
 function useCallback<T extends Function>(callback: T, deps: DependencyList): T;
 /**
 * `useMemo` will only recompute the memoized value when one of the `deps` has changed.
 *
 * @version 16.8.0
 * @see {@link https://react.dev/reference/react/useMemo}
 */
 // allow undefined, but don't make it optional as that is very likely a mistake
 function useMemo<T>(factory: () => T, deps: DependencyList): T;
 /**
 * `useDebugValue` can be used to display a label for custom hooks in React DevTools.
 *
 * NOTE: We don't recommend adding debug values to every custom hook.
 * It's most valuable for custom hooks that are part of shared libraries.
 *
 * @version 16.8.0
 * @see {@link https://react.dev/reference/react/useDebugValue}
 */
 // the name of the custom hook is itself derived from the function name at runtime:
 // it's just the function name without the "use" prefix.
 function useDebugValue<T>(value: T, format?: (value: T) => any): void;

```

```
export type TransitionFunction = () => VoidOrUndefinedOnly | Promise<VoidOrUndefinedOnly>;
// strange definition to allow vscode to show documentation on the invocation
export interface TransitionStartFunction {
 /**
 * State updates caused inside the callback are allowed to be deferred.
 *
 * **If some state update causes a component to suspend, that state update should be wrapped in a transition.**
 *
 * @param callback A function which causes state updates that can be deferred.
 */
 (callback: TransitionFunction): void;
}

/**
 * Returns a deferred version of the value that may "lag behind" it.
 *
 * This is commonly used to keep the interface responsive when you have something that renders immediately
 * based on user input and something that needs to wait for a data fetch.
 *
 * A good example of this is a text input.
 *
 * @param value The value that is going to be deferred
 * @param initialValue A value to use during the initial render of a component. If this option is omitted, `useDeferredValue` will not defer during the initial render, because there's no previous version of `value` that it can render instead.
 *
 * @see {@link https://react.dev/reference/react/useDeferredValue}
 */
export function useDeferredValue<T>(value: T, initialValue?: T): T;

/**
 * Allows components to avoid undesirable loading states by waiting for content to load
 * before transitioning to the next screen. It also allows components to defer slower,
 * data fetching updates until subsequent renders so that more crucial updates can be rendered immediately.
 *
 * The `useTransition` hook returns two values in an array.
 *
 * The first is a boolean, React's way of informing us whether we're waiting for the transition to finish.

```

```

 * The second is a function that takes a callback. We can use it to tell
React which state we want to defer.
 *
 * **If some state update causes a component to suspend, that state update
should be wrapped in a transition.**
 *
 * @see {@link https://react.dev/reference/react/useTransition}
 */
export function useTransition(): [boolean, TransitionStartFunction];

/**
 * Similar to `useTransition` but allows uses where hooks are not
available.
 *
 * @param callback A function which causes state updates that can be
deferred.
 */
export function startTransition(scope: TransitionFunction): void;

/**
 * Wrap any code rendering and triggering updates to your components into
`act()` calls.
 *
 * Ensures that the behavior in your tests matches what happens in the
browser
 * more closely by executing pending `useEffect`s before returning. This
also
 * reduces the amount of re-renders done.
 *
 * @param callback A synchronous, void callback that will execute as a
single, complete React commit.
 *
 * @see
https://reactjs.org/blog/2019/02/06/react-v16.8.0.html#testing-hooks
*/
// NOTES
// - the order of these signatures matters - typescript will check the
signatures in source order.
// If the `() => VoidOrUndefinedOnly` signature is first, it'll
erroneously match a Promise returning function for users with
// `strictNullChecks: false`.
// - VoidOrUndefinedOnly is there to forbid any non-void return values for
users with `strictNullChecks: true`
// While act does always return Thenable, if a void function is passed, we
pretend the return value is also void to not trigger dangling Promise lint
rules.
export function act(callback: () => VoidOrUndefinedOnly): void;
export function act<T>(callback: () => T | Promise<T>): Promise<T>;

```

```
export function useId(): string;

/**
 * @param effect Imperative function that can return a cleanup function
 * @param deps If present, effect will only activate if the values in the
list change.
 *
 * @see {@link https://github.com/facebook/react/pull/21913}
 */
export function useInsertionEffect(effect: EffectCallback, deps?: DependencyList): void;

/**
 * @param subscribe
 * @param getSnapshot
 *
 * @see {@link https://github.com/reactwg/react-18/discussions/86}
 */
// keep in sync with `useSyncExternalStore` from `use-sync-external-store`
export function useSyncExternalStore<Snapshot>(
 subscribe: (onStoreChange: () => void) => () => void,
 getSnapshot: () => Snapshot,
 getServerSnapshot?: () => Snapshot,
): Snapshot;

export function useOptimistic<State>(
 passthrough: State,
): [State, (action: State | ((pendingState: State) => State)) => void];
export function useOptimistic<State, Action>(
 passthrough: State,
 reducer: (state: State, action: Action) => State,
): [State, (action: Action) => void];

export type Usable<T> = PromiseLike<T> | Context<T>;

export function use<T>(usable: Usable<T>): T;

export function useActionState<State>(
 action: (state: Awaited<State>) => State | Promise<State>,
 initialState: Awaited<State>,
 permalink?: string,
): [state: Awaited<State>, dispatch: () => void, isPending: boolean];
export function useActionState<State, Payload>(
 action: (state: Awaited<State>, payload: Payload) => State | Promise<State>,
 initialState: Awaited<State>,
 permalink?: string,
): [state: Awaited<State>, dispatch: (payload: Payload) => void, isPending: boolean];
```

```
// eslint-disable-next-line @typescript-eslint/no-unsafe-function-type
export function cache<CachedFunction extends Function>(fn: CachedFunction): CachedFunction;

/**
 * Warning: Only available in development builds.
 *
 * @see {@link https://react.dev/reference/react/captureOwnerStack Reference docs}
 */
function captureOwnerStack(): string | null;

// -----
// Event System
// -----
// TODO: change any to unknown when moving to TS v3
interface BaseSyntheticEvent<E = object, C = any, T = any> {
 nativeEvent: E;
 currentTarget: C;
 target: T;
 bubbles: boolean;
 cancelable: boolean;
 defaultPrevented: boolean;
 eventPhase: number;
 isTrusted: boolean;
 preventDefault(): void;
 isDefaultPrevented(): boolean;
 stopPropagation(): void;
 isPropagationStopped(): boolean;
 persist(): void;
 timeStamp: number;
 type: string;
}

/**
 * currentTarget - a reference to the element on which the event listener is registered.
 *
 * target - a reference to the element from which the event was originally dispatched.
 * This might be a child element to the element on which the event listener is registered.
 * If you thought this should be `EventTarget & T`, see
https://github.com/DefinitelyTyped/DefinitelyTyped/issues/11508#issuecomment-256045682
 *
 * SyntheticEvent<T = Element, E = Event> extends BaseSyntheticEvent<E, EventTarget & T, EventTarget> {}
```

```
interface ClipboardEvent<T = Element> extends SyntheticEvent<T, NativeClipboardEvent> {
 clipboardData: DataTransfer;
}

interface CompositionEvent<T = Element> extends SyntheticEvent<T, NativeCompositionEvent> {
 data: string;
}

interface DragEvent<T = Element> extends MouseEvent<T, NativeDragEvent> {
 dataTransfer: DataTransfer;
}

interface PointerEvent<T = Element> extends MouseEvent<T, NativePointerEvent> {
 pointerId: number;
 pressure: number;
 tangentialPressure: number;
 tiltX: number;
 tiltY: number;
 twist: number;
 width: number;
 height: number;
 pointerType: "mouse" | "pen" | "touch";
 isPrimary: boolean;
}

interface FocusEvent<Target = Element, RelatedTarget = Element> extends SyntheticEvent<Target, NativeFocusEvent> {
 relatedTarget: (EventTarget & RelatedTarget) | null;
 target: EventTarget & Target;
}

interface FormEvent<T = Element> extends SyntheticEvent<T> {}

interface InvalidEvent<T = Element> extends SyntheticEvent<T> {
 target: EventTarget & T;
}

interface ChangeEvent<T = Element> extends SyntheticEvent<T> {
 target: EventTarget & T;
}

interface InputEvent<T = Element> extends SyntheticEvent<T, NativeInputEvent> {
 data: string;
```

```
}

export type ModifierKey =
 | "Alt"
 | "AltGraph"
 | "CapsLock"
 | "Control"
 | "Fn"
 | "FnLock"
 | "Hyper"
 | "Meta"
 | "NumLock"
 | "ScrollLock"
 | "Shift"
 | "Super"
 | "Symbol"
 | "SymbolLock";

interface KeyboardEvent<T = Element> extends UIEvent<T, NativeKeyboardEvent>
{
 altKey: boolean;
 /** @deprecated */
 charCode: number;
 ctrlKey: boolean;
 code: string;
 /**
 * See [DOM Level 3 Events
 spec] (https://www.w3.org/TR/uievents-key/#keys-modifier). for a list of valid
 (case-sensitive) arguments to this method.
 */
 getModifierState(key: ModifierKey): boolean;
 /**
 * See the [DOM Level 3 Events
 spec] (https://www.w3.org/TR/uievents-key/#named-key-attribute-values). for
 possible values
 */
 key: string;
 /** @deprecated */
 keyCode: number;
 locale: string;
 location: number;
 metaKey: boolean;
 repeat: boolean;
 shiftKey: boolean;
 /** @deprecated */
 which: number;
}
```

```
interface MouseEvent<T = Element, E = NativeMouseEvent> extends UIEvent<T, E> {
 altKey: boolean;
 button: number;
 buttons: number;
 clientX: number;
 clientY: number;
 ctrlKey: boolean;
 /**
 * See [DOM Level 3 Events spec] (https://www.w3.org/TR/uievents-key/#keys-modifier). for a list of valid (case-sensitive) arguments to this method.
 */
 getModifierState(key: ModifierKey): boolean;
 metaKey: boolean;
 movementX: number;
 movementY: number;
 pageX: number;
 pageY: number;
 relatedTarget: EventTarget | null;
 screenX: number;
 screenY: number;
 shiftKey: boolean;
}

interface TouchEvent<T = Element> extends UIEvent<T, NativeTouchEvent> {
 altKey: boolean;
 changedTouches: TouchList;
 ctrlKey: boolean;
 /**
 * See [DOM Level 3 Events spec] (https://www.w3.org/TR/uievents-key/#keys-modifier). for a list of valid (case-sensitive) arguments to this method.
 */
 getModifierState(key: ModifierKey): boolean;
 metaKey: boolean;
 shiftKey: boolean;
 targetTouches: TouchList;
 touches: TouchList;
}

interface UIEvent<T = Element, E = NativeUIEvent> extends SyntheticEvent<T, E> {
 detail: number;
 view: AbstractView;
}

interface WheelEvent<T = Element> extends MouseEvent<T, NativeWheelEvent> {
 deltaMode: number;
```

```
 deltaX: number;
 deltaY: number;
 deltaZ: number;
 }

 interface AnimationEvent<T = Element> extends SyntheticEvent<T,
NativeAnimationEvent> {
 animationName: string;
 elapsedTime: number;
 pseudoElement: string;
}

interface ToggleEvent<T = Element> extends SyntheticEvent<T,
NativeToggleEvent> {
 oldState: "closed" | "open";
 newState: "closed" | "open";
}

interface TransitionEvent<T = Element> extends SyntheticEvent<T,
NativeTransitionEvent> {
 elapsedTime: number;
 propertyName: string;
 pseudoElement: string;
}

// -----
// Event Handler Types
// -----

type EventHandler<E extends SyntheticEvent<any>> = { bivarianceHack(event: E): void }["bivarianceHack"];

type ReactEventHandler<T = Element> = EventHandler<SyntheticEvent<T>>;

type ClipboardEventHandler<T = Element> = EventHandler<ClipboardEvent<T>>;
type CompositionEventHandler<T = Element> =
EventHandler<CompositionEvent<T>>;
type DragEventHandler<T = Element> = EventHandler<DragEvent<T>>;
type FocusEventHandler<T = Element> = EventHandler<FocusEvent<T>>;
type FormEventHandler<T = Element> = EventHandler<FormEvent<T>>;
type ChangeEventHandler<T = Element> = EventHandler<ChangeEvent<T>>;
type InputEventHandler<T = Element> = EventHandler<InputEvent<T>>;
type KeyboardEventHandler<T = Element> = EventHandler<KeyboardEvent<T>>;
type MouseEventHandler<T = Element> = EventHandler<MouseEvent<T>>;
type TouchEventHandler<T = Element> = EventHandler<TouchEvent<T>>;
type PointerEventHandler<T = Element> = EventHandler<PointerEvent<T>>;
type UIEventHandler<T = Element> = EventHandler<UIEvent<T>>;
type WheelEventHandler<T = Element> = EventHandler<WheelEvent<T>>;
type AnimationEventHandler<T = Element> = EventHandler<AnimationEvent<T>>;
```

```
type ToggleEventHandler<T = Element> = EventHandler<ToggleEvent<T>>;
type TransitionEventHandler<T = Element> = EventHandler<TransitionEvent<T>>;

//
// Props / DOM Attributes
// -----

interface HTMLProps<T> extends AllHTMLAttributes<T>, ClassAttributes<T> {}

type DetailedHTMLProps<E extends HTMLAttributes<T>, T> = ClassAttributes<T>
& E;

interface SVGProps<T> extends SVGAttributes<T>, ClassAttributes<T> {}

interface SVGLineElementAttributes<T> extends SVGProps<T> {}
interface SVGTextElementAttributes<T> extends SVGProps<T> {}

interface DOMAttributes<T> {
 children?: ReactNode | undefined;
 dangerouslySetInnerHTML?: {
 // Should be InnerHTML['innerHTML'].
 // But unfortunately we're mixing renderer-specific type
 declarations.
 __html: string | TrustedHTML;
 } | undefined;

 // Clipboard Events
 onCopy?: ClipboardEventHandler<T> | undefined;
 onCopyCapture?: ClipboardEventHandler<T> | undefined;
 onCut?: ClipboardEventHandler<T> | undefined;
 onCutCapture?: ClipboardEventHandler<T> | undefined;
 onPaste?: ClipboardEventHandler<T> | undefined;
 onPasteCapture?: ClipboardEventHandler<T> | undefined;

 // Composition Events
 onCompositionEnd?: CompositionEventHandler<T> | undefined;
 onCompositionEndCapture?: CompositionEventHandler<T> | undefined;
 onCompositionStart?: CompositionEventHandler<T> | undefined;
 onCompositionStartCapture?: CompositionEventHandler<T> | undefined;
 onCompositionUpdate?: CompositionEventHandler<T> | undefined;
 onCompositionUpdateCapture?: CompositionEventHandler<T> | undefined;

 // Focus Events
 onFocus?: FocusEventHandler<T> | undefined;
 onFocusCapture?: FocusEventHandler<T> | undefined;
 onBlur?: FocusEventHandler<T> | undefined;
 onBlurCapture?: FocusEventHandler<T> | undefined;
```

```
// Form Events
onChange?: FormEventHandler<T> | undefined;
onChangeCapture?: FormEventHandler<T> | undefined;
onBeforeInput?: InputEventHandler<T> | undefined;
onBeforeInputCapture?: FormEventHandler<T> | undefined;
onInput?: FormEventHandler<T> | undefined;
onInputCapture?: FormEventHandler<T> | undefined;
onReset?: FormEventHandler<T> | undefined;
onResetCapture?: FormEventHandler<T> | undefined;
onSubmit?: FormEventHandler<T> | undefined;
onSubmitCapture?: FormEventHandler<T> | undefined;
onInvalid?: FormEventHandler<T> | undefined;
onInvalidCapture?: FormEventHandler<T> | undefined;

// Image Events
onLoad?: ReactEventHandler<T> | undefined;
onLoadCapture?: ReactEventHandler<T> | undefined;
onError?: ReactEventHandler<T> | undefined; // also a Media Event
onErrorCapture?: ReactEventHandler<T> | undefined; // also a Media Event

// Keyboard Events
onKeyDown?: KeyboardEventHandler<T> | undefined;
onKeyDownCapture?: KeyboardEventHandler<T> | undefined;
/** @deprecated Use `onKeyUp` or `onKeyDown` instead */
onKeyPress?: KeyboardEventHandler<T> | undefined;
/** @deprecated Use `onKeyUpCapture` or `onKeyDownCapture` instead */
onKeyPressCapture?: KeyboardEventHandler<T> | undefined;
onKeyUp?: KeyboardEventHandler<T> | undefined;
onKeyUpCapture?: KeyboardEventHandler<T> | undefined;

// Media Events
onAbort?: ReactEventHandler<T> | undefined;
onAbortCapture?: ReactEventHandler<T> | undefined;
onCanPlay?: ReactEventHandler<T> | undefined;
onCanPlayCapture?: ReactEventHandler<T> | undefined;
onCanPlayThrough?: ReactEventHandler<T> | undefined;
onCanPlayThroughCapture?: ReactEventHandler<T> | undefined;
onDurationChange?: ReactEventHandler<T> | undefined;
onDurationChangeCapture?: ReactEventHandler<T> | undefined;
onEmptied?: ReactEventHandler<T> | undefined;
onEmptiedCapture?: ReactEventHandler<T> | undefined;
onEncrypted?: ReactEventHandler<T> | undefined;
onEncryptedCapture?: ReactEventHandler<T> | undefined;
onEnded?: ReactEventHandler<T> | undefined;
onEndedCapture?: ReactEventHandler<T> | undefined;
onLoadedData?: ReactEventHandler<T> | undefined;
onLoadedDataCapture?: ReactEventHandler<T> | undefined;
onLoadedMetadata?: ReactEventHandler<T> | undefined;
```

```
onLoadedMetadataCapture?: ReactEventHandler<T> | undefined;
onLoadStart?: ReactEventHandler<T> | undefined;
onLoadStartCapture?: ReactEventHandler<T> | undefined;
onPause?: ReactEventHandler<T> | undefined;
onPauseCapture?: ReactEventHandler<T> | undefined;
onPlay?: ReactEventHandler<T> | undefined;
onPlayCapture?: ReactEventHandler<T> | undefined;
onPlaying?: ReactEventHandler<T> | undefined;
onPlayingCapture?: ReactEventHandler<T> | undefined;
onProgress?: ReactEventHandler<T> | undefined;
onProgressCapture?: ReactEventHandler<T> | undefined;
onRateChange?: ReactEventHandler<T> | undefined;
onRateChangeCapture?: ReactEventHandler<T> | undefined;
onSeeked?: ReactEventHandler<T> | undefined;
onSeekedCapture?: ReactEventHandler<T> | undefined;
onSeeking?: ReactEventHandler<T> | undefined;
onSeekingCapture?: ReactEventHandler<T> | undefined;
onStalled?: ReactEventHandler<T> | undefined;
onStalledCapture?: ReactEventHandler<T> | undefined;
onSuspend?: ReactEventHandler<T> | undefined;
onSuspendCapture?: ReactEventHandler<T> | undefined;
onTimeUpdate?: ReactEventHandler<T> | undefined;
onTimeUpdateCapture?: ReactEventHandler<T> | undefined;
onVolumeChange?: ReactEventHandler<T> | undefined;
onVolumeChangeCapture?: ReactEventHandler<T> | undefined;
onWaiting?: ReactEventHandler<T> | undefined;
onWaitingCapture?: ReactEventHandler<T> | undefined;

// MouseEvents
onAuxClick?: MouseEventHandler<T> | undefined;
onAuxClickCapture?: MouseEventHandler<T> | undefined;
onClick?: MouseEventHandler<T> | undefined;
onClickCapture?: MouseEventHandler<T> | undefined;
onContextMenu?: MouseEventHandler<T> | undefined;
onContextMenuCapture?: MouseEventHandler<T> | undefined;
onDoubleClick?: MouseEventHandler<T> | undefined;
onDoubleClickCapture?: MouseEventHandler<T> | undefined;
onDrag?: DragEventHandler<T> | undefined;
onDragCapture?: DragEventHandler<T> | undefined;
onDragEnd?: DragEventHandler<T> | undefined;
onDragEndCapture?: DragEventHandler<T> | undefined;
onDragEnter?: DragEventHandler<T> | undefined;
onDragEnterCapture?: DragEventHandler<T> | undefined;
onDragExit?: DragEventHandler<T> | undefined;
onDragExitCapture?: DragEventHandler<T> | undefined;
onDragLeave?: DragEventHandler<T> | undefined;
onDragLeaveCapture?: DragEventHandler<T> | undefined;
onDragOver?: DragEventHandler<T> | undefined;
onDragOverCapture?: DragEventHandler<T> | undefined;
```

```
onDragStart?: DragEventHandler<T> | undefined;
onDragStartCapture?: DragEventHandler<T> | undefined;
onDrop?: DragEventHandler<T> | undefined;
onDropCapture?: DragEventHandler<T> | undefined;
onMouseDown?: MouseEventHandler<T> | undefined;
onMouseDownCapture?: MouseEventHandler<T> | undefined;
onMouseEnter?: MouseEventHandler<T> | undefined;
onMouseLeave?: MouseEventHandler<T> | undefined;
onMouseMove?: MouseEventHandler<T> | undefined;
onMouseMoveCapture?: MouseEventHandler<T> | undefined;
onMouseOut?: MouseEventHandler<T> | undefined;
onMouseOutCapture?: MouseEventHandler<T> | undefined;
onMouseOver?: MouseEventHandler<T> | undefined;
onMouseOverCapture?: MouseEventHandler<T> | undefined;
onMouseUp?: MouseEventHandler<T> | undefined;
onMouseUpCapture?: MouseEventHandler<T> | undefined;

// Selection Events
onSelect?: ReactEventHandler<T> | undefined;
onSelectCapture?: ReactEventHandler<T> | undefined;

// Touch Events
onTouchCancel?: TouchEventHandler<T> | undefined;
onTouchCancelCapture?: TouchEventHandler<T> | undefined;
onTouchEnd?: TouchEventHandler<T> | undefined;
onTouchEndCapture?: TouchEventHandler<T> | undefined;
onTouchMove?: TouchEventHandler<T> | undefined;
onTouchMoveCapture?: TouchEventHandler<T> | undefined;
onTouchStart?: TouchEventHandler<T> | undefined;
onTouchStartCapture?: TouchEventHandler<T> | undefined;

// Pointer Events
onPointerDown?: PointerEventHandler<T> | undefined;
onPointerDownCapture?: PointerEventHandler<T> | undefined;
onPointerMove?: PointerEventHandler<T> | undefined;
onPointerMoveCapture?: PointerEventHandler<T> | undefined;
onPointerUp?: PointerEventHandler<T> | undefined;
onPointerUpCapture?: PointerEventHandler<T> | undefined;
onPointerCancel?: PointerEventHandler<T> | undefined;
onPointerCancelCapture?: PointerEventHandler<T> | undefined;
onPointerEnter?: PointerEventHandler<T> | undefined;
onPointerLeave?: PointerEventHandler<T> | undefined;
onPointerOver?: PointerEventHandler<T> | undefined;
onPointerOverCapture?: PointerEventHandler<T> | undefined;
onPointerOut?: PointerEventHandler<T> | undefined;
onPointerOutCapture?: PointerEventHandler<T> | undefined;
onGotPointerCapture?: PointerEventHandler<T> | undefined;
onGotPointerCaptureCapture?: PointerEventHandler<T> | undefined;
onLostPointerCapture?: PointerEventHandler<T> | undefined;
```

```

onLostPointerCapture?: PointerEventHandler<T> | undefined;

// UI Events
onScroll?: UIEventHandler<T> | undefined;
onScrollCapture?: UIEventHandler<T> | undefined;
onScrollEnd?: UIEventHandler<T> | undefined;
onScrollEndCapture?: UIEventHandler<T> | undefined;

// Wheel Events
onWheel?: WheelEventHandler<T> | undefined;
onWheelCapture?: WheelEventHandler<T> | undefined;

// Animation Events
onAnimationStart?: AnimationEventHandler<T> | undefined;
onAnimationStartCapture?: AnimationEventHandler<T> | undefined;
onAnimationEnd?: AnimationEventHandler<T> | undefined;
onAnimationEndCapture?: AnimationEventHandler<T> | undefined;
onAnimationIteration?: AnimationEventHandler<T> | undefined;
onAnimationIterationCapture?: AnimationEventHandler<T> | undefined;

// Toggle Events
onToggle?: ToggleEventHandler<T> | undefined;
onBeforeToggle?: ToggleEventHandler<T> | undefined;

// Transition Events
onTransitionCancel?: TransitionEventHandler<T> | undefined;
onTransitionCancelCapture?: TransitionEventHandler<T> | undefined;
onTransitionEnd?: TransitionEventHandler<T> | undefined;
onTransitionEndCapture?: TransitionEventHandler<T> | undefined;
onTransitionRun?: TransitionEventHandler<T> | undefined;
onTransitionRunCapture?: TransitionEventHandler<T> | undefined;
onTransitionStart?: TransitionEventHandler<T> | undefined;
onTransitionStartCapture?: TransitionEventHandler<T> | undefined;
}

export interface CSSProperties extends CSS.Properties<string | number> {
 /**
 * The index signature was removed to enable closed typing for style
 * using CSSType. You're able to use type assertion or module
 augmentation
 * to add properties or an index signature of your own.
 *
 * For examples and more information, visit:
 *
https://github.com/frenic/csstype#what-should-i-do-when-i-get-type-errors
 */
}

// All the WAI-ARIA 1.1 attributes from https://www.w3.org/TR/wai-aria-1.1/

```

```

interface AriaAttributes {
 /** Identifies the currently active element when DOM focus is on a
composite widget, textbox, group, or application. */
 "aria-activedescendant"?: string | undefined;
 /** Indicates whether assistive technologies will present all, or only
parts of, the changed region based on the change notifications defined by the
aria-relevant attribute. */
 "aria-atomic"?: Booleanish | undefined;
 /**
 * Indicates whether inputting text could trigger display of one or
more predictions of the user's intended value for an input and specifies how
predictions would be
 * presented if they are made.
 */
 "aria-autocomplete"?: "none" | "inline" | "list" | "both" | undefined;
 /** Indicates an element is being modified and that assistive
technologies MAY want to wait until the modifications are complete before
exposing them to the user. */
 /**
 * Defines a string value that labels the current element, which is
intended to be converted into Braille.
 * @see aria-label.
 */
 "aria-braillelabel"?: string | undefined;
 /**
 * Defines a human-readable, author-localized abbreviated description
for the role of an element, which is intended to be converted into Braille.
 * @see aria-roledescription.
 */
 "aria-brailleroledescription"?: string | undefined;
 "aria-busy"?: Booleanish | undefined;
 /**
 * Indicates the current "checked" state of checkboxes, radio buttons,
and other widgets.
 * @see aria-pressed @see aria-selected.
 */
 "aria-checked"?: boolean | "false" | "mixed" | "true" | undefined;
 /**
 * Defines the total number of columns in a table, grid, or treegrid.
 * @see aria-colindex.
 */
 "aria-colcount"?: number | undefined;
 /**
 * Defines an element's column index or position with respect to the
total number of columns within a table, grid, or treegrid.
 * @see aria-colcount @see aria-colspan.
 */
 "aria-colindex"?: number | undefined;
 /**

```

```
* Defines a human readable text alternative of aria-colindex.
* @see aria-rowindextext.
*/
"aria-colindextext"??: string | undefined;
/***
 * Defines the number of columns spanned by a cell or gridcell within a
table, grid, or treegrid.
 * @see aria-colindex @see aria-rowspan.
*/
"aria-colspan"??: number | undefined;
/***
 * Identifies the element (or elements) whose contents or presence are
controlled by the current element.
 * @see aria-owns.
*/
"aria-controls"??: string | undefined;
/***
 * Indicates the element that represents the current item within a
container or set of related elements. */
"aria-current"??: boolean | "false" | "true" | "page" | "step" |
"location" | "date" | "time" | undefined;
/***
 * Identifies the element (or elements) that describes the object.
 * @see aria-labelledby
*/
"aria-describedby"??: string | undefined;
/***
 * Defines a string value that describes or annotates the current
element.
 * @see related aria-describedby.
*/
"aria-description"??: string | undefined;
/***
 * Identifies the element that provides a detailed, extended
description for the object.
 * @see aria-describedby.
*/
"aria-details"??: string | undefined;
/***
 * Indicates that the element is perceivable but disabled, so it is not
editable or otherwise operable.
 * @see aria-hidden @see aria-readonly.
*/
"aria-disabled"??: Booleanish | undefined;
/***
 * Indicates what functions can be performed when a dragged object is
released on the drop target.
 * @deprecated in ARIA 1.1
*/
```

```
"aria-dropeffect"?:"none" | "copy" | "execute" | "link" | "move" |
"popup" | undefined;
/**
 * Identifies the element that provides an error message for the
object.
 * @see aria-invalid @see aria-describedby.
 */
"aria-errormessage"??: string | undefined;
/** Indicates whether the element, or another grouping element it
controls, is currently expanded or collapsed. */
"aria-expanded"??: Booleanish | undefined;
/**
 * Identifies the next element (or elements) in an alternate reading
order of content which, at the user's discretion,
* allows assistive technology to override the general default of
reading in document source order.
 */
"aria-flowto"??: string | undefined;
/**
 * Indicates an element's "grabbed" state in a drag-and-drop operation.
 * @deprecated in ARIA 1.1
 */
"aria-grabbed"??: Booleanish | undefined;
/** Indicates the availability and type of interactive popup element,
such as menu or dialog, that can be triggered by an element. */
"aria-haspopup"??: boolean | "false" | "true" | "menu" | "listbox" |
"tree" | "grid" | "dialog" | undefined;
/**
 * Indicates whether the element is exposed to an accessibility API.
 * @see aria-disabled.
 */
"aria-hidden"??: Booleanish | undefined;
/**
 * Indicates the entered value does not conform to the format expected
by the application.
 * @see aria-errormessage.
 */
"aria-invalid"??: boolean | "false" | "true" | "grammar" | "spelling" |
undefined;
/** Indicates keyboard shortcuts that an author has implemented to
activate or give focus to an element. */
"aria-keyshortcuts"??: string | undefined;
/**
 * Defines a string value that labels the current element.
 * @see aria-labelledby.
 */
"aria-label"??: string | undefined;
/**
```

```
 * Identifies the element (or elements) that labels the current
element.
 * @see aria-describedby.
 */
"aria-labelledby"? : string | undefined;
/** Defines the hierarchical level of an element within a structure. */
"aria-level"? : number | undefined;
/** Indicates that an element will be updated, and describes the types
of updates the user agents, assistive technologies, and user can expect from
the live region. */
"aria-live"? : "off" | "assertive" | "polite" | undefined;
/** Indicates whether an element is modal when displayed. */
"aria-modal"? : Booleanish | undefined;
/** Indicates whether a text box accepts multiple lines of input or
only a single line. */
"aria-multiline"? : Booleanish | undefined;
/** Indicates that the user may select more than one item from the
current selectable descendants. */
"aria-multipleselectable"? : Booleanish | undefined;
/** Indicates whether the element's orientation is horizontal,
vertical, or unknown/ambiguous. */
"aria-orientation"? : "horizontal" | "vertical" | undefined;
/**
 * Identifies an element (or elements) in order to define a visual,
functional, or contextual parent/child relationship
 * between DOM elements where the DOM hierarchy cannot be used to
represent the relationship.
 * @see aria-controls.
 */
"aria-owns"? : string | undefined;
/***
 * Defines a short hint (a word or short phrase) intended to aid the
user with data entry when the control has no value.
 * A hint could be a sample value or a brief description of the
expected format.
 */
"aria-placeholder"? : string | undefined;
/***
 * Defines an element's number or position in the current set of
listitems or treeitems. Not required if all elements in the set are present in
the DOM.
 * @see aria-setsize.
 */
"aria-posinset"? : number | undefined;
/***
 * Indicates the current "pressed" state of toggle buttons.
 * @see aria-checked @see aria-selected.
 */
"aria-pressed"? : boolean | "false" | "mixed" | "true" | undefined;
```

```
/**
 * Indicates that the element is not editable, but is otherwise
operable.
 * @see aria-disabled.
 */
"aria-readonly"?: Booleanish | undefined;
/**
 * Indicates what notifications the user agent will trigger when the
accessibility tree within a live region is modified.
 * @see aria-atomic.
 */
"aria-relevant"?:
 | "additions"
 | "additions removals"
 | "additions text"
 | "all"
 | "removals"
 | "removals additions"
 | "removals text"
 | "text"
 | "text additions"
 | "text removals"
 | undefined;
/** Indicates that user input is required on the element before a form
may be submitted. */
"aria-required"?: Booleanish | undefined;
/** Defines a human-readable, author-localized description for the role
of an element. */
"aria-roledescription"?: string | undefined;
/**
 * Defines the total number of rows in a table, grid, or treegrid.
 * @see aria-rowindex.
 */
"aria-rowcount"?: number | undefined;
/**
 * Defines an element's row index or position with respect to the total
number of rows within a table, grid, or treegrid.
 * @see aria-rowcount @see aria-rowspan.
 */
"aria-rowindex"?: number | undefined;
/**
 * Defines a human readable text alternative of aria-rowindex.
 * @see aria-colindextext.
 */
"aria-rowindextext"?: string | undefined;
/**
 * Defines the number of rows spanned by a cell or gridcell within a
table, grid, or treegrid.
 * @see aria-rowindex @see aria-colspan.
```

```

 */
 "aria-rowspan?: number | undefined;
 /**
 * Indicates the current "selected" state of various widgets.
 * @see aria-checked @see aria-pressed.
 */
 "aria-selected?: Booleanish | undefined;
 /**
 * Defines the number of items in the current set of listitems or
treeitems. Not required if all elements in the set are present in the DOM.
 * @see aria-posinset.
 */
 "aria-setsize?: number | undefined;
 /**
 * Indicates if items in a table or grid are sorted in ascending or
descending order. */
 "aria-sort?: "none" | "ascending" | "descending" | "other" | undefined;
 /**
 * Defines the maximum allowed value for a range widget. */
 "aria-valuemax?: number | undefined;
 /**
 * Defines the minimum allowed value for a range widget. */
 "aria-valuemin?: number | undefined;
 /**
 * Defines the current value for a range widget.
 * @see aria-valuetext.
 */
 "aria-valuenow?: number | undefined;
 /**
 * Defines the human readable text alternative of aria-valuenow for a
range widget. */
 "aria-valuetext?: string | undefined;
}

// All the WAI-ARIA 1.1 role attribute values from
https://www.w3.org/TR/wai-aria-1.1/#role_definitions
type AriaRole =
| "alert"
| "alertdialog"
| "application"
| "article"
| "banner"
| "button"
| "cell"
| "checkbox"
| "columnheader"
| "combobox"
| "complementary"
| "contentinfo"
| "definition"
| "dialog"
| "directory"
| "document"

```

```
| "feed"
| "figure"
| "form"
| "grid"
| "gridcell"
| "group"
| "heading"
| "img"
| "link"
| "list"
| "listbox"
| "listitem"
| "log"
| "main"
| "marquee"
| "math"
| "menu"
| "menubar"
| "menuitem"
| "menuitemcheckbox"
| "menuitemradio"
| "navigation"
| "none"
| "note"
| "option"
| "presentation"
| "progressbar"
| "radio"
| "radiogroup"
| "region"
| "row"
| "rowgroup"
| "rowheader"
| "scrollbar"
| "search"
| "searchbox"
| "separator"
| "slider"
| "spinbutton"
| "status"
| "switch"
| "tab"
| "table"
| "tablist"
| "tabpanel"
| "term"
| "textbox"
| "timer"
| "toolbar"
```

```
| "tooltip"
| "tree"
| "treegrid"
| "treeitem"
| (string & {});

interface HTMLAttributes<T> extends AriaAttributes, DOMAttributes<T> {
 // React-specific Attributes
 defaultChecked?: boolean | undefined;
 defaultValue?: string | number | readonly string[] | undefined;
 suppressContentEditableWarning?: boolean | undefined;
 suppressHydrationWarning?: boolean | undefined;

 // Standard HTML Attributes
 accessKey?: string | undefined;
 autoCapitalize?: "off" | "none" | "on" | "sentences" | "words" |
 "characters" | undefined | (string & {});
 autoFocus?: boolean | undefined;
 className?: string | undefined;
 contentEditable?: Booleanish | "inherit" | "plaintext-only" | undefined;
 contextMenu?: string | undefined;
 dir?: string | undefined;
 draggable?: Booleanish | undefined;
 enterKeyHint?: "enter" | "done" | "go" | "next" | "previous" | "search" |
 "send" | undefined;
 hidden?: boolean | undefined;
 id?: string | undefined;
 lang?: string | undefined;
 nonce?: string | undefined;
 slot?: string | undefined;
 spellCheck?: Booleanish | undefined;
 style?: CSSProperties | undefined;
 tabIndex?: number | undefined;
 title?: string | undefined;
 translate?: "yes" | "no" | undefined;

 // Unknown
 radioGroup?: string | undefined; // <command>, <menuitem>

 // WAI-ARIA
 role?: AriaRole | undefined;

 // RDFa Attributes
 about?: string | undefined;
 content?: string | undefined;
 datatype?: string | undefined;
 inlist?: any;
 prefix?: string | undefined;
 property?: string | undefined;
```

```

rel?: string | undefined;
resource?: string | undefined;
rev?: string | undefined;
typeof?: string | undefined;
vocab?: string | undefined;

// Non-standard Attributes
autoCorrect?: string | undefined;
autoSave?: string | undefined;
color?: string | undefined;
itemProp?: string | undefined;
itemScope?: boolean | undefined;
itemType?: string | undefined;
itemID?: string | undefined;
itemRef?: string | undefined;
results?: number | undefined;
security?: string | undefined;
unselectable?: "on" | "off" | undefined;

// Popover API
popover?: "" | "auto" | "manual" | undefined;
popoverTargetAction?: "toggle" | "show" | "hide" | undefined;
popoverTarget?: string | undefined;

// Living Standard
/**
 * @see
https://developer.mozilla.org/en-US/docs/Web/API/HTMLElement/inert
 */
inert?: boolean | undefined;
/**
 * Hints at the type of data that might be entered by the user while
editing the element or its contents
 * @see {@link
https://html.spec.whatwg.org/multipage/interaction.html#input-modalities:-the-
inputmode-attribute}
 */
inputMode?: "none" | "text" | "tel" | "url" | "email" | "numeric" |
"decimal" | "search" | undefined;
/**
 * Specify that a standard HTML element should behave like a defined
custom built-in element
 * @see {@link
https://html.spec.whatwg.org/multipage/custom-elements.html#attr-is}
 */
is?: string | undefined;
/**

```

```
 * @see {@link
https://developer.mozilla.org/en-US/docs/Web/HTML/Global_attributes/exportparts}
 */
exportparts?: string | undefined;
/***
 * @see {@link
https://developer.mozilla.org/en-US/docs/Web/HTML/Global_attributes/part}
 */
part?: string | undefined;
}

/**
 * For internal usage only.
 * Different release channels declare additional types of ReactNode this
particular release channel accepts.
 * App or library types should never augment this interface.
*/
interface DO_NOT_USE_OR_YOU_WILL_BE_FIRED_EXPERIMENTAL_FORM_ACTIONS {}

interface AllHTMLAttributes<T> extends HTMLAttributes<T> {
 // Standard HTML Attributes
 accept?: string | undefined;
 acceptCharset?: string | undefined;
 action?:
 | string
 | undefined
 | ((formData: FormData) => void | Promise<void>)
 | DO_NOT_USE_OR_YOU_WILL_BE_FIRED_EXPERIMENTAL_FORM_ACTIONS[
 keyof DO_NOT_USE_OR_YOU_WILL_BE_FIRED_EXPERIMENTAL_FORM_ACTIONS
];
 allowFullScreen?: boolean | undefined;
 allowTransparency?: boolean | undefined;
 alt?: string | undefined;
 as?: string | undefined;
 async?: boolean | undefined;
 autoComplete?: string | undefined;
 autoPlay?: boolean | undefined;
 capture?: boolean | "user" | "environment" | undefined;
 cellPadding?: number | string | undefined;
 cellSpacing?: number | string | undefined;
 charSet?: string | undefined;
 challenge?: string | undefined;
 checked?: boolean | undefined;
 cite?: string | undefined;
 classID?: string | undefined;
 cols?: number | undefined;
 colSpan?: number | undefined;
 controls?: boolean | undefined;
```

```
coords?: string | undefined;
crossOrigin?: CrossOrigin;
data?: string | undefined;
dateTime?: string | undefined;
default?: boolean | undefined;
defer?: boolean | undefined;
disabled?: boolean | undefined;
download?: any;
encType?: string | undefined;
form?: string | undefined;
formAction?:
 | string
 | undefined
 | ((formData: FormData) => void | Promise<void>)
 | DO_NOT_USE_OR_YOU_WILL_BE_FIRED_EXPERIMENTAL_FORM_ACTIONS[
 keyof DO_NOT_USE_OR_YOU_WILL_BE_FIRED_EXPERIMENTAL_FORM_ACTIONS
];
formEncType?: string | undefined;
formMethod?: string | undefined;
formNoValidate?: boolean | undefined;
formTarget?: string | undefined;
frameBorder?: number | string | undefined;
headers?: string | undefined;
height?: number | string | undefined;
high?: number | undefined;
href?: string | undefined;
hrefLang?: string | undefined;
htmlFor?: string | undefined;
httpEquiv?: string | undefined;
integrity?: string | undefined;
keyParams?: string | undefined;
keyType?: string | undefined;
kind?: string | undefined;
label?: string | undefined;
list?: string | undefined;
loop?: boolean | undefined;
low?: number | undefined;
manifest?: string | undefined;
marginHeight?: number | undefined;
marginWidth?: number | undefined;
max?: number | string | undefined;
maxLength?: number | undefined;
media?: string | undefined;
mediaGroup?: string | undefined;
method?: string | undefined;
min?: number | string | undefined;
minLength?: number | undefined;
multiple?: boolean | undefined;
muted?: boolean | undefined;
```

```
name?: string | undefined;
noValidate?: boolean | undefined;
open?: boolean | undefined;
optimum?: number | undefined;
pattern?: string | undefined;
placeholder?: string | undefined;
playsInline?: boolean | undefined;
poster?: string | undefined;
preload?: string | undefined;
readOnly?: boolean | undefined;
required?: boolean | undefined;
reversed?: boolean | undefined;
rows?: number | undefined;
rowSpan?: number | undefined;
sandbox?: string | undefined;
scope?: string | undefined;
scoped?: boolean | undefined;
scrolling?: string | undefined;
seamless?: boolean | undefined;
selected?: boolean | undefined;
shape?: string | undefined;
size?: number | undefined;
sizes?: string | undefined;
span?: number | undefined;
src?: string | undefined;
srcDoc?: string | undefined;
srcLang?: string | undefined;
srcSet?: string | undefined;
start?: number | undefined;
step?: number | string | undefined;
summary?: string | undefined;
target?: string | undefined;
type?: string | undefined;
useMap?: string | undefined;
value?: string | readonly string[] | number | undefined;
width?: number | string | undefined;
wmode?: string | undefined;
wrap?: string | undefined;
}

type HTMLAttributeReferrerPolicy =
| ""
| "no-referrer"
| "no-referrer-when-downgrade"
| "origin"
| "origin-when-cross-origin"
| "same-origin"
| "strict-origin"
| "strict-origin-when-cross-origin"
```

```
| "unsafe-url";
```

```
type HTMLAttributeAnchorTarget =
 | "_self"
 | "_blank"
 | "_parent"
 | "_top"
 | (string & {});
```

```
interface AnchorHTMLAttributes<T> extends HTMLAttributes<T> {
 download?: any;
 href?: string | undefined;
 hrefLang?: string | undefined;
 media?: string | undefined;
 ping?: string | undefined;
 target?: HTMLAttributeAnchorTarget | undefined;
 type?: string | undefined;
 referrerPolicy?: HTMLAttributeReferrerPolicy | undefined;
}
```

```
interface AudioHTMLAttributes<T> extends MediaHTMLAttributes<T> {}
```

```
interface AreaHTMLAttributes<T> extends HTMLAttributes<T> {
 alt?: string | undefined;
 coords?: string | undefined;
 download?: any;
 href?: string | undefined;
 hrefLang?: string | undefined;
 media?: string | undefined;
 referrerPolicy?: HTMLAttributeReferrerPolicy | undefined;
 shape?: string | undefined;
 target?: string | undefined;
}
```

```
interface BaseHTMLAttributes<T> extends HTMLAttributes<T> {
 href?: string | undefined;
 target?: string | undefined;
}
```

```
interface BlockquoteHTMLAttributes<T> extends HTMLAttributes<T> {
 cite?: string | undefined;
}
```

```
interface ButtonHTMLAttributes<T> extends HTMLAttributes<T> {
 disabled?: boolean | undefined;
 form?: string | undefined;
 formAction?:
 | string
 | ((formData: FormData) => void | Promise<void>)
```

```
| DO_NOT_USE_OR_YOU_WILL_BE_FIRED_EXPERIMENTAL_FORM_ACTIONS [
 keyof DO_NOT_USE_OR_YOU_WILL_BE_FIRED_EXPERIMENTAL_FORM_ACTIONS
]
| undefined;
formEncType?: string | undefined;
formMethod?: string | undefined;
formNoValidate?: boolean | undefined;
formTarget?: string | undefined;
name?: string | undefined;
type?: "submit" | "reset" | "button" | undefined;
value?: string | readonly string[] | number | undefined;
}

interface CanvasHTMLAttributes<T> extends HTMLAttributes<T> {
 height?: number | string | undefined;
 width?: number | string | undefined;
}

interface ColHTMLAttributes<T> extends HTMLAttributes<T> {
 span?: number | undefined;
 width?: number | string | undefined;
}

interface ColgroupHTMLAttributes<T> extends HTMLAttributes<T> {
 span?: number | undefined;
}

interface DataHTMLAttributes<T> extends HTMLAttributes<T> {
 value?: string | readonly string[] | number | undefined;
}

interface DetailsHTMLAttributes<T> extends HTMLAttributes<T> {
 open?: boolean | undefined;
 name?: string | undefined;
}

interface DelHTMLAttributes<T> extends HTMLAttributes<T> {
 cite?: string | undefined;
 dateTime?: string | undefined;
}

interface DialogHTMLAttributes<T> extends HTMLAttributes<T> {
 onCancel?: ReactEventHandler<T> | undefined;
 onClose?: ReactEventHandler<T> | undefined;
 open?: boolean | undefined;
}

interface EmbedHTMLAttributes<T> extends HTMLAttributes<T> {
 height?: number | string | undefined;
```

```
 src?: string | undefined;
 type?: string | undefined;
 width?: number | string | undefined;
}

interface FieldsetHTMLAttributes<T> extends HTMLAttributes<T> {
 disabled?: boolean | undefined;
 form?: string | undefined;
 name?: string | undefined;
}

interface FormHTMLAttributes<T> extends HTMLAttributes<T> {
 acceptCharset?: string | undefined;
 action?:
 | string
 | undefined
 | ((formData: FormData) => void | Promise<void>)
 | DO_NOT_USE_OR_YOU_WILL_BE_FIRED_EXPERIMENTAL_FORM_ACTIONS[
 keyof DO_NOT_USE_OR_YOU_WILL_BE_FIRED_EXPERIMENTAL_FORM_ACTIONS
];
 autoComplete?: string | undefined;
 encType?: string | undefined;
 method?: string | undefined;
 name?: string | undefined;
 noValidate?: boolean | undefined;
 target?: string | undefined;
}

interface HtmlHTMLAttributes<T> extends HTMLAttributes<T> {
 manifest?: string | undefined;
}

interface IframeHTMLAttributes<T> extends HTMLAttributes<T> {
 allow?: string | undefined;
 allowFullScreen?: boolean | undefined;
 allowTransparency?: boolean | undefined;
 /** @deprecated */
 frameBorder?: number | string | undefined;
 height?: number | string | undefined;
 loading?: "eager" | "lazy" | undefined;
 /** @deprecated */
 marginHeight?: number | undefined;
 /** @deprecated */
 marginWidth?: number | undefined;
 name?: string | undefined;
 referrerPolicy?: HTMLAttributeReferrerPolicy | undefined;
 sandbox?: string | undefined;
 /** @deprecated */
 scrolling?: string | undefined;
}
```

```
 seamless?: boolean | undefined;
 src?: string | undefined;
 srcDoc?: string | undefined;
 width?: number | string | undefined;
}

interface DO_NOT_USE_OR_YOU_WILL_BE_FIRED_EXPERIMENTAL_IMG_SRC_TYPES {}

interface ImgHTMLAttributes<T> extends HTMLAttributes<T> {
 alt?: string | undefined;
 crossOrigin?: CrossOrigin;
 decoding?: "async" | "auto" | "sync" | undefined;
 fetchPriority?: "high" | "low" | "auto";
 height?: number | string | undefined;
 loading?: "eager" | "lazy" | undefined;
 referrerPolicy?: HTMLAttributeReferrerPolicy | undefined;
 sizes?: string | undefined;
 src?:
 | string
 | DO_NOT_USE_OR_YOU_WILL_BE_FIRED_EXPERIMENTAL_IMG_SRC_TYPES[
 keyof DO_NOT_USE_OR_YOU_WILL_BE_FIRED_EXPERIMENTAL_IMG_SRC_TYPES
]
 | undefined;
 srcSet?: string | undefined;
 useMap?: string | undefined;
 width?: number | string | undefined;
}

interface InsHTMLAttributes<T> extends HTMLAttributes<T> {
 cite?: string | undefined;
 dateTime?: string | undefined;
}

type HTMLInputTypeAttribute =
 | "button"
 | "checkbox"
 | "color"
 | "date"
 | "datetime-local"
 | "email"
 | "file"
 | "hidden"
 | "image"
 | "month"
 | "number"
 | "password"
 | "radio"
 | "range"
 | "reset"
```

```
| "search"
| "submit"
| "tel"
| "text"
| "time"
| "url"
| "week"
| (string & { }) ;

type AutoFillAddressKind = "billing" | "shipping";
type AutoFillBase = "" | "off" | "on";
type AutoFillContactField =
 | "email"
 | "tel"
 | "tel-area-code"
 | "tel-country-code"
 | "tel-extension"
 | "tel-local"
 | "tel-local-prefix"
 | "tel-local-suffix"
 | "tel-national";
type AutoFillContactKind = "home" | "mobile" | "work";
type AutoFillCredentialField = "webauthn";
type AutoFillNormalField =
 | "additional-name"
 | "address-level1"
 | "address-level2"
 | "address-level3"
 | "address-level4"
 | "address-line1"
 | "address-line2"
 | "address-line3"
 | "bday-day"
 | "bday-month"
 | "bday-year"
 | "cc-csc"
 | "cc-exp"
 | "cc-exp-month"
 | "cc-exp-year"
 | "cc-family-name"
 | "cc-given-name"
 | "cc-name"
 | "cc-number"
 | "cc-type"
 | "country"
 | "country-name"
 | "current-password"
 | "family-name"
 | "given-name"
```

```
| "honorific-prefix"
| "honorific-suffix"
| "name"
| "new-password"
| "one-time-code"
| "organization"
| "postal-code"
| "street-address"
| "transaction-amount"
| "transaction-currency"
| "username";
type OptionalPrefixToken<T extends string> = `${T} ` | "";
type OptionalPostfixToken<T extends string> = ` ${T}` | "";
type AutoFillField = AutoFillNormalField |
` ${OptionalPrefixToken<AutoFillContactKind>} ${AutoFillContactField}`;
type AutoFillSection = `section-${string}`;
type AutoFill =
| AutoFillBase
| `${OptionalPrefixToken<AutoFillSection>} ${OptionalPrefixToken<
 AutoFillAddressKind
>} ${AutoFillField}` ${OptionalPostfixToken<AutoFillCredentialField>}`;
type HTMLInputAutoCompleteAttribute = AutoFill | (string & {});

interface InputHTMLAttributes<T> extends HTMLAttributes<T> {
 accept?: string | undefined;
 alt?: string | undefined;
 autoComplete?: HTMLInputAutoCompleteAttribute | undefined;
 capture?: boolean | "user" | "environment" | undefined; // https://www.w3.org/TR/html-media-capture/#the-capture-attribute
 checked?: boolean | undefined;
 disabled?: boolean | undefined;
 form?: string | undefined;
 formAction?:
 | string
 | ((formData: FormData) => void | Promise<void>)
 | DO_NOT_USE_OR_YOU_WILL_BE_FIRED_EXPERIMENTAL_FORM_ACTIONS[
 keyof DO_NOT_USE_OR_YOU_WILL_BE_FIRED_EXPERIMENTAL_FORM_ACTIONS
]
 | undefined;
 formEncType?: string | undefined;
 formMethod?: string | undefined;
 formNoValidate?: boolean | undefined;
 formTarget?: string | undefined;
 height?: number | string | undefined;
 list?: string | undefined;
 max?: number | string | undefined;
 maxLength?: number | undefined;
 min?: number | string | undefined;
 minLength?: number | undefined;
```

```
multiple?: boolean | undefined;
name?: string | undefined;
pattern?: string | undefined;
placeholder?: string | undefined;
readOnly?: boolean | undefined;
required?: boolean | undefined;
size?: number | undefined;
src?: string | undefined;
step?: number | string | undefined;
type?: HTMLInputTypeAttribute | undefined;
value?: string | readonly string[] | number | undefined;
width?: number | string | undefined;

onChange?: ChangeEventHandler<T> | undefined;
}

interface KeygenHTMLAttributes<T> extends HTMLAttributes<T> {
challenge?: string | undefined;
disabled?: boolean | undefined;
form?: string | undefined;
keyType?: string | undefined;
keyParams?: string | undefined;
name?: string | undefined;
}

interface LabelHTMLAttributes<T> extends HTMLAttributes<T> {
form?: string | undefined;
htmlFor?: string | undefined;
}

interface LiHTMLAttributes<T> extends HTMLAttributes<T> {
value?: string | readonly string[] | number | undefined;
}

interface LinkHTMLAttributes<T> extends HTMLAttributes<T> {
as?: string | undefined;
blocking?: "render" | (string & {}) | undefined;
crossOrigin?: CrossOrigin;
fetchPriority?: "high" | "low" | "auto";
href?: string | undefined;
hrefLang?: string | undefined;
integrity?: string | undefined;
media?: string | undefined;
imageSrcSet?: string | undefined;
imageSizes?: string | undefined;
referrerPolicy?: HTMLAttributeReferrerPolicy | undefined;
sizes?: string | undefined;
type?: string | undefined;
charSet?: string | undefined;
```

```
// React props
precedence?: string | undefined;
}

interface MapHTMLAttributes<T> extends HTMLAttributes<T> {
 name?: string | undefined;
}

interface MenuHTMLAttributes<T> extends HTMLAttributes<T> {
 type?: string | undefined;
}

interface DO_NOT_USE_OR_YOU_WILL_BE_FIRED_EXPERIMENTAL_MEDIA_SRC_TYPES {}

interface MediaHTMLAttributes<T> extends HTMLAttributes<T> {
 autoPlay?: boolean | undefined;
 controls?: boolean | undefined;
 controlsList?: string | undefined;
 crossOrigin?: CrossOrigin;
 loop?: boolean | undefined;
 mediaGroup?: string | undefined;
 muted?: boolean | undefined;
 playsInline?: boolean | undefined;
 preload?: string | undefined;
 src?:
 | string
 | DO_NOT_USE_OR_YOU_WILL_BE_FIRED_EXPERIMENTAL_MEDIA_SRC_TYPES[
 keyof
DO_NOT_USE_OR_YOU_WILL_BE_FIRED_EXPERIMENTAL_MEDIA_SRC_TYPES
]
 | undefined;
}

interface MetaHTMLAttributes<T> extends HTMLAttributes<T> {
 charSet?: string | undefined;
 content?: string | undefined;
 httpEquiv?: string | undefined;
 media?: string | undefined;
 name?: string | undefined;
}

interface MeterHTMLAttributes<T> extends HTMLAttributes<T> {
 form?: string | undefined;
 high?: number | undefined;
 low?: number | undefined;
 max?: number | string | undefined;
 min?: number | string | undefined;
 optimum?: number | undefined;
```

```
 value?: string | readonly string[] | number | undefined;
 }

interface QuoteHTMLAttributes<T> extends HTMLAttributes<T> {
 cite?: string | undefined;
}

interface ObjectHTMLAttributes<T> extends HTMLAttributes<T> {
 classID?: string | undefined;
 data?: string | undefined;
 form?: string | undefined;
 height?: number | string | undefined;
 name?: string | undefined;
 type?: string | undefined;
 useMap?: string | undefined;
 width?: number | string | undefined;
 wmode?: string | undefined;
}

interface OlHTMLAttributes<T> extends HTMLAttributes<T> {
 reversed?: boolean | undefined;
 start?: number | undefined;
 type?: "1" | "a" | "A" | "i" | "I" | undefined;
}

interface OptgroupHTMLAttributes<T> extends HTMLAttributes<T> {
 disabled?: boolean | undefined;
 label?: string | undefined;
}

interface OptionHTMLAttributes<T> extends HTMLAttributes<T> {
 disabled?: boolean | undefined;
 label?: string | undefined;
 selected?: boolean | undefined;
 value?: string | readonly string[] | number | undefined;
}

interface OutputHTMLAttributes<T> extends HTMLAttributes<T> {
 form?: string | undefined;
 htmlFor?: string | undefined;
 name?: string | undefined;
}

interface ParamHTMLAttributes<T> extends HTMLAttributes<T> {
 name?: string | undefined;
 value?: string | readonly string[] | number | undefined;
}

interface ProgressHTMLAttributes<T> extends HTMLAttributes<T> {
```

```
 max?: number | string | undefined;
 value?: string | readonly string[] | number | undefined;
}

interface SlotHTMLAttributes<T> extends HTMLAttributes<T> {
 name?: string | undefined;
}

interface ScriptHTMLAttributes<T> extends HTMLAttributes<T> {
 async?: boolean | undefined;
 blocking?: "render" | (string & {}) | undefined;
 /** @deprecated */
 charSet?: string | undefined;
 crossOrigin?: CrossOrigin;
 defer?: boolean | undefined;
 integrity?: string | undefined;
 noModule?: boolean | undefined;
 referrerPolicy?: HTMLAttributeReferrerPolicy | undefined;
 src?: string | undefined;
 type?: string | undefined;
}

interface SelectHTMLAttributes<T> extends HTMLAttributes<T> {
 autoComplete?: string | undefined;
 disabled?: boolean | undefined;
 form?: string | undefined;
 multiple?: boolean | undefined;
 name?: string | undefined;
 required?: boolean | undefined;
 size?: number | undefined;
 value?: string | readonly string[] | number | undefined;
 onChange?: ChangeEventHandler<T> | undefined;
}

interface SourceHTMLAttributes<T> extends HTMLAttributes<T> {
 height?: number | string | undefined;
 media?: string | undefined;
 sizes?: string | undefined;
 src?: string | undefined;
 srcSet?: string | undefined;
 type?: string | undefined;
 width?: number | string | undefined;
}

interface StyleHTMLAttributes<T> extends HTMLAttributes<T> {
 blocking?: "render" | (string & {}) | undefined;
 media?: string | undefined;
 scoped?: boolean | undefined;
 type?: string | undefined;
}
```

```
// React props
href?: string | undefined;
precedence?: string | undefined;
}

interface TableHTMLAttributes<T> extends HTMLAttributes<T> {
 align?: "left" | "center" | "right" | undefined;
 bgcolor?: string | undefined;
 border?: number | undefined;
 cellPadding?: number | string | undefined;
 cellSpacing?: number | string | undefined;
 frame?: boolean | undefined;
 rules?: "none" | "groups" | "rows" | "columns" | "all" | undefined;
 summary?: string | undefined;
 width?: number | string | undefined;
}

interface TextareaHTMLAttributes<T> extends HTMLAttributes<T> {
 autoComplete?: string | undefined;
 cols?: number | undefined;
 dirName?: string | undefined;
 disabled?: boolean | undefined;
 form?: string | undefined;
 maxLength?: number | undefined;
 minLength?: number | undefined;
 name?: string | undefined;
 placeholder?: string | undefined;
 readOnly?: boolean | undefined;
 required?: boolean | undefined;
 rows?: number | undefined;
 value?: string | readonly string[] | number | undefined;
 wrap?: string | undefined;

 onChange?: ChangeEventHandler<T> | undefined;
}

interface TdHTMLAttributes<T> extends HTMLAttributes<T> {
 align?: "left" | "center" | "right" | "justify" | "char" | undefined;
 colSpan?: number | undefined;
 headers?: string | undefined;
 rowSpan?: number | undefined;
 scope?: string | undefined;
 abbr?: string | undefined;
 height?: number | string | undefined;
 width?: number | string | undefined;
 valign?: "top" | "middle" | "bottom" | "baseline" | undefined;
}
```

```
interface ThHTMLAttributes<T> extends HTMLAttributes<T> {
 align?: "left" | "center" | "right" | "justify" | "char" | undefined;
 colSpan?: number | undefined;
 headers?: string | undefined;
 rowSpan?: number | undefined;
 scope?: string | undefined;
 abbr?: string | undefined;
}

interface TimeHTMLAttributes<T> extends HTMLAttributes<T> {
 dateTime?: string | undefined;
}

interface TrackHTMLAttributes<T> extends HTMLAttributes<T> {
 default?: boolean | undefined;
 kind?: string | undefined;
 label?: string | undefined;
 src?: string | undefined;
 srcLang?: string | undefined;
}

interface VideoHTMLAttributes<T> extends MediaHTMLAttributes<T> {
 height?: number | string | undefined;
 playsInline?: boolean | undefined;
 poster?: string | undefined;
 width?: number | string | undefined;
 disablePictureInPicture?: boolean | undefined;
 disableRemotePlayback?: boolean | undefined;

 onResize?: ReactEventHandler<T> | undefined;
 onResizeCapture?: ReactEventHandler<T> | undefined;
}

// this list is "complete" in that it contains every SVG attribute
// that React supports, but the types can be improved.
// Full list here: https://facebook.github.io/react/docs/dom-elements.html
//
// The three broad type categories are (in order of restrictiveness):
// - "number | string"
// - "string"
// - union of string literals
interface SVGAttributes<T> extends AriaAttributes, DOMAttributes<T> {
 // React-specific Attributes
 suppressHydrationWarning?: boolean | undefined;

 // Attributes which also defined in HTMLAttributes
 // See comment in SVGDOMPropertyConfig.js
 className?: string | undefined;
 color?: string | undefined;
```

```
height?: number | string | undefined;
id?: string | undefined;
lang?: string | undefined;
max?: number | string | undefined;
media?: string | undefined;
method?: string | undefined;
min?: number | string | undefined;
name?: string | undefined;
style?: CSSProperties | undefined;
target?: string | undefined;
type?: string | undefined;
width?: number | string | undefined;

// Other HTML properties supported by SVG elements in browsers
role?: AriaRole | undefined;
tabIndex?: number | undefined;
crossOrigin?: CrossOrigin;

// SVG Specific attributes
accentHeight?: number | string | undefined;
accumulate?: "none" | "sum" | undefined;
additive?: "replace" | "sum" | undefined;
alignmentBaseline?:
 | "auto"
 | "baseline"
 | "before-edge"
 | "text-before-edge"
 | "middle"
 | "central"
 | "after-edge"
 | "text-after-edge"
 | "ideographic"
 | "alphabetic"
 | "hanging"
 | "mathematical"
 | "inherit"
 | undefined;
allowReorder?: "no" | "yes" | undefined;
alphabetic?: number | string | undefined;
amplitude?: number | string | undefined;
arabicForm?: "initial" | "medial" | "terminal" | "isolated" | undefined;
ascent?: number | string | undefined;
attributeName?: string | undefined;
attributeType?: string | undefined;
autoReverse?: Booleanish | undefined;
azimuth?: number | string | undefined;
baseFrequency?: number | string | undefined;
baselineShift?: number | string | undefined;
baseProfile?: number | string | undefined;
```

```
bbox?: number | string | undefined;
begin?: number | string | undefined;
bias?: number | string | undefined;
by?: number | string | undefined;
calcMode?: number | string | undefined;
capHeight?: number | string | undefined;
clip?: number | string | undefined;
clipPath?: string | undefined;
clipPathUnits?: number | string | undefined;
clipRule?: number | string | undefined;
colorInterpolation?: number | string | undefined;
colorInterpolationFilters?: "auto" | "sRGB" | "linearRGB" | "inherit" | undefined;
colorProfile?: number | string | undefined;
colorRendering?: number | string | undefined;
contentScriptType?: number | string | undefined;
contentStyleType?: number | string | undefined;
cursor?: number | string | undefined;
cx?: number | string | undefined;
cy?: number | string | undefined;
d?: string | undefined;
decelerate?: number | string | undefined;
descent?: number | string | undefined;
diffuseConstant?: number | string | undefined;
direction?: number | string | undefined;
display?: number | string | undefined;
divisor?: number | string | undefined;
dominantBaseline?: number | string | undefined;
dur?: number | string | undefined;
dx?: number | string | undefined;
dy?: number | string | undefined;
edgeMode?: number | string | undefined;
elevation?: number | string | undefined;
enableBackground?: number | string | undefined;
end?: number | string | undefined;
exponent?: number | string | undefined;
externalResourcesRequired?: Booleanish | undefined;
fill?: string | undefined;
fillOpacity?: number | string | undefined;
fillRule?: "nonzero" | "evenodd" | "inherit" | undefined;
filter?: string | undefined;
filterRes?: number | string | undefined;
filterUnits?: number | string | undefined;
floodColor?: number | string | undefined;
floodOpacity?: number | string | undefined;
focusable?: Booleanish | "auto" | undefined;
fontFamily?: string | undefined;
fontSize?: number | string | undefined;
fontSizeAdjust?: number | string | undefined;
```

```
fontStretch?: number | string | undefined;
fontStyle?: number | string | undefined;
fontVariant?: number | string | undefined;
fontWeight?: number | string | undefined;
format?: number | string | undefined;
fr?: number | string | undefined;
from?: number | string | undefined;
fx?: number | string | undefined;
fy?: number | string | undefined;
g1?: number | string | undefined;
g2?: number | string | undefined;
glyphName?: number | string | undefined;
glyphOrientationHorizontal?: number | string | undefined;
glyphOrientationVertical?: number | string | undefined;
glyphRef?: number | string | undefined;
gradientTransform?: string | undefined;
gradientUnits?: string | undefined;
hanging?: number | string | undefined;
horizAdvX?: number | string | undefined;
horizOriginX?: number | string | undefined;
href?: string | undefined;
ideographic?: number | string | undefined;
imageRendering?: number | string | undefined;
in2?: number | string | undefined;
in?: string | undefined;
intercept?: number | string | undefined;
k1?: number | string | undefined;
k2?: number | string | undefined;
k3?: number | string | undefined;
k4?: number | string | undefined;
k?: number | string | undefined;
kernelMatrix?: number | string | undefined;
kernelUnitLength?: number | string | undefined;
kerning?: number | string | undefined;
keyPoints?: number | string | undefined;
keySplines?: number | string | undefined;
keyTimes?: number | string | undefined;
lengthAdjust?: number | string | undefined;
letterSpacing?: number | string | undefined;
lightingColor?: number | string | undefined;
limitingConeAngle?: number | string | undefined;
local?: number | string | undefined;
markerEnd?: string | undefined;
markerHeight?: number | string | undefined;
markerMid?: string | undefined;
markerStart?: string | undefined;
markerUnits?: number | string | undefined;
markerWidth?: number | string | undefined;
mask?: string | undefined;
```

```
maskContentUnits?: number | string | undefined;
maskUnits?: number | string | undefined;
mathematical?: number | string | undefined;
mode?: number | string | undefined;
numOctaves?: number | string | undefined;
offset?: number | string | undefined;
opacity?: number | string | undefined;
operator?: number | string | undefined;
order?: number | string | undefined;
orient?: number | string | undefined;
orientation?: number | string | undefined;
origin?: number | string | undefined;
overflow?: number | string | undefined;
overlinePosition?: number | string | undefined;
overlineThickness?: number | string | undefined;
paintOrder?: number | string | undefined;
panose1?: number | string | undefined;
path?: string | undefined;
pathLength?: number | string | undefined;
patternContentUnits?: string | undefined;
patternTransform?: number | string | undefined;
patternUnits?: string | undefined;
pointerEvents?: number | string | undefined;
points?: string | undefined;
pointsAtX?: number | string | undefined;
pointsAtY?: number | string | undefined;
pointsAtZ?: number | string | undefined;
preserveAlpha?: Booleanish | undefined;
preserveAspectRatio?: string | undefined;
primitiveUnits?: number | string | undefined;
r?: number | string | undefined;
radius?: number | string | undefined;
refX?: number | string | undefined;
refY?: number | string | undefined;
renderingIntent?: number | string | undefined;
repeatCount?: number | string | undefined;
repeatDur?: number | string | undefined;
requiredExtensions?: number | string | undefined;
requiredFeatures?: number | string | undefined;
restart?: number | string | undefined;
result?: string | undefined;
rotate?: number | string | undefined;
rx?: number | string | undefined;
ry?: number | string | undefined;
scale?: number | string | undefined;
seed?: number | string | undefined;
shapeRendering?: number | string | undefined;
slope?: number | string | undefined;
spacing?: number | string | undefined;
```

```
specularConstant?: number | string | undefined;
specularExponent?: number | string | undefined;
speed?: number | string | undefined;
spreadMethod?: string | undefined;
startOffset?: number | string | undefined;
stdDeviation?: number | string | undefined;
stemh?: number | string | undefined;
stemv?: number | string | undefined;
stitchTiles?: number | string | undefined;
stopColor?: string | undefined;
stopOpacity?: number | string | undefined;
strikethroughPosition?: number | string | undefined;
strikethroughThickness?: number | string | undefined;
string?: number | string | undefined;
stroke?: string | undefined;
strokeDasharray?: string | number | undefined;
strokeDashoffset?: string | number | undefined;
strokeLinecap?: "butt" | "round" | "square" | "inherit" | undefined;
strokeLinejoin?: "miter" | "round" | "bevel" | "inherit" | undefined;
strokeMiterlimit?: number | string | undefined;
strokeOpacity?: number | string | undefined;
strokeWidth?: number | string | undefined;
surfaceScale?: number | string | undefined;
systemLanguage?: number | string | undefined;
tableValues?: number | string | undefined;
targetX?: number | string | undefined;
targetY?: number | string | undefined;
textAnchor?: string | undefined;
textDecoration?: number | string | undefined;
textLength?: number | string | undefined;
textRendering?: number | string | undefined;
to?: number | string | undefined;
transform?: string | undefined;
u1?: number | string | undefined;
u2?: number | string | undefined;
underlinePosition?: number | string | undefined;
underlineThickness?: number | string | undefined;
unicode?: number | string | undefined;
unicodeBidi?: number | string | undefined;
unicodeRange?: number | string | undefined;
unitsPerEm?: number | string | undefined;
vAlphabetic?: number | string | undefined;
values?: string | undefined;
vectorEffect?: number | string | undefined;
version?: string | undefined;
vertAdvY?: number | string | undefined;
vertOriginX?: number | string | undefined;
vertOriginY?: number | string | undefined;
vHanging?: number | string | undefined;
```

```
vIdeographic?: number | string | undefined;
viewBox?: string | undefined;
viewTarget?: number | string | undefined;
visibility?: number | string | undefined;
vMathematical?: number | string | undefined;
widths?: number | string | undefined;
wordSpacing?: number | string | undefined;
writingMode?: number | string | undefined;
x1?: number | string | undefined;
x2?: number | string | undefined;
x?: number | string | undefined;
xChannelSelector?: string | undefined;
xHeight?: number | string | undefined;
xlinkActuate?: string | undefined;
xlinkArcrole?: string | undefined;
xlinkHref?: string | undefined;
xlinkRole?: string | undefined;
xlinkShow?: string | undefined;
xlinkTitle?: string | undefined;
xlinkType?: string | undefined;
xmlBase?: string | undefined;
xmlLang?: string | undefined;
xmlns?: string | undefined;
xmlnsXlink?: string | undefined;
xmlSpace?: string | undefined;
y1?: number | string | undefined;
y2?: number | string | undefined;
y?: number | string | undefined;
yChannelSelector?: string | undefined;
z?: number | string | undefined;
zoomAndPan?: string | undefined;
}

interface WebViewHTMLAttributes<T> extends HTMLAttributes<T> {
 allowFullScreen?: boolean | undefined;
 allowpopups?: boolean | undefined;
 autosize?: boolean | undefined;
 blinkfeatures?: string | undefined;
 disableblinkfeatures?: string | undefined;
 disableguestresize?: boolean | undefined;
 disablewebsecurity?: boolean | undefined;
 guestinstance?: string | undefined;
 httpreferrer?: string | undefined;
 nodeintegration?: boolean | undefined;
 partition?: string | undefined;
 plugins?: boolean | undefined;
 preload?: string | undefined;
 src?: string | undefined;
 useragent?: string | undefined;
```

```
 webpreferences?: string | undefined;
}

// TODO: Move to react-dom
type HTMLElementType =
| "a"
| "abbr"
| "address"
| "area"
| "article"
| "aside"
| "audio"
| "b"
| "base"
| "bdi"
| "bdo"
| "big"
| "blockquote"
| "body"
| "br"
| "button"
| "canvas"
| "caption"
| "center"
| "cite"
| "code"
| "col"
| "colgroup"
| "data"
| "datalist"
| "dd"
| "del"
| "details"
| "dfn"
| "dialog"
| "div"
| "dl"
| "dt"
| "em"
| "embed"
| "fieldset"
| "figcaption"
| "figure"
| "footer"
| "form"
| "h1"
| "h2"
| "h3"
| "h4"
```

```
| "h5"
| "h6"
| "head"
| "header"
| "hgroup"
| "hr"
| "html"
| "i"
| "iframe"
| "img"
| "input"
| "ins"
| "kbd"
| "keygen"
| "label"
| "legend"
| "li"
| "link"
| "main"
| "map"
| "mark"
| "menu"
| "menuitem"
| "meta"
| "meter"
| "nav"
| "noscript"
| "object"
| "ol"
| "optgroup"
| "option"
| "output"
| "p"
| "param"
| "picture"
| "pre"
| "progress"
| "q"
| "rp"
| "rt"
| "ruby"
| "s"
| "samp"
| "search"
| "slot"
| "script"
| "section"
| "select"
| "small"
```

```
| "source"
| "span"
| "strong"
| "style"
| "sub"
| "summary"
| "sup"
| "table"
| "template"
| "tbody"
| "td"
| "textarea"
| "tfoot"
| "th"
| "thead"
| "time"
| "title"
| "tr"
| "track"
| "u"
| "ul"
| "var"
| "video"
| "wbr"
| "webview";

// TODO: Move to react-dom
type SVGElementType =
 | "animate"
 | "circle"
 | "clipPath"
 | "defs"
 | "desc"
 | "ellipse"
 | "feBlend"
 | "feColorMatrix"
 | "feComponentTransfer"
 | "feComposite"
 | "feConvolveMatrix"
 | "feDiffuseLighting"
 | "feDisplacementMap"
 | "feDistantLight"
 | "feDropShadow"
 | "feFlood"
 | "feFuncA"
 | "feFuncB"
 | "feFuncG"
 | "feFuncR"
 | "feGaussianBlur"
```

```
| "feImage"
| "feMerge"
| "feMergeNode"
| "feMorphology"
| "feOffset"
| "fePointLight"
| "feSpecularLighting"
| "feSpotLight"
| "feTile"
| "feTurbulence"
| "filter"
| "foreignObject"
| "g"
| "image"
| "line"
| "linearGradient"
| "marker"
| "mask"
| "metadata"
| "path"
| "pattern"
| "polygon"
| "polyline"
| "radialGradient"
| "rect"
| "stop"
| "svg"
| "switch"
| "symbol"
| "text"
| "textPath"
| "tspan"
| "use"
| "view";

// -----
// Browser Interfaces
// https://github.com/nikeeee/2048-typescript/blob/master/2048/js/touch.d.ts
// -----

interface AbstractView {
 styleMedia: StyleMedia;
 document: Document;
}

interface Touch {
 identifier: number;
 target: EventTarget;
 screenX: number;
```

```

 screenY: number;
 clientX: number;
 clientY: number;
 pageX: number;
 pageY: number;
 }

interface TouchList {
 [index: number]: Touch;
 length: number;
 item(index: number): Touch;
 identifiedTouch(identifier: number): Touch;
}

// -----
// Error Interfaces
// -----
interface ErrorInfo {
 /**
 * Captures which component contained the exception, and its ancestors.
 */
 componentStack?: string | null;
 digest?: string | null;
}

// Keep in sync with JSX namespace in ./jsx-runtime.d.ts and
// ./jsx-dev-runtime.d.ts
namespace JSX {
 // We don't just alias React.ElementType because React.ElementType
 // historically does more than we need it to.
 // E.g. it also contains .propTypes and so TS also verifies the declared
 // props type does match the declared .propTypes.
 // But if libraries declared their .propTypes but not props type,
 // or they mismatch, you won't be able to use the class component
 // as a JSX.ElementType.
 // We could fix this everywhere but we're ultimately not interested in
 // .propTypes assignability so we might as well drop it entirely here to
 // reduce the work of the type-checker.
 // TODO: Check impact of making React.ElementType<P = any> =
React.JSXElementConstructor<P>
 type ElementType = string | React.JSXElementConstructor<any>;
 interface Element extends React.ReactElement<any, any> {}
 interface ElementClass extends React.Component<any> {
 render(): React.ReactNode;
 }
 interface ElementAttributesProperty {
 props: {};
 }
 interface ElementChildrenAttribute {
}
}

```

```
 children: {};
 }

 // We can't recurse forever because `type` can't be self-referential;
 // let's assume it's reasonable to do a single React.lazy() around a
single React.memo() / vice-versa
 type LibraryManagedAttributes<C, P> = C extends
 React.MemoExoticComponent<infer T> | React.LazyExoticComponent<infer
T>
 ? T extends React.MemoExoticComponent<infer U> |
React.LazyExoticComponent<infer U>
 ? ReactManagedAttributes<U, P>
 : ReactManagedAttributes<T, P>
 : ReactManagedAttributes<C, P>;

 interface IntrinsicAttributes extends React.Attributes {}
 interface IntrinsicClassAttributes<T> extends React.ClassAttributes<T>
{ }

 interface IntrinsicElements {
 // HTML
 a:
React.DetailedHTMLProps<React.AnchorHTMLAttributes<HTMLAnchorElement>,
HTMLAnchorElement>;
 abbr: React.DetailedHTMLProps<React.HTMLAttributes<HTMLElement>,
HTMLElement>;
 address: React.DetailedHTMLProps<React.HTMLAttributes<HTMLElement>,
HTMLElement>;
 area:
React.DetailedHTMLProps<React.AreaHTMLAttributes<HTMLAreaElement>,
HTMLAreaElement>;
 article: React.DetailedHTMLProps<React.HTMLAttributes<HTMLElement>,
HTMLElement>;
 aside: React.DetailedHTMLProps<React.HTMLAttributes<HTMLElement>,
HTMLElement>;
 audio:
React.DetailedHTMLProps<React.AudioHTMLAttributes<HTMLAudioElement>,
HTMLAudioElement>;
 b: React.DetailedHTMLProps<React.HTMLAttributes<HTMLElement>,
HTMLElement>;
 base:
React.DetailedHTMLProps<React.BaseHTMLAttributes<HTMLBaseElement>,
HTMLBaseElement>;
 bdi: React.DetailedHTMLProps<React.HTMLAttributes<HTMLElement>,
HTMLElement>;
 bdo: React.DetailedHTMLProps<React.HTMLAttributes<HTMLElement>,
HTMLElement>;
 big: React.DetailedHTMLProps<React.HTMLAttributes<HTMLElement>,
HTMLElement>;
```

```
 blockquote:
React.DetailedHTMLProps<React.BlockquoteHTMLAttributes<HTMLQuoteElement>,
HTMLQuoteElement>;
 body: React.DetailedHTMLProps<React.HTMLAttributes<HTMLBodyElement>,
HTMLBodyElement>;
 br: React.DetailedHTMLProps<React.HTMLAttributes<HTMLBRElement>,
HTMLBRElement>;
 button:
React.DetailedHTMLProps<React.ButtonHTMLAttributes<HTMLButtonElement>,
HTMLButtonElement>;
 canvas:
React.DetailedHTMLProps<React.CanvasHTMLAttributes<HTMLCanvasElement>,
HTMLCanvasElement>;
 caption: React.DetailedHTMLProps<React.HTMLAttributes<HTMLElement>,
HTMLElement>;
 center: React.DetailedHTMLProps<React.HTMLAttributes<HTMLElement>,
HTMLElement>;
 cite: React.DetailedHTMLProps<React.HTMLAttributes<HTMLElement>,
HTMLElement>;
 code: React.DetailedHTMLProps<React.HTMLAttributes<HTMLElement>,
HTMLElement>;
 col:
React.DetailedHTMLProps<React.ColHTMLAttributes<HTMLTableColElement>,
HTMLTableColElement>;
 colgroup:
React.DetailedHTMLProps<React.ColgroupHTMLAttributes<HTMLTableColElement>,
HTMLTableColElement>;
 data:
React.DetailedHTMLProps<React.DataHTMLAttributes<HTMLDataElement>,
HTMLDataElement>;
 datalist:
React.DetailedHTMLProps<React.HTMLAttributes<HTMLDataListElement>,
HTMLDataListElement>;
 dd: React.DetailedHTMLProps<React.HTMLAttributes<HTMLElement>,
HTMLElement>;
 del:
React.DetailedHTMLProps<React.DelHTMLAttributes<HTMLModElement>,
HTMLModElement>;
 details:
React.DetailedHTMLProps<React.DetailsHTMLAttributes<HTMLDetailsElement>,
HTMLDetailsElement>;
 dfn: React.DetailedHTMLProps<React.HTMLAttributes<HTMLElement>,
HTMLElement>;
 dialog:
React.DetailedHTMLProps<React.DialogHTMLAttributes<HTMLDialogElement>,
HTMLDialogElement>;
 div: React.DetailedHTMLProps<React.HTMLAttributes<HTMLDivElement>,
HTMLDivElement>;
```

```
 dl: React.DetailedHTMLProps<React.HTMLAttributes<HTMLDListElement>,
HTMLDListElement>;
 dt: React.DetailedHTMLProps<React.HTMLAttributes<HTMLElement>,
HTMLElement>;
 em: React.DetailedHTMLProps<React.HTMLAttributes<HTMLElement>,
HTMLElement>;
 embed:
React.DetailedHTMLProps<React.EmbedHTMLAttributes<HTMLEmbedElement>,
HTMLEmbedElement>;
 fieldset:
React.DetailedHTMLProps<React.FieldsetHTMLAttributes<HTMLFieldSetElement>,
HTMLFieldSetElement>;
 figcaption:
React.DetailedHTMLProps<React.HTMLAttributes<HTMLElement>, HTMLElement>;
 figure: React.DetailedHTMLProps<React.HTMLAttributes<HTMLElement>,
HTMLElement>;
 footer: React.DetailedHTMLProps<React.HTMLAttributes<HTMLElement>,
HTMLElement>;
 form:
React.DetailedHTMLProps<React.FormHTMLAttributes<HTMLFormElement>,
HTMLFormElement>;
 h1:
React.DetailedHTMLProps<React.HTMLAttributes<HTMLHeadingElement>,
HTMLHeadingElement>;
 h2:
React.DetailedHTMLProps<React.HTMLAttributes<HTMLHeadingElement>,
HTMLHeadingElement>;
 h3:
React.DetailedHTMLProps<React.HTMLAttributes<HTMLHeadingElement>,
HTMLHeadingElement>;
 h4:
React.DetailedHTMLProps<React.HTMLAttributes<HTMLHeadingElement>,
HTMLHeadingElement>;
 h5:
React.DetailedHTMLProps<React.HTMLAttributes<HTMLHeadingElement>,
HTMLHeadingElement>;
 h6:
React.DetailedHTMLProps<React.HTMLAttributes<HTMLHeadingElement>,
HTMLHeadingElement>;
 head: React.DetailedHTMLProps<React.HTMLAttributes<HTMLHeadElement>,
HTMLHeadElement>;
 header: React.DetailedHTMLProps<React.HTMLAttributes<HTMLElement>,
HTMLElement>;
 hgroup: React.DetailedHTMLProps<React.HTMLAttributes<HTMLElement>,
HTMLElement>;
 hr: React.DetailedHTMLProps<React.HTMLAttributes<HTMLHRElement>,
HTMLHRElement>;
```

```
 html:
React.DetailedHTMLProps<React.HtmlHTMLAttributes<HTMLHtmlElement>,
HTMLHtmlElement>;
 i: React.DetailedHTMLProps<React.HTMLAttributes<HTMLElement>,
HTMLElement>;
 iframe:
React.DetailedHTMLProps<React.IframeHTMLAttributes<HTMLIFrameElement>,
HTMLIFrameElement>;
 img:
React.DetailedHTMLProps<React.ImgHTMLAttributes<HTMLImageElement>,
HTMLImageElement>;
 input:
React.DetailedHTMLProps<React.InputHTMLAttributes<HTMLInputElement>,
HTMLInputElement>;
 ins:
React.DetailedHTMLProps<React.InsHTMLAttributes<HTMLModElement>,
HTMLModElement>;
 kbd: React.DetailedHTMLProps<React.HTMLAttributes<HTMLElement>,
HTMLElement>;
 keygen:
React.DetailedHTMLProps<React.KeygenHTMLAttributes<HTMLElement>, HTMLElement>;
 label:
React.DetailedHTMLProps<React.LabelHTMLAttributes<HTMLLabelElement>,
HTMLLabelElement>;
 legend:
React.DetailedHTMLProps<React.HTMLAttributes<HTMLLegendElement>,
HTMLLegendElement>;
 li: React.DetailedHTMLProps<React.LiHTMLAttributes<HTMLLIElement>,
HTMLLIElement>;
 link:
React.DetailedHTMLProps<React.LinkHTMLAttributes<HTMLLinkElement>,
HTMLLinkElement>;
 main: React.DetailedHTMLProps<React.HTMLAttributes<HTMLElement>,
HTMLElement>;
 map:
React.DetailedHTMLProps<React.MapHTMLAttributes<HTMLMapElement>,
HTMLMapElement>;
 mark: React.DetailedHTMLProps<React.HTMLAttributes<HTMLElement>,
HTMLElement>;
 menu: React.DetailedHTMLProps<React.MenuHTMLAttributes<HTMLElement>,
HTMLElement>;
 menuitem: React.DetailedHTMLProps<React.HTMLAttributes<HTMLElement>,
HTMLElement>;
 meta:
React.DetailedHTMLProps<React.MetaHTMLAttributes<HTMLMetaElement>,
HTMLMetaElement>;
 meter:
React.DetailedHTMLProps<React.MeterHTMLAttributes<HTMLMeterElement>,
HTMLMeterElement>;
```

```
 nav: React.DetailedHTMLProps<React.HTMLAttributes<HTMLElement>,
HTMLElement>;
 noindex: React.DetailedHTMLProps<React.HTMLAttributes<HTMLElement>,
HTMLElement>;
 noscript: React.DetailedHTMLProps<React.HTMLAttributes<HTMLElement>,
HTMLElement>;
 object:
React.DetailedHTMLProps<React.ObjectHTMLAttributes<HTMLObjectElement>,
HTMLObjectElement>;
 ol:
React.DetailedHTMLProps<React.OlHTMLAttributes<HTMLListElement>,
HTMLListElement>;
 optgroup:
React.DetailedHTMLProps<React.OptgroupHTMLAttributes<HTMLOptGroupElement>,
HTMLOptGroupElement>;
 option:
React.DetailedHTMLProps<React.OptionHTMLAttributes<HTMLOptionElement>,
HTMLOptionElement>;
 output:
React.DetailedHTMLProps<React.OutputHTMLAttributes<HTMLOutputElement>,
HTMLOutputElement>;
 p:
React.DetailedHTMLProps<React.HTMLAttributes<HTMLParagraphElement>,
HTMLParagraphElement>;
 param:
React.DetailedHTMLProps<React.ParamHTMLAttributes<HTMLParamElement>,
HTMLParamElement>;
 picture: React.DetailedHTMLProps<React.HTMLAttributes<HTMLElement>,
HTMLElement>;
 pre: React.DetailedHTMLProps<React.HTMLAttributes<HTMLPreElement>,
HTMLPreElement>;
 progress:
React.DetailedHTMLProps<React.ProgressHTMLAttributes<HTMLProgressElement>,
HTMLProgressElement>;
 q:
React.DetailedHTMLProps<React.QuoteHTMLAttributes<HTMLQuoteElement>,
HTMLQuoteElement>;
 rp: React.DetailedHTMLProps<React.HTMLAttributes<HTMLElement>,
HTMLElement>;
 rt: React.DetailedHTMLProps<React.HTMLAttributes<HTMLElement>,
HTMLElement>;
 ruby: React.DetailedHTMLProps<React.HTMLAttributes<HTMLElement>,
HTMLElement>;
 s: React.DetailedHTMLProps<React.HTMLAttributes<HTMLElement>,
HTMLElement>;
 samp: React.DetailedHTMLProps<React.HTMLAttributes<HTMLElement>,
HTMLElement>;
 search: React.DetailedHTMLProps<React.HTMLAttributes<HTMLElement>,
HTMLElement>;
```

```
slot:
React.DetailedHTMLProps<React.SlotHTMLAttributes<HTMLSlotElement>,
HTMLSlotElement>;
 script:
React.DetailedHTMLProps<React.ScriptHTMLAttributes<HTMLScriptElement>,
HTMLScriptElement>;
 section: React.DetailedHTMLProps<React.HTMLAttributes<HTMLElement>,
HTMLElement>;
 select:
React.DetailedHTMLProps<React.SelectHTMLAttributes<HTMLSelectElement>,
HTMLSelectElement>;
 small: React.DetailedHTMLProps<React.HTMLAttributes<HTMLElement>,
HTMLElement>;
 source:
React.DetailedHTMLProps<React.SourceHTMLAttributes<HTMLSourceElement>,
HTMLSourceElement>;
 span: React.DetailedHTMLProps<React.HTMLAttributes<HTMLSpanElement>,
HTMLSpanElement>;
 strong: React.DetailedHTMLProps<React.HTMLAttributes<HTMLElement>,
HTMLElement>;
 style:
React.DetailedHTMLProps<React.StyleHTMLAttributes<HTMLStyleElement>,
HTMLStyleElement>;
 sub: React.DetailedHTMLProps<React.HTMLAttributes<HTMLElement>,
HTMLElement>;
 summary: React.DetailedHTMLProps<React.HTMLAttributes<HTMLElement>,
HTMLElement>;
 sup: React.DetailedHTMLProps<React.HTMLAttributes<HTMLElement>,
HTMLElement>;
 table:
React.DetailedHTMLProps<React.TableHTMLAttributes<HTMLTableElement>,
HTMLTableElement>;
 template:
React.DetailedHTMLProps<React.HTMLAttributes<HTMLTemplateElement>,
HTMLTemplateElement>;
 tbody:
React.DetailedHTMLProps<React.HTMLAttributes<HTMLTableSectionElement>,
HTMLTableSectionElement>;
 td:
React.DetailedHTMLProps<React.TdHTMLAttributes<HTMLTableDataCellElement>,
HTMLTableDataCellElement>;
 textarea:
React.DetailedHTMLProps<React.TextareaHTMLAttributes<HTMLTextAreaElement>,
HTMLTextAreaElement>;
 tfoot:
React.DetailedHTMLProps<React.HTMLAttributes<HTMLTableSectionElement>,
HTMLTableSectionElement>;
```

```
 th:
React.DetailedHTMLProps<React.ThHTMLAttributes<HTMLTableHeaderCellElement>,
HTMLTableHeaderCellElement>;
 thead:
React.DetailedHTMLProps<React.HTMLAttributes<HTMLTableSectionElement>,
HTMLTableSectionElement>;
 time:
React.DetailedHTMLProps<React.TimeHTMLAttributes<HTMLTimeElement>,
HTMLTimeElement>;
 title:
React.DetailedHTMLProps<React.HTMLAttributes<HTMLTitleElement>,
HTMLTitleElement>;
 tr:
React.DetailedHTMLProps<React.HTMLAttributes<HTMLTableRowElement>,
HTMLTableRowElement>;
 track:
React.DetailedHTMLProps<React.TrackHTMLAttributes<HTMLTrackElement>,
HTMLTrackElement>;
 u: React.DetailedHTMLProps<React.HTMLAttributes<HTMLElement>,
HTMLElement>;
 ul: React.DetailedHTMLProps<React.HTMLAttributes<HTMLULListElement>,
HTMLULListElement>;
 "var": React.DetailedHTMLProps<React.HTMLAttributes<HTMLElement>,
HTMLElement>;
 video:
React.DetailedHTMLProps<React.VideoHTMLAttributes<HTMLVideoElement>,
HTMLVideoElement>;
 wbr: React.DetailedHTMLProps<React.HTMLAttributes<HTMLElement>,
HTMLElement>;
 webview:
React.DetailedHTMLProps<React.WebViewHTMLAttributes<HTMLWebViewElement>,
HTMLWebViewElement>;

 // SVG
 svg: React.SVGProps<SVGSVGElement>

 animate: React.SVGProps<SVGElement>; // TODO: It is
SVGAnimateElement but is not in TypeScript's lib.dom.d.ts for now.
 animateMotion: React.SVGProps<SVGElement>;
 animateTransform: React.SVGProps<SVGElement>; // TODO: It is
SVGAnimateTransformElement but is not in TypeScript's lib.dom.d.ts for now.
 circle: React.SVGProps<SVGCircleElement>;
 clipPath: React.SVGProps<SVGClipPathElement>;
 defs: React.SVGProps<SVGDefsElement>;
 desc: React.SVGProps<SVGDescElement>;
 ellipse: React.SVGProps<SVGEllipseElement>;
 feBlend: React.SVGProps<SVGFEBlendElement>;
 feColorMatrix: React.SVGProps<SVGFEColorMatrixElement>;
 feComponentTransfer: React.SVGProps<SVGFEComponentTransferElement>;
```

```
 feComposite: React.SVGProps<SVGFECompositeElement>;
 feConvolveMatrix: React.SVGProps<SVGFEConvolveMatrixElement>;
 feDiffuseLighting: React.SVGProps<SVGFEDiffuseLightingElement>;
 feDisplacementMap: React.SVGProps<SVGFEDisplacementMapElement>;
 feDistantLight: React.SVGProps<SVGFEDistantLightElement>;
 feDropShadow: React.SVGProps<SVGFEDropShadowElement>;
 feFlood: React.SVGProps<SVGFEFloodElement>;
 feFuncA: React.SVGProps<SVGFEEFuncAElement>;
 feFuncB: React.SVGProps<SVGFEEFuncBElement>;
 feFuncG: React.SVGProps<SVGFEEFuncGElement>;
 feFuncR: React.SVGProps<SVGFEEFuncRElement>;
 feGaussianBlur: React.SVGProps<SVGFEGaussianBlurElement>;
 feImage: React.SVGProps<SVGFEOImageElement>;
 feMerge: React.SVGProps<SVGFEMergeElement>;
 feMergeNode: React.SVGProps<SVGFEMergeNodeElement>;
 feMorphology: React.SVGProps<SVGFEMorphologyElement>;
 feOffset: React.SVGProps<SVGFEOffsetElement>;
 fePointLight: React.SVGProps<SVGFEPointLightElement>;
 feSpecularLighting: React.SVGProps<SVGFESpecularLightingElement>;
 feSpotLight: React.SVGProps<SVGFESpotLightElement>;
 feTile: React.SVGProps<SVGFETileElement>;
 feTurbulence: React.SVGProps<SVGFETurbulenceElement>;
 filter: React.SVGProps<SVGFILTERElement>;
 foreignObject: React.SVGProps<SVGForeignObjectElement>;
 g: React.SVGProps<SVGGElement>;
 image: React.SVGProps<SVGImageElement>;
 line: React.SVGLineElementAttributes<SVGLineElement>;
 linearGradient: React.SVGProps<SVGLinearGradientElement>;
 marker: React.SVGProps<SVGMarkerElement>;
 mask: React.SVGProps<SVGMaskElement>;
 metadata: React.SVGProps<SVGMetadataElement>;
 mpath: React.SVGProps<SVGELEMENT>;
 path: React.SVGProps<SVGPPathElement>;
 pattern: React.SVGProps<SVGPATTERNElement>;
 polygon: React.SVGProps<SVGPolygonElement>;
 polyline: React.SVGProps<SVGPolylineElement>;
 radialGradient: React.SVGProps<SVGRadialGradientElement>;
 rect: React.SVGProps<SVGRectElement>;
 set: React.SVGProps<SVGSetElement>;
 stop: React.SVGProps<SVGStopElement>;
 switch: React.SVGProps<SVGSwitchElement>;
 symbol: React.SVGProps<SVGSymbolElement>;
 text: React.SVGTextElementAttributes<SVGTextElement>;
 textPath: React.SVGProps<SVGTextPathElement>;
 tspan: React.SVGProps<SVGTSpanElement>;
 use: React.SVGProps<SVGUseElement>;
 view: React.SVGProps<SVGViewElement>;
 }
}
```

```

}

type InexactPartial<T> = { [K in keyof T]?: T[K] | undefined };

// Any prop that has a default prop becomes optional, but its type is unchanged
// Undeclared default props are augmented into the resulting allowable
// attributes
// If declared props have indexed properties, ignore default props entirely as
// keyof gets widened
// Wrap in an outer-level conditional type to allow distribution over props
// that are unions
type Defaultize<P, D> = P extends any ? string extends keyof P ? P
:
 & Pick<P, Exclude<keyof P, keyof D>>
 & InexactPartial<Pick<P, Extract<keyof P, keyof D>>>
 & InexactPartial<Pick<D, Exclude<keyof D, keyof P>>>
: never;

type ReactManagedAttributes<C, P> = C extends { defaultProps: infer D } ? Defaultize<P, D>
: P;
import React, { useState, ChangeEvent, KeyboardEvent, useEffect } from 'react';

interface OTPInputProps {
 length: number;
 onChange: (otp: string) => void;
 error?: boolean;
}

const OTPInput: React.FC<OTPInputProps> = ({ length, onChange, error = false }) => {
 const [otp, setOtp] = useState<string[]>(Array(length).fill(''));

 useEffect(() => {
 onChange(otp.join(''));
 }, [otp, onChange]);

 const handleChange = (e: ChangeEvent<HTMLInputElement>, index: number) => {
 const value = e.target.value.replace(/[^0-9]/g, '');
 if (value.length <= 1) {
 const newOtp = [...otp];
 newOtp[index] = value;
 setOtp(newOtp);

 // Move to the next input field if there is a next one
 if (value && index < otp.length - 1) {
 const nextInput = document.getElementById(`otp-input-${index + 1}`);
 if (nextInput) nextInput.focus();
 }
 }
 };
}

```

```

 }
 }
};

const handleKeyDown = (e: KeyboardEvent<HTMLInputElement>, index: number) =>
{
 if (e.key === 'Backspace' && !otp[index] && index > 0) {
 const prevInput = document.getElementById(`otp-input-${index - 1}`);
 if (prevInput) prevInput.focus();
 }
};

return (
 <div className='flex gap-3'>
 {otp.map((item, index) => (
 <input
 key={index}
 id={`otp-input-${index}`}
 type="text"
 value={item}
 onChange={(e) => handleChange(e, index)}
 onKeyDown={(e) => handleKeyDown(e, index)}
 maxLength={1}
 className={`${mt-1} block w-full px-3 py-2 border ${error ?
'border-red-500' : 'border-gray-300'}
 rounded-md shadow-sm focus:outline-none sm:text-sm
h-10 w-10 flex justify-center items-center text-center focus:border-teal-500`}
 />
)));
 </div>
);
};

export default OTPInput;
import React from 'react'
import { useAppSelector } from '../../../../../Redux Toolkit/Store'
import AddressCard from '../Checkout/AddressCard'
import UserAddressCard from './UserAddressCard'

const Addresses = () => {
 const { user } = useAppSelector(store => store)
 return (
 <>
 <div className='space-y-3'>
 {user.user?.addresses?.map((item, index) =>
 <UserAddressCard
 key={item.id}
 item={item} />))
 </div>
);
};

```

```

 </>
)
}

export default Addresses
import React, { useEffect } from 'react'
import OrderItemCard from './OrderItemCard'
import { useDispatch, useSelector } from '../../../../../Redux Toolkit/Store';
import { fetchUserOrderHistory } from '../../../../../Redux
Toolkit/Customer/OrderSlice';
import { Button } from '@mui/material';

const Order = () => {
 const dispatch = useDispatch()
 const { cart, auth, orders } = useSelector(store => store);

 useEffect(() => {
 dispatch(fetchUserOrderHistory(localStorage.getItem("jwt") || ""))
 }, [auth.jwt])
 return (
 <div className='text-sm min-h-screen'>
 <div className='pb-5'>
 <h1 className='font-semibold'>All orders
 </h1>
 <p>from anytime</p>
 </div>
 <div className='space-y-2'>

{orders?.orders?.map((order)=>order?.orderItems.map((item)=><OrderItemCard
item={item} order={order}/>))}

 </div>
 </div>
)
}

export default Order
import { Box, Button, Divider } from '@mui/material'
import React, { useEffect } from 'react'
import PaymentsIcon from '@mui/icons-material/Payments';
import OrderStepper from './OrderStepper';
import { useDispatch, useSelector } from '../../../../../Redux Toolkit/Store';
import { cancelOrder, fetchOrderById, fetchOrderItemById } from '../../../../../Redux
Toolkit/Customer/OrderSlice';
import { useNavigate, useParams } from 'react-router-dom';

const OrderDetails = () => {
 const dispatch = useDispatch()
 const { cart, auth, orders } = useSelector(store => store);

```

```

const { orderItemId, orderId } = useParams()
const navigate = useNavigate();

useEffect(() => {
 dispatch(fetchOrderItemById({
 orderItemId: Number(orderItemId),
 jwt: localStorage.getItem("jwt") || ""
 }))
 dispatch(fetchOrderById({
 orderId: Number(orderId),
 jwt: localStorage.getItem("jwt") || ""
 }))
}, [auth.jwt])

if (!orders.orders || !orders.orderItem) {
 return <div className='h-[80vh] flex justify-center items-center'>
 No order found
 </div>;
}

const handleCancelOrder = () => {
 dispatch(cancelOrder(orderId))
}

return (
<Box className='space-y-5 '>

 <section className='flex flex-col gap-5 justify-center items-center'>

 <div className='text-sm space-y-1 text-center'>
 <h1
 className='font-bold'>{orders.orderItem?.product.seller?.businessDetails.businessName}</h1>
 <p>{orders.orderItem?.product.title}</p>
 <p>Size: M</p>
 </div>
 <div>
 <Button onClick={() =>
 navigate(`~/reviews/${orders.orderItem?.product.id}/create`)}>Write
 Review</Button>
 </div>
 </section>

 <section className='border p-5'>
 <OrderStepper orderStatus={orders.currentOrder?.orderStatus} />
 </section>

```

```

<div className='border p-5'>
 <h1 className='font-bold pb-3'>Delivery Address</h1>
 <div className='text-sm space-y-2'>
 <div className='flex gap-5 font-medium'>
 <p> {orders.currentOrder?.shippingAddress.name}</p>
 <Divider flexItem orientation='vertical' />
 <p>{orders.currentOrder?.shippingAddress.mobile}</p>
 </div>

 <p>
 {orders.currentOrder?.shippingAddress.address},
 {orders.currentOrder?.shippingAddress.city},
 {orders.currentOrder?.shippingAddress.state} -
 {orders.currentOrder?.shippingAddress.pinCode}
 </p>
 </div>
</div>

<div className='border space-y-4'>

 <div className='flex justify-between text-sm pt-5 px-5'>
 <div className='space-y-1'>
 <p className='font-bold'>Total Item Price</p>
 <p>You saved ₹
 {orders.orderItem?.mrpPrice -
 orders.orderItem?.sellingPrice}.00 on this item</p>
 </div>

 <p className='font-medium'>₹ {orders.orderItem?.sellingPrice}.00</p>
 </div>

 <div className='px-5'>
 <div className='bg-teal-50 px-5 py-2 text-xs font-medium flex
 items-center gap-3'>
 <PaymentsIcon />
 <p>Pay On Delivery</p>

 </div>
 </div>

 <Divider />
 <div className='px-5 pb-5'>
 <p className='text-xs'>Sold by :
 {orders.orderItem.product.seller?.businessDetails.businessName}</p>
 </div>

 <div className='p-10'>

```

```

 <Button
 disabled={orders.currentOrder?.orderStatus === "CANCELLED"}
 onClick={handleCancelOrder}
 color='error' sx={{ py: "0.7rem" }} className=' ' variant='outlined'
 fullWidth>
 {orders.currentOrder?.orderStatus === "CANCELLED" ? "order
canceled" : "Cancel Order"}
 </Button>
 </div>
</div>
</Box>
)
}

export default OrderDetails
import React from 'react'
import ElectricBoltIcon from '@mui/icons-material/ElectricBolt';
import { Avatar, Button } from '@mui/material';
import { teal } from '@mui/material/colors';
import { useNavigate } from 'react-router-dom';
import type { Order, OrderItem } from '../../../../../types/orderTypes';
import { formatDate } from '../../../../../util/fomateDate';

interface OrderItemCardProps{
 item:OrderItem,
 order:Order
}
const OrderItemCard:React.FC<OrderItemCardProps> = ({item,order}) => {
 const navigate = useNavigate()
 return (
 <div onClick={() => navigate(`account/orders/${order.id}/${item.id}`)}>
 className='text-sm bg-white p-5 space-y-4 border rounded-md cursor-pointer'>

 <div className='flex items-center gap-3'>
 <div>
 <Avatar sizes='small' sx={{ bgcolor: teal[500] }}>
 <ElectricBoltIcon />
 </Avatar>
 </div>
 <div>
 <h1 className='font-bold text-teal-600'>{order.orderStatus}</h1>
 <p>Arriving by {formatDate(order.deliverDate)}</p>
 </div>
 </div>
 <div className='p-5 bg-teal-50 flex gap-3 '>
 <div className='>
 <img className='w-[70px]' /

```

```

 src={item.product.images[0]} alt="" />
 </div>
 <div className='w-full space-y-2'>
 <h1
className='font-bold'>{item.product.seller?.businessDetails.businessName}</h1>
 <p>
 {item.product.title}
 </p>
 <p>size :
 FREE</p>
 </div>

 </div>
</div>
)
}

export default OrderItemCard
import { Box, Divider } from "@mui/material";
import React, { useEffect, useState } from "react";
import FiberManualRecordIcon from "@mui/icons-material/FiberManualRecord";
import { Description } from "@mui/icons-material";
import CheckCircleIcon from "@mui/icons-material/CheckCircle";

const steps = [
 { name: "Order Placed", description: "on Thu, 11 Jul", value: "PLACED" },
 { name: "Packed", description: "Item Packed in Dispatch Warehouse", value: "CONFIRMED" },
 { name: "Shipped", description: "by Mon, 15 Jul", value: "SHIPPED" },
 { name: "Arriving", description: "by 16 Jul - 18 Jul", value: "ARRIVING" },
 { name: "Arrived", description: "by 16 Jul - 18 Jul", value: "DELIVERED" },
 // { name: "Canceled", description: "by 16 Jul - 18 Jul", value: "CANCELLED" }
],
];

const canceledStep = [
 { name: "Order Placed", description: "on Thu, 11 Jul", value: "PLACED" },
 { name: "Order Canceled", description: "on Thu, 11 Jul", value: "CANCELLED" }
],
];

const currentStep = 2; // Change this value based on the current step

const OrderStepper = ({ orderStatus }: any) => {

```

```

const [statusStep, setStatusStep] = useState(steps);

useEffect(() => {

 if (orderStatus === 'CANCELLED') {
 setStatusStep(canceledStep)
 } else {
 setStatusStep(steps)
 }

 // setCurrentStep(orderStatus==='Cancelled'? canceledStep : steps)
// .slice(0,orderStatus==="CANCELLED"?steps.length:steps.length-1)
}, [orderStatus])
return (
 <Box className=" mx-auto my-10">
 {statusStep.map((step, index) => (
 <>
 <div key={index} className={` flex px-4 `}>
 <div className="flex flex-col items-center">
 <Box
 sx={{ zIndex: -1 }}
 className={`${` w-8 h-8 rounded-full flex
items-center justify-center z-10 ${index <= currentStep
? " bg-gray-200 text-teal-500"
: "bg-gray-300 text-gray-600"
} `}`}
 >
 {step.value === orderStatus ? (
 <CheckCircleIcon />
) : (
 <FiberManualRecordIcon sx={{ zIndex: -1 }} />
)
 }
 </Box>
 {index < statusStep.length - 1 && (
 <div
 className={`${`border h-20 w-[2px] ${index <
currentStep
? " bg-teal-500"
: "bg-gray-300 text-gray-600"
} `}`}
 ></div>
)
 </div>

 <div className={`${`ml-2 w-full`}>
 <div
 className={`${` ${step.value === orderStatus
? " bg-teal-500 text-teal-500"
: "bg-gray-300 text-gray-600"
} `}`}
 >

```

```

 ? " bg-primary-color p-2 text-white
font-medium rounded-md -translate-y-3"
 : ""
 } ${ (orderStatus==="CANCELLED" &&
step.value==orderStatus)?"bg-red-500":""} w-full`}
>
<p
 className={`

`}
>
{step.name}
</p>
<p
 className={` ${step.value==orderStatus
 ? " text-gray-200"
 : "text-gray-500"
 } text-xs `}>{step.description}</p>
</div>
</div>
</div>
</div>
);
))}

</Box>
);

};

export default OrderStepper;
import { Alert, Divider, Snackbar } from '@mui/material'
import React, { useEffect, useState } from 'react'
import { Route, Routes, useLocation, useNavigate } from 'react-router-dom'
import Order from './Order'
import UserDetails from './UserDetails'
import SavedCards from './SavedCards'
import OrderDetails from './OrderDetails'
import { useDispatch, useSelector } from '../../../../../Redux Toolkit/Store'
import { performLogout } from '../../../../../Redux Toolkit/Customer/AuthSlice'
import Addresses from './Addresses'

const menu = [
 { name: "orders", path: "/account/orders" },
 { name: "profile", path: "/account/profile" },
 { name: "Saved Cards", path: "/account/saved-card" },

 { name: "Addresses", path: "/account/addresses" },
 { name: "Logout", path: "/" }
]
const Profile = () => {
 const navigate = useNavigate();
 const location = useLocation();

```

```

const dispatch = useAppDispatch()
const { user, orders } = useAppSelector(store => store)
const [snackbarOpen, setOpenSnackbar] = useState(false);

const handleLogout = () => {
 dispatch(performLogout())
 navigate("/")
}

const handleClick = (item: any) => {
 if (item.name === "Logout") {
 handleLogout()
 }
 else navigate(` ${item.path}`)
}
const handleCloseSnackbar = () => {
 setOpenSnackbar(false);
};

useEffect(() => {
 if (user.profileUpdated || orders.orderCanceled || user.error) {
 setOpenSnackbar(true);
 }
}, [user.profileUpdated, orders.orderCanceled]);
return (
 <div className='px-5 lg:px-52 min-h-screen mt-10 '>

 <div>
 <h1 className='text-xl font-bold
pb-5'>{user.user?.fullName}</h1>
 </div>
 <Divider />
 <div className='grid grid-cols-1 lg:grid-cols-3 lg:min-h-[78vh]'>

 <div className="col-span-1 lg:border-r lg:pr-5 py-5 h-full flex
flex-row flex-wrap lg:flex-col gap-3">

 {menu.map((item, index) => <div
 onClick={() => handleClick(item)}
 className={`${menu.length - 1 !== index ? "border-b" :
""} ${item.path == location.pathname ? "bg-primary-color text-white" : ""} px-5
py-3 rounded-md hover:bg-teal-500 hover:text-white cursor-pointer `}>
 <p>{item.name}</p>
 </div>)}

 </div>
 <div className='lg:col-span-2 lg:pl-5 py-5'>

 <Routes>

```

```

 <Route path='/' element={<UserDetails />} />
 <Route path='/orders' element={<Order />} />
 <Route path='/orders/:orderId/:orderItemId'
element={<OrderDetails />} />
 <Route path='/profile' element={<UserDetails />} />
 <Route path='/saved-card' element={<SavedCards />} />
 <Route path='/addresses' element={<Addresses />} />
 {/* addresses */}
 </Routes>

 </div>

 </div>
 <Snackbar
 anchorOrigin={{ vertical: "top", horizontal: "right" }}
 open={snackbarOpen}
 autoHideDuration={6000}
 onClose={handleCloseSnackbar}
 >
 <Alert
 onClose={handleCloseSnackbar}
 severity={user.error ? "error" : "success"}
 variant="filled"
 sx={{ width: "100%" }}
 >
 {user.error ? user.error : orders.orderCanceled?"order
canceled successfully": "success"}
 </Alert>
 </Snackbar>
</div>
)
}

export default Profile
import React from 'react'
import AddCardIcon from '@mui/icons-material/AddCard';
import { teal } from '@mui/material/colors';

const SavedCards = () => {
 return (
 <div className='flex flex-col justify-center items-center lg:min-h-[60vh]
gap-6'>

 <div>
 <AddCardIcon sx={{color:teal[400], fontSize:"150px"}}/>
 </div>

 <div className='text-center w-full lg:w-[68%] space-y-4'>

```

```

 <h1 className='font-bold text-lg textg'>SAVE YOUR CREDIT/DEBIT CARDS
DURING PAYMENT
 </h1>
 <p className='text-gray-700'>It's convenient to pay with saved
cards. Your card information will be secure, we use 128-bit encryption</p>
 </div>

 </div>
)
}

export default SavedCards
import React from 'react'
import type { Address } from '../../../../../types/userTypes'

const UserAddressCard = ({item}:{item: Address}) => {
 return (
 <div className='p-5 border rounded-md '>

 <div className='space-y-3'>
 <h1 className='font-semibold'>{item.name}</h1>
 <p className='w-[320px]'>
 {item.address},
 {item.locality},
 {item.city},
 {item.state} - {item.pinCode}</p>
 <p>Mobile : {item.mobile}</p>
 </div>
 </div>
)
}

export default UserAddressCard
import {
 Alert,
 Avatar,
 Box,
 Button,
 Divider,
 Modal,
 Snackbar,
} from "@mui/material";
import React, { useEffect, useState } from "react";
import EditIcon from "@mui/icons-material/Edit";
import ProfileFildCard from "../../../../../seller/pages/Account/ProfileFildCard";
import { useAppSelector } from "../../../../../Redux Toolkit/Store";
import { style } from "../../../../../seller/pages/Account/Profile";

```

```

const UserDetails = () => {
 const { user } = useAppSelector((store) => store);
 // const [open, setOpen] = useState(false);
 // const handleClose = () => setOpen(false);

 // const handleOpen = () => {
 // setOpen(true);
 // };

 return (
 <div className="flex justify-center py-10">
 <div className="w-full lg:w-[70%]">
 <div className="flex items-center pb-3 justify-between">
 <h1 className="text-2xl font-bold text-gray-600">
 Personal Details
 </h1>
 {/* <div>
 <Button
 onClick={handleOpen}
 size="small"
 sx={{ borderRadius: "2.9rem" }}
 variant="contained"
 className="w-16 h-16"
 >
 <EditIcon />
 </Button>
 </div> */}
 </div>
 <div className="space-y-5">
 {/* <Avatar
 sx={{ width: "10rem", height: "10rem" }}>
 src="https://cdn.pixabay.com/photo/2014/11/29/19/33/bald-eagle-550804_640.jpg"
 /> */}
 <div>
 <ProfileFieldCard keys={"Name"} value={user.user?.fullName} />
 <Divider />
 <ProfileFieldCard keys={"Email"} value={user.user?.email} />
 <Divider />
 <ProfileFieldCard keys={"Mobile"} value={user.user?.mobile} />
 </div>
 </div>
 </div>
 {/* <Modal
 open={open}
 onClose={handleClose}
 aria-labelledby="modal-modal-title"
 aria-describedby="modal-modal-description"
 > */}

```

```

 <Box sx={style}>Update UserProfile</Box>
 </Modal> */
}

</div>
);
};

export default UserDetails;
import React, { useEffect, useState } from 'react'
import LoginForm from './LoginForm'
import { Alert, Button, Snackbar } from '@mui/material';
import SignupForm from './SignupForm';
import { useAppSelector } from '../../../../../Redux Toolkit/Store';

const Auth = () => {
 const [isLoginPage, setIsLoginPage] = useState(true);
 const handleCloseSnackbar = () => setSnackbarOpen(false)
 const { auth } = useAppSelector(store => store)
 const [snackbarOpen, setSnackbarOpen] = useState(false);

 useEffect(() => {

 if (auth.otpSent || auth.error) {
 setSnackbarOpen(true);
 console.log("store ", auth.error)
 }

 }, [auth.otpSent, auth.error])

 return (
 <div className='flex justify-center h-[90vh] items-center'>
 <div className='max-w-md h-[85vh] rounded-md border shadow-lg '>

 <div className='mt-8 px-10'>
 {isLoginPage ? <LoginForm /> : <SignupForm />}

 <div className='flex items-center gap-1 justify-center
mt-5'>
 <p>{isLoginPage && "Don't"} have Account ?</p>
 <Button onClick={() => setIsLoginPage(!isLoginPage)} size='small'>{isLoginPage ? "create account" : "login"}</Button>
 </div>
 </div>

 </div>
 <Snackbar
 anchorOrigin={{ vertical: "top", horizontal: "right" }}>

```

```

 open={snackbarOpen} autoHideDuration={6000}
 onClose={handleCloseSnackbar}
 >
 <Alert
 onClose={handleCloseSnackbar}
 severity={auth.error?"error":"success"}
 variant="filled"
 sx={{ width: '100%' }}
 >
 {auth.error?auth.error : " otp sent to your email!"}
 </Alert>
 </Snackbar>
 </div>
)
}

export default Auth
/* eslint-disable @typescript-eslint/no-explicit-any */
import { Button, CircularProgress, TextField } from '@mui/material'
import { useEffect, useState } from 'react'
import OTPInput from '../../../../../components/OtpFild/OTPInput'
import { useFormik } from 'formik';
import { useDispatch, useSelector } from '../../../../../Redux Toolkit/Store';
import { navigate } from 'react-router-dom';
import { sendLoginSignupOtp, signin } from '../../../../../Redux Toolkit/Customer/AuthSlice';

const LoginForm = () => {

 const navigate = useNavigate();
 const [otp, setOtp] = useState("");
 const [isOtpSent, setIsOtpSent] = useState(false)
 const [timer, setTimer] = useState<number>(30); // Timer state
 const [isTimerActive, setIsTimerActive] = useState<boolean>(false);
 const dispatch = useDispatch();
 const { auth } = useSelector(store => store)

 const formik = useFormik({
 initialValues: {
 email: '',
 otp: ''
 },
 onSubmit: (values: any) => {
 // Handle form submission
 dispatch(signin({ email: values.email, otp, navigate }))
 console.log('Form data:', values);
 }
 });
}

```

```
const handleOtpChange = (otp: any) => {
 setOtp(otp);
};

const handleResendOTP = () => {
 // Implement OTP resend logic
 dispatch(sendLoginSignupOtp({ email: "signing_" + formik.values.email }));
 console.log('Resend OTP');
 setTimer(30);
 setIsTimerActive(true);
};

const handleSentOtp = () => {
 setIsOtpSent(true);
 handleResendOTP();
}

const handleLogin = () => {
 formik.handleSubmit()
}

useEffect(() => {
 let interval: any;

 if (isTimerActive) {
 interval = setInterval(() => {
 setTimer(prev => {
 if (prev === 1) {
 clearInterval(interval);
 setIsTimerActive(false);
 return 30; // Reset timer for next OTP request
 }
 return prev - 1;
 });
 }, 1000);
 }

 return () => {
 if (interval) clearInterval(interval);
 };
}, [isTimerActive]);
```

```

 return (
 <div>
 <h1 className='text-center font-bold text-xl text-primary-color pb-8'>Login</h1>
 <form className="space-y-5">

 <TextField
 fullWidth
 name="email"
 label="Enter Your Email"
 value={formik.values.email}
 onChange={formik.handleChange}
 onBlur={formik.handleBlur}
 error={formik.touched.email && Boolean(formik.errors.email)}
 helperText={formik.touched.email ? formik.errors.email as string : undefined}
 />

 {auth.otpSent && <div className="space-y-2">
 <p className="font-medium text-sm">
 * Enter OTP sent to your mobile number
 </p>
 <OTPIInput
 length={6}
 onChange={handleOtpChange}
 error={false}
 />
 <p className="text-xs space-x-2">
 {isTimerActive ? (
 Resend OTP in {timer} seconds
) : (
 <>
 Didn't receive OTP?{" "}
 <span
 onClick={handleResendOTP}
 className="text-teal-600 cursor-pointer hover:text-teal-800 font-semibold"
 >
 Resend OTP

 </>
) }
 </p>
 {formik.touched.otp && formik.errors.otp &&
 <p>{formik.errors.otp as string}</p>}
 </div>

 {auth.otpSent && <div>
 <Button disabled={auth.loading} onClick={handleLogin}>

```

```

 fullWidth variant='contained' sx={{ py: "11px" }}>
 auth.loading ? <CircularProgress />:
 "Login"}</Button>
 </div>

 {!auth.otpSent && <Button
 disabled={auth.loading}
 fullWidth
 variant='contained'
 onClick={handleSentOtp}
 sx={{ py: "11px" }}>
 auth.loading ? <CircularProgress />: "sent
otp"</Button>
 }

 </form>

 </div>
)
}

export default LoginForm
/* eslint-disable @typescript-eslint/no-explicit-any */
import { Button, CircularProgress, TextField } from '@mui/material'
import React, { useEffect, useState } from 'react'
import OTPInput from '../../../../../components/OtpFild/OTPInput'
import { useFormik } from 'formik';
import { useDispatch, useSelector } from '../../../../../Redux Toolkit/Store';
import { navigate } from 'react-router-dom';
import { sendLoginSignupOtp, signup } from '../../../../../Redux
Toolkit/Customer/AuthSlice';

const SignupForm = () => {

 const navigate = useNavigate();
 const [otp, setOtp] = useState("");
 const [isOtpSent, setIsOtpSent] = useState(false)
 const [timer, setTimer] = useState<number>(30); // Timer state
 const [isTimerActive, setIsTimerActive] = useState<boolean>(false);
 const dispatch = useDispatch();
 const { auth } = useSelector(store => store)

 const formik = useFormik({
 initialValues: {
 email: '',
 otp: '',
 }
 })
}
```

```
 name: ""
 } ,

 onSubmit: (values: any) => {
 // Handle form submission
 dispatch(signup({ fullName: values.name, email: values.email, otp,
navigate }))
 console.log('Form data:', values);
 }
}) ;

const handleOtpChange = (otp: any) => {

 setOtp(otp);
}

const handleResendOTP = () => {
 // Implement OTP resend logic
 dispatch(sendLoginSignupOtp({ email: formik.values.email }))
 console.log('Resend OTP');
 setTimer(30);
 setIsTimerActive(true);
}

const handleSentOtp = () => {
 setIsOtpSent(true);
 handleResendOTP();
}

const handleLogin = () => {
 formik.handleSubmit()
}

useEffect(() => {
 let interval: any;

 if (isTimerActive) {
 interval = setInterval(() => {
 setTimer(prev => {
 if (prev === 1) {
 clearInterval(interval);
 setIsTimerActive(false);
 return 30; // Reset timer for next OTP request
 }
 return prev - 1;
 });
 }, 1000);
 }
})
```

```

 return () => {
 if (interval) clearInterval(interval);
 };
 }, [isTimerActive]);

return (
 <div>
 <h1 className='text-center font-bold text-xl text-primary-color pb-5'>Signup</h1>
 <form className="space-y-5">

 <TextField
 fullWidth
 name="email"
 label="Enter Your Email"
 value={formik.values.email}
 onChange={formik.handleChange}
 onBlur={formik.handleBlur}
 error={formik.touched.email && Boolean(formik.errors.email)}
 helperText={formik.touched.email ? formik.errors.email as string : undefined}
 />

 {auth.otpSent && <div className="space-y-2">
 <p className="font-medium text-sm">
 * Enter OTP sent to your mobile number
 </p>
 <OTPIInput
 length={6}
 onChange={handleOtpChange}
 error={false}
 />
 <p className="text-xs space-x-2">
 {isTimerActive ? (
 Resend OTP in {timer} seconds
) : (
 <>
 Didn't receive OTP?{" "}
 <span
 onClick={handleResendOTP}
 className="text-teal-600 cursor-pointer hover:text-teal-800 font-semibold"
 >
 Resend OTP

 </>
)}
 </p>
 }
 </form>
 </div>
)

```

```

) }
 </p>
 { formik.touched.otp && formik.errors.otp &&
<p>{formik.errors.otp as string}</p>
 </div>

 {auth.otpSent && <TextField
 fullWidth
 name="name"
 label="Enter Your Name"
 value={formik.values.name}
 onChange={formik.handleChange}
 onBlur={formik.handleBlur}
 error={formik.touched.name && Boolean(formik.errors.name) }
 helperText={formik.touched.name ? formik.errors.name as
string : undefined}
 />

 {auth.otpSent && <div>
 <Button
 disabled={auth.loading}
 onClick={handleLogin}
 fullWidth variant='contained' sx={{ py: "11px" }}>
{auth.loading ? <CircularProgress size="small"
 sx={{ width: "27px", height: "27px" }} /> : " Signup
"} </Button>
 </div>

 {!auth.otpSent && <Button
 fullWidth
 variant='contained'
 onClick={handleSentOtp}
 disabled={auth.loading}
 sx={{ py: "11px" }}>
{auth.loading ? <CircularProgress size="small"
 sx={{ width: "27px", height: "27px" }} /> : "sent otp"
}

 </Button>
 }

 </div>
)
}

```

```

export default SignupForm
import { Alert, Button, Snackbar } from "@mui/material";
import React, { useEffect, useState } from "react";
import BecomeSellerFormStep1 from "./BecomeSellerFormStep1";
import BecomeSellerFormStep3 from "./BecomeSellerFormStep3";
import BecomeSellerFormStep2 from "./BecomeSellerFormStep2";
import BecomeSellerFormStep4 from "./BecomeSellerFormStep4";
import { useDispatch, useSelector } from "../../../../../Redux Toolkit/Store";
import SellerLoginForm from "./SellerLoginForm";
import { useLocation } from "react-router-dom";
import SellerAccountForm from "./SellerAccountForm";

const BecomeSeller = () => {
 const [activeStep, setActiveStep] = useState(0);
 const dispatch = useDispatch();
 const location = useLocation();
 const [isLoginPage, setIsLoginPage] = useState(false);
 const { sellerAuth } = useSelector(store => store)

 const handleCloseSnackbar = () => setSnackbarOpen(false)
 const [snackbarOpen, setSnackbarOpen] = useState(false);

 useEffect(() => {

 if (sellerAuth.sellerCreated || sellerAuth.error || sellerAuth.otpSent) {
 setSnackbarOpen(true);
 console.log("store ", sellerAuth.error)
 }

 }, [sellerAuth.sellerCreated, sellerAuth.error, sellerAuth.otpSent])

 return (
 <div className="grid md:gap-10 grid-cols-3 min-h-screen">
 <section className="lg:col-span-1 md:col-span-2 col-span-3 p-10 shadow-lg rounded-b-md">

 {!isLoginPage ?
 <SellerAccountForm /> :
 <SellerLoginForm />
 }

 <div className='mt-10 space-y-2'>
 <h1 className='text-center text-sm font-medium'>{isLoginPage && "Don't"} have account ? </h1>
 <Button onClick={() => setIsLoginPage(!isLoginPage)} fullWidth sx={{ py: "11px" }} variant='outlined'>{isLoginPage ? "Register" : "Login"}</Button>
 </div>
 </section>
 </div>
)
}

```

```

 <section className=" hidden md:col-span-1 md:flex lg:col-span-2
justify-center items-center">
 <div className="lg:w-[70%] px-5 space-y-10">
 <div className="borderr rounded-md space-y-2 font-bold text-center">
 <p className=" text-2xl">Join the Marketplace Revolution</p>
 <p className="text-lg text-teal-500"> Boost Your Sales Today</p>
 </div>

 {/* <div>
 <p className=" logo absolute p-6 rounded-t-full text-white
text-center top-0 left-16 right-11 bg-teal-500">Sajid Bazaar</p>
 </div> */}
 </div>

 </section>

 <Snackbar
 anchorOrigin={{ vertical: "top", horizontal: "right" }}
 open={snackBarOpen} autoHideDuration={6000}
 onClose={handleCloseSnackBar}
 >
 <Alert
 onClose={handleCloseSnackBar}
 severity={sellerAuth.error ? "error" : "success"}
 variant="filled"
 sx={{ width: '100%' }}
 >
 {sellerAuth.error ? sellerAuth.error : sellerAuth.sellerCreated ?
 sellerAuth.sellerCreated : " otp sent to your email!"}
 </Alert>
 </Snackbar>
 </div>
);

};

export default BecomeSeller;
import React, { useState } from "react";
import { useFormik } from "formik";
import * as Yup from "yup";
import { Box, Button, TextField, Grid } from "@mui/material";
import OTPInput from "../../components/OtpField/OTPInput";

// Validation schema

```

```

const BecomeSellerFormStep1 = ({ formik, handleOtpChange }: any) => {

 const handleResendOTP = () => {
 console.log("handle resend otp")
 }

 return (
 <Box >
 <p className="text-xl font-bold text-center pb-9">Contact
Details</p>

 <div className="space-y-9">

 <TextField
 fullWidth
 name="mobile"
 label="Mobile"
 value={formik.values.mobile}
 onChange={formik.handleChange}
 onBlur={formik.handleBlur}
 error={formik.touched.mobile &&
Boolean(formik.errors.mobile)}
 helperText={formik.touched.mobile && formik.errors.mobile}
 />

 {/* <div className="space-y-2">
 <p className="font-medium text-sm">
 * Enter OTP sent to your mobile number
 </p>
 <OTPINput
 length={6}
 onChange={handleOtpChange}
 error={false}
 />
 <p className="text-xs space-x-2">
 Didn't receive OTP?{" "}
 <span onClick={handleResendOTP} className="text-teal-600
cursor-pointer hover:text-teal-800 font-semibold">
 Resend OTP

 </p>
 </div> */}

 <TextField
 fullWidth

```

```

 name="GSTIN"
 label="GSTIN Number"
 value={formik.values.GSTIN}
 onChange={formik.handleChange}
 onBlur={formik.handleBlur}
 error={formik.touched.GSTIN && Boolean(formik.errors.GSTIN) }
 helperText={formik.touched.GSTIN && formik.errors.GSTIN}
 />
 </div>

 </Box>
);
};

export default BecomeSellerFormStep1;

```

import React from "react";

import { Grid, TextField } from "@mui/material";

```

interface BecomeSellerFormStep2Props {
 formik: any; // Replace 'any' with the correct type for formik instance
}

const BecomeSellerFormStep2: React.FC<BecomeSellerFormStep2Props> = ({ formik }) => {
 return (
 <div>
 <Grid container spacing={3}>
 <Grid item xs={12}>
 <TextField
 fullWidth
 name="pickupAddress.name"
 label="Name"
 value={formik.values.pickupAddress.name}
 onChange={formik.handleChange}
 onBlur={formik.handleBlur}
 error={formik.touched.name && Boolean(formik.errors.name) }
 helperText={formik.touched.name && formik.errors.name}
 />
 </Grid>
 <Grid item xs={6}>
 <TextField
 fullWidth
 name="pickupAddress.mobile"
 label="Mobile"
 value={formik.values.pickupAddress.mobile}
 onChange={formik.handleChange}
 onBlur={formik.handleBlur}
 error={formik.touched.mobile && Boolean(formik.errors.mobile) }
 />
 </Grid>
 </Grid>
 </div>
);
}

export default BecomeSellerFormStep2;

```

```
 helperText={formik.touched.mobile && formik.errors.mobile}
 />
 </Grid>
 <Grid item xs={6}>
 <TextField
 fullWidth
 name="pickupAddress.pincode"
 label="Pincode"
 value={formik.values.pickupAddress.pincode}
 onChange={formik.handleChange}
 onBlur={formik.handleBlur}
 error={formik.touched.pickupAddress?.pincode &&
Boolean(formik.errors.pickupAddress?.pincode)}
 helperText={formik.touched.pickupAddress?.pincode &&
formik.errors.pickupAddress?.pincode}
 />
 </Grid>
 <Grid item xs={12}>
 <TextField
 fullWidth
 name="pickupAddress.address"
 label="Address (House No, Building, Street)"
 value={formik.values.pickupAddress.address}
 onChange={formik.handleChange}
 onBlur={formik.handleBlur}
 error={formik.touched.pickupAddress?.address &&
Boolean(formik.errors.pickupAddress?.address)}
 helperText={formik.touched.pickupAddress?.address &&
formik.errors.pickupAddress?.address}
 />
 </Grid>
 <Grid item xs={12}>
 <TextField
 fullWidth
 name="pickupAddress.locality"
 label="Locality/Town"
 value={formik.values.pickupAddress.locality}
 onChange={formik.handleChange}
 onBlur={formik.handleBlur}
 error={formik.touched.pickupAddress?.locality &&
Boolean(formik.errors.pickupAddress?.locality)}
 helperText={formik.touched.pickupAddress?.locality &&
formik.errors.pickupAddress?.locality}
 />
 </Grid>
 <Grid item xs={6}>
 <TextField
 fullWidth
 name="pickupAddress.city"
```

```

 label="City"
 value={formik.values.pickupAddress.city}
 onChange={formik.handleChange}
 onBlur={formik.handleBlur}
 error={formik.touched.pickupAddress?.city &&
Boolean(formik.errors.pickupAddress?.city)}
 helperText={formik.touched.pickupAddress?.city &&
formik.errors.pickupAddress?.city}
 />
 </Grid>
 <Grid item xs={6}>
 <TextField
 fullWidth
 name="pickupAddress.state"
 label="State"
 value={formik.values.pickupAddress.state}
 onChange={formik.handleChange}
 onBlur={formik.handleBlur}
 error={formik.touched.pickupAddress?.state &&
Boolean(formik.errors.pickupAddress?.state)}
 helperText={formik.touched.pickupAddress?.state &&
formik.errors.pickupAddress?.state}
 />
 </Grid>
 </Grid>
</div>
);
};

export default BecomeSellerFormStep2;
import React from "react";
import { Grid, TextField } from "@mui/material";

interface BecomeSellerFormStep2Props {
 formik: any; // Replace 'any' with the correct type for formik instance
}

const BecomeSellerFormStep3: React.FC<BecomeSellerFormStep2Props> = ({ formik
}) => {
 return (
 <div className="space-y-5">

 <TextField
 fullWidth
 name="bankDetails.accountNumber"
 label="Account Number"
 value={formik.values.bankDetails.accountNumber}
 onChange={formik.handleChange}
 onBlur={formik.handleBlur}

```

```

 error={formik.touched.bankDetails?.accountNumber &&
Boolean(formik.errors.bankDetails?.accountNumber)}
 helperText={formik.touched.bankDetails?.accountNumber &&
formik.errors.bankDetails?.accountNumber}
 />
 <TextField
 fullWidth
 name="bankDetails.ifscCode"
 label="IFSC Code"
 value={formik.values.bankDetails.ifscCode}
 onChange={formik.handleChange}
 onBlur={formik.handleBlur}
 error={formik.touched.bankDetails?.ifscCode &&
Boolean(formik.errors.bankDetails?.ifscCode)}
 helperText={formik.touched.bankDetails?.ifscCode &&
formik.errors.bankDetails?.ifscCode}
 />
 <TextField
 fullWidth
 name="bankDetails.accountHolderName"
 label="Account Holder Name"
 value={formik.values.bankDetails.accountHolderName}
 onChange={formik.handleChange}
 onBlur={formik.handleBlur}
 error={formik.touched.bankDetails?.accountHolderName &&
Boolean(formik.errors.bankDetails?.accountHolderName)}
 helperText={formik.touched.bankDetails?.accountHolderName &&
formik.errors.bankDetails?.accountHolderName}
 />
</div>
);
};

export default BecomeSellerFormStep3;
import { TextField } from '@mui/material'
import React from 'react'
interface BecomeSellerFormStep2Props {
 formik: any; // Replace 'any' with the correct type for formik instance
}

const BecomeSellerFormStep4 = ({ formik }: BecomeSellerFormStep2Props) => {
 return (
 <div className='space-y-5'>
 <TextField
 fullWidth
 name="businessDetails.businessName"
 label="Business Name"
 value={formik.values.businessDetails.businessName}

```

```

 onChange={formik.handleChange}
 onBlur={formik.handleBlur}
 error={formik.touched?.businessDetails?.businessName &&
Boolean(formik.errors?.businessDetails?.businessName)}
 helperText={formik.touched?.businessDetails?.businessName &&
formik.errors?.businessDetails?.businessName}
 />

 <TextField
 fullWidth
 name="sellerName"
 label="Seller Name"
 value={formik.values.sellerName}
 onChange={formik.handleChange}
 onBlur={formik.handleBlur}
 error={formik.touched.sellerName && Boolean(formik.errors.sellerName) }
 helperText={formik.touched.sellerName && formik.errors.sellerName}
 />

 <TextField
 fullWidth
 name="email"
 label="Email"
 value={formik.values.email}
 onChange={formik.handleChange}
 onBlur={formik.handleBlur}
 error={formik.touched.email && Boolean(formik.errors.email) }
 helperText={formik.touched.email && formik.errors.email}
 />
 <TextField
 fullWidth
 name="password"
 label="Password"
 value={formik.values.password}
 onChange={formik.handleChange}
 onBlur={formik.handleBlur}
 error={formik.touched?.password && Boolean(formik.errors?.password) }
 helperText={formik.touched?.password && formik.errors?.password}
 />

 </div>
)
}

export default BecomeSellerFormStep4

```

```
import { Button, CircularProgress, Step, StepLabel, Stepper } from "@mui/material";
import React, { useState } from "react";
import BecomeSellerFormStep1 from "./BecomeSellerFormStep1";
import BecomeSellerFormStep3 from "./BecomeSellerFormStep3";
import BecomeSellerFormStep2 from "./BecomeSellerFormStep2";
import { useFormik } from "formik";
import * as Yup from "yup";
import BecomeSellerFormStep4 from "./BecomeSellerFormStep4";
import { useDispatch } from "react-redux";
import { useAppDispatch, useAppSelector } from "../../../../../Redux Toolkit/Store";
import SellerLoginForm from "./SellerLoginForm";
import { createSeller } from "../../../../../Redux Toolkit/Seller/sellerAuthenticationSlice";

const steps = [
 "Tax Details & Mobile",
 "Pickup Address",
 "Bank Details",
 "Supplier Details",
];

const SellerAccountForm = () => {
 const [activeStep, setActiveStep] = useState(0);
 const dispatch = useAppDispatch();
 const {sellerAuth}=useAppSelector(store=>store)

 const handleStep = (value: number) => {
 setActiveStep(activeStep + value);
 };

 const [otp, setOtp] = useState<any>();

 const formik = useFormik({
 initialValues: {
 mobile: "",
 otp: "",
 gstin: "",
 pickupAddress: {
 name: "",
 mobile: "",
 pincode: "",
 address: "",
 locality: "",
 city: "",
 state: "",
 },
 bankDetails: {

```

```

 accountNumber: "",
 ifscCode: "",
 accountHolderName: "",
 },
 sellerName: "",
 email: "",
 businessDetails: {
 businessName: "",
 businessEmail:"",
 businessMobile:"",
 logo:"",
 banner:"",
 businessAddress:""
 },
 password: ""
},
// validationSchema: FormSchema,
onSubmit: (values) => {
 console.log(values, "formik submitted");
 console.log("active step ", activeStep);
 dispatch(createSeller(formik.values))
},
)) ;

const handleOtpChange = (otpValue: string) => {
 setOtp(otpValue);
 console.log(otpValue);
 // formik.setFieldValue("opt",otpValue)
};

const handleSubmit = () => {
 //submit form data to server
 formik.handleSubmit();
 console.log("Form Submitted");
};

return (
 <div> <Stepper activeStep={activeStep} alternativeLabel>
 {steps.map((label) => (
 <Step key={label}>
 <StepLabel>{label}</StepLabel>
 </Step>
)));
 </Stepper>
 <div className="mt-20 space-y-10">
 <div>
 {activeStep === 0 ? (

```

```

 <BecomeSellerFormStep1
 formik={formik}
 handleOtpChange={handleOtpChange}
 />
) : activeStep === 1 ? (
 <BecomeSellerFormStep2 formik={formik} />
) : activeStep === 2 ? (
 <BecomeSellerFormStep3 formik={formik} />
) : (
 <BecomeSellerFormStep4 formik={formik} />
)
)
</div>

<div className="flex items-center justify-between">
 <Button
 disabled={activeStep === 0}
 onClick={() => handleStep(-1)}
 variant="contained"
 >
 Back
 </Button>
 <Button
 disabled={sellerAuth.loading}
 onClick={
 activeStep === steps.length - 1
 ? handleSubmit
 : () => handleStep(1)
 }
 variant="contained"
 >
 {activeStep === steps.length - 1 ? sellerAuth.loading ?
<CircularProgress size="small"
 sx={{ width: "27px", height: "27px" }} /> : "create
account" : "Continue"}
 </Button>
 </div>
</div> </div>
)
}

export default SellerAccountForm
/* eslint-disable @typescript-eslint/no-explicit-any */
import { Button, CircularProgress, TextField } from '@mui/material'
import { useEffect, useState } from 'react'
import OTPInput from '../../components/OtpField/OTPInput'

import { useDispatch, useSelector } from '../../../../../Redux Toolkit/Store';
import { sendLoginOtp, verifyLoginOtp } from '../../../../../Redux
Toolkit/Seller/sellerAuthenticationSlice';

```

```

import { useNavigate } from 'react-router-dom';
import { useFormik } from 'formik';

const SellerLoginForm = () => {

 const navigate = useNavigate();
 const [otp, setOtp] = useState("");
 const [isOtpSent, setIsOtpSent] = useState(false)
 const [timer, setTimer] = useState<number>(30); // Timer state
 const [isTimerActive, setIsTimerActive] = useState<boolean>(false);
 const dispatch=useAppDispatch();
 const {sellerAuth}=useAppSelector(store=>store)

 const formik = useFormik({
 initialValues: {
 email: '',
 otp: ''
 },
 onSubmit: (values: any) => {
 // Handle form submission
 dispatch(verifyLoginOtp({email:values.email, otp, navigate}))
 console.log('Form data:', values);
 }
 });

 const handleOtpChange = (otp: any) => {

 setOtp(otp);
 };

 const handleResendOTP = () => {
 // Implement OTP resend logic
 dispatch(sendLoginOtp(formik.values.email))
 console.log('Resend OTP');
 setTimer(30);
 setIsTimerActive(true);
 };

 const handleSentOtp=()=>{
 setIsOtpSent(true);
 handleResendOTP();
 }

 const handleLogin=()=>{
 formik.handleSubmit()
 }
}

```

```

useEffect(() => {
 let interval:any;

 if (isTimerActive) {
 interval = setInterval(() => {
 setTimer(prev => {
 if (prev === 1) {
 clearInterval(interval);
 setIsTimerActive(false);
 return 30; // Reset timer for next OTP request
 }
 return prev - 1;
 });
 }, 1000);
 }

 return () => {
 if (interval) clearInterval(interval);
 };
}, [isTimerActive]);

return (
 <div>
 <h1 className='text-center font-bold text-xl text-primary-color pb-5'>Login As Seller</h1>
 <form className="space-y-5">

 <TextField
 fullWidth
 name="email"
 label="Enter Your Email"
 value={formik.values.email}
 onChange={formik.handleChange}
 onBlur={formik.handleBlur}
 error={formik.touched.email && Boolean(formik.errors.email)}
 helperText={formik.touched.email ? formik.errors.email as string : undefined}
 />

 {sellerAuth.otpSent && <div className="space-y-2">
 <p className="font-medium text-sm">
 * Enter OTP sent to your email
 </p>
 <OTPIInput
 length={6}
 onChange={handleOtpChange}
 error={false}
 />
 }

```

```

 <p className="text-xs space-x-2">
 {isTimerActive ? (
 Resend OTP in {timer} seconds
) : (
 <>
 Didn't receive OTP?{" "}
 <span
 onClick={handleResendOTP}
 className="text-teal-600 cursor-pointer
 hover:text-teal-800 font-semibold"
 >
 Resend OTP

 </>
) }
 </p>
 { formik.touched.otp && formik.errors.otp &&
<p>{formik.errors.otp as string}</p>
 </div>

 {sellerAuth.otpSent &&<div>
 <Button onClick={handleLogin}
 fullWidth variant='contained' sx={{ py: "11px"
} }>Login</Button>
 </div>

 {!sellerAuth.otpSent && <Button
 disabled={sellerAuth.loading}
 fullWidth
 variant='contained'
 onClick={handleSentOtp}
 sx={{ py: "11px" }}>{
 sellerAuth.loading ? <CircularProgress />: "sent
otp"
 }</Button>
 }

 </div>
)
}

export default SellerLoginForm
import {
 Alert,

```

```
Button,
Snackbar,
TextField,
} from "@mui/material";
import React, { useEffect, useState } from "react";

import LocalOfferIcon from "@mui/icons-material/LocalOffer";
import { teal } from "@mui/material/colors";
import FavoriteIcon from "@mui/icons-material/Favorite";
import CartItemCard from "./CartItemCard";
import { useNavigate } from "react-router-dom";
import PricingCard from "./PricingCard";
import { useAppDispatch, useAppSelector } from "../../../../Redux Toolkit/Store";
import { fetchUserCart } from "../../../../Redux Toolkit/Customer/CartSlice";
import type { CartItem } from "../../../../types/cartTypes";
import { applyCoupon } from "../../../../Redux Toolkit/Customer/CouponSlice";
import { Close } from "@mui/icons-material";

const Cart = () => {
 const navigate = useNavigate();
 const dispatch = useAppDispatch();
 const { cart, auth, coupone } = useAppSelector((store) => store);
 const [couponCode, setCouponCode] = useState("");
 const [snackbarOpen, setOpenSnackbar] = useState(false);

 useEffect(() => {
 dispatch(fetchUserCart(localStorage.getItem("jwt") || ""));
 }, [auth.jwt]);

 const handleChange = (e: any) => {
 setCouponCode(e.target.value);
 };

 const handleApplyCoupon = (apply: string) => {
 // console.log(couponCode, apply)

 var code = couponCode;

 if (apply == "false") {
 code = cart.cart?.couponCode || "";
 }

 dispatch(
 applyCoupon({
 apply,
 code,
 orderValue: cart.cart?.totalSellingPrice || 100,
 jwt: localStorage.getItem("jwt") || "",
 })
);
 };
}
```

```

);
};

const handleCloseSnackbar = () => {
 setOpenSnackbar(false);
};

useEffect(() => {
 if (coupone.couponApplied || coupone.error) {
 setOpenSnackbar(true);
 setCouponCode("");
 }
}, [coupone.couponApplied, coupone.error]);

console.log("cart ", coupone);
return (
 <>
 {cart.cart && cart.cart?.cartItems.length !== 0 ? (
 <div className="pt-10 px-5 sm:px-10 md:px-60 lg:px-60 min-h-screen">
 <div className="grid grid-cols-1 lg:grid-cols-3 gap-5 ">
 <div className="lg:col-span-2 space-y-3 ">
 {cart.cart?.cartItems.map((item: CartItem) => (
 <CartItemCard key={item.id} item={item} />
)))
 </div>

 <div className="col-span-1 text-sm space-y-3">
 <div className="border rounded-md px-5 py-3 space-y-5">
 <div className="">
 <div className="flex gap-3 text-sm items-center">
 <LocalOfferIcon
 sx={{ color: teal[600], fontSize: "17px" }}
 />
 Apply Coupens
 </div>
 </div>
 { !cart.cart?.couponCode ? (
 <div className="flex justify-between items-center">
 <TextField
 value={couponCode}
 onChange={handleChange}
 placeholder="coupon code"
 className=""
 size="small"
 />
 <Button
 onClick={() => handleApllyCoupon("true")}
 disabled={couponCode ? false : true}
 size="small"
 >

```

```
 Aplly
 </Button>
 </div>
) : (
 <div className="flex">
 <div className="p-1 pl-5 pr-3 border rounded-full flex gap-2 items-center">
 {cart.cart.couponCode} Applied
 <IconButton
 onClick={() => handleApllyCoupon("false")}
 size="small"
 >
 <Close className="text-red-600" />
 </IconButton>
 </div>
 </div>
) }
</div>

<section className="border rounded-md">
 <PricingCard />
 <div className="p-5">
 <Button
 onClick={() => navigate("/checkout/address")}
 sx={{ py: "11px" }}
 variant="contained"
 fullWidth
 >
 BUY NOW
 </Button>
 </div>
</section>

<div className="border rounded-md px-5 py-4 flex justify-between items-center cursor-pointer">
 Add From Whishlist
 <FavoriteIcon sx={{ color: teal[600], fontSize: "21px" }} />
</div>
</div>
</div>
) : (
<div className="h-[85vh] flex justify-center items-center flex-col">
 <div className="text-center py-5">
 <h1 className="text-lg font-medium">hay its feels so light!</h1>
 <p className="text-gray-500 text-sm">
 there is nothing in your bag, lets add some items
 </p>
 </div>
</div>
```

```

 <Button variant="outlined" sx={{ py: "11px" }}>
 Add Item From Wishlist
 </Button>
 </div>
)
}

<Snackbar
 anchorOrigin={{ vertical: "top", horizontal: "right" }}
 open={snackbarOpen}
 autoHideDuration={6000}
 onClose={handleCloseSnackbar}
>
 <Alert
 onClose={handleCloseSnackbar}
 severity={coupone.error ? "error" : "success"}
 variant="filled"
 sx={{ width: "100%" }}
 >
 {coupone.error ? coupone.error : "Coupon Applied successfully"}
 </Alert>
</Snackbar>
</>
);
};

export default Cart;
import { Button, Divider, IconButton } from '@mui/material'
import React from 'react'
import RemoveIcon from '@mui/icons-material/Remove';
import AddIcon from '@mui/icons-material/Add';
import CloseIcon from '@mui/icons-material/Close';
import type { CartItem } from '../../../../../types/cartTypes';
import { useDispatch } from '../../../../../Redux Toolkit/Store';
import { deleteCartItem, updateCartItem } from '../../../../../Redux Toolkit/Customer/CartSlice';

interface CartItemProps {
 item:CartItem
}

const CartItemCard : React.FC<CartItemProps> = ({ item }) => {
 const dispatch = useDispatch();

 const handleUpdateQuantity=(value:number)=>{
 dispatch(updateCartItem({jwt:localStorage.getItem("jwt"),
 cartItemId:item.id, cartItem:{quantity:item.quantity + value}}))
 }
 const handleRemoveCartItem=()=>{
 dispatch(deleteCartItem({
 jwt:localStorage.getItem("jwt") || "",


```

```

 cartItemId:item.id}))

 }

 return (

 <div className=' border rounded-md relative'>

 <div className='p-5 flex gap-3'>

 <div>

 </div>

 <div className='space-y-2'>

 <h1 className='font-semibold

text-lg'>{item.product?.seller?.businessDetails.businessName}</h1>

 <p className='text-gray-600 font-medium text-sm'>Turquoise

Blue Stonework Satin Designer Saree</p>

 <p className='text-gray-400 text-xs'>Sold

by: Natural Lifestyle Products Private Limited</p>

 <p className='text-xs'>7 days replacement

available</p>

 <p className='text-sm text-gray-500'>quantity :

 {item.quantity}</p>

 </div>

 </div>

 <Divider />

 <div className='px-5 py-2 flex justify-between items-center'>

 <div className=' flex items-center gap-2 w-[140px]

justify-between'>

 <Button size='small' disabled={item.quantity == 1}

onClick={() => handleUpdateQuantity(-1)}>

 <RemoveIcon />

 </Button>

 {item.quantity}

 <Button size='small' onClick={() => handleUpdateQuantity(1)}>

 <AddIcon />

 </Button>

 </div>

 <div>

```

```
 <p className='text-gray-700
font-medium'>₹{item.sellingPrice}</p>
 </div>

 </div>
 <div className='absolute top-1 right-1'>
 <IconButton onClick={handleRemoveCartItem} color='primary' >
 <CloseIcon />
 </IconButton>
 </div>

 </div>
)
}

export default CartItemCard
```

import { Button, Divider } from "@mui/material";  
import React from "react";  
import { useNavigate } from "react-router-dom";  
import {  
 sumCartItemMrpPrice,  
 sumCartItemSellingPrice,  
} from "../../../../../util/cartCalculator";  
import { useAppSelector } from "../../../../../Redux Toolkit/Store";

const PricingCard = ({ showBuyButton, SubmitButton }: any) => {  
 const navigate = useNavigate();  
 const { cart, auth } = useAppSelector((store) => store);  
 return (  
 <div>  
 <div className="space-y-3 p-5">  
 <div className="flex justify-between items-center">  
 <span>Subtotal</span>  
 <span>₹ {cart.cart?.totalMrpPrice}</span>
 </div>  
 <div className="flex justify-between items-center">  
 <span>Discount</span>  
 <span>  
 ₹{ " " }  
 {sumCartItemMrpPrice(cart.cart?.cartItems || []) -  
 sumCartItemSellingPrice(cart.cart?.cartItems || [])}
 </span>
 </div>  
 <div className="flex justify-between items-center">  
 <span>Shipping</span>  
 <span>₹ 79</span>
 </div>
 <div className="flex justify-between items-center">

```

 plateform fee
 Free
 </div>
</div>
<Divider />

<div className="font-medium px-5 py-2 flex justify-between items-center">
 Total
 ₹ {cart.cart?.totalSellingPrice}
</div>
</div>
);
};

// sumCartItemSellingPrice(cart.cart?.cartItems || [])
// sumCartItemMrpPrice(cart.cart?.cartItems || [])

export default PricingCard;
import React, { useEffect, useRef, useState } from "react";
import { useDispatch, useSelector } from "../../Redux Toolkit/Store";
import { chatBot } from "../../Redux Toolkit/Customer/AiChatBotSlice";
import { Box, Button, IconButton } from "@mui/material";
import SendIcon from "@mui/icons-material/Send";
import PromptMessage from "./PromptMessage";
import ResponseMessage from "./ResponseMessage";
import CloseIcon from '@mui/icons-material/Close';

interface ChatBotProps{
 handleClose:(e:any)=>void;
 productId?:number
}

const ChatBot = ({handleClose,productId}:ChatBotProps) => {
 const dispatch = useDispatch();
 const [prompt, setPrompt] = useState("");
 const chatContainerRef = useRef<HTMLDivElement>(null);
 const [responses, setResponses] = useState<any>([]);
 const [error, setError] = useState("");
 const [loading, setLoading] = useState(false);
 const {aiChatBot}=useSelector(store=>store);

 const handleGivePrompt = (e:any) => {
 e.stopPropagation()
 dispatch(chatBot({ prompt: { prompt }, productId, userId: null }));
 };

 const handlePromptChange = (e: any) => {
 setPrompt(e.target.value);
 };
}

```

```
useEffect(() => {
 if (chatContainerRef.current) {
 chatContainerRef.current.scrollIntoView({ behavior: "smooth" });
 }
}, [aiChatBot.messages]);
// console.log(aiChatBot)
return (
 <div className="rounded-lg">
 <div className="w-full lg:w-[40vw] h-[82vh] shadow-2xl bg-white z-50 rounded-lg">
 <div className="h-[12%] flex justify-between items-center px-5 bg-slate-100 rounded-t-lg">
 <div className="flex items-center gap-3 ">
 <h1 className="logo">Sajid Bazzar</h1>
 <p>Assitant</p>
 </div>
 {/* {productId && <div className="flex items-center gap-3">
 <p>Product id :</p>
 <p>{productId}</p>
 </div>} */}
 <div>
 <IconButton
 onClick={handleClose}
 color="primary"
 >
 <CloseIcon/>
 </IconButton>
 </div>
 </div>
 </div>

 <div className="h-[78%] p-5 flex flex-col py-5 px-5 overflow-y-auto custom-scrollbar">
 <p>welcome to Sajid bazaar Ai Assistant, you can
 {productId?` Query About this Product : ${productId}`:""
 Query about your cart, and order history here"}
 </p>
 { aiChatBot.messages.map((item:any, index:number) =>
 item.role == "user" ? (
 <div ref={chatContainerRef} className="self-end"
key={index}>
 <PromptMessage message={item.message}
index={index} />
 </div>
) : (
 <div
 ref={chatContainerRef}

```

```

 className="self-start"
 key={index}
 >
 <ResponseMessage message={item.message} />
 </div>
)
)
)
}
{aiChatBot.loading && <p>fetching data...</p>

</div>

<div className=" h-[10%] flex items-center">
 <input
 onChange={handlePromptChange}
 type="text"
 placeholder="give your prompt"
 className="rounded-bl-lg pl-5 h-full w-full bg-slate-100
border-none outline-none"
 />
 <Button
 sx={{ borderRadius: "0 0 0.5rem 0" }}
 className="h-full "
 onClick={handleGivePrompt}
 variant="contained"
 >
 <SendIcon />
 </Button>
</div>
</div>
);
};

export default ChatBot;
import React from 'react'

interface PromptMessageProps{
 message:string,
 index:number
}
const PromptMessage = ({message,index}:PromptMessageProps) => {
 return (
 <div className='px-3 py-4'>{message} - {index}</div>
)
}

export default PromptMessage
import React from "react";

```

```

interface ResponseMessageProps {
 message: string;
}

const ResponseMessage = ({ message }: ResponseMessageProps) => {
 return (
 <div className="px-3 py-4 bg-opacity-50 bg-slate-100 rounded-md">
 {message}
 </div>
);
};

export default ResponseMessage;
import { FormControl, FormControlLabel, FormLabel, Radio, RadioGroup } from
'@mui/material'
import React from 'react'
import type { Address } from '../../../../../types/userTypes';

interface AddressCardProps {
 value: number;
 selectedValue: number;
 handleChange: (e: any) => void;
 item: Address
}
const AddressCard: React.FC<AddressCardProps> = ({ value, selectedValue,
handleChange, item }) => {

 return (
 <div className='p-5 border rounded-md flex '>
 <div>
 <Radio
 checked={value == selectedValue}
 onChange={handleChange}
 value={value}
 name="radio-buttons"
 inputProps={{ 'aria-label': 'B' }}
 />
 </div>

 <div className='space-y-3 pt-3'>
 <h1>{item.name}</h1>
 <p className='w-[320px]'>
 {item.address},
 {item.locality},
 {item.city},
 {item.state} - {item.pinCode}</p>
 <p>Mobile : {item.mobile}</p>
 </div>

```

```

 </div>
)
}

export default AddressCard
import React, { useState } from 'react'
import PricingCard from '../Cart/PricingCard'
import { Box, Button, FormControlLabel, Modal, Radio, RadioGroup } from '@mui/material'
import { useNavigate } from 'react-router-dom'
import AddressForm from './AddresssForm'
import AddressCard from './AddressCard'
import AddIcon from '@mui/icons-material/Add';
import { createOrder } from '../../../../../Redux Toolkit/Customer/OrderSlice'
import { useDispatch, useSelector } from '../../../../../Redux Toolkit/Store'

const style = {
 position: 'absolute' as 'absolute',
 top: '50%',
 left: '50%',
 transform: 'translate(-50%, -50%)',
 width: 450,
 bgcolor: 'background.paper',
 boxShadow: 24,
 p: 4,
};

const paymentGatwayList = [
 {
 value: "RAZORPAY",
 image:
 "https://razorpay.com/newsroom-content/uploads/2020/12/output-onlinepngtools-1-1.png",
 label: "Razarpay"
 },
 {
 value: "STRIPE",
 image: "/stripe_logo.png",
 label: "Stripe"
 }
]
const AddressPage = () => {
 const navigate = useNavigate()
 const [value, setValue] = useState(0);
 const dispatch = useDispatch();
 const { user } = useSelector(store => store)
 const [paymentGateway, setPaymentGateway] =
 useState(paymentGatwayList[0].value);

```

```

const [open, setOpen] = React.useState(false);
const handleOpen = () => setOpen(true);
const handleClose = () => setOpen(false);

const handleChange = (event: any) => {
 console.log("----", event.target.value)
 setValue(event.target.value);
};

const handleCreateOrder = () => {
 if (user.user?.addresses)
 dispatch(createOrder({
 paymentGateway,
 address: user.user?.addresses[value],
 jwt: localStorage.getItem('jwt') || ""
 }))
}

const handlePaymentChange = (event: React.ChangeEvent<HTMLInputElement>) =>
{
 setPaymentGateway((event.target as HTMLInputElement).value);
};

return (
 <div className='pt-10 px-5 sm:px-10 md:px-44 lg:px-60 min-h-screen '>
 <div className='space-y-5 lg:space-y-0 lg:grid grid-cols-3 lg:gap-9
'>

 <div className="col-span-2 space-y-5">

 <div className='flex justify-between items-center'>
 Select Dilivery
Address
 <Button onClick={handleOpen} variant='outlined'>Add New
Address</Button>

 </div>
 <div className='text-xs font-medium space-y-5'>
 <p>Saved Addresses</p>
 <div className='space-y-3'>
 {user.user?.addresses?.map((item, index) =>
<AddressCard
 key={item.id}
 item={item}
 selectedValue={value} value={index}
 handleChange={handleChange} />)}
 </div>
 </div>
 <div className='py-4 px-5 rounded-md border'>

```

```

 <Button onClick={handleOpen} startIcon={<AddIcon />}>Add
New Address</Button>

 </div>
 </div>
 <div className="col-span-1 text-sm space-y-3 ">
 <section className='space-y-3 border p-5 rounded-md'>
 <h1 className='text-primary-color font-medium pb-2
text-center'>Chose Payment Gatway</h1>

 <RadioGroup
 row
 aria-labelledby="demo-row-radio-buttons-group-label"
 name="row-radio-buttons-group"
 className='flex justify-between pr-0'
 onChange={handlePaymentChange}
 value={paymentGateway}
 >
 {paymentGatwayList.map((item) => <FormControlLabel
 className={`border w-[45%] flex justify-center rounded-md pr-2 ${paymentGateway
 === item.value ? "border-primary-color" : ""}`}>
 value={item.value}
 control={<Radio />}
 label={<div>
 <img
 className={`${item.value === "stripe" ?
 "w-14" : ""} object-cover`}
 src={item.image}
 alt={item.label}
 />
 </div>}>
)}>
 </RadioGroup>
 </section>
 <section className='border rounded-md'>
 <PricingCard />
 <div className='p-5'>
 <Button
 onClick={handleCreateOrder} sx={{ py: "11px" }}
 variant='contained' fullWidth>Checkout</Button>
 </div>
 </section>
 </div>
</div>

```

```

 <Modal
 open={open}
 onClose={handleClose}
 aria-labelledby="modal-modal-title"
 aria-describedby="modal-modal-description"
 >
 <Box sx={style}>
 <AddressForm paymentGateway={paymentGateway}
 handleClose={handleClose} />
 </Box>
 </Modal>
 </div>
)
}

export default AddressPage
import React from 'react';
import { useFormik } from 'formik';
import * as Yup from 'yup';
import {
 Box,
 Button,
 TextField,
 Typography,
 Grid,
} from '@mui/material';
import { useDispatch } from 'react-redux';
import { useAppDispatch } from '../../../../../Redux Toolkit/Store';
import { createOrder } from '../../../../../Redux Toolkit/Customer/OrderSlice';
import type { Address } from '../../../../../types/userTypes';

// Validation schema
const ContactSchema = Yup.object().shape({
 name: Yup.string().required('Required'),
 mobile: Yup.string()
 .matches(/^[6-9]\d{9}$/, 'Invalid mobile number')
 .required('Required'),
 pinCode: Yup.string()
 .matches(/^\d{6}$/, 'Invalid pincode')
 .required('Required'),
 address: Yup.string().required('Required'),
 locality: Yup.string().required('Required'),
 city: Yup.string().required('Required'),
 state: Yup.string().required('Required'),
});

interface AddressFormProp {
 handleClose: () => void;
 paymentGateway:string
}

```

```

}

const AddressForm:React.FC<AddressFormProp> = ({handleClose,paymentGateway}) =>
{
 const dispatch=useAppDispatch()
 const formik = useFormik({
 initialValues: {
 name: '',
 mobile: '',
 pinCode: '',
 address: '',
 locality: '',
 city: '',
 state: '',
 },
 validationSchema: ContactSchema,
 onSubmit: (values) => {
 console.log("form submited", values);
 handleCreateOrder(values as Address);
 handleClose();
 },
 });
}

const handleCreateOrder=(address:Address)=>{
 dispatch(createOrder({address,jwt:localStorage.getItem('jwt') || "" ,paymentGateway}))
}

return (
 <Box sx={{ maxWidth: 600, mx: 'auto' }}>
 <p className='text-xl font-bold text-center pb-5'>
 Contact Details
 </p>
 <form onSubmit={formik.handleSubmit}>
 <Grid container spacing={3}>
 <Grid item xs={12}>
 <TextField
 fullWidth
 name="name"
 label="Name"
 value={formik.values.name}
 onChange={formik.handleChange}
 onBlur={formik.handleBlur}
 error={formik.touched.name && Boolean(formik.errors.name) }
 helperText={formik.touched.name && formik.errors.name}
 />
 </Grid>
 <Grid item xs={6}>
 <TextField

```

```
 fullWidth
 name="mobile"
 label="Mobile"
 value={formik.values.mobile}
 onChange={formik.handleChange}
 onBlur={formik.handleBlur}
 error={formik.touched.mobile && Boolean(formik.errors.mobile)}
 helperText={formik.touched.mobile && formik.errors.mobile}
 />
</Grid>
<Grid item xs={6}>
 <TextField
 fullWidth
 name="pinCode"
 label="Pin Code"
 value={formik.values.pinCode}
 onChange={formik.handleChange}
 onBlur={formik.handleBlur}
 error={formik.touched.pinCode && Boolean(formik.errors.pinCode)}
 helperText={formik.touched.pinCode && formik.errors.pinCode}
 />
</Grid>
<Grid item xs={12}>
 <TextField
 fullWidth
 name="address"
 label="Address (House No, Building, Street)"
 value={formik.values.address}
 onChange={formik.handleChange}
 onBlur={formik.handleBlur}
 error={formik.touched.address && Boolean(formik.errors.address)}
 helperText={formik.touched.address && formik.errors.address}
 />
</Grid>
<Grid item xs={12}>
 <TextField
 fullWidth
 name="locality"
 label="Locality/Town"
 value={formik.values.locality}
 onChange={formik.handleChange}
 onBlur={formik.handleBlur}
 error={formik.touched.locality && Boolean(formik.errors.locality)}
 helperText={formik.touched.locality && formik.errors.locality}
 />
</Grid>
<Grid item xs={6}>
 <TextField
 fullWidth
```

```

 name="city"
 label="City"
 value={formik.values.city}
 onChange={formik.handleChange}
 onBlur={formik.handleBlur}
 error={formik.touched.city && Boolean(formik.errors.city)}
 helperText={formik.touched.city && formik.errors.city}
 />
 </Grid>
 <Grid item xs={6}>
 <TextField
 fullWidth
 name="state"
 label="State"
 value={formik.values.state}
 onChange={formik.handleChange}
 onBlur={formik.handleBlur}
 error={formik.touched.state && Boolean(formik.errors.state)}
 helperText={formik.touched.state && formik.errors.state}
 />
 </Grid>
 <Grid item xs={12}>
 <Button sx={{py:"14px"}} type="submit" variant="contained"
color="primary" fullWidth>
 Add Address
 </Button>
 </Grid>
 </Grid>
</form>
</Box>
);
};

export default AddressForm;
import React from 'react'

const Banner = () => {
 return (
 <div>

 </div>
)
}

export default Banner
import React from 'react'
import type { Deal } from '../../../../../types/dealTypes'

```

```
import { useNavigate } from 'react-router-dom'

const DealCard = ({deal}:{deal:Deal}) => {
 const navigate=useNavigate();
 return (
 <div onClick={()=>navigate(`/products/${deal.category.categoryId}`)} className='w-full cursor-pointer'>

 <div className='border-4 border-black bg-black text-white p-2 text-center'>
 <p className='text-lg font-semibold'>{deal.category.categoryId.split("_").join(" ")}</p>
 <p className='text-2xl font-bold'>{deal.discount}% OFF</p>
 <p className='text-balance text-lg'>shop now</p>
 </div>
 </div>
)
}

export default DealCard
import React from "react";
import Slider from "react-slick";
import "slick-carousel/slick/slick.css";
import "slick-carousel/slick/slick-theme.css";
import DealCard from "./DealCard";
import { useAppSelector } from "../../../../../Redux Toolkit/Store";
import type { Deal } from "../../../../../types/dealTypes";

export default function DealSlider() {
 const {homePage}=useAppSelector(store=>store)
 var settings = {
 dots: true,
 infinite: true,
 slidesToShow: 6,
 slidesToScroll: 1,
 autoplay: true,
 speed: 2000,
 autoplaySpeed: 2000,
 cssEase: "linear",
 responsive: [
 {
 breakpoint: 1024, // Large screen
 settings: {
 slidesToShow: 4,
 slidesToScroll: 1,
 },
 },
],
 }
}
```

```

 },
 breakpoint: 768, // Tablet
 settings: {
 slidesToShow: 2,
 slidesToScroll: 1,
 },
 },
 {
 breakpoint: 480, // Mobile
 settings: {
 slidesToShow: 1,
 slidesToScroll: 1,
 },
 },
],
};

return (
 <div className=" py-5 lg:px-20">
 <div className="slide-container ">
 <Slider {...settings}>
 {homePage.homePageData?.deals?.map((item:Deal) => <div
 className="border flex flex-col items-center justify-center">
 <DealCard deal={item}/>
 </div>)}
 </Slider>
 </div>
 </div>
);

}
import React from "react";
import ElectronicCategoryCard from "./ElectronicCategoryCard";
import { useMediaQuery } from "@mui/material";
import { useAppSelector } from "../../../../../Redux Toolkit/Store";
const electronics = [
{
 section: "ELECTRIC_CATEGORIES",
 name: "Laptop",
 image:
"https://rukminim2.flixcart.com/image/312/312/xif0q/computer/x/9/j/-original-im
ahyjzh7m2zsqdg.jpeg?q=70",
 categoryId:"laptops"
},
{
 section: "ELECTRIC_CATEGORIES",

```

```
 name: "Mobile",
 image:

"https://rukminim2.flixcart.com/image/416/416/xif0q/mobile/5/t/j/edge-50-fusion
-pb300002in-motorola-original-imahywzrfagkuyxx.jpeg?q=70&crop=false",

 categoryId:"mobiles"
},
{
 section: "ELECTRIC_CATEGORIES",
 name: "Smartwatch",
 image:

"https://rukminim2.flixcart.com/image/612/612/xif0q/smartwatch/f/g/g/-original-
imagynz46fngcks.jpeg?q=70",

 categoryId:"smart_watches"
},
{
 section: "ELECTRIC_CATEGORIES",
 name: "Headphones",
 image:

"https://rukminim2.flixcart.com/image/612/612/kz4gh3k0/headphone/c/v/r/-origina
l-imagb7bmhdgghzxq.jpeg?q=70",

 categoryId:"headphones_headsets"
},
{
 section: "ELECTRIC_CATEGORIES",
 name: "Speaker",
 image:

"https://rukminim2.flixcart.com/image/612/612/xif0q/speaker/6/z/2/-original-ima
hgfkr5gkk9aq.jpeg?q=70",

 categoryId:"speakers"
},
{
 section: "ELECTRIC_CATEGORIES",
 name: "Tv",
 image:

"https://rukminim2.flixcart.com/image/312/312/xif0q/television/9/p/9/-original-
imah2v29z86u7b79.jpeg?q=70",

 categoryId:"television"
},
```

```

{
 section: "ELECTRIC_CATEGORIES",
 name: "Camera",
 image:
 "https://rukminim2.flixcart.com/image/312/312/jfbfde80/camera/n/r/n/canon-eos-eos-3000d-dslr-original-imaf3t5h9yuyc5zu.jpeg?q=70",
 categoryId:"cameras"
},
];
};

const ElectronicCategory = () => {
 const {homePage}=useAppSelector(store=>store)
 const isSmallScreen = useMediaQuery("(max-width:600px)");
 return (
 <div className="flex flex-wrap justify-between py-5 lg:px-20 border-b">
 {homePage.homePageData?.electricCategories
 .slice(0, isSmallScreen ? 5 : electronics.length)
 .map((item) => (
 <ElectronicCategoryCard item={item} />
)));
 </div>
);
};

export default ElectronicCategory;
import React from 'react'
import { useNavigate } from 'react-router-dom'

const ElectronicCategoryCard = ({item}:any) => {
 const navigate=useNavigate();

 return (
 <div onClick={()=>navigate(`/products/${item.categoryId}`)} className='flex w-20 flex-col items-center gap-3 cursor-pointer'>

 <h2 className='font-semibold text-sm text-center'>{item.name}</h2>
 </div>
)
}

export default ElectronicCategoryCard
import React from 'react'
import HomeCategoryCard from './HomeCategoryCard'
import { useAppSelector } from '../../../../../Redux Toolkit/Store';

```

```

const homeCategory=[

{
 "name": "Home Décor",
 "categoryId": "home_decor",
 "parentCategoryName": "Furniture",
 "parentCategoryId": "furniture",

 "level": 2,
 "section": "SHOP_BY_CATEGORIES",

image:"https://assets.myntassets.com/h_720,q_90,w_540/v1/assets/images/28460938
/2024/3/22/7fb09e9c-86e0-4602-b54e-fa5c0171b50b1711104156746IrregularMirrorHome
Decor1.jpg"
},

{
 "name": "Kitchen & Table",
 "categoryId": "kitchen_table",
 "parentCategoryName": "Furniture",
 "parentCategoryId": "furniture",

 "level": 2,
 "section": "SHOP_BY_CATEGORIES",

image:"https://assets.myntassets.com/h_720,q_90,w_540/v1/assets/images/19873492
/2022/9/10/fd72939c-c379-4bab-9fd6-918c89172d161662797928387AquarelleBlue100Cot
tonPrintedTableRunner1.jpg"
},

{
 "parentCategoryId": "women",
 "level": 2,
 "name": "Sports & Active Wear",
 "categoryId": "women_sports_active_wear",
 "section": "SHOP_BY_CATEGORIES",

image:"https://assets.myntassets.com/h_1440,q_90,w_1080/v1/assets/images/221094
80/2023/9/5/06a17ac3-46b0-4f9d-bcb1-2d3582fed041693895310152PumaWomenBrandLogo
PrintedPureCottonOutdoorT-shirt1.jpg"
},

{
 "parentCategoryId": "women",
 "level": 2,
 "name": "Lingerie Sleepwear",
 "categoryId": "women_lingerie_sleepwear",
 "section": "SHOP_BY_CATEGORIES",

image:"https://assets.myntassets.com/h_720,q_90,w_540/v1/assets/images/362895/2
023/10/18/bfb55058-f1a-45d6-a39d-1b8d421e02001697603774147TriumphShapeSensatio
n33withHighWaistTummyandThighControlMaxi1.jpg"
}

```

```
},

{
 "parentCategoryId": "women",
 "level": 2,
 "name": "Indian & fusion Wear",
 "categoryId": "women_indian_and_fusion_wear",
 "section": "SHOP_BY_CATEGORIES",

 image: "https://assets.myntassets.com/h_720,q_90,w_540/v1/assets/images/22866694
/2023/4/24/98951db4-e0a5-47f8-a1be-353863d24dc01682349679664KalinjiOrangeSilkBle
ndEthnicWovenDesignFestiveSareewithMatchi2.jpg"
,
{
 "parentCategoryId": "women",
 "level": 2,
 "name": "western wear",
 "categoryId": "women_western_wear",
 "section": "SHOP_BY_CATEGORIES",

 image: "https://assets.myntassets.com/h_720,q_90,w_540/v1/assets/images/22391504
/2023/3/17/3259c109-060a-4c39-aba2-e9d32e2068e41679049035856StyleQuotientPeach-
ColouredTie-UpNeckPuffSleeveCottonTop1.jpg"
,
{
 "parentCategoryId": "women",
 "level": 2,
 "name": "Women Footwear",
 "categoryId": "women_footwear",
 "section": "SHOP_BY_CATEGORIES",

 image: "https://assets.myntassets.com/h_720,q_90,w_540/v1/assets/images/28024048
/2024/3/5/fca98389-f9d6-4f19-b82a-53c7ee0518ec1709633175836CORSICABlockSandalsw
ithBows1.jpg"
,
{
 "name": "Topwere",
 "categoryId": "men_topwear",
 "parentCategoryId": "men",
 "level": 2,
 "section": "SHOP_BY_CATEGORIES",

 image: "https://assets.myntassets.com/h_720,q_90,w_540/v1/assets/images/23029834
/2023/9/18/96c015ae-1090-4036-954b-d9c80085b1d71695022844653-HRX-by-Hrithik-Ros
han-Men-Jackets-6981695022843934-1.jpg"
,
{
 "name": "Bottomwere",
 "categoryId": "men_bottomwear",
```

```
"parentCategoryId": "men",
"level": 2,
"section": "SHOP_BY_CATEGORIES",

image: "https://assets.myntassets.com/h_720,q_90,w_540/v1/assets/images/20122324
/2022/9/22/91c61c45-fe17-4d1d-8e20-0aaaf90186b61663827920015RaymondSlimFitBlueJeansForMen1.jpg"
},
{
 "name": "Innerwere And Sleepwere",
 "categoryId": "men_innerwear_and_sleepwear",
 "parentCategoryId": "men",
 "level": 2,
 "section": "SHOP_BY_CATEGORIES",

image: "https://assets.myntassets.com/h_720,q_90,w_540/v1/assets/images/24528350
/2023/8/17/d9ee03c9-7e15-49e4-8f15-0b6f38e568121692275415567TrackPants1.jpg"
},
{
 "name": "Footwere",
 "categoryId": "men_footwear",
 "parentCategoryId": "men",
 "level": 2,
 "section": "SHOP_BY_CATEGORIES",

image: "https://assets.myntassets.com/h_720,q_90,w_540/v1/assets/images/27107914
/2024/1/30/bd09f650-d8a4-4570-8e53-e2cfafc4ea6c1706609911015SparxMenMeshRunning
Shoes1.jpg"
},
{
 "name": "Bed Linen & Furnishing",
 "categoryId": "bed_linen_furnishing",
 "parentCategoryName": "Furniture",
 "parentCategoryId": "furniture",
 "level": 2,
 "section": "SHOP_BY_CATEGORIES",

image: "https://assets.myntassets.com/h_720,q_90,w_540/v1/assets/images/19284508
/2022/7/28/92df52de-27dc-4d72-8ab4-fee2c82c85081659003977664DreamscapeUnisexPin
kBedsheets1.jpg"
},
{
 "name": "Flooring",
 "categoryId": "flooring",
 "parentCategoryName": "Furniture",
 "parentCategoryId": "furniture",
 "level": 2,
```

```

 "section": "SHOP_BY_CATEGORIES",
 image: "https://assets.myntassets.com/h_720,q_90,w_540/v1/assets/images/27868878
 /2024/2/25/53b7c07f-24a7-48e2-a791-f63dce4a0c981708844793060SANACARPETMulticoloredFloralAnti-SkidPolyesterCarpet1.jpg"
 },
 {
 "name": "Bath",
 "categoryId": "bath",
 "parentCategoryName": "Furniture",
 "parentCategoryId": "furniture",
 "level": 2,
 "section": "SHOP_BY_CATEGORIES",
 image: "https://assets.myntassets.com/h_720,q_90,w_540/v1/assets/images/16719378
 /2024/7/2/97418bad-8216-494f-a863-9b159aec93cf1719897710069JockeyPackof2CottonTerry500GSMUltrasoftandDurableSolidHandTo1.jpg"
 },
 {
 "name": "Lamps & Lighting",
 "categoryId": "lamps_lighting",
 "parentCategoryName": "Furniture",
 "parentCategoryId": "furniture",
 "level": 2,
 "section": "SHOP_BY_CATEGORIES",
 image: "https://assets.myntassets.com/h_720,q_90,w_540/v1/assets/images/producti
 mage/2021/6/10/5be80654-037c-4591-99c9-a403646971981623344562713-1.jpg"
 },
]

const HomeCategory = () => {
 const { homePage } = useAppSelector((store) => store);
 return (
 <div className='flex justify-center gap-7 flex-wrap'>
 {homePage.homePageData?.shopByCategories.map((item) => <HomeCategoryCard
item={item} />)}
 </div>
)
}

export default HomeCategory
.custom-border {

```

```

 border: 9px solid;
 border-color: rgb(255, 58, 91) rgb(99, 99, 99);
 }

 .custom-border img {
 border-radius: 50%;
 }

import React from 'react'
import "./HomeCategoryCard.css"
import { useNavigate } from 'react-router-dom'

const HomeCategoryCard = ({item}:any) => {
 const navigate=useNavigate()
 return (
 <div onClick={()=>navigate(`/products/${item.categoryId}`)} className='flex gap-3 flex-col justify-center items-center group cursor-pointer'>
 <div className='custom-border w-[150px] lg:w-[249px] h-[150px] lg:h-[249px] rounded-full bg-teal-400'>

 </div>
 <h1 className='font-medium'>{item.name}</h1>
 </div>
)
}

export default HomeCategoryCard
export const homeCategoryData=[]
import React from "react";
import { useSelector } from "../../../../../Redux Toolkit/Store";

const grid = [
 {"categoryId": "women_lehenga_cholis",
 "section": "GRID",
 "name": "women lehenga cholis",
 image:
 "https://assets.myntassets.com/h_720,q_90,w_540/v1/assets/images/23807268/2023/6/29/9930b235-5318-4755-abbe-08f99e969e781688026636544LehengaCholi7.jpg",
 },
 {"categoryId": "men_formal_shoes",
 "section": "GRID",
 "name": "men formal shoes",
 image:
 "https://assets.myntassets.com/h_720,q_90,w_540/v1/assets/images/24651572/2023/8/25/4fbf6d8c-d093-46c5-a5a6-7dd67c0c76551692964752597HouseofPataudiMenTanFauxLeatherFormalSlipOnLoafers1.jpg",
 }
]

```

```
},
{"categoryId":"women_lehenga_cholis",
 "section": "GRID",
 "name": "women lehenga cholis",
 image:

"https://images.pexels.com/photos/12730873/pexels-photo-12730873.jpeg?auto=compress&cs=tinysrgb&w=1260&h=750&dpr=1",
},
{"categoryId":"men_sherwanis",
 "section": "GRID",
 "name": "men sherwanis",
 image:

"https://shreeman.in/cdn/shop/files/20_3cfbd5a3-ecb6-482a-b798-7ffd9de1c784.jpg?v=1712061674&width=700",
},
 {"categoryId":"women_jewellery",
 "section": "GRID",
 "name": "women jewellery",
 image:

"https://media.istockphoto.com/id/1276740597/photo/indian-traditional-gold-necklace.jpg?b=1&s=612x612&w=0&k=20&c=S-QnNZKqf2u3L-GIaDiIinNRU74GBWQaIDwY7gYJboY="
,
},
 {"categoryId":"women_footwear",
 "section": "GRID",
 "name": "women footwear",
 image:

"https://assets.myntassets.com/h_720,q_90,w_540/v1/assets/images/13837166/2021/8/19/04e40e02-4c56-4705-94d0-f444b29973aa1629373611707-House-of-Pataudi-Women-Maroon-Embellished-Handcrafted-Wedges-1.jpg",
},
];
const TopBrand = () => {
 const {homePage}=useAppSelector(store=>store)
 return (
 <div className="grid gap-4 grid-rows-12 grid-cols-12 lg:h-[600px] px-5 lg:px-20">
 <div className=" col-span-3 row-span-12 text-white rounded ">
 <img
 className="w-full h-full object-cover border-fuchsia-800 lg:border-[9px]s rounded-md"
 src={homePage.homePageData?.grid[0].image}
 alt=""
 />
 </div>
 </div>
)
}
```

```
<div className="col-span-2 row-span-6 text-white rounded">
 <img
 className="w-full h-full object-cover border-fuchsia-800
lg:border-[9px]s rounded-md"
 src={homePage.homePageData?.grid[1].image}
 alt=""
 />
</div>

<div className="col-span-4 row-span-6 text-white rounded ">
 <img
 className="w-full h-full object-cover object-top border-fuchsia-800
lg:border-[9px]s rounded-md"
 src={homePage.homePageData?.grid[2].image}
 alt=""
 />
</div>

<div className="col-span-3 row-span-12 text-white rounded ">
 <img
 className="w-full h-full object-cover object-top border-fuchsia-800
lg:border-[9px]s rounded-md"
 src={homePage.homePageData?.grid[3].image}
 alt=""
 />
</div>

<div className="col-span-4 row-span-6 text-white rounded ">
 <img
 className="w-full h-full object-cover object-top border-fuchsia-800
lg:border-[9px]s rounded-md"
 src={homePage.homePageData?.grid[4].image}
 alt=""
 />
</div>
<div className="col-span-2 row-span-6 text-white rounded ">
 <img
 className="w-full h-full object-cover border-fuchsia-800
lg:border-[9px]s rounded-md"
 src={homePage.homePageData?.grid[5].image}
 alt=""
 />
</div>

 /* https://tristenwallace.com/wp-content/uploads/2022/06/wed-7.jpg */
</div>
) ;
} ;
```

```
export default TopBrand;
export const topBrandData=[
 {name:"Rubans",
 image:"https://assets.myntassets.com/h_720,q_90,w_540/v1/assets/images/15189594/2022/2/1/efde4d54-98ca-40cc-89f1-b5787bc532b11643659231979Rubans24KGold-PlatedRuby-StuddedBeadedHandcraftedJewellerySe5.jpg"},

 {
 name:"Fabcartz",
 image:"https://assets.myntassets.com/h_720,q_90,w_540/v1/assets/images/23807268/2023/6/29/9930b235-5318-4755-abbe-08f99e969e781688026636544LehengaCholi7.jpg"},

 },
 {name:"manyawar",
 image:"https://shreeman.in/cdn/shop/files/20_3cfbd5a3-ecb6-482a-b798-7ffd9de1c784.jpg?v=1712061674&width=700"},

 },
 {
 name:"LOUIS STITCH",
 image:"https://assets.myntassets.com/h_720,q_90,w_540/v1/assets/images/24651572/2023/8/25/4fbf6d8c-d093-46c5-a5a6-7dd67c0c76551692964752597HouseofPataudiMenTanFauxLeatherFormalSlipOnLoafers1.jpg"},

 },
 {
 name:"Anouk",
 image:"https://assets.myntassets.com/h_720,q_90,w_540/v1/assets/images/13837166/2021/8/19/04e40e02-4c56-4705-94d0-f444b29973aa1629373611707-House-of-Pataudi-Women-Maroon-Embellished-Handcrafted-Wedges-1.jpg"},

 }
]

import React, { useState } from 'react'
import Banner from './Banner/Banner'
import HomeCategory from './HomeCategory/HomeCategory'
import TopBrand from './TopBrands/Grid'
import ElectronicCategory from './Electronic Category/ElectronicCategory'
import ChatBubbleIcon from '@mui/icons-material/ChatBubble';
import { Backdrop, Button, CircularProgress } from '@mui/material'
import ChatBot from '../ChatBot/ChatBot'
import { useNavigate } from 'react-router-dom'
import StorefrontIcon from '@mui/icons-material/Storefront';
import { useSelector } from '....Redux Toolkit/Store'
import DealSlider from './Deals/Deals'
```

```

const Home = () => {
 const [showChatBot, setShowChatBot] = useState(false)
 const { homepage } = useAppSelector(store => store)
 const navigate = useNavigate()

 const handleShowChatBot = () => {
 setShowChatBot(!showChatBot)
 }
 const handleCloseChatBot = () => {
 setShowChatBot(false)
 }
 const becomeSellerClick = () => {
 navigate("/become-seller")
 }
 return (
 <>
 {(!homepage.loading)?<div className='space-y-5 lg:space-y-10 relative'>
 {homepage.homepageData?.electricCategories && <ElectronicCategory
 />}
 {/* <Banner /> */}

 {homepage.homepageData?.grid && <section >
 /* <h1 className='text-lg lg:text-4xl font-bold text-[#00927c] pb-5 lg:pb-20 text-center'>SHOP FOR WEDDING</h1> */
 <TopBrand />
 </section>}
 {homepage.homepageData?.deals && <section className='pt-10'>
 <h1 className='text-center text-lg lg:text-4xl font-bold text-[#00927c] pb-5 lg:pb-10'>Today's Deals</h1>
 <DealSlider/>
 </section>}
 {homepage.homepageData?.shopByCategories && <section className='flex flex-col justify-center items-center py-20 px-5 lg:px-20'>
 <h1 className='text-lg lg:text-4xl font-bold text-[#00927c] pb-5 lg:pb-20'>SHOP BY CATEGORY</h1>
 <HomeCategory />
 </section>}
 <section className='lg:px-20 relative h-[200px] lg:h-[450px] object-cover'>

 <div className='absolute top-1/2 left-4 lg:left-[15rem] transform -translate-y-1/2 font-semibold lg:text-4xl space-y-3 '>
 <h1 className='>
 Sell Your Product
 </h1>
 </div>
 </section>
)
}

```

```

 <p className='text-lg md:text-2xl'>With <strong
className='logo text-3xl md:text-5xl pl-2'>Sajid bazaar</p>

 <div className='pt-6 flex justify-center'>
 <Button
 onClick={becomeSellerClick}
 startIcon={<StorefrontIcon />}
 variant="contained"
 >
 Become Seller
 </Button>
 </div>

 </div>

</section>

<section className='fixed bottom-10 right-10'>
 {showChatBot ? <ChatBot handleClose={handleCloseChatBot} /> :
<Button onClick={handleShowChatBot} sx={{ borderRadius: "2rem" }}
variant='contained' className='h-16 w-16 flex justify-center items-center rounded-full'>
 <ChatBubbleIcon sx={{ color: "white", fontSize: "2rem" }} />
</Button>}
</section>

</div>: <Backdrop
 open={true}

 >
 <CircularProgress color="inherit" />
</Backdrop>

</>

)
}

export default Home
import React from 'react'

const NotFound = () => {
 return (

```

```
<div className='h-screen flex justify-center items-center'>
 <h1 className='text-xl lg:text-5xl font-bold'>Page Not Found</h1>
</div>
)

}

export default NotFound

.card {
 @apply relative w-[250px] h-[350px] overflow-hidden;
}

.card-media {
 position: absolute;
 top: 0;
 left: 0;
 width: 100%;
 height: 100%;
 transition: transform 0.5s ease-in-out;
 cursor: pointer;
 object-fit: cover;
}

.indicator {
 position: absolute;
 bottom: 16px;
 left: 50%;
 transform: translateX(-50%);
 /* display: flex;
 gap: 8px; */
}

.indicator-button {
 width: 10px;
 height: 10px;
 background-color: rgba(255, 255, 255, 0.5);
 border: none;
 border-radius: 50%;
 cursor: pointer;
}

.indicator-button.active {
 background-color: rgba(255, 255, 255, 1);
}

.thin-line-through {
 text-decoration: line-through;
```

```

 text-decoration-thickness: 1px; /* Adjust the thickness as needed */
 }

 /* styles.css */
.group-hover-effect {
 padding: 1rem; /* Equivalent to p-4 */
 transition: transform 0.3s; /* Equivalent to transition-transform duration-300 */
}
}

.group:hover .group-hover-effect {
 transform: translateY(0.25rem); /* Equivalent to translate-y-1 */
 box-shadow: 0 4px 6px rgba(0, 0, 0, 0.1); /* Equivalent to shadow-md */
}

import React, { useState, useEffect, MouseEvent } from "react";
import "./ProductCard.css";
import FavoriteIcon from "@mui/icons-material/Favorite";
import { teal } from "@mui/material/colors";
import { Box, Button, IconButton, Modal } from "@mui/material";
import { useNavigate } from "react-router-dom";
import type { Product } from "../../../../../types/productTypes";
import {
 useAppDispatch,
 useAppSelector,
} from "../../../../../Redux Toolkit/Store";
import { addProductToWishlist } from "../../../../../Redux Toolkit/Customer/WishlistSlice";
import FavoriteBorderIcon from "@mui/icons-material/FavoriteBorder";
import { isWishlisted } from "../../../../../util/isWishlisted";
import ModeCommentIcon from '@mui/icons-material/ModeComment';
import ChatBot from "../../../../../ChatBot/ChatBot";

interface ProductCardProps {
 // images: string[];
 // categoryId: string | undefined;
 item: Product;
}

const style = {
 position: 'absolute' as 'absolute',
 top: '50%',
 left: '50%',
 transform: 'translate(-50%, -50%)',
 width: "auto",
 borderRadius: ".5rem",
 boxShadow: 24,
};

const ProductCard: React.FC<ProductCardProps> = ({ item }) => {

```

```

const [currentImage, setCurrentImage] = useState(0);
const [isHovered, setIsHovered] = useState(false);
const [isFavorite, setIsFavorite] = useState(false);
const { wishlist } = useAppSelector((store) => store);
const navigate = useNavigate();
const dispatch = useAppDispatch();
const [showChatBot, setShowChatBot] = useState(false)

const handleAddWishlist = (event: MouseEvent) => {
 event.stopPropagation();
 setIsFavorite((prev) => !prev);
 if (item.id) dispatch(addProductToWishlist({ productId: item.id }));
}

useEffect(() => {
 let interval: any;
 if (isHovered) {
 interval = setInterval(() => {
 setCurrentImage((prevImage) => (prevImage + 1) %
item.images.length);
 }, 1000); // Change image every 1 second
 } else if (interval) {
 clearInterval(interval);
 }
 return () => clearInterval(interval);
}, [isHovered, item.images.length]);

const handleShowChatBot = (event: MouseEvent) => {
 event.stopPropagation();
 setShowChatBot(true)
}
const handleCloseChatBot = (e: MouseEvent) => {
 e.stopPropagation();
 setShowChatBot(false)
}

return (
 <>
 <div
 onClick={() =>
 navigate(
 `/product-details/${item.category?.categoryId}/${item.title}/${item.id}`
)
 }
 className="group px-4 relative"
 >
 <div
 className="card "

```

```

 onMouseEnter={() => setIsHovered(true)}
 onMouseLeave={() => setIsHovered(false)}
 >
 {item.images.map((image: any, index: number) => (
 <img
 key={index}
 className="card-media object-top"
 src={image}
 alt={`product-${index}`}
 style={{
 transform: `translateX(${(index - currentImage) * 100}%)`,
 }}>
))}>
 {isHovered && (
 <div className="indicator flex flex-col items-center space-y-2">
 <div className="flex gap-4">
 {item.images.map((item: any, index: number) => (
 <button
 key={index}
 className={`${'indicator-button ${index === currentImage ? "active" : ""}'}`}
 onClick={() => setCurrentImage(index)}>
))}>
 </div>
 <div className="flex gap-3">
 {wishlist.wishlist && (
 <Button
 variant="contained"
 color="secondary"
 sx={{ zIndex: 10 }}
 className="z-50"
 onClick={handleAddWishlist}>
)>
 {isWishlisted(wishlist.wishlist, item) ?
 (
 <FavoriteIcon sx={{ color: teal[500] }} />
) :
 (
 <FavoriteBorderIcon sx={{ color: "gray" }} />
)
 }>
 </div>
 </div>
)}>

```

```

) }
 <Button onClick={handleShowChatBot}
color="secondary" variant="contained">
 <ModeCommentIcon sx={{ color: teal[500] }} />
</div>

 </div>
) }
</div>
<div className="details pt-3 space-y-1 group-hover-effect rounded-md ">
 <div className="name space-y ">
 <h1 className="font-semibold text-lg">
 {item.seller?.businessDetails.businessName}
 </h1>
 <p className="">{item.title}</p>
 </div>
 <div className="price flex items-center gap-3 ">

 {" "}
 ₹{item.sellingPrice}

 ₹{item.mrpPrice}

 {item.discountPercent}% off

 </div>
</div>

</div>
{showChatBot && <section className="absolute left-16 top-0">
 <Modal
 open={true}
 onClose={handleCloseChatBot}
 aria-labelledby="modal-modal-title"
 aria-describedby="modal-modal-description"
 >
 <Box sx={style}>
 <ChatBot handleClose={handleCloseChatBot} productId={item.id} />
 </Box>
 </Modal>
</section>}

```

```

 </>
);
};

export default ProductCard;
export const productDetailsData={
 images:[
 "https://www.karagiri.com/cdn/shop/products/patola-saree-hot-pink-patola-saree-silk-saree-online-32261950898369.jpg?v=1704965372",
 "https://www.karagiri.com/cdn/shop/products/patola-saree-hot-pink-patola-saree-silk-saree-online-32261950996673_540x.jpg?v=1704965372",
 "https://www.karagiri.com/cdn/shop/products/patola-saree-hot-pink-patola-saree-silk-saree-online-32261950931137_540x.jpg?v=1704965372",
 "https://www.karagiri.com/cdn/shop/products/patola-saree-hot-pink-patola-saree-silk-saree-online-32261951029441_540x.jpg?v=1704965372",
 "https://www.karagiri.com/cdn/shop/products/patola-saree-hot-pink-patola-saree-silk-saree-online-32261950865601_540x.jpg?v=1704965372"
]
}
import React, { useEffect, useState } from 'react'
import StarIcon from '@mui/icons-material/Star';
import { teal } from '@mui/material/colors';
import { Box, Button, Divider, Grid, IconButton, LinearProgress, Modal, Rating } from '@mui/material';
import ShieldIcon from '@mui/icons-material/Shield';
import LocalShippingIcon from '@mui/icons-material/LocalShipping';
import WorkspacePremiumIcon from '@mui/icons-material/WorkspacePremium';
import AccountBalanceWalletIcon from '@mui/icons-material/AccountBalanceWallet';
import { Wallet } from '@mui/icons-material';
import RemoveIcon from '@mui/icons-material/Remove';
import AddIcon from '@mui/icons-material/Add';
import AddShoppingCartIcon from '@mui/icons-material/AddShoppingCart';
import FavoriteBorderIcon from '@mui/icons-material/FavoriteBorder';
import SmilarProduct from '../SimilarProduct/SmilarProduct';
import ZoomableImage from './ZoomableImage';
import { useDispatch, useSelector } from '../../../../../Redux Toolkit/Store';
import { useNavigate, useParams } from 'react-router-dom';
import { fetchProductById, getAllProducts } from '../../../../../Redux Toolkit/Customer/ProductSlice';
import { addItemToCart } from '../../../../../Redux Toolkit/Customer/CartSlice';
import ProductReviewCard from '../../../../../Review/ProductReviewCard';
import RatingCard from '../../../../../Review/RatingCard';

```

```
import { fetchReviewsByProductId } from '../../../../../Redux
Toolkit/Customer/ReviewSlice';

const style = {
 position: 'absolute' as 'absolute',
 top: '50%',
 left: '50%',
 transform: 'translate(-50%, -50%)',
 width: "auto",
 height: "100%",
 // backgroundColor: 'background.paper',
 boxShadow: 24,
 outline: "none",
};

const ProductDetails = () => {
 const [open, setOpen] = React.useState(false);
 const handleOpen = () => setOpen(true);
 const handleClose = () => setOpen(false);
 const dispatch = useAppDispatch();
 const { products, review } = useAppSelector(store => store)
 const navigate = useNavigate()
 const { productId, categoryId } = useParams()
 const [selectedImage, setSelectedImage] = useState(0);
 const [quantity, setQuantity] = useState(1)

 useEffect(() => {

 if (productId) {
 dispatch(fetchProductById(Number(productId)))
 dispatch(fetchReviewsByProductId({ productId: Number(productId) }))
 }
 dispatch(getAllProducts({ category: categoryId }));

 }, [productId])

 const handleAddCart = () => {
 dispatch(addItemToCart({
 jwt: localStorage.getItem('jwt'),
 request: { productId: Number(productId), size: "FREE", quantity }

 }))
 }
}
```

```

 return (
 <div className='px-5 lg:px-20 pt-10 '>
 <div className='grid grid-cols-1 lg:grid-cols-2 gap-10'>

 <section className='flex flex-col lg:flex-row gap-5'>
 <div className='w-full lg:w-[15%] flex flex-wrap lg:flex-col gap-3'>
 {products.product?.images.map((item, index) => setSelectedImage(index)} className='lg:w-full w-[50px]
 cursor-pointer rounded-md' src={item} alt="" />)
 </div>
 <div className='w-full lg:w-[85%]'>
 <img onClick={handleOpen} className='w-full rounded-md
 cursor-zoom-out' src={products.product?.images[selectedImage]} alt="" />
 </div>

 <Modal
 open={open}
 onClose={handleClose}
 aria-labelledby="modal-modal-title"
 aria-describedby="modal-modal-description"
 >
 <Box sx={style}>

 <ZoomableImage
 src={products.product?.images[selectedImage]} alt="" />
 </Box>
 </Modal>
 </section>
 </div>
 </div>
)
)
}

const handleOpen = () => {
 setOpen(true);
}

const handleClose = () => {
 setOpen(false);
}

const [selectedImage, setSelectedImage] = useState(0);

```

```
<div className='space-y-2'>
 <div className='price flex items-center gap-3 mt-5
text-lg'>

₹{products.product?.sellingPrice}
 <span className='text thin-line-through
text-gray-400 '>₹{products.product?.mrpPrice}
 <span className='text-[#00927c]
font-semibold'>{products.product?.discountPercent}% off
 </div>
 <p className='text-sm'>Inclusive of all taxes. Free
Shipping above ₹1500.</p>
</div>

<div className='mt-7 space-y-3'>

 <div className='flex items-center gap-4'>
 <ShieldIcon sx={{ color: teal[400] }} />
 <p>Authentic & Quality Assured</p>
 </div>

 <div className='flex items-center gap-4'>
 <WorkspacePremiumIcon sx={{ color: teal[400] }} />
 <p>100% money back guarantee</p>
 </div>

 <div className='flex items-center gap-4'>
 <LocalShippingIcon sx={{ color: teal[400] }} />
 <p>Free Shipping & Returns</p>
 </div>

 <div className='flex items-center gap-4'>
 <Wallet sx={{ color: teal[400] }} />
 <p>Pay on delivery might be available</p>
 </div>

</div>

<div className='mt-7 space-y-2'>
 <h1>QUANTITY:</h1>
 <div className=' flex items-center gap-2 w-[140px]
justify-between'>
```

```
 <Button disabled={quantity == 1} onClick={() =>
setQuantity(quantity - 1)} variant='outlined'>
 <RemoveIcon />
 </Button>

 {quantity}

 <Button onClick={() => setQuantity(quantity + 1)} variant='outlined'>
 <AddIcon />
 </Button>
 </div>
</div>

<div className="mt-12 flex items-center gap-5">
 <Button
 onClick={handleAddCart}
 sx={{ py: "1rem" }}
 variant='contained' fullWidth
startIcon={<AddShoppingCartIcon />}>
 Add To Bag
 </Button>
 <Button
 sx={{ py: "1rem" }}
 variant='outlined' fullWidth
startIcon={<FavoriteBorderIcon />}>
 Whishlist
 </Button>
</div>
<div className='mt-5'>
 <p >
 {products.product?.description}
 </p>
</div>
<div className="ratings w-full mt-10">
 <h1 className="font-semibold text-lg pb-4">
 Review & Ratings
 </h1>

 <RatingCard totalReview={review.reviews.length} />
 <div className='mt-10'>
 <div className="space-y-5">
 {review.reviews.map((item, i) => (
 <div className='space-y-5'>
 <ProductReviewCard item={item} />
 <Divider />
 </div>
))}
 </div>
 </div>

```

```

)) }
 <Button onClick={() =>
navigate(`/reviews/${productId}`)}>View All {review.reviews.length}
Reviews</Button>
 </div>
</div>

 </div>
</section>

 </div>
<section className='mt-20'>
 <h1 className='text-lg font-bold'>Similar Product</h1>

 <div className='pt-5'>
 <SmilarProduct />
 </div>

 </section>
</div>
)
}

export default ProductDetails
import React, { useState, useRef, useEffect, MouseEvent } from 'react';

interface ZoomableImageProps {
 src: string | undefined;
 alt: string;
}

interface Offset {
 x: number;
 y: number;
}

const ZoomableImage: React.FC<ZoomableImageProps> = ({ src, alt }) => {
 const [isZoomed, setIsZoomed] = useState<boolean>(false);
 const [offset, setOffset] = useState<Offset>({ x: 0, y: 0 });
 const [start, setStart] = useState<Offset>({ x: 0, y: 0 });
 const [isDragging, setIsDragging] = useState<boolean>(false);
 const imgRef = useRef<HTMLImageElement>(null);

 const handleMouseDown = (e: MouseEvent<HTMLDivElement>) => {
 if (e.button === 0) {

```

```
 setStart({ x: e.clientX - offset.x, y: e.clientY - offset.y });
 setIsDragging(true);
 e.preventDefault();
 }
};

const handleMouseMove = (e: MouseEvent<HTMLDivElement>) => {
 if (isDragging) {
 setOffset({
 x: e.clientX - start.x,
 y: e.clientY - start.y,
 });
 }
};

const handleMouseUp = () => {
 setIsDragging(false);
};

const handleContextMenu = (e: MouseEvent<HTMLDivElement>) => {
 e.preventDefault();
};

const toggleZoom = () => {
 setIsZoomed(!isZoomed);
 setOffset({ x: 0, y: 0 });
 console.log("toggle zoom ----- ", isZoomed)
};

useEffect(() => {
 if (imgRef.current) {
 imgRef.current.style.cursor = isDragging ? 'grabbing' : 'grab';
 }
}, [isDragging]);

return (
 <div
 style={{
 overflow: 'hidden',
 cursor: isZoomed ? 'zoom-out' : 'zoom-in',
 width: isZoomed ? '100%' : '100%',
 height: 'auto',
 position: 'relative',
 }}
 onClick={toggleZoom}
 onMouseDown={handleMouseDown}
 onMouseMove={handleMouseMove}
 onMouseUp={handleMouseUp}
)
```

```
onMouseLeave={handleMouseUp}
onContextMenu={handleContextMenu}
>
<img
 ref={imgRef}
 src={src}
 alt={alt}
 style={{
 width: isZoomed ? '200%' : '200',
 height: isZoomed ? '200%' : 'auto',
 transform: `translate(${offset.x}px, ${offset.y}px)`,
 transition: isDragging ? 'none' : 'transform 0.3s',
 userSelect: 'none',
 }}
/>
</div>
);
};

export default ZoomableImage;
import React, { useState, useEffect } from 'react';
import FavoriteIcon from '@mui/icons-material/Favorite';
import { teal } from '@mui/material/colors';
import { IconButton } from '@mui/material';
import { useNavigate } from 'react-router-dom';
import type { Product } from '../../../../../types/productTypes';

const SimilarProductCard = ({ product }: any) => {
 const [currentImage, setCurrentImage] = useState(0);
 const [isHovered, setIsHovered] = useState(false);
 const [isFavorite, setIsFavorite] = useState(false);
 const navigate=useNavigate();

 const handleIconClick = () => {
 setIsFavorite((prev) => !prev);
 };

 return (
 <div
 onClick={()=> navigate(
`/product-details/${product.category?.categoryId}/${product.title}/${product.id}`
)}
 >
 <div
 className='group'
 <div
 className="relative h-[300px]"

```

```

 >
 <img
 className="h-full w-full object-cover"
 src={product.images[0]}
 alt={`product-similar`}
 />

 </div>
 <div className='details pt-3 space-y-1 group-hover-effect rounded-md'>
 <div className='name space-y '>
 <h1 className='font-semibold text-lg'>{product.seller?.businessDetails.businessName}</h1>
 <p className='>{product.title}</p>
 </div>
 <div className='price flex items-center gap-3 '>

 ₹{product.sellingPrice}
 ₹{product.mrpPrice}
 {product.discountPercent}% off
 </div>
 </div>
 </div>
);

};

export default SimilarProductCard;

import SimilarProductCard from './SimilarProductCard'
import { useAppSelector } from '../../../../../Redux Toolkit/Store'

const SmilarProduct = () => {
 const { products } = useAppSelector((store) => store);
 return (
 <div>
 <div className='grid lg:grid-cols-6 md:grid-cols-4 sm:grid-cols-2 grid-cols-1 justify-between gap-4 gap-y-8'>
 {products.products.map((item) => <div
 key = {item.id} className='>

```

```
 <SimilarProductCard product={item} />
 </div>) }

 </div>
 </div>
)

}

export default SmilarProduct
import {
 Button,
 Divider,
 FormControl,
 FormControlLabel,
 FormLabel,
 Radio,
 RadioGroup,
} from "@mui/material";
import React, { useState } from "react";
import { brands } from "../../../../../data/Filter/brand";
import { teal } from "@mui/material/colors";
import { colors } from "../../../../../data/Filter/color";
import { price } from "../../../../../data/Filter/price";
import { discount } from "../../../../../data/Filter/discount";
import { useSearchParams } from "react-router-dom";

const FilterSection = () => {
 const [expendColor, setExpendColor] = useState(false);
 const [expendBrand, setExpendBrand] = useState(false);

 const [searchParams, setSearchParams] = useSearchParams();

 const handleExpendBrand = () => {
 setExpendBrand(!expendBrand);
 };
 const handleExpendColor = () => {
 setExpendColor(!expendColor);
 };

 const updateFilterParams = (e: any) => {
 const { value, name } = e.target;
 if (value) {
 searchParams.set(name, value);
 } else {
 searchParams.delete(name);
 }
 setSearchParams(searchParams);
 };
}
```

```
const clearAllFilters = () => {
 console.log("clearAllFilters", searchParams)
 searchParams.forEach((value: any, key: any) => {
 searchParams.delete(key);
 });
 setSearchParams(searchParams);
};

return (
 <div className="-z-50 space-y-5 bg-white">
 <div className="flex items-center justify-between h-[40px] px-9 lg:border-r">
 <p className="text-lg font-semibold">Filters</p>
 <Button
 onClick={clearAllFilters}
 size="small"
 className="text-teal-600 cursor-pointer font-semibold"
 >
 clear all
 </Button>
 </div>
 <Divider />
 <div className="px-9 space-y-6">
 {/* <section>
 <FormControl>
 <FormLabel
 sx={{
 fontSize: "16px",
 fontWeight: "bold",
 pb: "14px",
 color: teal[600],
 }}
 className="text-2xl font-semibold"
 id="brand"
 >
 Brand
 </FormLabel>
 <RadioGroup
 onChange={updateFilterParams}
 aria-labelledby="brand"
 defaultValue=""
 name="brand"
 >
 {brands
 .slice(0, expendBrand ? brands.length : 5)
 .map((item, index) => (
 <FormControlLabel
 key={item.name}
 value={item.value}
 >
))}
 </RadioGroup>
 </FormControl>
 </section> */}
 </div>
 </div>
)
```

```

 control={<Radio size="small" />}
 label={item.name}
 />
))
</RadioGroup>
</FormControl>
<div>
 <button
 onClick={handleExpendBrand}
 className="text-teal-600 cursor-pointer hover:text-teal-900 flex
items-center"
 >
 {expendBrand ? "hide" : `+ ${brands.length - 5} more`}
 </button>
</div>
</section>
<Divider /> */
<section>
 <FormControl sx={{ zIndex: 0 }}>
 <FormLabel
 sx={{
 fontSize: "16px",
 fontWeight: "bold",
 pb: "14px",
 color: teal[600],
 }}
 className="text-2xl font-semibold"
 id="color"
 >
 Color
 </FormLabel>
 <RadioGroup
 onChange={updateFilterParams}
 aria-labelledby="color"
 defaultValue=""
 name="color"
 >
 {colors
 .slice(0, expendColor ? colors.length : 5)
 .map((item, index) => (
 <FormControlLabel
 sx={{ fontSize: "12px" }}
 key={item.name}
 value={item.name}
 control={<Radio size="small" />}
 label={
 <div className="flex items-center gap-3">
 <p>{item.name}</p>


```

```
 style={{ backgroundColor: item.hex }}
 className={` h-5 w-5 rounded-full ${
 item.name === "White" ? "border" : "border"
 }`}
 >
 </div>
 }
/>
)))
</RadioGroup>
</FormControl>
<div>
 <button
 onClick={handleExpendColor}
 className="text-teal-600 cursor-pointer hover:text-teal-900 flex
items-center"
 >
 {expendColor ? "hide" : `+ ${colors.length - 5} more`}
 </button>
</div>
</section>
<Divider />

<section>
<FormControl>
 <FormLabel
 sx={{
 fontSize: "16px",
 fontWeight: "bold",
 pb: "14px",
 color: teal[600],
 }}
 className="text-2xl font-semibold"
 id="price"
 >
 Price
 </FormLabel>
 <RadioGroup
 name="price"
 onChange={updateFilterParams}
 aria-labelledby="price"
 defaultValue=""
 >
 {price.map((item, index) => (
 <FormControlLabel
 key={item.name}
 value={item.value}
 control=<Radio size="small" />
 label={item.name}
 </FormControlLabel>
))}
 </RadioGroup>
</FormControl>
</section>
```

```
 />
)) }
</RadioGroup>
</FormControl>
</section>
<Divider />
<section>
<FormControl>
<FormLabel
 sx={{{
 fontSize: "16px",
 fontWeight: "bold",
 pb: "14px",
 color: teal[600],
 }}}
 className="text-2xl font-semibold"
 id="brand"
>
 Discount
</FormLabel>
<RadioGroup
 name="discount"
 onChange={updateFilterParams}
 aria-labelledby="brand"
 defaultValue=""
>
 {discount.map((item, index) => (
 <FormControlLabel
 key={item.name}
 value={item.value}
 control={<Radio size="small" />}
 label={item.name}
 />
))}
</RadioGroup>
</FormControl>
</section>
</div>
</div>
);
};

export default FilterSection;
/* eslint-disable @typescript-eslint/no-explicit-any */
import React, { useEffect, useState } from "react";
import ProductCard from "./ProductCard/ProductCard";
import FilterSection from "./FilterSection";
import {
 Box,
```

```

Divider,
FormControl,
IconButton,
InputLabel,
MenuItem,
Pagination,
Select,
useMediaQuery,
useTheme,
type SelectChangeEvent,
} from "@mui/material";

import FilterAltIcon from "@mui/icons-material/FilterAlt";
import { useParams, useSearchParams } from "react-router-dom";
import { useDispatch, useSelector } from "../../Redux Toolkit/Store";
import { getAllProducts } from "../../Redux Toolkit/Customer/ProductSlice";

const Products = () => {
 const [sort, setSort] = React.useState("");
 const theme = useTheme();
 const isLarge = useMediaQuery(theme.breakpoints.up("lg"));
 const [showFilter, setShowFilter] = useState(false);
 const { categoryId } = useParams();
 const dispatch = useDispatch();
 const { products } = useSelector((store) => store);
 const [searchParams] = useSearchParams();
 const [page, setPage] = useState(1)

 const handleSortProduct = (event: SelectChangeEvent) => {
 setSort(event.target.value as string);
 };

 const handleShowFilter = () => {
 setShowFilter((prev) => !prev);
 console.log("showFilter ", showFilter);
 };

 const handlePageChange = (value: any) => {
 setPage(value)
 console.log("page nummmber ", value);
 };

 useEffect(() => {
 const [minPrice, maxPrice] = searchParams.get("price")?.split("-") || [];
 const newFilters = {
 brand: searchParams.get("brand") || "",

```

```

 color: searchParams.get("color") || "",
 minPrice: minPrice ? Number(minPrice) : undefined,
 maxPrice: maxPrice ? Number(maxPrice) : undefined,
 pageNumber:page-1,
 minDiscount: searchParams.get("discount")
 ? Number(searchParams.get("discount"))
 : undefined,
 };

 dispatch(getAllProducts({ category: categoryId, sort, ...newFilters }));
}, [searchParams, categoryId, sort,page]);

// console.log(" store ", products)
return (
 <div className="-z-10 mt-10">
 <div className="">

 <h1 className="text-3xl text-center font-bold text-gray-700 pb-5 px-9 uppercase space-x-2">
 {categoryId?.split("_").map((item) => (
 {item}
)))
 </h1>
 </div>
 <div className="lg:flex">
 <section className="hidden lg:block w-[20%]">
 <FilterSection />
 </section>
 <div className="w-full lg:w-[80%] space-y-5">
 <div className="flex justify-between items-center px-9 h-[40px]">
 <div className="relative w-[50%]">
 {!isLarge && (
 <IconButton onClick={handleShowFilter}>
 <FilterAltIcon />
 </IconButton>
) }
 {showFilter && !isLarge && (
 <Box sx={{ zIndex: 3 }} className="absolute top-[60px]">
 <FilterSection />
 </Box>
) }
 </div>
 <FormControl size="small" sx={{ width: "200px" }}>
 <InputLabel id="sort">Sort</InputLabel>
 <Select
 labelId="sort"
 id="sort"
 value={sort}

```

```

 label="Sort"
 onChange={handleSortProduct}
 >
 <MenuItem value={"price_low"}>Price : Low - High</MenuItem>
 <MenuItem value={"price_high"}>Price : High - Low</MenuItem>
 </Select>
 </FormControl>
</div>
<Divider />

{products.products?.length > 0 ? (
 <section className="grid sm:grid-cols-2 md:grid-cols-3
lg:grid-cols-4 gap-y-5 px-5 justify-center">
 {products.products.map((item: any, index: number) => (
 <div key={index * 9} className="">
 <ProductCard item={item} />
 </div>
)))
 </section>
) : (
 <section className="items-center flex flex-col gap-5 justify-center
h-[67vh] border">

 <h1 className="font-bold text-xl text-center flex items-center
gap-2">
 Product Not Found For{" "}
 <p className="text-primary-color flex gap-2 uppercase">
 {" "}
 {categoryId?.split("_").map((item) => (
 {item}
))}{ " "}
 </p>{ " "}
 </h1>
 </section>
)
<div className="flex justify-center pt-10">
 <Pagination
 page={page}
 onChange={(e, value) => handlePageChange(value)}
 color="primary"
 count={products?.totalPages}
 shape="rounded"
 />
</div>

```

```

 </div>
 </div>
</div>
);

};

export default Products;
import { Backdrop, Button, CircularProgress } from "@mui/material";
import React, { useEffect } from "react";
import store, { useDispatch, useSelector } from "../../../../../Redux Toolkit/Store";
import { paymentSuccess } from "../../../../../Redux Toolkit/Customer/OrderSlice";
import { useLocation, useNavigate } from "react-router-dom";

const PaymentSuccessHandler = () => {
 const dispatch = useDispatch();
 const location = useLocation();
 const { orders } = useSelector(store => store)
 const navigate=useNavigate();

 const getQueryParam = (key: string): string | null => {
 const params = new URLSearchParams(location.search);
 return params.get(key);
 };
 const paymentId = getQueryParam("razorpay_payment_id");
 const paymentLinkId = getQueryParam("razorpay_payment_link_id");
 // const
paymentId="cs_test_a1eU8pFuXZJlg3tiahN153QykvQ16LI5hLgSnUUh01alidIPrMU8KyDx67"

 useEffect(() => {
 if (paymentId) {
 dispatch(
 paymentSuccess({
 paymentId,
 paymentLinkId: paymentLinkId || "",
 jwt: localStorage.getItem("jwt") || "",
 })
);
 }
 }, [paymentId]);

 return (
 <div className="min-h-[90vh] flex justify-center items-center">
 {orders ? <div className="bg-primary-color text-white p-8 w-[90%] lg:w-[25%] border rounded-md h-[40vh] flex flex-col gap-7 items-center justify-center">
 <h1 className="text-3xl font-semibold">Congratulations!</h1>

```

```

 <h1 className="text-2xl font-semibold">Your Order Get
Success</h1>
 <div>
 <Button onClick={()=>navigate("/")} color="secondary"
variant="contained">Shopping More</Button>
 </div>

 </div> : <Backdrop
 sx={{ color: "#fff", zIndex: (theme) => theme.zIndex.drawer + 1
} }
 open={true}
 // onClick={handleClose}
 >
 <CircularProgress color="inherit" />
 </Backdrop>
 </div>
);
};

export default PaymentSuccessHandler;
import React from "react";
import { Avatar, IconButton } from "@mui/material";
import { Rating, Box, Typography, Grid } from "@mui/material";
import type { Review } from "../../../../../types/reviewTypes";
import DeleteIcon from '@mui/icons-material/Delete';
import { red } from "@mui/material/colors";
import { useDispatch, useSelector } from "../../../../../Redux Toolkit/Store";
import { deleteReview } from "../../../../../Redux Toolkit/Customer/ReviewSlice";

interface ProductReviewCardProps {
 item: Review;
}

const ProductReviewCard = ({ item }: ProductReviewCardProps) => {
 const [value, setValue] = React.useState(4.5);
 const { auth, user } = useSelector(store => store);
 const dispatch = useDispatch()
 const handleDeleteReview = () => {
 dispatch(deleteReview({ reviewId: item.id, jwt: localStorage.getItem("jwt")
|| "" }))
 };
 return (
 <div className="flex justify-between">
 <Grid container spacing={2} gap={3}>
 <Grid item xs={1}>
 <Box>
 <Avatar
 className="text-white"

```

```

 sx={{ width: 56, height: 56, bgcolor: "#9155FD" }}}
 alt={item.user.fullName}
 src=""
 >
 {item.user.fullName[0].toUpperCase()}
 </Avatar>
</Box>
</Grid>
<Grid item xs={9}>
 <div className="space-y-2">
 <div className="">
 <p className="font-semibold text-lg">{item.user.fullName}</p>
 <p className="opacity-70">{item.createdAt}</p>
 </div>
 <div>

 <Rating
 readOnly
 value={item.rating}
 name="half-rating"
 defaultValue={2.5}
 precision={0.5}
 />

 </div>
 <p>
 {item.reviewText}
 </p>
 <div>
 {item.productImages.map((image) => <img key={image}
className="w-24 h-24 object-cover" src={image} alt="" />)}
 </div>
 </div>
 </Grid>
</Grid>
{item.user.id === user.user?.id && <div className="">
 <IconButton onClick={handleDeleteReview}>
 <DeleteIcon sx={{ color: red[700] }} />
 </IconButton>
</div>}
</div>
);
};

export default ProductReviewCard;
import { Avatar, Box, Grid, LinearProgress, Rating } from '@mui/material'
import React from 'react'
import type { Review } from '../../../../../types/reviewTypes';

```

```
const RatingCard = ({totalReview}:any) => {
 return (
 <div className="border p-5 rounded-md">

 <div className="flex items-center space-x-3 pb-10">
 <Rating
 name="read-only"
 value={4.6}
 precision={0.5}
 readOnly
 />

 <p className="opacity-60">{totalReview} Ratings</p>
 </div>
 <Box>
 <Grid
 container
 justifyContent="center"
 alignItems="center"
 gap={2}
 >
 <Grid xs={2}>
 <p className="p-0">Excellent</p>
 </Grid>
 <Grid xs={7}>
 <LinearProgress
 className=""
 sx={{ bgcolor: "#d0d0d0", borderRadius: 4, height: 7
}} variant="determinate"
 value={40}
 color="success"
 />
 </Grid>
 <Grid xs={2}>
 <p className="opacity-50 p-2">19259</p>
 </Grid>
 </Grid>
 </Box>
 <Box>
 <Grid
 container
 justifyContent="center"
 alignItems="center"
 gap={2}
 >
```

```
>
 <Grid xs={2}>
 <p className="p-0">Very Good</p>
 </Grid>
 <Grid xs={7}>
 <LinearProgress
 className=""
 sx={{ bgcolor: "#d0d0d0", borderRadius: 4, height: 7
} }
 variant="determinate"
 value={30}
 color="success"
 />
 </Grid>
 <Grid xs={2}>
 <p className="opacity-50 p-2">19259</p>
 </Grid>
</Grid>
</Box>
<Box>
 <Grid
 container
 justifyContent="center"
 alignItems="center"
 gap={2}
 >
 <Grid xs={2}>
 <p className="p-0">Good</p>
 </Grid>
 <Grid xs={7}>
 <LinearProgress
 className="bg-[#885c0a]"
 sx={{ bgcolor: "#d0d0d0", borderRadius: 4, height: 7
} }
 variant="determinate"
 value={25}

 />
 </Grid>
 <Grid xs={2}>
 <p className="opacity-50 p-2">19259</p>
 </Grid>
 </Grid>
</Box>
<Box>
 <Grid
 container
 justifyContent="center"
 alignItems="center"

```

```
 gap={2}
 >
 <Grid xs={2}>
 <p className="p-0">Avarage</p>
 </Grid>
 <Grid xs={7}>
 <LinearProgress
 className=""
 sx={{
 bgcolor: "#d0d0d0",
 borderRadius: 4,
 height: 7,
 "& .MuiLinearProgress-bar": {
 bgcolor: "#885c0a", // stroke color
 },
 }}
 variant="determinate"
 value={21}
 color="success"
 />
 </Grid>
 <Grid xs={2}>
 <p className="opacity-50 p-2">19259</p>
 </Grid>
</Grid>
</Box>
<Box>
 <Grid
 container
 justifyContent="center"
 alignItems="center"
 gap={2}
 >
 <Grid xs={2}>
 <p className="p-0">Poor</p>
 </Grid>
 <Grid xs={7}>
 <LinearProgress
 className=""
 sx={{ bgcolor: "#d0d0d0", borderRadius: 4, height: 7
 }}
 variant="determinate"
 value={10}
 color="error"
 />
 </Grid>
 <Grid xs={2}>
 <p className="opacity-50 p-2">19259</p>
 </Grid>

```

```

 </Grid>
 </Box>

 </div>
)
}

export default RatingCard
import React, { useState } from 'react';
import { useFormik } from 'formik';
import * as Yup from 'yup';
import {
 TextField,
 Button,
 Box,
 Rating,
 InputLabel,
 Typography,
 IconButton,
 CircularProgress,
} from '@mui/material';
import CloseIcon from "@mui/icons-material/Close";
import { uploadToCloudinary } from '../../../../../util/uploadToCloudinary';
import AddPhotoAlternateIcon from "@mui/icons-material/AddPhotoAlternate";
import { useDispatch } from '../../../../../Redux Toolkit/Store';
import { createReview } from '../../../../../Redux Toolkit/Customer/ReviewSlice';
import { useNavigate, useParams } from 'react-router-dom';

interface CreateReviewRequest {
 reviewText: string;
 reviewRating: number;
 productImages: string[];
}

const ReviewForm: React.FC = () => {
 const [uploadImage, setUploadingImage] = useState(false);
 const dispatch = useDispatch();
 const { productId } = useParams();
 const navigate=useNavigate();

 const formik = useFormik<CreateReviewRequest>({
 initialValues: {
 reviewText: '',
 reviewRating: 0,
 productImages: [], // Initializing with an empty string array
 },
 validationSchema: Yup.object({
 reviewText: Yup.string()

```

```

 .required('Review text is required')
 .min(10, 'Review must be at least 10 characters long'),
 reviewRating: Yup.number()
 .required('Rating is required')
 .min(0, 'Rating must be at least 0')
 .max(5, 'Rating cannot be more than 5'),

)),
 onSubmit: (values) => {
 if (productId) {
 dispatch(createReview({
 productId: Number(productId),
 review: values,
 jwt: localStorage.getItem("jwt") || "",
 navigate
 }))
 console.log('Form Submitted:', values);
 }
 },
);
}

const handleImageChange = async (event: any) => {
 const file = event.target.files[0];
 setUploadingImage(true);
 const image = await uploadToCloudinary(file);
 // const image = URL.createObjectURL(file);
 formik.setFieldValue("productImages", [...formik.values.productImages,
image]);
 setUploadingImage(false);
};

const handleRemoveImage = (index: number) => {
 const updatedImages = [...formik.values.productImages];
 updatedImages.splice(index, 1);
 formik.setFieldValue("images", updatedImages);
};

return (
 <Box
 component="form"
 onSubmit={formik.handleSubmit}
 noValidate
 sx={{ mt: 3 }}
 className='space-y-5'
 >

 <TextField
 fullWidth
 id="reviewText"
 name="reviewText"

```

```
 label="Review Text"
 variant="outlined"
 multiline
 rows={4}
 value={formik.values.reviewText}
 onChange={formik.handleChange}
 onBlur={formik.handleBlur}
 error={formik.touched.reviewText &&
Boolean(formik.errors.reviewText)}
 helperText={formik.touched.reviewText &&
formik.errors.reviewText}

 />

 <div className='space-y-2'>
 <InputLabel>Rating</InputLabel>
 <Rating
 id="reviewRating"
 name="reviewRating"
 value={formik.values.reviewRating}
 onChange={(event, newValue) =>
 formik.setFieldValue('reviewRating', newValue)
 }
 onBlur={formik.handleBlur}
 precision={0.5}
 />
 </div>
 {formik.touched.reviewRating && formik.errors.reviewRating && (
 <Typography color="error" variant="body2">
 {formik.errors.reviewRating}
 </Typography>
)}

 <div className="flex flex-wrap gap-5 py-3">
 <input
 type="file"
 accept="image/*"
 id="fileInput"
 style={{ display: "none" }}
 onChange={handleImageChange}
 />

 <label className="relative" htmlFor="fileInput">
 <span className="w-24 h-24 cursor-pointer flex items-center
justify-center p-3 border rounded-md border-gray-400">
 <AddPhotoAlternateIcon className="text-gray-700" />

 {uploadImage && (

```

```

 <div className="absolute left-0 right-0 top-0 bottom-0
w-24 h-24 flex justify-center items-center">
 <CircularProgress />
 </div>
) }
</label>

<div className="flex flex-wrap gap-2">
 {formik.values.productImages.map((image, index) => (
 <div className="relative">
 <img
 className="w-24 h-24 object-cover"
 key={index}
 src={image}
 alt={`ProductImage ${index + 1}`}
 />
 <IconButton
 onClick={() => handleRemoveImage(index)}
 className=""
 size="small"
 color="error"
 sx={{
 position: "absolute",
 top: 0,
 right: 0,
 outline: "none",
 }}
 >
 <CloseIcon sx={{ fontSize: "1rem" }} />
 </IconButton>
 </div>
)))
 </div>
</div>

<Button color="primary" variant="contained" type="submit">
 Submit Review
</Button>
</Box>
);
};

export default ReviewForm;
import React, { useEffect } from 'react'
import { useDispatch, useSelector } from '../../../../../Redux Toolkit/Store';
import { useParams } from 'react-router-dom';
import { fetchProductById } from '../../../../../Redux
Toolkit/Customer/ProductSlice';
import { Divider, Rating } from '@mui/material';

```

```
import ProductReviewCard from './ProductReviewCard';
import { fetchReviewsByProductId } from '../../../../../Redux
Toolkit/Customer/ReviewSlice';
import RatingCard from './RatingCard';

const Reviews = () => {
 const dispatch = useAppDispatch();
 const { products, review } = useAppSelector(store => store)

 const { productId } = useParams()

 useEffect(() => {

 if (productId) {
 dispatch(fetchProductById(Number(productId)))
 dispatch(fetchReviewsByProductId({ productId: Number(productId) }))
 }
 }, [productId])

 return (
 <div className='p-5 lg:p-20 flex flex-col lg:flex-row gap-20'>
 <section className='w-full md:w-1/2 lg:w-[30%] space-y-2'>
 <img className='w-full' src={products.product?.images[0]}
 alt="" />
 <div>
 <div>
 <p className='font-bold text-xl'>
 {products.product?.seller?.businessDetails.businessName}
 </p>
 <p className='text-lg text-gray-600'>{products.product?.title}</p>
 </div>

 <div className='price flex items-center gap-3 mt-5 text-lg'>

 ₹{products.product?.sellingPrice}
 ₹{products.product?.mrpPrice}
 {products.product?.discountPercent}% off
 </div>
 </div>
 </section>
 <section className="w-full md:w-1/2 lg:w-[70%]">
 <h1 className="font-semibold text-lg pb-4">
```

```

 Review & Ratings
 </h1>

 <RatingCard/>
 <div className='mt-10'>
 <div className="space-y-5">
 {review.reviews.map((item, i) => (
 <div className='space-y-5'>
 <ProductReviewCard item={item} />
 {review.reviews.length - 1 !== i && <Divider />}
 </div>
))}
 </div>
 </div>

 </section>
</div>
)
}

export default Reviews
import React, { useEffect } from 'react'
import { useDispatch, useSelector } from '../../../../../Redux Toolkit/Store';
import { useParams } from 'react-router-dom';
import { fetchProductById } from '../../../../../Redux
Toolkit/Customer/ProductSlice';
import { Box, Grid, LinearProgress, Rating } from '@mui/material';
import ProductReviewCard from './ProductReviewCard';
import ReviewForm from './ReviewForm';

const WriteReviews = () => {
 const dispatch = useDispatch();
 const { products, review } = useSelector(store => store)

 const { productId } = useParams()

 useEffect(() => {

 if (productId) {
 dispatch(fetchProductById(Number(productId)))
 }
 }, [productId])

 return (
 <div className='p-5 lg:p-20 flex flex-col lg:flex-row gap-10'>
 <div className='w-full md:w-1/2 lg:w-[30%] space-y-2'>

```

```

 <img className='w-full' src={
 products.product?.images[0]
 } alt="" />
 <div>
 <div>
 <p className='font-bold text-xl'>
{products.product?.seller?.businessDetails.businessName}

 </p>
 <p className='text-lg
text-gray-600'>{products.product?.title}</p>
 </div>

 <div className='price flex items-center gap-3 mt-5 text-lg'>

₹{products.product?.sellingPrice}
 <span className='text thin-line-through text-gray-400
'>₹{products.product?.mrpPrice}
 <span className='text-[#00927c]
font-semibold'>{products.product?.discountPercent}% off
 </div>
 </div>
 </div>
 <section className="w-full md:w-1/2 lg:w-[70%]">
 <h1 className="font-semibold text-2xl pb-4 text-gray-700">
 Write Your Review & Give Ratings
 </h1>
 <ReviewForm />
 </section>
</div>
)
}

export default WriteReviews
import React, { ChangeEvent, useState } from 'react'
import { searchProduct } from '../../../../../Redux Toolkit/Customer/ProductSlice';
import { useAppDispatch, useAppSelector } from '../../../../../Redux Toolkit/Store';
import ProductCard from '../Products/ProductCard/ProductCard';

const SearchProducts = () => {
 const [searchQuery, setSearchQuery] = useState("");
 const dispatch = useAppDispatch();
 const { products } = useAppSelector(store => store)

 const handleSearchChange = (e: ChangeEvent<HTMLInputElement>) => {
 setSearchQuery(e.target.value);
 }
}

```

```

const handleProductSearch = () => {
 dispatch(searchProduct(searchQuery))
}
return (
 <div className='min-h-screen px-20'>
 <div className="flex justify-center py-5">
 <input
 onKeyPress={ (e) => {
 if (e.key === 'Enter') {
 handleProductSearch();
 console.log("Searching for ", searchQuery);
 }
 } }
 onChange={handleSearchChange}
 className="border-none outline-none bg-slate-100 px-10 py-3 w-full
lg:w-1/2" type="text" placeholder="Search Product..." />
 </div>
 <section>
 {products.searchProduct?.length > 0 ? (
 <section className="grid sm:grid-cols-2 md:grid-cols-3
lg:grid-cols-4 gap-y-5 px-5 justify-center">
 {products.searchProduct.map((item: any, index: number) => (
 <div key={index * 9} className="">
 <ProductCard item={item} />
 </div>
)));
 </section>
) :
 // searchQuery ? (
 // <section className="items-center flex flex-col gap-5 justify-center
h-[67vh] border">
 //
 // <h1 className="font-bold text-xl text-center flex items-center
gap-2">
 // Product Not Found For
 // <p className="text-primary-color flex gap-2 uppercase">
 // {searchQuery}
 // </p>
 // </h1>
 // </section>
 //) :
 <div className='h-[70vh] flex flex-col justify-center items-center'>
 <h1 className='font-bold text-6xl'>

```

```

 Search Product Here
 </h1></div>}
</section>
</div>
)
}

export default SearchProducts
import React, { useEffect } from 'react'
import { useDispatch, useSelector } from '../../../../../Redux Toolkit/Store'
import { getWishlistByUserId } from '../../../../../Redux
Toolkit/Customer/WishlistSlice';
import WishlistProductCard from './WishlistProductCard';

const Wishlist = () => {
 const dispatch = useDispatch();
 const { wishlist } = useSelector(store => store

 console.log("wishlist", wishlist)
 return (
 <div className='h-[85vh] p-5 lg:p-20'>
 {wishlist.wishlist?.products.length ?
 <section>
 <h1>My Wishlist
{wishlist.wishlist.products.length} items</h1>
 <div className='pt-10 flex flex-wrap gap-5'>

 {wishlist.wishlist?.products?.map((item) =>
<WishlistProductCard item={item} />)}

 </div>
 </section> :
 <div className="h-full flex justify-center items-center
flex-col">
 <div className="text-center py-5">
 <h1 className="text-lg font-medium">hay its feels so
light!</h1>
 <p className="text-gray-500 text-sm">
 there is nothing in your wishlist, lets add some
items
 </p>
 </div>

 </div>
)
 }

```

```
export default Wishlist

import React, { useState, useEffect, MouseEvent } from 'react';
import FavoriteIcon from '@mui/icons-material/Favorite';
import { teal } from '@mui/material/colors';
import { Button, IconButton } from '@mui/material';
import { useNavigate, useParams } from 'react-router-dom';
import type { Product } from '../../../../../types/productTypes';
import { useDispatch } from '../../../../../Redux Toolkit/Store';
import CloseIcon from '@mui/icons-material/Close';
import CancelIcon from '@mui/icons-material/Cancel';
import { addProductToWishlist } from '../../../../../Redux Toolkit/Customer/WishlistSlice';

interface ProductCardProps {
 item: Product;
}

const WishlistProductCard: React.FC<ProductCardProps> = ({ item }) => {
 const [currentImage, setCurrentImage] = useState(0);
 const [isHovered, setIsHovered] = useState(false);
 const [isFavorite, setIsFavorite] = useState(false);
 const navigate = useNavigate();
 const dispatch = useDispatch();

 const handleIconClick = (e:MouseEvent) => {
 setIsFavorite((prev) => !prev);
 if(item.id)
 dispatch(addProductToWishlist({productId:item.id}))
 };

 return (
 <div className='w-60 relative '>
 <div
 className="w-full"
 >

 <img
 className=" object-top w-full"
 src={item.images[0]}
 alt={`product-${item.title}`}
 />

 </div>
 <div className='pt-3 space-y-1 rounded-md '>
```

```
<div className=' space-y '>

 <p className=''{>{item.title}</p>

</div>
<div className=' flex items-center gap-3 '>

₹{item.sellingPrice}
 <span className='text thin-line-through text-gray-400
'>₹{item.mrpPrice}
 <span className='text-[#00927c]
font-semibold'>{item.discountPercent}% off
</div>

</div>

<div className='absolute top-1 right-1'>

 <button
 onClick={handleIconClick}
 >
<CloseIcon className='cursor-pointer bg-white rounded-full p-1' sx={{ color:
teal[500], fontSize:"2rem" }} />
 </button>

</div>
);
};

};

export default WishlistProductCard
export function formatDate(date:any) {
 // Define options for formatting
 date = new Date(date);
 const options = {
 weekday: 'short', // Sun
 month: 'short' ,
 day: '2-digit',
 };

 // Format the date
 return date.toLocaleDateString('en-US', options);
}
```

```
export const electronicsLevelThree = [
 {
 parentCategoryId: "mobiles",
 categoryId: "mi_mobile",
 level: 3,
 name: "Mi Mobile",
 parentCategoryName: "Mobiles",
 },
 {
 parentCategoryId: "mobiles",
 categoryId: "realme_mobile",
 level: 3,
 name: "Realme Mobile",
 parentCategoryName: "Mobiles",
 },
 {
 parentCategoryId: "mobiles",
 categoryId: "samsung_mobile",
 level: 3,
 name: "Samsung Mobile",
 parentCategoryName: "Mobiles",
 },
 {
 parentCategoryId: "mobiles",
 categoryId: "infinix_mobile",
 level: 3,
 name: "Infinix Mobile",
 parentCategoryName: "Mobiles",
 },
 {
 parentCategoryId: "mobiles",
 categoryId: "oppo_mobile",
 level: 3,
 name: "OPPO Mobile",
 parentCategoryName: "Mobiles",
 },
 {
 parentCategoryId: "mobiles",
 categoryId: "apple_mobile",
 level: 3,
 name: "Apple Mobile",
 parentCategoryName: "Mobiles",
 },
 {
 parentCategoryId: "mobiles",
 categoryId: "vivo_mobile",
 level: 3,
 name: "Vivo Mobile",
 parentCategoryName: "Mobiles",
 },
]
```

```
},
{
 parentCategoryId: "mobiles",
 categoryId: "honor_mobile",
 level: 3,
 name: "Honor Mobile",
 parentCategoryName: "Mobiles",
},
{
 parentCategoryId: "mobiles",
 categoryId: "asus_mobile",
 level: 3,
 name: "Asus Mobile",
 parentCategoryName: "Mobiles",
},
{
 parentCategoryId: "mobiles",
 categoryId: "poco_x2_mobile",
 level: 3,
 name: "Poco X2 Mobile",
 parentCategoryName: "Mobiles",
},
{
 parentCategoryId: "mobiles",
 categoryId: "realme_narzo_10_mobile",
 level: 3,
 name: "realme Narzo 10 Mobile",
 parentCategoryName: "Mobiles",
},
{
 parentCategoryId: "mobiles",
 categoryId: "infinix_hot_9_mobile",
 level: 3,
 name: "Infinix Hot 9 Mobile",
 parentCategoryName: "Mobiles",
},
{
 parentCategoryId: "mobiles",
 categoryId: "iqoo_3_mobile",
 level: 3,
 name: "IQOO 3 Mobile",
 parentCategoryName: "Mobiles",
},
{
 parentCategoryId: "mobiles",
 categoryId: "iphone_se_mobile",
 level: 3,
 name: "iPhone SE Mobile",
 parentCategoryName: "Mobiles",
```

```
},
{
 parentCategoryId: "mobiles",
 categoryId: "motorola_razr_mobile",
 level: 3,
 name: "Motorola razr Mobile",
 parentCategoryName: "Mobiles",
},
{
 parentCategoryId: "mobiles",
 categoryId: "realme_narzo_10a_mobile",
 level: 3,
 name: "realme Narzo 10A Mobile",
 parentCategoryName: "Mobiles",
},
{
 parentCategoryId: "mobiles",
 categoryId: "motorola_g8_power_lite_mobile",
 level: 3,
 name: "Motorola g8 power lite Mobile",
 parentCategoryName: "Mobiles",
},
{
 parentCategoryId: "mobile_accessories",
 categoryId: "mobile_cases",
 level: 3,
 name: "Mobile Cases",
 parentCategoryName: "Mobile Accessories",
},
{
 parentCategoryId: "mobile_accessories",
 categoryId: "headphones_headsets",
 level: 3,
 name: "Headphones & Headsets",
 parentCategoryName: "Mobile Accessories",
},
{
 parentCategoryId: "mobile_accessories",
 categoryId: "power_banks",
 level: 3,
 name: "Power Banks",
 parentCategoryName: "Mobile Accessories",
},
{
 parentCategoryId: "mobile_accessories",
 categoryId: "screenguards",
 level: 3,
 name: "Screenguards",
```

```
 parentCategoryName: "Mobile Accessories",
 },
{
 parentCategoryId: "mobile_accessories",
 categoryId: "memory_cards",
 level: 3,
 name: "Memory Cards",
 parentCategoryName: "Mobile Accessories",
},
{
 parentCategoryId: "mobile_accessories",
 categoryId: "smart_headphones",
 level: 3,
 name: "Smart Headphones",
 parentCategoryName: "Mobile Accessories",
},
{
 parentCategoryId: "mobile_accessories",
 categoryId: "mobile_cables",
 level: 3,
 name: "Mobile Cables",
 parentCategoryName: "Mobile Accessories",
},
{
 parentCategoryId: "mobile_accessories",
 categoryId: "mobile_chargers",
 level: 3,
 name: "Mobile Chargers",
 parentCategoryName: "Mobile Accessories",
},
{
 parentCategoryId: "mobile_accessories",
 categoryId: "mobile_holders",
 level: 3,
 name: "Mobile Holders",
 parentCategoryName: "Mobile Accessories",
},
{
 parentCategoryId: "smart_wearable_tech",
 categoryId: "smart_watches",
 level: 3,
 name: "Smart Watches",
 parentCategoryName: "Smart Wearable Tech",
},
{
 parentCategoryId: "smart_wearable_tech",
 categoryId: "smart_glasses_vr",
 level: 3,
```

```
 name: "Smart Glasses (VR)",
 parentCategoryName: "Smart Wearable Tech",
 },
{
 parentCategoryId: "smart_wearable_tech",
 categoryId: "smart_bands",
 level: 3,
 name: "Smart Bands",
 parentCategoryName: "Smart Wearable Tech",
},
{
 parentCategoryId: "health_care_appliances",
 categoryId: "bp_monitors",
 level: 3,
 name: "Bp Monitors",
 parentCategoryName: "Health Care Appliances",
},
{
 parentCategoryId: "health_care_appliances",
 categoryId: "weighing_scale",
 level: 3,
 name: "Weighing Scale",
 parentCategoryName: "Health Care Appliances",
},
{
 parentCategoryId: "laptops",
 categoryId: "gaming_laptops",
 level: 3,
 name: "Gaming Laptops",
 parentCategoryName: "Laptops",
},
{
 parentCategoryId: "laptops",
 categoryId: "desktop_pcs",
 level: 3,
 name: "Desktop PCs",
 parentCategoryName: "Laptops",
},
{
 parentCategoryId: "laptops",
 categoryId: "gaming_accessories",
 level: 3,
 name: "Gaming & Accessories",
 parentCategoryName: "Laptops",
},
{
 parentCategoryId: "laptops",
```

```
 categoryId: "computer_accessories",
 level: 3,
 name: "Computer Accessories",
 parentCategoryName: "Laptops",
 },
{
 parentCategoryId: "laptops",
 categoryId: "external_hard_disks",
 level: 3,
 name: "External Hard Disks",
 parentCategoryName: "Laptops",
},
{
 parentCategoryId: "laptops",
 categoryId: "pendrives",
 level: 3,
 name: "Pendrives",
 parentCategoryName: "Laptops",
},
{
 parentCategoryId: "laptops",
 categoryId: "laptop_skins_decals",
 level: 3,
 name: "Laptop Skins & Decals",
 parentCategoryName: "Laptops",
},
{
 parentCategoryId: "laptops",
 categoryId: "laptop_bags",
 level: 3,
 name: "Laptop Bags",
 parentCategoryName: "Laptops",
},
{
 parentCategoryId: "laptops",
 categoryId: "mouse",
 level: 3,
 name: "Mouse",
 parentCategoryName: "Laptops",
},
{
 parentCategoryId: "laptops",
 categoryId: "computer_peripherals",
 level: 3,
 name: "Computer Peripherals",
 parentCategoryName: "Laptops",
},
{
 parentCategoryId: "laptops",
```

```
 categoryId: "printers_ink_cartridges",
 level: 3,
 name: "Printers & Ink Cartridges",
 parentCategoryName: "Laptops",
 },
{
 parentCategoryId: "laptops",
 categoryId: "monitors",
 level: 3,
 name: "Monitors",
 parentCategoryName: "Laptops",
},
{
 parentCategoryId: "tablets",
 categoryId: "apple_ipads",
 level: 3,
 name: "Apple iPads",
 parentCategoryName: "Tablets",
},
{
 parentCategoryId: "speakers",
 categoryId: "home_audio_speakers",
 level: 3,
 name: "Home Audio Speakers",
 parentCategoryName: "Speakers",
},
{
 parentCategoryId: "speakers",
 categoryId: "home_theatres",
 level: 3,
 name: "Home Theatres",
 parentCategoryName: "Speakers",
},
{
 parentCategoryId: "speakers",
 categoryId: "soundbars",
 level: 3,
 name: "Soundbars",
 parentCategoryName: "Speakers",
},
{
 parentCategoryId: "speakers",
 categoryId: "bluetooth_speakers",
 level: 3,
 name: "Bluetooth Speakers",
 parentCategoryName: "Speakers",
},
```

```
{
 parentCategoryId: "speakers",
 categoryId: "dth_set_top_box",
 level: 3,
 name: "DTH Set Top Box",
 parentCategoryName: "Speakers",
,

{
 parentCategoryId: "smart_home_automation",
 categoryId: "google_nest",
 level: 3,
 name: "Google Nest",
 parentCategoryName: "Smart Home Automation",
,

{
 parentCategoryId: "camera",
 categoryId: "dslr_mirrorless_camera",
 level: 3,
 name: "DSLR & Mirrorless Camera",
 parentCategoryName: "Camera",
,
{
 parentCategoryId: "camera",
 categoryId: "compact_bridge_cameras",
 level: 3,
 name: "Compact & Bridge Cameras",
 parentCategoryName: "Camera",
,
{
 parentCategoryId: "camera",
 categoryId: "sports_action_camera",
 level: 3,
 name: "Sports & Action Camera",
 parentCategoryName: "Camera",
,
{
 parentCategoryId: "camera",
 categoryId: "camera_accessories",
 level: 3,
 name: "Camera Accessories",
 parentCategoryName: "Camera",
,
{
 parentCategoryId: "camera",
 categoryId: "lens",
 level: 3,
 name: "Camera Lens",
```

```
 parentCategoryName: "Camera",
 },
{
 parentCategoryId: "camera",
 categoryId: "camera_tripods",
 level: 3,
 name: "Camera Tripods",
 parentCategoryName: "Camera",
},
{
 parentCategoryId: "network_components",
 categoryId: "routers",
 level: 3,
 name: "Routers",
 parentCategoryName: "Network Components",
},
{
 parentCategoryId: "featured",
 categoryId: "google_assistant_store",
 level: 3,
 name: "Google Assistant Store",
 parentCategoryName: "Featured",
},
{
 parentCategoryId: "featured",
 categoryId: "laptops_on_buyback_guarantee",
 level: 3,
 name: "Laptops on Buyback Guarantee",
 parentCategoryName: "Featured",
},
{
 parentCategoryId: "featured",
 categoryId: "flipkart_smartbuy",
 level: 3,
 name: "Flipkart SmartBuy",
 parentCategoryName: "Featured",
},
{
 parentCategoryId: "featured",
 categoryId: "li_polymer_power_banks",
 level: 3,
 name: "Li-Polymer Power Banks",
 parentCategoryName: "Featured",
},
{
 parentCategoryId: "featured",
 categoryId: "sony_ps4_pro_slim",
```

```
 level: 3,
 name: "Sony PS4 Pro & Slim",
 parentCategoryName: "Featured",
},
{
 parentCategoryId: "featured",
 categoryId: "apple_products",
 level: 3,
 name: "Apple Products",
 parentCategoryName: "Featured",
},
{
 parentCategoryId: "featured",
 categoryId: "microsoft_store",
 level: 3,
 name: "Microsoft Store",
 parentCategoryName: "Featured",
},
{
 parentCategoryId: "featured",
 categoryId: "lenovo_phab_series",
 level: 3,
 name: "Lenovo Phab Series",
 parentCategoryName: "Featured",
},
{
 parentCategoryId: "featured",
 categoryId: "jbl_speakers",
 level: 3,
 name: "JBL Speakers",
 parentCategoryName: "Featured",
},
{
 parentCategoryId: "featured",
 categoryId: "smartphones_on_buyback_guarantee",
 level: 3,
 name: "Smartphones On Buyback Guarantee",
 parentCategoryName: "Featured",
},
{
 parentCategoryId: "featured",
 categoryId: "philips",
 level: 3,
 name: "Philips",
 parentCategoryName: "Featured",
},
{
 parentCategoryId: "featured",
 categoryId: "dr_morepen",
```

```
 "level": 3,
 "name": "Dr. Morepen",
 "parentCategoryName": "Featured",
 },
];
export const furnitureLevelThree = [
{
 "name": "Bed Runners",
 "categoryId": "bed_runners",
 "parentCategoryName": "Bed Linen & Furnishing",
 "parentCategoryId": "bed_linen_furnishing",
 "level": 3
},
{
 "name": "Mattress Protectors",
 "categoryId": "mattress_protectors",
 "parentCategoryName": "Bed Linen & Furnishing",
 "parentCategoryId": "bed_linen_furnishing",
 "level": 3
},
{
 "name": "Bedsheets",
 "categoryId": "bedsheets",
 "parentCategoryName": "Bed Linen & Furnishing",
 "parentCategoryId": "bed_linen_furnishing",
 "level": 3
},
{
 "name": "Bedding Sets",
 "categoryId": "bedding_sets",
 "parentCategoryName": "Bed Linen & Furnishing",
 "parentCategoryId": "bed_linen_furnishing",
 "level": 3
},
{
 "name": "Blankets, Quilts & Dohars",
 "categoryId": "blankets_quilts_dohars",
 "parentCategoryName": "Bed Linen & Furnishing",
 "parentCategoryId": "bed_linen_furnishing",
 "level": 3
},
{
 "name": "Pillows & Pillow Covers",
 "categoryId": "pillows_pillow_covers",
 "parentCategoryName": "Bed Linen & Furnishing",
 "parentCategoryId": "bed_linen_furnishing",
 "level": 3
},
{
```

```
 "name": "Bed Covers",
 "categoryId": "bed_covers",
 "parentCategoryName": "Bed Linen & Furnishing",
 "parentCategoryId": "bed_linen_furnishing",
 "level": 3
 },
 {
 "name": "Diwan Sets",
 "categoryId": "diwan_sets",
 "parentCategoryName": "Bed Linen & Furnishing",
 "parentCategoryId": "bed_linen_furnishing",
 "level": 3
 },
 {
 "name": "Chair Pads & Covers",
 "categoryId": "chair_pads_covers",
 "parentCategoryName": "Bed Linen & Furnishing",
 "parentCategoryId": "bed_linen_furnishing",
 "level": 3
 },
 {
 "name": "Sofa Covers",
 "categoryId": "sofa_covers",
 "parentCategoryName": "Bed Linen & Furnishing",
 "parentCategoryId": "bed_linen_furnishing",
 "level": 3
 },
 {
 "name": "Floor Runners",
 "categoryId": "floor_runners",
 "parentCategoryName": "Flooring",
 "parentCategoryId": "flooring",
 "level": 3
 },
 {
 "name": "Carpets",
 "categoryId": "carpets",
 "parentCategoryName": "Flooring",
 "parentCategoryId": "flooring",
 "level": 3
 },
 {
 "name": "Floor Mats & Dhurries",
 "categoryId": "floor_mats_dhurries",
 "parentCategoryName": "Flooring",
 "parentCategoryId": "flooring",
 "level": 3
 },
 {

```

```
 "name": "Door Mats",
 "categoryId": "door_mats",
 "parentCategoryName": "Flooring",
 "parentCategoryId": "flooring",
 "level": 3
 },
{
 "name": "Bath Towels",
 "categoryId": "bath_towels",
 "parentCategoryName": "Bath",
 "parentCategoryId": "bath",
 "level": 3
},
{
 "name": "Hand & Face Towels",
 "categoryId": "hand_face_towels",
 "parentCategoryName": "Bath",
 "parentCategoryId": "bath",
 "level": 3
},
{
 "name": "Beach Towels",
 "categoryId": "beach_towels",
 "parentCategoryName": "Bath",
 "parentCategoryId": "bath",
 "level": 3
},
{
 "name": "Towels Set",
 "categoryId": "towels_set",
 "parentCategoryName": "Bath",
 "parentCategoryId": "bath",
 "level": 3
},
{
 "name": "Bath Rugs",
 "categoryId": "bath_rugs",
 "parentCategoryName": "Bath",
 "parentCategoryId": "bath",
 "level": 3
},
{
 "name": "Bath Robes",
 "categoryId": "bath_robies",
 "parentCategoryName": "Bath",
 "parentCategoryId": "bath",
 "level": 3
},
{
```

```
 "name": "Bathroom Accessories",
 "categoryId": "bathroom_accessories",
 "parentCategoryName": "Bath",
 "parentCategoryId": "bath",
 "level": 3
 },
{
 "name": "Shower Curtains",
 "categoryId": "shower_curtains",
 "parentCategoryName": "Bath",
 "parentCategoryId": "bath",
 "level": 3
},
{
 "name": "Floor Lamps",
 "categoryId": "floor_lamps",
 "parentCategoryName": "Lamps & Lighting",
 "parentCategoryId": "lamps_lighting",
 "level": 3
},
{
 "name": "Ceiling Lamps",
 "categoryId": "ceiling_lamps",
 "parentCategoryName": "Lamps & Lighting",
 "parentCategoryId": "lamps_lighting",
 "level": 3
},
{
 "name": "Table Lamps",
 "categoryId": "table_lamps",
 "parentCategoryName": "Lamps & Lighting",
 "parentCategoryId": "lamps_lighting",
 "level": 3
},
{
 "name": "Wall Lamps",
 "categoryId": "wall_lamps",
 "parentCategoryName": "Lamps & Lighting",
 "parentCategoryId": "lamps_lighting",
 "level": 3
},
{
 "name": "Outdoor Lamps",
 "categoryId": "outdoor_lamps",
 "parentCategoryName": "Lamps & Lighting",
 "parentCategoryId": "lamps_lighting",
 "level": 3
},
{
```

```
 "name": "String Lights",
 "categoryId": "string_lights",
 "parentCategoryName": "Lamps & Lighting",
 "parentCategoryId": "lamps_lighting",
 "level": 3
 },
{
 "name": "Plants & Planters",
 "categoryId": "plants_planters",
 "parentCategoryName": "Home Décor",
 "parentCategoryId": "home_decor",
 "level": 3
},
{
 "name": "Aromas & Candles",
 "categoryId": "aromas_candles",
 "parentCategoryName": "Home Décor",
 "parentCategoryId": "home_decor",
 "level": 3
},
{
 "name": "Clocks",
 "categoryId": "clocks",
 "parentCategoryName": "Home Décor",
 "parentCategoryId": "home_decor",
 "level": 3
},
{
 "name": "Mirrors",
 "categoryId": "mirrors",
 "parentCategoryName": "Home Décor",
 "parentCategoryId": "home_decor",
 "level": 3
},
{
 "name": "Wall Décor",
 "categoryId": "wall_decor",
 "parentCategoryName": "Home Décor",
 "parentCategoryId": "home_decor",
 "level": 3
},
{
 "name": "Festive Decor",
 "categoryId": "festive_decor",
 "parentCategoryName": "Home Décor",
 "parentCategoryId": "home_decor",
 "level": 3
},
{
```

```
"name": "Pooja Essentials",
"categoryId": "pooja_essentials",
"parentCategoryName": "Home Décor",
"parentCategoryId": "home_decor",
"level": 3
},
{
 "name": "Wall Shelves",
 "categoryId": "wall_shelves",
 "parentCategoryName": "Home Décor",
 "parentCategoryId": "home_decor",
 "level": 3
},
{
 "name": "Fountains",
 "categoryId": "fountains",
 "parentCategoryName": "Home Décor",
 "parentCategoryId": "home_decor",
 "level": 3
},
{
 "name": "Showpieces & Vases",
 "categoryId": "showpieces_vases",
 "parentCategoryName": "Home Décor",
 "parentCategoryId": "home_decor",
 "level": 3
},
{
 "name": "Ottoman",
 "categoryId": "ottoman",
 "parentCategoryName": "Home Décor",
 "parentCategoryId": "home_decor",
 "level": 3
},
{
 "name": "Cushions & Cushion Covers",
 "categoryId": "cushions_cushion_covers",
 "parentCategoryName": "Home Décor",
 "parentCategoryId": "home_decor",
 "level": 3
},
{
 "name": "Curtains",
 "categoryId": "curtains",
 "parentCategoryName": "Home Décor",
 "parentCategoryId": "home_decor",
 "level": 3
},
```

```
 "name": "Home Gift Sets",
 "categoryId": "home_gift_sets",
 "parentCategoryName": "Home Décor",
 "parentCategoryId": "home_decor",
 "level": 3
 },
{
 "name": "Table Runners",
 "categoryId": "table_runners",
 "parentCategoryName": "Kitchen & Table",
 "parentCategoryId": "kitchen_table",
 "level": 3
},
{
 "name": "Dinnerware & Serveware",
 "categoryId": "dinnerware_serveware",
 "parentCategoryName": "Kitchen & Table",
 "parentCategoryId": "kitchen_table",
 "level": 3
},
{
 "name": "Cups and Mugs",
 "categoryId": "cups_mugs",
 "parentCategoryName": "Kitchen & Table",
 "parentCategoryId": "kitchen_table",
 "level": 3
},
{
 "name": "Bakeware & Cookware",
 "categoryId": "bakeware_cookware",
 "parentCategoryName": "Kitchen & Table",
 "parentCategoryId": "kitchen_table",
 "level": 3
},
{
 "name": "Kitchen Storage & Tools",
 "categoryId": "kitchen_storage_tools",
 "parentCategoryName": "Kitchen & Table",
 "parentCategoryId": "kitchen_table",
 "level": 3
},
{
 "name": "Bar & Drinkware",
 "categoryId": "bar_drinkware",
 "parentCategoryName": "Kitchen & Table",
 "parentCategoryId": "kitchen_table",
 "level": 3
},
{
```

```
 "name": "Table Covers & Furnishings",
 "categoryId": "table_covers_furnishings",
 "parentCategoryName": "Kitchen & Table",
 "parentCategoryId": "kitchen_table",
 "level": 3
 },
 {
 "name": "Bins",
 "categoryId": "bins",
 "parentCategoryName": "Storage",
 "parentCategoryId": "storage",
 "level": 3
 },
 {
 "name": "Hangers",
 "categoryId": "hangers",
 "parentCategoryName": "Storage",
 "parentCategoryId": "storage",
 "level": 3
 },
 {
 "name": "Organisers",
 "categoryId": "organisers",
 "parentCategoryName": "Storage",
 "parentCategoryId": "storage",
 "level": 3
 },
 {
 "name": "Hooks & Holders",
 "categoryId": "hooks_holders",
 "parentCategoryName": "Storage",
 "parentCategoryId": "storage",
 "level": 3
 },
 {
 "name": "Laundry Bags",
 "categoryId": "laundry_bags",
 "parentCategoryName": "Storage",
 "parentCategoryId": "storage",
 "level": 3
 },
],
 export const menLevelThree=[
```

{

```
 "name": "Men T-Shirts",
 "categoryId": "men_t_shirts",
 "parentCategoryName": "Topwear",
 "parentCategoryId": "men_topwear"
```

```
},
{
 "name": "Men Casual Shirts",
 "categoryId": "men_casual_shirts",
 "parentCategoryName": "Topwear",
 "parentCategoryId": "men_topwear"
},
{
 "name": "Men Formal Shirts",
 "categoryId": "men_formal_shirts",
 "parentCategoryName": "Topwear",
 "parentCategoryId": "men_topwear"
},
{
 "name": "Men Sweatshirts",
 "categoryId": "men_sweatshirts",
 "parentCategoryName": "Topwear",
 "parentCategoryId": "men_topwear"
},
{
 "name": "Men Sweaters",
 "categoryId": "men_sweaters",
 "parentCategoryName": "Topwear",
 "parentCategoryId": "men_topwear"
},
{
 "name": "Men Jackets",
 "categoryId": "men_jackets",
 "parentCategoryName": "Topwear",
 "parentCategoryId": "men_topwear"
},
{
 "name": "Men Blazers & Coats",
 "categoryId": "men Blazers_and_coats",
 "parentCategoryName": "Topwear",
 "parentCategoryId": "men_topwear"
},
{
 "name": "Men Suits",
 "categoryId": "men_suits",
 "parentCategoryName": "Topwear",
 "parentCategoryId": "men_topwear"
},
{
 "name": "Men Rain Jackets",
 "categoryId": "men_rain_jackets",
 "parentCategoryName": "Topwear",
 "parentCategoryId": "men_topwear"
},
```

```
{
 "name": "Men Indian & Festive Wear",
 "categoryId": "men_indian_and_festive_wear",
 "parentCategoryName": "Topwear",
 "parentCategoryId": "men_topwear"
},
{
 "name": "Men Kurtas & Kurta Sets",
 "categoryId": "men_kurtas_and_kurta_sets",
 "parentCategoryName": "Topwear",
 "parentCategoryId": "men_topwear"
},
{
 "name": "Men Sherwanis",
 "categoryId": "men_sherwanis",
 "parentCategoryName": "Topwear",
 "parentCategoryId": "men_topwear"
},
{
 "name": "Men Nehru Jackets",
 "categoryId": "men_nehru_jackets",
 "parentCategoryName": "Topwear",
 "parentCategoryId": "men_topwear"
},
{
 "name": "Men Dhotis",
 "categoryId": "men_dhotis",
 "parentCategoryName": "Topwear",
 "parentCategoryId": "men_topwear"
},
{
 "name": "Men Jeans",
 "categoryId": "men_jeans",
 "parentCategoryName": "Bottomwear",
 "parentCategoryId": "men_bottomwear"
},
{
 "name": "Men Casual Trousers",
 "categoryId": "men_casual_trousers",
 "parentCategoryName": "Bottomwear",
 "parentCategoryId": "men_bottomwear"
},
{
 "name": "Men Formal Trousers",
 "categoryId": "men_formal_trousers",
 "parentCategoryName": "Bottomwear",
 "parentCategoryId": "men_bottomwear"
},
{
```

```
"name": "Men Shorts",
"categoryId": "men_shorts",
"parentCategoryName": "Bottomwear",
"parentCategoryId": "men_bottomwear"
},
{
 "name": "Men Track Pants & Joggers",
 "categoryId": "men_track_pants_and_joggers",
 "parentCategoryName": "Bottomwear",
 "parentCategoryId": "men_bottomwear"
},
{
 "name": "Men Briefs & Trunks",
 "categoryId": "men_briefs_and_trunks",
 "parentCategoryName": "Innerwear & Sleepwear",
 "parentCategoryId": "men_innerwear_and_sleepwear"
},
{
 "name": "Men Boxers",
 "categoryId": "men_boxers",
 "parentCategoryName": "Innerwear & Sleepwear",
 "parentCategoryId": "men_innerwear_and_sleepwear"
},
{
 "name": "Men Vests",
 "categoryId": "men_vests",
 "parentCategoryName": "Innerwear & Sleepwear",
 "parentCategoryId": "men_innerwear_and_sleepwear"
},
{
 "name": "Men Sleepwear & Loungewear",
 "categoryId": "men_sleepwear_and_loungewear",
 "parentCategoryName": "Innerwear & Sleepwear",
 "parentCategoryId": "men_innerwear_and_sleepwear"
},
{
 "name": "Men Thermals",
 "categoryId": "men_thermals",
 "parentCategoryName": "Innerwear & Sleepwear",
 "parentCategoryId": "men_innerwear_and_sleepwear"
},
{
 "name": "Men Plus Size",
 "categoryId": "men_plus_size",
 "parentCategoryName": "Innerwear & Sleepwear",
 "parentCategoryId": "men_innerwear_and_sleepwear"
},
{
 "name": "Men Casual Shoes",
```

```
"categoryId": "men_casual_shoes",
"parentCategoryName": "Footwear",
"parentCategoryId": "men_footwear"
},
{
 "name": "Men Sports Shoes",
 "categoryId": "men_sports_shoes",
 "parentCategoryName": "Footwear",
 "parentCategoryId": "men_footwear"
},
{
 "name": "Men Formal Shoes",
 "categoryId": "men_formal_shoes",
 "parentCategoryName": "Footwear",
 "parentCategoryId": "men_footwear"
},
{
 "name": "Men Sneakers",
 "categoryId": "men_sneakers",
 "parentCategoryName": "Footwear",
 "parentCategoryId": "men_footwear"
},
{
 "name": "Men Sandals & Floater",
 "categoryId": "men_sandals_and_floater",
 "parentCategoryName": "Footwear",
 "parentCategoryId": "men_footwear"
},
{
 "name": "Men Flip Flops",
 "categoryId": "men_flip_flops",
 "parentCategoryName": "Footwear",
 "parentCategoryId": "men_footwear"
},
{
 "name": "Men Socks",
 "categoryId": "men_socks",
 "parentCategoryName": "Footwear",
 "parentCategoryId": "men_footwear"
},
{
 "name": "Men Sunglasses & Frames",
 "categoryId": "men_sunglasses_and_frames",
 "parentCategoryName": "Personal Care & Grooming",
 "parentCategoryId": "men_personal_care_and_grooming"
},
{
 "name": "Men Watches",
 "categoryId": "men_watches",
```

```
 "parentCategoryName": "Personal Care & Grooming",
 "parentCategoryId": "men_personal_care_and_grooming"
 },
 {
 "name": "Men Sports & Active Wear",
 "categoryId": "men_sports_and_active_wear",
 "parentCategoryName": "Personal Care & Grooming",
 "parentCategoryId": "men_personal_care_and_grooming"
 },
 {
 "name": "Men Sports Sandals",
 "categoryId": "men_sports_sandals",
 "parentCategoryName": "Personal Care & Grooming",
 "parentCategoryId": "men_personal_care_and_grooming"
 },
 {
 "name": "Men Active T-Shirts",
 "categoryId": "men_active_t_shirts",
 "parentCategoryName": "Personal Care & Grooming",
 "parentCategoryId": "men_personal_care_and_grooming"
 },
 {
 "name": "Men Track Pants & Shorts",
 "categoryId": "men_track_pants_and_shorts",
 "parentCategoryName": "Personal Care & Grooming",
 "parentCategoryId": "men_personal_care_and_grooming"
 },
 {
 "name": "Men Tracksuits",
 "categoryId": "men_tracksuits",
 "parentCategoryName": "Personal Care & Grooming",
 "parentCategoryId": "men_personal_care_and_grooming"
 },
 {
 "name": "Men Jackets & Sweatshirts",
 "categoryId": "men_jackets_and_sweatshirts",
 "parentCategoryName": "Personal Care & Grooming",
 "parentCategoryId": "men_personal_care_and_grooming"
 },
 {
 "name": "Men Sports Accessories",
 "categoryId": "men_sports_accessories",
 "parentCategoryName": "Personal Care & Grooming",
 "parentCategoryId": "men_personal_care_and_grooming"
 },
 {
 "name": "Men Swimwear",
 "categoryId": "men_swimwear",
 "parentCategoryName": "Personal Care & Grooming",
 "parentCategoryId": "men_personal_care_and_grooming"
 }
```

```
 "parentCategoryId": "men_personal_care_and_grooming"
 },
 {
 "name": "Men Smart Wearables",
 "categoryId": "men_smart_wearables",
 "parentCategoryName": "Gadgets",
 "parentCategoryId": "men_gadgets"
 },
 {
 "name": "Men Fitness Gadgets",
 "categoryId": "men_fitness_gadgets",
 "parentCategoryName": "Gadgets",
 "parentCategoryId": "men_gadgets"
 },
 {
 "name": "Men Headphones",
 "categoryId": "men_headphones",
 "parentCategoryName": "Gadgets",
 "parentCategoryId": "men_gadgets"
 },
 {
 "name": "Men Speakers",
 "categoryId": "men_speakers",
 "parentCategoryName": "Gadgets",
 "parentCategoryId": "men_gadgets"
 },
 {
 "name": "Men Wallets",
 "categoryId": "men_wallets",
 "parentCategoryName": "Fashion Accessories",
 "parentCategoryId": "men_fashion_accessories"
 },
 {
 "name": "Men Belts",
 "categoryId": "men_belts",
 "parentCategoryName": "Fashion Accessories",
 "parentCategoryId": "men_fashion_accessories"
 },
 {
 "name": "Men Perfumes & Body Mists",
 "categoryId": "men_perfumes_and_body_mists",
 "parentCategoryName": "Fashion Accessories",
 "parentCategoryId": "men_fashion_accessories"
 },
 {
 "name": "Men Trimmers",
 "categoryId": "men_trimmers",
 "parentCategoryName": "Fashion Accessories",
 "parentCategoryId": "men_fashion_accessories"
 }
]
```

```
},
{
 "name": "Men Deodorants",
 "categoryId": "men_deodorants",
 "parentCategoryName": "Fashion Accessories",
 "parentCategoryId": "men_fashion_accessories"
},
{
 "name": "Men Ties, Cufflinks & Pocket Squares",
 "categoryId": "men_ties_cufflinks_and_pocket_squares",
 "parentCategoryName": "Fashion Accessories",
 "parentCategoryId": "men_fashion_accessories"
},
{
 "name": "Men Accessory Gift Sets",
 "categoryId": "men_accessory_gift_sets",
 "parentCategoryName": "Fashion Accessories",
 "parentCategoryId": "men_fashion_accessories"
},
{
 "name": "Men Caps & Hats",
 "categoryId": "men_caps_and_hats",
 "parentCategoryName": "Fashion Accessories",
 "parentCategoryId": "men_fashion_accessories"
},
{
 "name": "Men Mufflers, Scarves & Gloves",
 "categoryId": "men_mufflers_scarves_and_gloves",
 "parentCategoryName": "Fashion Accessories",
 "parentCategoryId": "men_fashion_accessories"
},
{
 "name": "Men Phone Cases",
 "categoryId": "men_phone_cases",
 "parentCategoryName": "Fashion Accessories",
 "parentCategoryId": "men_fashion_accessories"
},
{
 "name": "Men Rings & Wristwear",
 "categoryId": "men_rings_and_wristwear",
 "parentCategoryName": "Fashion Accessories",
 "parentCategoryId": "men_fashion_accessories"
},
{
 "name": "Men Helmets",
 "categoryId": "men_helmets",
 "parentCategoryName": "Fashion Accessories",
 "parentCategoryId": "men_fashion_accessories"
},
```

```

 },
 "name": "Men Bags & Backpacks",
 "categoryId": "men_bags_and_backpacks",
 "parentCategoryName": "Bags & Backpacks",
 "parentCategoryId": "men_bags_and_backpacks"
},
{
 "name": "Men Luggages & Trolleys",
 "categoryId": "men_luggages_and_trolleys",
 "parentCategoryName": "Bags & Backpacks",
 "parentCategoryId": "men_bags_and_backpacks"
}
]
export const womenLevelThree = [
{
 "name": "Women Kurtas & Suits",
 "categoryId": "women_kurtas_and_suits",
 "parentCategoryName": "Indian & Fusion Wear",
 "parentCategoryId": "women_indian_and_fusion_wear",
 "level":3
},
{
 "name": "Women Kurtis, Tunics & Tops",
 "categoryId": "women_kurtis_tunics_tops",
 "parentCategoryName": "Indian & Fusion Wear",
 "parentCategoryId": "women_indian_and_fusion_wear",
 "level":3
},
{
 "name": "Women Sarees",
 "categoryId": "women_sarees",
 "parentCategoryName": "Indian & Fusion Wear",
 "parentCategoryId": "women_indian_and_fusion_wear",
 "level":3
},
{
 "name": "Women Ethnic Wear",
 "categoryId": "women_ethnic_wear",
 "parentCategoryName": "Indian & Fusion Wear",
 "parentCategoryId": "women_indian_and_fusion_wear",
 "level":3
},
{
 "name": "Women Leggings, Salwars & Churidars",
 "categoryId": "women_leggings_salwars_churidars",
 "parentCategoryName": "Indian & Fusion Wear",
 "parentCategoryId": "women_indian_and_fusion_wear",
 "level":3
},
{
}
]
```

```
{
 "name": "Women Skirts & Palazzos",
 "categoryId": "women_skirts_palazzos",
 "parentCategoryName": "Indian & Fusion Wear",
 "parentCategoryId": "women_indian_and_fusion_wear",
 "level":3
,
{
 "name": "Women Dress Materials",
 "categoryId": "women_dress_materials",
 "parentCategoryName": "Indian & Fusion Wear",
 "parentCategoryId": "women_indian_and_fusion_wear",
 "level":3
,
{
 "name": "Women Lehenga Cholis",
 "categoryId": "women_lehenga_cholis",
 "parentCategoryName": "Indian & Fusion Wear",
 "parentCategoryId": "women_indian_and_fusion_wear",
 "level":3
,
{
 "name": "Women Dupattas & Shawls",
 "categoryId": "women_dupattas_shawls",
 "parentCategoryName": "Indian & Fusion Wear",
 "parentCategoryId": "women_indian_and_fusion_wear",
 "level":3
,
{
 "name": "Women Jackets",
 "categoryId": "women_jackets",
 "parentCategoryName": "Indian & Fusion Wear",
 "parentCategoryId": "women_indian_and_fusion_wear",
 "level":3
,
{
 "name": "Women Belts, Scarves & More",
 "categoryId": "women_belts_scarves_more",
 "parentCategoryName": "Indian & Fusion Wear",
 "parentCategoryId": "women_indian_and_fusion_wear",
 "level":3
,
{
 "name": "Women Watches & Wearables",
 "categoryId": "women_watches_wearables",
 "parentCategoryName": "Indian & Fusion Wear",
 "parentCategoryId": "women_indian_and_fusion_wear",
 "level":3
,
}
```

```
{
 "name": "Women Western Wear",
 "categoryId": "women_western_wear",
 "parentCategoryName": "Western Wear",
 "parentCategoryId": "women_western_wear",
 "level":3
,
{
 "name": "Women Dresses",
 "categoryId": "women_dresses",
 "parentCategoryName": "Western Wear",
 "parentCategoryId": "women_western_wear",
 "level":3
,
{
 "name": "Women Tops",
 "categoryId": "women_tops",
 "parentCategoryName": "Western Wear",
 "parentCategoryId": "women_western_wear",
 "level":3
,
{
 "name": "Women Tshirts",
 "categoryId": "women_tshirts",
 "parentCategoryName": "Western Wear",
 "parentCategoryId": "women_western_wear",
 "level":3
,
{
 "name": "Women Jeans",
 "categoryId": "women_jeans",
 "parentCategoryName": "Western Wear",
 "parentCategoryId": "women_western_wear",
 "level":3
,
{
 "name": "Women Trouzers & Capris",
 "categoryId": "women_trousers_capris",
 "parentCategoryName": "Western Wear",
 "parentCategoryId": "women_western_wear",
 "level":3
,
{
 "name": "Women Shorts & Skirts",
 "categoryId": "women_shorts_skirts",
 "parentCategoryName": "Western Wear",
 "parentCategoryId": "women_western_wear",
 "level":3
,
}
```

```
{
 "name": "Women Co-ords",
 "categoryId": "women_coords",
 "parentCategoryName": "Western Wear",
 "parentCategoryId": "women_western_wear",
 "level":3
},
{
 "name": "Women Playsuits",
 "categoryId": "women_playsuits",
 "parentCategoryName": "Western Wear",
 "parentCategoryId": "women_western_wear",
 "level":3
},
{
 "name": "Women Jumpsuits",
 "categoryId": "women_jumpsuits",
 "parentCategoryName": "Western Wear",
 "parentCategoryId": "women_western_wear",
 "level":3
},
{
 "name": "Women Shrugs",
 "categoryId": "women_shrugs",
 "parentCategoryName": "Western Wear",
 "parentCategoryId": "women_western_wear",
 "level":3
},
{
 "name": "Women Sweaters & Sweatshirts",
 "categoryId": "women_sweaters_sweatshirts",
 "parentCategoryName": "Western Wear",
 "parentCategoryId": "women_western_wear",
 "level":3
},
{
 "name": "Women Jackets & Coats",
 "categoryId": "women_jackets_coats",
 "parentCategoryName": "Western Wear",
 "parentCategoryId": "women_western_wear",
 "level":3
},
{
 "name": "Women Blazers & Waistcoats",
 "categoryId": "women Blazers_waistcoats",
 "parentCategoryName": "Western Wear",
 "parentCategoryId": "women_western_wear",
 "level":3
},
}
```

```
{
 "name": "Women Plus Size",
 "categoryId": "women_plus_size",
 "parentCategoryName": "Western Wear",
 "parentCategoryId": "women_western_wear",
 "level":3
},
{
 "name": "Women Maternity",
 "categoryId": "women_maternity",
 "parentCategoryName": "Western Wear",
 "parentCategoryId": "women_western_wear",
 "level":3
},
{
 "name": "Women Sunglasses & Frames",
 "categoryId": "women_sunglasses_frames",
 "parentCategoryName": "Western Wear",
 "parentCategoryId": "women_western_wear",
 "level":3
},
{
 "name": "Women Footwear",
 "categoryId": "women_footwear",
 "parentCategoryName": "Footwear",
 "parentCategoryId": "women_footwear",
 "level":3
},
{
 "name": "Women Flats",
 "categoryId": "women_flats",
 "parentCategoryName": "Footwear",
 "parentCategoryId": "women_footwear",
 "level":3
},
{
 "name": "Women Casual Shoes",
 "categoryId": "women_casual_shoes",
 "parentCategoryName": "Footwear",
 "parentCategoryId": "women_footwear",
 "level":3
},
{
 "name": "Women Heels",
 "categoryId": "women_heels",
 "parentCategoryName": "Footwear",
 "parentCategoryId": "women_footwear",
 "level":3
},
}
```

```
{
 "name": "Women Boots",
 "categoryId": "women_boots",
 "parentCategoryName": "Footwear",
 "parentCategoryId": "women_footwear",
 "level":3
,
{
 "name": "Women Sports Shoes & Floater",
 "categoryId": "women_sports_shoes_floater",
 "parentCategoryName": "Footwear",
 "parentCategoryId": "women_footwear",
 "level":3
,
{
 "name": "Women Sports & Active Wear",
 "categoryId": "women_sports_active_wear",
 "parentCategoryName": "Sports & Active Wear",
 "parentCategoryId": "women_sports_active_wear",
 "level":3
,
{
 "name": "Women Clothing",
 "categoryId": "women_clothing",
 "parentCategoryName": "Sports & Active Wear",
 "parentCategoryId": "women_sports_active_wear",
 "level":3
,
{
 "name": "Women Footwear",
 "categoryId": "women_footwear",
 "parentCategoryName": "Sports & Active Wear",
 "parentCategoryId": "women_sports_active_wear",
 "level":3
,
{
 "name": "Women Sports Accessories",
 "categoryId": "women_sports_accessories",
 "parentCategoryName": "Sports & Active Wear",
 "parentCategoryId": "women_sports_active_wear",
 "level":3
,
{
 "name": "Women Sports Equipment",
 "categoryId": "women_sports_equipment",
 "parentCategoryName": "Sports & Active Wear",
 "parentCategoryId": "women_sports_active_wear",
 "level":3
,
}
```

```
{
 "name": "Women Lingerie & Sleepwear",
 "categoryId": "women_lingerie_sleepwear",
 "parentCategoryName": "Lingerie & Sleepwear",
 "parentCategoryId": "women_lingerie_sleepwear",
 "level":3
},
{
 "name": "Women Bra",
 "categoryId": "women_bra",
 "parentCategoryName": "Lingerie & Sleepwear",
 "parentCategoryId": "women_lingerie_sleepwear",
 "level":3
},
{
 "name": "Women Briefs",
 "categoryId": "women_briefs",
 "parentCategoryName": "Lingerie & Sleepwear",
 "parentCategoryId": "women_lingerie_sleepwear",
 "level":3
},
{
 "name": "Women Shapewear",
 "categoryId": "women_shapewear",
 "parentCategoryName": "Lingerie & Sleepwear",
 "parentCategoryId": "women_lingerie_sleepwear",
 "level":3
},
{
 "name": "Women Sleepwear & Loungewear",
 "categoryId": "women_sleepwear_loungewear",
 "parentCategoryName": "Lingerie & Sleepwear",
 "parentCategoryId": "women_lingerie_sleepwear",
 "level":3
},
{
 "name": "Women Swimwear",
 "categoryId": "women_swimwear",
 "parentCategoryName": "Lingerie & Sleepwear",
 "parentCategoryId": "women_lingerie_sleepwear",
 "level":3
},
{
 "name": "Women Camisoles & Thermals",
 "categoryId": "women_camisoles_thermals",
 "parentCategoryName": "Lingerie & Sleepwear",
 "parentCategoryId": "women_lingerie_sleepwear",
 "level":3
},
}
```

```
{
 "name": "Women Beauty & Personal Care",
 "categoryId": "women_beauty_personal_care",
 "parentCategoryName": "Beauty & Personal Care",
 "parentCategoryId": "women_beauty_personal_care",
 "level":3
},
{
 "name": "Women Makeup",
 "categoryId": "women_makeup",
 "parentCategoryName": "Beauty & Personal Care",
 "parentCategoryId": "women_beauty_personal_care",
 "level":3
},
{
 "name": "Women Skincare",
 "categoryId": "women_skincare",
 "parentCategoryName": "Beauty & Personal Care",
 "parentCategoryId": "women_beauty_personal_care",
 "level":3
},
{
 "name": "Women Premium Beauty",
 "categoryId": "women_premium_beauty",
 "parentCategoryName": "Beauty & Personal Care",
 "parentCategoryId": "women_beauty_personal_care",
 "level":3
},
{
 "name": "Women Lipsticks",
 "categoryId": "women_lipsticks",
 "parentCategoryName": "Beauty & Personal Care",
 "parentCategoryId": "women_beauty_personal_care",
 "level":3
},
{
 "name": "Women Fragrances",
 "categoryId": "women_fragrances",
 "parentCategoryName": "Beauty & Personal Care",
 "parentCategoryId": "women_beauty_personal_care",
 "level":3
},
{
 "name": "Women Gadgets",
 "categoryId": "women_gadgets",
 "parentCategoryName": "Gadgets",
 "parentCategoryId": "women_gadgets",
 "level":3
},
}
```

```
{
 "name": "Women Smart Wearables",
 "categoryId": "women_smart_wearables",
 "parentCategoryName": "Gadgets",
 "parentCategoryId": "women_gadgets",
 "level":3
,
{
 "name": "Women Fitness Gadgets",
 "categoryId": "women_fitness_gadgets",
 "parentCategoryName": "Gadgets",
 "parentCategoryId": "women_gadgets",
 "level":3
,
{
 "name": "Women Headphones",
 "categoryId": "women_headphones",
 "parentCategoryName": "Gadgets",
 "parentCategoryId": "women_gadgets",
 "level":3
,
{
 "name": "Women Speakers",
 "categoryId": "women_speakers",
 "parentCategoryName": "Gadgets",
 "parentCategoryId": "women_gadgets",
 "level":3
,
{
 "name": "Women Jewellery",
 "categoryId": "women_jewellery",
 "parentCategoryName": "Jewellery",
 "parentCategoryId": "women_jewellery",
 "level":3
,
{
 "name": "Women Fashion Jewellery",
 "categoryId": "women_fashion_jewellery",
 "parentCategoryName": "Jewellery",
 "parentCategoryId": "women_jewellery",
 "level":3
,
{
 "name": "Women Fine Jewellery",
 "categoryId": "women_fine_jewellery",
 "parentCategoryName": "Jewellery",
 "parentCategoryId": "women_jewellery",
 "level":3
,
}
```

```
{
 "name": "Women Earrings",
 "categoryId": "women_earrings",
 "parentCategoryName": "Jewellery",
 "parentCategoryId": "women_jewellery",
 "level":3
},
{
 "name": "Women Backpacks",
 "categoryId": "women_backpacks",
 "parentCategoryName": "Handbags, Bags & Wallets",
 "parentCategoryId": "women_backpacks_bags_wallets",
 "level":3
},
{
 "name": "Women Handbags, Bags & Wallets",
 "categoryId": "women_handbags_bags_wallets",
 "parentCategoryName": "Handbags, Bags & Wallets",
 "parentCategoryId": "women_handbags_bags_wallets",
 "level":3
},
{
 "name": "Women Luggages & Trolleys",
 "categoryId": "women_luggages_trolleys",
 "parentCategoryName": "Handbags, Bags & Wallets",
 "parentCategoryId": "women_handbags_bags_wallets",
 "level":3
}
]
export const electronicsLevelTwo = [
{
 "parentCategoryId": "electronics",
 "categoryId": "mobiles",
 "level": 2,
 "name": "Mobiles",
 "parentCategoryName": "Electronics"
},
{
 "parentCategoryId": "electronics",
 "categoryId": "mobile_accessories",
 "level": 2,
 "name": "Mobile Accessories",
 "parentCategoryName": "Electronics"
},
{
 "parentCategoryId": "electronics",
 "categoryId": "smart_wearable_tech",
 "level": 2,
 "name": "Smart Wearable Tech",
 "parentCategoryName": "Electronics"
}
```

```
 "parentCategoryName": "Electronics"
 },
 // {
 // "parentCategoryId": "electronics",
 // "categoryId": "health_care_appliances",
 // "level": 2,
 // "name": "Health Care Appliances",
 // "parentCategoryName": "Electronics"
 // },
 {
 "parentCategoryId": "electronics",
 "categoryId": "laptops",
 "level": 2,
 "name": "Laptops",
 "parentCategoryName": "Electronics"
 },
 {
 "parentCategoryId": "electronics",
 "categoryId": "tablets",
 "level": 2,
 "name": "Tablets",
 "parentCategoryName": "Electronics"
 },
 // {
 // "parentCategoryId": "electronics",
 // "categoryId": "televisions",
 // "level": 2,
 // "name": "Televisions",
 // "parentCategoryName": "Electronics"
 // },
 {
 "parentCategoryId": "electronics",
 "categoryId": "speakers",
 "level": 2,
 "name": "Speakers",
 "parentCategoryName": "Electronics"
 },
 // {
 // "parentCategoryId": "electronics",
 // "categoryId": "smart_home_automation",
 // "level": 2,
 // "name": "Smart Home Automation",
 // "parentCategoryName": "Electronics"
 // },
 {
 "parentCategoryId": "electronics",
 "categoryId": "camera",
 "level": 2,
 "name": "Camera",
```

```
 "parentCategoryName": "Electronics"
 },
 // {
 // "parentCategoryId": "electronics",
 // "categoryId": "network_components",
 // "level": 2,
 // "name": "Network Components",
 // "parentCategoryName": "Electronics"
 // },
 // {
 // "parentCategoryId": "electronics",
 // "categoryId": "featured",
 // "level": 2,
 // "name": "Featured",
 // "parentCategoryName": "Electronics"
 // }
]

export const furnitureLevelTwo=[
 {
 "name": "Bed Linen & Furnishing",
 "categoryId": "bed_linen_furnishing",
 "parentCategoryName": "Furniture",
 "parentCategoryId": "furniture",
 "level": 2
 },
 {
 "name": "Flooring",
 "categoryId": "flooring",
 "parentCategoryName": "Furniture",
 "parentCategoryId": "furniture",
 "level": 2
 },
 {
 "name": "Bath",
 "categoryId": "bath",
 "parentCategoryName": "Furniture",
 "parentCategoryId": "furniture",
 "level": 2
 },
 {
 "name": "Lamps & Lighting",
 "categoryId": "lamps_lighting",
 "parentCategoryName": "Furniture",
 "parentCategoryId": "furniture",
 "level": 2
 },
 {
 "name": "Home Décor",
 "categoryId": "home_decor",
 "parentCategoryName": "Furniture",
 "parentCategoryId": "furniture",
 "level": 2
 }]
```

```
"parentCategoryName": "Furniture",
"parentCategoryId": "furniture",
"level": 2
},
{
 "name": "Kitchen & Table",
 "categoryId": "kitchen_table",
 "parentCategoryName": "Furniture",
 "parentCategoryId": "furniture",
 "level": 2
},
// {
// "name": "Storage",
// "categoryId": "storage",
// "parentCategoryName": "Furniture",
// "parentCategoryId": "furniture",
// "level": 2
// },
]

export const menLevelTwo=[
{
 "name": "Topwere",
 "categoryId": "men_topwear",
 "parentCategoryId": "men",
 "level":2
},
{
 "name": "Bottomwere",
 "categoryId": "men_bottomwear",
 "parentCategoryId": "men",
 "level":2
},
{
 "name": "Innerwere And Sleepwere",
 "categoryId": "men_innerwear_and_sleepwear",
 "parentCategoryId": "men",
 "level":2
},
{
 "name": "Footwere",
 "categoryId": "men_footwear",
 "parentCategoryId": "men",
 "level":2
},
{
 "name": "Personal Care And grooming",
 "categoryId": "men_personal_care_and_grooming",
 "parentCategoryId": "men",
```

```
 "level":2
 },
{
 "name": "Fashion Accessories",
 "categoryId": "men_fashion_accessories",
 "parentCategoryId": "men",
 "level":2
},
// {
// "name": "Gadgets",
// "categoryId": "men_gadgets",
// "parentCategoryId": "men",
// "level":2
// },
// {
// "name": "Bags And Backpacks",
// "categoryId": "men_bags_and_backpacks",
// "parentCategoryId": "men",
// "level":2
// }
]
export const womenLevelTwo=[
 {
 "parentCategoryId": "women",
 "level":2,
 "name": "Indian & fusion Wear",
 "categoryId": "women_indian_and_fusion_wear"
 },
 {
 "parentCategoryId": "women",
 "level":2,
 "name": "western wear",
 "categoryId": "women_western_wear"
 },
 {
 "parentCategoryId": "women",
 "level":2,
 "name": "Footwear",
 "categoryId": "women_footwear"
 },
 {
 "parentCategoryId": "women",
 "level":2,
 "name": "Sports & Active Wear",
 "categoryId": "women_sports_active_wear"
 },
 {
 "parentCategoryId": "women",
 "level":2,
```

```

 "name": "Lingerie Sleepwear",
 "categoryId": "women_lingerie_sleepwear"
 },
 {
 "parentCategoryId": "women",
 "level": 2,
 "name": "Buauty & Personal Care",
 "categoryId": "women_beauty_personal_care"
 },
 {
 "parentCategoryId": "women",
 "level": 2,
 "name": "Gadgets",
 "categoryId": "women_gadgets"
 },
 {
 "parentCategoryId": "women",
 "level": 2,
 "name": "Jewellery",
 "categoryId": "women_jewellery"
 },
 {
 "parentCategoryId": "women",
 "level": 2,
 "name": "Handbags, Bags & wallets",
 "categoryId": "women_handbags_bags_wallets"
 }
]
export const mainCategory = [
{
 name: "Men",
 categoryId: "men",
 level: 1,
 levelTwoCategory:[
 {
 "name": "Topwere",
 "categoryId": "men_topwear",
 "parentCategoryId": "men",
 "level": 2
 },
 {
 "name": "Bottomwere",
 "categoryId": "men_bottomwear",
 "parentCategoryId": "men",
 "level": 2
 },
 {
 "name": "Innerwere And Sleepwere",
 "categoryId": "men_innerwear_and_sleepwear",

```

```
 "parentCategoryId": "men",
 "level": 2
 },
 {
 "name": "Footwere",
 "categoryId": "men_footwear",
 "parentCategoryId": "men",
 "level": 2
 },
 {
 "name": "Personal Care And grooming",
 "categoryId": "men_personal_care_and_grooming",
 "parentCategoryId": "men",
 "level": 2
 },
 {
 "name": "Fashion Accessories",
 "categoryId": "men_fashion_accessories",
 "parentCategoryId": "men",
 "level": 2
 },
 {
 "name": "Gadgets",
 "categoryId": "men_gadgets",
 "parentCategoryId": "men",
 "level": 2
 },
 {
 "name": "Bags And Backpacks",
 "categoryId": "men_bags_and_backpacks",
 "parentCategoryId": "men",
 "level": 2
 }
]
},
{
 name: "Women",
 categoryId: "women",
 level: 1,
 levelTowCategory: [
 {
 "parentCategoryId": "women",
 "level": 2,
 "name": "Indian & fusion Wear",
 "categoryId": "women_indian_and_fusion_wear"
 },
 {
 "parentCategoryId": "women",
 "level": 2,
```

```
 "name": "western wear",
 "categoryId": "women_western_wear"
 },
 {
 "parentCategoryId": "women",
 "level": 2,
 "name": "Footwear",
 "categoryId": "women_footwear"
 },
 {
 "parentCategoryId": "women",
 "level": 2,
 "name": "Sports & Active Wear",
 "categoryId": "women_sports_active_wear"
 },
 {
 "parentCategoryId": "women",
 "level": 2,
 "name": "Lingerie Sleepwear",
 "categoryId": "women_lingerie_sleepwear"
 },
 {
 "parentCategoryId": "women",
 "level": 2,
 "name": "Beauty & Personal Care",
 "categoryId": "women_beauty_personal_care"
 },
 {
 "parentCategoryId": "women",
 "level": 2,
 "name": "Gadgets",
 "categoryId": "women_gadgets"
 },
 {
 "parentCategoryId": "women",
 "level": 2,
 "name": "Jewellery",
 "categoryId": "women_jewellery"
 },
 {
 "parentCategoryId": "women",
 "level": 2,
 "name": "Handbags, Bags & wallets",
 "categoryId": "women_handbags_bags_wallets"
 }
]
},
{
```

```
 name: "Home & Furniture",
 categoryId: "home_furniture",
 level: 1,
 },
 {
 name: "Electronics", categoryId: "electronics", level: 1
 }
];
[
 {
 "title": "Ethnic Motifs Woven Design Zari Kanjeevaram Saree",
 "description": "Purple and silver-toned kanjeevaram saree\nEthnic motifs woven design saree with woven design border\nHas zari detail\n\nThe saree comes with an unstitched blouse piece\nThe blouse worn by the model might be for modelling purpose only. Check the image of the blouse piece to understand how the actual blouse piece looks like.",
 "mrpPrice": 4999,
 "sellingPrice": 2499,
 "quantity": "",
 "color": "Purple",
 "images": [
 "http://res.cloudinary.com/dxoqwusir/image/upload/v1725359611/SUHANI-2011_5_nrw_whr.webp",
 "http://res.cloudinary.com/dxoqwusir/image/upload/v1725359617/SUHANI-2011_4_rsj_092.webp",
 "http://res.cloudinary.com/dxoqwusir/image/upload/v1725359626/silk-saree_xfukfd.jpg",
 "http://res.cloudinary.com/dxoqwusir/image/upload/v1725359634/SUHANI-2011_2_byf_z48.webp"
],
 "category": "women",
 "category2": "women_indian_and_fusion_wear",
 "category3": "women_sarees",
 "sizes": "FREE"
 },
 {
 "title": "Ethnic Motifs Woven Design Zari Kanjeevaram Saree",
 "description": "Purple and silver-toned kanjeevaram saree\nEthnic motifs woven design saree with woven design border\nHas zari detail\n\nThe saree comes with an unstitched blouse piece\nThe blouse worn by the model might be for modelling purpose only. Check the image of the blouse piece to understand how the actual blouse piece looks like.",
 "mrpPrice": 4999,
 "sellingPrice": 2499,
 "quantity": "",
 "color": "Purple",
 }
]
```

```

 "images": [
 "http://res.cloudinary.com/dxoqwusir/image/upload/v1725359611/SUHANI-2011_5_nrw
 whr.webp",
 "http://res.cloudinary.com/dxoqwusir/image/upload/v1725359617/SUHANI-2011_4_rsj
 092.webp",
 "http://res.cloudinary.com/dxoqwusir/image/upload/v1725359626/silk-saree_xfukfd
 .jpg",
 "http://res.cloudinary.com/dxoqwusir/image/upload/v1725359634/SUHANI-2011_2_byf
 z48.webp"
],
 "category": "women",
 "category2": "women_indian_and_fusion_wear",
 "category3": "women_sarees",
 "sizes": "FREE"
 }
]
export const brands=[
 {"name": "nike", "value": "nike"},

 {"name": "FabIndia", "value": "fabindia"},

 {"name": "Biba", "value": "biba"},

 {"name": "Manyavar", "value": "manyavar"},

 {"name": "W for Woman", "value": "wforwoman"},

 {"name": "Allen Solly", "value": "allen_solly"},

 {"name": "Peter England", "value": "peter_england"},

 {"name": "Pantaloons", "value": "pantaloons"}

]
export const colors=[

 {"name": "Pink", "hex": "#FFC0CB"},

 {"name": "Green", "hex": "#008000"},

 {"name": "Blue", "hex": "#0000FF"},

 {"name": "Red", "hex": "#FF0000"},

 {"name": "Yellow", "hex": "#FFFF00"},

 {"name": "Black", "hex": "#000000"},

 {"name": "Purple", "hex": "#800080"},

 {"name": "Navy Blue", "hex": "#000080"},

 {"name": "Maroon", "hex": "#800000"},

 {"name": "Peach", "hex": "#FFDAB9"},

 {"name": "White", "hex": "#FFFFFF"},

 {"name": "Grey", "hex": "#808080"},

 {"name": "Teal", "hex": "#008080"},

 {"name": "Turquoise Blue", "hex": "#00CED1"},

 {"name": "Mustard", "hex": "#FFDB58"},

 {"name": "Beige", "hex": "#F5F5DC"},

 {"name": "Cream", "hex": "#FFFDD0"},

 {"name": "Sea Green", "hex": "#2E8B57"},

]

```

```

 {"name": "Orange", "hex": "#FFA500"},

 {"name": "Brown", "hex": "#A52A2A"},

 {"name": "Off White", "hex": "#FAF0E6"},

 {"name": "Burgundy", "hex": "#800020"},

 {"name": "Lavender", "hex": "#E6E6FA"},

 {"name": "Magenta", "hex": "#FF00FF"},

 {"name": "Olive", "hex": "#808000"},

 {"name": "Mauve", "hex": "#E0B0FF"},

 {"name": "Lime Green", "hex": "#32CD32"},

 {"name": "Gold", "hex": "#FFD700"},

 {"name": "Rust", "hex": "#B7410E"},

 {"name": "Coral", "hex": "#FF7F50"},

 {"name": "Rose", "hex": "#FF007F"},

 {"name": "Coffee Brown", "hex": "#4B3621"},

 {"name": "Fluorescent Green", "hex": "#00FF00"},

 {"name": "Silver", "hex": "#C0C0C0"},

 {"name": "Taupe", "hex": "#483C32"},

 {"name": "Tan", "hex": "#D2B48C"},

 {"name": "Nude", "hex": "#FFFAE6"},

 {"name": "Metallic", "hex": "#D4AF37"},

 {"name": "Khaki", "hex": "#C3B091"},

 {"name": "Grey Melange", "hex": "#BEBEBE"},

 {"name": "Charcoal", "hex": "#36454F"},

 {"name": "Bronze", "hex": "#CD7F32"}

]

 export const discount=[

 {"name": "10% and above", "value": 10},

 {"name": "20% and above", "value": 20},

 {"name": "30% and above", "value": 30},

 {"name": "40% and above", "value": 40},

 {"name": "50% and above", "value": 50},

 {"name": "60% and above", "value": 60},

 {"name": "70% and above", "value": 70},

 {"name": "80% and above", "value": 80}

]

 export const price =[

 {"name": "Below ₹500", "min": 0, "max": 500, value:"500"},

 {"name": "₹500 - ₹1000", "min": 500, "max": 1000, value:"500 - 1000"},

 {"name": "₹1000 - ₹2000", "min": 1000, "max": 2000, value:"1000 - 2000"},

 {"name": "₹2000 - ₹3000", "min": 2000, "max": 3000, value:"2000 - 3000"},

 {"name": "₹3000 - ₹5000", "min": 3000, "max": 5000, value:"3000 - 5000"},

 {"name": "₹5000 - ₹10000", "min": 5000, "max": 10000, value: "5000 - 10000"},

 {"name": "₹10000 and above", "min": 10000, "max": null, value:"10000"}

]

 import React from 'react'

 const Mobile = () => {

 return (

```

```
<div>

 Image 1
 Image 2
 Image 3
 Image 4
 Image 5
 Image 6
 Image 7
 Image 8
 Image 9
 Image 10
 Image 11
 Image 12
 Image 13
 Image 14
 Image 15

</div>
```

```
)
}

export default Mobile
export const banners = [

 "https://www.taneira.com/dw/image/v2/BKMH_PRD/on/demandware.static/-/Sites-Taneira-Library/default/dw53b6ec27/HomePage/Banners/Apsara/N_Apsara_WebBanners_1920x768_D.jpg",

 "https://www.taneira.com/dw/image/v2/BKMH_PRD/on/demandware.static/-/Sites-Taneira-Library/default/dwdc296071/HomePage/Banners/Queens/HP_Queen_D.jpg",

 [

 "https://www.taneira.com/on/demandware.static/-/Sites-Taneira-Library/default/dwbc5a4d1c/images/login_signup/Login_COI.jpg",
],
];
[
 {
 "categoryId": "laptops",
 "section": "ELECTRIC_CATEGORIES",
 "name": "Laptop",
 "image":
 "https://rukminim2.flixcart.com/image/312/312/xif0q/computer/x/9/j/-original-imahyjzh7m2zsqdg.jpeg?q=70"
 },
 {
 "categoryId": "mobiles",
 "section": "ELECTRIC_CATEGORIES",
 "name": "Mobile",
 "image":
 "https://rukminim2.flixcart.com/image/416/416/xif0q/mobile/5/t/j/edge-50-fusion-pb300002in-motorola-original-imahywzrfagkuyxx.jpeg?q=70&crop=false"
 },
 {
 "categoryId": "smart_watches",
 "section": "ELECTRIC_CATEGORIES",
 "name": "Smartwatch",
 "image":
 "https://rukminim2.flixcart.com/image/612/612/xif0q/smartwatch/f/g/g/-original-imagynz46fngcks.jpeg?q=70"
 },
 {
 "categoryId": "headphones_headsets",
 "section": "ELECTRIC_CATEGORIES",
 "name": "Headphones",
 }]
```

```
 "image":
"https://rukminim2.flixcart.com/image/612/612/kz4gh3k0/headphone/c/v/r/-original-
l-imagb7bmhdgghzxq.jpeg?q=70"
,
{
 "categoryId": "speakers",
 "section": "ELECTRIC_CATEGORIES",
 "name": "Speaker",
 "image":
"https://rukminim2.flixcart.com/image/612/612/xif0q/speaker/6/z/2/-original-ima
hfgfkr5gkk9aq.jpeg?q=70"
,
{
 "categoryId": "television",
 "section": "ELECTRIC_CATEGORIES",
 "name": "Tv",
 "image":
"https://rukminim2.flixcart.com/image/312/312/xif0q/television/9/p/9/-original-
imah2v29z86u7b79.jpeg?q=70"
,
{
 "categoryId": "cameras",
 "section": "ELECTRIC_CATEGORIES",
 "name": "Camera",
 "image":
"https://rukminim2.flixcart.com/image/312/312/jfbfde80/camera/n/r/n/canon-eos-e
os-3000d-dslr-original-imaf3t5h9yuyc5zu.jpeg?q=70"
,
{
 "categoryId": "women_lehenga_cholis",
 "section": "GRID",
 "name": "women lehenga cholis",
 "image":
"https://assets.myntassets.com/h_720,q_90,w_540/v1/assets/images/23807268/2023/
6/29/9930b235-5318-4755-abbe-08f99e969e781688026636544LehengaCholi7.jpg"
,
{
 "categoryId": "men_formal_shoes",
 "section": "GRID",
 "name": "men formal shoes",
 "image":
"https://assets.myntassets.com/h_720,q_90,w_540/v1/assets/images/24651572/2023/
8/25/4fbf6d8c-d093-46c5-a5a6-7dd67c0c76551692964752597HouseofPataudiMenTanFauxL
eatherFormalSlipOnLoafers1.jpg"
,
{
 "categoryId": "women_lehenga_cholis",
 "section": "GRID",
 "name": "women lehenga cholis",
```

```
 "image":
"https://images.pexels.com/photos/12730873/pexels-photo-12730873.jpeg?auto=compress&cs=tinysrgb&w=1260&h=750&dpr=1"
,
{
 "categoryId": "men_sherwanis",
 "section": "GRID",
 "name": "men sherwanis",
 "image":
"https://shreeman.in/cdn/shop/files/20_3cfbd5a3-ecb6-482a-b798-7ffd9de1c784.jpg
?v=1712061674&width=700"
,
{
 "categoryId": "women_jewellery",
 "section": "GRID",
 "name": "women jewellery",
 "image":
"https://media.istockphoto.com/id/1276740597/photo/indian-traditional-gold-necklace.jpg?b=1&s=612x612&w=0&k=20&c=S-QnNZKqf2u3L-GIaDiIinNRU74GBWQaIDwY7gYJboY="
,
{
 "categoryId": "women_footwear",
 "section": "GRID",
 "name": "women footwear",
 "image":
"https://assets.myntassets.com/h_720,q_90,w_540/v1/assets/images/13837166/2021/
8/19/04e40e02-4c56-4705-94d0-f444b29973aa1629373611707-House-of-Pataudi-Women-M
aroon-Embellished-Handcrafted-Wedges-1.jpg"
,
{
 "name": "Home Décor",
 "categoryId": "home_decor",

 "section": "SHOP_BY_CATEGORIES",

 "image": "https://assets.myntassets.com/h_720,q_90,w_540/v1/assets/images/284609
38/2024/3/22/7fb09e9c-86e0-4602-b54e-fa5c0171b50b1711104156746IrregularMirrorHo
meDecor1.jpg"
,
{
 "name": "Kitchen & Table",
 "categoryId": "kitchen_table",
```

```
"section":"SHOP_BY_CATEGORIES",

"image":"https://assets.myntassets.com/h_720,q_90,w_540/v1/assets/images/198734
92/2022/9/10/fd72939c-c379-4bab-9fd6-918c89172d161662797928387AquarelleBlue100C
ottonPrintedTableRunner1.jpg"
},

{
 "parentCategoryId":"women",

 "name":"Sports & Active Wear",
 "categoryId": "women_sports_active_wear",
 "section":"SHOP_BY_CATEGORIES",

"image":"https://assets.myntassets.com/h_1440,q_90,w_1080/v1/assets/images/2210
9480/2023/9/5/06a17ac3-46b0-4f9d-bcb1-2d3582feda041693895310152PumaWomenBrandLo
goPrintedPureCottonOutdoorT-shirt1.jpg"
},

{
 "parentCategoryId":"women",

 "name":"Lingerie Sleepwear",
 "categoryId": "women_lingerie_sleepwear",
 "section":"SHOP_BY_CATEGORIES",

"image":"https://assets.myntassets.com/h_720,q_90,w_540/v1/assets/images/362895
/2023/10/18/bfb55058-fe1a-45d6-a39d-1b8d421e02001697603774147TriumphShapeSensat
ion33withHighWaistTummyandThighControlMaxi1.jpg"
},

{
 "parentCategoryId":"women",

 "name":"Indian & fusion Wear",
 "categoryId": "women_indian_and_fusion_wear",
 "section":"SHOP_BY_CATEGORIES",

"image":"https://assets.myntassets.com/h_720,q_90,w_540/v1/assets/images/228666
94/2023/4/24/98951db4-e0a5-47f8-a1be-353863d24dc01682349679664KalinOrangeSilkB
lendEthnicWovenDesignFestiveSareewithMatchi2.jpg"
},

{
 "parentCategoryId":"women",

 "name":"western wear",
 "categoryId": "women_western_wear",
 "section":"SHOP_BY_CATEGORIES",

"image":"https://assets.myntassets.com/h_720,q_90,w_540/v1/assets/images/223915
```

```
04/2023/3/17/3259c109-060a-4c39-aba2-e9d32e2068e41679049035856StyleQuotientPeac
h-ColouredTie-UpNeckPuffSleeveCottonTop1.jpg"
},
{
 "parentCategoryId": "women",
 "name": "Women Footwear",
 "categoryId": "women_footwear",
 "section": "SHOP_BY_CATEGORIES",
}

"image": "https://assets.myntassets.com/h_720,q_90,w_540/v1/assets/images/280240
48/2024/3/5/fca98389-f9d6-4f19-b82a-53c7ee0518ec1709633175836CORSICABlockSandal
switchBows1.jpg"
},
{
 "name": "Topwere",
 "categoryId": "men_topwear",
 "parentCategoryId": "men",
 "section": "SHOP_BY_CATEGORIES",
}

"image": "https://assets.myntassets.com/h_720,q_90,w_540/v1/assets/images/230298
34/2023/9/18/96c015ae-1090-4036-954b-d9c80085b1d71695022844653-HRX-by-Hrithik-R
oshan-Men-Jackets-6981695022843934-1.jpg"
},
{
 "name": "Bottomwere",
 "categoryId": "men_bottomwear",
 "parentCategoryId": "men",
 "section": "SHOP_BY_CATEGORIES",
}

"image": "https://assets.myntassets.com/h_720,q_90,w_540/v1/assets/images/201223
24/2022/9/22/91c61c45-fe17-4d1d-8e20-0aaaf90186b61663827920015RaymondSlimFitBlu
eJeansForMen1.jpg"
},
{
 "name": "Innerwere And Sleepwere",
 "categoryId": "men_innerwear_and_sleepwear",
 "parentCategoryId": "men",
 "section": "SHOP_BY_CATEGORIES",
}

"image": "https://assets.myntassets.com/h_720,q_90,w_540/v1/assets/images/245283
50/2023/8/17/d9ee03c9-7e15-49e4-8f15-0b6f38e568121692275415567TrackPants1.jpg"
},
{
 "name": "Footwere",
 "categoryId": "men_footwear",

```

```
 "parentCategoryId": "men",
 "section": "SHOP_BY_CATEGORIES",
 "image": "https://assets.myntassets.com/h_720,q_90,w_540/v1/assets/images/27107914/2024/1/30/bd09f650-d8a4-4570-8e53-e2cfafc4ea6c1706609911015SparxMenMeshRunningShoes1.jpg",
 },
 {
 "name": "Bed Linen & Furnishing",
 "categoryId": "bed_linen_furnishing",
 "section": "SHOP_BY_CATEGORIES",
 "image": "https://assets.myntassets.com/h_720,q_90,w_540/v1/assets/images/19284508/2022/7/28/92df52de-27dc-4d72-8ab4-fee2c82c85081659003977664DreamscapeUnisexPinkBedsheets1.jpg",
 },
 {
 "name": "Flooring",
 "categoryId": "flooring",
 "section": "SHOP_BY_CATEGORIES",
 "image": "https://assets.myntassets.com/h_720,q_90,w_540/v1/assets/images/27868878/2024/2/25/53b7c07f-24a7-48e2-a791-f63dce4a0c981708844793060SANACARPETMulticolouredFloralAnti-SkidPolyesterCarpet1.jpg",
 },
 {
 "name": "Bath",
 "categoryId": "bath",
 "section": "SHOP_BY_CATEGORIES",
 "image": "https://assets.myntassets.com/h_720,q_90,w_540/v1/assets/images/16719378/2024/7/2/97418bad-8216-494f-a863-9b159aec93cf1719897710069JockeyPackof2CottonTerry500GSMUltrasoftandDurableSolidHandTo1.jpg",
 },
 {
 "name": "Lamps & Lighting",
 "categoryId": "lamps_lighting",
 "section": "SHOP_BY_CATEGORIES",
 "image": "https://assets.myntassets.com/h_720,q_90,w_540/v1/assets/images/productimage/2021/6/10/5be80654-037c-4591-99c9-a403646971981623344562713-1.jpg"
 }
}
```

```
 }
]
export const homeCategories = [
{
 categoryId: "gaming_laptops",
 section: "ELECTRIC_CATEGORIES",
 name: "Gaming Laptop",
 image:

"https://rukminim2.flixcart.com/image/312/312/xif0q/computer/x/9/j/-original-im
ahyjzh7m2zsqdg.jpeg?q=70",
},
{
 categoryId: "mobiles",
 section: "ELECTRIC_CATEGORIES",
 name: "Mobile",
 image:

"https://rukminim2.flixcart.com/image/416/416/xif0q/mobile/5/t/j/edge-50-fusion
-pb300002in-motorola-original-imahywzrfagkuyxx.jpeg?q=70&crop=false",
},
{
 categoryId: "smart_watches",
 section: "ELECTRIC_CATEGORIES",
 name: "Smartwatch",
 image:

"https://rukminim2.flixcart.com/image/612/612/xif0q/smartwatch/f/g/g/-original-
imagynz46fngcks.jpeg?q=70",
},
{
 categoryId: "headphones_headsets",
 section: "ELECTRIC_CATEGORIES",
 name: "Headphones",
 image:

"https://rukminim2.flixcart.com/image/612/612/kz4gh3k0/headphone/c/v/r/-origina
l-imagb7bmhdggzxq.jpeg?q=70",
},
{
 categoryId: "home_theatres",
 section: "ELECTRIC_CATEGORIES",
 name: "Speaker",
 image:

"https://rukminim2.flixcart.com/image/612/612/xif0q/speaker/6/z/2/-original-ima
hgfkr5gkk9aq.jpeg?q=70",
},
```

```
 categoryId: "television",
 section: "ELECTRIC_CATEGORIES",
 name: "Tv",
 image:

"https://rukminim2.flixcart.com/image/312/312/xif0q/television/9/p/9/-original-imah2v29z86u7b79.jpeg?q=70",
},
{
 categoryId: "cameras",
 section: "ELECTRIC_CATEGORIES",
 name: "Camera",
 image:

"https://rukminim2.flixcart.com/image/312/312/jfbfde80/camera/n/r/n/canon-eos-eos-3000d-dslr-original-imaf3t5h9yuyc5zu.jpeg?q=70",
},
{
 categoryId: "women_lehenga_cholis",
 section: "GRID",
 name: "women lehenga cholis",
 image:

"https://assets.myntassets.com/h_720,q_90,w_540/v1/assets/images/23807268/2023/6/29/9930b235-5318-4755-abbe-08f99e969e781688026636544LehengaCholi7.jpg",
},
{
 categoryId: "men_formal_shoes",
 section: "GRID",
 name: "men formal shoes",
 image:

"https://assets.myntassets.com/h_720,q_90,w_540/v1/assets/images/24651572/2023/8/25/4fbf6d8c-d093-46c5-a5a6-7dd67c0c76551692964752597HouseofPataudiMenTanFauxLeatherFormalSlipOnLoafers1.jpg",
},
{
 categoryId: "women_lehenga_cholis",
 section: "GRID",
 name: "women lehenga cholis",
 image:

"https://images.pexels.com/photos/12730873/pexels-photo-12730873.jpeg?auto=compress&cs=tinysrgb&w=1260&h=750&dpr=1",
},
{
 categoryId: "men_sherwanis",
 section: "GRID",
 name: "men sherwanis",
```

```
 image:

"https://shreeman.in/cdn/shop/files/20_3cfbd5a3-ecb6-482a-b798-7ffd9de1c784.jpg
?v=1712061674&width=700",
},
{
 categoryId: "women_jewellery",
 section: "GRID",
 name: "women jewellery",
 image:

"https://media.istockphoto.com/id/1276740597/photo/indian-traditional-gold-neck
lace.jpg?b=1&s=612x612&w=0&k=20&c=S-QnNZKqf2u3L-GIaDiIinNRU74GBWQaIDwY7gYJboY="
,
},
{
 categoryId: "women_footwear",
 section: "GRID",
 name: "women footwear",
 image:

"https://assets.myntassets.com/h_720,q_90,w_540/v1/assets/images/13837166/2021/
8/19/04e40e02-4c56-4705-94d0-f444b29973aa1629373611707-House-of-Pataudi-Women-M
aroon-Embellished-Handcrafted-Wedges-1.jpg",
},
{
 name: "Home Décor",
 categoryId: "home_decor",

 section: "SHOP_BY_CATEGORIES",
 image:

"https://assets.myntassets.com/h_720,q_90,w_540/v1/assets/images/28460938/2024/
3/22/7fb09e9c-86e0-4602-b54e-fa5c0171b50b1711104156746IrregularMirrorHomeDecor1
.jpg",
},
{
 name: "Kitchen & Table",
 categoryId: "kitchen_table",

 section: "SHOP_BY_CATEGORIES",
 image:

"https://assets.myntassets.com/h_720,q_90,w_540/v1/assets/images/19873492/2022/
9/10/fd72939c-c379-4bab-9fd6-918c89172d161662797928387AquarelleBlue100CottonPri
ntedTableRunner1.jpg",
},
{
}
```

```
parentCategoryId: "women",

name: "Sports & Active Wear",
categoryId: "women_sports_active_wear",
section: "SHOP_BY_CATEGORIES",
image:

"https://assets.myntassets.com/h_1440,q_90,w_1080/v1/assets/images/22109480/2023/9/5/06a17ac3-46b0-4f9d-bcb1-2d3582feda041693895310152PumaWomenBrandLogoPrintedPureCottonOutdoorT-shirt1.jpg",
},
{
parentCategoryId: "women",

name: "Lingerie Sleepwear",
categoryId: "women_lingerie_sleepwear",
section: "SHOP_BY_CATEGORIES",
image:

"https://assets.myntassets.com/h_720,q_90,w_540/v1/assets/images/362895/2023/10/18/bfb55058-fela-45d6-a39d-1b8d421e02001697603774147TriumphShapeSensation33withHighWaistTummyandThighControlMaxi1.jpg",
},
{
parentCategoryId: "women",

name: "Indian & fusion Wear",
categoryId: "women_indian_and_fusion_wear",
section: "SHOP_BY_CATEGORIES",
image:

"https://assets.myntassets.com/h_720,q_90,w_540/v1/assets/images/22866694/2023/4/24/98951db4-e0a5-47f8-a1be-353863d24dc01682349679664KaliniOrangeSilkBlendEthnicWovenDesignFestiveSareewithMatchi2.jpg",
},
{
parentCategoryId: "women",

name: "western wear",
categoryId: "women_western_wear",
section: "SHOP_BY_CATEGORIES",
image:

"https://assets.myntassets.com/h_720,q_90,w_540/v1/assets/images/22391504/2023/3/17/3259c109-060a-4c39-aba2-e9d32e2068e41679049035856StyleQuotientPeach-ColouredTie-UpNeckPuffSleeveCottonTop1.jpg",
},
```

```
parentCategoryId: "women",

name: "Women Footwear",
categoryId: "women_footwear",
section: "SHOP_BY_CATEGORIES",
image:

"https://assets.myntassets.com/h_720,q_90,w_540/v1/assets/images/28024048/2024/
3/5/fca98389-f9d6-4f19-b82a-53c7ee0518ec1709633175836CORSICABlockSandalswithBow
s1.jpg",
},
{
name: "Topwere",
categoryId: "men_topwear",
parentCategoryId: "men",

section: "SHOP_BY_CATEGORIES",
image:

"https://assets.myntassets.com/h_720,q_90,w_540/v1/assets/images/23029834/2023/
9/18/96c015ae-1090-4036-954b-d9c80085b1d71695022844653-HRX-by-Hrithik-Roshan-Me
n-Jackets-6981695022843934-1.jpg",
},
{
name: "Bottomwere",
categoryId: "men_bottomwear",
parentCategoryId: "men",

section: "SHOP_BY_CATEGORIES",
image:

"https://assets.myntassets.com/h_720,q_90,w_540/v1/assets/images/20122324/2022/
9/22/91c61c45-fe17-4d1d-8e20-0aaaf90186b61663827920015RaymondSlimFitBlueJeansFo
rMen1.jpg",
},
{
name: "Innerwere And Sleepwere",
categoryId: "men_innerwear_and_sleepwear",
parentCategoryId: "men",

section: "SHOP_BY_CATEGORIES",
image:

"https://assets.myntassets.com/h_720,q_90,w_540/v1/assets/images/24528350/2023/
8/17/d9ee03c9-7e15-49e4-8f15-0b6f38e568121692275415567TrackPants1.jpg",
},
{
name: "Footwere",
categoryId: "men_footwear",
```

```
parentCategoryId: "men",

section: "SHOP_BY_CATEGORIES",
image:

"https://assets.myntassets.com/h_720,q_90,w_540/v1/assets/images/27107914/2024/1/30/bd09f650-d8a4-4570-8e53-e2cfafc4ea6c1706609911015SparxMenMeshRunningShoes1.jpg",
},
{
 name: "Bed Linen & Furnishing",
 categoryId: "bed_linen_furnishing",

 section: "SHOP_BY_CATEGORIES",
 image:

"https://assets.myntassets.com/h_720,q_90,w_540/v1/assets/images/19284508/2022/7/28/92df52de-27dc-4d72-8ab4-fee2c82c85081659003977664DreamscapeUnisexPinkBedsheets1.jpg",
},
{
 name: "Flooring",
 categoryId: "flooring",

 section: "SHOP_BY_CATEGORIES",
 image:

"https://assets.myntassets.com/h_720,q_90,w_540/v1/assets/images/27868878/2024/2/25/53b7c07f-24a7-48e2-a791-f63dce4a0c981708844793060SANACARPETMulticolouredFloralAnti-SkidPolyesterCarpet1.jpg",
},
{
 name: "Bath",
 categoryId: "bath",

 section: "SHOP_BY_CATEGORIES",
 image:

"https://assets.myntassets.com/h_720,q_90,w_540/v1/assets/images/16719378/2024/7/2/97418bad-8216-494f-a863-9b159aec93cf1719897710069JockeyPackof2CottonTerry500GSMULtrasoftandDurableSolidHandTo1.jpg",
},
{
 name: "Lamps & Lighting",
 categoryId: "lamps_lighting",
 section: "SHOP_BY_CATEGORIES",
 image:
```

```
"https://assets.myntassets.com/h_720,q_90,w_540/v1/assets/images/productimage/2021/6/10/5be80654-037c-4591-99c9-a403646971981623344562713-1.jpg",
},
{
 name: "Men T Shirts",
 categoryId: "men_t_shirts",
 section: "DEALS",
 image:
"https://assets.myntassets.com/h_1440,q_90,w_1080/v1/assets/images/2024/AUGUST/26/zLmM5KEB_e4781ac0cc0945119e653f4130994bd2.jpg",
},
{
 name: "Women Skirts",
 categoryId: "women_skirts_palazzos",
 section: "DEALS",
 image:
"https://rukminim2.flixcart.com/image/612/612/xif0q/skirt/k/o/h/free-1-00003-customer-fashion-original-imagztykafwbqupf.jpeg?q=70",
},
{
 name: "Men Formal Shirts",
 categoryId: "men_formal_shirts",
 section: "DEALS",
 image:
"https://rukminim2.flixcart.com/image/612/612/xif0q/shirt/p/b/f/xl-new-fashion-cotton-shirts-for-men-solstice-original-imah3cqbg7cak4nyp.jpeg?q=70",
},
{
 name: "Women Saree",
 categoryId: "women_sarees",
 section: "DEALS",
 image:
"https://rukminim2.flixcart.com/image/832/832/xif0q/sari/z/a/4/free-saree-new2023-simple-light-weight-saree-chiffon-print-saree-original-imagzyhp2azy7rwz.jpeg?q=70&crop=false",
},
{
 name: "Smart Watches",
 categoryId: "smart_watches",
 section: "DEALS",
 image:
"https://rukminim2.flixcart.com/image/416/416/xif0q/smartwatch/u/r/h/-original-imagyxxyfgz52vzqn.jpeg?q=70&crop=false",
},
{
 name: "Men Festive Wear",
 categoryId: "men_indian_and_festive_wear",

```

```

 section: "DEALS",
 image:
"https://rukminim2.flixcart.com/image/612/612/xif0q/ethnic-set/o/n/o/l-02-langi
t-original-imah4jwwctkjtman.jpeg?q=70",
},
];
/* eslint-disable @typescript-eslint/no-explicit-any */
import { createSlice, createAsyncThunk, type PayloadAction } from
"@reduxjs/toolkit";
import type { Coupon, CouponState } from "../../types/couponTypes";
import { api } from "../../Config/Api";

const API_URL = "/api/coupons";

// Async thunks

export const createCoupon = createAsyncThunk<
Coupon,
{ coupon: any; jwt: string },
{ rejectValue: string }
>("coupon/createCoupon", async ({ coupon, jwt }, { rejectWithValue }) => {
try {
 const response = await api.post(`/${API_URL}/admin/create`, coupon, {
 headers: { Authorization: `Bearer ${jwt}` },
 });
 console.log(" created coupon ", response.data)
 return response.data;
} catch (error: any) {
 return rejectWithValue(error.response?.data || "Failed to create coupon");
}
});

export const deleteCoupon = createAsyncThunk<
string,
{ id: number; jwt: string },
{ rejectValue: string }
>("coupon/deleteCoupon", async ({ id, jwt }, { rejectWithValue }) => {
try {
 const response = await api.delete(`/${API_URL}/admin/delete/${id}`, {
 headers: { Authorization: `Bearer ${jwt}` },
 });
 return response.data;
} catch (error: any) {
 return rejectWithValue(error.response?.data || "Failed to delete coupon");
}
});

export const fetchAllCoupons = createAsyncThunk<
Coupon[],

```

```

string,
{ rejectWithValue: string }
>("coupon/fetchAllCoupons", async (jwt, { rejectWithValue }) => {
try {
 const response = await api.get(`${API_URL}/admin/all`, {
 headers: { Authorization: `Bearer ${jwt}` },
 });
 console.log("all coupons ", response.data)
 return response.data;
} catch (error: any) {
 return rejectWithValue(error.response?.data || "Failed to fetch coupons");
}
});

// Initial state
const initialState: CouponState = {
 coupons: [],
 cart: null,
 loading: false,
 error: null,
 couponCreated: false,
 couponApplied: false,
};

// Slice
const couponSlice = createSlice({
 name: "coupon",
 initialState,
 reducers: {},
 extraReducers: (builder) => {
 builder
 .addCase(createCoupon.pending, (state) => {
 state.loading = true;
 state.error = null;
 state.couponCreated = false;
 })
 .addCase(
 createCoupon.fulfilled,
 (state, action: PayloadAction<Coupon>) => {
 state.loading = false;
 state.coupons.push(action.payload);
 state.couponCreated = true;
 }
)
 .addCase(
 createCoupon.rejected,
 (state, action: PayloadAction<string | undefined>) => {
 state.loading = false;
 state.error = action.payload || "Failed to create coupon";
 }
);
 }
});

```

```

 state.couponCreated = false;
 }
)
.addCase(deleteCoupon.pending, (state) => {
 state.loading = true;
 state.error = null;
})
.addCase(deleteCoupon.fulfilled, (state, action) => {
 state.loading = false;
 state.coupons = state.coupons.filter(
 (coupon) => coupon.id !== parseInt(action.meta.arg.id.toString())
);
})
.addCase(
 deleteCoupon.rejected,
 (state, action: PayloadAction<string | undefined>) => {
 state.loading = false;
 state.error = action.payload || "Failed to delete coupon";
 }
)
.addCase(fetchAllCoupons.pending, (state) => {
 state.loading = true;
 state.error = null;
 state.couponCreated = false;
})
.addCase(
 fetchAllCoupons.fulfilled,
 (state, action: PayloadAction<Coupon[]>) => {
 state.loading = false;
 state.coupons = action.payload;
 }
)
.addCase(
 fetchAllCoupons.rejected,
 (state, action: PayloadAction<string | undefined>) => {
 state.loading = false;
 state.error = action.payload || "Failed to fetch coupons";
 }
);
},
);

export default couponSlice.reducer;
import { createAsyncThunk, createSlice} from '@reduxjs/toolkit';
import type { HomeCategory } from '../../../../../types/homeDataTypes';
import { api } from '../../../../../Config/Api';

const API_URL = '/admin';

```

```

export const updateHomeCategory = createAsyncThunk<HomeCategory, { id: number;
data: HomeCategory }>(
 'homeCategory/updateHomeCategory',
 async ({ id, data }, { rejectWithValue }) => {
 try {
 const response = await api.patch(`/${API_URL}/home-category/${id}`, data);
 console.log("category updated ", response);
 return response.data;
 } catch (error: any) {
 console.log("error ", error);
 if (error.response && error.response.data) {

 return rejectWithValue(error.response.data); // Return error response
 data if available
 } else {
 return rejectWithValue('An error occurred while updating the
category.');
 }
 }
 }
);

export const fetchHomeCategories = createAsyncThunk<HomeCategory[]>(
 'homeCategory/fetchHomeCategories',
 async (_, { rejectWithValue }) => {
 try {
 const response = await api.get(`${API_URL}/home-category`);
 console.log(" categories ", response.data);
 return response.data;
 } catch (error: any) {
 console.log("error ", error.response);
 return rejectWithValue(error.response?.data?.message || 'Failed to fetch
categories');
 }
 }
);

interface HomeCategoryState {
 categories: HomeCategory[];
 loading: boolean;
 error: string | null;
 categoryUpdated: boolean;
}

const initialState: HomeCategoryState = {
 categories: [],
 loading: false,
 error: null,
 categoryUpdated: false,
}

```

```
};

// Create the slice
const homeCategorySlice = createSlice({
 name: 'homeCategory',
 initialState,
 reducers: {},
 extraReducers: (builder) => {
 // Handle the pending state for updateHomeCategory
 builder.addCase(updateHomeCategory.pending, (state) => {
 state.loading = true;
 state.error = null;
 state.categoryUpdated = false;
 });

 // Handle the fulfilled state for updateHomeCategory
 builder.addCase(updateHomeCategory.fulfilled, (state, action) => {
 state.loading = false;
 state.categoryUpdated = true; // Set categoryUpdated flag to true
 // Find the category by ID and update it in the state
 const index = state.categories.findIndex((category) => category.id === action.payload.id);
 if (index !== -1) {
 state.categories[index] = action.payload;
 } else {
 state.categories.push(action.payload); // If the category doesn't exist, add it
 }
 });

 // Handle the rejected state for updateHomeCategory
 builder.addCase(updateHomeCategory.rejected, (state, action) => {
 state.loading = false;
 state.error = action.payload as string;
 });
 }
});

// fetch home category
builder.addCase(fetchHomeCategories.pending, (state) => {
 state.loading = true;
 state.error = null;
 state.categoryUpdated = false; // Reset categoryUpdated flag to false
})
.addCase(fetchHomeCategories.fulfilled, (state, action) => {
 state.loading = false;
 state.categories = action.payload;
})
.addCase(fetchHomeCategories.rejected, (state, action) => {
 state.loading = false;
 state.error = action.payload as string;
})
```

```
 });
 },
});

// Export the reducer to be used in the store
export default homeCategorySlice.reducer;
// dealSlice.ts

import { createSlice, createAsyncThunk, type PayloadAction } from
"@reduxjs/toolkit";
import type { ApiResponse, Deal, DealsState } from "../../types/dealTypes";
import { api } from "../../Config/Api";
// Define the initial state
const initialState: DealsState = {
 deals: [],
 loading: false,
 error: null,
 dealCreated:false,
 dealUpdated:false,
};

export const createDeal = createAsyncThunk(
 "deals/createDeal",
 async (deal: any, { rejectWithValue }) => {
 try {
 const response = await api.post("/admin/deals", deal, {
 headers: {
 "Content-Type": "application/json",
 Authorization: `Bearer ${localStorage.getItem("jwt")}`,
 },
 });
 console.log("created deal", response.data);
 return response.data;
 } catch (error: any) {
 console.log("error ", error.response);
 return rejectWithValue(
 error.response?.data?.message || "Failed to create deal"
);
 }
 }
);

export const getAllDeals = createAsyncThunk(
 "deals/getAllDeals",
 async (_, { rejectWithValue }) => {
 try {
 const response = await api.get("/admin/deals", {
 headers: {
 "Content-Type": "application/json",
 },
 });
 return response.data;
 } catch (error: any) {
 console.log("error ", error.response);
 return rejectWithValue(
 error.response?.data?.message || "Failed to get deals"
);
 }
 }
);
```

```

 Authorization: `Bearer ${localStorage.getItem("jwt")}`,
 },
});
console.log("get all deal", response.data);
return response.data;
} catch (error: any) {
console.log("error ", error.response);
return rejectWithValue(
 error.response?.data?.message || "Failed to create deal"
);
}
}

// Create async thunk for deleting a deal
export const deleteDeal = createAsyncThunk<ApiResponse, number>(
 "deals/deleteDeal",
async (id: number, { rejectWithValue }) => {
try {
 const response = await api.delete(`admin/deals/${id}`, {
 headers: {
 "Content-Type": "application/json",
 Authorization: `Bearer ${localStorage.getItem("jwt")}`,
 },
 });
 return response.data;
} catch (error: any) {
 console.log("error ", error.response);
 return rejectWithValue(
 error.response?.data?.message || "Failed to delete deal"
);
}
}
);

export const updateDeal = createAsyncThunk<Deal, { id: number; deal: any }>(
 "deals/updateDeal",
async ({ id, deal }, { rejectWithValue }) => {
try {
 const response = await api.patch(`admin/deals/${id}`, deal, {
 headers: {
 "Content-Type": "application/json",
 Authorization: `Bearer ${localStorage.getItem("jwt")}`,
 },
 });
 console.log("updated deal", response.data);
 return response.data;
} catch (error: any) {
 console.log("error ", error.response);
}
}
);

```

```
 return rejectWithValue(
 error.response?.data?.message || "Failed to update deal"
);
 }
);
);

// Create the slice
const dealSlice = createSlice({
 name: "deals",
 initialState,
 reducers: {},
 extraReducers: (builder) => {
 builder
 .addCase(getAllDeals.pending, (state) => {
 state.loading = true;
 state.error = null;
 state.dealCreated=false;
 state.dealUpdated=false;
 })
 .addCase(getAllDeals.fulfilled, (state, action) => {
 state.loading = false;
 state.deals=action.payload;
 })
 .addCase(getAllDeals.rejected, (state, action) => {
 state.loading = false;
 state.error = action.payload as string;
 })
 .addCase(createDeal.pending, (state) => {
 state.loading = true;
 state.error = null;
 state.dealCreated=false;
 })
 .addCase(createDeal.fulfilled, (state, action: PayloadAction<Deal>) => {
 state.loading = false;
 state.deals.push(action.payload);
 state.dealCreated=true;
 })
 .addCase(createDeal.rejected, (state, action) => {
 state.loading = false;
 state.error = action.payload as string;
 })
 .addCase(deleteDeal.pending, (state) => {
 state.loading = true;
 state.error = null;
 })
 .addCase(deleteDeal.fulfilled, (state, action) => {
 state.loading = false;
 if (action.payload.status) {
```

```

 state.deals = state.deals.filter(
 (deal) => deal.id !== action.meta.arg
);
 }
})
.addCase(deleteDeal.rejected, (state, action) => {
 state.loading = false;
 state.error = action.payload as string;
})
.addCase(updateDeal.pending, (state) => {
 state.loading = true;
 state.error = null;
 state.dealUpdated=false;
})
.addCase(updateDeal.fulfilled, (state, action: PayloadAction<Deal>) => {
 state.loading = false;
 state.dealUpdated=true;
 const index = state.deals.findIndex((deal) => deal.id ===
action.payload.id);
 if (index !== -1) {
 state.deals[index] = action.payload;
 }
})
.addCase(updateDeal.rejected, (state, action) => {
 state.loading = false;
 state.error = action.payload as string;
});
},
);

export default dealSlice.reducer;
import { createAsyncThunk } from '@reduxjs/toolkit';
import axios from 'axios';
import { api } from '../../../../../Config/Api';
import type { HomeCategory, HomeData } from '../../../../../types/homeDataTypes';

// Async thunk to fetch home page data with try-catch for error handling
export const fetchHomePageData = createAsyncThunk<HomeData>(
 'home/fetchHomePageData',
 async (_, { rejectWithValue }) => {
 try {
 const response = await api.get('/home-page');
 console.log("home page ",response.data)
 return response.data;
 } catch (error: any) {
 // Handle the error and return it to be used in rejected action
 const errorMessage = error.response?.data?.message || error.message ||
'Failed to fetch home page data';
 console.log("errr ",errorMessage,error)
 }
 }
);

```

```

 return rejectWithValue(errorMessage);
 }
}
);

export const createHomeCategories = createAsyncThunk<HomeData, HomeCategory[]>(
 'home/createHomeCategories',
 async (homeCategories, { rejectWithValue }) => {
 try {
 const response = await api.post('/home/categories', homeCategories);
 console.log("home categories ", response.data);
 return response.data;
 } catch (error: any) {
 // Handle the error and return it to be used in rejected action
 const errorMessage = error.response?.data?.message || error.message || 'Failed to create home categories';
 console.log("errr ", errorMessage, error);
 return rejectWithValue(errorMessage);
 }
 }
);
// homeSlice.ts
import { createSlice, type PayloadAction } from '@reduxjs/toolkit';
import { createHomeCategories, fetchHomePageData } from './AsyncThunk';
import type { HomeCategory, HomeData } from '../../../../../types/homeDataTypes';

interface HomeState {
 HomePageData: HomeData | null;
 homeCategories: HomeCategory[];
 loading: boolean;
 error: string | null;
}

const initialState: HomeState = {
 HomePageData: null,
 homeCategories: [],
 loading: false,
 error: null,
};

const homeSlice = createSlice({
 name: 'home',
 initialState,
 reducers: {},
 extraReducers: (builder) => {
 // Handle fetchHomePageData lifecycle
 builder.addCase(fetchHomePageData.pending, (state) => {
 state.loading = true;
 state.error = null;
 })
 }
});

```

```

 });

 builder.addCase(fetchHomePageData.fulfilled, (state, action: PayloadAction<HomeData>) => {
 state.loading = false;
 state.homePageData = action.payload;
 });
 builder.addCase(fetchHomePageData.rejected, (state, action) => {
 state.loading = false;
 state.error = action.error.message || 'Failed to load home page data';
 });

 // Handle createHomeCategories lifecycle
 builder.addCase(createHomeCategories.pending, (state) => {
 state.loading = true;
 state.error = null;
 });
 builder.addCase(createHomeCategories.fulfilled, (state, action) => {
 state.loading = false;
 state.homePageData = action.payload;
 });
 builder.addCase(createHomeCategories.rejected, (state, action) => {
 state.loading = false;
 state.error = action.error.message || 'Failed to create home categories';
 });
},
);

export default homeSlice.reducer;
import { createSlice, createAsyncThunk } from "@reduxjs/toolkit";
import { api } from "../../Config/Api";

// Define the initial state using an interface
interface AiChatBotState {
 response: string | null;
 loading: boolean;
 error: string | null;
 messages: any[]
}

const initialState: AiChatBotState = {
 response: null,
 loading: false,
 error: null,
 messages: []
};

// Define the async thunk for sending the message to the chatbot
export const chatBot = createAsyncThunk<
 any,

```

```

{ prompt: any; productId: number | null | undefined; userId: number | null }
>(
 "aiChatBot/generateResponse",
 async ({ prompt, productId, userId }, { rejectWithValue }) => {
 try {
 const response = await api.post("/ai/chat", prompt, {
 headers: {
 "Content-Type": "application/json",
 "Authorization": `Bearer ${localStorage.getItem("jwt")}`
 },
 params: {
 userId,
 productId,
 },
 });
 console.log("response ", productId, response.data);
 return response.data;
 } catch (error: any) {
 console.log("error ", error.response);
 return rejectWithValue(
 error.response?.data?.message || "Failed to generate chatbot response"
);
 }
 }
);

// Create the slice
const aiChatBotSlice = createSlice({
 name: "aiChatBot",
 initialState,
 reducers: {},
 extraReducers: (builder) => {
 builder
 .addCase(chatBot.pending, (state, action) => {
 state.loading = true;
 state.error = null;
 const { prompt } = action.meta.arg;

 // You can log or use the data here
 // console.log('Pending request:', { prompt, productId, userId });
 const userPrompt={message:prompt.prompt,role:"user"}
 state.messages=[...state.messages,userPrompt]
 })
 .addCase(chatBot.fulfilled, (state, action) => {
 state.loading = false;
 state.response = action.payload;
 state.messages=[...state.messages,action.payload]
 })
 .addCase(chatBot.rejected, (state, action) => {

```

```

 state.loading = false;
 state.error = action.payload as string;
 });
},
);

// Export the reducer
export default aiChatBotSlice.reducer;
// src/slices/authSlice.ts
import { createSlice, createAsyncThunk, type PayloadAction } from
'@reduxjs/toolkit';
import axios from 'axios';
import { api } from '../../../../../Config/Api';
import type {
 AuthResponse,
 LoginRequest,
 SignupRequest,
 ResetPasswordRequest,
 ApiResponse,
 AuthState,
} from '../../../../../types/authTypes';
import type { RootState } from '../Store';
import { resetUserState } from './UserSlice';
import { resetCartState } from './CartSlice';

const initialState: AuthState = {
 jwt: null,
 role: null,
 loading: false,
 error: null,
 otpSent:false
};

// Define the base URL for the API
const API_URL = '/auth';

export const sendLoginSignupOtp = createAsyncThunk<ApiResponse, { email: string }>(
 'auth/sendLoginSignupOtp',
 async ({ email }, { rejectWithValue }) => {
 try {
 const response = await api.post(`/${API_URL}/sent/login-signup-otp`, { email });
 console.log("otp sent successfully",response.data);
 return response.data;
 } catch (error:any) {
 console.log("error",error.response)
 }
 }
);

```

```
 return rejectWithValue(error.response.data.error || 'Failed to send
OTP');
 }
}
);

export const signup = createAsyncThunk<AuthResponse, SignupRequest>(
 'auth/signup',
 async (signupRequest, { rejectWithValue }) => {
 console.log("signup ", signupRequest)
 try {

 const response = await api.post<AuthResponse>(` ${API_URL}/signup`,
signupRequest);
 signupRequest.navigate("/")
 localStorage.setItem("jwt",response.data.jwt)
 return response.data;
 } catch (error:any) {
 return rejectWithValue('Signup failed');
 }
 }
);

export const signin = createAsyncThunk<AuthResponse, LoginRequest>(
 'auth/signin',
 async (loginRequest, { rejectWithValue }) => {
 try {
 const response = await api.post<AuthResponse>(` ${API_URL}/signin`,
loginRequest);
 console.log("login successful", response.data)
 localStorage.setItem("jwt",response.data.jwt)
 loginRequest.navigate("/");
 return response.data;
 } catch (error:any) {
 console.log("error ", error.response)
 return rejectWithValue('Signin failed');
 }
 }
);

export const resetPassword = createAsyncThunk<ApiResponse,
ResetPasswordRequest>(
 'auth/resetPassword',
 async (resetPasswordRequest, { rejectWithValue }) => {
 try {
 const response = await
api.post<ApiResponse>(` ${API_URL}/reset-password`, resetPasswordRequest);
 return response.data;
 } catch (error:any) {
```

```

 return rejectWithValue('Reset password failed');
 }
}
);

export const resetPasswordRequest = createAsyncThunk<ApiResponse, { email:
string }>(
 'auth/resetPasswordRequest',
 async ({ email }, { rejectWithValue }) => {
 try {
 const response = await
api.post<ApiResponse>(` ${API_URL}/reset-password-request`, { email });
 return response.data;
 } catch (error:any) {
 return rejectWithValue('Reset password request failed');
 }
 }
);
}

const authSlice = createSlice({
 name: 'auth',
 initialState,
 reducers: {
 logout: (state) => {
 state.jwt = null;
 state.role = null;
 localStorage.clear()
 },
 },
 extraReducers: (builder) => {
 builder
 .addCase(sendLoginSignupOtp.pending, (state) => {
 state.loading = true;
 state.error = null;
 })
 .addCase(sendLoginSignupOtp.fulfilled, (state) => {
 state.loading = false;
 state.otpSent = true;
 })
 .addCase(sendLoginSignupOtp.rejected, (state, action) => {
 state.loading = false;
 state.error = action.payload as string;
 })
 .addCase(signup.pending, (state) => {
 state.loading = true;
 state.error = null;
 })
 .addCase(signup.fulfilled, (state, action:
PayloadAction<AuthResponse>) => {

```

```
 state.jwt = action.payload.jwt;
 state.role = action.payload.role;
 state.loading = false;
 })
.addCase(signup.rejected, (state, action) => {
 state.loading = false;
 state.error = action.payload as string;
})
.addCase(signin.pending, (state) => {
 state.loading = true;
 state.error = null;
})
.addCase(signin.fulfilled, (state, action:
PayloadAction<AuthResponse>) => {
 state.jwt = action.payload.jwt;
 state.role = action.payload.role;
 state.loading = false;
})
.addCase(signin.rejected, (state, action) => {
 state.loading = false;
 state.error = action.payload as string;
})
.addCase(resetPassword.pending, (state) => {
 state.loading = true;
 state.error = null;
})
.addCase(resetPassword.fulfilled, (state) => {
 state.loading = false;
})
.addCase(resetPassword.rejected, (state, action) => {
 state.loading = false;
 state.error = action.payload as string;
})
.addCase(resetPasswordRequest.pending, (state) => {
 state.loading = true;
 state.error = null;
})
.addCase(resetPasswordRequest.fulfilled, (state) => {
 state.loading = false;
})
.addCase(resetPasswordRequest.rejected, (state, action) => {
 state.loading = false;
 state.error = action.payload as string;
}),
),
);
export const { logout } = authSlice.actions;
```

```
export default authSlice.reducer;

export const performLogout = () => async (dispatch: any) => {
 dispatch(logout());
 dispatch(resetUserState());
 dispatch(resetCartState());
};

export const selectAuth = (state: RootState) => state.auth;
export const selectAuthLoading = (state: RootState) => state.auth.loading;
export const selectAuthError = (state: RootState) => state.auth.error;
// src/slices/cartSlice.ts
import { createSlice, createAsyncThunk, type PayloadAction } from
"@reduxjs/toolkit";
import { api } from "../../Config/Api";
import type { Cart, CartItem } from "../../types/cartTypes";
import type { RootState } from "../Store";
import { applyCoupon } from "./CouponSlice";
import { sumCartItemMrpPrice, sumCartItemSellingPrice } from
"../../util/cartCalculator";

interface CartState {
 cart: Cart | null;
 loading: boolean;
 error: string | null;
}

const initialState: CartState = {
 cart: null,
 loading: false,
 error: null,
};

// Define the base URL for the API
const API_URL = "/api/cart";

export const fetchUserCart = createAsyncThunk<Cart, string>(
 "cart/fetchUserCart",
 async (jwt: string, { rejectWithValue }) => {
 try {
 const response = await api.get(API_URL, {
 headers: {
 Authorization: `Bearer ${jwt}`,
 },
 });
 console.log("Cart fetched ", response.data);
 return response.data;
 } catch (error) {
 return rejectWithValue(error.message);
 }
 },
);
```

```

 } catch (error: any) {
 console.log("error ", error.response);
 return rejectWithValue("Failed to fetch user cart");
 }
 }
);

interface AddItemRequest {
 productId: number | undefined;
 size: string;
 quantity: number;
}

export const addItemToCart = createAsyncThunk<
 CartItem,
 { jwt: string | null; request: AddItemRequest }
>("cart/addItemToCart", async ({ jwt, request }, { rejectWithValue }) => {
 try {
 const response = await api.put(`/${API_URL}/add`, request, {
 headers: {
 Authorization: `Bearer ${jwt}`,
 },
 });

 console.log("Cart added ", response.data);
 return response.data;
 } catch (error: any) {
 console.log("error ", error.response);
 return rejectWithValue("Failed to add item to cart");
 }
});

export const deleteCartItem = createAsyncThunk<
 any,
 { jwt: string; cartItemId: number }
>("cart/deleteCartItem", async ({ jwt, cartItemId }, { rejectWithValue }) => {
 try {
 const response = await api.delete(`/${API_URL}/item/${cartItemId}`, {
 headers: { Authorization: `Bearer ${jwt}` },
 });
 return response.data;
 } catch (error: any) {
 return rejectWithValue(
 error.response.data.message || "Failed to delete cart item"
);
 }
});

export const updateCartItem = createAsyncThunk<

```

```

any,
{ jwt: string | null; cartItemId: number; cartItem: any }
>(
 "cart/updateCartItem",
 async ({ jwt, cartItemId, cartItem }, { rejectWithValue }) => {
 try {
 const response = await api.put(
 `${API_URL}/item/${cartItemId}`,
 cartItem,
 {
 headers: { Authorization: `Bearer ${jwt}` },
 }
);
 return response.data;
 } catch (error: any) {
 return rejectWithValue(
 error.response.data.message || "Failed to update cart item"
);
 }
 }
);

const cartSlice = createSlice({
 name: "cart",
 initialState,
 reducers: {
 resetCartState: (state) => {
 state.cart = null;
 state.loading = false;
 state.error = null;
 },
 },
 extraReducers: (builder) => {
 builder
 .addCase(fetchUserCart.pending, (state) => {
 state.loading = true;
 state.error = null;
 })
 .addCase(
 fetchUserCart.fulfilled,
 (state, action: PayloadAction<Cart>) => {
 state.cart = action.payload;
 state.loading = false;
 }
)
 .addCase(fetchUserCart.rejected, (state, action) => {
 state.loading = false;
 state.error = action.payload as string;
 })
 }
});

```

```

.addCase(addItemToCart.pending, (state) => {
 state.loading = true;
 state.error = null;
})
.addCase(
 addItemToCart.fulfilled,
 (state, action: PayloadAction<CartItem>) => {
 if (state.cart) {
 state.cart.cartItems.push(action.payload);
 }
 state.loading = false;
 }
)
.addCase(addItemToCart.rejected, (state, action) => {
 state.loading = false;
 state.error = action.payload as string;
})

// cart item
.addCase(deleteCartItem.pending, (state) => {
 state.loading = true;
 state.error = null;
})
.addCase(deleteCartItem.fulfilled, (state, action) => {
 if (state.cart) {
 state.cart.cartItems = state.cart.cartItems.filter(
 (item:CartItem) => item.id !== action.meta.arg.cartItemId
);
 const mrpPrice=sumCartItemMrpPrice(state.cart?.cartItems || [])
 const sellingPrice=sumCartItemSellingPrice(state.cart?.cartItems || []
[])
 state.cart.totalSellingPrice=sellingPrice;
 state.cart.totalMrpPrice=mrpPrice;
 }
 state.loading = false;
})
.addCase(deleteCartItem.rejected, (state, action) => {
 state.loading = false;
 state.error = action.payload as string;
})
.addCase(updateCartItem.pending, (state) => {
 state.loading = true;
 state.error = null;
})
.addCase(updateCartItem.fulfilled, (state, action) => {
 if (state.cart) {
 const index = state.cart.cartItems.findIndex(
 (item:CartItem) => item.id === action.meta.arg.cartItemId
)
 state.cart.cartItems[index] = action.payload;
 }
 state.loading = false;
})

```

```

);
 if (index !== -1) {
 state.cart.cartItems[index] = {
 ...state.cart.cartItems[index],
 ...action.payload,
 };
 }
 const mrpPrice=sumCartItemMrpPrice(state.cart?.cartItems || [])
 const sellingPrice=sumCartItemSellingPrice(state.cart?.cartItems || [])
 state.cart.totalSellingPrice=sellingPrice;
 state.cart.totalMrpPrice=mrpPrice;
 }
 state.loading = false;
}
.addCase(updateCartItem.rejected, (state, action) => {
 state.loading = false;
 state.error = action.payload as string;
})
.addCase(applyCoupon.fulfilled, (state, action) => {
 state.loading = false;
 state.cart = action.payload;
});

),
);

export default cartSlice.reducer;
export const { resetCartState } = cartSlice.actions;

export const selectCart = (state: RootState) => state.cart.cart;
export const selectCartLoading = (state: RootState) => state.cart.loading;
export const selectCartError = (state: RootState) => state.cart.error;
import { createSlice, createAsyncThunk, type PayloadAction } from
"@reduxjs/toolkit";
import axios from "axios";
import { api } from "../../Config/Api";
import type { Cart } from "../../types/cartTypes";
import type { Coupon, CouponState } from "../../types/couponTypes";

const API_URL = "/api/coupons";

// Async thunks
export const applyCoupon = createAsyncThunk<
 Cart,
 {
 apply: string;
 code: string;
 orderValue: number;
 }
>;

```

```

 jwt: string;
 },
 { rejectValue: string }
>(
"coupon/applyCoupon",
async ({ apply, code, orderValue, jwt }, { rejectWithValue }) => {
 try {
 const response = await api.post(`$API_URL}/apply`, null, {
 params: { apply, code, orderValue },
 headers: { Authorization: `Bearer ${jwt}` },
 });
 console.log("apply coupon ", response.data);
 return response.data;
 } catch (error: any) {
 console.log("error", error);
 return rejectWithValue(error.response?.data.error || "Failed to apply
coupon");
 }
}
);

// Initial state
const initialState: CouponState = {
 coupons: [],
 cart: null,
 loading: false,
 error: null,
 couponCreated: false,
 couponApplied: false,
};

// Slice
const couponSlice = createSlice({
 name: "coupon",
 initialState,
 reducers: {},
 extraReducers: (builder) => {
 builder
 .addCase(applyCoupon.pending, (state) => {
 state.loading = true;
 state.error = null;
 state.couponApplied=false;
 })
 .addCase(applyCoupon.fulfilled, (state, action) => {
 state.loading = false;
 state.cart = action.payload;

 if(action.meta.arg.apply=="true") {

```

```

 state.couponApplied=true
 }

})
.addCase(
 applyCoupon.rejected,
 (state, action: PayloadAction<string | undefined>) => {
 state.loading = false;
 state.error = action.payload || "Failed to apply coupon";
 state.couponApplied=false;
 }
);
},
);
};

export default couponSlice.reducer;
// src/slices/orderSlice.ts

import { createSlice, createAsyncThunk, type PayloadAction } from
"@reduxjs/toolkit";
import axios from "axios";
import { api } from "../../Config/Api";
import type { Order, OrderItem, OrderState } from "../../types/orderTypes";
import type { Address } from "../../types/userTypes";
import type { ApiResponse } from "../../types/authTypes";
import type { RootState } from "../Store";

const initialState: OrderState = {
 orders: [],
 orderItem:null,
 currentOrder: null,
 paymentOrder: null,
 loading: false,
 error: null,
 orderCanceled:false
};

const API_URL = "/api/orders";

// Fetch user order history
export const fetchUserOrderHistory = createAsyncThunk<Order[], string>(
 "orders/fetchUserOrderHistory",
 async (jwt, { rejectWithValue }) => {
 try {
 const response = await api.get<Order[]>(` ${API_URL}/user`, {
 headers: { Authorization: `Bearer ${jwt}` },
 });
 console.log("order history fetched ", response.data);
 return response.data;
 }
 }
);

```

```
 } catch (error: any) {
 console.log("error ", error.response);
 return rejectWithValue(
 error.response.data.error || "Failed to fetch order history"
);
 }
);
 }

// Fetch order by ID
export const fetchOrderById = createAsyncThunk<
 Order,
 { orderId: number; jwt: string }
>("orders/fetchOrderById", async ({ orderId, jwt }, { rejectWithValue }) => {
 try {
 const response = await api.get<Order>(`${API_URL}/${orderId}`, {
 headers: { Authorization: `Bearer ${jwt}` },
 });
 console.log("order fetched ", response.data);
 return response.data;
 } catch (error: any) {
 console.log("error ", error.response);
 return rejectWithValue("Failed to fetch order");
 }
});

// Create a new order
export const createOrder = createAsyncThunk<
 any,
 { address: Address; jwt: string, paymentGateway: string }
>("orders/createOrder", async ({ address, jwt, paymentGateway }, {
 rejectWithValue
}) => {
 try {
 const response = await api.post<any>(API_URL, address, {
 headers: { Authorization: `Bearer ${jwt}` },
 params:{paymentMethod:paymentGateway}
 });
 console.log("order created ", response.data);
 if(response.data.payment_link_url){
 window.location.href=response.data.payment_link_url
 }

 return response.data;
 } catch (error: any) {
 console.log("error ", error.response);
 return rejectWithValue("Failed to create order");
 }
});
```

```

export const fetchOrderItemById = createAsyncThunk<
 OrderItem,
 { orderItemId: number; jwt: string }
>("orders/fetchOrderItemById", async ({ orderItemId, jwt }, { rejectWithValue }) => {
 try {
 const response = await api.get<OrderItem>(`${
 API_URL
 }/item/${orderItemId}`),
 {
 headers: { Authorization: `Bearer ${jwt}` },
 };
 console.log("order item fetched ", response.data);
 return response.data;
 } catch (error: any) {
 console.log("error ", error.response);
 return rejectWithValue("Failed to create order");
 }
});

// payment success handler

export const paymentSuccess = createAsyncThunk<
 ApiResponse,
 { paymentId: string; jwt: string, paymentLinkId:string },
 { rejectValue: string }
>('orders/paymentSuccess', async ({ paymentId, jwt, paymentLinkId }, { rejectWithValue }) => {
 try {
 const response = await api.get(`api/payment/${paymentId}`, {
 headers: {
 Authorization: `Bearer ${jwt}`,
 },
 params:{paymentLinkId}
 });

 console.log("payment success ",response.data)

 return response.data;
 } catch (error: any) {
 console.log("error ",error.response)
 if (error.response) {
 return rejectWithValue(error.response.data.message);
 }
 return rejectWithValue('Failed to process payment');
 }
});

export const cancelOrder = createAsyncThunk<Order, any>(
 'orders/cancelOrder',

```

```
async (orderId, { rejectWithValue }) => {
 try {
 const response = await api.put(` ${API_URL} / ${orderId} / cancel `, {}, {
 headers: {
 Authorization: ` Bearer ${localStorage.getItem("jwt")}` ,
 },
 });
 console.log("cancel order ", response.data)
 return response.data;
 } catch (error: any) {
 console.log("error ", error.response)
 if (axios.isAxiosError(error) && error.response) {
 return rejectWithValue(error.response.data);
 }
 return rejectWithValue('An error occurred while cancelling the order.');
 }
}

const orderSlice = createSlice({
 name: "orders",
 initialState,
 reducers: {} ,
 extraReducers: (builder) => {
 builder
 // Fetch user order history
 .addCase(fetchUserOrderHistory.pending, (state) => {
 state.loading = true;
 state.error = null;
 state.orderCanceled = false;
 })
 .addCase(
 fetchUserOrderHistory.fulfilled,
 (state, action: PayloadAction<Order[]>) => {
 state.orders = action.payload;
 state.loading = false;
 }
)
 .addCase(fetchUserOrderHistory.rejected, (state, action) => {
 state.loading = false;
 state.error = action.payload as string;
 })

 // Fetch order by ID
 .addCase(fetchOrderById.pending, (state) => {
 state.loading = true;
 state.error = null;
 })
 .addCase(
```

```

fetchOrderById.fulfilled,
(state, action: PayloadAction<Order>) => {
 state.currentOrder = action.payload;
 state.loading = false;
}
)
.addCase(fetchOrderById.rejected, (state, action) => {
 state.loading = false;
 state.error = action.payload as string;
})

// Create a new order
.addCase(createOrder.pending, (state) => {
 state.loading = true;
 state.error = null;
})
.addCase(createOrder.fulfilled, (state, action: PayloadAction<any>) => {
 state.paymentOrder = action.payload;
 state.loading = false;
})
.addCase(createOrder.rejected, (state, action) => {
 state.loading = false;
 state.error = action.payload as string;
})

// Fetch Order Item by ID
.addCase(fetchOrderItemById.pending, (state) => {
 state.loading = true;
 state.error = null;
})
.addCase(fetchOrderItemById.fulfilled, (state, action) => {
 state.loading = false;
 state.orderItem = action.payload;
})
.addCase(fetchOrderItemById.rejected, (state, action) => {
 state.loading = false;
 state.error = action.payload as string;
})

// payment success handler
.addCase(paymentSuccess.pending, (state) => {
 state.loading = true;
 state.error = null;
})
.addCase(paymentSuccess.fulfilled, (state, action) => {
 state.loading = false;
 console.log('Payment successful:', action.payload);
})
.addCase(paymentSuccess.rejected, (state, action) => {

```

```

 state.loading = false;
 state.error = action.payload as string;
 })
.addCase(cancelOrder.pending, (state) => {
 state.loading = true;
 state.error = null;
 state.orderCanceled = false;
})
.addCase(cancelOrder.fulfilled, (state, action) => {
 state.loading = false;
 state.orders = state.orders.map((order) =>
 order.id === action.payload.id ? action.payload : order
);
 state.orderCanceled = true;
 state.currentOrder = action.payload
})
.addCase(cancelOrder.rejected, (state, action) => {
 state.loading = false;
 state.error = action.payload as string;
});
},
);

export default orderSlice.reducer;

export const selectOrders = (state: RootState) => state.orders.orders;
export const selectCurrentOrder = (state: RootState) =>
 state.orders.currentOrder;
export const selectPaymentOrder = (state: RootState) =>
 state.orders.paymentOrder;
export const selectOrdersLoading = (state: RootState) => state.orders.loading;
export const selectOrdersError = (state: RootState) => state.orders.error;
import { createSlice, createAsyncThunk, type PayloadAction } from
"@reduxjs/toolkit";
import axios from "axios";
import { api } from "../../Config/Api";
import type { Product } from "../../types/productTypes";
import type { RootState } from "../Store";

// Define the base URL for the API
const API_URL = "/products";

// Define the initial state type
interface ProductState {
 product: Product | null;
 products: Product[];
 paginatedProducts: any;
 totalPages: number;
 loading: boolean;
}

```

```

 error: string | null;
 searchProduct:Product[]
}

// Define the initial state
const initialState: ProductState = {
 product: null,
 products: [],
 paginatedProducts: null,
 totalPages:1,
 loading: false,
 error: null,
 searchProduct: []
};

// Create async thunks for API calls
export const fetchProductById = createAsyncThunk<Product, number>(
 "products/fetchProductById",
 async (productId, { rejectWithValue }) => {
 try {
 const response = await api.get<Product>(` ${API_URL} /${productId}`);
 console.log("product details ", response.data);
 return response.data;
 } catch (error: any) {
 console.log("error ", error.response);
 return rejectWithValue(error.response.data);
 }
 }
);

export const searchProduct = createAsyncThunk<Product[], string>(
 "products/searchProduct",
 async (query, { rejectWithValue }) => {
 try {
 const response = await api.get<Product[]>(` ${API_URL} /search`, {
 params: { query },
 });
 console.log("search products ", response.data)
 return response.data;
 } catch (error: any) {
 console.log("error ", error.response)
 return rejectWithValue(error.response.data);
 }
 }
);

export const getAllProducts = createAsyncThunk<
 any,
 {

```

```
category?: string;
brand?:string;
color?: string;
size?: string;
minPrice?: number;
maxPrice?: number;
minDiscount?: number;
sort?: string;
stock?: string;
pageNumber?: number;
}
>("products/getAllProducts", async (params, { rejectWithValue }) => {
try {
 const response = await api.get<any>(API_URL, {
 params: {
 ...params,
 pageNumber: params.pageNumber || 0,
 },
 });
 console.log("all products ", response.data);
 return response.data;
} catch (error: any) {
 console.log("error ", error.response);
 return rejectWithValue(error.response.data);
}
});

// Create the slice
const productSlice = createSlice({
 name: "products",
 initialState,
 reducers: {},
 extraReducers: (builder) => {
 builder
 .addCase(fetchProductById.pending, (state) => {
 state.loading = true;
 state.error = null;
 })
 .addCase(
 fetchProductById.fulfilled,
 (state, action: PayloadAction<Product>) => {
 state.product = action.payload;
 state.loading = false;
 }
)
 .addCase(fetchProductById.rejected, (state, action) => {
 state.loading = false;
 state.error = action.error.message || "Failed to fetch product";
 })

```

```

 .addCase(searchProduct.pending, (state) => {
 state.loading = true;
 state.error = null;
 })
 .addCase(
 searchProduct.fulfilled,
 (state, action: PayloadAction<Product[]>) => {
 state.searchProduct = action.payload;
 state.loading = false;
 }
)
 .addCase(searchProduct.rejected, (state, action) => {
 state.loading = false;
 state.error = action.error.message || "Failed to search products";
 })
 .addCase(getAllProducts.pending, (state) => {
 state.loading = true;
 state.error = null;
 })
 .addCase(
 getAllProducts.fulfilled,
 (state, action: PayloadAction<any>) => {
 state.paginatedProducts = action.payload;
 state.products = action.payload.content;
 state.totalPages=action.payload.totalPages
 state.loading = false;
 console.log("-----" , action.payload.totalPages)
 }
)
 .addCase(getAllProducts.rejected, (state, action) => {
 state.loading = false;
 state.error = action.error.message || "Failed to fetch products";
 });
 },
);
 });

export default productSlice.reducer;

// Define selector functions
export const selectProduct = (state: RootState) => state.products.product;
export const selectProducts = (state: RootState) => state.products.products;
export const selectPaginatedProducts = (state: RootState) =>
 state.products.paginatedProducts;
export const selectProductLoading = (state: RootState) =>
 state.products.loading;
export const selectProductError = (state: RootState) => state.products.error;
import { createSlice, createAsyncThunk, type PayloadAction } from
"@reduxjs/toolkit";
import axios from "axios";

```

```
import { api } from "../../Config/Api";
import type {
 ApiResponse,
 CreateReviewRequest,
 Review,
 ReviewState,
} from "../../types/reviewTypes";
import type { RootState } from "../Store";

const API_URL = "/api";

// Async thunks
export const fetchReviewsByProductId = createAsyncThunk<
 Review[],
 { productId: number },
 { rejectWithValue: string }
>(
 "review/fetchReviewsByProductId",
 async ({ productId }, { rejectWithValue }) => {
 try {
 const response = await api.get(
 `${API_URL}/products/${productId}/reviews`,
 {
 headers: {
 Authorization: `Bearer ${localStorage.getItem("jwt")}`,
 },
 }
);
 console.log("fetch all reviews for product ", response.data);
 return response.data;
 } catch (error: any) {
 console.log("error - ", error.response?.data);
 return rejectWithValue(error.response?.data || "Failed to fetch reviews");
 }
 }
);

export const createReview = createAsyncThunk<
 Review,
 { productId: number; review: CreateReviewRequest; jwt: string, navigate: any },
 { rejectWithValue: string }
>(
 "review/createReview",
 async ({ productId, review, jwt, navigate }, { rejectWithValue }) => {
 try {
 const response = await api.post(
 `${API_URL}/products/${productId}/reviews`,
 review,
 {

```

```

 headers: {
 Authorization: `Bearer ${jwt}`,
 },
 },
);
 navigate(`/${reviews}/${productId}`);
 console.log("create reviews for product ", response.data);
 return response.data;
} catch (error: any) {
 console.log("error ", error);
 return rejectWithValue(error.response?.data || "Failed to create review");
}
}

export const updateReview = createAsyncThunk<
 Review,
 { reviewId: number; review: CreateReviewRequest; jwt: string },
 { rejectValue: string }
>(
 "review/updateReview",
 async ({ reviewId, review, jwt }, { rejectWithValue }) => {
 try {
 const response = await api.patch(
 `${API_URL}/reviews/${reviewId}`,
 review,
 {
 headers: {
 Authorization: `Bearer ${jwt}`,
 },
 }
);
 console.log("updated reviews for product ", response.data);
 return response.data;
 } catch (error: any) {
 console.log("error ", error);
 return rejectWithValue(error.response?.data || "Failed to update review");
 }
 }
);

export const deleteReview = createAsyncThunk<
 ApiResponse,
 { reviewId: number; jwt: string },
 { rejectValue: string }
>("review/deleteReview", async ({ reviewId, jwt }, { rejectWithValue }) => {
 try {
 const response = await api.delete(`${API_URL}/reviews/${reviewId}`, {
 headers: {

```

```
 Authorization: `Bearer ${jwt}`,
 },
});
return response.data;
} catch (error: any) {
 console.log("error ", error);
 return rejectWithValue(error.response?.data || "Failed to delete review");
}
);

// Initial state
const initialState: ReviewState = {
 reviews: [],
 loading: false,
 error: null,
 reviewCreated: false,
 reviewUpdated: false,
 reviewDeleted: false,
};

// Slice
const reviewSlice = createSlice({
 name: "review",
 initialState,
 reducers: {
 resetReviewState: (state) => {
 state.reviews = [];
 state.loading = false;
 state.error = null;
 state.reviewCreated = false;
 state.reviewUpdated = false;
 state.reviewDeleted = false;
 },
 },
 extraReducers: (builder) => {
 builder
 .addCase(fetchReviewsByProductId.pending, (state) => {
 state.loading = true;
 state.error = null;
 })
 .addCase(
 fetchReviewsByProductId.fulfilled,
 (state, action: PayloadAction<Review[]>) => {
 state.reviews = action.payload;
 state.loading = false;
 }
)
 .addCase(fetchReviewsByProductId.rejected, (state, action) => {
 state.loading = false;
```

```
 state.error = action.payload as string;
 })
.addCase(createReview.pending, (state) => {
 state.loading = true;
 state.error = null;
 state.reviewCreated = false;
})
.addCase(
 createReview.fulfilled,
 (state, action: PayloadAction<Review>) => {
 state.reviews.push(action.payload);
 state.loading = false;
 state.reviewCreated = true;
 }
)
.addCase(createReview.rejected, (state, action) => {
 state.loading = false;
 state.error = action.payload as string;
 state.reviewCreated = false;
})
.addCase(updateReview.pending, (state) => {
 state.loading = true;
 state.error = null;
 state.reviewUpdated = false;
})
.addCase(
 updateReview.fulfilled,
 (state, action: PayloadAction<Review>) => {
 const index = state.reviews.findIndex(
 (r) => r.id === action.payload.id
);
 if (index !== -1) {
 state.reviews[index] = action.payload;
 }
 state.loading = false;
 state.reviewUpdated = true;
 }
)
.addCase(updateReview.rejected, (state, action) => {
 state.loading = false;
 state.error = action.payload as string;
 state.reviewUpdated = false;
})
.addCase(deleteReview.pending, (state) => {
 state.loading = true;
 state.error = null;
 state.reviewDeleted = false;
})
.addCase(deleteReview.fulfilled, (state, action) => {
```

```

 state.reviews = state.reviews.filter(
 (r) => r.id !== action.meta.arg.reviewId
);
 state.loading = false;
 state.reviewDeleted = true;
 })
 .addCase(deleteReview.rejected, (state, action) => {
 state.loading = false;
 state.error = action.payload as string;
 state.reviewDeleted = false;
 });
 },
);
}

export default reviewSlice.reducer;
export const { resetReviewState } = reviewSlice.actions;

// src/slices/userSlice.ts
import { createSlice, createAsyncThunk, type PayloadAction } from
"@reduxjs/toolkit";
import axios from "axios";
import { api } from "../../Config/Api";
import type { User, UserState } from "../../types/userTypes";
import type { RootState } from "../Store";

const initialState: UserState = {
 user: null,
 loading: false,
 error: null,
 profileUpdated: false,
};

// Define the base URL for the API
const API_URL = "/api/users";

export const fetchUserProfile = createAsyncThunk<
 User,
 { jwt: string; navigate: any }
>(
 "user/fetchUserProfile",
 async (
 { jwt, navigate }: { jwt: string; navigate: any },
 { rejectWithValue }
) => {
 try {
 const response = await api.get(`${API_URL}/profile`, {
 headers: { Authorization: `Bearer ${jwt}` },
 });
 console.log(" user profile ", response.data);
 }
 }
);

```

```
 if (response.data.role === "ROLE_ADMIN") {
 navigate("/admin");
 }
 return response.data;
 } catch (error: any) {
 console.log("error ", error.response);
 return rejectWithValue("Failed to fetch user profile");
 }
}

const userSlice = createSlice({
 name: "user",
 initialState,
 reducers: {
 resetUserState: (state) => {
 state.user = null;
 state.loading = false;
 state.error = null;
 state.profileUpdated = false;
 },
 },
 extraReducers: (builder) => {
 builder
 .addCase(fetchUserProfile.pending, (state) => {
 state.loading = true;
 state.error = null;
 })
 .addCase(
 fetchUserProfile.fulfilled,
 (state, action: PayloadAction<User>) => {
 state.user = action.payload;
 state.loading = false;
 }
)
 .addCase(fetchUserProfile.rejected, (state, action) => {
 state.loading = false;
 state.error = action.payload as string;
 });
 },
});

export const { resetUserState } = userSlice.actions;

export default userSlice.reducer;

export const selectUser = (state: RootState) => state.user.user;
export const selectUserLoading = (state: RootState) => state.user.loading;
export const selectUserError = (state: RootState) => state.user.error;
```

```
import { createSlice, createAsyncThunk, type PayloadAction } from
"@reduxjs/toolkit";
import { api } from "../../Config/Api";
import type { Wishlist, WishlistState } from "../../types/wishlistTypes";

const initialState: WishlistState = {
 wishlist: null,
 loading: false,
 error: null,
};

export const getWishlistByUserId = createAsyncThunk(
 "wishlist/getWishlistByUserId",
 async (_, { rejectWithValue }) => {
 try {
 const response = await api.get(`api/wishlist`, {
 headers: {
 Authorization: `Bearer ${localStorage.getItem("jwt")}`,
 },
 });
 console.log("wishlist fetch ", response.data);
 return response.data;
 } catch (error: any) {
 console.log("error ", error);
 return rejectWithValue(
 error.response?.data.message || "Failed to fetch wishlist"
);
 }
 }
);

export const addProductToWishlist = createAsyncThunk(
 "wishlist/addProductToWishlist",
 async (
 { productId }: { productId: number },
 { rejectWithValue }
) => {
 try {
 const response = await api.post(
 `/api/wishlist/add-product/${productId}`,
 {},
 {
 headers: {
 Authorization: `Bearer ${localStorage.getItem("jwt")}`,
 },
 }
);
 console.log(" add product ", response.data);
 return response.data;
 } catch (error: any) {
 console.log("error ", error);
 return rejectWithValue(error.message);
 }
 }
);
```

```
 } catch (error: any) {
 return rejectWithValue(
 error.response?.data.message || "Failed to add product to wishlist"
);
 }
);
 }

// Slice
const wishlistSlice = createSlice({
 name: "wishlist",
 initialState,
 reducers: {
 resetWishlistState: (state) => {
 state.wishlist = null;
 state.loading = false;
 state.error = null;
 },
 },
 extraReducers: (builder) => {
 // getWishlistByUserId
 builder.addCase(getWishlistByUserId.pending, (state) => {
 state.loading = true;
 state.error = null;
 });
 builder.addCase(
 getWishlistByUserId.fulfilled,
 (state, action: PayloadAction<Wishlist>) => {
 state.wishlist = action.payload;
 state.loading = false;
 }
);
 builder.addCase(
 getWishlistByUserId.rejected,
 (state, action: PayloadAction<any>) => {
 state.loading = false;
 state.error = action.payload;
 }
);
 };
}

// addProductToWishlist
builder.addCase(addProductToWishlist.pending, (state) => {
 state.loading = true;
 state.error = null;
});
builder.addCase(
 addProductToWishlist.fulfilled,
 (state, action: PayloadAction<Wishlist>) => {
 state.wishlist = action.payload;
 }
);
```

```

 state.loading = false;
 }
);
builder.addCase(
 addProductToWishlist.rejected,
 (state, action: PayloadAction<any>) => {
 state.loading = false;
 state.error = action.payload;
 }
);
},
))];

export const { resetWishlistState } = wishlistSlice.actions;

export default wishlistSlice.reducer;
import { createSlice, createAsyncThunk } from "@reduxjs/toolkit";
import axios from "axios";
import { api } from "../../Config/Api";
import type { Payouts } from "../../types/payoutsType";
import type { Transaction } from "../../types/Transaction";

interface PayoutsState {
 payouts: Payouts[];
 payout: Payouts | null;
 loading: boolean;
 error: string | null;
}

const initialState: PayoutsState = {
 payouts: [],
 payout: null,
 loading: false,
 error: null,
};

// Thunks
export const fetchPayoutsBySeller = createAsyncThunk<
 Payouts[],
 string,
 { rejectValue: string }
>("payouts/fetchPayoutsBySeller", async (jwt, { rejectWithValue }) => {
 try {
 const response = await api.get<Payouts[]>("/api/payouts/seller", {
 headers: {
 Authorization: `Bearer ${jwt}`,
 },
 });
 console.log("Payouts ", response.data)
 } catch (error) {
 rejectWithValue(error.message);
 }
});

```

```
 return response.data;
 } catch (error: any) {
 if (error.response) {
 return rejectWithValue(error.response.data.message);
 }
 return rejectWithValue("Failed to fetch payouts");
 }
}) ;

export const fetchPayoutById = createAsyncThunk<
 Payouts,
 number,
 { rejectValue: string }
>("payouts/fetchPayoutById", async (id, { rejectWithValue }) => {
 try {
 const response = await api.get<Payouts>(`/api/payouts/${id}`);
 return response.data;
 } catch (error: any) {
 if (error.response) {
 return rejectWithValue(error.response.data.message);
 }
 return rejectWithValue("Failed to fetch payout");
 }
}) ;

export const updatePayoutStatus = createAsyncThunk<
 Payouts,
 { id: number; status: string },
 { rejectValue: string }
>("payouts/updatePayoutStatus", async ({ id, status }, { rejectWithValue }) =>
{
 try {
 const response = await api.put<Payouts>(
 `/api/payouts/${id}/status`,
 null,
 {
 params: { status },
 }
);
 return response.data;
 } catch (error: any) {
 if (error.response) {
 return rejectWithValue(error.response.data.message);
 }
 return rejectWithValue("Failed to update payout status");
 }
}) ;

// Slice
```

```
const payoutsSlice = createSlice({
 name: "payouts",
 initialState,
 reducers: {},
 extraReducers: (builder) => {
 builder
 .addCase(fetchPayoutsBySeller.pending, (state) => {
 state.loading = true;
 state.error = null;
 })
 .addCase(fetchPayoutsBySeller.fulfilled, (state, action) => {
 state.loading = false;
 state.payouts = action.payload;
 })
 .addCase(fetchPayoutsBySeller.rejected, (state, action) => {
 state.loading = false;
 state.error = action.payload as string;
 })
 .addCase(fetchPayoutById.pending, (state) => {
 state.loading = true;
 state.error = null;
 })
 .addCase(fetchPayoutById.fulfilled, (state, action) => {
 state.loading = false;
 state.payout = action.payload;
 })
 .addCase(fetchPayoutById.rejected, (state, action) => {
 state.loading = false;
 state.error = action.payload as string;
 })
 // update payouts
 .addCase(updatePayoutStatus.pending, (state) => {
 state.loading = true;
 state.error = null;
 })
 .addCase(updatePayoutStatus.fulfilled, (state, action) => {
 state.loading = false;
 const index = state.payouts.findIndex(
 (p) => p.id === action.payload.id
);
 if (index !== -1) {
 state.payouts[index] = action.payload;
 }
 })
 .addCase(updatePayoutStatus.rejected, (state, action) => {
 state.loading = false;
 state.error = action.payload as string;
 });
 };
});
```

```

},
);

export default payoutsSlice.reducer;
import { createSlice, createAsyncThunk } from '@reduxjs/toolkit';
import axios from 'axios';
import { api } from '../../../../../Config/Api';

// Define the base URL for the API
const API_BASE_URL = '/api/seller/revenue/chart';
interface RevenueChart{
 date: string;
 revenue:number;
}
// Define interfaces for the state
interface RevenueState {
 chart:RevenueChart[];
 loading: boolean;
 error: string | null;
}

// Initial state for the slice
const initialState: RevenueState = {
 chart: [],
 loading: false,
 error: null,
};

export const fetchRevenueChart = createAsyncThunk(
 'revenue/fetchRevenueChart',
 async ({ type }: { type: string }, { rejectWithValue }) => {
 console.log("type ---- ",type)
 try {
 const token = localStorage.getItem('jwt');
 const response = await api.get(` ${API_BASE_URL}` , {
 params: { type },
 headers: { Authorization: `Bearer ${token}` },
 });
 console.log("revienue chart #####",response.data)
 return response.data;
 } catch (error: any) {
 console.log("error ",error.response)
 return rejectWithValue(error.response?.data || 'Failed to fetch daily revenue');
 }
 }
);

```

```
// Create RevenueSlice
const revenueSlice = createSlice({
 name: 'revenue',
 initialState,
 reducers: {},
 extraReducers: (builder) => {
 builder
 .addCase(fetchRevenueChart.pending, (state) => {
 state.loading = true;
 state.error = null;
 })
 .addCase(fetchRevenueChart.fulfilled, (state, action) => {
 state.loading = false;
 state.chart = action.payload;
 })
 .addCase(fetchRevenueChart.rejected, (state, action) => {
 state.loading = false;
 state.error = action.payload as string;
 });
 },
});

export default revenueSlice.reducer;
import { createSlice, createAsyncThunk, type PayloadAction } from
'@reduxjs/toolkit';
import { api } from '../../../../../Config/Api';
import type { Seller } from '../../../../../types/sellerTypes';
import axios from 'axios';

// Define initial state
interface SellerAuthState {
 otpSent: boolean;
 error: string | null;
 loading: boolean;
 jwt: string | null;
 sellerCreated: string | null;
}

const initialState: SellerAuthState = {
 otpSent: false,
 error: null,
 loading: false,
```

```

 jwt: null,
 sellerCreated:""
 };

const API_URL = '/sellers';

// Define async thunks for sending and verifying OTP
export const sendLoginOtp = createAsyncThunk('otp/sendLoginOtp', async (email: string, { rejectWithValue }) => {
 try {
 const {data}=await api.post('/sellers/sent/login-top', { email });
 console.log("otp sent - ", email, data)
 return { email };
 } catch (error:any) {
 console.log("error",error)
 return rejectWithValue(error.response?.data?.message || 'Failed to send OTP');
 }
});

export const verifyLoginOtp = createAsyncThunk('otp/verifyLoginOtp',
 async (data: { email: string; otp: string, navigate:any }, { rejectWithValue }) => {
 try {
 const response = await api.post('/sellers/verify/login-top', data);
 console.log("login seller success - ", response.data)
 localStorage.setItem("jwt",response.data.jwt)
 data.navigate("/seller")
 return response.data;
 } catch (error:any) {
 console.log("error",error.response?.data)
 return rejectWithValue(error.response?.data?.message || 'Failed to verify OTP');
 }
});

export const createSeller = createAsyncThunk<Seller, Seller>(
 'sellers/createSeller',
 async (seller: Seller, { rejectWithValue }) => {
 try {
 const response = await api.post<Seller>(API_URL, seller);
 console.log('create seller', response.data);
 return response.data;
 } catch (error:any) {
 if (axios.isAxiosError(error) && error.response) {
 console.error('Create seller error response data:', error.response.data);
 console.error('Create seller error response status:', error.response.status);
 }
 }
 }
);

```

```

 console.error('Create seller error response headers:', error.response.headers);
 return rejectWithValue(error.message);
 } else {
 console.error('Create seller error message:', error.message);
 return rejectWithValue('Failed to create seller');
 }
}

// Create the slice
const sellerAuthSlice = createSlice({
 name: 'sellerAuth',
 initialState,
 reducers: {
 resetSellerAuthState: (state) => {
 state.otpSent = false;
 state.error = null;
 state.loading = false;
 state.jwt = null;
 },
 },
 extraReducers: (builder) => {
 // Handle sendLoginOtp actions
 builder
 .addCase(sendLoginOtp.pending, (state) => {
 state.loading = true;
 state.error = null;
 })
 .addCase(sendLoginOtp.fulfilled, (state) => {
 state.loading = false;
 state.otpSent = true;
 state.error = null;
 })
 .addCase(sendLoginOtp.rejected, (state, action) => {
 state.loading = false;
 state.error = action.payload as string;
 });
 // Handle verifyLoginOtp actions
 builder
 .addCase(verifyLoginOtp.pending, (state) => {
 state.loading = true;
 state.error = null;
 })
 .addCase(verifyLoginOtp.fulfilled, (state, action) => {
 state.loading = false;
 state.jwt = action.payload.jwt;
 })
 }
});

```

```

 state.error = null;
 })
 .addCase(verifyLoginOtp.rejected, (state, action) => {
 state.loading = false;
 state.error = action.payload as string;
 })
 // create new seller
 .addCase(createSeller.pending, (state) => {
 state.loading = true;
 state.error = null;
 })
 .addCase(createSeller.fulfilled, (state, action:
PayloadAction<Seller>) => {
 // state.sellers.push(action.payload);
 state.sellerCreated = "verification email sent to you"
 state.loading = false;
 })
 .addCase(createSeller.rejected, (state, action) => {
 state.loading = false;
 state.error = action.payload as string || 'Failed to create
seller';
 })
 ;
},
));
;

// Export actions and reducer
export const { resetSellerAuthState } = sellerAuthSlice.actions;
export default sellerAuthSlice.reducer;
// src/redux/slices/sellerOrderSlice.ts

import { createSlice, createAsyncThunk, type PayloadAction } from
'@reduxjs/toolkit';
import { api } from '../Config/Api';
import type { Order, OrderStatus } from '../../types/orderTypes';
import type { ApiResponse } from '../../types/authTypes';

interface SellerOrderState {
 orders: Order[];
 loading: boolean;
 error: string | null;
}

const initialState: SellerOrderState = {
 orders: [],
 loading: false,
 error: null,
};


```

```

// Thunks for async actions
export const fetchSellerOrders = createAsyncThunk<Order[], string>(
 'sellerOrders/fetchSellerOrders',
 async (jwt, { rejectWithValue }) => {
 try {
 const response = await api.get('/seller/orders', {
 headers: { Authorization: `Bearer ${jwt}` },
 });

 console.log("fetch seller orders", response.data)
 return response.data;
 } catch (error: any) {
 console.log("error", error.response)
 return rejectWithValue(error.response.data);
 }
 }
);

export const updateOrderStatus = createAsyncThunk<Order,
{ jwt: string,
orderId: number,
orderStatus: OrderStatus
}>(
 'sellerOrders/updateOrderStatus',
 async ({ jwt, orderId, orderStatus }, { rejectWithValue }) => {
 try {
 const response = await
api.patch(`/seller/orders/${orderId}/status/${orderStatus}`,
 null,
 {
 headers: { Authorization: `Bearer ${jwt}` },
 });
 console.log("order status updated", response.data)
 return response.data;
 } catch (error: any) {
 return rejectWithValue(error.response.data);
 }
 }
);

export const deleteOrder = createAsyncThunk<ApiResponse, { jwt: string,
orderId: number }>(
 'sellerOrders/deleteOrder',
 async ({ jwt, orderId }, { rejectWithValue }) => {
 try {
 const response = await api.delete(`/seller/orders/${orderId}/delete`, {
 headers: { Authorization: `Bearer ${jwt}` },
 });
 return response.data;
 } catch (error: any) {

```

```
 return rejectWithValue(error.response.data);
 }
}
);

const sellerOrderSlice = createSlice({
 name: 'sellerOrders',
 initialState,
 reducers: {},
 extraReducers: (builder) => {
 builder
 .addCase(fetchSellerOrders.pending, (state) => {
 state.loading = true;
 state.error = null;
 })
 .addCase(fetchSellerOrders.fulfilled, (state, action: PayloadAction<Order[]>) => {
 state.loading = false;
 state.orders = action.payload;
 })
 .addCase(fetchSellerOrders.rejected, (state, action) => {
 state.loading = false;
 state.error = action.payload as string;
 })
 .addCase(updateOrderStatus.pending, (state) => {
 state.loading = true;
 state.error = null;
 })
 .addCase(updateOrderStatus.fulfilled, (state, action: PayloadAction<Order>) => {
 state.loading = false;
 const index = state.orders.findIndex(order => order.id === action.payload.id);
 if (index !== -1) {
 state.orders[index] = action.payload;
 }
 })
 .addCase(updateOrderStatus.rejected, (state, action) => {
 state.loading = false;
 state.error = action.payload as string;
 })
 .addCase(deleteOrder.pending, (state) => {
 state.loading = true;
 state.error = null;
 })
 .addCase(deleteOrder.fulfilled, (state, action) => {
 state.loading = false;
 state.orders = state.orders.filter(order => order.id !== action.meta.arg.orderId);
 })
 },
});
```

```

 })
 .addCase(deleteOrder.rejected, (state, action) => {
 state.loading = false;
 state.error = action.payload as string;
 });
 },
);
}

export default sellerOrderSlice.reducer;
import { createSlice, createAsyncThunk, type PayloadAction } from
"@reduxjs/toolkit";
import axios from "axios";
import { api } from "../../Config/Api";
import type { Product } from "../../types/productTypes";

const API_URL = "/sellers/product";

export const fetchSellerProducts = createAsyncThunk<Product[], any>(
 "sellerProduct/fetchSellerProducts",
 async (jwt, { rejectWithValue }) => {
 try {
 const response = await api.get<Product[]>(API_URL, {
 headers: { Authorization: `Bearer ${jwt}` },
 });
 console.log("seller products ", response.data);
 return response.data;
 } catch (error: any) {
 console.log("error ", error.response);
 return rejectWithValue(error.response.data);
 }
 }
);

export const createProduct = createAsyncThunk<
 Product,
 { request: any; jwt: string | null }
>(
 "sellerProduct/createProduct",
 async ({ request, jwt }, { rejectWithValue }) => {
 try {
 const response = await api.post<Product>(API_URL, request, {
 headers: { Authorization: `Bearer ${jwt}` },
 });
 console.log("product created ", response.data);
 return response.data;
 } catch (error: any) {
 console.log("error ", error.response);
 return rejectWithValue(error.response.data);
 }
 }
);

```

```
}

);

export const updateProduct = createAsyncThunk<
any,
{ productId: number; product: any }
>(
 "sellerProduct/updateProduct",
 async ({ productId, product }, { rejectWithValue }) => {
 try {
 const response = await api.patch(`/${API_URL}/${productId}`, product, {
 headers: { Authorization: `Bearer ${localStorage.getItem("jwt")}` },
 });
 console.log("product updated ", response.data);
 return response.data;
 } catch (error: any) {
 console.log("error ", error);
 return rejectWithValue(error.response.data);
 }
 }
);

export const updateProductStock = createAsyncThunk<any, any>(
 "sellerProduct/updateProductStock",
 async (productId, { rejectWithValue }) => {
 try {
 const response = await api.patch(
 `${API_URL}/${productId}/stock`,
 {},
 {
 headers: { Authorization: `Bearer ${localStorage.getItem("jwt")}` },
 }
);
 console.log("product stock updated ", response.data);
 return response.data;
 } catch (error: any) {
 console.log("error ", error);
 return rejectWithValue(error.response.data);
 }
 }
);

export const deleteProduct = createAsyncThunk<void, number>(
 "sellerProduct/deleteProduct",
 async (productId, { rejectWithValue }) => {
 try {
 await api.delete(`/${API_URL}/${productId}`);
 } catch (error: any) {
 return rejectWithValue(error.response.data);
 }
 }
);
```

```
 }
 }
);

interface SellerProductState {
 products: Product[];
 loading: boolean;
 error: string | null;
 productCreated: boolean;
}

const initialState: SellerProductState = {
 products: [],
 loading: false,
 error: null,
 productCreated: false,
};

const sellerProductSlice = createSlice({
 name: "sellerProduct",
 initialState,
 reducers: {},
 extraReducers: (builder) => {
 builder
 .addCase(fetchSellerProducts.pending, (state) => {
 state.loading = true;
 state.error = null;
 state.productCreated = false;
 })
 .addCase(
 fetchSellerProducts.fulfilled,
 (state, action: PayloadAction<Product[]>) => {
 state.products = action.payload;
 state.loading = false;
 }
)
 .addCase(fetchSellerProducts.rejected, (state, action) => {
 state.loading = false;
 state.error = action.error.message || "Failed to fetch products";
 })
 .addCase(createProduct.pending, (state) => {
 state.loading = true;
 state.error = null;
 state.productCreated = false;
 })
 .addCase(
 createProduct.fulfilled,
 (state, action: PayloadAction<Product>) => {
 state.products.push(action.payload);
 }
)
 }
});
```

```
 state.loading = false;
 state.productCreated = true;
 }
)
.addCase(createProduct.rejected, (state, action) => {
 state.loading = false;
 state.error = action.error.message || "Failed to create product";
 state.productCreated = false;
})
.addCase(updateProduct.pending, (state) => {
 state.loading = true;
 state.error = null;
})
.addCase(
 updateProduct.fulfilled,
 (state, action: PayloadAction<Product>) => {
 const index = state.products.findIndex(
 (product) => product.id === action.payload.id
);
 if (index !== -1) {
 state.products[index] = action.payload;
 }
 state.loading = false;
 }
)
.addCase(updateProduct.rejected, (state, action) => {
 state.loading = false;
 state.error = action.error.message || "Failed to update product";
})
.addCase(
 updateProductStock.fulfilled,
 (state, action: PayloadAction<Product>) => {
 const index = state.products.findIndex(
 (product) => product.id === action.payload.id
);
 if (index !== -1) {
 state.products[index] = action.payload;
 }
 state.loading = false;
 }
)
.addCase(deleteProduct.pending, (state) => {
 state.loading = true;
 state.error = null;
})
.addCase(deleteProduct.fulfilled, (state, action) => {
 state.products = state.products.filter(
 (product) => product.id !== action.meta.arg
);
})
```

```

 state.loading = false;
 })
 .addCase(deleteProduct.rejected, (state, action) => {
 state.loading = false;
 state.error = action.error.message || "Failed to delete product";
 });
},
))];

export default sellerProductSlice.reducer;
/* eslint-disable @typescript-eslint/no-explicit-any */
import { createSlice, createAsyncThunk, type PayloadAction } from
"@reduxjs/toolkit";
import axios from "axios";
import { api } from "../../Config/Api";
import type { Seller, SellerReport } from "../../types/sellerTypes";
import type { RootState } from "../Store";

// Define the initial state type
interface SellerState {
 sellers: Seller[];
 selectedSeller: Seller | null;
 profile: Seller | null;
 loading: boolean;
 error: string | null;
 report: SellerReport | null;
 profileUpdated: boolean;
}

// Define the initial state
const initialState: SellerState = {
 sellers: [],
 selectedSeller: null,
 loading: false,
 error: null,
 profile: null,
 report: null,
 profileUpdated: false,
};

// Define the base URL for the API
const API_URL = "/sellers";

// Create async thunks for API calls
export const fetchSellerProfile = createAsyncThunk<Seller, any>(
 "sellers/fetchSellerProfile",
 async (jwt: string, { rejectWithValue }) => {
 try {
 const response = await api.get<Seller>(` ${API_URL}/profile`, {

```

```
 headers: {
 Authorization: `Bearer ${jwt}`,
 },
 });
 console.log("fetch seller profile", response.data);
 return response.data;
} catch (error: any) {
 if (axios.isAxiosError(error) && error.response) {
 console.error(
 "Fetch sellers error response data:",
 error.response.data
);
 console.error(
 "Fetch sellers error response status:",
 error.response.status
);
 console.error(
 "Fetch sellers error response headers:",
 error.response.headers
);
 return rejectWithValue(error.message);
 } else {
 console.error("Fetch sellers error message:", error.message);
 return rejectWithValue("Failed to fetch sellers");
 }
}
}

export const fetchSellers = createAsyncThunk<Seller[], string>(
 "sellers/fetchSellers",
 async (status: string, { rejectWithValue }) => {
 try {
 const response = await api.get<Seller[]>(API_URL, {
 params: {
 status,
 },
 });
 console.log("fetch sellers", response.data);
 return response.data;
 } catch (error: any) {
 if (axios.isAxiosError(error) && error.response) {
 console.error(
 "Fetch sellers error response data:",
 error.response.data
);
 console.error(
 "Fetch sellers error response status:",
 error.response.status
);
 }
 }
 }
);
```

```

);
 console.error(
 "Fetch sellers error response headers:",
 error.response.headers
);
 return rejectWithValue(error.message);
 } else {
 console.error("Fetch sellers error message:", error.message);
 return rejectWithValue("Failed to fetch sellers");
 }
}

export const fetchSellerReport = createAsyncThunk<
 SellerReport,
 string, // JWT token type
 { rejectValue: string }
>("sellers/fetchSellerReport", async (jwt: string, { rejectWithValue }) => {
 try {
 const response = await api.get<SellerReport>(` ${API_URL} /report`, {
 headers: {
 Authorization: `Bearer ${jwt}`,
 },
 });
 console.log("Fetch seller report", response.data);
 return response.data;
 } catch (error: any) {
 console.log("error ", error);
 if (error.response) {
 return rejectWithValue(error.response.data.message);
 }
 return rejectWithValue("Failed to fetch seller report");
 }
});

export const fetchSellerById = createAsyncThunk<Seller, number>(
 "sellers/fetchSellerById",
 async (id: number, { rejectWithValue }) => {
 try {
 const response = await api.get<Seller>(` ${API_URL} /${id}`);
 return response.data;
 } catch (error: any) {
 if (axios.isAxiosError(error) && error.response) {
 console.error(
 "Fetch seller by ID error response data:",
 error.response.data
);
 console.error(

```

```

 "Fetch seller by ID error response status:",
 error.response.status
);
 console.error(
 "Fetch seller by ID error response headers:",
 error.response.headers
);
 return rejectWithValue(error.message);
} else {
 console.error("Fetch seller by ID error message:", error.message);
 return rejectWithValue("Failed to fetch seller");
}
}

export const updateSeller = createAsyncThunk<
 Seller, any
>(
 "sellers/updateSeller",
 async (
 seller : any,
 { rejectWithValue }
) => {
 console.log("seller update request ",seller)
 try {
 const response = await api.patch(`/${API_URL}`, seller,{
 headers: {
 Authorization: `Bearer ${localStorage.getItem("jwt")}`
 },
 });
 console.log("seller updated successfully", response.data);
 return response.data;
 } catch (error: any) {
 if (axios.isAxiosError(error) && error.response) {
 console.error(
 "Update seller error response data:",
 error.response
);
 return rejectWithValue(error.message);
 } else {
 console.error("Update seller error message:", error);
 return rejectWithValue("Failed to update seller");
 }
 }
 }
);

export const updateSellerAccountStatus = createAsyncThunk<

```

```
Seller,
{ id: number; status: string }
>(
 "sellers/updateSellerAccountStatus",
 async (
 { id, status }: { id: number; status: string },
 { rejectWithValue }
) => {
 try {
 const response = await api.patch(`admin/seller/${id}/status/${status}`);
 console.log("update seller status: ", response.data);
 return response.data;
 } catch (error: any) {
 if (axios.isAxiosError(error) && error.response) {
 console.error(
 "Update seller error response data:",
 error.response.data
);
 return rejectWithValue(error.message);
 } else {
 console.error("Update seller error message:", error.message);
 return rejectWithValue("Failed to update seller");
 }
 }
 }
);

export const verifySellerEmail = createAsyncThunk<
 any,
 { otp: number; navigate: any }
>(
 "sellers/verifySellerEmail",
 async ({ otp, navigate }, { rejectWithValue }) => {
 try {
 const response = await api.patch(`${API_URL}/verify/${otp}`);
 navigate("/seller-account-verified");
 console.log("verifiy seller email ", response.data);
 return response.data;
 } catch (error: any) {
 if (axios.isAxiosError(error) && error.response) {
 console.error(
 "Update seller error response data:",
 error.response.data
);
 return rejectWithValue(error.message);
 } else {
 console.error("Update seller error message:", error.message);
 return rejectWithValue("Failed to update seller");
 }
 }
 }
);
```

```
 }
 }
}

);

export const deleteSeller = createAsyncThunk<void, number>(
 "sellers/deleteSeller",
 async (id: number, { rejectWithValue }) => {
 try {
 await api.delete(` ${API_URL}/${id}`);
 } catch (error: any) {
 if (axios.isAxiosError(error) && error.response) {
 console.error(
 "Delete seller error response data:",
 error.response.data
);
 console.error(
 "Delete seller error response status:",
 error.response.status
);
 console.error(
 "Delete seller error response headers:",
 error.response.headers
);
 return rejectWithValue(error.message);
 } else {
 console.error("Delete seller error message:", error.message);
 return rejectWithValue("Failed to delete seller");
 }
 }
 }
);

// Create the slice
const sellerSlice = createSlice({
 name: "sellers",
 initialState,
 reducers: {},
 extraReducers: (builder) => {
 builder

 // fetch seller profile
 .addCase(fetchSellerProfile.pending, (state) => {
 state.loading = true;
 state.error = null;
 state.profileUpdated=false;
 })
 .addCase(
 fetchSellerProfile.fulfilled,
```

```
(state, action: PayloadAction<Seller>) => {
 state.profile = action.payload;
 state.loading = false;

}
)
.addCase(fetchSellerProfile.rejected, (state, action) => {
 state.loading = false;
 state.error = (action.payload as string) || "Failed to fetch sellers";
})
// fetch sellers
.addCase(fetchSellers.pending, (state) => {
 state.loading = true;
 state.error = null;
})
.addCase(
 fetchSellers.fulfilled,
 (state, action: PayloadAction<Seller[]>) => {
 state.sellers = action.payload;
 state.loading = false;
 }
)
.addCase(fetchSellers.rejected, (state, action) => {
 state.loading = false;
 state.error = (action.payload as string) || "Failed to fetch sellers";
})
.addCase(fetchSellerById.pending, (state) => {
 state.loading = true;
 state.error = null;
})
.addCase(
 fetchSellerById.fulfilled,
 (state, action: PayloadAction<Seller>) => {
 state.selectedSeller = action.payload;
 state.loading = false;
 }
)
.addCase(fetchSellerById.rejected, (state, action) => {
 state.loading = false;
 state.error = (action.payload as string) || "Failed to fetch seller";
})

.addCase(updateSeller.pending, (state) => {
 state.loading = true;
 state.error = null;
 state.profileUpdated=false;
})
.addCase(
 updateSeller.fulfilled,
```

```
(state, action: PayloadAction<Seller>) => {
 const index = state.sellers.findIndex(
 (seller) => seller.id === action.payload.id
);
 if (index !== -1) {
 state.sellers[index] = action.payload;
 }
 state.profile=action.payload
 state.loading = false;
 state.profileUpdated=true;
}
)
.addCase(updateSeller.rejected, (state, action) => {
 state.loading = false;
 state.error = (action.payload as string) || "Failed to update seller";
})

// update seller status
.addCase(updateSellerAccountStatus.pending, (state) => {
 state.loading = true;
 state.error = null;
})
.addCase(
 updateSellerAccountStatus.fulfilled,
 (state, action: PayloadAction<Seller>) => {
 const index = state.sellers.findIndex(
 (seller) => seller.id === action.payload.id
);
 if (index !== -1) {
 state.sellers[index] = action.payload;
 }
 state.loading = false;
 }
)
.addCase(updateSellerAccountStatus.rejected, (state, action) => {
 state.loading = false;
 state.error = (action.payload as string) || "Failed to update seller";
})
.addCase(deleteSeller.pending, (state) => {
 state.loading = true;
 state.error = null;
})
.addCase(deleteSeller.fulfilled, (state, action) => {
 state.sellers = state.sellers.filter(
 (seller) => seller.id !== action.meta.arg
);
 state.loading = false;
})
.addCase(deleteSeller.rejected, (state, action) => {
```

```

 state.loading = false;
 state.error = (action.payload as string) || "Failed to delete seller";
 })
 // seller report
 .addCase(fetchSellerReport.pending, (state) => {
 state.loading = true;
 state.error = null;
 })
 .addCase(fetchSellerReport.fulfilled, (state, action) => {
 state.loading = false;
 state.report = action.payload;
 })
 .addCase(fetchSellerReport.rejected, (state, action) => {
 state.loading = false;
 state.error = action.payload as string;
 });
},
);

export default sellerSlice.reducer;

// Define selector functions
export const selectSellers = (state: RootState) => state.sellers.sellers;
export const selectSelectedSeller = (state: RootState) =>
 state.sellers.selectedSeller;
export const selectSellerLoading = (state: RootState) => state.sellers.loading;
export const selectSellerError = (state: RootState) => state.sellers.error;
import { createSlice, createAsyncThunk } from '@reduxjs/toolkit';
import axios from 'axios';
import { api } from '../../Config/Api';
import type { Transaction } from '../../types/Transaction';
import type { Order } from '../../types/orderTypes';

interface TransactionState {
 transactions: Transaction[];
 transaction: Transaction | null;
 loading: boolean;
 error: string | null;
}

// Initial state
const initialState: TransactionState = {
 transactions: [],
 transaction: null,
 loading: false,
 error: null,
};

// Thunks

```

```
export const fetchTransactionsBySeller = createAsyncThunk<
 Transaction[],
 string,
 { rejectValue: string }
>('transactions/fetchTransactionsBySeller', async (jwt, { rejectWithValue }) =>
{
 try {
 const response = await api.get<Transaction[]>('/api/transactions/seller', {
 headers: {
 Authorization: `Bearer ${jwt}`,
 },
 });
 console.log("fetchTransactionsBySeller", response.data)
 return response.data;
 } catch (error: any) {
 if (error.response) {
 return rejectWithValue(error.response.data.message);
 }
 return rejectWithValue('Failed to fetch transactions');
 }
});

export const fetchAllTransactions = createAsyncThunk<
 Transaction[],
 void,
 { rejectValue: string }
>('transactions/fetchAllTransactions', async (_, { rejectWithValue }) => {
 try {
 const response = await api.get<Transaction[]>('/api/transactions');
 return response.data;
 } catch (error: any) {
 if (error.response) {
 return rejectWithValue(error.response.data.message);
 }
 return rejectWithValue('Failed to fetch all transactions');
 }
});

// Slice
const transactionSlice = createSlice({
 name: 'transactions',
 initialState,
 reducers: {},
 extraReducers: (builder) => {
 builder
 .addCase(fetchTransactionsBySeller.pending, (state) => {
 state.loading = true;
 })
 }
});
```

```
 state.error = null;
 })
.addCase(fetchTransactionsBySeller.fulfilled, (state, action) => {
 state.loading = false;
 state.transactions = action.payload;
})
.addCase(fetchTransactionsBySeller.rejected, (state, action) => {
 state.loading = false;
 state.error = action.payload as string;
})
.addCase(fetchAllTransactions.pending, (state) => {
 state.loading = true;
 state.error = null;
})
.addCase(fetchAllTransactions.fulfilled, (state, action) => {
 state.loading = false;
 state.transactions = action.payload;
})
.addCase(fetchAllTransactions.rejected, (state, action) => {
 state.loading = false;
 state.error = action.payload as string;
});
},
);

export default transactionSlice.reducer;
import {
 configureStore,
 combineReducers,
} from "@reduxjs/toolkit";
import { useDispatch, useSelector, type TypedUseSelectorHook } from "react-redux";

// Customer slices
import sellerSlice from "./Seller/sellerSlice";
import sellerAuthenticationSlice from "./Seller/sellerAuthenticationSlice";
import sellerProductSlice from "./Seller/sellerProductSlice";
import ProductSlice from "./Customer/ProductSlice";
import CartSlice from "./Customer/CartSlice";
import AuthSlice from "./Customer/AuthSlice";
import UserSlice from "./Customer/UserSlice";
import OrderSlice from "./Customer/OrderSlice";
import sellerOrderSlice from "./Seller/sellerOrderSlice";
import payoutSlice from "./Seller/payoutSlice";
import transactionSlice from "./Seller/transactionSlice";
import CouponSlice from "./Customer/CouponSlice";
import AdminCouponSlice from "./Admin/AdminCouponSlice";
import ReviewSlice from "./Customer/ReviewSlice";
import WishlistSlice from "./Customer/WishlistSlice";
```

```
import AiChatBotSlice from "./Customer/AiChatBotSlice";
import revenueChartSlice from "./Seller/revenueChartSlice";
import CustomerSlice from "./Customer/Customer/CustomerSlice";
import DealSlice from "./Admin/DealSlice";
import AdminSlice from "./Admin/AdminSlice";

const rootReducer = combineReducers({
 // customer
 auth: AuthSlice,
 user: UserSlice,
 products: ProductSlice,
 cart: CartSlice,
 orders: OrderSlice,
 coupone: CouponSlice,
 review: ReviewSlice,
 wishlist: WishlistSlice,
 aiChatBot: AiChatBotSlice,
 homePage: CustomerSlice,

 // seller
 sellers: sellerSlice,
 sellerAuth: sellerAuthenticationSlice,
 sellerProduct: sellerProductSlice,
 sellerOrder: sellerOrderSlice,
 payouts: payoutSlice,
 transaction: transactionSlice,
 revenueChart: revenueChartSlice,

 // admin
 adminCoupon: AdminCouponSlice,
 adminDeals: DealSlice,
 admin: AdminSlice,
});

const store = configureStore({
 reducer: rootReducer,
 // No need to define middleware unless you're adding custom ones
});

export type AppDispatch = typeof store.dispatch;
export type RootState = ReturnType<typeof rootReducer>;

export const useAppDispatch = () => useDispatch<AppDispatch>();
export const useAppSelector: TypedUseSelectorHook<RootState> = useSelector;

export default store;
import React from 'react'
import { Route, Routes } from 'react-router-dom'
import SellersTable from '../admin/pages/sellers/SellersTable'
```

```

import Coupon from '../admin/pages/Coupon/Coupon'
import CouponForm from '../admin/pages/Coupon/CreateCouponForm'
import GridTable from '../admin/pages/Home Page/GridTable'
import ElectronicsTable from '../admin/pages/Home Page/ElectronicsTable'
import ShopByCategoryTable from '../admin/pages/Home Page/ShopByCategoryTable'
import Deal from '../admin/pages/Home Page/Deal'

const AdminRoutes = () => {
 return (
 <Routes>
 <Route path='/' element={<SellersTable/>}/>
 <Route path='/coupon' element={<Coupon/>}/>
 <Route path='/add-coupon' element={<CouponForm/>}/>
 <Route path='/home-grid' element={<GridTable/>}/>
 <Route path='/electronics-category' element={<ElectronicsTable/>}/>
 <Route path='/shop-by-category' element={<ShopByCategoryTable/>}/>
 <Route path='/deals' element={<Deal/>}/>
 </Routes>
)
}

export default AdminRoutes

import React, { useEffect } from 'react'
import { Route, Routes } from 'react-router-dom'
import Home from '../customer/pages/Home/Home'
import Products from '../customer/pages/Products/Products'
import ProductDetails from
'../customer/pages/Products/ProductDetails/ProductDetails'
import Cart from '../customer/pages/Cart/Cart'
import Address from '../customer/pages/Checkout/AddressPage'
import Profile from '../customer/pages/Account/Profile'
import BecomeSeller from '../customer/pages/BecomeSeller/BecomeSeller'
import Footer from '../customer/components/Footer/Footer'
import Navbar from '../customer/components/Navbar/Navbar'
import NotFound from '../customer/pages/NotFound/NotFound'
import Auth from '../customer/pages/Auth/Auth'
import { useDispatch, useSelector } from '../Redux Toolkit/Store'
import { fetchUserCart } from '../Redux Toolkit/Customer/CartSlice'
import PaymentSuccessHandler from
'../customer/pages/Payment/PaymentSuccessHandler'
import Reviews from '../customer/pages/Review/Reviews'
import WriteReviews from '../customer/pages/Review/WriteReview'
import Wishlist from '../customer/pages/Wishlist/Wishlist'
import { getWishlistById } from '../Redux Toolkit/Customer/WishlistSlice'
import ChatBot from '../customer/pages/ChatBot/ChatBot'
import SearchProducts from '../customer/pages/Search/SearchProducts'

const CustomerRoutes = () => {

```

```

const dispatch = useAppDispatch()
const { cart, auth } = useAppSelector(store => store);

useEffect(() => {
 dispatch(fetchUserCart(localStorage.getItem("jwt") || ""))
 dispatch(getWishlistByUserId())
}, [auth.jwt])
return (
<>
<Navbar />
<Routes>
<Route path='/' element={<Home />} />
{ /* <Route path='/chat-bot' element={<ChatBot />} /> */}
<Route path='/products/:categoryId' element={<Products />} />
<Route path='/search-products' element={<SearchProducts />} />
<Route path='/reviews/:productId' element={<Reviews />} />
<Route path='/reviews/:productId/create' element={<WriteReviews />} />
<Route path='/product-details/:categoryId/:name/:productId'
element={<ProductDetails />} />
<Route path='/cart' element={<Cart />} />
<Route path='/wishlist' element={<Wishlist />} />
<Route path='/checkout/address' element={<Address />} />
<Route path='/account/*' element={<Profile />} />
<Route path='/login' element={<Auth/>} />
<Route path='/payment-success/:orderId'
element={<PaymentSuccessHandler/>} />
<Route path='*' element={<NotFound />} />
</Routes>
<Footer />
</>

)
}

export default CustomerRoutes
import React from 'react'
import { Route, Routes } from 'react-router-dom'
import HomePage from '../seller/pages/SellerDashboard/HomePage'
import Products from '../seller/pages/Products/Products'
import ProductForm from '../seller/pages/Products/AddProductForm'
import Orders from '../seller/pages/Orders/Orders'
import Profile from '../seller/pages/Account/Profile'
import Payment from '../seller/pages/Payment/Payment'
import TransactionTable from '../seller/pages/Payment/TransactionTable'
import Invetory from '../seller/pages/Invetory/Invetory'
import UpdateProductForm from '../seller/pages/Products/UpdateProductForm'

const SellerRoutes = () => {
 return (

```

```

<Routes>
 <Route path='/' element={<HomePage />} />
 <Route path='/products' element={<Products />} />
 <Route path='/add-product' element={<ProductForm />} />
 <Route path='/update-product/:productId' element={<UpdateProductForm />} />
</Routes>
)
}

export default SellerRoutes
.css-1nj0gs7 {
 transition: box-shadow 300ms cubic-bezier(0.4, 0, 0.2, 1) 0ms;
 /* box-shadow: none; */
 background-image: none;
 border-radius: 8px;
 border: none rgb(227, 232, 239);
 background-color: rgb(9, 31, 60);
 color: rgb(236, 239, 241);
 overflow: hidden;
 position: relative;
}

/* before */

.css-1nj0gs7::before {
 content: "";
 position: absolute;
 width: 210px;
 height: 210px;
 background: linear-gradient(140.9deg, rgb(133, 145, 161) -14.02%, rgba(144, 202, 249, 0) 77.58%);
 border-radius: 50%;
 top: -160px;
 right: -130px;
}

/* after */

.css-1nj0gs7::after {
 content: "";
 position: absolute;
 width: 210px;

```

```
height: 210px;
background: linear-gradient(210.04deg, rgb(133, 145, 161) -50.94%, rgba(144, 202, 249, 0) 83.49%);
border-radius: 50%;
top: -30px;
right: -180px;
}

import React from "react";
import "./Demo.css";
import AccountBalanceIcon from "@mui/icons-material/AccountBalance";

const Demo = () => {
 return (
 <div className="grid grid-cols-4 gap-2">
 <div className="col-span-4 md:col-span-2 lg:col-span-1 flex gap-5 items-center p-5 w-full border rounded-md h-[75px] css-1nj0gs7">
 <div className="rounded-md p-2 bg-[#000025]">
 <AccountBalanceIcon />
 </div>
 <div>
 <p className="font-bold text-lg">$203k</p>
 <p className="font-medium">Total Income</p>
 </div>
 </div>
 <div className="col-span-4 md:col-span-2 lg:col-span-1 flex gap-5 items-center p-5 w-full border rounded-md h-[75px] css-1nj0gs7">
 <div className="rounded-md p-2 bg-[#000025]">
 <AccountBalanceIcon />
 </div>
 <div>
 <p className="font-bold text-lg">$203k</p>
 <p className="font-medium">Total Income</p>
 </div>
 </div>
 <div className="col-span-4 md:col-span-2 lg:col-span-1 flex gap-5 items-center p-5 w-full border rounded-md h-[75px] css-1nj0gs7">
 <div className="rounded-md p-2 bg-[#000025]">
 <AccountBalanceIcon />
 </div>
 <div>
 <p className="font-bold text-lg">$203k</p>
 <p className="font-medium">Total Income</p>
 </div>
 </div>
 <div className="col-span-4 md:col-span-2 lg:col-span-1 flex gap-5 items-center p-5 w-full border rounded-md h-[75px] css-1nj0gs7">
 <div className="rounded-md p-2 bg-[#000025]">
 <AccountBalanceIcon />
 </div>
 </div>
 </div>
);
}
```

```

 </div>
 <div>
 <p className="font-bold text-lg">$203k</p>
 <p className="font-medium">Total Income</p>
 </div>
 </div>
 </div>
);

};

export default Demo;
import * as React from "react";
import DrawerList from "../../../../../admin/seller/components/drawerList/DrawerList";
import { AccountBox } from "@mui/icons-material";
import LogoutIcon from '@mui/icons-material/Logout';
import DashboardIcon from '@mui/icons-material/Dashboard';
import ReceiptIcon from '@mui/icons-material/Receipt';
import ShoppingBagIcon from '@mui/icons-material/ShoppingBag';
import InventoryIcon from '@mui/icons-material/Inventory';
import AddIcon from '@mui/icons-material/Add';
import AccountBalanceWalletIcon from '@mui/icons-material/AccountBalanceWallet';

const menu = [
{
 name: "Dashboard",
 path: "/seller",
 icon: <DashboardIcon className="text-primary-color" />,
 activeIcon: <DashboardIcon className="text-white" />,
},
{
 name: "Orders",
 path: "/seller/orders",
 icon: <ShoppingBagIcon className="text-primary-color" />,
 activeIcon: <ShoppingBagIcon className="text-white" />,
},
{
 name: "Products",
 path: "/seller/products",
 icon: <InventoryIcon className="text-primary-color" />,
 activeIcon: <InventoryIcon className="text-white" />,
},
{
 name: "Add Product",
 path: "/seller/add-product",
 icon: <AddIcon className="text-primary-color" />,
 activeIcon: <AddIcon className="text-white" />,
},
];

```

```

{
 name: "Payment",
 path: "/seller/payment",
 icon: <AccountBalanceWalletIcon className="text-primary-color" />,
 activeIcon: <AccountBalanceWalletIcon className="text-white" />,
},
{
 name: "Transaction",
 path: "/seller/transaction",
 icon: <ReceiptIcon className="text-primary-color" />,
 activeIcon: <ReceiptIcon className="text-white" />,
},
// {
// name: "Inventory",
// path: "/seller/inventory",
// icon: <MailIcon className="text-primary-color" />,
// activeIcon: <MailIcon className="text-white" />,
// },
];

```

```

const menu2 = [
 {
 name: "Account",
 path: "/seller/account",
 icon: <AccountBox className="text-primary-color" />,
 activeIcon: <AccountBox className="text-white" />,
 },
 {
 name: "Logout",
 path: "/",
 icon: <LogoutIcon className="text-primary-color" />,
 activeIcon: <LogoutIcon className="text-white" />,
 },
];

```

```

interface DrawerListProps {
 toggleDrawer?: any;
}

const SellerDrawerList = ({ toggleDrawer }: DrawerListProps) => {
 return <DrawerList menu={menu} menu2={menu2} toggleDrawer={toggleDrawer} />;
};

export default SellerDrawerList;
import React, { useEffect } from "react";
import { useFormik } from "formik";
import * as Yup from "yup";
import { TextField, Button } from "@mui/material";

```

```

import type { UpdateDetailsFormProps } from "./BussinessDetailsForm";
import { useAppDispatch, useAppSelector } from "../../Redux Toolkit/Store";
import { updateSeller } from "../../Redux Toolkit/Seller/sellerSlice";

const BankDetailsForm = ({ onClose }: UpdateDetailsFormProps) => {
 const { sellers } = useAppSelector((store) => store);
 const dispatch = useAppDispatch();

 const formik = useFormik({
 initialValues: {
 accountHolderName: "",
 accountNumber: "",
 ifscCode: "",
 },
 validationSchema: Yup.object({
 accountHolderName: Yup.string().required(
 "Account Holder Name is required"
),
 accountNumber: Yup.string().required("Account Number is required"),
 ifscCode: Yup.string().required("IFSC Code is required"),
 }),
 onSubmit: (values) => {
 console.log(values);
 dispatch(
 updateSeller({
 bankDetails: values,
 })
);
 onClose();
 },
 });

 useEffect(() => {
 if (sellers.profile) {
 formik.setValues({
 accountHolderName: sellers.profile.bankDetails?.accountHolderName || "",
 accountNumber: sellers.profile.bankDetails?.accountNumber || "",
 ifscCode: sellers.profile.bankDetails?.ifscCode || "",
 });
 }
 }, [sellers.profile]);

 return (
 <>
 <h1 className="text-xl pb-5 text-center font-bold text-gray-600">
 Bank Details
 </h1>
 <form className="space-y-5" onSubmit={formik.handleSubmit}>

```

```
<TextField
 fullWidth
 id="accountHolderName"
 name="accountHolderName"
 label="Account Holder Name"
 value={formik.values.accountHolderName}
 onChange={formik.handleChange}
 error={
 formik.touched.accountHolderName &&
 Boolean(formik.errors.accountHolderName)
 }
 helperText={
 formik.touched.accountHolderName && formik.errors.accountHolderName
 }
/>
<TextField
 fullWidth
 id="accountNumber"
 name="accountNumber"
 label="Account Number"
 value={formik.values.accountNumber}
 onChange={formik.handleChange}
 error={
 formik.touched.accountNumber && Boolean(formik.errors.accountNumber)
 }
 helperText={
 formik.touched.accountNumber && formik.errors.accountNumber
 }
/>
<TextField
 fullWidth
 id="ifscCode"
 name="ifscCode"
 label="IFSC Code"
 value={formik.values.ifscCode}
 onChange={formik.handleChange}
 error={formik.touched.ifscCode && Boolean(formik.errors.ifscCode)}
 helperText={formik.touched.ifscCode && formik.errors.ifscCode}
/>
<Button
 sx={{ py: ".9rem" }}
 color="primary"
 variant="contained"
 fullWidth
 type="submit"
>
 Save
</Button>
</form>
```

```

 </>
);
};

export default BankDetailsForm;

```

```

import React, { useEffect } from "react";
import { useFormik } from "formik";
import * as Yup from "yup";
import { TextField, Button } from "@mui/material";
import { useDispatch, useSelector } from "../../../../../Redux Toolkit/Store";
import { updateSeller } from "../../../../../Redux Toolkit/Seller/sellerSlice";

export interface UpdateDetailsFormProps {
 onClose: () => void;
}

```

```

const BusinessDetailsForm = ({ onClose }: UpdateDetailsFormProps) => {
 const dispatch = useDispatch();
 const { sellers } = useSelector((store) => store);
 const formik = useFormik({
 initialValues: {
 businessName: "",
 gstin: "",
 accountStatus: "",
 },
 validationSchema: Yup.object({
 businessName: Yup.string().required("Business Name is required"),
 gstin: Yup.string().required("GSTIN is required"),
 accountStatus: Yup.string().required("Account Status is required"),
 }),
 onSubmit: (values) => {
 console.log(values);
 dispatch(
 updateSeller({
 ...values,
 businessDetails: { businessName: values.businessName },
 })
);
 onClose();
 },
 });
}

useEffect(() => {
 if (sellers.profile) {
 formik.setValues({
 businessName: sellers.profile?.businessDetails?.businessName,
 gstin: sellers.profile?.gstin,
 accountStatus: sellers.profile?.accountStatus ?? "",
 });
 }
}

```

```
}, [sellers.profile]);

return (
 <>
 <h1 className="text-xl pb-5 text-center font-bold text-gray-600">
 Business Details
 </h1>
 <form className="space-y-5" onSubmit={formik.handleSubmit}>
 <TextField
 fullWidth
 id="businessName"
 name="businessName"
 label="Business Name"
 value={formik.values.businessName}
 onChange={formik.handleChange}
 error={
 formik.touched.businessName && Boolean(formik.errors.businessName)
 }
 helperText={formik.touched.businessName && formik.errors.businessName}
 />
 <TextField
 fullWidth
 id="gstIn"
 name="gstIn"
 label="GSTIN"
 value={formik.values.gstIn}
 onChange={formik.handleChange}
 error={formik.touched.gstIn && Boolean(formik.errors.gstIn) }
 helperText={formik.touched.gstIn && formik.errors.gstIn}
 />
 <TextField
 fullWidth
 id="accountStatus"
 name="accountStatus"
 label="Account Status"
 value={formik.values.accountStatus}
 onChange={formik.handleChange}
 error={
 formik.touched.accountStatus && Boolean(formik.errors.accountStatus)
 }
 helperText={
 formik.touched.accountStatus && formik.errors.accountStatus
 }
 />
 <Button
 sx={{ py: ".9rem" }}
 color="primary"
 variant="contained"
 fullWidth
 >
```

```

 type="submit"
 >
 Save
 </Button>
 </form>
</>
);
};

export default BusinessDetailsForm;
import React, { useEffect } from "react";
import { useFormik } from "formik";
import * as Yup from "yup";
import { TextField, Button } from "@mui/material";
import type { UpdateDetailsFormProps } from "./BusinessDetailsForm";
import { useAppDispatch, useAppSelector } from "../../../../../Redux Toolkit/Store";
import { updateSeller } from "../../../../../Redux Toolkit/Seller/sellerSlice";

const PersonalDetailsForm = ({ onClose }: UpdateDetailsFormProps) => {
 const { sellers } = useAppSelector(store => store)
 const dispatch=useAppDispatch();

 const formik = useFormik({
 initialValues: {
 sellerName: '',
 email: '',
 mobile: '',
 },
 validationSchema: Yup.object({
 sellerName: Yup.string().required("Seller Name is required"),
 email: Yup.string().email("Invalid email address").required("Email is required"),
 mobile: Yup.string().required("Mobile number is required"),
 }),
 onSubmit: (values) => {

 console.log("data ----- ",values);
 dispatch(updateSeller(values))
 onClose()
 },
 });
}

useEffect(() => {

 if (sellers.profile) {
 formik.setValues({
 sellerName: sellers.profile?.sellerName,
 email: sellers.profile?.email,
 mobile: sellers.profile?.mobile,
 })
 }
});

```

```

 }
 }

 , [sellers.profile])

return (
<>
 <h1 className="text-xl pb-5 text-center font-bold text-gray-600">
 Personal Details
 </h1>
 <form className="space-y-5" onSubmit={formik.handleSubmit}>
 <TextField
 fullWidth
 id="sellerName"
 name="sellerName"
 label="Seller Name"
 value={formik.values.sellerName}
 onChange={formik.handleChange}
 error={formik.touched.sellerName &&
Boolean(formik.errors.sellerName)}
 helperText={formik.touched.sellerName &&
formik.errors.sellerName}
 />
 <TextField
 fullWidth
 id="email"
 name="email"
 label="Seller Email"
 value={formik.values.email}
 onChange={formik.handleChange}
 error={formik.touched.email && Boolean(formik.errors.email)}
 helperText={formik.touched.email && formik.errors.email}
 />
 <TextField
 fullWidth
 id="mobile"
 name="mobile"
 label="Seller Mobile"
 value={formik.values.mobile}
 onChange={formik.handleChange}
 error={formik.touched.mobile &&
Boolean(formik.errors.mobile)}
 helperText={formik.touched.mobile && formik.errors.mobile}
 />
 <Button sx={{ py: ".9rem" }} color="primary" variant="contained"
fullWidth type="submit">
 Save
 </Button>

```

```
 </form>
 </>

);
};

export default PersonalDetailsForm;
import React, { useEffect } from "react";
import { useFormik } from "formik";
import * as Yup from "yup";
import { TextField, Button } from "@mui/material";
import type { UpdateDetailsFormProps } from "./BussinessDetailsForm";
import { useAppDispatch, useAppSelector } from "../../../../../Redux Toolkit/Store";
import { updateSeller } from "../../../../../Redux Toolkit/Seller/sellerSlice";

const PickupAddressForm = ({ onClose }: UpdateDetailsFormProps) => {
 const { sellers } = useAppSelector((store) => store);
 const dispatch = useAppDispatch();

 const formik = useFormik({
 initialValues: {
 address: "",
 city: "",
 state: "",
 mobile: ""
 },
 validationSchema: Yup.object({
 address: Yup.string().required("Address is required"),
 city: Yup.string().required("City is required"),
 state: Yup.string().required("State is required"),
 mobile: Yup.string().required("Mobile number is required"),
 }),
 onSubmit: (values) => {
 console.log(values);
 dispatch(
 updateSeller({
 pickupAddress: values,
 })
);
 onClose();
 },
 });
}

useEffect(() => {
 if (sellers.profile) {
 formik.setValues({
 address: sellers.profile.pickupAddress.address,
 city: sellers.profile.pickupAddress.city,
 });
 }
});
```

```
 state: sellers.profile.pickupAddress.state,
 mobile: sellers.profile.pickupAddress.mobile,
)},
 }
 , [sellers.profile]);

return (
 <>
 <h1 className="text-xl pb-5 text-center font-bold text-gray-600">
 Pickup Address
 </h1>
 <form className="space-y-5" onSubmit={formik.handleSubmit}>
 <TextField
 fullWidth
 id="address"
 name="address"
 label="Address"
 value={formik.values.address}
 onChange={formik.handleChange}
 error={formik.touched.address && Boolean(formik.errors.address)}
 helperText={formik.touched.address && formik.errors.address}
 />
 <TextField
 fullWidth
 id="city"
 name="city"
 label="City"
 value={formik.values.city}
 onChange={formik.handleChange}
 error={formik.touched.city && Boolean(formik.errors.city)}
 helperText={formik.touched.city && formik.errors.city}
 />
 <TextField
 fullWidth
 id="state"
 name="state"
 label="State"
 value={formik.values.state}
 onChange={formik.handleChange}
 error={formik.touched.state && Boolean(formik.errors.state)}
 helperText={formik.touched.state && formik.errors.state}
 />
 <TextField
 fullWidth
 id="mobile"
 name="mobile"
 label="Mobile"
 value={formik.values.mobile}
 onChange={formik.handleChange}
```

```

 error={formik.touched.mobile && Boolean(formik.errors.mobile)}
 helperText={formik.touched.mobile && formik.errors.mobile}
 />
 <Button
 sx={{ py: ".9rem" }}
 color="primary"
 variant="contained"
 fullWidth
 type="submit"
 >
 Save
 </Button>
</form>
</>
);
};

export default PickupAddressForm;
import React, { useEffect, useState } from "react";
import { useDispatch, useSelector } from "../../../../../Redux Toolkit/Store";
import {
 Alert,
 Avatar,
 Box,
 Button,
 Divider,
 Modal,
 Snackbar,
} from "@mui/material";
import ProfileFieldCard from "./ProfileFieldCard";
import EditIcon from "@mui/icons-material/Edit";
import PersonalDetailsForm from "./PersonalDetailsForm";
import BusinessDetailsForm from "./BusinessDetailsForm";
import PickupAddressForm from "./PickupAddressForm";
import BankDetailsForm from "./BankDetailsForm";

export const style = {
 position: "absolute" as "absolute",
 top: "50%",
 left: "50%",
 transform: "translate(-50%, -50%)",
 width: 400,
 bgcolor: "background.paper",
 boxShadow: 24,
 p: 4,
};

const Profile = () => {
 const { sellers } = useSelector((store) => store);

```



```
>
 <EditIcon />
</Button>
</div>
</div>
<div className="space-y-5">
 <Avatar
 sx={{ width: "10rem", height: "10rem" }}>
 {src="https://cdn.pixabay.com/photo/2014/11/29/19/33/bald-eagle-550804_640.jpg"
 />
 <div>
 <ProfileFildCard
 keys={"Seller Name"}
 value={sellers.profile?.sellerName}>
 />
 <Divider />
 <ProfileFildCard
 keys={"Seller Email"}
 value={sellers.profile?.email}>
 />
 <Divider />
 <ProfileFildCard
 keys={"Seller Mobile"}
 value={sellers.profile?.mobile}>
 />
 </div>
 </div>
</div>
<div className="mt-10 lg:w-[70%]">
 <div className="flex items-center pb-3 justify-between">
 <h1 className="text-2xl font-bold text-gray-600 ">
 Bussiness Details
 </h1>
 <div>
 <Button
 onClick={() => handleOpen("businessDetails")}
 size="small"
 sx={{ borderRadius: "2.9rem" }}
 variant="contained"
 className="w-16 h-16">
 <EditIcon />
 </Button>
 </div>
 </div>
</div>

<div className="">
 <ProfileFildCard
```

```
 keys={"Business Name/Brand Name"}
 value={sellers.profile?.businessDetails?.businessName}
 />
 <Divider />
 <ProfileFildCard
 keys={"GSTIN"}
 value={sellers.profile?.gstin || "not provided"}
 />
 <Divider />
 <ProfileFildCard
 keys={"Account Status"}
 value={sellers.profile?.accountStatus}
 />
</div>
</div>
<div className="mt-10 lg:w-[70%]">
 <div className="flex items-center pb-3 justify-between">
 <h1 className="text-2xl font-bold text-gray-600 ">Pickup Address</h1>
 <div>
 <Button
 onClick={() => handleOpen("pickupAddress")}
 size="small"
 sx={{ borderRadius: "2.9rem" }}
 variant="contained"
 className="w-16 h-16"
 >
 <EditIcon />
 </Button>
 </div>
 </div>
 <div className="space-y-5">
 <div className="">
 <ProfileFildCard
 keys={"Adress"}
 value={sellers.profile?.pickupAddress?.address}
 />
 <Divider />
 <ProfileFildCard
 keys={"City"}
 value={sellers.profile?.pickupAddress?.city || "not provided"}
 />
 <Divider />
 <ProfileFildCard
 keys={"State"}
 value={sellers.profile?.pickupAddress?.state}
 />
 <Divider />
 <ProfileFildCard
 keys={"Mobile"}>
```

```

 value={sellers.profile?.pickupAddress?.mobile}
 />
 </div>
 </div>
</div>
<div className="mt-10 lg:w-[70%]">
 <div className="flex items-center pb-3 justify-between">
 <h1 className="text-2xl font-bold text-gray-600 ">Bank Details</h1>
 <div>
 <Button
 onClick={() => handleOpen("bankDetails")}
 size="small"
 sx={{ borderRadius: "2.9rem" }}
 variant="contained"
 className="w-16 h-16"
 >
 <EditIcon />
 </Button>
 </div>
 </div>
<div className="space-y-5">
 <div className="">
 <ProfileFildCard
 keys={"Account Holder Name"}
 value={sellers.profile?.bankDetails?.accountHolderName}
 />
 <Divider />
 <ProfileFildCard
 keys={"Account Number"}
 value={
 sellers.profile?.bankDetails?.accountNumber || "not provided"
 }
 />
 <Divider />
 <ProfileFildCard
 keys={"IFSC CODE"}
 value={sellers.profile?.bankDetails?.ifscCode}
 />
 </div>
</div>
</div>

<Modal
 open={open}
 onClose={handleClose}
 aria-labelledby="modal-modal-title"
 aria-describedby="modal-modal-description"
>
 <Box sx={style}>{renderSelectedForm()}</Box>

```

```

 </Modal>
 <Snackbar
 anchorOrigin={{ vertical: "top", horizontal: "right" }}
 open={snackbarOpen}
 autoHideDuration={6000}
 onClose={handleCloseSnackbar}
 >
 <Alert
 onClose={handleCloseSnackbar}
 severity={sellers.error ? "error" : "success"}
 variant="filled"
 sx={{ width: "100%" }}
 >
 {sellers.error ? sellers.error : "Profile Updated Successfully"}
 </Alert>
 </Snackbar>
</div>
);
};

export default Profile;
/* eslint-disable @typescript-eslint/no-explicit-any */
import { Divider } from '@mui/material'

const ProfileFieldCard = ({value, keys}:any) => {
 return (
 <div className='p-5 flex items-center bg-slate-50 '>
 <p className='w-20 lg:w-36 pr-5'>{keys}</p>
 <Divider orientation="vertical" flexItem />
 <p className='pl-4 lg:pl-10 font-semibold lg:text-lg'>{value}</p>
 </div>
)
}

export default ProfileFieldCard
import React from 'react'

const Inventory = () => {
 return (
 <div>Inventory</div>
)
}

export default Inventory
import React from 'react'
import OrderTable from './OrderTable'

const Orders = () => {

```

```

 return (
 <div>
 <OrderTable/>
 </div>
)
 }

export default Orders
import * as React from 'react';
import Table from '@mui/material/Table';
import TableBody from '@mui/material/TableBody';
import TableCell, { tableCellClasses } from '@mui/material/TableCell';
import TableContainer from '@mui/material/TableContainer';
import TableHead from '@mui/material/TableHead';
import TableRow from '@mui/material/TableRow';
import Paper from '@mui/material/Paper';
import { Box, Button, Menu, MenuItem, styled } from '@mui/material';
import { useDispatch, useSelector } from '../../../../../Redux Toolkit/Store';
import { fetchSellerOrders, updateOrderStatus } from '../../../../../Redux Toolkit/Seller/sellerOrderSlice';
import type { Order, OrderItem } from '../../../../../types/orderTypes';

const StyledTableCell = styled(TableCell)(({ theme }) => ({
 [`&.${tableCellClasses.head}`]: {
 backgroundColor: theme.palette.common.black,
 color: theme.palette.common.white,
 },
 [`&.${tableCellClasses.body}`]: {
 fontSize: 14,
 },
}));

const StyledTableRow = styled(TableRow)(({ theme }) => ({
 '&:nth-of-type(odd)': {
 backgroundColor: theme.palette.action.hover,
 },
 '&:last-child td, &:last-child th': {
 border: 0,
 },
}));

const orderStatus = [
 { color: '#FFA500', label: 'PENDING' },
 { color: '#F5BCBA', label: 'PLACED' },
 { color: '#F5BCBA', label: 'CONFIRMED' },
 { color: '#1E90FF', label: 'SHIPPED' },
 { color: '#32CD32', label: 'DELIVERED' },
 { color: '#FF0000', label: 'CANCELLED' },
]

```

```

];
const orderStatusColor = {
 PENDING: { color: '#FFA500', label: 'PENDING' }, // Orange
 CONFIRMED: { color: '#F5BCBA', label: 'CONFIRMED' },
 PLACED: { color: '#F5BCBA', label: 'PLACED' },
 SHIPPED: { color: '#1E90FF', label: 'SHIPPED' }, // DodgerBlue
 DELIVERED: { color: '#32CD32', label: 'DELIVERED' }, // LimeGreen
 CANCELLED: { color: '#FF0000', label: 'CANCELLED' } // Red
};

export default function OrderTable() {
 const [page, setPage] = React.useState(0);
 const [rowsPerPage, setRowsPerPage] = React.useState(5);
 const { sellerOrder } = useAppSelector(store => store);
 const dispatch = useAppDispatch();

 const [anchorEl, setAnchorEl] = React.useState<{ [key: number]: HTMLElement | null }>({});

 const handleClick = (event: React.MouseEvent<HTMLElement>, orderId: number) => {
 setAnchorEl((prev) => ({ ...prev, [orderId]: event.currentTarget }));
 };

 const handleClose = (orderId: number) => {
 setAnchorEl((prev) => ({ ...prev, [orderId]: null }));
 };

 React.useEffect(() => {
 dispatch(fetchSellerOrders(localStorage.getItem("jwt") || ""));
 }, [dispatch]);

 const handleUpdateOrder = (orderId: number, orderStatus: any) => {
 dispatch(updateOrderStatus({
 jwt: localStorage.getItem("jwt") || "",
 orderId,
 orderStatus,
 }));
 handleClose(orderId);
 };

 return (
 <>
 <h1 className='pb-5 font-bold text-xl'>All Orders</h1>

 <TableContainer component={Paper}>
 <Table sx={{ minWidth: 700 }} aria-label="customized table">
 <TableHead>
 <TableRow>

```

```

<StyledTableCell>Order Id</StyledTableCell>
<StyledTableCell>Products</StyledTableCell>
<StyledTableCell>Shipping Address</StyledTableCell>
<StyledTableCell align="right">Order Status</StyledTableCell>
<StyledTableCell align="right">Update</StyledTableCell>
</TableRow>
</TableHead>
<TableBody>
 {sellerOrder.orders.map((item: Order) => (
 <StyledTableRow key={item.id}>
 <StyledTableCell align="left">{item.id}</StyledTableCell>
 <StyledTableCell component="th" scope="row">
 <div className='flex gap-1 flex-wrap'>
 {item.orderItems.map((orderItem: OrderItem) =>
 <div key={orderItem.id} className='flex gap-5'>
 <img className='w-20 rounded-md'
src={orderItem.product.images[0]} alt="" />
 <div className='flex flex-col justify-between py-2'>
 <h1>Title: {orderItem.product.title}</h1>
 <h1>Price: Rs.{orderItem.product.sellingPrice}</h1>
 <h1>Color: {orderItem.product.color}</h1>
 <h1>Size: {orderItem.size}</h1>
 </div>
 </div>
)}
 </div>
 </StyledTableCell>
 <StyledTableCell>
 <div className='flex flex-col gap-y-2'>
 <h1>{item.shippingAddress.name}</h1>
 <h1>{item.shippingAddress.address},

{item.shippingAddress.city}</h1>
 <h1>{item.shippingAddress.state} -

{item.shippingAddress.pinCode}</h1>
 <h1>Mobile:
{item.shippingAddress.mobile}</h1>
 </div>
 </StyledTableCell>
 <StyledTableCell
 sx={{color:orderStatusColor[item.orderStatus].color}}
 align="center"> <Box
sx={{borderColor:orderStatusColor[item.orderStatus].color}} className={`border
px-2 py-1 rounded-full text-xs`}>
 {item.orderStatus}</Box>
 </StyledTableCell>
 <StyledTableCell align="right">
 <Button
 size='small'
 onClick={(e) => handleClick(e, item.id)}>

```

```

 color='primary'
 className='bg-primary-color'>
 Status
 </Button>
 <Menu
 id={`status-menu ${item.id}`}
 anchorEl={anchorEl[item.id]}
 open={Boolean(anchorEl[item.id])}
 onClose={() => handleClose(item.id)}
 MenuListProps={{
 'aria-labelledby': `status-menu ${item.id}`,
 }}
 >
 {orderStatus.map((status) =>
 <MenuItem
 key={status.label}
 onClick={() => handleUpdateOrder(item.id, status.label)}>
 {status.label}</MenuItem>
)})
 </Menu>
 </StyledTableCell>
</StyledTableRow>
))})
</TableBody>
</Table>
</TableContainer>
</>
);
}
import { Button, Card, Divider } from '@mui/material'
import React, { useState } from 'react'
import TransactionTable from './TransactionTable';
import Payouts from './PayoutsTable';
import { useAppSelector } from '../../../../../Redux Toolkit/Store';

const tab = [
 { name: "Transaction" },
 // { name: "Payouts" }
]
const Payment = () => {
 const [activeTab, setActiveTab] = useState(tab[0].name);
 const { sellers } = useAppSelector((store) => store);

 const handleActiveTab = (item:any) => {
 setActiveTab(item.name);
 }
 return (
 <div>
 <div className='grid grid-cols-1 lg:grid-cols-2 gap-3'>

```

```

 <Card className='col-span-1 p-5 rounded-md space-y-4'>
 <h1 className='text-gray-600 font-medium'>Total Earnings</h1>
 <h1 className='font-bold text-xl pb-1'>₹{sellers.report?.totalEarnings}</h1>
 <Divider />
 <p className='text-gray-600 font-medium pt-1'>Last Payment : ₹0</p>
 </Card>
 {/* <Card className='col-span-1 p-5 rounded-md space-y-4'>
 <h1 className='text-gray-600 font-medium'>Payments To Be Settled</h1>
 <h1 className='font-bold text-xl pb-1'>₹0</h1>
 <Divider />
 <p className='text-gray-600 font-medium pt-1'>Next Payment : ₹0</p>
 </Card> */}
 </div>
 <div className='mt-20'>

 <div className='flex gap-4'>
 {tab.map((item) => <Button
 onClick={()=>handleActiveTab(item)} variant={activeTab === item.name ? "contained" : "outlined"}>{item.name}</Button>)}
 </div>
 <div className='mt-5'>
 {activeTab === "Transaction" ? <TransactionTable /> :
 <Payouts />}
 </div>
 </div>
)
}

export default Payment
import { Paper, Table, TableBody, TableCell, TableContainer, TableHead, TableRow } from '@mui/material'
import React from 'react'
import { useDispatch, useSelector } from '../../../../../Redux Toolkit/Store';
import type { Order, OrderItem } from '../../../../../types/orderTypes';
import { fetchPayoutsBySeller } from '../../../../../Redux Toolkit/Seller/payoutSlice';

const PayoutsTable = () => {
 const [page, setPage] = React.useState(0);
 const [rowsPerPage, setRowsPerPage] = React.useState(5);
 const { sellerOrder } = useSelector(store => store);
 const dispatch = useDispatch();

```

```
React.useEffect(() => {
 dispatch(fetchPayoutsBySeller(localStorage.getItem("jwt") || ""));
}, [dispatch]);

return (
 <div>
 <TableContainer component={Paper}>
 <Table sx={{ minWidth: 700 }} aria-label="customized table">
 <TableHead>
 <TableRow>
 <TableCell>Date</TableCell>
 <TableCell>Amount</TableCell>
 <TableCell align='right'>Status</TableCell>
 {/* <TableCell align="right">Amount</TableCell> */}
 </TableRow>
 </TableHead>
 <TableBody>
 {sellerOrder.orders.map((item: Order) => (
 <TableRow key={item.id}>
 <TableCell align="left">{item.id}</TableCell>
 <TableCell component="th" scope="row">
 <div className='flex gap-1 flex-wrap'>
 {item.orderItems.map((orderItem: OrderItem) =>
 <div key={orderItem.id} className='flex gap-5'>

 <div className='flex flex-col justify-between py-2'>
 <h1>Title: {orderItem.product.title}</h1>
 <h1>Price: Rs.{orderItem.product.sellingPrice}</h1>
 <h1>Color: {orderItem.product.color}</h1>
 <h1>Size: {orderItem.size}</h1>
 </div>
 </div>
)}
 </div>
 </TableCell>
 {/* <TableCell>
 <div className='flex flex-col gap-y-2'>
 <h1>{item.shippingAddress.name}</h1>
 <h1>{item.shippingAddress.address},
 {item.shippingAddress.city}</h1>
 <h1>{item.shippingAddress.state} -
 {item.shippingAddress.pinCode}</h1>
 <h1>Mobile:
 {item.shippingAddress.mobile}</h1>
 </div>
 </TableCell> */}
 {/* <TableCell
 <div> */}

```

```

 sx={{color:orderStatusColor[item.orderStatus].color}}
 align="center"> <Box
sx={{borderColor:orderStatusColor[item.orderStatus].color}} className={`border
px-2 py-1 rounded-full text-xs`}>
 {item.orderStatus}</Box>
</TableCell> */
/* <TableCell align="right">
<Button
 size='small'
 onClick={(e) => handleClick(e, item.id)}
 color='primary'
 className='bg-primary-color'>
 Status
</Button>
<Menu
 id={`status-menu ${item.id}`}
 anchorEl={anchorEl[item.id]}
 open={Boolean(anchorEl[item.id])}
 onClose={() => handleClose(item.id)}
 MenuListProps={{
 'aria-labelledby': `status-menu ${item.id}`,
 }}
>
 {orderStatus.map((status) =>
 <MenuItem
 key={status.label}
 onClick={() => handleUpdateOrder(item.id, status.label)}>
 {status.label}</MenuItem>
)}
</Menu>
</TableCell> */
</TableRow>
))}

</TableBody>
</Table>
</TableContainer>
</div>
)
}

export default PayoutsTable
import * as React from 'react';
import Table from '@mui/material/Table';
import TableBody from '@mui/material/TableBody';
import TableCell from '@mui/material/TableCell';
import TableContainer from '@mui/material/TableContainer';
import TableHead from '@mui/material/TableHead';
import TableRow from '@mui/material/TableRow';
import Paper from '@mui/material/Paper';

```

```

import { useAppDispatch, useAppSelector } from '../../../../../Redux Toolkit/Store';
import { fetchTransactionsBySeller } from '../../../../../Redux Toolkit/Seller/transactionSlice';
import type { Transaction } from '../../../../../types/Transaction';
import { redableDateTime } from '../../../../../util/redableDateTime';

const orderStatusColor = {
 PENDING: { color: '#FFA500', label: 'PENDING' }, // Orange
 SHIPPED: { color: '#1E90FF', label: 'SHIPPED' }, // DodgerBlue
 DELIVERED: { color: '#32CD32', label: 'DELIVERED' }, // LimeGreen
 CANCELLED: { color: '#FF0000', label: 'CANCELLED' } // Red
};

export default function TransactionTable() {
 const [page, setPage] = React.useState(0);
 const [rowsPerPage, setRowsPerPage] = React.useState(5);
 const { sellerOrder, transaction } = useAppSelector(store => store);
 const dispatch = useAppDispatch();

 React.useEffect(() => {
 dispatch(fetchTransactionsBySeller(localStorage.getItem("jwt") || ""));
 }, [dispatch]);

 return (
 <>
 <TableContainer component={Paper}>
 <Table sx={{ minWidth: 700 }} aria-label="customized table">
 <TableHead>
 <TableRow>
 <TableCell>Date</TableCell>
 <TableCell>Customer Details</TableCell>
 <TableCell>Order</TableCell>
 <TableCell align="right">Amount</TableCell>
 </TableRow>
 </TableHead>
 <TableBody>
 {transaction.transactions.map((item: Transaction) => (
 <TableRow key={item.id}>
 <TableCell align="left"><div className='space-y-1'>
 <h1
 className='font-medium'>{redableDateTime(item.date).split("at")[0]}</h1>
 <h1
 className='text-xs text-gray-600 font-semibold'>{redableDateTime(item.date).split("at")[1]}</h1>
 </div></TableCell>
 <TableCell component="th" scope="row">
 <div className='space-y-2'>
 <h1>{item.customer.fullName}</h1>
 <h1
 className='font-semibold'>{item.customer.email}</h1>
 </div>
 </TableCell>
 </TableRow>
))}
 </TableBody>
 </Table>
 </TableContainer>
 </>
);
}

```

```
 <h1 className='font-bold text-gray-600'>{item.customer.mobile}</h1>
 </div>
 </TableCell>
 <TableCell>
 Order Id : {item.order.id}
 </TableCell>
 <TableCell
 align="right">
 ₹{item.order.totalSellingPrice}
 </TableCell>
 </TableRow>
)
)
</TableBody>
</Table>
</TableContainer>
</>
);
}
import React, { useEffect, useState } from "react";
import { useFormik } from "formik";
import * as Yup from "yup";
import {
 TextField,
 Button,
 MenuItem,
 Select,
 InputLabel,
 FormControl,
 FormHelperText,
 Grid,
 CircularProgress,
 IconButton,
 Snackbar,
 Alert,
} from "@mui/material";
import AddPhotoAlternateIcon from "@mui/icons-material/AddPhotoAlternate";
import CloseIcon from "@mui/icons-material/Close";
import { mainCategory } from "../../data/category/mainCategory";
import { menLevelTwo } from "../../data/category/level two/menLevelTwo";
import { womenLevelTwo } from "../../data/category/level two/womenLevelTwo";
import { menLevelThree } from "../../data/category/level three/menLevelThree";
import { womenLevelThree } from "../../data/category/level three/womenLevelThree";
import { colors } from "../../data/Filter/color";
import { useDispatch, useSelector } from "../../Redux Toolkit/Store";
import { createProduct } from "../../Redux Toolkit/Seller/sellerProductSlice";
```

```
import { uploadToCloudinary } from "../../util/uploadToCloudinary";
import { electronicsLevelThree } from "../../data/category/level
three/electronicsLevelThree";
import { electronicsLevelTwo } from "../../data/category/level
two/electronicsLevelTwo";
import { furnitureLevelTwo } from "../../data/category/level
two/furnitureLevelTwo";
import { furnitureLevelThree } from "../../data/category/level
three/furnitureLevelThree";

const categoryTwo: { [key: string]: any[] } = {
 men: menLevelTwo,
 women: womenLevelTwo,
 kids: [],
 home_furniture: furnitureLevelTwo,
 beauty: [],
 electronics: electronicsLevelTwo,
};

const categoryThree: { [key: string]: any[] } = {
 men: menLevelThree,
 women: womenLevelThree,
 kids: [],
 home_furniture: furnitureLevelThree,
 beauty: [],
 electronics: electronicsLevelThree,
};

const validationSchema = Yup.object({
 title: Yup.string()
 .min(5, "Title should be at least 5 characters long")
 .required("Title is required"),
 description: Yup.string()
 .min(10, "Description should be at least 10 characters long")
 .required("Description is required"),
 price: Yup.number()
 .positive("Price should be greater than zero")
 .required("Price is required"),
 discountedPrice: Yup.number()
 .positive("Discounted Price should be greater than zero")
 .required("Discounted Price is required"),
 discountPercent: Yup.number()
 .positive("Discount Percent should be greater than zero")
 .required("Discount Percent is required"),
 quantity: Yup.number()
 .positive("Quantity should be greater than zero")
 .required("Quantity is required"),
 color: Yup.string().required("Color is required"),
 category: Yup.string().required("Category is required"),
```

```
sizes: Yup.string().required("Sizes are required"),
})

const ProductForm = () => {
 const [uploadImage, setUploadingImage] = useState(false);
 const dispatch = useAppDispatch();
 const { sellers, sellerProduct } = useAppSelector(store => store);

 const [snackbarOpen, setOpenSnackbar] = useState(false);

 const formik = useFormik({
 initialValues: {
 title: "",
 description: "",
 mrpPrice: "",
 sellingPrice: "",
 quantity: "",
 color: "",
 images: [],
 category: "",
 category2: "",
 category3: "",
 sizes: ""
 },
 // validationSchema: validationSchema,
 onSubmit: (values) => {
 dispatch(createProduct({ request: values, jwt: localStorage.getItem("jwt") }));
 console.log(values);
 },
 });
}

const handleImageChange = async (event: any) => {
 const file = event.target.files[0];
 setUploadingImage(true);
 const image = await uploadToCloudinary(file);
 // const image = URL.createObjectURL(file);
 formik.setFieldValue("images", [...formik.values.images, image]);
 setUploadingImage(false);
};

const handleRemoveImage = (index: number) => {
 const updatedImages = [...formik.values.images];
 updatedImages.splice(index, 1);
 formik.setFieldValue("images", updatedImages);
};

const childCategory = (category: any, parentCategoryId: any) => {
 return category.filter((child: any) => {
```

```

 // console.log("Category", parentCategoryId, child)
 return child.parentCategoryId == parentCategoryId;
));
};

const handleCloseSnackbar = () => {
 setOpenSnackbar(false);
}

useEffect(() => {
 if (sellerProduct.productCreated || sellerProduct.error) {
 setOpenSnackbar(true)
 }
}, [sellerProduct.productCreated, sellerProduct.error])

return (
 <div>
 <form onSubmit={formik.handleSubmit} className="space-y-4 p-4">
 <Grid container spacing={2}>
 <Grid className="flex flex-wrap gap-5" item xs={12}>
 <input
 type="file"
 accept="image/*"
 id="fileInput"
 style={{ display: "none" }}
 onChange={handleImageChange}
 />

 <label className="relative" htmlFor="fileInput">

 <AddPhotoAlternateIcon className="text-gray-700" />

 {uploadImage && (
 <div className="absolute left-0 right-0 top-0 bottom-0 w-24 h-24 flex justify-center items-center">
 <CircularProgress />
 </div>
) }
 </label>

 <div className="flex flex-wrap gap-2">
 {formik.values.images.map((image, index) => (
 <div className="relative">
 <img
 className="w-24 h-24 object-cover"
 key={index}
 src={image}
 alt={`ProductImage ${index + 1}`}
 />
 </div>
))
 </div>
 </Grid>
 </Grid>
 </form>
 </div>
)

```

```
 />
 <IconButton
 onClick={() => handleRemoveImage(index)}
 className=""
 size="small"
 color="error"
 sx={{
 position: "absolute",
 top: 0,
 right: 0,
 outline: "none",
 }}>
 <CloseIcon sx={{ fontSize: "1rem" }} />
 </IconButton>
 </div>
)))
</div>
</Grid>
<Grid item xs={12} sm={12}>
 <TextField
 fullWidth
 id="title"
 name="title"
 label="Title"
 value={formik.values.title}
 onChange={formik.handleChange}
 error={formik.touched.title && Boolean(formik.errors.title)}
 helperText={formik.touched.title && formik.errors.title}
 required
 />
</Grid>
<Grid item xs={12} sm={12}>
 <TextField
 multiline
 rows={4}
 fullWidth
 id="description"
 name="description"
 label="Description"
 value={formik.values.description}
 onChange={formik.handleChange}
 error={
 formik.touched.description && Boolean(formik.errors.description)
 }
 helperText={formik.touched.description &&
formik.errors.description}
 required
 />
```



```
>
 <MenuItem value="">
 None
 </MenuItem>

 {colors.map((color, index) => <MenuItem value={color.name}>
 <div className="flex gap-3">
 <span style={{ backgroundColor: color.hex }} className={`h-5
w-5 rounded-full ${color.name === "White" ? "border" : ""}`}>
 <p>{color.name}</p>
 </div>
 </MenuItem>) }
</Select>
{formik.touched.color && formik.errors.color && (
 <FormHelperText>{formik.errors.color}</FormHelperText>
) }
</FormControl>
</Grid>
<Grid item xs={12} sm={6} lg={3}>
 <FormControl
 fullWidth
 error={formik.touched.sizes && Boolean(formik.errors.sizes)}
 required
>
 <InputLabel id="sizes-label">Sizes</InputLabel>
 <Select
 labelId="sizes-label"
 id="sizes"
 name="sizes"
 value={formik.values.sizes}
 onChange={formik.handleChange}
 label="Sizes"
 >
 <MenuItem value="">
 None
 </MenuItem>
 <MenuItem value="FREE">FREE</MenuItem>
 <MenuItem value="S">S</MenuItem>
 <MenuItem value="M">M</MenuItem>
 <MenuItem value="L">L</MenuItem>
 <MenuItem value="XL">XL</MenuItem>
 </Select>
 {formik.touched.sizes && formik.errors.sizes && (
 <FormHelperText>{formik.errors.sizes}</FormHelperText>
) }
</FormControl>
</Grid>
<Grid item xs={12} sm={6} lg={4}>
 <FormControl
```

```
 fullWidth
 error={formik.touched.category && Boolean(formik.errors.category) }
 required
 >
 <InputLabel id="category-label">Category</InputLabel>
 <Select
 labelId="category-label"
 id="category"
 name="category"
 value={formik.values.category}
 onChange={formik.handleChange}
 label="Category"
 >
 {/* <MenuItem value="">None</MenuItem> */}
 {mainCategory.map((item) => (
 <MenuItem value={item.categoryId}>{item.name}</MenuItem>
)))
 </Select>
 {formik.touched.category && formik.errors.category && (
 <FormHelperText>{formik.errors.category}</FormHelperText>
) }
 </FormControl>
</Grid>

<Grid item xs={12} sm={6} lg={4}>
 <FormControl
 fullWidth
 error={formik.touched.category && Boolean(formik.errors.category) }
 required
 >
 <InputLabel id="category2-label">Second Category</InputLabel>
 <Select
 labelId="category2-label"
 id="category2"
 name="category2"
 value={formik.values.category2}
 onChange={formik.handleChange}
 label="Second Category"
 >
 {formik.values.category &&
 categoryTwo[formik.values.category] ?.map((item) => (
 <MenuItem value={item.categoryId}>{item.name}</MenuItem>
)))
 </Select>
 {formik.touched.category && formik.errors.category && (
 <FormHelperText>{formik.errors.category}</FormHelperText>
) }
 </FormControl>
</Grid>
```

```

<Grid item xs={12} sm={6} lg={4}>
 <FormControl
 fullWidth
 error={formik.touched.category && Boolean(formik.errors.category) }
 required
 >
 <InputLabel id="category-label">Third Category</InputLabel>
 <Select
 labelId="category-label"
 id="category"
 name="category3"
 value={formik.values.category3}
 onChange={formik.handleChange}
 label="Third Category"
 >
 <MenuItem value="">
 None
 </MenuItem>
 {formik.values.category2 &&
 childCategory(
 categoryThree[formik.values.category],
 formik.values.category2
)?.map((item: any) => (
 <MenuItem value={item.categoryId}>{item.name}</MenuItem>
)))
 </Select>
 {formik.touched.category && formik.errors.category && (
 <FormHelperText>{formik.errors.category}</FormHelperText>
)}
 </FormControl>
</Grid>
<Grid item xs={12}>
 <Button
 sx={{ p: "14px" }}
 color="primary"
 variant="contained"
 fullWidth
 type="submit"
 disabled={sellerProduct.loading}
 >
 {sellerProduct.loading ? <CircularProgress size="small"
 sx={{ width: "27px", height: "27px" }} /> : "Add Product"}
 </Button>
</Grid>
</Grid>
</form>
<Snackbar
 anchorOrigin={{ vertical: "top", horizontal: "right" }}
 open={snackBarOpen} autoHideDuration={6000}

```

```
 onClose={handleCloseSnackbar}
 >
 <Alert
 onClose={handleCloseSnackbar}
 severity={sellerProduct.error ? "error" : "success"}
 variant="filled"
 sx={{ width: '100%' }}
 >
 {sellerProduct.error ? sellerProduct.error : "Product created
successfully"}
 </Alert>
 </Snackbar>
 </div>

);
};

export default ProductForm;
import React from 'react'
import ProductTable from './ProductTable'

const Products = () => {
 return (
 <div>
 <ProductTable/>
 </div>
)
}

export default Products
import * as React from 'react';
import Table from '@mui/material/Table';
import TableBody from '@mui/material/TableBody';
import TableCell, { tableCellClasses } from '@mui/material/TableCell';
import TableContainer from '@mui/material/TableContainer';
import TableHead from '@mui/material/TableHead';
import TableRow from '@mui/material/TableRow';
import Paper from '@mui/material/Paper';
import { Button, IconButton, styled } from '@mui/material';
import { useDispatch, useSelector } from '../../../../../Redux Toolkit/Store';
import { fetchSellerProducts, updateProductStock } from '../../../../../Redux
Toolkit/Seller/sellerProductSlice';
import EditIcon from '@mui/icons-material/Edit';
import { useNavigate } from 'react-router-dom';
```

```

const StyledTableCell = styled(TableCell)(({ theme }) => ({
 [`&.${tableCellClasses.head}`]: {
 backgroundColor: theme.palette.common.black,
 color: theme.palette.common.white,
 },
 [`&.${tableCellClasses.body}`]: {
 fontSize: 14,
 },
}));

const StyledTableRow = styled(TableRow)(({ theme }) => ({
 '&:nth-of-type(odd)': {
 backgroundColor: theme.palette.action.hover,
 },
 // hide last border
 '&:last-child td, &:last-child th': {
 border: 0,
 },
}));

export default function ProductTable() {

 const { sellerProduct } = useAppSelector(store => store);
 const dispatch = useAppDispatch();
 const navigate=useNavigate();

 React.useEffect(() => {
 dispatch(fetchSellerProducts(localStorage.getItem("jwt")))
 , []))

 const handleUpdateStack = (id: number | undefined)=>() => {
 dispatch(updateProductStock(id))
 }

 return (
 <>
 <h1 className='pb-5 font-bold text-xl'>Products</h1>
 <TableContainer component={Paper}>
 <Table sx={{ minWidth: 700 }} aria-label="customized table">
 <TableHead>
 <TableRow>
 <StyledTableCell>Images</StyledTableCell>
 <StyledTableCell align="right">Title</StyledTableCell>
 <StyledTableCell align="right">MRP</StyledTableCell>

```

```

 <StyledTableCell align="right">Selling Price</StyledTableCell>
 <StyledTableCell align="right">Color</StyledTableCell>
 <StyledTableCell align="right">Update Stock</StyledTableCell>
 <StyledTableCell align="right">Update</StyledTableCell>
 </TableRow>
</TableHead>
<TableBody>
 {sellerProduct.products.map((item) => (
 <StyledTableRow key={item.id}>
 <StyledTableCell component="th" scope="row">
 <div className='flex gap-1 flex-wrap'>
 {item.images.map((image) =>)}
 </div>
 </StyledTableCell>
 <StyledTableCell align="right">{item.title}</StyledTableCell>
 <StyledTableCell align="right">
₹{item.mrpPrice}.0</StyledTableCell>
 <StyledTableCell align="right">
₹{item.sellingPrice}.0</StyledTableCell>
 <StyledTableCell align="right">{item.color}</StyledTableCell>
 <StyledTableCell align="right"> <Button
onClick={handleUpdateStack(item.id)}
size='small'>{item.in_stock?"in_stock":"out_stock"}</Button></StyledTableCell>
 <StyledTableCell align="right">
 <IconButton
onClick={()=>navigate("/seller/update-product/" + item.id)} color='primary'
className='bg-primary-color'>
 <EditIcon />
 </IconButton>
 </StyledTableCell>
 </StyledTableRow>
)));
 </TableBody>
</Table>
</TableContainer>
</>

);
}

import React, { useEffect, useState } from "react";
import { useFormik } from "formik";
import * as Yup from "yup";
import {
 TextField,
 Button,
 MenuItem,
 Select,

```

```

 InputLabel,
 FormControl,
 FormHelperText,
 CircularProgress,
 IconButton,
 Snackbar,
 Alert,
 Unstable_Grid2 as Grid
} from "@mui/material";
import AddPhotoAlternateIcon from "@mui/icons-material/AddPhotoAlternate";
import CloseIcon from "@mui/icons-material/Close";
// import { mainCategory } from "../../../../data/category/mainCategory";
import { menLevelTwo } from "../../../../data/category/level two/menLevelTwo";
import { womenLevelTwo } from "../../../../data/category/level two/womenLevelTwo";
import { menLevelThree } from "../../../../data/category/level
three/menLevelThree";
import { womenLevelThree } from "../../../../data/category/level
three/womenLevelThree";
import { colors } from "../../../../data/Filter/color";
import { useAppDispatch, useAppSelector } from "../../../../Redux Toolkit/Store";
import { updateProduct } from "../../../../Redux
Toolkit/Seller/sellerProductSlice";
import { uploadToCloudinary } from "../../../../util/uploadToCloudinary";
import { electronicsLevelThree } from "../../../../data/category/level
three/electronicsLevelThree";
import { electronicsLevelTwo } from "../../../../data/category/level
two/electronicsLevelTwo";
import { furnitureLevelTwo } from "../../../../data/category/level
two/furnitureLevleTwo";
import { furnitureLevelThree } from "../../../../data/category/level
three/furnitureLevelThree";
import { useParams } from "react-router-dom";
import { fetchProductById } from "../../../../Redux
Toolkit/Customer/ProductSlice";
import type { Seller } from "../../../../types/sellerTypes";

const categoryTwo: { [key: string]: any[] } = {
 men: menLevelTwo,
 women: womenLevelTwo,
 kids: [],
 home_furniture: furnitureLevelTwo,
 beauty: [],
 electronics: electronicsLevelTwo,
};

const categoryThree: { [key: string]: any[] } = {
 men: menLevelThree,
 women: womenLevelThree,
 kids: [],
}

```

```

 home_furniture: furnitureLevelThree,
 beauty: [],
 electronics: electronicsLevelThree,
 };

const validationSchema = Yup.object({
 title: Yup.string()
 .min(5, "Title should be at least 5 characters long")
 .required("Title is required"),
 description: Yup.string()
 .min(10, "Description should be at least 10 characters long")
 .required("Description is required"),
 price: Yup.number()
 .positive("Price should be greater than zero")
 .required("Price is required"),
 discountedPrice: Yup.number()
 .positive("Discounted Price should be greater than zero")
 .required("Discounted Price is required"),
 discountPercent: Yup.number()
 .positive("Discount Percent should be greater than zero")
 .required("Discount Percent is required"),
 quantity: Yup.number()
 .positive("Quantity should be greater than zero")
 .required("Quantity is required"),
 color: Yup.string().required("Color is required"),
 category: Yup.string().required("Category is required"),
 sizes: Yup.string().required("Sizes are required"),
})

interface FormValues {
 title: string;
 description: string;
 mrpPrice: number;
 sellingPrice: number;
 quantity: number;
 color: string;
 images: string[];
 category: any;
 sizes: string;
 seller: Seller | undefined,
 createdAt: any,
 numRatings: number;
 in_stock: boolean;
}
const UpdateProductForm = () => {
 const [uploadImage, setUploadingImage] = useState(false);
 const dispatch = useAppDispatch();
 const { sellers, sellerProduct, products } = useAppSelector(store => store);
 const { productId } = useParams();

```

```

const [snackbarOpen, setOpenSnackbar] = useState(false);

const formik = useFormik<FormValues>({
 initialValues: {
 title: "",
 description: "",
 mrpPrice: 0,
 sellingPrice: 0,
 quantity: 0,
 color: "",
 images: [],
 category: null,
 sizes: "",
 seller: undefined,
 createdAt: null,
 numRatings: 0,
 in_stock: true,
 },
 // validationSchema: validationSchema,
 onSubmit: (values) => {
 dispatch(updateProduct({
 productId: Number(productId), product: values}))
 console.log(values);
 },
});

const handleImageChange = async (event: any) => {
 const file = event.target.files[0];
 setUploadingImage(true);
 const image = await uploadToCloudinary(file);
 // const image = URL.createObjectURL(file);
 formik.setFieldValue("images", [...formik.values.images, image]);
 setUploadingImage(false);
};

const handleRemoveImage = (index: number) => {
 const updatedImages = [...formik.values.images];
 updatedImages.splice(index, 1);
 formik.setFieldValue("images", updatedImages);
};

const childCategory = (category: any, parentCategoryId: any) => {
 return category.filter((child: any) => {
 // console.log("Category", parentCategoryId, child)
 return child.parentCategoryId == parentCategoryId;
 });
};

```

```

const handleCloseSnackbar = () => {
 setOpenSnackbar(false);
}
useEffect(() => {
 dispatch(fetchProductById(Number(productId)));
}, [productId])

useEffect(() => {
 if (sellerProduct.productCreated || sellerProduct.error) {
 setOpenSnackbar(true)
 }
}, [sellerProduct.productCreated, sellerProduct.error])

useEffect(() => {
 formik.setValues({
 title: products.product?.title || "",
 description: products.product?.description || "",
 mrpPrice: products.product?.mrpPrice || 0,
 sellingPrice: products.product?.sellingPrice || 0,
 quantity: products.product?.quantity || 0,
 color: products.product?.color || "",
 images: products.product?.images || ["image/"],
 category: products.product?.category,
 sizes: products.product?.sizes || "",
 seller:products.product?.seller ,
 createdAt:products.product?.createdAt || "",
 numRatings:products.product?.numRatings || 0,
 in_stock: products.product?.in_stock || true,
 })
}, [products.product])

return (
 <div>
 <form onSubmit={formik.handleSubmit} className="space-y-4 p-4">
 <Grid container spacing={2}>
 <Grid className="flex flex-wrap gap-5" item xs={12}>
 <input
 type="file"
 accept="image/*"
 id="fileInput"
 style={{ display: "none" }}
 onChange={handleImageChange}
 />

```

```
<label className="relative" htmlFor="fileInput">
 <span className="w-24 h-24 cursor-pointer flex
items-center justify-center p-3 border rounded-md border-gray-400">
 <AddPhotoAlternateIcon className="text-gray-700"
/>

 {uploadImage && (
 <div className="absolute left-0 right-0 top-0
bottom-0 w-24 h-24 flex justify-center items-center">
 <CircularProgress />
 </div>
) }
</label>

<div className="flex flex-wrap gap-2">
 {formik.values.images.map((image, index) => (
 <div className="relative">
 <img
 className="w-24 h-24 object-cover"
 key={index}
 src={image}
 alt={`ProductImage ${index + 1}`}
 />
 <IconButton
 onClick={() => handleRemoveImage(index)}
 className=""
 size="small"
 color="error"
 sx={{ {
 position: "absolute",
 top: 0,
 right: 0,
 outline: "none",
 } }}
 >
 <CloseIcon sx={{ fontSize: "1rem" }} />
 </IconButton>
 </div>
)))
</div>
</Grid>
<Grid item xs={12} sm={12}>
 <TextField
 fullWidth
 id="title"
 name="title"
 label="Title"
 value={formik.values.title}
 onChange={formik.handleChange}
 </Grid>

```

```
 error={formik.touched.title &&
Boolean(formik.errors.title)}
 helperText={formik.touched.title &&
formik.errors.title}
 required
 />
 </Grid>
<Grid item xs={12} sm={12}>
 <TextField
 multiline
 rows={4}
 fullWidth
 id="description"
 name="description"
 label="Description"
 value={formik.values.description}
 onChange={formik.handleChange}
 error={
 formik.touched.description &&
Boolean(formik.errors.description)
 }
 helperText={formik.touched.description &&
formik.errors.description}
 required
 />
 </Grid>
<Grid item xs={12} sm={6} lg={3}>
 <TextField
 fullWidth
 id="mrp_price"
 name="mrpPrice"
 label="MRP Price"
 type="number"
 value={formik.values.mrpPrice}
 onChange={formik.handleChange}
 error={formik.touched.mrpPrice &&
Boolean(formik.errors.mrpPrice)}
 helperText={formik.touched.mrpPrice &&
formik.errors.mrpPrice}
 required
 />
 </Grid>
<Grid item xs={12} sm={6} lg={3}>
 <TextField
 fullWidth
 id="sellingPrice"
 name="sellingPrice"
 label="Selling Price"
 type="number"
```

```

 value={formik.values.sellingPrice}
 onChange={formik.handleChange}
 error={
 formik.touched.sellingPrice &&
 Boolean(formik.errors.sellingPrice)
 }
 helperText={
 formik.touched.sellingPrice &&
 formik.errors.sellingPrice
 }
 required
 />
</Grid>

<Grid item xs={12} sm={6} lg={3}>
 <FormControl
 fullWidth
 error={formik.touched.color &&
Boolean(formik.errors.color)}
 required
 >
 <InputLabel id="color-label">Color</InputLabel>
 <Select
 labelId="color-label"
 id="color"
 name="color"
 value={formik.values.color}
 onChange={formik.handleChange}
 label="Color"
 >
 <MenuItem value="">
 None
 </MenuItem>

 {colors.map((color, index) => <MenuItem
value={color.name}>
 <div className="flex gap-3">
 <span style={{ backgroundColor:
color.hex }} className={`h-5 w-5 rounded-full ${color.name === "White" ?
"border" : ""}}>
 <p>{color.name}</p>
 </div>
 </MenuItem>) }
 </Select>
 {formik.touched.color && formik.errors.color && (
 <FormHelperText>{formik.errors.color}</FormHelperText>
) }
 </FormControl>

```

```

 </Grid>
 <Grid item xs={12} sm={6} lg={3}>
 <FormControl
 fullWidth
 error={formik.touched.sizes &&
Boolean(formik.errors.sizes) }
 required
 >
 <InputLabel id="sizes-label">Sizes</InputLabel>
 <Select
 labelId="sizes-label"
 id="sizes"
 name="sizes"
 value={formik.values.sizes}
 onChange={formik.handleChange}
 label="Sizes"
 >
 <MenuItem value="">
 None
 </MenuItem>
 <MenuItem value="FREE">FREE</MenuItem>
 <MenuItem value="S">S</MenuItem>
 <MenuItem value="M">M</MenuItem>
 <MenuItem value="L">L</MenuItem>
 <MenuItem value="XL">XL</MenuItem>
 </Select>
 {formik.touched.sizes && formik.errors.sizes && (
 <FormHelperText>{formik.errors.sizes}</FormHelperText>
) }
 </FormControl>
 </Grid>

 <Grid item xs={12}>
 <Button
 sx={{ p: "14px" }}
 color="primary"
 variant="contained"
 fullWidth
 type="submit"
 disabled={sellerProduct.loading}
 >
 {sellerProduct.loading ? <CircularProgress
size="small"
 sx={{ width: "27px", height: "27px" }} /> :
 "Update Product"}
 </Button>
 </Grid>

```

```

 </Grid>
 </form>
 <Snackbar
 anchorOrigin={{ vertical: "top", horizontal: "right" }}
 open={snackbarOpen} autoHideDuration={6000}
 onClose={handleCloseSnackbar}
 >
 <Alert
 onClose={handleCloseSnackbar}
 severity={sellerProduct.error ? "error" : "success"}
 variant="filled"
 sx={{ width: '100%' }}
 >
 {sellerProduct.error ? sellerProduct.error : "Product
created successfully"}
 </Alert>
 </Snackbar>
</div>

);

};

export default UpdateProductForm;
import React, { useEffect } from 'react'
import { useNavigate, useParams } from 'react-router-dom'
import { useDispatch } from '../../../../../Redux Toolkit/Store';
import { verifySellerEmail } from '../../../../../Redux Toolkit/Seller/sellerSlice';

const SellerAccountVerification = () => {
 const{ otp}=useParams();
 const navigate=useNavigate()
 const dispatch=useDispatch();

 useEffect(()=>{
 dispatch(verifySellerEmail({otp:Number(otp),navigate}))
 ,[otp])

 return (
 <div>SellerAccountVerification</div>
)
}

export default SellerAccountVerification
import { Alert, Button } from '@mui/material'
import React from 'react'
import { useNavigate } from 'react-router-dom'

const SellerAccountVerified = () => {

```

```

const navigate = useNavigate()
return (
 <div className='h-[80vh] flex flex-col justify-center items-center space-y-3'>

 <Alert variant="filled" severity="success">
 Your Email Get Verified Successfully
 </Alert>
 <div>
 <Button variant='contained' onClick={() =>
navigate("/become-seller")}>
 Login As Seller
 </Button>
 </div>

 </div>
)
}

export default SellerAccountVerified
import { createTheme } from "@mui/material";

const customeTheme = createTheme({
 palette: {
 mode: "light", // This sets the theme to dark mode
 primary: {
 main: "#00927c",
 },
 secondary: {
 main: "#EAF0F1",
 // main:"#ffad4d"
 },
 },
});

export default customeTheme;
// src/types/authTypes.ts
export interface AuthResponse {
 jwt: string;
 message: string;
 role: string;
}

export interface ApiResponse {
 message: string;
 status: boolean;
}

```

```
}

export interface LoginRequest {
 email: string;
 otp: string;
 navigate:any;
}

export interface SignupRequest {
 email: string;
 fullName: string;
 otp: string;
 navigate:any
}

export interface ResetPasswordRequest {
 token: string;
 password: string;
}

export interface AuthState {
 jwt: string | null;
 role: string | null;
 loading: boolean;
 error: string | null;
 otpSent:boolean
}
import type { Product } from "./productTypes";
import type { User } from "./userTypes";

export interface CartItem {
 id: number;
 cart?: Cart;
 product: Product;
 size: string;
 quantity: number;
 mrpPrice: number;
 sellingPrice: number;
 userId: number;
}

export interface Cart {
 id: number;
 user: User;
 cartItems: CartItem[];
 totalSellingPrice: number;
 totalItem: number;
 totalMrpPrice: number;
```

```
 discount: number;
 couponCode: string | null;
}
import type { Cart } from "./cartTypes";

export interface Coupon {
 id: number;
 code: string;
 discountPercentage: number;
 validityStartDate: string;
 validityEndDate: string;
 minimumOrderValue: number;
 active: boolean;
}

export interface CouponState {
 coupons: Coupon[];
 cart: Cart | null;
 loading: boolean;
 error: string | null;
 couponCreated:boolean;
 couponApplied: boolean,
}
import type { HomeCategory } from "./homeDataTypes";

export interface Deal{
 id?: number;
 discount:number;
 category: HomeCategory;
}
// types.ts

export interface ApiResponse {
 message: string;
 status: boolean;
}

export interface DealsState {
 deals: Deal[];
 loading: boolean;
 error: string | null;
 dealCreated:boolean,
 dealUpdated:boolean,
}
// types.ts
// export interface HomeData {
// id: string;
// title: string;
// description: string;
```

```
// imageUrl: string;
// }

interface Deal{
 category:HomeCategory;
 discount:number;
}

export interface HomeData {
 id: number;
 grid: HomeCategory[];
 shopByCategories: HomeCategory[];
 electricCategories: HomeCategory[];
 deals: Deal[];
 dealCategories:HomeCategory[];
}

export interface HomeCategory {
 id?:number;
 categoryId: string;
 section?: string;
 name?: string;
 image: string;
 parentCategoryId?: string;
}

import type { Product } from './productTypes';
import type { Address, User } from './userTypes';

export interface OrderState {
 orders: Order[];
 orderItem:OrderItem | null;
 currentOrder: Order | null;
 paymentOrder: any | null;
 loading: boolean;
 error: string | null;
 orderCanceled: boolean
}

export interface Order {
 id: number;
 orderId: string;
 user: User;
 sellerId: number;
 orderItems: OrderItem[];
 orderDate: string;
 shippingAddress: Address;
 paymentDetails: any;
 totalMrpPrice: number;
 totalSellingPrice?: number; // Optional field
}
```

```

 discount?: number; // Optional field
 orderStatus: OrderStatus;
 totalItem: number;
 deliverDate:string;
 }

export enum OrderStatus {
 PENDING = 'PENDING',
 SHIPPED = 'SHIPPED',
 DELIVERED = 'DELIVERED',
 CANCELLED = 'CANCELLED'
}

export interface OrderItem {
 id: number;
 order: Order;
 product: Product;
 size: string;
 quantity: number;
 mrpPrice: number;
 sellingPrice: number;
 userId: number;
}
// types/payoutsTypes.ts

import type { Order } from "./orderTypes";
import type { Seller } from "./sellerTypes";
import type { Transaction } from "./Transaction";
import type { User } from "./userTypes";

export interface Payouts {
 id: number;
 transactions: Transaction[];
 seller: Seller;
 amount: number;
 status: "PENDING" | "SUCCESS" | "REJECTED";
 date: string;
}
import type { Seller } from "./sellerTypes";

export interface Category {
 id?: number; // Optional since id is auto-generated
 name: string;
 categoryId: string;
 parentCategory?: Category; // Optional since a category might not have a
parent
 level: number;
}

```

```
export interface Product {
 id?: number; // Optional since id is auto-generated
 title: string;
 description: string;
 mrpPrice: number;
 sellingPrice: number;
 discountPercent?: number;
 quantity?: number;
 color: string;
 images: any[]; // Array of strings for image URLs
 numRatings?: number;
 category?: Category; // Optional since a product might not have a category
 assigned yet
 seller?: Seller; // Placeholder for Seller interface (assuming it exists)
 createdAt?: Date; // Assuming LocalDateTime can be mapped to Date
 sizes: string; // Array of strings for product sizes
 in_stock?: boolean; //
}
import type { Product } from "./productTypes";
import type { User } from "./userTypes";

export interface Review {
 id: number;
 reviewText: string;
 rating: number;
 user: User;
 product: Product;
 productImages:string[];
 createdAt: string;
 updatedAt: string;
}

export interface CreateReviewRequest {
 reviewText: string;
 reviewRating: number;
}

export interface ApiResponse {
 message: string;
 status: boolean;
}

export interface ReviewState {
 reviews: Review[];
 loading: boolean;
 error: string | null;
 reviewCreated: boolean;
 reviewUpdated: boolean;
 reviewDeleted: boolean;
```

```
}

// src/types/seller.ts

export interface PickupAddress {
 name: string;
 mobile: string;
 pincode: string;
 address: string;
 locality: string;
 city: string;
 state: string;
}

export interface BankDetails {
 accountNumber: string;
 ifscCode: string;
 accountHolderName: string;
}

export interface BusinessDetails {
 businessName: string;
}

export interface Seller {
 id?:number;
 mobile: string;
 otp: string;
 gstin: string;
 pickupAddress: PickupAddress;
 bankDetails: BankDetails;
 sellerName: string;
 email: string;
 businessDetails: BusinessDetails;
 password: string;
 accountStatus?:string;
}

export interface SellerReport {
 id: number;
 seller: Seller;
 totalEarnings: number;
 totalSales: number;
 totalRefunds: number;
 totalTax: number;
 netEarnings: number;
 totalOrders: number;
 canceledOrders: number;
 totalTransactions: number;
}
```

```
import type { Order } from "./orderTypes";
import type { Seller } from "./sellerTypes";
import type { User } from "./userTypes";

export interface Transaction {
 id: number;
 customer: User;
 order: Order;
 seller: Seller;
 date: string;
}
// src/types/addressTypes.ts
export interface Address {
 id?: number;
 name: string;
 mobile: string;
 pinCode: string;
 address: string;
 locality: string;
 city: string;
 state: string;
}

export enum UserRole {
 ROLE_CUSTOMER = 'ROLE_CUSTOMER',
 ROLE_ADMIN = 'ROLE_ADMIN',
 ROLE_SELLER = 'ROLE_SELLER',
}

export interface User {
 id?: number;
 password?: string;
 email: string;
 fullName: string;
 mobile?: string;
 role: UserRole;
 addresses?: Address[];
}

export interface UserState {
 user: User | null;
 loading: boolean;
 error: string | null;
 profileUpdated: boolean;
}

import type { Product } from "./productTypes";
import type { User } from "./userTypes";

export interface Wishlist {
```

```
id: number;
user: User;
products: Product[];
}

export interface WishlistState {
wishlist: Wishlist | null;
loading: boolean;
error: string | null;
}

// Payload interfaces for async thunks
export interface AddProductToWishlistPayload {
wishlistId: number;
productId: number;
}

.App {
font-family: "Open Sans", sans-serif;
font-optical-sizing: auto;
font-weight: 300;
font-style: normal;
font-variation-settings: "wdth" 100;
}

import './App.css';
import { ThemeProvider } from '@emotion/react';
import customeTheme from './Theme/customeTheme';
import { Button } from '@mui/material';
import Navbar from './customer/components/Navbar/Navbar';
import Home from './customer/pages/Home/Home';
import Footer from './customer/components/Footer/Footer';
import Products from './customer/pages/Products/Products';
import { Route, Routes, useNavigate } from 'react-router-dom';

import SellerDashboard from './seller/pages/SellerDashboard/SellerDashboard';
import CustomerRoutes from './routes/CustomerRoutes';
import AdminDashboard from './admin/pages/Dashboard/Dashboard';
import SellerAccountVerification from
'./seller/pages/SellerAccountVerification';
import SellerAccountVerified from './seller/pages/SellerAccountVerified';
import { useAppDispatch, useAppSelector } from './Redux Toolkit/Store';
import { useEffect } from 'react';
import { fetchSellerProfile } from './Redux Toolkit/Seller/sellerSlice';
import BecomeSeller from './customer/pages/BecomeSeller/BecomeSeller';
import AdminLoginForm from './admin/pages/Auth/AdminLogin';
import AdminAuth from './admin/pages/Auth/AdminAuth';
import { fetchUserProfile } from './Redux Toolkit/Customer/UserSlice';
import { createHomeCategories, fetchHomePageData } from './Redux Toolkit/Customer/Customer/AsyncThunk';
```

```

import { homeCategories } from './data/homeCategories';
import Mobile from './data/Products/mobile';

function App() {
 const dispatch = useAppDispatch()
 // Use scoped selectors
 const jwt = useAppSelector(state => state.auth.jwt);
 const sellerJwt = useAppSelector(state => state.sellerAuth.jwt);
 const user = useAppSelector(state => state.user.user);
 const sellers = useAppSelector(state => state.sellers);
 const navigate = useNavigate();

 useEffect(() => {
 const storedJwt = localStorage.getItem("jwt");
 if (storedJwt) {
 // Only fetch user profile if we don't have user data
 if (!user) {
 dispatch(fetchUserProfile({ jwt: storedJwt, navigate }));
 }

 // Only fetch seller profile if user is a seller and we have a seller JWT
 if (sellerJwt && user?.role === 'ROLE_SELLER') {
 dispatch(fetchSellerProfile(storedJwt));
 }
 }
 }, [jwt, sellerJwt, user, dispatch, navigate]);

 useEffect(() => {
 dispatch(createHomeCategories(homeCategories))
 // dispatch(fetchHomePageData())
 }, [dispatch])

 return (
 <ThemeProvider theme={customTheme}>
 <div className='App' >

 <Routes>
 {sellers?.profile && <Route path='/seller/*' element={<SellerDashboard />} />}
 {user?.role === "ROLE_ADMIN" && <Route path='/admin/*' element={<AdminDashboard />} />}
 <Route path='/verify-seller/:otp' element={<SellerAccountVerification />} />
 <Route path='/seller-account-verified' element={<SellerAccountVerified />} />
 <Route path='/become-seller' element={<BecomeSeller />} />
 <Route path='/admin-login' element={<AdminAuth />} />
 </Routes>
 </div>
 </ThemeProvider>
);
}

export default App;

```

```
<Route path='/dummy' element={<Mobile />} />

<Route path='*' element={<CustomerRoutes />} />

</Routes>
{ /* <Footer/> */}
</div>

</ThemeProvider>
);
}

export default App;
/* @import "tailwindcss"; */

@tailwind base;
@tailwind components;
@tailwind utilities;
import { StrictMode } from 'react'
import { createRoot } from 'react-dom/client'
import './index.css'
import App from './App.tsx'
import { Provider } from 'react-redux'
import store from './Redux Toolkit/Store.ts'
import { BrowserRouter } from 'react-router-dom'

createRoot(document.getElementById('root')!).render(
<StrictMode>
 <Provider store={store}>
 <BrowserRouter>
 <App />
 </BrowserRouter>
 </Provider>
</StrictMode>,
)
/// <reference types="vite/client" />
node_modules
npm-debug.log
build
.dockerignore
**/.git
**/.DS_Store
**/node_modules
README.md
```