**1. Add authentication features to the project created in Assignment-9. Add API end points and React pages for Registration and Login. Use a separate collection called *Admin* in MongoDB with attributes like Name, email, mobile and Password, which will be used for authentication details.**

**Ans-**

**Admin.model.js**

const mongoose = require("mongoose");

const userSchema = new mongoose.Schema(

{

name: {

type: String,

required: true,

},

mobile: {

type: Number,

unique: true,

required: true,

},

email: {

type: String,

required: true,

unique: true,

},

password: {

type: String,

required: true,

},

},

{

timestamps: true,

}

);

const userModel = new mongoose.model("Admin", userSchema);

module.exports = userModel;

**user.controller.js**

const adminModel = require("../models/admin.model");

const bcrypt = require("bcrypt");

const jwt = require("jsonwebtoken");

const signup = async (req, res) => {

try {

const newUser = req.body;

const { email, password } = newUser;

const user = await adminModel.findOne({ email });

if (user) {

res.status(404).json({ message: "Email Already exists!!" });

return;

} else {

const salt = bcrypt.genSaltSync(10);

const hashedPassword = bcrypt.hashSync(password, salt);

newUser.password = hashedPassword;

await adminModel.create(newUser);

res

.status(201)

.json({ message: "User Successfully Registered!", user: newUser });

}

} catch (error) {

console.log(error);

res.status(500).json({ message: error });

}

};

const signin = async (req, res) => {

try {

const user = req.body;

const { email, password } = user;

const dbUser = await adminModel.findOne({ email });

if (!dbUser) {

return res.status(401).json({ message: "Invalid Email" });

}

const isValidPass = bcrypt.compareSync(password, dbUser.password);

if (!isValidPass) {

return res.status(401).json({ auth: false, message: "Invalid Password" });

}

const token = jwt.sign(

{

user: {

id: dbUser.\_id,

name: dbUser.name,

mobile: dbUser.mobile,

email: dbUser.email,

},

},

process.env.JWT\_SECRET,

{

expiresIn: "6h",

}

);

res.status(200).send({ message: "Successfully Logged In!", token });

} catch (error) {

console.log(error);

res.status(500).json({ message: error.message });

}

};

module.exports = {

signup,

signin,

};

**Register.jsx**

/\* eslint-disable react/no-unknown-property \*/

import { useState } from "react";

import { TextField, Button, Snackbar, Alert } from "@mui/material";

import { AccountCircle, Phone, Email, LockOutlined } from "@mui/icons-material";

import userService from "../service/UserService";

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

const Register = () => {

const navigate = useNavigate();

const [name, setName] = useState("");

const [mobile, setMobile] = useState("");

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

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

const [open, setOpen] = useState(false);

const [success, setSuccess] = useState(false);

const handleClose = (event, reason) => {

if (reason === "clickaway") {

return;

}

setOpen(false);

};

const handleSubmit = async (event) => {

event.preventDefault();

const user = {

name,

mobile,

email,

password,

};

let res = await userService.registerUser(user);

if (res.status) {

setName("");

setMobile("");

setEmail("");

setPassword("");

setOpen(true);

setSuccess(true);

setTimeout(() => {

navigate("/login");

}, 3000);

} else {

setOpen(true);

setSuccess(false);

}

};

return (

<>

<Snackbar

open={open}

autoHideDuration={6000}

onClose={handleClose}

anchorOrigin={{ vertical: "top", horizontal: "right" }}

>

<Alert

onClose={handleClose}

severity={success ? "success" : "error"}

variant="filled"

sx={{ width: "100%" }}

>

{success

? "Successfully Signed In! Log In to continue!"

: "Unable to Sign In! Please Try Again!!"}

</Alert>

</Snackbar>

<h1 align="center">Register</h1>

<div

style={{

display: "flex",

justifyContent: "center",

alignItems: "center",

height: "50vh",

}}

>

<form onSubmit={handleSubmit}>

<TextField

label="Name"

value={name}

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

required

margin="normal"

fullWidth

InputProps={{

startAdornment: <AccountCircle />,

}}

/>

<br />

<TextField

label="Mobile Number"

type="tel"

value={mobile}

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

required

margin="normal"

fullWidth

InputProps={{

startAdornment: <Phone />,

}}

/>

<TextField

label="Email"

type="email"

value={email}

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

required

margin="normal"

fullWidth

InputProps={{

startAdornment: <Email />,

}}

/>

<TextField

label="Password"

type="password"

value={password}

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

required

margin="normal"

fullWidth

InputProps={{

startAdornment: <LockOutlined />,

}}

/>

<Button type="submit" variant="contained" color="primary" fullWidth>

Register

</Button>

</form>

</div>

</>

);

};

export default Register;

**Login.jsx**

/\* eslint-disable react/no-unknown-property \*/

import { useState } from "react";

import { TextField, Button, Snackbar, Alert } from "@mui/material";

import { AccountCircle, LockOutlined } from "@mui/icons-material";

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

import userService from "../service/UserService";

import { useAuth } from "../contexts/AuthContext";

const Login = () => {

const authContext = useAuth();

const { login } = authContext;

const navigate = useNavigate();

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

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

const [open, setOpen] = useState(false);

const [success, setSuccess] = useState(false);

const handleClose = (event, reason) => {

if (reason === "clickaway") {

return;

}

setOpen(false);

};

const handleSubmit = async (event) => {

event.preventDefault();

const user = {

email,

password,

};

let res = await userService.loginUser(user);

if (res.status) {

setEmail("");

setPassword("");

login(res.data.token);

setOpen(true);

setSuccess(true);

setTimeout(() => {

navigate("/");

}, 3000);

} else {

setOpen(true);

setSuccess(false);

}

};

return (

<>

<Snackbar

open={open}

autoHideDuration={6000}

onClose={handleClose}

anchorOrigin={{ vertical: "top", horizontal: "right" }}

>

<Alert

onClose={handleClose}

severity={success ? "success" : "error"}

variant="filled"

sx={{ width: "100%" }}

>

{success

? "Successfully Logged In!!"

: "Unable to Log In! Please Try Again!!"}

</Alert>

</Snackbar>

<h1 align="center">Login</h1>

<div

style={{

display: "flex",

justifyContent: "center",

alignItems: "center",

height: "50vh",

}}

>

<form onSubmit={handleSubmit}>

<TextField

label="Email"

type="email"

value={email}

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

required

margin="normal"

fullWidth

InputProps={{

startAdornment: <AccountCircle />,

}}

/>

<TextField

label="Password"

type="password"

value={password}

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

required

margin="normal"

fullWidth

InputProps={{

startAdornment: <LockOutlined />,

}}

/>

<Button type="submit" variant="contained" color="primary" fullWidth>

Login

</Button>

</form>

</div>

</>

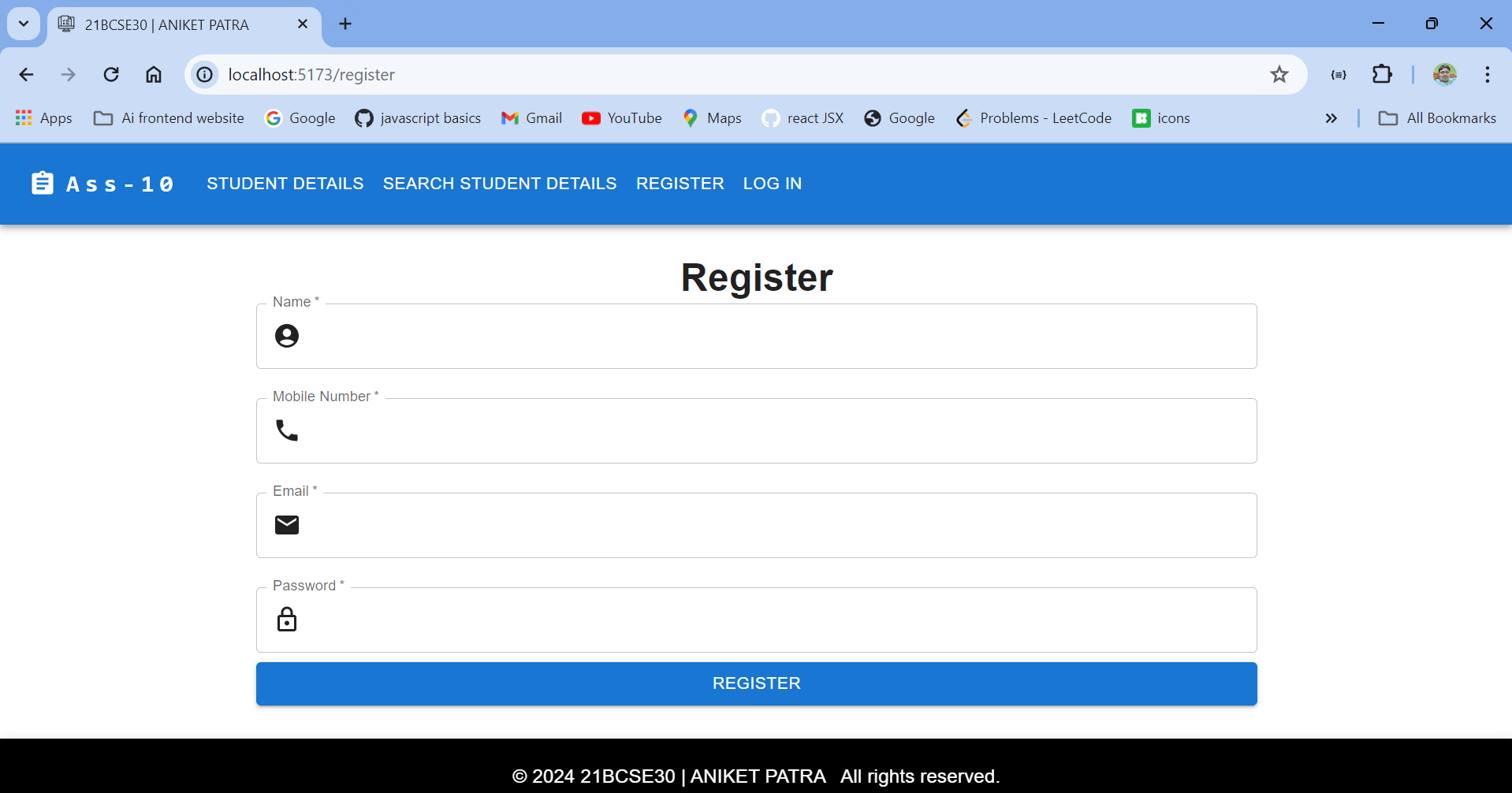
);

};

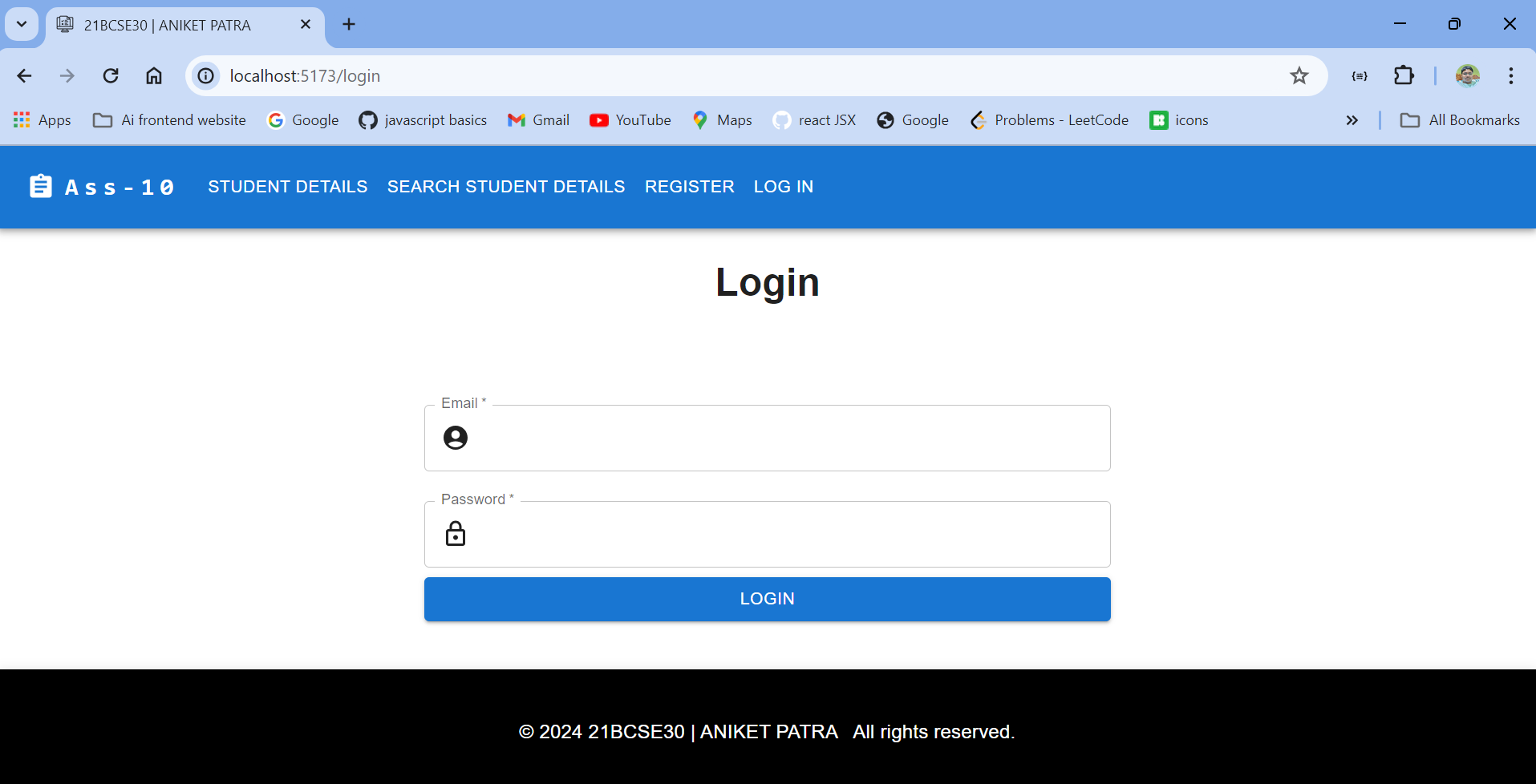
export default Login;

**Output**

**Register.jsx**



**Login.jsx**



**2. Use JWT token based authentication. If the user is logged in successfully, return a JWT token from the backend. Store the JWT token in local storage of browser for further uses.**

**Ans-**

**Auth.middleware.js**

const jwt = require("jsonwebtoken");

const authenticateUser = (req, res, next) => {

const token = req.header("Authorization");

if (!token) {

return res.status(401).json({ message: "Unauthorized User" });

}

let jwttoken = token.split(" ")[1];

try {

let decodedData = jwt.verify(jwttoken, process.env.JWT\_SECRET);

req.user = decodedData.user;

next();

} catch (error) {

res.status(400).json({ message: "Invalid Token" });

}

};

module.exports = authenticateUser;

**3. Design a custom middleware in the backend, which will help us to design the protected routes for backend. Protect all the routes (accessible after login only) of assignment-9.**

**Ans-**

**Studentinfos.routes.js**

const express = require("express");

const {

getAllStudentDetails,

addAllStudentDetails,

getStudentDetailsByRoll,

deleteStudentDetailsByRoll,

updateStudentDetailsByRoll,

} = require("../controllers/allstudentinfo.controller");

const authenticateUser = require("../middlewares/auth.middleware");

const AllStudentInfoRouter = express.Router();

AllStudentInfoRouter.get("/", getAllStudentDetails);

AllStudentInfoRouter.post("/",authenticateUser, addAllStudentDetails);

AllStudentInfoRouter.get("/:id", getStudentDetailsByRoll);

AllStudentInfoRouter.delete("/:id",authenticateUser, deleteStudentDetailsByRoll);

AllStudentInfoRouter.put("/:id",authenticateUser, updateStudentDetailsByRoll);

module.exports = {

AllStudentInfoRouter,

};

**4. Use context API in the React application to protect the frontend routes and maintain logged in user’s information at a central place.**

**Ans-**

**AuthContext.js**

/\* eslint-disable react-refresh/only-export-components \*/

/\* eslint-disable react/prop-types \*/

import { useEffect } from "react";

import { createContext, useContext, useState } from "react";

import { decodeToken } from 'react-jwt'

export const AuthContext = createContext(null)

export const AuthProvider = (props) => {

const [ token, setToken ] = useState(null)

const [ isLoggedIn, setIsLoggedIn ] = useState(false)

const [ user, setUser ] = useState(null)

useEffect(()=>{

const jwttoken = localStorage.getItem("token")

if(jwttoken){

setToken(jwttoken)

setIsLoggedIn(true)

const decodedData = decodeToken(jwttoken)

setUser(decodedData.user)

}

}, [token])

const login = (token) => {

localStorage.setItem("token", token)

setToken(token)

setIsLoggedIn(true)

}

const logout = () =>{

localStorage.removeItem("token")

setToken(null)

setIsLoggedIn(false)

}

return(

<AuthContext.Provider value={{token, isLoggedIn, login, logout, user}}>

{props.children}

</AuthContext.Provider>

)

}

export const useAuth = () =>{

const authContext = useContext(AuthContext)

return authContext

}

**Navbar.jsx**

import \* as React from "react";

import AppBar from "@mui/material/AppBar";

import Box from "@mui/material/Box";

import Toolbar from "@mui/material/Toolbar";

import IconButton from "@mui/material/IconButton";

import Typography from "@mui/material/Typography";

import Menu from "@mui/material/Menu";

import MenuIcon from "@mui/icons-material/Menu";

import Container from "@mui/material/Container";

import Button from "@mui/material/Button";

import MenuItem from "@mui/material/MenuItem";

import AssignmentIcon from "@mui/icons-material/Assignment";

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

import { useAuth } from "../contexts/AuthContext";

const pages = [

"Student Details",

"Search Student Details",

"Add Student Details",

"Update Student Details",

"Delete Student Details",

"Register",

"Log In",

"Log Out",

];

function ResponsiveAppBar() {

const authContext = useAuth();

const { isLoggedIn, logout } = authContext;

const logoutUser = () => {

logout();

};

const [anchorElNav, setAnchorElNav] = React.useState(null);

const navigate = useNavigate();

const handleOpenNavMenu = (event) => {

setAnchorElNav(event.currentTarget);

};

const handleNavigate = (event) => {

setAnchorElNav(null);

const pageToNavigate = event.target.innerText.toLowerCase();

navigate(`/${pageToNavigate.split(" ").join("")}`);

};

return (

<AppBar position="static">

<Container maxWidth="xl">

<Toolbar disableGutters>

<AssignmentIcon sx={{ display: { xs: "none", md: "flex" }, mr: 1 }} />

<Typography

variant="h6"

noWrap

component="a"

href="/"

sx={{

mr: 2,

display: { xs: "none", md: "flex" },

fontFamily: "monospace",

fontWeight: 700,

letterSpacing: ".3rem",

color: "inherit",

textDecoration: "none",

}}

>

Ass-10

</Typography>

<Box sx={{ flexGrow: 1, display: { xs: "flex", md: "none" } }}>

<IconButton

size="large"

aria-label="account of current user"

aria-controls="menu-appbar"

aria-haspopup="true"

onClick={handleOpenNavMenu}

color="inherit"

>

<MenuIcon />

</IconButton>

<Menu

id="menu-appbar"

anchorEl={anchorElNav}

anchorOrigin={{

vertical: "bottom",

horizontal: "left",

}}

keepMounted

transformOrigin={{

vertical: "top",

horizontal: "left",

}}

open={Boolean(anchorElNav)}

onClose={handleNavigate}

sx={{

display: { xs: "block", md: "none" },

}}

>

{pages.map((page, index) => {

if (index > 1) return;

return (

<MenuItem key={index} onClick={handleNavigate}>

<Typography textAlign="center">{page}</Typography>

</MenuItem>

);

})}

{isLoggedIn ? (

<div>

<MenuItem onClick={handleNavigate}>

<Typography textAlign="center">{pages[2]}</Typography>

</MenuItem>

<MenuItem onClick={handleNavigate}>

<Typography textAlign="center">{pages[3]}</Typography>

</MenuItem>

<MenuItem onClick={handleNavigate}>

<Typography textAlign="center">{pages[4]}</Typography>

</MenuItem>

<MenuItem onClick={logoutUser}>

<Typography textAlign="center">{pages[7]}</Typography>

</MenuItem>

</div>

) : (

<div>

<MenuItem onClick={handleNavigate}>

<Typography textAlign="center">{pages[5]}</Typography>

</MenuItem>

<MenuItem onClick={handleNavigate}>

<Typography textAlign="center">{pages[6]}</Typography>

</MenuItem>

</div>

)}

</Menu>

</Box>

<AssignmentIcon sx={{ display: { xs: "flex", md: "none" }, mr: 1 }} />

<Typography

variant="h5"

noWrap

component="a"

href="/"

sx={{

mr: 2,

display: { xs: "flex", md: "none" },

flexGrow: 1,

fontFamily: "monospace",

fontWeight: 700,

letterSpacing: ".3rem",

color: "inherit",

textDecoration: "none",

}}

>

Assignment-10

</Typography>

<Box sx={{ flexGrow: 1, display: { xs: "none", md: "flex" } }}>

{pages.map((page, index) => {

if (index > 1) return;

return (

<Button

key={page}

onClick={handleNavigate}

sx={{ my: 2, color: "white", display: "block" }}

>

{page}

</Button>

);

})}

{isLoggedIn ? (

<>

<Button

onClick={logoutUser}

sx={{ my: 2, color: "white", display: "block" }}

>

{pages[2]}

</Button>

<Button

onClick={logoutUser}

sx={{ my: 2, color: "white", display: "block" }}

>

{pages[3]}

</Button>

<Button

onClick={logoutUser}

sx={{ my: 2, color: "white", display: "block" }}

>

{pages[4]}

</Button>

<Button

onClick={logoutUser}

sx={{ my: 2, color: "white", display: "block" }}

>

{pages[7]}

</Button>

</>

) : (

<>

<Button

onClick={handleNavigate}

sx={{ my: 2, color: "white", display: "block" }}

>

{pages[5]}

</Button>

<Button

onClick={handleNavigate}

sx={{ my: 2, color: "white", display: "block" }}

>

{pages[6]}

</Button>

</>

)}

</Box>

{/\* <Box sx={{ flexGrow: 0 }}>

<Avatar alt="user" src="/avatars/user.png" />

</Box> \*/}

</Toolbar>

</Container>

</AppBar>

);

}

export default ResponsiveAppBar;