NETFLIX FULL STACK

Build Comprises of:

* Backend
* Front end
* Admin Panel

1. About Backend:

Develop backend using nodejs, express and mongodb for database alongwith firebase to store files.

Dependencies:

yarn add express mongoose nodemon dotenv jsonwebtoken morgan crypto-js

1. About Frontend:

Frontend is build using reactjs and scss along with material ui designing tool

Repo Link:

Dependencies:

Yarn add react-router-dom

Yarn add axios

1. About Admin Panel:

It is also build using reactjs

Repo Link:

App Link

SIMPLE JWT AUTHENTICATION:

Always copy paste these files without any changes

SIGNUP AND LOGIN API:

const express = require('express');

const router = express.Router();

const CryptoJS = require('crypto-js')

const jwt = require('jsonwebtoken')

const User = require('../schema/User');

//Register

router.post('/register', (req, res, next)=>{

    const newUser = new User({

        username: req.body.username,

        email: req.body.email,

        password: CryptoJS.AES.encrypt(req.body.password, process.env.SECRET\_KEY).toString(),

    });

    newUser

    .save()

    .then(result=>{

        res.status(201).json(result)

    })

    .catch(error => {

        res.status(500).json(error)

    })

})

//login

router.post('/login', (req, res)=>{

    User.findOne({email: req.body.email})

        .exec()

        .then(user => {

            if (!user){

                return res.status(401).json({

                    message: 'Auth Failed'

                })

            }

            //Decrypting password for comparision

            const bytes = CryptoJS.AES.decrypt(user.password, process.env.SECRET\_KEY);

            const originalPassword = bytes.toString(CryptoJS.enc.Utf8);

            if(originalPassword !== req.body.password){

                return res.status(401).json({

                    message: 'Please Enter Correct Password'

                })

            }else{

                const accessToken = jwt.sign(

                    {id: user.\_id, isAdmin: user.isAdmin},

                    process.env.SECRET\_KEY, {expiresIn: '1h'}

                    )

                //destructure to avoid getting password and \_\_v in response

                const {password, \_\_v, ...info} = user.\_doc;

                res.status(200).json({...info, accessToken})

            }

        })

        .catch(error=>{

            res.status(500).json(user)

        })

})

module.exports = router

checkAuth.js

const jwt = require('jsonwebtoken')

module.exports = (req, res, next) => {

    try {

        const token = req.headers.authorization.split(" ")[1];

        const decoded = jwt.verify(token, process.env.SECRET\_KEY, (err,userData)=>{

            if (err) {

                res.status(403).json("Token is not valid")

            };

            req.user = userData

             next(); //if successful go to router

    });

    }catch (error){

        return res.status(401).json({

            message: 'Auth failed'

        })

    }

}

Now pass this checkAuth in your routes like;

First make sure to pass in headers these things in postman

Key: authorization

Value : Bearer <Token>

//Update

router.put('/:id', checkAuth,(req, res, next)=>{

    //First check whether user updating his info or admin updating some user info or someone else trying to update info of any other user which we dont want

    //this req.user is from checkAuth file, it is for verification whether person same or not

    if (req.user.id === req.params.id || req.user.admin){

        //First checking if user want to update password

        if(req.body.password){

            req.body.password =CryptoJS.AES.encrypt(req.body.password, process.env.SECRET\_KEY).toString();

        }

        //update whatever info user wants to update

        User.findByIdAndUpdate({\_id:req.params.id}, req.body, {new:true})

            .exec()

            .then(result => {

                //destructuring result object by separating password and \_\_v and passing only info

                const {password, \_\_v, ...info} = result.\_doc;

                res.status(200).json(info)

            })

            .catch(err => {

                res.status(500).json({

                    error: err

                })

            })

    }else{

        res.status(403).json('You can update only your account')

    }

})

Aggregate and query and params in api:

List Schema:

const mongoose = require('mongoose');

const listSchema = new mongoose.Schema(

    {

    title: { type: String, required: true, unique: true },

    type: { type: String},

    genre: { type: String },

    content: { type: Array }

    },

    {

        timestamps: true

    }

);

module.exports = mongoose.model('List', listSchema);

Now querying data based on type and genre using aggregate and query params

const express = require('express');

const router = express.Router();

const CryptoJS = require('crypto-js')

const checkAuth = require('../auth/checkAuth');

const List = require('../schema/List');

//Create

router.post('/', checkAuth,(req, res, next)=>{

    //Only admin can create movie

    if (req.user.isAdmin){

        const newList = new List(req.body)

        newList

            .save()

            .then(result=>{

                res.status(201).json(result)

            })

            .catch(err => {

                res.status(500).json({

                    error: err

                })

            })

    }else{

        res.status(403).json('You are not allowed')

    }

})

//Delete

//Create

router.delete('/:id', checkAuth,(req, res, next)=>{

    //Only admin can create movie

    if (req.user.isAdmin){

        List.findByIdAndDelete({\_id:req.params.id})

            .exec()

            .then(() => {

                res.status(200).json({

                    message: "List has been deleted"

                })

            })

            .catch(err => {

                res.status(500).json({

                    error: err

                })

            })

    }else{

        res.status(403).json('You are not allowed')

    }

})

//get list from home page

router.get('/', checkAuth, (req, res, next)=>{

    const typeQuery = req.query.type;

    const genreQuery = req.query.genre;

    let list = [];

    //if type series or movies

    if (typeQuery){

        //if genre like action comedy in type series or movies

        if(genreQuery){

            List.aggregate([

                { $sample: { size: 10 } },

                { $match:{ type: typeQuery, genre: genreQuery } }

            ])

            .exec()

            .then(result => {

                res.status(200).json(result)

            })

            .catch(err => {

                res.status(500).json({

                    error: err

                })

            })

        }else{

            //if no genre specified, then 10 random provided type list either of series or movies

            List.aggregate([

                { $sample: { size: 10 } },

                { $match:{ type: typeQuery} }

            ])

            .exec()

            .then(result => {

                res.status(200).json(result)

            })

            .catch(err => {

                res.status(500).json({

                    error: err

                })

            })

        }

    }else{

        //homepage

        List.aggregate([

            { $sample: { size:10 } }

        ])

        .exec()

        .then(result => {

            res.status(200).json(result)

        })

        .catch(err => {

            res.status(500).json({

                error: err

            })

        })

    }

})

module.exports = router;

Redirect component in react-router-dom, routing or switch case:

import './App.scss';

import Home from './pages/home/Home';

import Login from './pages/login/Login';

import Register from './pages/register/Register';

import Watch from './pages/watch/Watch';

import { BrowserRouter as Router, Switch, Route, Redirect} from 'react-router-dom'

import { useState } from 'react';

function App() {

  const [user, setUser] = useState(true)

  return (

    <Router>

      <Switch>

        <Route exact path="/">

          {user ? <Home/> : <Redirect to="/register"/>}

        </Route>

        <Route path="/register">

          {!user ? <Register/> : <Redirect to="/"/>}

        </Route>

        <Route path="/login">

          {!user ? <Login/> : <Redirect to="/"/>}

        </Route>

        {user && (

          <>

          <Route path="/movies">

            <Home type="movie"/>

          </Route>

          <Route path="/series">

            <Home type="series"/>

          </Route>

          <Route path="/watch">

            <Watch/>

          </Route>

        </>

        )}

      </Switch>

    </Router>

  );

}

export default App;

How to Fetch API in Frontend:

Create constant using useState

Const [get, setGet[ = useState([]) or useState({})

1.

axios.get(‘/’).then(response => {

console.log(response.data)

setGet(response.data)

).catch(err => {

console.log(err)

})

Now using get.anykey you have passed database

2. If header authentication (jugaar):

axios.get(‘/’, {headers : {authorization: Bearer <tokenfrompostman}}).then(response => {

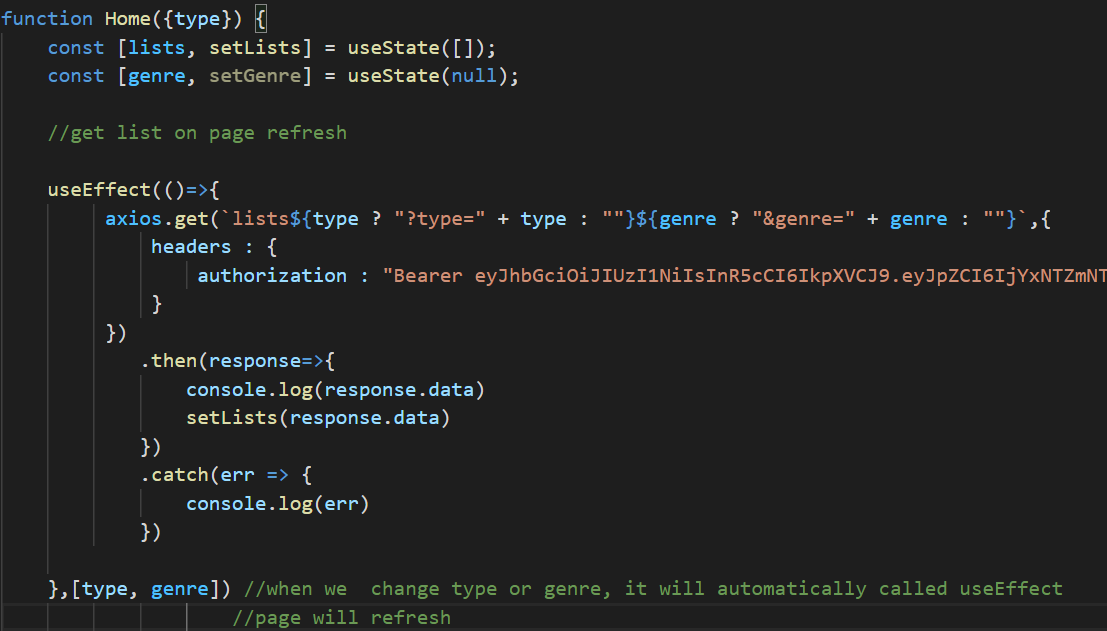
console.log(response.data)

).catch(err => {

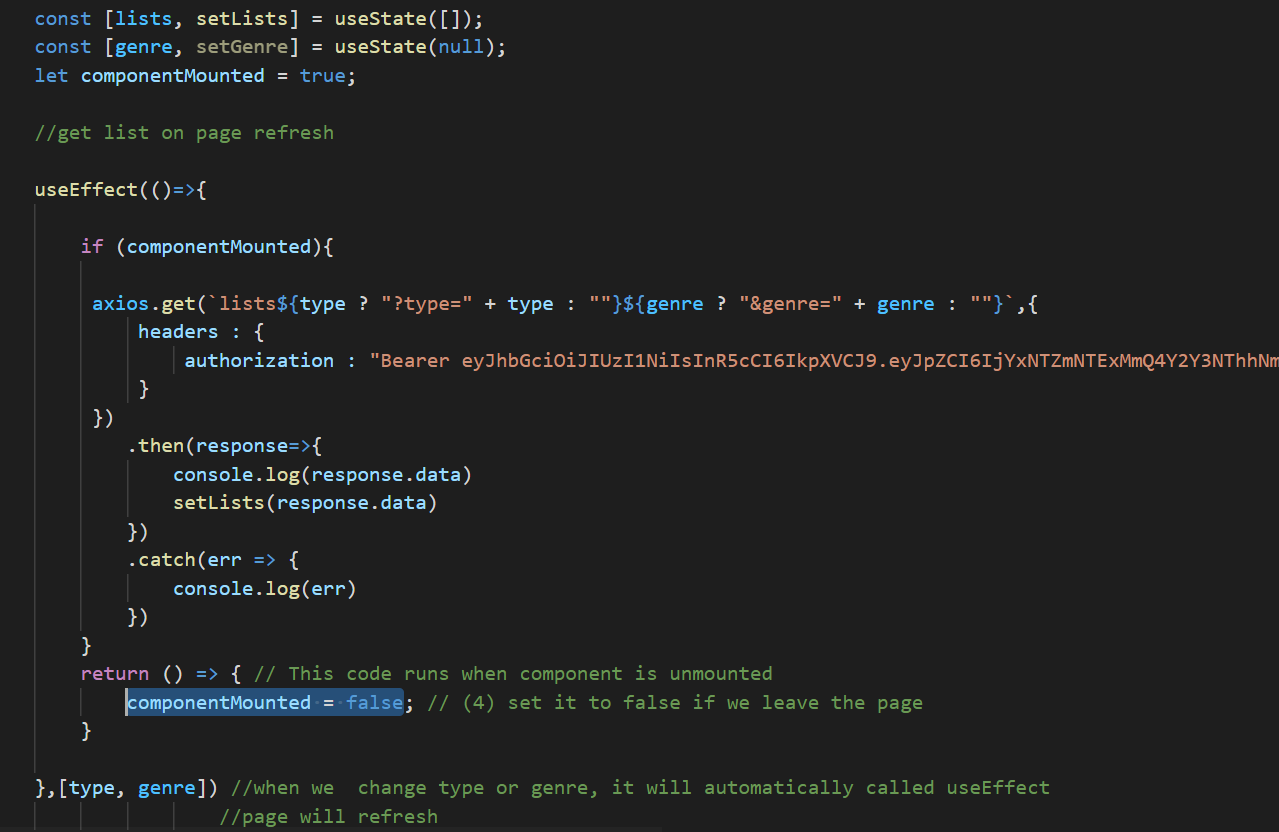
console.log(err)

})

3 With query params:



If issue of component mount then use below code



Link component with pathname and passing constant with it and use in another file

ListItem.jsx

import React, { useEffect } from 'react'

import './ListItem.scss'

import { Link } from 'react-router-dom'

import {

    PlayArrow,

    Add,

    ThumbUpAltOutlined,

    ThumbDownOutlined,

  } from "@material-ui/icons";

import { useState } from 'react';

import axios from 'axios';

function ListItem({index, item}) {

    const [isHovered, setIsHovered] = useState(false);

    const [movie, setMovie] = useState({})

    useEffect(()=>{

        axios.get('movies/find/' + item, {

            headers : {

                authorization : "Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6IjYxNTZmNTExMmQ4Y2Y3NThhNmM0ZGJiNiIsImlzQWRtaW4iOnRydWUsImlhdCI6MTYzMzE0ODc4NiwiZXhwIjoxNjMzMTUyMzg2fQ.D\_hHmTE50Y57aibX3FvVJn9T95pviDa0a3ikloQWSlo"

            }

        })

            .then(response => {

                setMovie(response.data)

            })

            .catch(err => {

                console.log(err)

            })

    },[item]) //Whenever item changes, useEffect will get fire

    return (

        <Link to={{pathname: "/watch", movie: movie}}>

            <div className="list\_\_item"

                style={{left: isHovered && index \* 225 - 50 + index \* 2.5}}

                onMouseEnter={()=>setIsHovered(true)}

                onMouseLeave={()=>setIsHovered(false)}

            >

                <img src={movie.img} alt="" />

                {isHovered && (

                <>

                <video src={movie.trailer} autoPlay={true} loop/>

                <div className="item\_\_info">

                    <div className="icons">

                        <PlayArrow className="icon" />

                        <Add className="icon" />

                        <ThumbUpAltOutlined className="icon" />

                        <ThumbDownOutlined className="icon" />

                    </div>

                    <div className="itemInfo\_\_top">

                        <span>{movie.duration}</span>

                        <span className="limit">{movie.limit}</span>

                        <span>{movie.year}</span>

                    </div>

                    <div className="desc">

                        {movie.desc}

                    </div>

                    <div className="genre">{movie.genre}</div>

                </div>

                </>

                )}

            </div>

        </Link>

    )

}

export default ListItem

Using movie constant in watch.jsx:

import { ArrowBackOutlined } from '@material-ui/icons'

import React from 'react'

import { useLocation } from 'react-router'

import './Watch.scss'

function Watch() {

    const location = useLocation()

    const movie = location.movie;

    console.log(location)

    return (

        <div className="watch">

            <div className="back">

                <ArrowBackOutlined />

                Home

            </div>

            <video

                className="video"

                autoPlay

                progress

                controls

                src={movie.video}

             />

        </div>

    )

}

export default Watch

User login with redux mongodb:

############################### START ########################

authSlice.js

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

export const authSlice = createSlice({

  name: 'user',

  initialState:{

    user: JSON.parse(localStorage.getItem("user")) || null,

    isFetching: false,

    error: false

  },

  reducers: {

    loginStart: (state) => {

      state.user = null;

      state.isFetching = true;

      state.error = false;

    },

    loginSuccess: (state, action) => {

      state.user = action.payload;

      state.isFetching = false;

      state.error = false;

    },

    loginFailure: (state) => {

      state.user = null;

      state.isFetching = false;

      state.error = true;

    },

    logout: (state) => {

      state.user = null;

      state.isFetching = false;

      state.error = false;

    },

  },

});

export const { loginStart, loginSuccess, loginFailure, logout} = authSlice.actions;

export default authSlice.reducer;

Store.js

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

import authReducer from './auth/authSlice';

export const store = configureStore({

  reducer: {

    user: authReducer,

  },

});

Now apiCall.js

import axios from "axios";

import { loginFailure, loginStart, loginSuccess } from './authSlice';

const Login = async (user, dispatch) => {

  dispatch(loginStart());

  try {

    const res = await axios.post("auth/login", user);

    dispatch(loginSuccess(res.data));

  } catch (err) {

    dispatch(loginFailure());

  }

};

export default Login

Now Login page.js where above login function will be used

import React, { useState } from 'react'

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

import Login from '../../redux/auth/apiCall'

import './LoginPage.css'

function LoginPage() {

    const [email, setEmail] = useState('');

    const [password, setPassword] = useState('');

    const dispatch = useDispatch()

    const isFetching = useSelector(state=>state.user.isFetching);

    console.log(isFetching)

    const handleSubmit = (e)=>{

        e.preventDefault();

        Login({email, password}, dispatch)

    }

    return (

        <div className="login">

            <form className="login\_\_form">

                <input type="text" placeholder="email" className="login\_\_input" value={email} onChange={e=>setEmail(e.target.value)}/>

                <input type="password" placeholder="password" className="login\_\_input" value={password} onChange={e=>setPassword(e.target.value)}/>

                <button className="login\_\_button" onClick={handleSubmit} disabled={isFetching}>Login</button>

            </form>

        </div>

    )

}

export default LoginPage

import React, { useState, useEffect, useMemo } from 'react'

import Sidebar from './components/Sidebar/Sidebar';

import TopBar from  './components/TopBar/TopBar'

import './App.css'

import Home from './pages/Home/Home';

import {BrowserRouter as Router, Switch, Route, Link} from 'react-router-dom'

import UserList from './pages/UserList/UserList';

import UserPage from './pages/UserPage/UserPage';

import NewUser from './pages/NewUser/NewUser';

import ProductList from './pages/ProductList/ProductList';

import Product from './pages/Product/Product';

import NewProduct from './pages/NewProduct/NewProduct';

import axios from 'axios';

import LoginPage from './pages/login/LoginPage';

import { useSelector } from 'react-redux';

function App() {

  const user = useSelector(state=>state.user.user)

  useEffect(()=>{

    localStorage.setItem("user", JSON.stringify(user))//user is not name of store, it the name of key inside state

  }, [user])

  return (

    <Router>

      <Switch>

          <Route path="/login">

              <LoginPage/>

          </Route>

        <TopBar/>

        <div className="container">

          <Sidebar/>

            <Route exact path="/">

              <Home/>

            </Route>

            <Route path="/users">

              <UserList/>

            </Route>

            <Route path="/user/:userId">

              <UserPage/>

            </Route>

            <Route path="/newUser">

              <NewUser/>

            </Route>

            <Route path="/movies">

              <ProductList/>

            </Route>

            <Route path="/product/:productId">

              <Product/>

            </Route>

            <Route path="/newproduct">

              <NewProduct/>

            </Route>

        </div>

        </Switch>

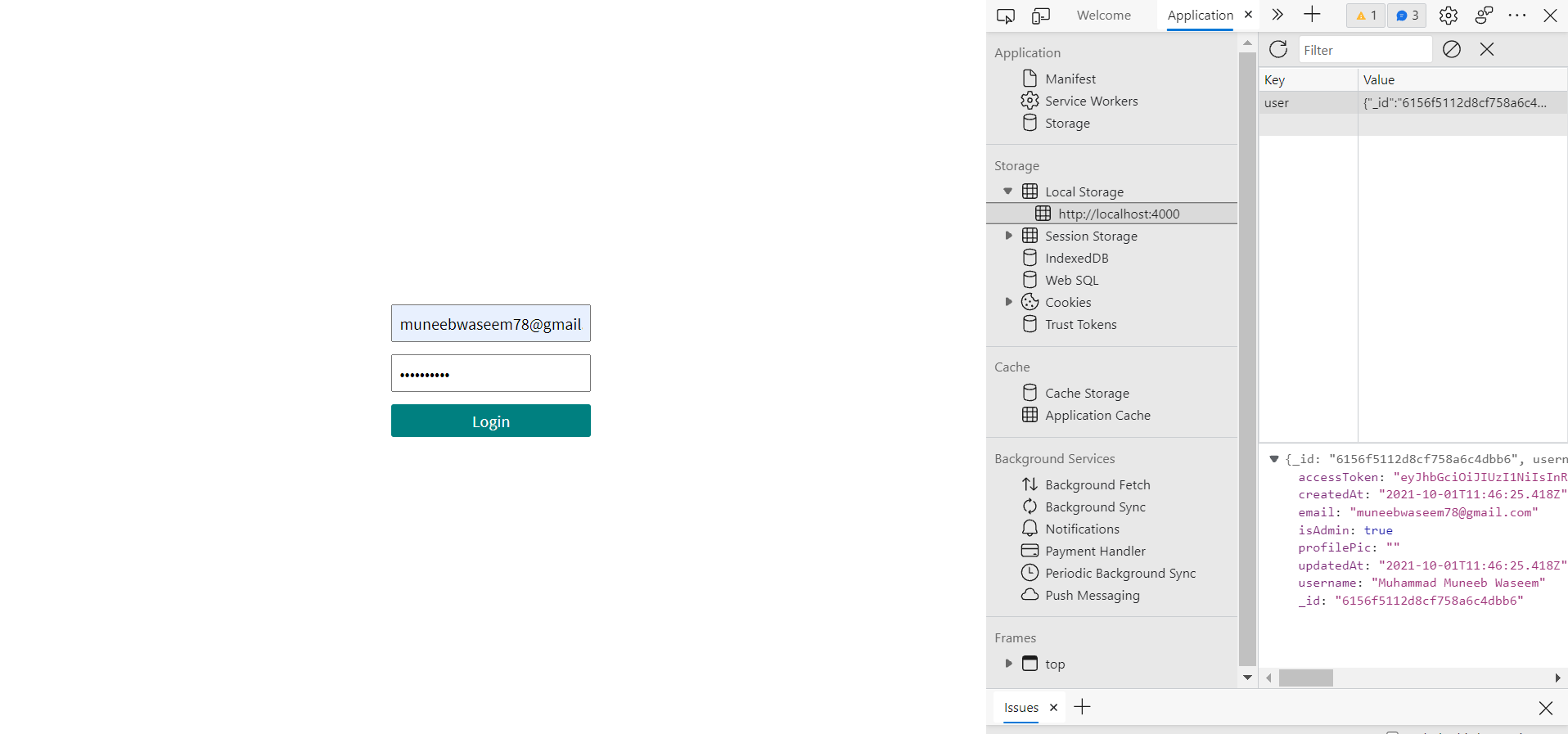
      </Router>

  );

}

export default App;

In application under localstorage, we can find that user not logged out on page refresh



############################ END ###########################################