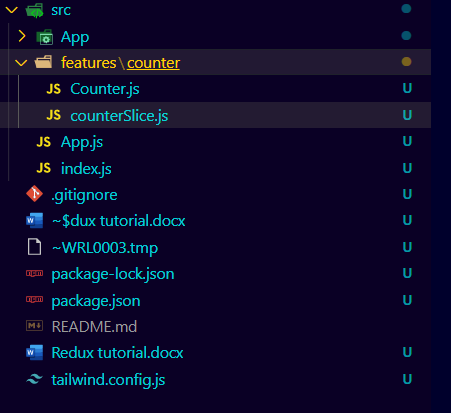
**Redux full course:**

Lets create folder structure of App first

Slice: the counter have a slice which divide the counter app into multiple slices

We have the following folder structure and we have have created simple counter peoject using redux toolkit:



1). **Index.js file**

import React from "react";

import ReactDom from 'react-dom';

import { BrowserRouter } from 'react-router-dom';

import App from './App';

import { store } from './App/Store';

import { Provider } from "react-redux";

ReactDom.render(

   <>

      <BrowserRouter>

         <Provider store={store}>

            <App />

         </Provider>

      </BrowserRouter>

   </>,

   document.getElementById("root")

);

2). App.js file

import React from 'react';

import "../node\_modules/bootstrap/dist/css/bootstrap.min.css";

import "../node\_modules/bootstrap/dist/js/bootstrap.bundle";

import Spinner from 'react-bootstrap/Spinner';

import Counter from './features/counter/Counter';

const App = () => {

    return (

        <>

            <section className=' d-flex justify-content-center align-items-center border ' style={{ height: "95vh" }}>

         <Counter />

            </section>

        </>

    )

}

export default App;

**3) Counter.jsx file:**

This is the file where our actual project exists:

import React from 'react'

import { useSelector, useDispatch } from 'react-redux';

import { increment, decrement, reset, incrementByAmount } from './counterSlice';

import { useState } from 'react';

const Counter = () => {

*//useState is used for the incrementtByamount action defines inside the redux store and for reset All action as well*

    const [incrementAmount, setincrementAmount] = useState(0);  *// this usestate is used for the incrementByAmount function we haev created inside counterSlice*

    const addValue = Number(incrementAmount) || 0; *//Number function will convert the increment amount into the number and check wheather it is 0 or not*

*//For reset All action*

    const resetAll = () => {

        setincrementAmount(0);

        dispatch(reset());

    }

*//to retrieve the data from the redux*

    const count = useSelector((state) => state.counter.count);

    const dispatch = useDispatch();

    return (

        <>

            <section className='border bg-dark p-5 rounded-lg'>

                <h1 className='text-center text-light text-danger'> {count} </h1>

                <div className='text-center py-5 m-4'>

                    <button className='btn btn-primary text-light m-3' onClick={() => dispatch(increment())}> + </button>

                    <button className='btn btn-primary text-light' onClick={() => dispatch(decrement())}> - </button>

                </div>

                <input className='my-auto mt-5 ml-5'

                    type="text"

                    value={incrementAmount}

                    onChange={(e) => setincrementAmount(e.target.value)}

                />

                <div className='p-5 text-center'>

                    <button className="btn btn-outline-primary" onClick={() => dispatch(incrementByAmount(addValue))}> Add amount </button>

                    <button className="btn btn-outline-primary" onClick={resetAll}> Reset </button>

                </div>

            </section>

        </>

    )

}

export default Counter;

export default Counter;

**4) counterSlice.js file**

This is the file where we have created the slices of the states of the project

import { createSlice } from "@reduxjs/toolkit";

const initialState = {

    count: 0

}

export const counterSlice = createSlice({

    name: 'counter',

    initialState,

    reducers: {

        increment: (state) =>{

            state.count +=1

        },

        decrement: (state) =>{

            state.count -=1

        },

        reset: (state)=>{

            state.count = 0

        },

        incrementByAmount: (state, action) =>{

            state.count += action.payload;

        }

    }

})

export const {increment, decrement, reset, incrementByAmount} = counterSlice.actions;

export default counterSlice.reducer;

**Store.js file:**

This is the file where our states is being stored and that store is called inside the index.js file.

import { configureStore } from '@reduxjs/toolkit';

import counterSlice from '../features/counter/counterSlice';

export const store = configureStore({

    reducer: {

        counter: counterSlice,

    }

})

Follow the following steps to understand the above concept

1. Look at the counterSlice file where we have our states and action stored. Inside this file we have initial state at the top in which we have passes the object counter: 0. Initially its value will be 0.
2. We have created the function with the name of createSlice where we passed the object these objects will be following.
3. The firstOne will be initial state, secondOne will be the name of the state like this name:’” counter”.
4. Then we will pass the action like we use objects and above we have created the four actions increment, decrement, reset, IncrementByAmount.
5. We will use these action inside reducers like this reducer : {action1, action2}.
6. Inside increment we have passed the anonymous callBack function having the argument state.
7. We will use (.) operator infront of the argument we have passed inside the increment method like this state.count+=1; mean increasing the initial state by 1.
8. Dame process for the decrement
9. Inside reset we have passed the initial state which was 0.
10. Inside incrementByAmount we have passed two things state and action here we did state.count+=action.payload means we are increasing the state by the action.payload here payload means using any other additional information
11. At the end we have destructured all the action with the counterSlice.actions and at the we have exported them.

Check other files and folders by yourself

**Project 2 Post in React Redux:**

We just have to remove the counter feature from the above code.

See the in e\lesson2 foder

**Lesson3 React Redux:**

**Async Logics and thunks:**

Thunk means a piece of code that does some delayed work.

We are going to study how to use async logic in redux and fetching the api instead of writing the api data inside the initial state.