Handle Api Calls With Redux Toolkit and RTK Query

* Project Structure for perform API calls with RTK Query
* React-Project
* |
* |---> Public
* |
* |---> src
* |     |
* |     |---> AuthConfig
* |     |       |
* |     |       |---> app
* |     |       |      |
* |     |       |      | ---> store.js
* |     |       |
* |     |       |---> features
* |     |       |      |
* |     |       |      |---> authSlice.js
* |     |       |
* |     |       |---> services
* |     |       |      |
* |     |       |      |---> localStorageServices.js
* |     |       |      |
* |     |       |      |---> userAuthApi.js
* |     |       |
* |     |       |---> baseUrls.js
* |     |       |

**Step 1**

* Install Requirements

**Reference Link :**

<https://redux-toolkit.js.org/tutorials/quick-start>

Install this library

>>> npm install @reduxjs/toolkit react-redux

RTK Query

<https://redux-toolkit.js.org/tutorials/rtk-query>

**Step 2**

Setup Base Url inside **AuthConfig**

**baseUrls.js**

// Setup baseUrl ( server IP with 8000 django port )

export const baseUrl = 'http://192.168.1.13:8000/'

**Step 3**

* Create **authSlice.js** inside **features** **folder**

Note : It’s a Reducer ( for redux state management )

**authSlice.js**

import { createSlice } from '@reduxjs/toolkit'

const initialState = {

    access\_token: null,

    refresh\_token: null

}

export const authSlice = createSlice({

    name: 'auth\_token',

    initialState,

    reducers: {

        setUserToken: (state, action) => {

            state.access\_token = action.payload.access\_token

            state.refresh\_token = action.payload.refresh\_token

        },

        unSetUserToken: (state, action) => {

            state.access\_token = null

            state.refresh\_token = null

        },

    },

})

export const { setUserToken ,unSetUserToken } = authSlice.actions

export default authSlice.reducer

**Step 4**

Create **LocalStorageServices.js** inside a **services** folder

Note : it is used to store access\_token and refresh\_token also stores user details inside a Local Storage

**LocalStorageServices.js**

// to store tokens into a local storage

const storeToken = (value) => {

    if (value) {

        const { access, refresh } = value

        localStorage.setItem('access\_token', access)

        localStorage.setItem('refresh\_token', refresh)

    }

}

// to access tokens from localstorage

const getToken = () => {

    let access\_token = localStorage.getItem('access\_token')

    let refresh\_token = localStorage.getItem('refresh\_token')

    return { access\_token, refresh\_token }

}

// to remove token from local storage

const removeToken = () => {

    localStorage.removeItem('access\_token')

    localStorage.removeItem('refresh\_token')

}

const storeUserProfile = (userData) => {

    if (userData) {

        const { id, name, email, designation, user\_roll, user\_profile, is\_active, is\_staff, is\_admin, dashboard\_perms, sheets\_perms, uploads\_perms, users\_perms,

            activity\_perms, other\_perms, created\_at, updated\_at } = userData

        localStorage.setItem('userId', id)

        localStorage.setItem('userName', name)

        localStorage.setItem('userEmail', email)

        localStorage.setItem('userDesignation', designation)

        localStorage.setItem('userRoll', user\_roll)

        localStorage.setItem('userProfile', user\_profile)

        localStorage.setItem('userIsActive', is\_active)

        localStorage.setItem('userIsStaff', is\_staff)

        localStorage.setItem('userIsAdmin', is\_admin)

        localStorage.setItem('userDashboardPerms', dashboard\_perms)

        localStorage.setItem('userSheetPerms', sheets\_perms)

        localStorage.setItem('userUplooadPerms', uploads\_perms)

        localStorage.setItem('userUsersPerms', users\_perms)

        localStorage.setItem('userActivityPerms', activity\_perms)

        localStorage.setItem('userOtherPerms', other\_perms)

        localStorage.setItem('userCreatedAt', created\_at)

        localStorage.setItem('userUpdatedAt', updated\_at)

    }

}

const getUserProfile = () => {

    let userId = localStorage.getItem('userId')

    let userName = localStorage.getItem('userName')

    let userEmail = localStorage.getItem('userEmail')

    let userDesignation = localStorage.getItem('userDesignation')

    let userRoll = localStorage.getItem('userRoll')

    let userProfile = localStorage.getItem('userProfile')

    let userIsActive = localStorage.getItem('userIsActive')

    let userIsStaff = localStorage.getItem('userIsStaff')

    let userIsAdmin = localStorage.getItem('userIsAdmin')

    let userDashboardPerms = localStorage.getItem('userDashboardPerms')

    let userSheetPerms = localStorage.getItem('userSheetPerms')

    let userUplooadPerms = localStorage.getItem('userUplooadPerms')

    let userUsersPerms = localStorage.getItem('userUsersPerms')

    let userActivityPerms = localStorage.getItem('userActivityPerms')

    let userOtherPerms = localStorage.getItem('userOtherPerms')

    let userCreatedAt = localStorage.getItem('userCreatedAt')

    let userUpdatedAt = localStorage.getItem('userUpdatedAt')

    return {

        userId, userName, userEmail, userDesignation, userRoll, userProfile, userIsActive,

        userIsStaff, userIsAdmin, userDashboardPerms, userSheetPerms, userUplooadPerms, userUsersPerms,

        userActivityPerms, userOtherPerms, userCreatedAt, userUpdatedAt

    }

}

const removeAllLocalStorageData = () => {

    localStorage.removeItem('access\_token')

    localStorage.removeItem('refresh\_token')

    localStorage.removeItem('userId')

    localStorage.removeItem('userName')

    localStorage.removeItem('userEmail')

    localStorage.removeItem('userDesignation')

    localStorage.removeItem('userRoll')

    localStorage.removeItem('userProfile')

    localStorage.removeItem('userIsActive')

    localStorage.removeItem('userIsStaff')

    localStorage.removeItem('userIsAdmin')

    localStorage.removeItem('userDashboardPerms')

    localStorage.removeItem('userSheetPerms')

    localStorage.removeItem('userUplooadPerms')

    localStorage.removeItem('userUsersPerms')

    localStorage.removeItem('userActivityPerms')

    localStorage.removeItem('userOtherPerms')

    localStorage.removeItem('userCreatedAt')

    localStorage.removeItem('userUpdatedAt')

}

const storeNewAccessToken = (accessJson) => {

    const { access } = accessJson

    localStorage.setItem('access\_token', access)

}

export { storeToken, getToken, removeToken, storeUserProfile, getUserProfile, removeAllLocalStorageData, storeNewAccessToken }

**Step 5**

Inside **services** folder **userAuthApi.js**

**userAuthApi.js**

import { createApi, fetchBaseQuery } from '@reduxjs/toolkit/query/react'

import { baseUrl } from '../baseUrls'

import { setUserToken, unSetUserToken } from '../features/authSlice'

import { getToken, removeToken, removeAllLocalStorageData } from './localStorageServices'

const baseQuery = fetchBaseQuery({

    baseUrl: baseUrl //set base url

})

const baseQueryWithReauth = async (args, api, extraOptions) => {

    let result = await baseQuery(args, api, extraOptions)

    const { refresh\_token } = getToken()

    if (result?.error?.status === 401 | result?.error?.status === 403) {

        const urlHeader = {

            url: 'api/auth/token/refresh/',

            method: "POST",

            body: { 'refresh': refresh\_token },

            headers: {

                'Content-type': "application/json"

            }

        }

        const refreshResult = await baseQuery(urlHeader, api, extraOptions)

        if (refreshResult.data) {

            localStorage.setItem('access\_token', refreshResult.data.access)

            api.dispatch(setUserToken({ access\_token: refreshResult.data.access, refresh\_token: refresh\_token }))

        } else {

            removeAllLocalStorageData()

            alert('Session time out !')

        }

    }

    return result

}

export const userAuthApi = createApi({

    reducerPath: 'userAuthApi',

    baseQuery: baseQueryWithReauth,

    endpoints: (builder) => ({

        // 1 - register user

        registerUser: builder.mutation({

            query: ({ access\_token, userRegisterData }) => {

                return {

                    url: 'api/auth/register-user/',

                    method: "POST",

                    body: userRegisterData,

                    headers: {

                        'Content-type': "application/json",

                        'authorization': `Bearer ${access\_token}`

                    }

                }

            }

        }),

        // 2 - Login user

        loginUser: builder.mutation({

            query: (loginData) => {

                return {

                    url: 'api/auth/user-login/',

                    method: "POST",

                    body: loginData,

                    headers: {

                        'Content-type': "application/json"

                    }

                }

            }

        }),

    }),

})

export const { useRegisterUserMutation, useLoginUserMutation } = userAuthApi

Step 6

Create **app** folder inside **src** folder

Inside **app** folder create **store.js** file

Import your servevices and implement

**store.js**

import { configureStore } from '@reduxjs/toolkit'

import { setupListeners } from '@reduxjs/toolkit/query'

import { userAuthApi } from '../services/userAuthApi'

// import auth slic

import authReducer from '../features/authSlice'

export const store = configureStore({

    reducer: {

        [userAuthApi.reducerPath]: userAuthApi.reducer,

        // setup auth slice

        auth:authReducer

    },

    middleware: (getDefaultMiddleware) =>

        getDefaultMiddleware().concat(userAuthApi.middleware),

})

setupListeners(store.dispatch)

Step 7

### Wrap your application with the Provider

Open **index.js** and add this code to wrap redux provider

Import provider and import store in index.js

**Index.js**

import React from 'react';

import ReactDOM from 'react-dom/client';

import './index.css';

import App from './App';

import reportWebVitals from './reportWebVitals';

import { Provider } from 'react-redux';

import { store } from './AuthConfig/app/store';

const root = ReactDOM.createRoot(document.getElementById('root'));

root.render(

  <React.StrictMode>

    <Provider store={store} >

      {/\* Add Routing from here  \*/}

      <App />

    </Provider>

  </React.StrictMode>

);

reportWebVitals();

Step 8

Call Login API inside a Login.js

Login .js

import React, { useState } from 'react';

import { Form, Input, Typography, Button, Alert } from 'antd';

import { UserOutlined, LockOutlined, EyeTwoTone, EyeInvisibleOutlined, } from '@ant-design/icons'

import './Login.css';

// Import APi Hook for Login

import { useLoginUserMutation } from '../AuthConfig/services/userAuthApi';

import { useDispatch } from 'react-redux';

import { getToken, storeToken, storeUserProfile } from '../AuthConfig/services/localStorageServices';

import { setUserToken } from '../AuthConfig/features/authSlice';

const Login = () => {

    const [userEmail, setUserEmail] = useState('') //store user email

    const [userPassword, setUserPassword] = useState('') //store user password

    // State to show Alert Mesg for login success and login failure

    const [loginError, setLoginError] = useState(

        {

            state: false,

            msg: "",

            error: ""

        }

    );

    // Destructure APi Mutation for api call

    const [loginUser, { isLoading }] = useLoginUserMutation();

    const dispatch = useDispatch();

    const handleLoginForm = async (e) => {

        e.preventDefault();

        // prepare data for api call

        if (!userEmail && !userPassword) {

            return false

        }

        else {

            const loginData = {

                email: userEmail,

                password: userPassword

            }

            const res = await loginUser(loginData);

            // console.log("res :", res)

            if (res?.data) {

                // show success message

                setLoginError(

                    {

                        state: true,

                        msg: "Login Successful",

                        error: "success"

                    }

                )

                setTimeout(() => {

                    handleClearErrorMsg()

                }, 5000)

                // store token in local storage

                storeToken(res.data.token)

                // store user details in local storage

                storeUserProfile(res.data.user\_data)

                // Store token into a redux state

                let { access\_token, refresh\_token } = getToken()

                dispatch(setUserToken({ access\_token: access\_token, refresh\_token: refresh\_token }))

                // Navigation to dashboard page

            }

            else if (res?.error) {

                console.log("Error :", res.error.data.error)

                // show success message

                setLoginError(

                    {

                        state: true,

                        msg: res.error.data.error,

                        error: "error"

                    }

                )

            }

        }

    }

    // clear Error msg

    const handleClearErrorMsg = () => {

        setLoginError({

            state: false,

            msg: "",

            error: ""

        })

    }

    return (

        <div className='login-page-container'>

            <div className='login-form-container'>

                <Form className='loginForm' id='loginForm' onSubmitCapture={handleLoginForm} >

                    <div className='errorBox'>

                        <Typography.Title style={{ textAlign: "center" }} className='form-heading'>IDE TeleRealty Manager</Typography.Title>

                    </div>

                    {/\* Alert Msg For Login and Error  \*/}

                    {loginError.state ?

                        <Alert message={loginError.msg} style={{ marginBottom: 20 }} type={loginError.error} closable onClose={handleClearErrorMsg} />

                        : ''

                    }

                    {/\* Login Form  \*/}

                    <div className='form-content'>

                        <Form.Item

                            rules={[{ required: true, type: "email", message: "Please enter valid email" }]}

                            name={"email"}>

                            <Input className='mail' placeholder="Email" prefix={<UserOutlined />}

                                // onChange={(e) => { setUserEmail(e.target.value) }}

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

                            />

                        </Form.Item>

                    </div>

                    <div className='form-content'>

                        <Form.Item

                            rules={[{ required: true, max: 15, min: 5, message: "Please enter valid password" }]}

                            name={"password"}>

                            <Input.Password className='mail' placeholder="password" prefix={<LockOutlined />}

                                iconRender={(visible) => (visible ? <EyeTwoTone /> : <EyeInvisibleOutlined />)}

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

                            />

                        </Form.Item>

                    </div>

                    <div className='form-btn-content'>

                        <Button type="primary" htmlType="submit" className='login-btn'>Login</Button>

                    </div>

                </Form>

            </div>

            {/\* Login footer  \*/}

            <div className="login-footer">

                <Typography className='login-footer-text'>

                    Powered By ItsDigitalEra.in

                </Typography>

            </div>

        </div>

    );

}

export default Login;