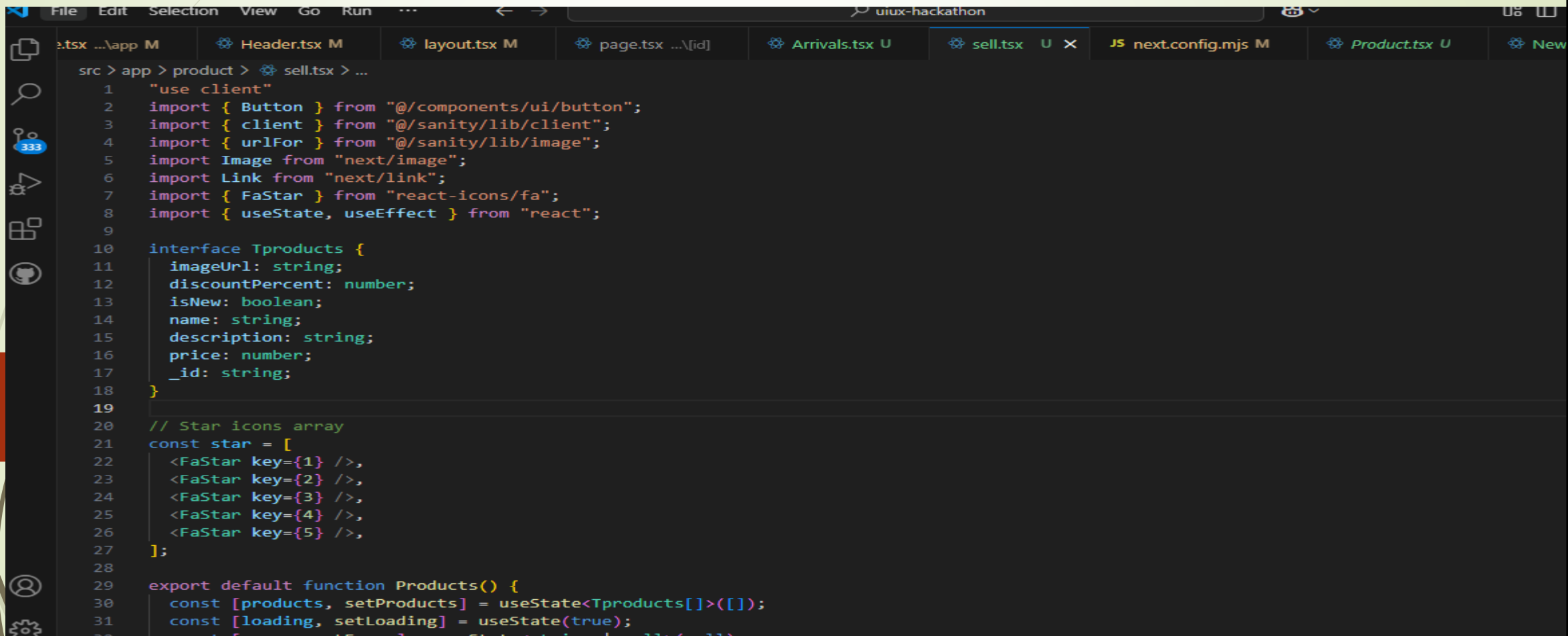
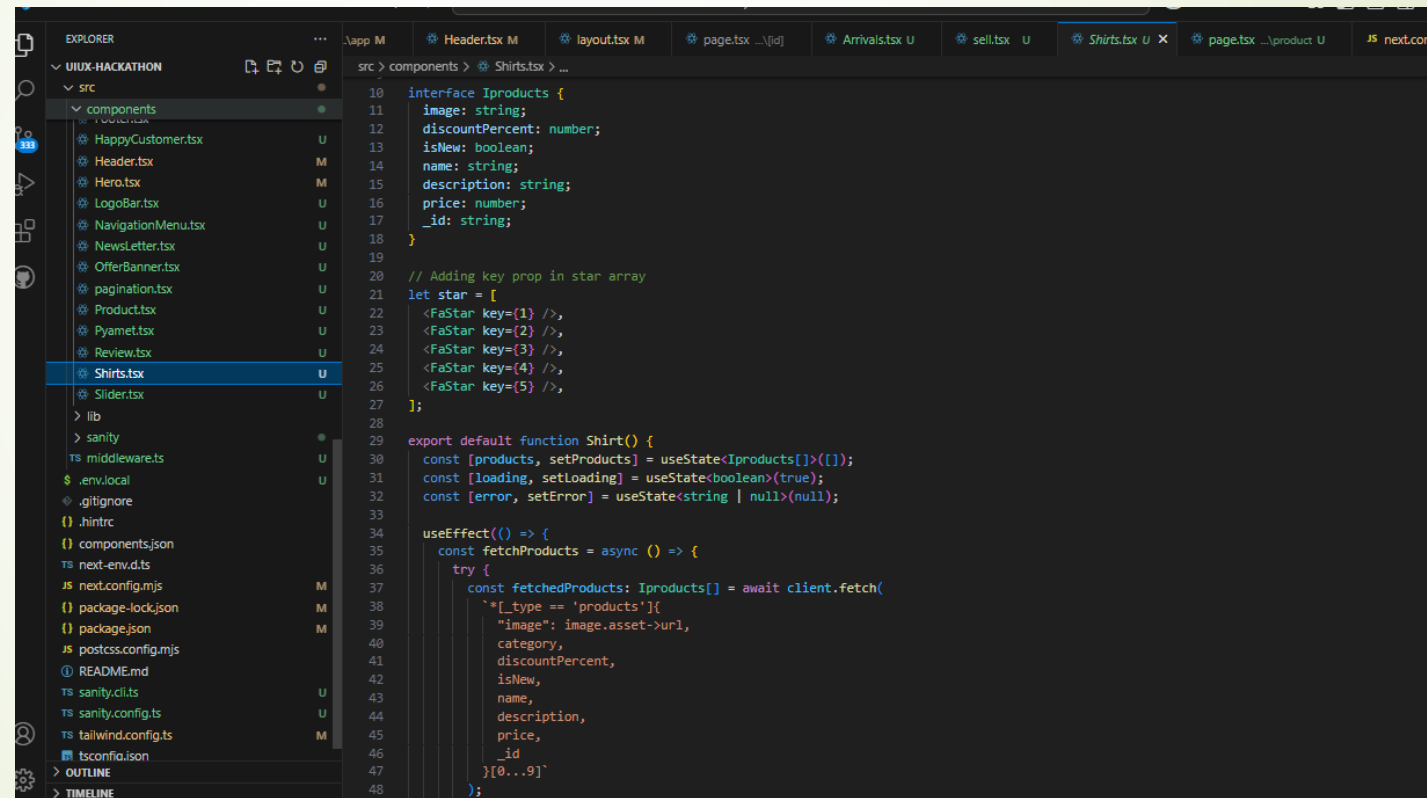


Day4- Building Dynamic Frontened Components:



```
File Edit Selection View Go Run ... uiux-hackathon
sell.tsx ...app M Header.tsx M layout.tsx M page.tsx ...[id] Arrivals.tsx U sell.tsx U X JS next.config.mjs M Product.tsx U New
src > app > product > sell.tsx > ...
1  "use client"
2  import { Button } from "@components/ui/button";
3  import { client } from "@sanity/lib/client";
4  import { urlFor } from "@sanity/lib/image";
5  import Image from "next/image";
6  import Link from "next/link";
7  import { FaStar } from "react-icons/fa";
8  import { useState, useEffect } from "react";
9
10 interface Tproducts {
11   imageUrl: string;
12   discountPercent: number;
13   isNew: boolean;
14   name: string;
15   description: string;
16   price: number;
17   _id: string;
18 }
19
20 // Star icons array
21 const star = [
22   <FaStar key={1} />,
23   <FaStar key={2} />,
24   <FaStar key={3} />,
25   <FaStar key={4} />,
26   <FaStar key={5} />,
27 ];
28
29 export default function Products() {
30   const [products, setProducts] = useState<Tproducts[]>([]);
31   const [loading, setLoading] = useState(true);
32   const [error, setError] = useState<string | null>(null);
33   useEffect(() => {
34     client.fetch<Tproducts[]>('*')
35       .then((products) => setProducts(products))
36       .catch((error) => setError(error));
37   }, []);
38 }
```

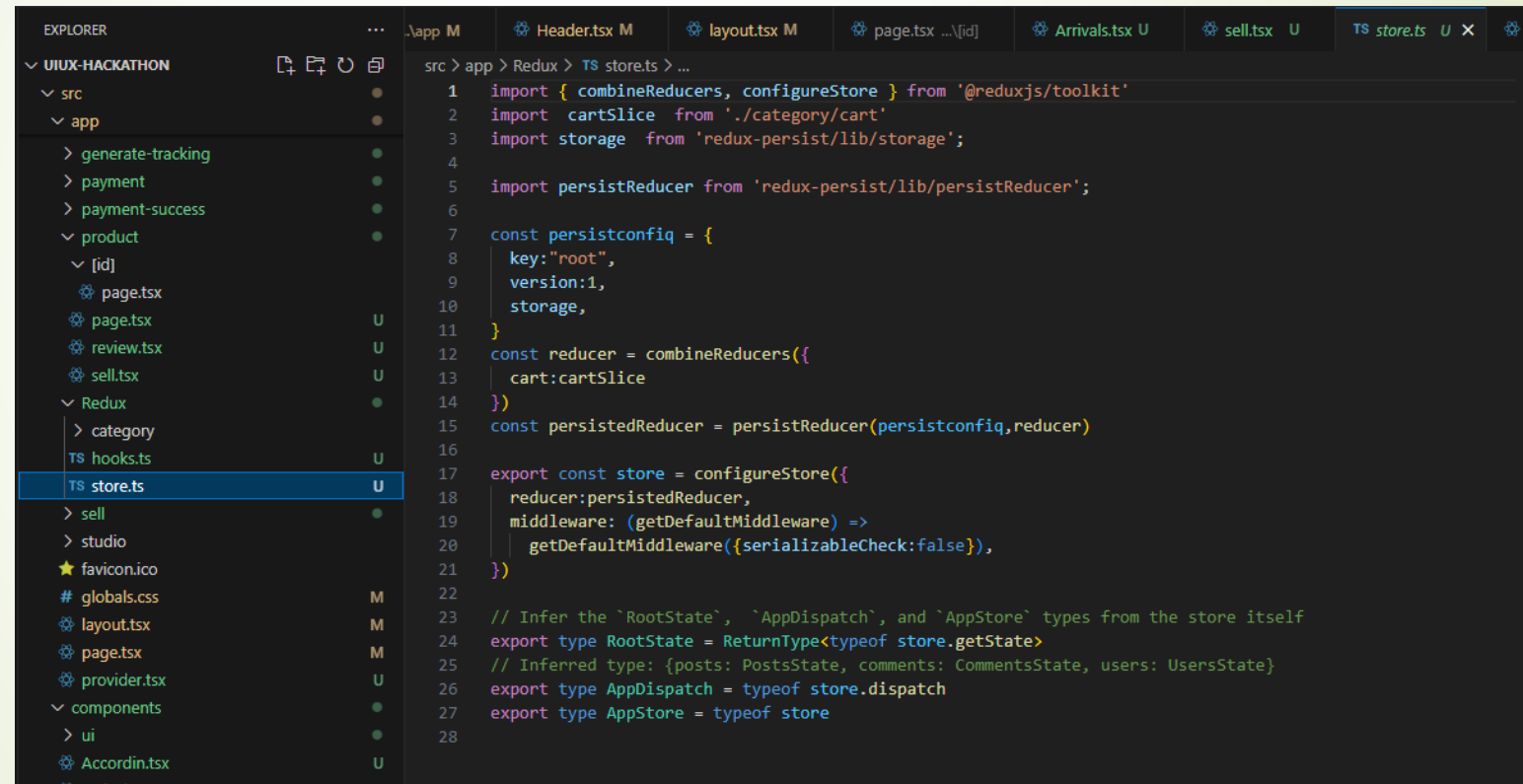
Product Detail Page/Dynamic Route:



The screenshot shows a VS Code editor with a project named 'UI/UX-HACKATHON'. The Explorer sidebar on the left displays the file structure, with 'Shirts.tsx' selected under the 'components' directory. The main editor area shows the code for 'Shirts.tsx', which includes an interface for 'Iproducts' and a function 'Shirt()' that uses state and an effect to fetch product data.

```
10 interface Iproducts {
11   image: string;
12   discountPercent: number;
13   isNew: boolean;
14   name: string;
15   description: string;
16   price: number;
17   _id: string;
18 }
19
20 // Adding key prop in star array
21 let star = [
22   <FaStar key={1} />,
23   <FaStar key={2} />,
24   <FaStar key={3} />,
25   <FaStar key={4} />,
26   <FaStar key={5} />,
27 ];
28
29 export default function Shirt() {
30   const [products, setProducts] = useState<Iproducts[]>([]);
31   const [loading, setLoading] = useState<boolean>(true);
32   const [error, setError] = useState<string | null>(null);
33
34   useEffect(() => {
35     const fetchProducts = async () => {
36       try {
37         const fetchedProducts: Iproducts[] = await client.fetch(
38           `*[_type == 'products']{
39             "image": image.asset->url,
40             category,
41             discountPercent,
42             isNew,
43             name,
44             description,
45             price,
46             _id
47           }[0...9]`
48         );
49       } catch (error) {
50         setError(error.message);
51       }
52     };
53     fetchProducts();
54   });
55 }
```

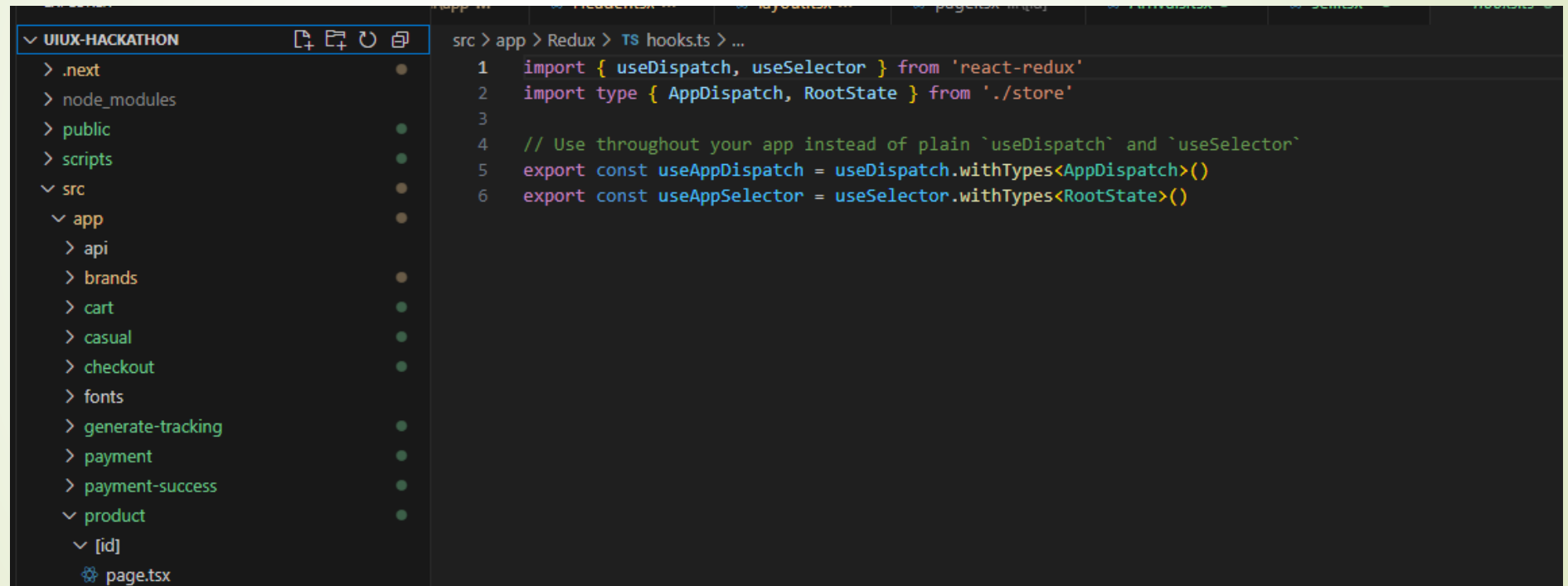
Redux Store:



The screenshot shows a Visual Studio Code editor with a file explorer on the left and a code editor on the right. The file explorer shows a project structure for 'UIUX-HACKATHON' with a 'src' directory containing 'app', 'product', 'components', and 'ui'. The 'app' directory contains 'generate-tracking', 'payment', 'payment-success', 'product', '[id]', 'page.tsx', 'review.tsx', 'sell.tsx', 'Redux', 'category', 'hooks.ts', and 'store.ts'. The 'store.ts' file is selected and highlighted in blue. The code editor shows the implementation of the Redux Store in 'src > app > Redux > TS store.ts'. The code imports 'combineReducers', 'configureStore' from '@reduxjs/toolkit', 'cartSlice' from './category/cart', and 'storage' from 'redux-persist/lib/storage'. It also imports 'persistReducer' from 'redux-persist/lib/persistReducer'. A 'persistconfig' object is defined with 'key: "root"', 'version: 1', and 'storage'. A 'reducer' is defined as 'combineReducers({ cart: cartSlice })'. A 'persistedReducer' is created using 'persistReducer(persistconfig, reducer)'. The 'store' is configured with 'configureStore({ reducer: persistedReducer, middleware: (getDefaultMiddleware) => getDefaultMiddleware({ serializableCheck: false }) })'. The code also includes type definitions for 'RootState', 'AppDispatch', and 'AppStore' based on the store's state and dispatch methods.

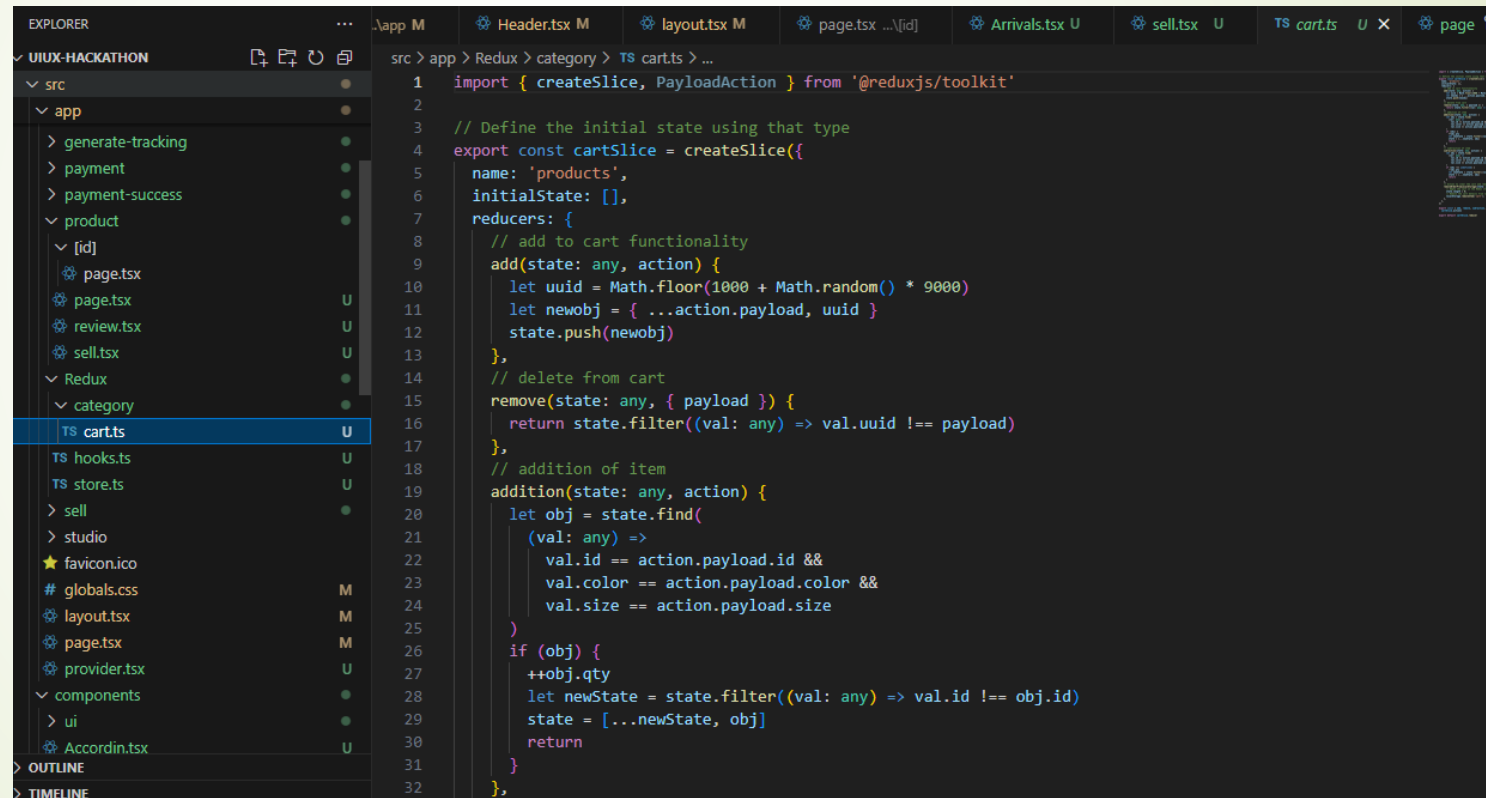
```
1 import { combineReducers, configureStore } from '@reduxjs/toolkit'
2 import cartSlice from './category/cart'
3 import storage from 'redux-persist/lib/storage';
4
5 import persistReducer from 'redux-persist/lib/persistReducer';
6
7 const persistconfig = {
8   key: "root",
9   version: 1,
10  storage,
11 }
12 const reducer = combineReducers({
13   cart: cartSlice
14 })
15 const persistedReducer = persistReducer(persistconfig, reducer)
16
17 export const store = configureStore({
18   reducer: persistedReducer,
19   middleware: (getDefaultMiddleware) =>
20     getDefaultMiddleware({ serializableCheck: false }),
21 })
22
23 // Infer the `RootState`, `AppDispatch`, and `AppStore` types from the store itself
24 export type RootState = ReturnType<typeof store.getState>
25 // Inferred type: {posts: PostsState, comments: CommentsState, users: UsersState}
26 export type AppDispatch = typeof store.dispatch
27 export type AppStore = typeof store
28
```

Hooks:



```
src > app > Redux > TS hooks.ts > ...  
1  import { useDispatch, useSelector } from 'react-redux'  
2  import type { AppDispatch, RootState } from './store'  
3  
4  // Use throughout your app instead of plain `useDispatch` and `useSelector`  
5  export const useAppDispatch = useDispatch.withTypes<AppDispatch>()  
6  export const useAppSelector = useSelector.withTypes<RootState>()
```

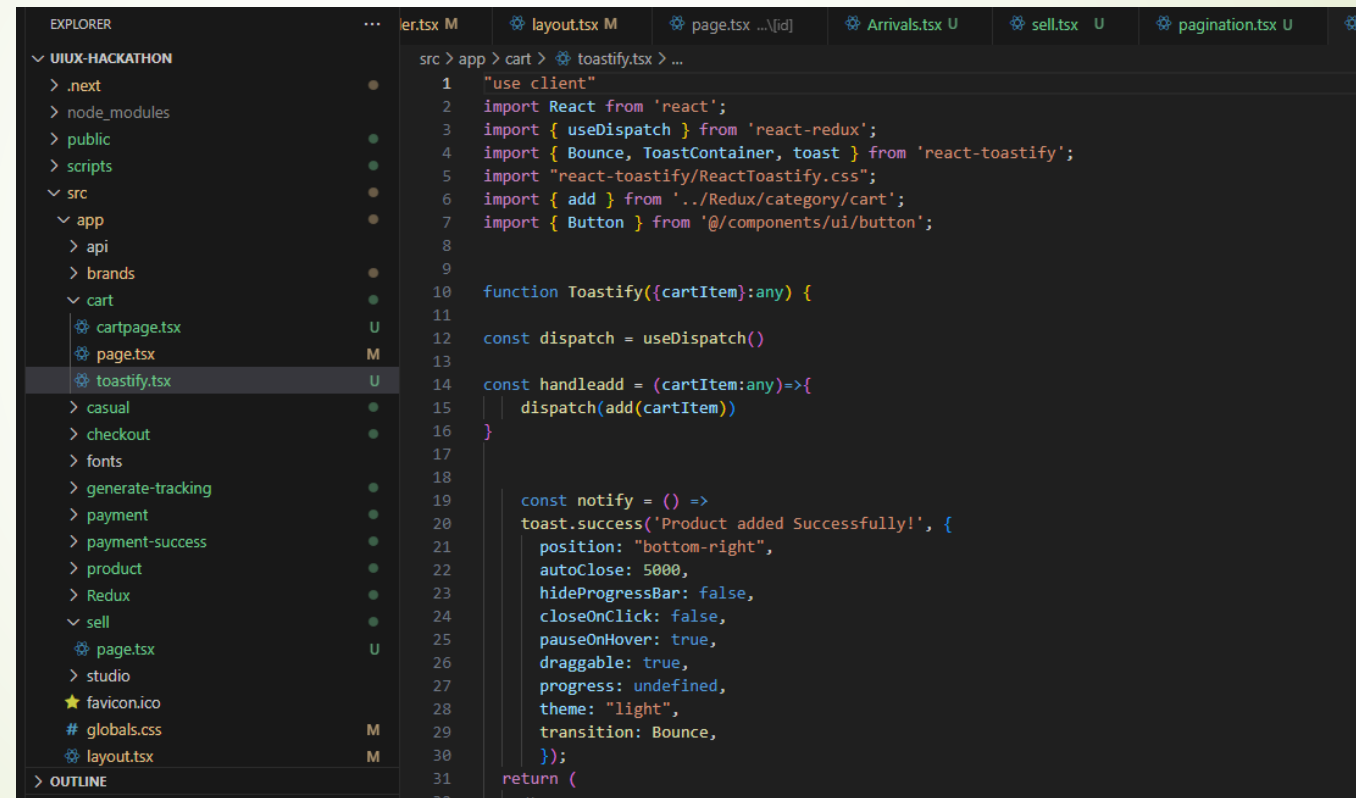
Cart Slice:



The screenshot shows a Visual Studio Code editor with a project named 'UIUX-HACKATHON'. The Explorer sidebar on the left shows the file structure, with 'src > app > Redux > category > TS cart.ts' selected. The main editor displays the code for 'TS cart.ts', which uses Redux Toolkit to create a 'cartSlice'.

```
1 import { createSlice, PayloadAction } from '@reduxjs/toolkit'
2
3 // Define the initial state using that type
4 export const cartSlice = createSlice({
5   name: 'products',
6   initialState: [],
7   reducers: {
8     // add to cart functionality
9     add(state: any, action) {
10       let uuid = Math.floor(1000 + Math.random() * 9000)
11       let newObj = { ...action.payload, uuid }
12       state.push(newObj)
13     },
14     // delete from cart
15     remove(state: any, { payload }) {
16       return state.filter((val: any) => val.uuid !== payload)
17     },
18     // addition of item
19     addition(state: any, action) {
20       let obj = state.find(
21         (val: any) =>
22           val.id === action.payload.id &&
23           val.color === action.payload.color &&
24           val.size === action.payload.size
25       )
26       if (obj) {
27         ++obj.qty
28         let newState = state.filter((val: any) => val.id !== obj.id)
29         state = [...newState, obj]
30         return
31       }
32     },
```

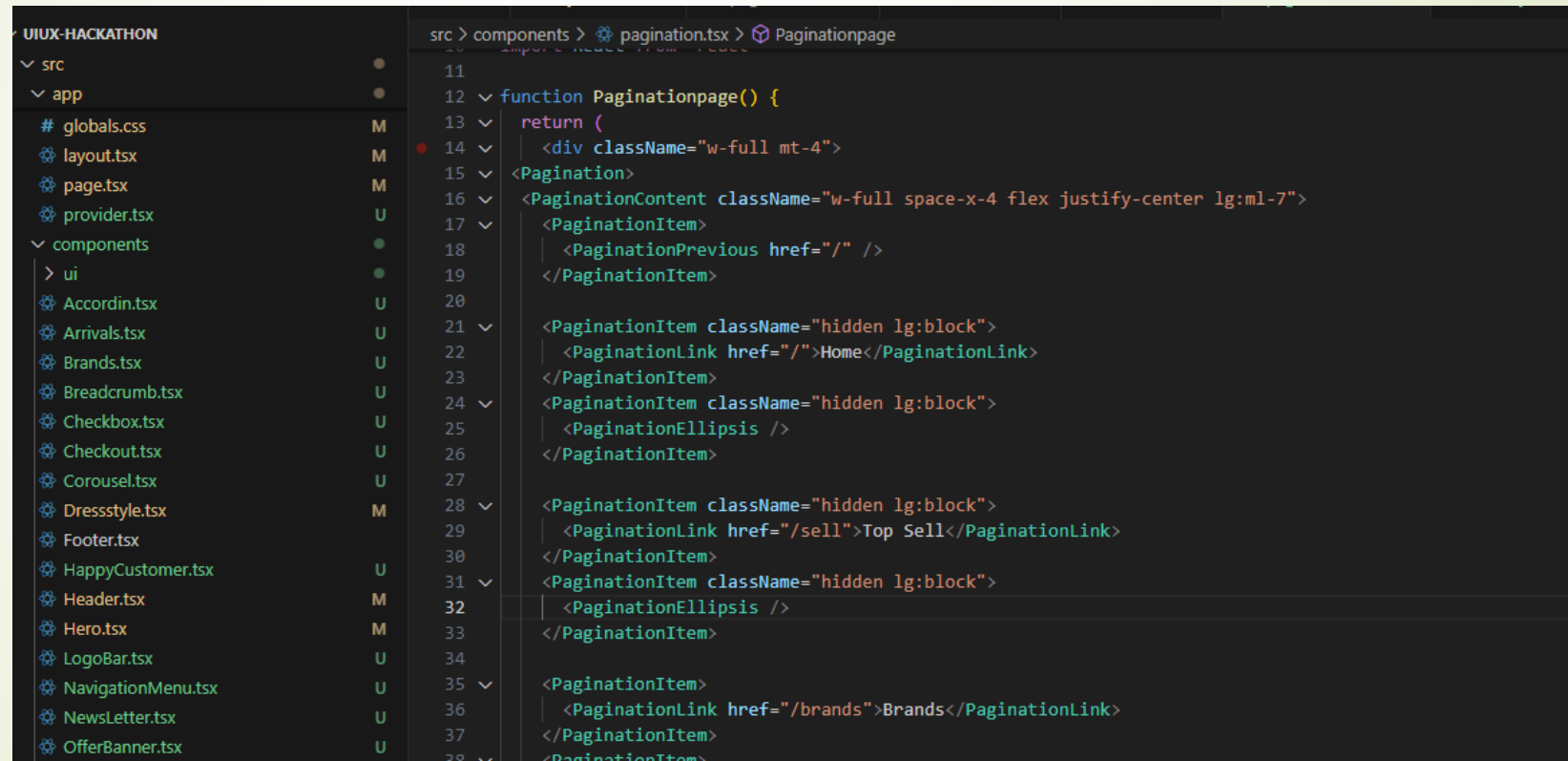
Toastify:



The screenshot shows a VS Code editor with a project named 'UIUX-HACKATHON'. The Explorer sidebar on the left lists the project structure, including folders like '.next', 'node_modules', 'public', 'scripts', 'src', 'app', 'api', 'brands', 'cart', 'casual', 'checkout', 'fonts', 'generate-tracking', 'payment', 'payment-success', 'product', 'Redux', 'sell', 'studio', 'favicon.ico', 'globals.css', and 'layout.tsx'. The 'cart' folder is expanded, showing 'cartpage.tsx', 'page.tsx', and 'toastify.tsx'. The 'toastify.tsx' file is selected and its content is displayed in the main editor area. The code in 'toastify.tsx' imports 'React', 'useDispatch' from 'react-redux', 'Bounce', 'ToastContainer', and 'toast' from 'react-toastify', and 'ReactToastify.css' from 'react-toastify/ReactToastify.css'. It also imports 'add' from '../Redux/category/cart' and 'Button' from '@components/ui/button'. The code defines a 'Toastify' function that takes a 'cartItem' and dispatches an 'add' action. It also defines a 'notify' function that uses 'toast.success' to show a success message with specific options like 'position', 'autoClose', 'hideProgressBar', 'closeOnClick', 'pauseOnHover', 'draggable', 'progress', 'theme', and 'transition'.

```
src > app > cart > toastify.tsx > ...
1  "use client"
2  import React from 'react';
3  import { useDispatch } from 'react-redux';
4  import { Bounce, ToastContainer, toast } from 'react-toastify';
5  import "react-toastify/ReactToastify.css";
6  import { add } from '../Redux/category/cart';
7  import { Button } from '@components/ui/button';
8
9
10 function Toastify({cartItem}:any) {
11
12   const dispatch = useDispatch()
13
14   const handleadd = (cartItem:any)=>{
15     dispatch(add(cartItem))
16   }
17
18
19   const notify = () =>
20   toast.success('Product added Successfully!', {
21     position: "bottom-right",
22     autoClose: 5000,
23     hideProgressBar: false,
24     closeOnClick: false,
25     pauseOnHover: true,
26     draggable: true,
27     progress: undefined,
28     theme: "light",
29     transition: Bounce,
30   });
31   return (
32     <
```

Pagination:



```
src > components > pagination.tsx > Paginationpage
11
12 function Paginationpage() {
13   return (
14     <div className="w-full mt-4">
15       <Pagination>
16         <PaginationContent className="w-full space-x-4 flex justify-center lg:ml-7">
17           <PaginationItem>
18             <PaginationPrevious href="/" />
19           </PaginationItem>
20
21           <PaginationItem className="hidden lg:block">
22             <PaginationLink href="/">Home</PaginationLink>
23           </PaginationItem>
24           <PaginationItem className="hidden lg:block">
25             <PaginationEllipsis />
26           </PaginationItem>
27
28           <PaginationItem className="hidden lg:block">
29             <PaginationLink href="/sell">Top Sell</PaginationLink>
30           </PaginationItem>
31           <PaginationItem className="hidden lg:block">
32             <PaginationEllipsis />
33           </PaginationItem>
34
35           <PaginationItem>
36             <PaginationLink href="/brands">Brands</PaginationLink>
37           </PaginationItem>
38           <PaginationItem>
```