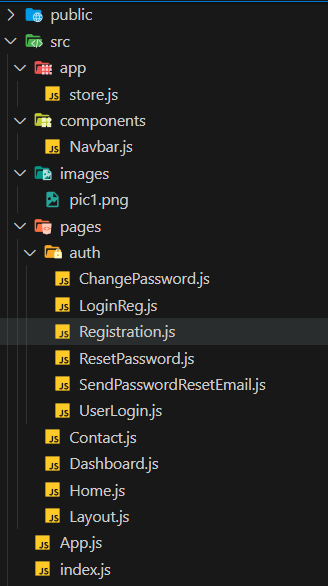
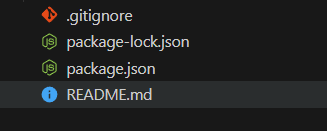
DJANGO-REACT AUTHENTICATION

* **Frontend Project Structure**





* **To run the frontend project**
* Run below commands in terminal

*>> npm i*

*>> npm start*

* **Installing redux**
* Run below commands in terminal

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

* Here we Want to use RTK from redux toolkit instead of fetch and axios.
* **Add the service to your store**
* We followed this documentation

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

* Create a app/store.js file inside src directory and paste the below codes

import { configureStore } from '@reduxjs/toolkit'

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

import { pokemonApi } from './services/pokemon'

export const store = configureStore({

  reducer: {

    [pokemonApi.reducerPath]: pokemonApi.reducer,

  },

  middleware: (getDefaultMiddleware) =>

    getDefaultMiddleware().concat(pokemonApi.middleware),

})

setupListeners(store.dispatch)

* Here we will remove the pokemon and will use our own service.
* **Wrap your application with the Provider**
* In src/index.js file wrap your <App /> with provider

import React from 'react';

import ReactDOM from 'react-dom';

import App from './App';

import reportWebVitals from './reportWebVitals';

import { Provider } from 'react-redux'

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

ReactDOM.render(

  <React.StrictMode>

    <Provider store={store}>

      <App />

    </Provider>

  </React.StrictMode>,

  document.getElementById('root')

);

reportWebVitals();

* **Create an API service**
* Create a services/userAuthApi.js file inside src directory and paste the below codes

// Need to use the React-specific entry point to import createApi

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

// Define a service using a base URL and expected endpoints

export const userAuthApi = createApi({

    reducerPath: 'userAuthApi',

    baseQuery: fetchBaseQuery({ baseUrl: 'http://127.0.0.1:8000/' }),

    endpoints: (builder) => ({

        registerUser: builder.mutation({

            query: (user) => {

                return {

                    url: 'register/',

                    method: 'POST',

                    body: user,

                    headers: {

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

                    }

                }

            }

        })

    }),

})

export const { useRegisterUserMutation } = userAuthApi

* In redux, ‘mutation’ is used when we have to modify (create/delete/update) the data and ‘query’ to read the data.
* Backend url : <http://127.0.0.1:8000/>
* ‘useRegisterUserMutation’ Its is a hook provided by redux by default.
* registerUser is a endpoint for user registration

Cookies

* npm install js-cookie
* Create a cookies.js file inside src/services directory and paste the below code
* import Cookies from 'js-cookie';
* // Function to set cookies
* const storeToken = (value) => {
* if (value) {
* console.log("Store Token");
* const { access, refresh } = value;
* Cookies.set("access\_token", access);
* Cookies.set("refresh\_token", refresh);
* }
* };
* // Function to get cookies
* const getToken = () => {
* let access\_token = Cookies.get('access\_token');
* let refresh\_token = Cookies.get('refresh\_token');
* return { access\_token, refresh\_token };
* };
* // Function to remove cookies
* const removeToken = () => {
* Cookies.remove('access\_token');
* Cookies.remove('refresh\_token');
* };
* npx browserslist@latest --update-db (to install the updated browserlist)

"Browserslist: caniuse-lite is outdated. Please run," is related to the Browserslist configuration in your project. Browserslist is a tool that allows you to specify the browsers and their versions that your project should support.

### Now we have to configure if there is access\_token in the cookies then disable the /login and /register page. To do this we have to Create a Redux State Slice

* Create a feature/authSlice.js inside src directory and paste the below mentioned code
* import { createSlice } from '@reduxjs/toolkit'
* const initialState = {
* access\_token : null,
* }
* export const authSlice = createSlice({
* name: 'auth\_token',
* initialState,
* reducers: {
* setUserToken : (state,action)=>{
* state.access\_token = action.payload.access\_token
* },
* unSetUserToken : (state,action)=>{
* state.access\_token = action.payload.access\_token
* }
* },
* })
* // Action creators are generated for each case reducer function
* export const { setUserToken, unSetUserToken } = counterSlice.actions
* export default counterSlice.reducer
* In app/store.js, you have to mention the above reducers

import { configureStore } from '@reduxjs/toolkit'

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

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

import authReducer from '../features/authSlice' // add this

export const store = configureStore({

  reducer: {

    [userAuthApi.reducerPath]: userAuthApi.reducer,

    auth : authReducer // add this

  },

    middleware: (getDefaultMiddleware) =>

    getDefaultMiddleware().concat(userAuthApi.middleware),

})

setupListeners(store.dispatch)

* Now you have to change all the files in which we set the token

1. userVerify.js

import { TextField, Button, Box, Alert } from '@mui/material';

import { useState, useEffect } from 'react';  // add useEffect

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

// login hook

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

import { storeToken, getToken } from '../../services/cookies';

// dispatch // add this

import { useDispatch } from 'react-redux';

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

const UserVerify = () => {

    // To get the unique token

    // const { token } = useParams();

    const urlParams = new URLSearchParams(window.location.search);

    const token = urlParams.get('token');

    // const token = new URLSearchParams().get("token");

    // setToken(token);

    // console.log("tokennn",token)

    const [error, setError] = useState({

        status: false,

        msg: "",

        type: ""

    })

    const navigate = useNavigate();

    // Used to store backend errors

    const [server\_error, setServerError] = useState({})

    const dispatch = useDispatch()   // add this

    // hook's functionality

    const [verifyOTP, { isLoading }] = useVerifyOTPMutation()

    const handleSubmit = async (e) => {

        e.preventDefault();

        const data = new FormData(e.currentTarget);

        // ------------- get form data -------------

        // console.log("Token is",token)

        const actualData = {

            otp: data.get('otp'),

            token: token

        }

        // start-form frontend validation : These are used for showing form validation

        if (!actualData.otp) {

            setError({ status: true, msg: "OTP is Required", type: 'error' })

            return;

        }

        // end-form validation

        // ------------- sending form data to userAuthApi -------------

        const res = await verifyOTP(actualData)

        if (res.error) {

            console.log(res.error)

            setServerError(res.error.data.errors)

        }

        if (res.data) {

            console.log(res.data)

            storeToken(res.data.token)

            // setting in state // add this

            let { access\_token } = getToken()

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

            navigate('/dashboard')

        }

    }

    // but if someone refresh the page then redux lose its state, to avoid this // add this

    let { access\_token } = getToken()

    useEffect(() => {

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

    }, [access\_token, dispatch])

    return <>

        <Box component='form' noValidate sx={{ mt: 1 }} id='login-form' onSubmit={handleSubmit}>

            <TextField margin='normal' required fullWidth id='otp' name='otp' label='OTP' />

            <Button type='submit' variant='contained' sx={{ mt: 3, mb: 2, px: 5 }}>Submit</Button>

        </Box>

        {/\* frontend validation error \*/}

        {error.status ? <Alert severity={error.type} sx={{ mt: 3 }}>{error.msg}</Alert> : ''}

        {/\* backend validation error \*/}

        {server\_error.token ? <Alert severity='error'>Timed Out! Try Again</Alert> : ''}

        {server\_error.non\_field\_errors ? <Alert severity='error'>{server\_error.non\_field\_errors[0]}</Alert> : ''}

    </>;

};

export default UserVerify;

1. userLogin.js

import { TextField, Button, Box, Alert } from '@mui/material';

import { useEffect, useState } from 'react';

import { NavLink, useNavigate } from 'react-router-dom';

import { getToken, storeToken } from '../../services/cookies';

// login hook

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

// dispatch // add this

import { useDispatch } from 'react-redux';

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

const UserLogin = () => {

  const [error, setError] = useState({

    status: false,

    msg: "",

    type: ""

  })

  const navigate = useNavigate();

  // Used to store backend errors

  const [server\_error, setServerError] = useState({})

  const dispatch = useDispatch()   // add this

  // hook's functionality

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

  const handleSubmit = async (e) => {

    e.preventDefault();

    const data = new FormData(e.currentTarget);

    // ------------- get form data -------------

    const actualData = {

      email: data.get('email'),

      password: data.get('password'),

    }

    // start-form frontend validation : These are used for showing form validation

    if (!(actualData.email && actualData.password)) {

      setError({ status: true, msg: "All Fields are Required", type: 'error' })

      return;

    }

    // end-form validation

    // ------------- sending form data to userAuthApi -------------

    const res = await loginUser(actualData)

    if (res.error) {

      // console.log(res.error)

      setServerError(res.error.data.errors)

    }

    if (res.data) {

      // console.log(res.data)

      if (res.data.verify){

        storeToken(res.data.token)

        // setting in state // add this

        let {access\_token } = getToken()

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

        navigate('/dashboard')

      }else{

        navigate(`/verify/?token=${res.data.token}`)

      }

    }

  }

  // but if someone refresh the page then redux lose its state, to avoid this // add this

  let {access\_token } = getToken()

  useEffect(() =>{

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

  }, [access\_token, dispatch] )

  return <>

    <Box component='form' noValidate sx={{ mt: 1 }} id='login-form' onSubmit={handleSubmit}>

      <TextField margin='normal' required fullWidth id='email' name='email' label='Email Address' />

      {server\_error.email ? <i>{server\_error.email[0]}</i> : ""}

      <TextField margin='normal' required fullWidth id='password' name='password' label='Password' type='password' />

      <Box textAlign='center'>

        {/\* <Button type='submit' variant='contained' sx={{ mt: 3, mb: 2, px: 5 }}>Login</Button> \*/}

        {isLoading ? <Button type='submit' variant='contained' disabled sx={{ mt: 3, mb: 2, px: 5 }}>Login</Button> :

        <Button type='submit' variant='contained' sx={{ mt: 3, mb: 2, px: 5 }}>Login</Button> }

      </Box>

      <NavLink to='/sendpasswordresetemail' >Forgot Password ?</NavLink> <br />

      <NavLink to='/register' >Don't have account?</NavLink>

      {/\* frontend validation error \*/}

      {error.status ? <Alert severity={error.type} sx={{ mt: 3 }}>{error.msg}</Alert> : ''}

      {/\* backend validation error \*/}

      {server\_error.non\_field\_errors ? <Alert severity='error'>{server\_error.non\_field\_errors[0]}</Alert> : ''}

      {/\* {server\_error.verify ? <Alert severity='error'>{server\_error.verify}</Alert> : ''} \*/}

    </Box>

  </>;

};

export default UserLogin;

* Now make changes in app.js so that if there is access token in the state then disable /login and /verify routes
* import { BrowserRouter, Navigate, Route, Routes } from "react-router-dom";
* // import LoginReg from "./pages/auth/LoginReg";
* import Registration from "./pages/auth/Registration";
* import UserLogin from "./pages/auth/UserLogin";
* import UserVerify from "./pages/auth/UserVerify";
* import ResetPassword from "./pages/auth/ResetPassword";
* import SendPasswordResetEmail from "./pages/auth/SendPasswordResetEmail";
* import Contact from "./pages/Contact";
* import Dashboard from "./pages/Dashboard";
* import Home from "./pages/Home";
* import Layout from "./pages/Layout";
* import { useSelector } from "react-redux"; // add this
* function App() {
* // to get access token from auth state
* const {access\_token} = useSelector(state => state.auth)
* return (
* <>
* <BrowserRouter>
* <Routes>
* <Route path="/" element={<Layout />}>
* <Route index element={<Home />} />
* <Route path="contact" element={<Contact />} />
* {/\* change \*/}
* <Route path="login" element={!access\_token ? <UserLogin /> : <Navigate to="/dashboard" />} />
* <Route path="register" element={<Registration />} />
* {/\* change \*/}
* <Route path="verify" element={!access\_token ? <UserVerify /> : <Navigate to="/dashboard" />} />
* <Route path="sendpasswordresetemail" element={<SendPasswordResetEmail />} />
* <Route path="reset" element={<ResetPassword />} />
* {/\* change \*/}
* <Route path="/dashboard" element={access\_token ? <Dashboard /> : <Navigate to="/login" />} />
* </Route>
* <Route path="\*" element={<h1>Error 404 Page not found !!</h1>} />
* </Routes>
* </BrowserRouter>
* </>
* );
* }
* export default App;
* Now we have to configure that when user click on logout then set access\_token to null in auth state. Then make changes in Dashboard where Logout button is present

import { Button, CssBaseline, Grid, Typography } from '@mui/material';

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

import ChangePassword from './auth/ChangePassword';

import { removeToken } from '../services/cookies';

import { useDispatch } from 'react-redux'; // add this

import { unSetUserToken } from '../features/authSlice';

const Dashboard = () => {

  const navigate = useNavigate()

  const dispatch = useDispatch() // add this

  const handleLogout = () => {

    // console.log("Logout Clicked");

    dispatch(unSetUserToken({ access\_token : null})) // add this

    removeToken()

    navigate('/login')

  }

  return <>

    <CssBaseline />

    <Grid container>

      <Grid item sm={4} sx={{ backgroundColor: 'gray', p: 5, color: 'white' }}>

        <h1>Dashboard</h1>

        <Typography variant='h5'>Email: sonam@gmail.com</Typography>

        <Typography variant='h6'>Name: Sonam</Typography>

        <Button variant='contained' color='warning' size='large' onClick={handleLogout} sx={{ mt: 8 }}>Logout</Button>

      </Grid>

      <Grid item sm={8}>

        <ChangePassword />

      </Grid>

    </Grid>

  </>;

};

export default Dashboard;