



























Login

Frontend:

import React from 'react'

import { useState } from 'react';

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

import Axios from 'axios';

import useAuth from '../Components/useAuth';

const Login = () => {

    const navigate = useNavigate();

    const { setAuth } = useAuth();

    const [username, setusername] = useState("");

    const [pass, setpass] = useState("");

    const [error, seterror] = useState(false);

    Axios.defaults.withCredentials = true;

    const submitLogin = async (e) => {

        e.preventDefault();

        if (username.length === 0 || pass.length === 0) {

          seterror(true);

        } else {

          try {

            const response = await Axios.post('http://localhost:3002/api/login', {

              username: username,

              pass: pass,

            });

            if (response.data.message) {

              if (response.data.message === 'Wrong username/password') {

                alert('Wrong password');

              } else {

                alert('No user exists');

              }

            } else {

              const user = response.data[0];

              user.role = [user.role];

              setAuth(user);

              window.alert('LogIn successfully');

              navigate('/profile');

            }

          } catch (error) {

            console.log('Error during login:', error);

            alert('Error during login. Please try again later.');

          }

        }

      };

  return (

    <>

      <div className='bg-gradient-to-r

                 from-[#012f66] to-[#0568a1]  py-3' >

        <img className="m-5 px-20" src='/Images/drdo\_logo.png' alt='logo' />

      </div>

      <div className='relative'>

          <img src='/Images/banner.jpg' alt='background' />

      </div>

      <div className="w-full max-w-xs mx-auto mt-10">

            <form  className=" bg-white shadow-2xl rounded px-8 pt-6 pb-8 mb-4">

                <div className="justify-between items-center mb-8 ">

                    <h1 className="text-center block text-3xl text-gray-800 font-medium">LOG IN</h1>

                </div>

                <div className="mb-4">

                    <label className="block text-gray-700 text-sm font-bold mb-2">

                        Username

                    </label>

                    <input onChange={(e) => {

                            setusername(e.target.value)

                        }}

                    placeholder='Username'

                     className="shadow appearance-none border rounded w-full py-2 px-3 text-gray-700 leading-tight focus:outline-none focus:shadow-outline" type="text" />

                {error &&username.length<=0?<label className='text-red-500 text-sm px-2 '>Username cannot be empty</label>:""}

                </div>

                <div className="mb-6">

                    <label className="block text-gray-700 text-sm font-bold mb-2" >

                        Password

                    </label>

                    <input onChange={(e) => {

                            setpass(e.target.value)

                        }}

                    placeholder='Password'

                    className="shadow appearance-none border rounded w-full py-2 px-3 text-gray-700 mb-3 leading-tight focus:outline-none focus:shadow-outline" type="password" />

                {error &&pass.length<=0?<label className='text-red-500 text-sm px-2 '>Password cannot be empty</label>:""}

                </div>

                <div className="flex items-center justify-between">

                    <button onClick={submitLogin}

                    className="bg-blue-500 py-2 px-4 border-2 rounded-md shadow-2xl text-white  transition ease-in-out delay-150  hover:scale-110 hover:bg-green-500 duration-300 ... focus:outline-none focus:shadow-outline" type="button">

                        Log In

                    </button>

                    <a className="inline-block align-baseline  text-sm text-blue-500 hover:text-blue-800" href='/Signup' >

                        Don't have an account?

                    </a>

                </div>

            </form>

        </div>

    </>

  )

}

export default Login

Backend:

app.post('/api/login', (req, res) => {

    const username = req.body.username;

    const pass = req.body.pass;

    db.query("SELECT \* FROM  client WHERE username= ?", username, (err, data) => {

        if (err) {

            console.log(err)

            return res.json("Error")

        }

        if (data.length > 0) {

            console.log(data)

            bcrypt.compare(pass, data[0].pass, (error, response) => {

                if (error) return res.json({ message: "Password compare error" });

                if (response) {

                    const username = data[0].username;

                    const token = jwt.sign({ username }, "jwt-secret-key", { expiresIn: "1d" });

                    res.cookie('token', token)

                    res.send(data)

                }

                else {

                    res.json({ message: "Wrong username/password" });

                }

            })

        }

        else {

            res.send({ message: "user doesnt exist" })

        }

    });

})

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

    const token = req.cookies.token;

    if (!token) {

        return res.json({ message: "not auth" })

    }

    else {

        jwt.verify(token, "jwt-secret-key", (err, decoded) => {

            if (err) {

                return res.json({ message: "not auth" })

            }

            else {

                req.username = decoded.username;

                next()

            }

        })

    }

}

SignUp

Frontend:

import React from 'react'

import Axios from 'axios'

import { useState } from 'react';

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

const Signup = () => {

    const navigate = useNavigate();

    const [username, setusername] = useState("");

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

    const [pass, setpass] = useState("");

    const [department, setdepartment] = useState("");

    const [confirmpass, setconfirmpass] = useState("");

    const [error, seterror] = useState(false);

    Axios.defaults.withCredentials = true;

    const submitSignup = async (e) => {

        e.preventDefault();

        if (

          name.length === 0 ||

          username.length === 0 ||

          pass.length === 0 ||

          department.length === 0 ||

          confirmpass.length === 0

        ) {

          seterror(true);

        } else if (confirmpass !== pass) {

          window.alert('Confirm password does not match');

        } else {

          try {

            const response = await Axios.post('http://localhost:3002/api/signup', {

              username: username,

              name: name,

              pass: pass,

              department: department,

            });

            if (response.data === 'Error') {

              window.alert('Username already exists');

            } else {

              navigate('/');

            }

          } catch (error) {

            console.log('Error during signup:', error);

            alert('Error during signup. Please try again later.');

          }

        }

      };

    return (

        <>

            <div className='bg-gradient-to-r

                 from-[#012f66] to-[#0568a1]  py-3' >

                <img className="m-5 px-20" src='/Images/drdo\_logo.png' alt='logo' />

            </div>

            <div className='relative'>

                <img src='/Images/banner.jpg' alt='background' />

            </div>

            <div className="w-full max-w-xs mx-auto mt-10">

                <form className=" bg-white shadow-2xl rounded px-8 pt-6 pb-8 mb-4">

                    <div className="justify-between items-center mb-8 ">

                        <h className="text-center block text-3xl text-gray-800 font-medium">SIGN UP</h>

                    </div>

                    <div className="mb-4">

                        <label className="block text-gray-700 text-sm font-bold mb-2">

                            Name

                        </label>

                        <input onChange={(e) => {

                            setname(e.target.value)

                        }}

                            placeholder='Name'

                            className="shadow appearance-none border rounded w-full py-2 px-3 text-gray-700 leading-tight focus:outline-none focus:shadow-outline" type="text" />

                    {error &&name.length<=0?<label className='text-red-500 text-sm px-2 '>Name cannot be empty</label>:""}

                    </div>

                    <div className="mb-4">

                        <label className="block text-gray-700 text-sm font-bold mb-2">

                            Department

                        </label>

                        <input onChange={(e) => {

                            setdepartment(e.target.value)

                        }}

                            placeholder='Department'

                            className="shadow appearance-none border rounded w-full py-2 px-3 text-gray-700 leading-tight focus:outline-none focus:shadow-outline" type="text" />

                    {error &&department.length<=0?<label className='text-red-500 text-sm px-2 '>Department cannot be empty</label>:""}

                    </div>

                    <div className="mb-4">

                        <label className="block text-gray-700 text-sm font-bold mb-2">

                            Username

                        </label>

                        <input onChange={(e) => {

                            setusername(e.target.value)

                        }}

                            placeholder='Username'

                            className="shadow appearance-none border rounded w-full py-2 px-3 text-gray-700 leading-tight focus:outline-none focus:shadow-outline" type="text" />

                    {error &&username.length<=0?<label className='text-red-500 text-sm px-2 '>Username cannot be empty</label>:""}

                    </div>

                    <div className="mb-4">

                        <label className="block text-gray-700 text-sm font-bold mb-2" >

                            Password

                        </label>

                        <input onChange={(e) => {

                            setpass(e.target.value)

                        }}

                            placeholder='Password'

                            className="shadow appearance-none border rounded w-full py-2 px-3 text-gray-700 mb-3 leading-tight focus:outline-none focus:shadow-outline" type="password" />

                    {error &&pass.length<=0?<label className='text-red-500 text-sm px-2 '>Password cannot be empty</label>:""}

                    </div>

                    <div className="mb-6">

                        <label className="block text-gray-700 text-sm font-bold mb-2">

                            Confirm Password

                        </label>

                        <input onChange={(e) => {

                            setconfirmpass(e.target.value)

                        }}

                            placeholder='Confirm Password'

                            className="shadow appearance-none border rounded w-full py-2 px-3 text-gray-700 leading-tight focus:outline-none focus:shadow-outline" type="text" />

                    {error &&confirmpass.length<=0?<label className='text-red-500 text-sm px-2 '>Confirm Password cannot be empty</label>:""}

                    </div>

                    <div className="mt-2 flex items-center justify-between">

                        <button onClick={submitSignup}

                            className="w-full bg-blue-500  border-2 rounded-md shadow-2xl text-white  transition ease-in-out delay-150  hover:scale-110 hover:bg-green-500 duration-300 ...  font-bold py-2 px-4  focus:outline-none focus:shadow-outline" type="button">

                            Signup

                        </button>

                    </div>

                </form>

            </div>

        </>

    )

}

export default Signup

Backend:

app.post('/api/signup', (req, res) => {

    const username = req.body.username;

    const name = req.body.name;

    const pass = req.body.pass;

    const department = req.body.department;

    bcrypt.hash(pass, saltRounds, (err, hash) => {

        if (err) {

            console.log(err)

        }

        db.query("INSERT INTO client (username,name, pass, department) VALUES (?,?,?,?)", [username, name, hash, department], (err, result, data) => {

            if (err) {

                console.log(err)

                return res.json("Error")

            }

            console.log(result)

            return res.json(data);

        });

    })

})

Profile

Frontend:

import React from 'react'

import Nav from '../Components/Nav'

import Axios from 'axios'

import { useEffect, useState } from 'react'

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

const Profile = () => {

    const Navigate = useNavigate();

    Axios.defaults.withCredentials = true;

    const [data, setData] = useState(null);

    const [noOfComplaints, setNoOfComplaints] = useState(null);

    const [loading, setLoading] = useState(true);

    useEffect(() => {

      const fetchData = async () => {

        try {

          const userData = await Axios.get('http://localhost:3002');

          setData(userData.data[0]);

          setLoading(false);

        } catch (error) {

          console.error('Error fetching user data:', error);

          setLoading(false);

        }

      };

      fetchData();

    }, []);

    useEffect(() => {

      const fetchComplaintsData = async () => {

        try {

          const complaintsData = await Axios.get('http://localhost:3002/complaint\_data');

          setNoOfComplaints(complaintsData.data.length);

        } catch (error) {

          console.error('Error fetching complaints data:', error);

        }

      };

      fetchComplaintsData();

    }, []);

    const handleLogout = async (e) => {

      e.preventDefault();

      try {

        await Axios.get('http://localhost:3002/logout');

        window.alert('Logged out');

        Navigate('/');

      } catch (error) {

        console.error('Error during logout:', error);

      }

    };

    return (

      <div className=''>

        <Nav />

        <div className='mt-10'>

          <div className='mx-auto max-w-md sm:max-w-lg'>

            {loading ? (

              <div className='text-center'>Loading...</div>

            ) : (

              <div className='bg-[#f5f5f5] shadow-2xl  rounded px-8 pt-6 pb-8 mb-4'>

                <div className='mb-2 '>

                  <img

                    className='w-20 border-4 rounded-full m-auto '

                    src='/Images/user-icon.jpg'

                    alt='user'

                  />

                </div>

                <div className='mb-10 text-center '>

                  <h1 className='block text-gray-700 font-bold text-2xl'>{`${data.name.toUpperCase()}'s profile`}</h1>

                </div>

                <div className=''>

                  <div>

                    {/\* Add more styling and formatting \*/}

                    <div className='flex justify-between mb-4'>

                      <span className='font-semibold'>Name:</span>

                      <span>{data.name.toUpperCase()}</span>

                    </div>

                    <div className='flex justify-between mb-4'>

                      <span className='font-semibold'>Username:</span>

                      <span>{data.username.toUpperCase()}</span>

                    </div>

                    <div className='flex justify-between mb-4'>

                      <span className='font-semibold'>Department:</span>

                      <span>{data.department.toUpperCase()}</span>

                    </div>

                    <div className='flex justify-between mb-4'>

                      <span className='font-semibold'>No of Complaints:</span>

                      <span>{noOfComplaints}</span>

                    </div>

                  </div>

                </div>

              </div>

            )}

          </div>

        </div>

        <div className='text-center'>

          <button

            className='bg-blue-500 py-2 px-4 border-2 rounded-md shadow-2xl text-white  transition ease-in-out delay-150  hover:scale-110 hover:bg-red-500 duration-300 ...'

            onClick={handleLogout}

          >

            Log Out

          </button>

        </div>

      </div>

    );

  };

  export default Profile;

Backend:

app.get('/', verifyUser, (req, res) => {

    const username = req.username

    db.query("SELECT \* FROM  client WHERE username= ?", username, (err, data) => {

        if (err) {

            console.log(err)

            return res.json("Error")

        }

        if (data) {

            return res.send(data)

        };

    })

})

app.get('/singlecomplaint\_data/:complaint\_id', (req, res) => {

    const complaint\_id = req.params.complaint\_id

    db.query("SELECT \* FROM  complaints WHERE complaint\_id= ?", [complaint\_id], (err, data) => {

        if (err) {

            console.log(err)

            return res.json("Error")

        }

        else {

            return res.send(data)

        };

    })

})

New Complaint

Frontend:

import React from 'react';

import Nav from '../Components/Nav';

import Axios from 'axios';

import { useState,useEffect } from 'react';

const Main = () => {

  const [complaint, setComplaint] = useState('');

  const [complaint\_type, setComplaintType] = useState('');

  const [error, setError] = useState(false);

  const [complaint\_id, setComplaintId] = useState(null);

  Axios.defaults.withCredentials = true;

  useEffect(() => {

    getRandomNumber();

  }, []);

  const getRandomNumber = () => {

    let result = '';

    const characters = 'ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789';

    const charactersLength = characters.length;

    for (let i = 0; i < 8; i++) {

      result += characters.charAt(Math.floor(Math.random() \* charactersLength));

    }

    setComplaintId(result);

  };

  const submitComplaint = async () => {

    if (complaint.length === 0 || complaint\_type.length === 0) {

      setError(true);

    } else {

      try {

        const res = await Axios.post('http://localhost:3002/complaint', {

          complaint: complaint,

          complaint\_type: complaint\_type,

          complaint\_id: complaint\_id,

        });

        if (res.data === 'Error') {

          window.alert('Complaint not filled properly');

        } else {

          window.location.reload();

          window.alert('Complaint filled');

        }

      } catch (error) {

        console.error('Error submitting complaint:', error);

      }

    }

  };

  return (

    <>

      <Nav />

      <div className='w-full mt-10'>

        <div className='max-w-md sm:max-w-lg mx-auto'>

          <form className='bg-[#f5f5f5] shadow-2xl rounded px-8 pt-6 pb-8 mb-4'>

            <div className='mb-10 text-center'>

              <h1 className='block text-gray-700 font-bold text-xl'>

                Enter Your Complaint

              </h1>

            </div>

            <div className='mb-4 flex justify-between'>

              <label className='block text-gray-700 text-sm font-bold mb-2' htmlFor='complaint\_type'>

                Type of Complaint :

              </label>

              <select

                className='px-10 border focus:outline-none focus:shadow-outline'

                id='complaint\_type'

                placeholder='select'

                onChange={(e) => {

                  setComplaintType(e.target.value);

                }}

              >

                <option value='' disabled selected>

                  Select your option

                </option>

                <option value='Electrical'>Electrical</option>

                <option value='Civil'>Civil</option>

                <option value='option3'>option3</option>

                <option value='option4'>Option 4</option>

              </select>

            </div>

            {error && complaint\_type.length <= 0 ? (

              <label className='text-red-500 text-sm px-2'>

                Complaint type cannot be empty

              </label>

            ) : (

              ''

            )}

            <div className='mb-6 mt-12'>

              <label className='block text-gray-700 text-sm font-bold mb-2' for='complaint'>

                Complaint:

              </label>

              <textarea

                className='shadow appearance-none border rounded w-full py-2 px-3 text-gray-700 leading-tight focus:outline-none focus:shadow-outline'

                id='complaint'

                rows='5'

                placeholder='Enter your Complaint'

                required

                onChange={(e) => {

                  setComplaint(e.target.value);

                }}

              ></textarea>

              {error && complaint.length <= 0 ? (

                <label className='text-red-500 text-sm px-2'>

                  Complaint cannot be empty

                </label>

              ) : (

                ''

              )}

            </div>

            <div className='flex items-center justify-center'>

              <button

                onClick={() => {

                  submitComplaint();

                  getRandomNumber();

                }}

                className='bg-blue-500 py-2 px-4 border-2 rounded-md shadow-2xl text-white  transition ease-in-out delay-150  hover:scale-110 hover:bg-green-500 duration-300 ...'

                type='button'

              >

                Submit

              </button>

            </div>

          </form>

        </div>

      </div>

    </>

  );

};

export default Main;

Backend:

app.post('/complaint', verifyUser, (req, res) => {

    const username = req.username

    const complaint = req.body.complaint

    const complaint\_type = req.body.complaint\_type

    const complaint\_id = req.body.complaint\_id

    db.query("INSERT INTO complaints (username,complaint,complaint\_type,complaint\_id) VALUES (?,?,?,?)", [username, complaint, complaint\_type, complaint\_id], (err, data) => {

        if (err) {

            console.log(err)

            return res.json("Error")

        }

        else {

            console.log(data)

            return res.send(data)

        };

    })

})

Summary

Frontend:

import React from 'react'

import Nav from '../Components/Nav'

import { useState, useEffect } from 'react';

import Axios from 'axios';

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

import Spinner from '../Components/Spinner';

const Summary = () => {

    const [data, setdata] = useState([])

    const [loading, setLoading] = useState(true);

    const [order, setorder] = useState("ASC");

    const [alldata, setalldata] = useState([])

    const [role, setrole] = useState("")

    const [search, setSearch] = useState('');

    useEffect(() => {

        const complaintdata = async () => {

            try {

                const response = await Axios.get('http://localhost:3002/complaint\_data');

                setdata(response.data);

            } catch (error) {

                console.error('Error fetching data:', error);

            } finally {

                setLoading(false);

            }

        };

        complaintdata();

    }, []);

    useEffect(() => {

        const allcomplaintdata = async () => {

            try {

                const response = await Axios.get('http://localhost:3002/allcomplaint\_data');

                setalldata(response.data);

            } catch (error) {

                console.error('Error fetching all data:', error);

            } finally {

                setLoading(false)

            }

        };

        allcomplaintdata();

    }, []);

    Axios.defaults.withCredentials = true;

    useEffect(() => {

        const userdata = async () => {

            try {

                const response = await Axios.get('http://localhost:3002')

                setrole(response.data[0].role)

            } catch (error) {

                console.error('Error fetching role:', error);

            }

        };

        userdata();

    }, []);

    const sorting = (col) => {

        if (order === "ASC") {

            const sorted = [...data].sort((a, b) =>

                a[col].toLowerCase() > b[col].toLowerCase() ? 1 : -1);

            setdata(sorted);

            setorder("DSC")

        }

        if (order === "DSC") {

            const sorted = [...data].sort((a, b) =>

                a[col].toLowerCase() < b[col].toLowerCase() ? 1 : -1);

            setdata(sorted);

            setorder("ASC")

        }

    }

    const [allorder, setallorder] = useState("ASC");

    const allsorting = (col) => {

        if (allorder === "ASC") {

            const sorted = [...alldata].sort((a, b) =>

                a[col].toLowerCase() > b[col].toLowerCase() ? 1 : -1);

            setalldata(sorted);

            setallorder("DSC")

        }

        if (allorder === "DSC") {

            const sorted = [...alldata].sort((a, b) =>

                a[col].toLowerCase() < b[col].toLowerCase() ? 1 : -1);

            setalldata(sorted);

            setallorder("ASC")

        }

    }

    const customSort = (a, b) => {

        const statusOrder = {

            pending: 1,

            assigned: 2,

            complete: 3,

        };

        return statusOrder[a.state] - statusOrder[b.state];

    };

    if (role === "General") {

        return (

            <div>

                <Nav />

                <div className="flex overflow-hidden shadow-2xl bg-[#f5f5f5] m-10 flex-col">

                    <div className='mb-10 mt-10 text-center'>

                        <h1 className='block text-gray-700 font-bold text-2xl'>Summary of Complaints</h1>

                    </div>

                    <div className='flex justify-center '>

                        <input onChange={(e) => setSearch(e.target.value.toLowerCase())}

                            type="text" id="default-input" className="bg-gray-50 border border-gray-300 mb-3 text-center justify-between items-center text-gray-900 text-sm rounded-full focus:ring-blue-500 focus:border-blue-500 block w-11/12 p-2.5 dark:bg-gray-700 dark:border-gray-600 dark:placeholder-gray-400 dark:text-white

      dark:focus:ring-blue-500 dark:focus:border-blue-500 " placeholder="Search....." /></div>

                    {loading ? <Spinner /> :

                        <div className="overflow-x-auto  sm:-mx-6 lg:-mx-8">

                            <div className="inline-block min-w-full py-2 sm:px-6 lg:px-8">

                                <div className="overflow-hidden">

                                    <table className="min-w-full text-left text-sm font-light ">

                                        <thead className="border-b font-medium dark:border-neutral-500">

                                            <tr className='text-center'>

                                                <th><button className="px-4 py-4 " onClick={() => sorting("complaint\_id")}>Complaint Id</button></th>

                                                <th><button className="px-6 py-4  " onClick={() => sorting("username")}>Username</button></th>

                                                <th> <button className="px-6 py-4 " onClick={() => sorting("complaint\_type")}>Complaint Type</button></th>

                                                <th> <button className="px-6 py-4 " onClick={() => sorting("complaint")}>Complaint</button></th>

                                                <th> <button className="px-6 py-4 " onClick={() => sorting("state")}>Status</button></th>

                                            </tr>

                                        </thead>

                                        <tbody>{data.filter((item) => {

                                            return search.toLowerCase() === ''

                                                ? item

                                                : (item.username.toLowerCase().includes(search)

                                                    || item.complaint\_type.toLowerCase().includes(search)

                                                    || item.complaint\_id.toLowerCase().includes(search)

                                                    || item.state.toLowerCase().includes(search)

                                                    || item.complaint.toLowerCase().includes(search));;

                                        }).sort((a, b) => {

                                            if (a.state === 'pending' && b.state !== 'pending') {

                                                return -1;

                                            }

                                            if (a.state !== 'pending' && b.state === 'pending') {

                                                return 1;

                                            }

                                            return customSort(a, b);})

                                        .map((item) => {

                                            return (

                                                <tr className="border-b dark:border-neutral-500 text-center" key={item.complaint\_id}>

                                                    <td className="whitespace-nowrap px-6 py-4 font-sans" >{item.complaint\_id}</td>

                                                    <td className="whitespace-nowrap px-6 py-4" >{item.username}</td>

                                                    <td className="whitespace-nowrap px-6 py-4" >{item.complaint\_type}</td>

                                                    <td className="whitespace-nowrap px-6 py-4" > {item.complaint.length > 30 ? (

                                                        <div>

                                                            {item.complaint.substr(0, 30)}

                                                            <br />

                                                            {item.complaint.substr(30, 90)}

                                                        </div>

                                                    ) : (

                                                        item.complaint

                                                    )}</td>

                                                    <td className={`border-b dark:border-neutral-500 text-center font-bold ${item.state === 'pending' ? 'text-red-700' : item.state === 'assigned' ? 'text-yellow-700' : item.state === 'complete' ? 'text-green-700' : ''}`}>{item.state}</td>

                                                    <td className="whitespace-nowrap px-6 py-4">  <Link to={`/Status/${item.complaint\_id}`}><button className="bg-blue-500 py-2 px-3 border-2 rounded-md shadow-2xl text-white  transition ease-in-out delay-150  hover:scale-110 hover:bg-green-500 duration-300 ...">Check</button> </Link></td>

                                                </tr>

                                            )

                                        })}

                                        </tbody>

                                    </table>

                                </div>

                            </div>

                        </div>}

                </div>

            </div>

        )

    }

    else {

        return (

            <div className=''>

                <Nav />

                <div className="flex overflow-hidden bg-[#f5f5f5] shadow-2xl m-10 flex-col">

                    <div></div>

                    <div className='mb-10 mt-10 text-center '>

                        <h1 className='block text-gray-700 font-bold mx-auto text-2xl'>Summary of Complaints</h1>

                    </div>

                    <div className='flex justify-center'>

                        <input onChange={(e) => setSearch(e.target.value.toLowerCase())}

                            type="text" id="default-input" className="bg-gray-50 border border-gray-300 mb-3 text-center justify-between items-center text-gray-900 text-sm rounded-lg focus:ring-blue-500 focus:border-blue-500 block w-11/12 p-2.5 dark:bg-gray-700 dark:border-gray-600 dark:placeholder-gray-400 dark:text-white

      dark:focus:ring-blue-500 dark:focus:border-blue-500 " placeholder='Search "username or complaint id or complaint type"' /></div>

                    {loading ? <Spinner /> :

                        <div className="overflow-x-auto  sm:-mx-6 lg:-mx-8">

                            <div className="inline-block min-w-full py-2 sm:px-6 lg:px-8">

                                <div className="overflow-hidden">

                                    <table className="min-w-full text-left text-sm font-light">

                                        <thead className="border-b font-medium dark:border-neutral-500">

                                            <tr className='text-center '>

                                                <th><button className="px-4 py-4 " onClick={() => allsorting("complaint\_id")}>Complaint Id</button></th>

                                                <th><button className="px-6 py-4  " onClick={() => allsorting("username")}>Username</button></th>

                                                <th> <button className="px-6 py-4 " onClick={() => allsorting("complaint\_type")}>Complaint Type</button></th>

                                                <th> <button className="px-6 py-4 " onClick={() => allsorting("complaint")}>Complaint</button></th>

                                                <th> <button className="px-6 py-4 " onClick={() => allsorting("state")}>Status</button></th>

                                            </tr>

                                        </thead>

                                        <tbody>{alldata.filter((item) => {

                                            return search.toLowerCase() === ''

                                                ? item

                                                : (item.username.toLowerCase().includes(search)

                                                    || item.complaint\_type.toLowerCase().includes(search)

                                                    || item.complaint\_id.toLowerCase().includes(search)

                                                    || item.state.toLowerCase().includes(search)

                                                    || item.complaint.toLowerCase().includes(search));

                                        }).sort((a, b) => {

                                            if (a.state === 'pending' && b.state !== 'pending') {

                                                return -1;

                                            }

                                            if (a.state !== 'pending' && b.state === 'pending') {

                                                return 1;

                                            }

                                            return customSort(a, b);})

                                            .map((item) => {

                                            return (

                                                <tr className="border-b dark:border-neutral-500 text-center " key={item.complaint\_id}>

                                                    <td className="whitespace-nowrap px-6 py-4" >{item.complaint\_id}</td>

                                                    <td className="whitespace-nowrap px-6 py-4" >{item.username}</td>

                                                    <td className="whitespace-nowrap px-6 py-4" >{item.complaint\_type}</td>

                                                    <td className="whitespace-nowrap px-6 py-4 " >{item.complaint.length > 30 ? (

                                                        <div>

                                                            {item.complaint.substr(0, 30)}

                                                            <br />

                                                            {item.complaint.substr(30, 90)}

                                                        </div>

                                                    ) : (

                                                        item.complaint

                                                    )}</td>

                                                    <td className={`border-b dark:border-neutral-500 text-center font-bold ${item.state === 'pending' ? 'text-red-700' : item.state === 'assigned' ? 'text-yellow-700' : item.state === 'complete' ? 'text-green-700' : ''}`}>{item.state}</td>

                                                     <td className="whitespace-nowrap px-6 py-4"> <Link to={`/Status/${item.complaint\_id}`}><button className="bg-blue-500 py-2 px-3 border-2 rounded-md shadow-2xl text-white  transition ease-in-out delay-150  hover:scale-110 hover:bg-green-500 duration-300 ...">Check</button></Link></td>

                                                </tr>

                                            )

                                        })}

                                        </tbody>

                                    </table>

                                </div>

                            </div>

                        </div>}

                </div>

            </div>

        )

    }

}

export default Summary

Backend:

app.get('/complaint\_data', verifyUser, (req, res) => {

    const username = req.username

    db.query("SELECT \* FROM  complaints WHERE username= ?", username, (err, data) => {

        if (err) {

            console.log(err)

            return res.json("Error")

        }

        if (data) {

            return res.send(data)

        };

    })

})

app.get('/allcomplaint\_data', verifyUser, (req, res) => {

    db.query("SELECT \* FROM  complaints", (err, data) => {

        if (err) {

            console.log(err)

            return res.json("Error")

        }

        if (data) {

            return res.send(data)

        };

    })

})

Complaint Status

Frontend:

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

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

import Axios from 'axios';

import Nav from '../Components/Nav';

import Delete from '../Components/Delete';

import Update from '../Components/Update';

import Assign from '../Components/Assign';

import Feedback from '../Components/Feedback';

import Complete from '../Components/Complete';

import Spinner from '../Components/Spinner'; // Import a spinner component for loading state

const Status = () => {

  const [complaint, setComplaint] = useState({});

  const [assigned, setAssigned] = useState(false);

  const [complete, setComplete] = useState(false);

  const [joindata, setJoinData] = useState({});

  const [role, setRole] = useState(false);

  const [assignData, setAssignData] = useState({});

  const [feedbackCheck, setFeedbackCheck] = useState(false);

  const [loading, setLoading] = useState(true); // Add loading state

  const [reason, setReason ] = useState(false)

  const params = useParams();

  const complaintId = params.complaint\_id;

  useEffect(() => {

    const fetchData = async () => {

      try {

        const [singleComplaintRes, tableJoinRes, userData] = await Promise.all([

          Axios.get(`http://localhost:3002/singlecomplaint\_data/${complaintId}`),

          Axios.get(`http://localhost:3002/tablejoin/${complaintId}`),

          Axios.get('http://localhost:3002'),

        ]);

        setComplaint(singleComplaintRes.data[0]);

        setJoinData(tableJoinRes.data[0]);

        const userRole = userData.data[0].role;

        if(userRole==="General"){

          setRole(true)

        }

        else{

          setRole(false)

        }

      } catch (error) {

        // Handle error here (e.g., show an error message)

        console.error('Error fetching data:', error);

      } finally {

        setLoading(false); // Hide loading state

      }

    };

    fetchData();

  }, [complaintId]);

  useEffect(() => {

    setAssigned(complaint.state === 'assigned');

    setComplete(complaint.state === 'complete');

  }, [complaint.state]);

  useEffect(() => {

    const assignCheck = async () => {

      try {

        const assignCheckRes = await Axios.get(

          `http://localhost:3002/assignCheck/${complaintId}`

        );

        setAssignData(assignCheckRes.data[0]);

        if(assignCheckRes.data[0].delay\_reason==="None"){

          setReason(false)

        }

        else{

          setReason(true)

        }

      } catch (error) {

        // Handle error here (e.g., show an error message)

        console.error('Error checking assignment:', error);

      }

    };

    assignCheck();

  }, [complaintId]);

  useEffect(() => {

    const feedbackData = async () => {

      try {

        const feedbackDataRes = await Axios.get(

          `http://localhost:3002/feedbackdata/${complaintId}`

        );

        setFeedbackCheck(feedbackDataRes.data.message === 'Nodata');

      } catch (error) {

        // Handle error here (e.g., show an error message)

        console.error('Error checking feedback:', error);

      }

    };

    feedbackData();

  }, [complaintId]);

  return (

    <div>

      <Nav />

      <div className='mt-5'>

        {loading ? (

          <Spinner /> // Show spinner while data is being fetched

        ) : (

          <div className=' mx-auto max-w-md sm:max-w-lg'>

            <div className=' shadow-2xl rounded bg-[#f5f5f5] px-8 pt-6 pb-8 mb-4 '>

              <div className='mb-10 text-center'>

                <h1 className='block text-gray-700 font-bold text-2xl'>

                  Complaint Status

                </h1>

              </div>

              <div className='divide-y-2'>

                <div>

                  <div className='flex justify-between mb-4'>

                    <label>Name</label>

                    <span>{joindata.name}</span>

                  </div>

                  {!role ? <div className='flex justify-between mb-4'>

                    <label>Username</label>

                    <span>{complaint.username}</span>

                  </div>:null}

                  <div className='flex justify-between mb-4'>

                    <label>Department</label>

                    <span>{joindata.department}</span>

                  </div>

                  <div className='flex justify-between mb-4'>

                    <label>Type of Complaint</label>

                    <span>{complaint.complaint\_type}</span>

                  </div>

                  <div className='flex justify-between mb-4'>

                    <label>Complaint</label>

                    <span className='text-end'>{complaint.complaint.length > 30 ? (

                      <div>

                        {complaint.complaint.substr(0, 30)}

                        <br />

                        {complaint.complaint.substr(30, 90)}

                      </div>

                    ) : (

                      complaint.complaint

                    )}</span>

                  </div>

                </div>

                <div className=''>

                  <div className='pt-4 flex justify-between'>

                    <label>Status</label>

                    <span>{complaint.state}</span>

                  </div>

                  {assigned || complete ? (

                    <div className="divide-y-2">

                    <div className='pt-2 flex justify-between'>

                      <label>Assigned To</label>

                      <span>{assignData.assigned\_to}</span>

                    </div>

                    {assigned&&reason?(<div className='pt-2 mt-4 '>

                    <h1 className='text-center'>Sorry Your Complaint is Delayed</h1>

                    <div  className='pt-4 flex justify-between'>

                    <label>Reason:</label>

                    <span className='text-end'>{assignData.delay\_reason.length > 30 ? (

                      <div>

                        {assignData.delay\_reason.substr(0, 30)}

                        <br />

                        {assignData.delay\_reason.substr(30, 90)}

                      </div>

                    ) : (

                      assignData.delay\_reason

                    )}</span></div>

                  </div>):null}

                  </div>

                  ) : null}

                </div>

              </div>

            </div>

            <div className='mx-auto text-center mb-4'>

              <Delete />

            </div>

            <div className='mx-auto text-center mb-10 mt-10 '>

              {assigned && !role ? <Complete /> : null}

            </div>

            <div className='mx-auto text-center mt-3'>

              {role ? <Update /> : <Assign />}

            </div>

            <div className='mx-auto text-center mb-4'>

              {complete && feedbackCheck ? <Feedback /> : null}

            </div>

          </div>

        )}

      </div>

    </div>

  );

};

export default Status;

Backend:

app.get('/singlecomplaint\_data/:complaint\_id', (req, res) => {

    const complaint\_id = req.params.complaint\_id

    db.query("SELECT \* FROM  complaints WHERE complaint\_id= ?", [complaint\_id], (err, data) => {

        if (err) {

            console.log(err)

            return res.json("Error")

        }

        else {

            return res.send(data)

        };

    })

})

app.get('/tablejoin/:complaint\_id', (req, res) => {

    const complaint\_id= req.params.complaint\_id

    db.query("SELECT client.department, client.name,client.role,complaints.complaint\_id FROM client INNER JOIN complaints ON complaints.username=client.username WHERE complaint\_id=?",[complaint\_id] ,(err, data) => {

        if (err) {

            console.log(err)

            return res.json("Error")

        }

        if (data) {

            return res.send(data)

        };

    })

})

Feedbacks

Frontend:

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

import Axios from 'axios';

import Nav from '../Components/Nav';

import Spinner from '../Components/Spinner';

const Feedbacks = () => {

  const [data, setData] = useState([]);

  const [loading, setLoading] = useState(false);

  const [order, setOrder] = useState('ASC');

  const [search, setSearch] = useState('');

  useEffect(() => {

    const fetchFeedbackData = async () => {

      setLoading(true);

      try {

        const res = await Axios.get('http://localhost:3002/feedbackjoin');

        setData(res.data);

      } catch (error) {

        console.error('Error fetching feedback data:', error);

        window.alert('An error occurred while fetching feedback data');

      } finally {

        setLoading(false);

      }

    };

    fetchFeedbackData();

  }, []);

  const sorting = (col) => {

    if (order === 'ASC') {

      const sorted = [...data].sort((a, b) => (a[col].toLowerCase() > b[col].toLowerCase() ? 1 : -1));

      setData(sorted);

      setOrder('DSC');

    }

    if (order === 'DSC') {

      const sorted = [...data].sort((a, b) => (a[col].toLowerCase() < b[col].toLowerCase() ? 1 : -1));

      setData(sorted);

      setOrder('ASC');

    }

  };

  return (

    <div>

      <Nav />

      <div className="flex overflow-hidden bg-[#f5f5f5] shadow-2xl m-10 flex-col">

        <div className="mb-10 mt-10 text-center">

          <h1 className="block text-gray-700 font-bold text-2xl">User's Feedback</h1>

        </div>

        <div className="flex justify-center">

          <input

            onChange={(e) => setSearch(e.target.value.toLowerCase())}

            type="text"

            id="default-input"

            className="bg-gray-50 border border-gray-300 text-center justify-between items-center text-gray-900 text-sm rounded-lg focus:ring-blue-500 focus:border-blue-500 block w-11/12 p-2.5 dark:bg-gray-700 dark:border-gray-600 dark:placeholder-gray-400 dark:text-white dark:focus:ring-blue-500 dark:focus:border-blue-500"

            placeholder='Search "Status or complaint id or complaint type"'

          />

        </div>

        <div className="overflow-auto sm:-mx-6 lg:-mx-8">

          <div className="inline-block min-w-full py-2 sm:px-6 lg:px-8">

            <div className="overflow-hidden">

              <table className="min-w-full text-left text-sm font-light">

                <thead className="border-b font-medium dark:border-neutral-500">

                  <tr className="text-center">

                    <th>

                      <button className="px-4 py-4 " onClick={() => sorting('complaint\_id')}>

                        Complaint Id

                      </button>

                    </th>

                    <th>

                      <button className="px-6 py-4 " onClick={() => sorting('username')}>

                      Complaint

                      </button>

                    </th>

                    <th>

                      <button className="px-6 py-4 " onClick={() => sorting('complaint')}>

                        Assigned To

                      </button>

                    </th>

                    <th>

                      <button className="px-6 py-4 " onClick={() => sorting('feedback')}>

                        Feedback

                      </button>

                    </th>

                    <th>

                      <button className="px-6 py-4 " onClick={() => sorting('feedback\_date')}>

                        Feedback date

                      </button>

                    </th>

                    <th>

                      <button className="px-6 py-4 " onClick={() => sorting('rate')}>

                        Feedback Rating

                      </button>

                    </th>

                  </tr>

                </thead>

                <tbody>

                  {loading ? (

                    // Show a spinner while loading

                    <tr>

                      <td colSpan="6" className="text-center py-4">

                        <div className="spinner"><Spinner/></div>

                      </td>

                    </tr>

                  ) : data.length === 0 ? (

                    // Show 'No Data' message when data is empty

                    <tr>

                      <td colSpan="6" className="text-center py-4">

                        <img className="mx-auto w-1/2 mt-12" src="/images/nodata1.avif" alt="" />

                      </td>

                    </tr>

                  ) : (

                    // Render the table rows when data is available

                    data

                      .filter((item) => {

                        return (

                          search.toLowerCase() === '' ||

                          item.assigned\_to.toLowerCase().includes(search) ||

                          item.feedbacks.toLowerCase().includes(search) ||

                          item.complaint\_id.toLowerCase().includes(search) ||

                          item.complaint.toLowerCase().includes(search) ||

                          item.feedback\_date.toLowerCase().includes(search) ||

                          item.rate.toLowerCase().includes(search)

                        );

                      })

                      .map((item) => (

                        <tr className="border-b dark:border-neutral-500 text-center" key={item.complaint\_id}>

                          <td className="whitespace-nowrap px-6 py-4 transition ease-in-out ">{item.complaint\_id}</td>

                          <td className="whitespace-nowrap px-6 py-4 transition ease-in-out ">{item.complaint}</td>

                          <td className="whitespace-nowrap px-6 py-4 transition ease-in-out ">{item.assigned\_to}</td>

                          <td className="whitespace-nowrap px-6 py-4 transition ease-in-out ">{item.feedbacks}</td>

                          <td className="whitespace-nowrap px-6 py-4 transition ease-in-out  ">{item.feedback\_date.substring(0, 10)}</td>

                          <td className="whitespace-nowrap px-6 py-4 transition ease-in-out delay-150  hover:scale-110 duration-300"><div className="rating flex justify-center">

                            {Array.from({ length: 5 }, (\_, index) => (

                              <svg

                                key={index}

                                xmlns="http://www.w3.org/2000/svg"

                                fill={index + 1 <= item.rate ? '#ffac00' : '#B7B8BC'}

                                className="h-6 w-6"

                                viewBox="0 0 24 24"

                                strokeLinecap="round"

                                strokeLinejoin="round"

                                strokeWidth="2"

                                stroke=""

                              >

                                <path d="M12 2l3.09 6.09L22 9.29l-4.55 4.41L18 22l-6-5-6 5 1.55-8.3L2 9.3l6.91-1.21L12 2z"></path>

                              </svg>

                            ))}

                          </div></td>

                        </tr>

                      ))

                  )}

                </tbody>

              </table>

            </div>

          </div>

        </div>

      </div>

    </div>

  );

};

export default Feedbacks;

Backend:

app.get('/feedbackjoin', (req, res) => {

    db.query("SELECT complaints.username, complaints.complaint ,feedback.feedbacks,feedback.feedback\_date,feedback.rate,complaints.complaint\_id,assigned.assigned\_to FROM complaints INNER JOIN feedback ON complaints.complaint\_id=feedback.complaint\_id INNER JOIN assigned ON assigned.complaint\_id=feedback.complaint\_id",(err, data) => {

        if (err) {

            console.log(err)

            return res.json("Error")

        }

        if (data) {

            return res.send(data)

        };

    })

})

Assign Complaint

Frontend:

import React from 'react'

import { useEffect ,useState} from 'react'

import { useNavigate, useParams } from 'react-router-dom'

import  Axios from 'axios'

const Assign = () => {

  const [assign, setassign] = useState(false)

  const assignButton=()=>{

    setassign(true)

  }

  const cancelButtons=()=>{

    setassign(false)

  }

  const [showButton, setShowButton] = useState(true);

  const [worker\_role, setWorker\_Role ]= useState("")

  const [worker, setWorker ]= useState("")

  const toggleButton = () => {

    setShowButton(!showButton);

  };

  const [Complaint, setComplaint] = useState({});

  const params = useParams();

  const complaint\_id = params.complaint\_id

  useEffect(() => {

    const state = "assigned"

    setnewstate(state)

  }, [])

  useEffect(() => {

    const singlecomplaintdata = async () => {

      try {

        const response = await Axios.get(

          `http://localhost:3002/singlecomplaint\_data/${complaint\_id}`

        );

        setComplaint(response.data[0]);

        const complaintType = response.data[0].complaint\_type;

        setWorker\_Role(complaintType);

        if (Complaint.state !== 'pending') {

          setShowButton(false);

        } else {

          setShowButton(true);

        }

      } catch (error) {

        console.log('Error fetching single complaint data:', error);

      }

    };

    singlecomplaintdata();

  }, [complaint\_id,Complaint.state,Complaint.complaint\_type]);

  useEffect(() => {

    const workersData = async () => {

      try {

        console.log(worker\_role);

        const response = await Axios.get(

          `http://localhost:3002/workers\_data/${worker\_role}`

        );

        setWorker(response.data);

      } catch (error) {

        console.log('Error fetching worker data:', error);

      }

    };

    workersData();

  }, [worker\_role]);

   console.log(worker)

    const [assignedto, setassignedto] = useState("")

    const [newstate, setnewstate] = useState("")

    const navigate = useNavigate();

    const assignComplaint = async () => {

      try {

        await Axios.all([

          Axios.put(`http://localhost:3002/assignedstate/${complaint\_id}`, { newstate: newstate }),

          Axios.post(`http://localhost:3002/assigned/${complaint\_id}`, { assignedto: assignedto }),

          Axios.put("http://localhost:3002/increment\_assigned",{  assignedto: assignedto })

        ]);

        window.alert('Complaint is assigned');

        navigate('/Summary');

      } catch (error) {

        console.log('Error during complaint assignment:', error);

        window.alert('Error assigning complaint. Please try again later.');

      }

    };

  return (

    <div  className={`shadow-2xl rounded px-8 pt-6 pb-8 mb-4 ${

      assign ? 'bg-[#f5f5f5]' : 'bg-white'

    } ${assign ? 'shadow-2xl' : 'shadow-none'}`}>

     {showButton&& <button  className="bg-blue-500 hover:bg-blue-700 text-white font-bold py-2 px-4 rounded" onClick={()=>{toggleButton();assignButton()}} id="btn">Assign</button>}

      {assign?<div>

        <div className='mb-10 mt-2 text-center'>

                <h1 className='block text-gray-700 font-bold text-xl'>

                  Assign The Complaint

                </h1>

              </div>

        <div className="mb-4 flex  justify-between">

                                <label className="block text-gray-700 text-sm font-bold mb-2" >Status :</label>

                                <span className='px-10  '  >

                                    Assigned

                                </span>

                            </div>

                            <div className="mb-10 mt-5 flex  justify-between">

                                <label className="block text-gray-700 text-md font-bold mb-2">Select the Worker :</label>

                                <select

            className='px-10 focus:outline-none focus:shadow-outline border'

            id="cars"

            placeholder='select'

            defaultValue={'default'}

            onChange={(e) => { setassignedto(e.target.value) }} // Store the selected worker in state

          >

            <option value="default" disabled>Select your option</option>

            {worker.map((workerData) => (

              <option key={workerData.worker\_name} value={workerData.worker\_name}>{workerData.worker\_name}</option>

            ))}

          </select>

                            </div>

     <div className='mt-4 mb-4'> <button  className="bg-blue-500 py-2 px-4 border-2 rounded-md mx-3 shadow-2xl text-white  transition ease-in-out delay-150  hover:scale-110 hover:bg-green-500 duration-300 ..." onClick={assignComplaint}>Submit

        </button >

        <button  className="bg-blue-500 py-2 px-4 border-2 rounded-md shadow-2xl text-white mx-3 transition ease-in-out delay-150  hover:scale-110 hover:bg-red-500 duration-300 ..." onClick={()=>{cancelButtons();toggleButton()}}>cancel</button></div></div>:""}

    </div>

  )

}

export default Assign;

Backend:

app.post('/assigned/:complaint\_id', (req, res) => {

    const complaint\_id = req.params.complaint\_id

    const assigned\_to = req.body.assignedto

    db.query("INSERT INTO assigned (complaint\_id,assigned\_to) VALUES (?,?)", [complaint\_id,assigned\_to], (err, data) => {

        if (err) {

            console.log(err)

            return res.json("Error")

        }

        if (data) {

            console.log("assigned")

            return res.send({ message: "assigned" })

        };

    })

})

app.put('/assignedstate/:complaint\_id', (req, res) => {

    const complaint\_id = req.params.complaint\_id

    const newstate = req.body.newstate

    const complete\_date = req.body.date

    console.log(newstate)

    db.query("UPDATE complaints SET state=?,complete\_date=? WHERE complaint\_id=?", [newstate,complete\_date,complaint\_id], (err, data) => {

        if (err) {

            console.log(err)

            return res.json("Error")

        }

        if (data) {

            console.log(data)

            return res.send({ message: "assigned" })

        };

    })

})

app.put('/increment\_assigned', (req, res) => {

    const worker\_name = req.body.assignedto

    db.query("UPDATE workers SET complaint\_assigned=complaint\_assigned+1 WHERE worker\_name=?", [worker\_name], (err, data) => {

        if (err) {

            console.log(err)

            return res.json("Error")

        }

        if (data) {

            return res.send(data)

        };

    })

})

Complete Complaint

Frontend:

import React from 'react'

import { useState, useEffect, useRef } from 'react'

import { useNavigate, useParams } from 'react-router-dom'

import Axios from 'axios'

const Complete = () => {

  const params = useParams();

  const complaint\_id = params.complaint\_id

  useEffect(() => {

    const state = "complete"

    setnewstate(state)

  }, [])

  const [newstate, setnewstate] = useState("")

  const navigate = useNavigate();

  const [date, setDate] = useState("")

  const [delay\_reason, setdelay\_reason ] = useState("")

  const [worker , setWorker] = useState("")

  useEffect(() => {

    const todayDate = () => {

      const currentDate = new Date();

      const year = currentDate.getFullYear();

      const month = String(currentDate.getMonth() + 1).padStart(2, '0');

      const day = String(currentDate.getDate()).padStart(2, '0');

      const formattedDate = `${year}-${month}-${day}`;

      setDate(formattedDate)

    }

    todayDate()

  }, [])

  const newstateRef = useRef();

  newstateRef.current = newstate;

  useEffect(() => {

     const AssignCheck = async() =>{

      try{

        const res = await Axios.get(`http://localhost:3002/assignCheck/${complaint\_id}`)

        setWorker(res.data[0].assigned\_to)

      }catch(error){

        console.log('Error during Assign Check:', error);

        window.alert('Error getting worker. Please try again later.')

      }

     }

     AssignCheck();

  }, [complaint\_id])

  const compeletestate = async () => {

    try {

      const [assignres, workerRes] = await Promise.all([

       Axios.put( `http://localhost:3002/assignedstate/${complaint\_id}`, { newstate: newstate, date: date }),

       Axios.put("http://localhost:3002/increment\_completed",{worker:worker} )

       ]);

      if (assignres.data.message && workerRes.data.message === "assigned") {

        window.alert('Complaint is updated');

        navigate('/Summary');

      }

      else {

        window.alert('Error updating complaint. Please try again later.')

      }

    } catch (error) {

      console.log('Error during complaint update:', error);

      window.alert('Error updating complaint. Please try again later.');

    }

  };

  const complaintDelay = async () => {

    try {

      const response = await Axios.put(`http://localhost:3002/complaintdelay/${complaint\_id}`, { delay\_reason:delay\_reason });

      if (response.data.message === "delayed") {

        window.alert('Delay reason is given');

        navigate('/Summary');

      }

      else {

        window.alert('Error updating delay reason. Please try again later.')

      }

    } catch (error) {

      console.log('Error during delay reason update:', error);

      window.alert('Error updating delay reason. Please try again later.')

    }

  };

  return (

    <div className='shadow-2xl rounded px-8 pt-6 pb-8 mb-4 bg-[#f5f5f5]'>

      <div>

        <div className='mb-10 text-center'>

          <h1 className='block text-gray-700 font-bold text-xl'>

            Complaint Is Completed ?

          </h1>

        </div>

        <div className="mb-4 mt-2 flex  justify-between">

          <label className="block text-gray-700 text-md font-bold mb-2" >If Yes Then Click  :</label>

          <button className=" bg-blue-500 py-2 px-4 border-2 rounded-md shadow-2xl text-white  transition ease-in-out delay-150  hover:scale-110 hover:bg-green-500 duration-300 ..." onClick={compeletestate}>Yes</button >

        </div>

        <div className="mb-4 mt-10 flex-col ">

          <label className="block text-gray-700 text-md font-bold mb-5 text-start" >If No Write The Reason Of Delay :</label>

          <textarea

            className='shadow appearance-none border rounded w-full py-2 px-3 mb-5 text-gray-700 leading-tight focus:outline-none focus:shadow-outline'

            id='delay'

            rows='5'

            placeholder='Write the reason of delay'

            required

            onChange={(e)=>{setdelay\_reason(e.target.value)}}

          ></textarea>

          <button  className=" bg-blue-500 py-2 px-4 border-2 rounded-md mt-3 shadow-2xl text-white  transition ease-in-out delay-150  hover:scale-110 hover:bg-red-500 duration-300 ..."onClick={complaintDelay} >Submit Reason</button >

        </div>

        <div className='mt-10 '>

        </div></div>

    </div>

  )

}

export default Complete;

Backend:

app.put('/assignedstate/:complaint\_id', (req, res) => {

    const complaint\_id = req.params.complaint\_id

    const newstate = req.body.newstate

    const complete\_date = req.body.date

    console.log(newstate)

    db.query("UPDATE complaints SET state=?,complete\_date=? WHERE complaint\_id=?", [newstate,complete\_date,complaint\_id], (err, data) => {

        if (err) {

            console.log(err)

            return res.json("Error")

        }

        if (data) {

            console.log(data)

            return res.send({ message: "assigned" })

        };

    })

})

app.put('/increment\_completed', (req, res) => {

    const worker\_name = req.body.worker

    console.log(worker\_name)

    db.query("UPDATE workers SET complaint\_completed=complaint\_completed+1 WHERE worker\_name=?", [worker\_name], (err, data) => {

        if (err) {

            console.log(err)

            return res.json("Error")

        }

        if (data) {

            return res.send({message:"assigned"})

        };

    })

})

Delete Complaint

Frontend:

import React from 'react'

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

import  Axios  from 'axios';

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

const Delete = () => {

    const params = useParams();

    const complaint\_id = params.complaint\_id

    const navigate = useNavigate();

    const complaintDelete = async () => {

      try {

        await Axios.delete(`http://localhost:3002/singlecomplaint\_delete/${complaint\_id}`);

        window.alert('Complaint deleted');

        navigate('/Summary');

      } catch (error) {

        console.log('Error during complaint delete:', error);

        window.alert('Error deleting complaint. Please try again later.');

      }

    };

  return (

    <div ><button  className="bg-blue-500 py-2 px-4 border-2 rounded-md shadow-2xl text-white  transition ease-in-out delay-150  hover:scale-110 hover:bg-red-500 duration-300 ..." onClick={complaintDelete}>Delete</button></div>

  )

}

export default Delete

Backend:

app.delete('/singlecomplaint\_delete/:complaint\_id', (req, res) => {

    const complaint\_id = req.params.complaint\_id

    db.query("DELETE FROM complaints WHERE complaint\_id= ?", [complaint\_id], (err, data) => {

        if (err) {

            console.log(err)

            return res.json("Error")

        }

        if (data) {

            return res.send(data)

        };

    })

})

Complaint Feedback

Frontend:

import React, { useState } from 'react';

import Axios from 'axios';

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

const Feedback = () => {

  const [feedbacks, setFeedbacks] = useState('');

  const [rate, setRate] = useState([]);

  const [loading, setLoading] = useState(false); // New state to track loading

  const params = useParams();

  const complaint\_id = params.complaint\_id;

  const submitFeedback = async () => {

    setLoading(true);

    try {

      const res = await Axios.post(`http://localhost:3002/feedback/${complaint\_id}`, {

        feedbacks: feedbacks,

        rate: rate,

      });

      if (res.data.message === 'sumbited') {

        window.alert('Feedback submitted');

        window.location.reload();

      } else {

        window.alert('Feedback not submitted');

      }

    } catch (error) {

      console.error('Error submitting feedback:', error);

      window.alert('An error occurred while submitting feedback');

    } finally {

      setLoading(false);

    }

  };

  return (

    <div className='shadow-2xl rounded px-8 pt-6 pb-8 mb-4 bg-[#f5f5f5]'>

       <div className='mb-10 text-center mt-2'>

                <h1 className='block text-gray-700 font-bold text-xl'>

                  How Was the Service ?

                </h1>

              </div>

      <div className="mb-4 mt-2 flex  justify-between ">

        <label className="block text-gray-700 text-sm font-bold mb-2" for="email">

          Rate Service:

        </label>

        <div className="rating">

          <input

            type="radio"

            value="1"

            name="rating-2"

            onChange={(e) => {

              setRate(e.target.value);

            }}

            className="mask mask-star-2 bg-orange-400"

          />

          <input

            type="radio"

            value="2"

            name="rating-2"

            onChange={(e) => {

              setRate(e.target.value);

            }}

            className="mask mask-star-2 bg-orange-400"

          />

          <input

            type="radio"

            value="3"

            name="rating-2"

            onChange={(e) => {

              setRate(e.target.value);

            }}

            className="mask mask-star-2 bg-orange-400"

          />

          <input

            type="radio"

            value="4"

            name="rating-2"

            onChange={(e) => {

              setRate(e.target.value);

            }}

            className="mask mask-star-2 bg-orange-400"

          />

          <input

            type="radio"

            value="5"

            name="rating-2"

            onChange={(e) => {

              setRate(e.target.value);

            }}

            className="mask mask-star-2 bg-orange-400"

          />

        </div>

      </div>

      <div>

        <textarea

          className="shadow appearance-none border rounded w-full py-2 px-3 text-gray-700 leading-tight focus:outline-none focus:shadow-outline"

          id="message"

          rows="5"

          placeholder="Enter your Feedback"

          required

          onChange={(e) => {

            setFeedbacks(e.target.value);

          }}

        ></textarea>

        <div className="mt-4 mb-4">

          <button

            className="bg-blue-500 py-2 px-4 border-2 rounded-md shadow-2xl text-white  transition ease-in-out delay-150  hover:scale-110 hover:bg-green-500 duration-300 ..."

            onClick={submitFeedback}

            disabled={loading} // Disable the button while loading

          >

            {loading ? 'Submitting...' : 'Submit'}

          </button>

          {/\* You can also add a spinner while loading \*/}

          {loading && <div className="spinner"></div>}

        </div>

      </div>

    </div>

  );

};

export default Feedback;

Backend:

app.post('/feedback/:complaint\_id', verifyUser, (req, res) => {

    const feedbacks = req.body.feedbacks

    const complaint\_id = req.params.complaint\_id

    const rate = req.body.rate

    db.query("INSERT INTO feedback (complaint\_id,feedbacks,rate) VALUES (?,?,?)", [complaint\_id,feedbacks,rate], (err, data) => {

        if (err) {

            console.log(err)

            return res.json("Error")

        }

        else {

            console.log(data)

            return res.send({message:"sumbited"})

        };

    })

})

Complaint Update

Frontend:

import React from 'react'

import { useEffect ,useState} from 'react'

import { useNavigate, useParams } from 'react-router-dom'

import  Axios from 'axios'

const Update = () => {

  const [update, setupdate] = useState(false)

  const updateButton=()=>{

    setupdate(true)

  }

  const cancelButtons=()=>{

    setupdate(false)

  }

  const [showButton, setShowButton] = useState(true);

  const toggleButton = () => {

    setShowButton(!showButton);

  };

  const [Complaint, setComplaint] = useState({});

  const params = useParams();

  const complaint\_id = params.complaint\_id

  useEffect(() => {

    const singlecomplaintdata = async () => {

      try {

        const response = await Axios.get(

          `http://localhost:3002/singlecomplaint\_data/${complaint\_id}`

        );

        setComplaint(response.data[0]);

        if (Complaint.state !== 'pending') {

          setShowButton(false);

        } else {

          setShowButton(true);

        }

      } catch (error) {

        console.log('Error fetching single complaint data:', error);

      }

    };

    singlecomplaintdata();

  }, [complaint\_id,Complaint.state]);

    const [newcomplaint, setnewcomplaint] = useState("")

    const navigate = useNavigate();

    const updateComplaint = async () => {

      try {

        await Axios.put(`http://localhost:3002/newcomplaint/${complaint\_id}`, {

          newcomplaint: newcomplaint,

        });

        window.alert('Complaint is updated');

        navigate('/Summary');

      } catch (error) {

        console.log('Error during complaint update:', error);

        window.alert('Error updating complaint. Please try again later.');

      }

    };

  return (

    <div>

     {showButton&& <button  className="bg-blue-500 py-2 px-4 border-2 rounded-md shadow-2xl text-white  transition ease-in-out delay-150  hover:scale-110 hover:bg-green-500 duration-300 ..." onClick={()=>{toggleButton();updateButton()}} id="btn">update</button>}

      {update?<div><textarea className="shadow appearance-none border rounded w-full py-2 px-3 text-gray-700 leading-tight focus:outline-none focus:shadow-outline" id="message" rows="5" placeholder="Enter your Complaint" required onChange={(e) => {

                            setnewcomplaint(e.target.value)

                        }}></textarea>

     <div className='mt-4 mb-4'> <button  className="bg-blue-500 py-2 px-4 border-2 rounded-md mx-3 shadow-2xl text-white  transition ease-in-out delay-150  hover:scale-110 hover:bg-green-500 duration-300 ..." onClick={updateComplaint}>Submit

        </button >

        <button  className="bg-blue-500 py-2 px-4 border-2 rounded-md mx-3 shadow-2xl text-white  transition ease-in-out delay-150  hover:scale-110 hover:bg-red-500 duration-300 ..." onClick={()=>{cancelButtons();toggleButton()}}>cancel</button></div></div>:""}

    </div>

  )

}

export default Update;

Backend:

app.put('/newcomplaint/:complaint\_id', verifyUser, (req, res) => {

    const complaint\_id = req.params.complaint\_id

    const newcomplaint = req.body.newcomplaint

    db.query("UPDATE complaints SET complaint=? WHERE complaint\_id= ?", [newcomplaint, complaint\_id], (err, data) => {

        if (err) {

            console.log(err)

            return res.json("Error")

        }

        if (data) {

            return res.send({ message: "updated" })

        };

    })

})

NavBar Component

Frontend:

import { useEffect, useState } from 'react';

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

import { Menu, X,ChevronUp,ChevronDown } from 'lucide-react';

import Axios from 'axios';

const navstyle = {

  margin: '1rem',

  color: 'white',

  font: 'bold',

};

const boxStyleHover = {

  height: '50px',

  backgroundColor: 'white',

  color: 'black',

  borderRadius: '5px',

  boxShadow: '2px 2px 4px rgba(0, 0, 0, 0.2)',

  display: 'flex',

  justifyContent: 'center',

  alignItems: 'center',

};

const submenu = {

  hover: 'black',

  font: 'bold',

  color:"white",

};

const boxStylemenuHover = {

    height: '50px',

    width:'150px',

    backgroundColor: 'white',

    color: 'black',

    borderRadius: '5px',

    boxShadow: '2px 2px 4px rgba(0, 0, 0, 0.2)',

    display: 'flex',

    justifyContent: 'center',

    alignItems: 'center',

  };

const Submenu = ({ isOpen ,ismenuOpen}) => {

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

    const handleMouseEnter = (index) => {

      const updatedHoverState = [...isHovered];

      updatedHoverState[index] = true;

      setIsHovered(updatedHoverState);

    };

    const handleMouseLeave = (index) => {

      const updatedHoverState = [...isHovered];

      updatedHoverState[index] = false;

      setIsHovered(updatedHoverState);

    };

  return (

    <>

      {isOpen && (

        <div className='hidden lg:flex bg-gradient-to-r  duration-300   from-[#012f66] to-[#0568a1] items-center justify-between text-center p-4 shadow-lg  '>

            <NavLink className="block mt-2 hover:text-black mx-auto duration-300" style={isHovered[0] ? { ...submenu, ...boxStylemenuHover } : submenu}

          onMouseEnter={() => handleMouseEnter(0)}

          onMouseLeave={() => handleMouseLeave(0)} to="/Main">Add Complaint</NavLink>

            <NavLink className="block mt-2 hover:text-black mx-auto duration-300" style={isHovered[1] ? { ...submenu, ...boxStylemenuHover } : submenu}

          onMouseEnter={() => handleMouseEnter(1)}

          onMouseLeave={() => handleMouseLeave(1)}>Delete Complaint</NavLink>

            <NavLink className="block mt-2 hover:text-black mx-auto duration-300" style={isHovered[2] ? { ...submenu, ...boxStylemenuHover } : submenu}

          onMouseEnter={() => handleMouseEnter(2)}

          onMouseLeave={() => handleMouseLeave(2)}>Modify Complaint</NavLink>

        </div>

      )}

    </>

  );

};

const NavLinks = ({isLGScreen, toggleNavUser }) => {

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

  const handleMouseEnter = (index) => {

    const updatedHoverState = [...isHovered];

    updatedHoverState[index] = true;

    setIsHovered(updatedHoverState);

  };

  const handleMouseLeave = (index) => {

    const updatedHoverState = [...isHovered];

    updatedHoverState[index] = false;

    setIsHovered(updatedHoverState);

  };

  Axios.defaults.withCredentials = true;

  const [role, setRole] = useState("");

  useEffect(() => {

    const userRole = async () => {

      try {

        const res = await Axios.get('http://localhost:3002');

        setRole(res.data[0].role);

      } catch (error) {

        console.error('Error fetching user role:', error);

      }

    };

    userRole();

  }, []);

  const [isthisOpen, setisThisOpen] = useState(false);

  const toggleNavedUser = () => {

    setisThisOpen(!isthisOpen);

  };

  const [isSubMenuOpen, setIsSubMenuOpen] = useState(false);

  const toggleSubMenu = () => {

    toggleNavedUser();

    setIsSubMenuOpen(!isSubMenuOpen);

  };

  if (role === "General") {

    return (

      <>

           <NavLink

          className="text-2xl px-4  duration-300"

          style={isHovered[3] && isLGScreen ? { ...navstyle, ...boxStyleHover } : navstyle}

          onMouseEnter={() => handleMouseEnter(3)}

          onMouseLeave={() => handleMouseLeave(3)}

          onClick={() => {

            toggleNavUser();

            toggleNavedUser();

            toggleSubMenu()

          }}

        >

           <span  className='flex justify-between'>

            Lodge Complaint

            {isLGScreen ? (

              isSubMenuOpen ? (

                <ChevronUp className="inline ml-2 lg:hidden " />

              ) : (

                <ChevronDown className="inline ml-2 lg:hidden" />

              )

            ) : (

              <button className="focus:outline-none" onClick={toggleSubMenu}>

                {isSubMenuOpen  ? <ChevronUp /> : <ChevronDown />}

              </button>

            )}  </span>

          { isthisOpen &&   (

            <div className="lg:hidden mt-2   flex-col items-center justify-between  text-center p-4">

              <NavLink className="block mt-3 hover:text-black mx-auto text-md " style={submenu} to="/Main">

                Add Complaint

              </NavLink>

              <NavLink className="block mt-3 hover:text-black mx-auto text-md " style={submenu}>

                Modify Complaint

              </NavLink>

              <NavLink className="block mt-3 hover:text-black mx-auto text-md " style={submenu}>

                Delete Complaint

              </NavLink>

            </div>

          )}

        </NavLink>

        <NavLink

          style={isHovered[4] && isLGScreen ? { ...navstyle, ...boxStyleHover } : navstyle}

          onMouseEnter={() => handleMouseEnter(4)}

          onMouseLeave={() => handleMouseLeave(4)}

          className="hover:text-black text-2xl px-4 duration-500"

          to="/Summary"

        >

          Summary

        </NavLink>

        <NavLink

          style={isHovered[5] && isLGScreen ? { ...navstyle, ...boxStyleHover } : navstyle}

          onMouseEnter={() => handleMouseEnter(5)}

          onMouseLeave={() => handleMouseLeave(5)}

          className="hover:text-black text-2xl px-4 duration-500"

          to="/Profile"

        >

          Profile

        </NavLink>

      </>

    );

  } else {

    return (

      <>

        <NavLink style={isHovered[6] && isLGScreen ? { ...navstyle, ...boxStyleHover } : navstyle}

          onMouseEnter={() => handleMouseEnter(6)}

          onMouseLeave={() => handleMouseLeave(6)}

          className="hover:text-black text-2xl px-4 duration-500" to="/workers">Workers</NavLink>

        <NavLink style={isHovered[0] && isLGScreen ? { ...navstyle, ...boxStyleHover } : navstyle}

          onMouseEnter={() => handleMouseEnter(0)}

          onMouseLeave={() => handleMouseLeave(0)}

          className="hover:text-black text-2xl px-4 duration-500" to="/Feedbacks">Feedbacks</NavLink>

        <NavLink style={isHovered[1] && isLGScreen ? { ...navstyle, ...boxStyleHover } : navstyle}

          onMouseEnter={() => handleMouseEnter(1)}

          onMouseLeave={() => handleMouseLeave(1)}

          className="hover:text-black text-2xl px-4 duration-500" to="/Summary">Summary</NavLink>

        <NavLink style={isHovered[2] && isLGScreen ? { ...navstyle, ...boxStyleHover } : navstyle}

          onMouseEnter={() => handleMouseEnter(2)}

          onMouseLeave={() => handleMouseLeave(2)}

          className="hover:text-black text-2xl px-4 duration-500" to="/Profile">Profile</NavLink>

      </>

    );

  }

};

const Nav = () => {

  const [isOpen, setIsOpen] = useState(false);

  const [ismenuOpen, setIsmenuOpen] = useState(false);

  const [isLGScreen, setIsLGScreen] = useState(false);

  const toggleNavUser = () => {

    setIsOpen(!isOpen);

    setIsmenuOpen(!ismenuOpen)

  };

  const [wasOpen, setWasOpen] = useState(false);

  const toggleNaviUser = () => {

    setWasOpen(!wasOpen);

  };

  useEffect(() => {

    const handleResize = () => {

      setIsLGScreen(window.innerWidth >= 1024);

    };

    handleResize(); // Check initial size

    window.addEventListener('resize', handleResize); // Listen for resize events

    return () => {

      window.removeEventListener('resize', handleResize); // Cleanup on component unmount

    };

  }, []);

  return (

    <>

      <header className='bg-gradient-to-r from-[#012f66] to-[#0568a1] sticky top-0 z-[20] items-center justify-between p-8 flex-wrap mx-auto flex w-full'>

        <div className=''>

          <img className="m-0 md:object-none hidden sm:block " src='/Images/drdo\_logo.png' alt='logo' />

        </div>

        <nav className='flex w-auto justify-end'>

          <div></div>

          <div className='hidden lg:flex w-full justify-end'>

            <NavLinks  isOpen={isOpen}  toggleNavUser={toggleNavUser} isLGScreen={isLGScreen} />

          </div>

          <div className='lg:hidden'>

            <button onClick={toggleNaviUser}>

              {wasOpen ? <X /> : <Menu />}

            </button>

          </div>

        </nav>

        {wasOpen && (

          <div className='flex flex-col  lg:hidden basis-full mt-4'>

            <NavLinks  isOpen={isOpen} toggleNavUser={toggleNavUser} isLGScreen={isLGScreen}/>

          </div>

        )}

      </header>

       <Submenu isOpen={isOpen}  />

    </>

  );

};

export default Nav;

WorkerRole

Frontend:

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

import { Link, useParams } from 'react-router-dom';

import Nav from '../Components/Nav';

import Axios from 'axios';

const WorkerRole = () => {

    const params = useParams();

    const worker\_role = params.data;

    const [worker, setWorker] = useState([]);

    const [isLoading, setIsLoading] = useState(true); // Add isLoading state

    useEffect(() => {

        const workersData = async () => {

            try {

                const response = await Axios.get(

                    `http://localhost:3002/workers\_data/${worker\_role}`

                );

                setWorker(response.data);

                setIsLoading(false); // Set isLoading to false once data is fetched

            } catch (error) {

                console.log('Error fetching worker data:', error);

                setIsLoading(false); // Set isLoading to false even if there is an error

            }

        };

        workersData();

    }, [worker\_role]);

    return (

        <div>

            <Nav />

            <div className="mb-10 mt-10 text-center">

                <h1 className="block text-gray-700 font-bold text-2xl">Workers Profile</h1>

            </div>

            {!isLoading ? (

                <div>

                    {worker.length > 0 ? (

                        <div className='mx-auto grid grid-cols-1 lg:grid-cols-2 my-8 gap-10'>

                            {worker.map((workerData) => (

                                <div key={workerData.worker\_name} className='container mb-4 mx-auto max-w-sm '>

                                    <div className='container mt-12 bg-[#f5f5f5] shadow-2xl rounded px-8 pt-6 text-center pb-8  '>

                                        <div className='mb-2 '>

                                            <img className='w-20 border-4 rounded-full m-auto ' src='/Images/user-icon.jpg' alt='user' />

                                        </div>

                                        <div className='divide-y-2'>

                                            <div className='mb-5 text-center '>

                                                <h1 className='block text-gray-700 font-bold text-2xl'>{workerData.worker\_name}</h1>

                                            </div>

                                            <div className=''>

                                                <div className='flex justify-between mb-2 mt-5'>

                                                    <span className='text-sm font-semibold'>Work Title</span>

                                                    <span className='text-sm font-semibold'>{workerData.worker\_role}</span>

                                                </div>

                                                <div className='flex justify-between mb-2'>

                                                    <span className='text-sm font-semibold'>Complaints Assigned</span>

                                                    <span className='text-sm font-semibold'>{workerData.complaint\_assigned}</span>

                                                </div>

                                                <div className='flex justify-between mb-5'>

                                                    <span className='text-sm font-semibold'>Complaints Completed</span>

                                                    <span className='text-sm font-semibold'>{workerData.complaint\_completed}</span>

                                                </div>

                                                <div className='text-center mt-10'>

                                                   <Link to={`/worker\_role/${worker\_role}/${workerData.worker\_name}`}> <button

                                                        className='bg-blue-500 py-2 px-4 border-2 rounded-md shadow-2xl text-white  transition ease-in-out delay-150  hover:scale-110 hover:bg-green-500 duration-300 ...' >

                                                        More Details

                                                    </button> </Link>

                                                </div>

                                            </div>

                                        </div>

                                    </div>

                                </div>

                            ))}

                        </div>

                    ) : (

                        <div className="mx-auto w-1/2 mt-12">

                            <img src="/images/nodata1.avif" alt="" />

                        </div>

                    )}

                </div>

            ) : (

                <div className="text-center">Loading...</div>

            )}

        </div>

    );

};

export default WorkerRole;

Backend:

app.get('/workers\_data/:worker\_role', (req, res) => {

    const worker\_role = req.params.worker\_role

    db.query("SELECT \* FROM workers WHERE worker\_role=?", [worker\_role], (err, data) => {

        if (err) {

            console.log(err)

            return res.json("Error")

        }

        else {

            return res.send(data)

        };

    })

})

WorkerMain

Frontend:

import React from 'react'

import Nav from '../Components/Nav'

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

const WorkerMain = () => {

  const data ={

    "Electrical" : "Electrical",

    "Civil" : "Civil",

    "Option3" : "Option3",

    "Option4" : "Option4"

  }

  return (

    <>

      <Nav />

      <div className='mt-20'>

      <div className="mb-10 mt-10 text-center">

          <h1 className="block text-gray-700 font-bold text-2xl">Select Workers Role</h1>

        </div>

      <div className='mx-auto grid grid-cols-1 lg:grid-cols-2 my-8 gap-10'>

        {Object.keys(data).map((key) => (

       <div key={key} className="container mb-4 mx-auto  max-w-xs ">

            <Link to={`/worker\_role/${data[key]}`}>

   <div className="container mt-12 bg-[#f5f5f5] shadow-2xl h-60 rounded px-8 pt-6 text-center pb-8 transition ease-in-out delay-150  hover:scale-110 duration-300 flex items-center justify-center">

              <h1 className='text-2xl'>{data[key]}</h1>

            </div> </Link>

          </div>

        ))}

      </div>

      </div>

    </>

  )

}

export default WorkerMain

Single Worker

Frontend:

import React from 'react';

import { useParams, Link } from 'react-router-dom';

import Nav from '../Components/Nav';

import { Chart as ChartJS, ArcElement, Tooltip, Legend, CategoryScale, LinearScale, BarElement } from 'chart.js';

import { Bar } from 'react-chartjs-2';

import { useState, useEffect } from 'react';

import Axios from 'axios';

import Spinner from '../Components/Spinner';

ChartJS.register(ArcElement, Tooltip, Legend, CategoryScale, LinearScale, BarElement);

const labels = ["5", "4", "3", "2", "1"];

const chartOptions = {

  indexAxis: 'y', // Set the indexAxis option to 'y' for horizontal bar chart

  scales: {

    y: {

      grid: {

        display: false, // Remove background lines (grid) on the y-axis

      },

    },

    x: {

      grid: {

        display: false, // Remove background lines (grid) on the x-axis

      },

    },

  },

};

const SingleWorker = () => {

  const params = useParams();

  const worker\_name = params.worker\_name;

  const [workerjoin, setWorkerjoin] = useState([]);

  const [singledata, setSingleData] = useState([]);

  const [feedbackdata, setFeedbackData] = useState([]);

  const [loading, setLoading] = useState(true);

  useEffect(() => {

    const singleworkerdata = async () => {

      try {

        const [workerjoinres, workerres, feedbackres] = await Promise.all([

          Axios.get(`http://localhost:3002/singleworkerdatajoin/${worker\_name}`),

          Axios.get(`http://localhost:3002/singleworkerdata/${worker\_name}`),

          Axios.get(`http://localhost:3002/singleworkerfeedback/${worker\_name}`),

        ]);

        setWorkerjoin(workerjoinres.data);

        setSingleData(workerres.data);

        setFeedbackData(feedbackres.data);

        setLoading(false);

      } catch (error) {

        console.log('Error fetching worker data:', error);

        setLoading(false);

      }

    };

    singleworkerdata();

  }, [worker\_name]);

  const feedbackCounts = feedbackdata.reduce((acc, feedback) => {

    const { rate } = feedback;

    if (acc[rate]) {

      acc[rate] += 1;

    } else {

      acc[rate] = 1;

    }

    return acc;

  }, {});

  // Map feedback counts into data array

  const feedbackDataArray = labels.map((label) => feedbackCounts[label] || 0);

  const horizontalData = {

    labels: labels,

    datasets: [{

      axis: 'y',

      label: 'No of Feedbacks',

      data: feedbackDataArray,

      backgroundColor: [

        '#ffac00',

        '#ffac00',

        '#ffac00',

        '#ffac00',

        '#ffac00',

      ],

      borderWidth: 0,

      borderRadius: 10,

      barThickness: 10,

    }],

  };

  // Calculate feedback counts for each rating

  // Calculate average rate

  let totalFeedbacks = 0;

  let totalRateSum = 0;

  Object.entries(feedbackCounts).forEach(([rating, count]) => {

    totalFeedbacks += count;

    totalRateSum += count \* parseInt(rating);

  });

  const averageRate = totalFeedbacks !== 0 ? totalRateSum / totalFeedbacks : 0;

  return (

    <>

      <Nav />

      {loading ? ( // Conditionally render the Spinner when loading is true

        <div className="flex justify-center items-center h-screen">

          <Spinner />

        </div>

      ) : (

        <div className="flex flex-col lg:flex-row divide-x-2">

          {singledata.map((singleWorker) => (

            <div className="w-full lg:w-1/3" key={singleWorker.worker\_name}>

              <div className="mb-2 mt-10">

                <img className="w-1/3 border-4 rounded-full m-auto" src="/Images/user-icon.jpg" alt="user" />

              </div>

              <div className="mb-5 text-center">

                <h1 className="block text-gray-700 font-bold text-4xl">{worker\_name}</h1>

              </div>

              <div className="divide-y-2">

                <div className="mx-5 mt-10 mb-10">

                  <div className="flex justify-between mb-2">

                    <span className="text-lg font-semibold">Complaints Assigned</span>

                    <span className="text-lg font-semibold">{singleWorker.complaint\_assigned}</span>

                  </div>

                  <div className="flex justify-between mb-5">

                    <span className="text-lg font-semibold">Complaints Completed</span>

                    <span className="text-lg font-semibold">{singleWorker.complaint\_completed}</span>

                  </div>

                </div>

                <div className="mt-10">

                  <div className="mb-5 mt-10 text-center">

                    <h1 className="block text-gray-700 font-bold text-2xl">Feedbacks Rating</h1>

                  </div>

                  <div className="flex flex-col items-center">

                    <div className="ml-2 text-gray-700 font-bold text-4xl mb-5">{averageRate.toFixed(2)}</div>

                    <div className="flex">

                      {Array.from({ length: 5 }, (\_, index) => (

                        <svg

                          key={index}

                          xmlns="http://www.w3.org/2000/svg"

                          fill={index + 1 <= averageRate ? '#ffac00' : '#B7B8BC'}

                          className="h-5 w-5"

                          viewBox="0 0 24 24"

                          strokeLinecap="round"

                          strokeLinejoin="round"

                          strokeWidth="2"

                          stroke=""

                        >

                          <path d="M12 2l3.09 6.09L22 9.29l-4.55 4.41L18 22l-6-5-6 5 1.55-8.3L2 9.3l6.91-1.21L12 2z" />

                        </svg>

                      ))}

                    </div>

                    <div className=" text-gray-700 font-bold text-xs mb-5">{`${totalFeedbacks}  Feedbacks`}</div>

                  </div>

                  <div className="mx-auto mt-10 mb-10" style={{ width: '80%', height: '200px' }}>

                    <Bar data={horizontalData} options={chartOptions} />

                  </div>

                </div>

              </div>

            </div>

          ))}

          <div className="w-full lg:w-2/3">

            <div className="mb-10 mt-10 text-center">

              <h1 className="block text-gray-700 font-bold mx-auto text-2xl">Ayush Work Status </h1>

            </div>

            <div className="overflow-x-auto px-4">

              <div className="inline-block min-w-full overflow-x-auto">

                <div className="overflow-hidden">

                  <div className="w-full max-w-screen-lg mx-auto">

                    <table className="w-full text-left text-sm font-light">

                      <thead className="border-b font-medium dark:border-neutral-500">

                        <tr className="text-center">

                          <th><button className="px-4 py-4">Complaint Id</button></th>

                          <th><button className="px-6 py-4">Assigned By</button></th>

                          <th><button className="px-6 py-4">Assigned Date</button></th>

                          <th><button className="px-6 py-4">Status</button></th>

                        </tr>

                      </thead>

                      <tbody>

                        {workerjoin.length === 0 ? (

                          <tr className="border-b dark:border-neutral-500 text-center">

                            <td colSpan="4">

                              <img className="w-1/2 mx-auto" src="/Images/nodata1.avif" alt="No Data" />

                              <p className="text-gray-500 mt-3">No data available</p>

                            </td>

                          </tr>

                        ) : (

                          workerjoin.map((workerData) => (

                            <tr className="border-b dark:border-neutral-500 text-center" key={workerData.complaint\_id}>

                              <td className="whitespace-nowrap px-6 py-4">{workerData.complaint\_id}</td>

                              <td className="whitespace-nowrap px-6 py-4">{workerData.username}</td>

                              <td className="whitespace-nowrap px-6 py-4">{workerData.assigned\_date.substring(0, 10)}</td>

                              <td className="whitespace-nowrap px-6 py-4">{workerData.state}</td>

                              <td className="whitespace-nowrap px-6 py-4"> <Link to={`/Status/${workerData.complaint\_id}`}><button className="bg-blue-500 py-2 px-3 border-2 rounded-md shadow-2xl text-white  transition ease-in-out delay-150  hover:scale-110 hover:bg-green-500 duration-300 ...">Check</button></Link></td>

                            </tr>

                          ))

                        )}

                      </tbody>

                    </table>

                  </div>

                </div>

              </div>

            </div>

          </div>

        </div>

      )}

    </>

  );

};

export default SingleWorker;

Backend:

app.get('/singleworkerdatajoin/:worker\_name', (req, res) => {

    const worker\_name = req.params.worker\_name

    db.query("SELECT complaints.complaint, complaints.state, complaints.username,assigned.complaint\_id, assigned.assigned\_date FROM complaints INNER JOIN assigned ON complaints.complaint\_id=assigned.complaint\_id WHERE assigned\_to=?", [worker\_name], (err, data) => {

        if (err) {

            console.log(err)

            return res.json("Error")

        }

        else {

            console.log(data)

            return res.send(data)

        };

    })

})

app.get('/singleworkerdata/:worker\_name', (req, res) => {

    const worker\_name = req.params.worker\_name

    db.query("SELECT \* FROM workers WHERE worker\_name=?", [worker\_name], (err, data) => {

        if (err) {

            console.log(err)

            return res.json("Error")

        }

        else {

            return res.send(data)

        };

    })

})

app.get('/singleworkerfeedback/:worker\_name', (req, res) => {

    const worker\_name = req.params.worker\_name

    db.query("SELECT feedback.rate, feedback.feedback\_date, feedback.feedbacks,assigned.complaint\_id  FROM feedback INNER JOIN assigned ON feedback.complaint\_id=assigned.complaint\_id WHERE assigned\_to=?", [worker\_name], (err, data) => {

        if (err) {

            console.log(err)

            return res.json("Error")

        }

        else {

            console.log(data)

            return res.send(data)

        };

    })

})

App.js

import React from 'react';

import { Route, Routes } from 'react-router-dom';

import Login from './Login/Login';

import Signup from './Signup/Signup';

import Main from './Main/Main';

import Status from './Status/Status';

import Summary from './Summary/Summary';

import Profile from './Profile/Profile';

import Feedbacks from './Feedbacks/Feedbacks';

import ErrorPage from './Error/ErrorPage';

import Unauthorized from './Components/Unauthorized'

import RequireAuth from './Components/RequireAuth';

import WorkerMain from './Workers/WorkerMain';

import WorkerRole from './Workers/WorkerRole';

import SingleWorker from './Workers/SingleWorker';

const App = () => {

  const ROLES = {

    'User':"General",

    'Admin':"ADMIN"

  }

  return (

    <>

      <Routes>

        <Route path="/" element={<Login />} />

        <Route path="/Signup" element={<Signup />} />

        <Route path="/Unauthorized" element={<Unauthorized />} />

        <Route element={<RequireAuth allowedRoles={[ROLES.User,ROLES.Admin]}/>}>

        <Route path="/main" element={<Main />} />

        <Route path="/status/:complaint\_id" element={<Status />} />

        <Route path="/summary" element={<Summary />} />

        <Route path="/profile" element={<Profile />} />

        </Route>

        <Route element={<RequireAuth allowedRoles={[ROLES.Admin]}/>}>

        <Route path="/feedbacks" element={<Feedbacks />} />

        <Route path="/workers" element={<WorkerMain />} />

        <Route path="/worker\_role/:data" element={<WorkerRole />} />

        <Route path="/worker\_role/:data/:worker\_name" element={<SingleWorker />} />

        </Route>

        <Route path="\*" element={<ErrorPage />} />

      </Routes>

    </>

  );

};

export default App;

Data Base Connectivity

const mysql = require('mysql')

const db = mysql.createConnection({

host: "localhost",

user: "root",

password: "",

database:"works\_dept"

})

module.exports = db;

