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

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;