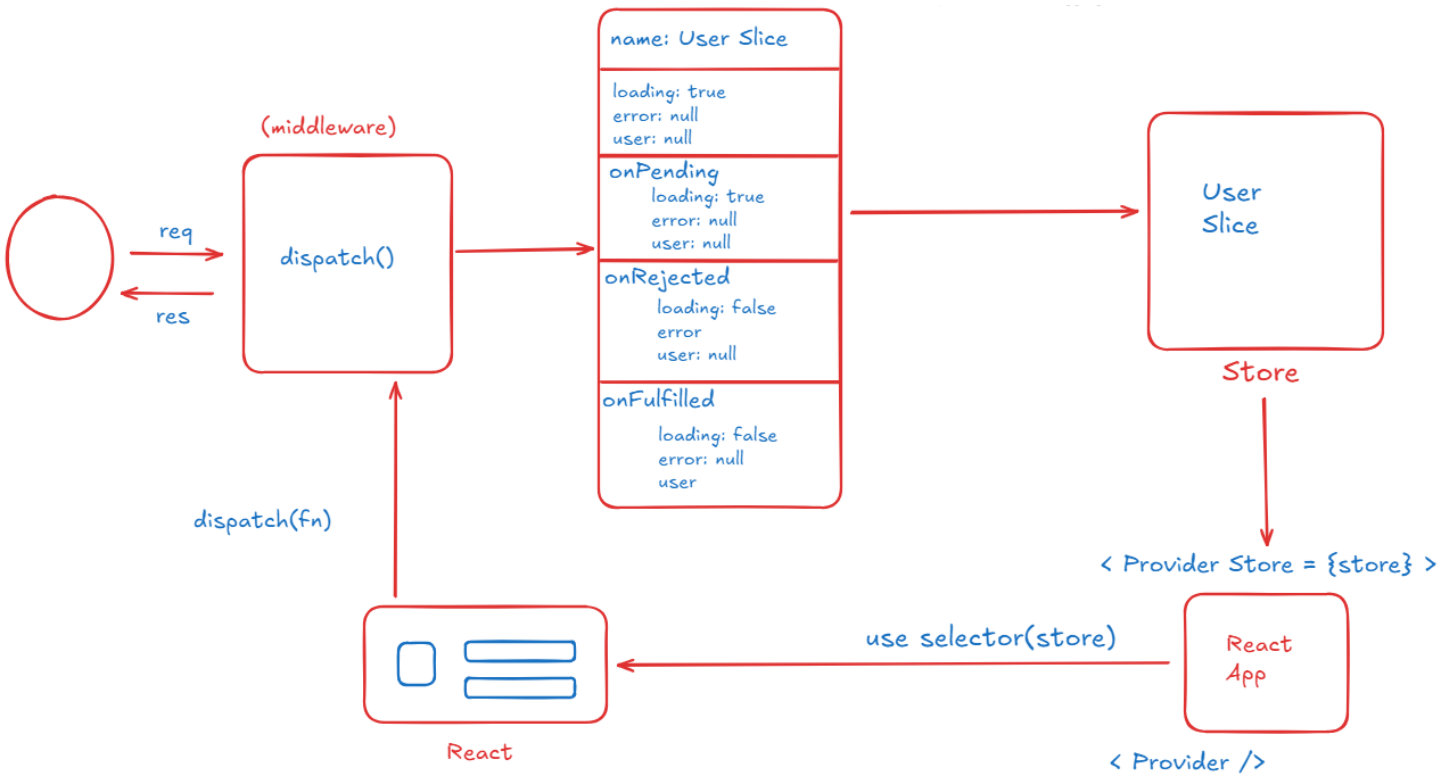
Async Redux & UI

Agenda

- Async task (Redux)
- UI (components)

Async Redux (middleware)



- We can not directly call dispatch an async task.
- If you have any state that comes asynchronously then you will add middleware. Why do we have to add a middleware function?

In dispatch, we have to call only one between these three functions (onPending, onRejected & onFulfilled) because this is async work so you don't have exact data in synchronous work you know exactly the data but in async we don't know the data so that's why we make middleware and it is the responsibility of middleware to call dispatch when data will come.

UserSlice.js

```
import { createSlice } from "@reduxjs/toolkit";
const userSlice = createSlice({
```

```
    name: "userSlice",
    initialState: {
        user: null,
        loading: true,
        error: null,
    },
    reducers: {
        onPending: (state) => {
            state.user = null;
            state.loading = true;
            state.error = null
        },
        onRejected: (state, params) => {
            state.user = null;
            state.loading = false;
            state.error = params.payload
        }
        ,
        onFulfilled: (state, params) => {
            state.user = params.payload;
            state.loading = false;
            state.error = null
        }
    }
})
const userActions=userSlice.actions;
export {userActions};
export default userSlice;
```

store.js

```
import { configureStore } from "@reduxjs/toolkit";
import counterSlice from "./slice/CounterSlice.js";
import counterInputSlice from "./slice/CounterInputSlice.js";
import userSlice from "./slice/UserSlice.js";
const store = configureStore({
    reducer: {
        counterSection: counterSlice.reducer,
        counterInputSlice: counterInputSlice.reducer,
        userSlice: userSlice.reducer,
    },

})
```

```
export default store;
```

fetchUserMiddleWare.js

```
import { userActions } from "../slice/UserSlice.js";

const fetchUserMiddleware = (params) => {
    // redux intrenally handle
    return async (dispatch) => {
        try {
            dispatch(userActions.onPending());
            // fetch user data from API and set it to the state
            const userResp = await
fetch(`https://jsonplaceholder.typicode.com/users/${params}`);
            const userData = await userResp.json();
            dispatch(userActions.onFulfilled(userData));
        } catch (err) {
            dispatch(userActions.onRejected(err));
        }
    }
}
export default fetchUserMiddleware;
```

- You will add one extra step when you work with an async task, which is middleware.
- So, the point to be noted is that you have some async task in which third-party work is involved and react directly doesn't know what changes are to be made in states so they don't call these three functions that's why you have to add middleware.

ReduxUserComponent.js

```
import React, { useEffect, useState } from 'react'
import { useSelector, useDispatch } from 'react-redux';
import fetchUserMiddleWare from '../redux/middleWare/fetchUserMiddleWare';

function ReduxUserComponent() {
    const { user, loading, error } = useSelector(store => store.userSlice);
    const dispatch = useDispatch();

    useEffect(() => {
        dispatch(fetchUserMiddleWare(4));
    }, [])
```

```
    if (loading) {
        return <div>Loading ... </div>
    }
    if (error) {
        return <div>Error: {error.message}</div>
    }
    return (
        <div style={{
            display: "flex",
            justifyContent: "center",
            alignItems: "center",
            width: "100vw",

        }}>
            <div>

                <h4>Name: {user.name}</h4>
                <h4>Phone: {user.phone}</h4>
            </div>
        </div>
    )
}

export default ReduxUserComponent;
```

App.js

```
import NormalCounter from "./Components/NormalCounter.jsx";
import NormalInputCounter from "./Components/NormalInputCounter.jsx";
import ReduxCounter from "./Components/ReduxCounter.jsx";
import ReduxInputContainer from "./Components/ReduxInputCounter.jsx";
import ReduxUserComponent from "./Components/ReduxUser.jsx";
import UserComponent from "./Components/User.jsx";
function App() {
  return (
    <>
      {/* <ReduxCounter /> */}
      {/* <NormalInputCounter /> */}
      {/* <ReduxInputContainer /> */}
      {/* <UserComponent /> */}
      <ReduxUserComponent/>
    </>
```

```
  )
}

export default App
```

- If there is an async task so instead of directly dispatching the actual actions out of these what you will do is add all the logic in your middleware and then you will call dispatch as simple as that.

Output

First `Loading...` will come and after that `data` will be visible on your screen.
Data fetched successfully the user data is displayed with name and phone number.

More Details:

UI (components)

Let's start our Jio clone project :

We have already seen how to do the setup of nextjs and shadcn in our project
After setting both we have to open the `app` directory and go to `layout.jsx` file.
Here we want to put our header on the top of the body.
To do this we are using the `V0` AI tool.
`V0` is a simple pro typing tool with a rough layout.
By using `V0` AI we are going to create our header section.

Header.jsx

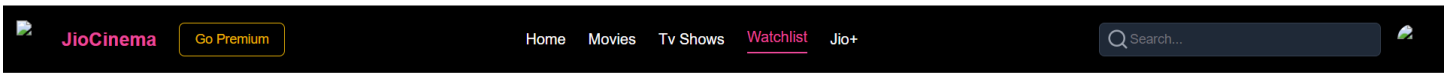
```
import { Search } from "lucide-react"
import Image from "next/image"
import Link from "next/link"

import { Button } from "@/components/ui/button"
import { Input } from "@/components/ui/input"


export default function Header() {

  return (
```

```
    <header className="bg-black text-white p-4 flex items-center justify-
between">
      <div className="flex items-center space-x-4 lg:space-x-6">
        <Link href="/" className="flex items-center space-x-2">
          <Image
            src="/placeholder.svg?height=40&width=40"
            alt="JioCinema Logo"
            width={40}
            height={40}
            className="w-10 h-10"
          />
          <span className="text-xl font-bold text-pink-500">JioCinema</span>
        </Link>
        <Button variant="outline" className="hidden sm:inline-flex bg-
transparent text-yellow-500 border-yellow-500 hover:bg-yellow-500 hover:text-
black">
          Go Premium
        </Button>
      </div>
      <nav className="hidden lg:flex items-center space-x-6">
        <Link href="/" className="hover:text-gray-300">Home</Link>
        <Link href="/movies" className="hover:text-gray-300">Movies</Link>
        <Link href="/tv" className="hover:text-gray-300">Tv Shows</Link>
        <Link href="/watchlist" className="text-pink-500 border-b-2 border-
pink-500 pb-1">Watchlist</Link>
        <Link href="/jio-plus" className="hover:text-gray-300">Jio+</Link>
      </nav>
      <div className="flex items-center space-x-4">
        <form className="relative">
          <Search className="absolute left-2 top-1/2 transform -translate-y-1/2
text-gray-400" />
          <Input
            type="search"
            placeholder="Search..."
            className="pl-8 bg-gray-800 border-gray-700 focus:border-pink-500
text-white w-[200px] lg:w-[300px]"
          />
        </form>
        <Button size="icon" variant="ghost" className="rounded-full">
          <Image
            src="/placeholder.svg?height=32&width=32"
            alt="User Avatar"
            width={32}
            height={32}
```

```
            className="rounded-full"
          />
          <span className="sr-only">User menu</span>
        </Button>
      </div>
    </header>
  )
}
```

Import Button component and Input Component from shadcn/ui.
You can also hide options screen resolution-wise.

Output



Next js

Next, we are going to build our routes.

The first page of the app is the Home page.

page.jsx

```
export default function Home(){
        return(
                <h1>Home Page</h1>
        )
}
```

Let's create all the folders in the app directory

- movies/watch
- tv/watch
- watchList
- jio-plus/watch

In every folder we created `page.jsx` .
After that's routes works properly.