npm install redux redux-saga react-redux axios

export const FETCH\_USER\_REQUEST = "FETCH\_USER\_REQUEST";

export const FETCH\_USER\_SUCCESS = "FETCH\_USER\_SUCCESS";

export const FETCH\_USER\_FAILURE = "FETCH\_USER\_FAILURE";

export const fetchUserRequest = () => ({

  type: FETCH\_USER\_REQUEST,

});

export const fetchUserSuccess = (user) => ({

  type: FETCH\_USER\_SUCCESS,

  payload: user,

});

export const fetchUserFailure = (error) => ({

  type: FETCH\_USER\_FAILURE,

  payload: error,

});

import { FETCH\_USER\_REQUEST, FETCH\_USER\_SUCCESS, FETCH\_USER\_FAILURE } from "./actions";

const initialState = {

  loading: false,

  user: null,

  error: null,

};

const userReducer = (state = initialState, action) => {

  switch (action.type) {

    case FETCH\_USER\_REQUEST:

      return { ...state, loading: true };

    case FETCH\_USER\_SUCCESS:

      return { loading: false, user: action.payload, error: null };

    case FETCH\_USER\_FAILURE:

      return { loading: false, user: null, error: action.payload };

    default:

      return state;

  }

};

export default userReducer;

import { takeLatest, call, put } from "redux-saga/effects";

import axios from "axios";

import { FETCH\_USER\_REQUEST, fetchUserSuccess, fetchUserFailure } from "./actions";

// API Call

const fetchUserApi = () => axios.get("https://jsonplaceholder.typicode.com/users/1");

// Worker Saga: Handles API Call

function\* fetchUserSaga() {

  try {

    const response = yield call(fetchUserApi);

    yield put(fetchUserSuccess(response.data));

  } catch (error) {

    yield put(fetchUserFailure(error.message));

  }

}

// Watcher Saga: Listens for Actions

export function\* watchUserSaga() {

  yield takeLatest(FETCH\_USER\_REQUEST, fetchUserSaga);

}

import { createStore, applyMiddleware } from "redux";

import createSagaMiddleware from "redux-saga";

import userReducer from "./reducer";

import { watchUserSaga } from "./sagas";

// Create Saga Middleware

const sagaMiddleware = createSagaMiddleware();

// Create Store

const store = createStore(userReducer, applyMiddleware(sagaMiddleware));

// Run Saga

sagaMiddleware.run(watchUserSaga);

export default store;

import React from "react";

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

import { fetchUserRequest } from "./actions";

const App = () => {

  const dispatch = useDispatch();

  const { loading, user, error } = useSelector((state) => state);

  return (

    <div>

      <h1>Redux-Saga Example</h1>

      <button onClick={() => dispatch(fetchUserRequest())}>

        {loading ? "Loading..." : "Fetch User"}

      </button>

      {user && <p>Name: {user.name}</p>}

      {error && <p style={{ color: "red" }}>Error: {error}</p>}

    </div>

  );

};

export default App;

import React from "react";

import ReactDOM from "react-dom";

import { Provider } from "react-redux";

import store from "./store";

import App from "./App";

ReactDOM.render(

  <Provider store={store}>

    <App />

  </Provider>,

  document.getElementById("root")

);

import { all } from "redux-saga/effects";

import { watchUserSaga } from "./sagas";

import { watchOtherSaga } from "./otherSaga";

export default function\* rootSaga() {

  yield all([watchUserSaga(), watchOtherSaga()]);

}

import { eventChannel } from "redux-saga";

import { take, put } from "redux-saga/effects";

// Create Event Channel for WebSocket

function createWebSocketChannel(socket) {

  return eventChannel((emit) => {

    socket.on("message", (data) => {

      emit({ type: "NEW\_MESSAGE", payload: data });

    });

    return () => socket.close();

  });

}

// Saga to Handle WebSocket Messages

function\* watchWebSocket(socket) {

  const channel = yield call(createWebSocketChannel, socket);

  while (true) {

    const action = yield take(channel);

    yield put(action);

  }

}

















