ResolveNow:

→Platform For Online

**Complaints.**

**Student name:** SABHAVATHULA NEERAJ NAYAK

**Roll No:** 23481A04J2

**Department:** Electronics and communication engineering

**Team ID:** LTVIP2025TMID55000

**PROJECT OBJECTIVE:**

1. **Provide a User-Friendly Complaints Submission Interface.**
2. **Ensure Efficient Complaints Tracking and Management.**
3. **Enable Automated Complaint Routing.**
4. **Facilitate Transparent Communication.**
5. **Ensure Data Privacy and Security.**
6. **Provide Admin Dashboard For Complaint Monitoring.**
7. **Support Feedback and Satisfaction Rating.**
8. **Enable Multi-Language Support (if needed).**
9. **Maintain a Complaint History Log.**

**10.Generate Reports for Governance and Audit.**

**PROJECT REPORT:**

INTRODUCTION:

An online complaint registration and management system is a software application or platform that allows individuals or organizations to submit and track complaints or issues they have encountered. It can help optimize the complaint handling process and empower organizations to develop a safety management system to efficiently resolve customer complaints, while staying in line with industry guidelines and regulatory compliance obligations. It provides a centralized platform for managing complaints, streamlining the complaint resolution process, and improving customer satisfaction.

KEY FEATURES:

1. User registration: Users can create accounts to submit complaints and track their progress.
2. Complaint submission: Users can enter details of their complaints, including relevant information such name, description of the issue, address etc.
3. Tracking and notifications: Users can track the progress of their complaints, view updates, and receive notifications via email or SMS when there are any changes or resolutions.
4. User can interact with the agent who has assigned the complaint.
5. Assigning and routing complaints: The system assigns complaints to the appropriate department or personnel responsible for handling them. It may use intelligent routing algorithms to ensure efficient allocation of resources.

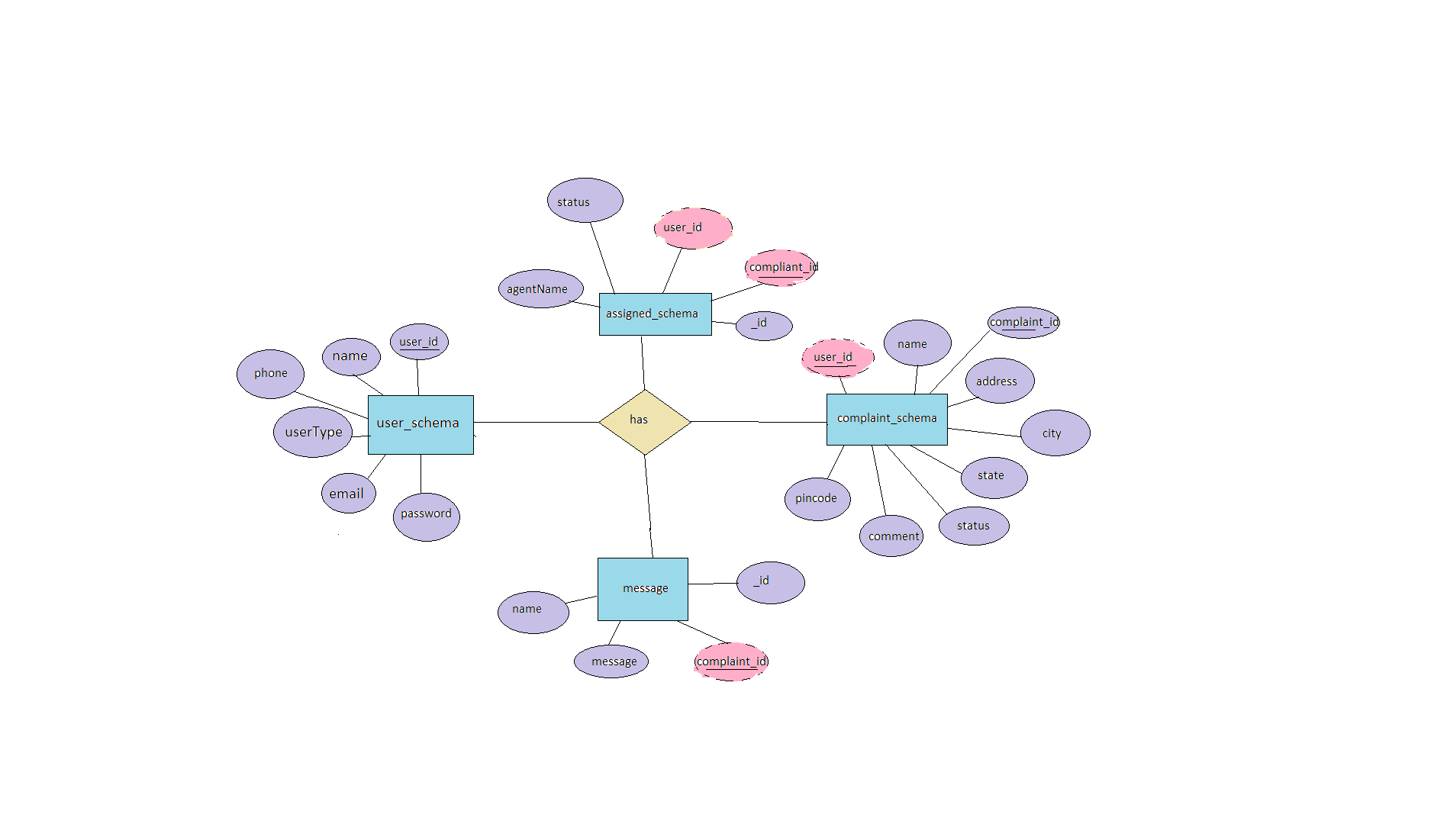
Security and confidentiality: The system ensures the security and confidentiality of user data and complaint information through measures such as user authentication, data encryption, access controls, and compliance with relevant data protection regulations.

**Technical Architecture:**



* The technical architecture of our online complaint registration and management app follows a client-server model, where the frontend serves as the client and the backend acts as the server. The frontend encompasses not only the user interface and presentation but also incorporates the axios library to connect with backend easily by using RESTful Apis.
* The frontend utilizes the bootstrap and material UI library to establish real-time and better UI experience for any user whether it is agent, admin or ordinary user working on it.
* On the backend side, we employ Express.js frameworks to handle the server-side logic and communication.
* For data storage and retrieval, our backend relies on MongoDB. MongoDB allows for efficient and scalable storage of user data, including user profiles, for complaints registration, etc. It ensures reliable and quick access to the necessary information during registration of user or any complaints.

**ER DIAGRAM:**



* This is the er diagram of the project which shows the relationship between user and agent
* It shows how user which have required fields can raise a complaint by fillings required fields.
* It illustrates how these entities relate to each other, helping us understand the underlying database structure and the flow of information within the app. He / She can also communicate with the agent with chat window which follows the message schema which uses userId and complaintId from other schemas.

**PROJECT STRUCTURE:**

* **FRONEND:**



* **BACKEND:**



**PRE-REQUISITES:**

Here are the key prerequisites for developing a full-stack application using Node.js, Express.js, MongoDB, React.js:

✔**Node.js and npm:**

Node.js is a powerful JavaScript runtime environment that allows you to run JavaScript code on the server-side. It provides a scalable and efficient platform for building network applications.

Install Node.js and npm on your development machine, as they are required to run JavaScript on the server-side.

✔**Express.js:**

Express.js is a fast and minimalist web application framework for Node.js. It simplifies the process of creating robust APIs and web applications, offering features like routing, middleware support, and modular architecture.

Install Express.js, a web application framework for Node.js, which handles server-side routing, middleware, and API development.

Installation: Open your command prompt or terminal and run the following command:

✔**MongoDB:**

MongoDB is a flexible and scalable NoSQL database that stores data in a JSON-like format. It provides high performance, horizontal scalability, and seamless integration with Node.js, making it ideal for handling large amounts of structured and unstructured data.

Set up a MongoDB database to store your application's data.

✔**React.js:**

React.js is a popular JavaScript library for building user interfaces. It enables developers to create interactive and reusable UI components, making it easier to build dynamic and responsive web applications.

Install React.js, a JavaScript library for building user interfaces.

✔**HTML, CSS, and JavaScript**: Basic knowledge of HTML for creating the structure of your app, CSS for styling, and JavaScript for client-side interactivity is essential.

✔**Database Connectivity**: Use a MongoDB driver or an Object-Document Mapping (ODM) library like Mongoose to connect your Node.js server with the MongoDB database and perform CRUD (Create, Read, Update, Delete) operations. To Connect the Database with Node JS go through the below provided link:

✔**Front-end Framework**: Utilize Reactjs to build the user-facing part of the application, including entering complaints, status of the complaints, and user interfaces for the admin dashboard.

For making better UI we have also used some libraries like material UI and boostrap.

✔**Version Control**: Use Git for version control, enabling collaboration and tracking changes throughout the development process. Platforms like GitHub or Bitbucket can host your repository.

✔**Development Environment**: Choose a code editor or Integrated Development Environment (IDE) that suits your preferences, such as Visual Studio Code, Sublime Text, or WebStorm.

**Roles and Responsibilities:**

The project has two types of users – Agents and Customer and third is Admin which takes care to all the user whether it is Agent or simple user. The roles and responsibilities of these two types of users can be inferred from the API endpoints defined in the code. Here is a summary:

Customer/Ordinary:

1. Create an account and log in to the system using their email and password.
2. Browse and fill the form of your complaint or any issues for the agent to solve.
3. After filling your complaint, he/she can view the status of complaint in the status section.
4. He/She can connect to the agent directly by sending message and talk more about the complaints by using chat window.
5. Manage their profile information, including personal details and shipping addresses.

Agent:

1. Create an account and log in to the system using their email and password.
2. Manage all the complaints assigned by the Admin.
3. He/She can connect directly to the user of the complaint by sending the message through chat window.
4. If complaints are resolved, he can change the status by clicking the button a
5. Interact with customers by responding to inquiries, resolving issues, and addressing feedback.

Admin:

1. Manage and monitor the overall operation of the complaint registering platform.
2. Monitor and moderate all the complaints that are coming from the user
3. Easily assigned the complaints to the desired agent.
4. Manage user as well as agents accounts.
5. Implement and enforce platform policies, terms of service, and privacy regulations.
6. Continuously improve the platform's functionality, user experience, and security measures.

**PROJECT FLOW:**

**Milestone 1: Project setup and configuration.**

* **Folder setup:**

1. Create frontend and
2. Backend folders

* **Installation of required tools:**

1. Open the frontend folder to install necessary tools

For frontend, we use:

* + - React JS
    - Bootstrap
    - Material UI
    - Axios

2. Open the backend folder to install necessary tools

For backend, we use:

* + - cors
    - express
    - mongoose

**Milestone 2: Backend Development**

* **Setup express server**

1. Create index.js file in the server (backend folder).
2. define port number to access it.
3. Configure the server by adding cors, body-parser.

* **Configure MongoDB**

1. Import mongoose.
2. Add Database URL to the config.js file.
3. Connect the database to the server.
4. Create a ‘schema’ file in the server to store all the DB models.

* **Add authentication**

1. Create the “User Schema” model for the MongoDB.
2. Define registration & login activities.
3. Using Axios library, make request from the frontend and vice-versa.

**Milestone 3: Web Applications development**

* **Add complaints and the status of the complaints**

1. Create the “complaint\_schema” model for the MongoDB to register complaint
2. Defined well RESTful API for better fetching the records

* **Sending message to agent through chat window**

1. Create the “message” model for the MongoDB to sending more description about the complaint easily.
2. It can keep records by using various attributes present in rest of the collections.
3. The chat window can adjust himself with the latest message arrived at the chat window.

* **CRUD operation are done by admin**

1. Admin has the right to perform deletion, updating or add user in the database.
2. He can assign the issues or complaints by himself to the right agents present in collection.

**PROJECT CODE:**

* **ADMINE:**
* AccordionAdmin.jsx

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

**import Accordion from 'react-bootstrap/Accordion';**

**import Card from 'react-bootstrap/Card';**

**import Dropdown from 'react-bootstrap/Dropdown';**

**import Alert from 'react-bootstrap/Alert';**

**import Footer from '../common/FooterC'**

**import axios from 'axios';**

**const AccordionAdmin = () => {**

**const [complaintList, setComplaintList] = useState([]);**

**const [agentList, setAgentList] = useState([]);**

**useEffect(() => {**

**const getComplaints = async () => {**

**try {**

**const response = await axios.get('http://localhost:8000/status');**

**const complaints = response.data;**

**setComplaintList(complaints);**

**} catch (error) {**

**console.log(error);**

**}**

**};**

**getComplaints();**

**const getAgentsRecords = async () => {**

**try {**

**const response = await axios.get('http://localhost:8000/AgentUsers');**

**const agents = response.data;**

**setAgentList(agents);**

**} catch (error) {**

**console.log(error);**

**}**

**};**

**getAgentsRecords();**

**}, []);**

**const handleSelection = async (agentId, complaintId, status, agentName) => {**

**try {**

**await axios.get(`http://localhost:8000/AgentUsers/${agentId}`);**

**const assignedComplaint = {**

**agentId,**

**complaintId,**

**status,**

**agentName,**

**};**

**await axios.post('http://localhost:8000/assignedComplaints', assignedComplaint);**

**const updatedComplaintList = complaintList.filter((complaint) => complaint.id !== complaintId);**

**setComplaintList(updatedComplaintList);**

**alert(`Compliant assigned to the Agent ${agentName}`)**

**} catch (error) {**

**console.log(error);**

**}**

**};**

**return (**

**<div>**

**<Accordion className='accordian' alwaysOpen>**

**<Accordion.Item eventKey="0">**

**<Accordion.Header>Users Complaints</Accordion.Header>**

**<Accordion.Body style={{background:'aliceblue'}}>**

**<div style={{ display: "flex", flexWrap: "wrap", margin: "20px" }}>**

**{complaintList.length > 0 ? (**

**complaintList.map((complaint, index) => (**

**<Card key={index} style={{ width: '15rem', margin: '0 10px 15px 0' }}>**

**<Card.Body style={{textAlign:'center'}}>**

**<Card.Title>Name: {complaint.name}</Card.Title>**

**<div style={{fontSize:'14px',marginTop: '20px'}}>**

**<Card.Text>Address: {complaint.address}</Card.Text>**

**<Card.Text>City: {complaint.city}</Card.Text>**

**<Card.Text>State: {complaint.state}</Card.Text>**

**<Card.Text>Pincode: {complaint.pincode}</Card.Text>**

**<Card.Text>Comment: {complaint.comment}</Card.Text>**

**<Card.Text>Status: {complaint.status}</Card.Text>**

**</div>**

**{(complaint.status === "completed") ?**

**<></>**

**: <Dropdown className='mt-2'>**

**<Dropdown.Toggle variant="warning" id="dropdown-basic">**

**Assign**

**</Dropdown.Toggle>**

**<Dropdown.Menu>**

**{**

**agentList.map((agent, index) => {**

**return (**

**<Dropdown.Item key={index} onClick={() => handleSelection(agent.\_id, complaint.\_id, complaint.status, agent.name)}>{agent.name}</Dropdown.Item>**

**)**

**})**

**}**

**</Dropdown.Menu>**

**</Dropdown>**

**}**

**</Card.Body>**

**</Card>**

**))**

**) : (**

**<Alert variant="info">**

**<Alert.Heading>No complaints to show</Alert.Heading>**

**</Alert>**

**)}**

**</div>**

**</Accordion.Body>**

**</Accordion.Item>**

**<Accordion.Item eventKey="1">**

**<Accordion.Header>Agents</Accordion.Header>**

**<Accordion.Body style={{background:'aliceblue'}}>**

**<div style={{ display: "flex", flexWrap: "wrap", margin: "20px" }}>**

**{agentList.length > 0 ? (**

**agentList.map((agent, index) => (**

**<Card key={index} style={{ width: '22rem', margin: '0 10px 15px 0' }}>**

**<Card.Body>**

**<Card.Title>Name: {agent.name}</Card.Title>**

**<Card.Text>Email: {agent.email}</Card.Text>**

**</Card.Body>**

**</Card>**

**))**

**) : (**

**<Alert variant="info">**

**<Alert.Heading>No Agents to show</Alert.Heading>**

**</Alert>**

**)}**

**</div>**

**</Accordion.Body>**

**</Accordion.Item>**

**</Accordion>**

**<Footer/>**

**</div >**

**)**

**}**

**export default AccordionAdmin**

* AdminHome.jsx

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

import Button from 'react-bootstrap/Button';

import Container from 'react-bootstrap/Container';

import Nav from 'react-bootstrap/Nav';

import Navbar from 'react-bootstrap/Navbar';

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

import UserInfo from './UserInfo';

import AccordionAdmin from "./AccordionAdmin";

import AgentInfo from './AgentInfo';

const AdminHome = () => {

const navigate = useNavigate();

const [activeComponent, setActiveComponent] = useState('dashboard');

const [userName, setUserName] = useState('');

useEffect(() => {

const getData = async () => {

try {

const user = JSON.parse(localStorage.getItem('user'));

if (user) {

const { name } = user;

setUserName(name);

} else {

navigate('/');

}

} catch (error) {

console.log(error);

}

};

getData();

}, [navigate]);

const handleNavLinkClick = (componentName) => {

setActiveComponent(componentName);

};

const LogOut = () => {

localStorage.removeItem('user');

navigate('/');

};

return (

<>

<Navbar className="text-white" bg="dark" expand="lg">

<Container fluid>

<Navbar.Brand className="text-white" href="#">

Hi Admin {userName}

</Navbar.Brand>

<Navbar.Toggle aria-controls="navbarScroll" />

<Navbar.Collapse id="navbarScroll">

<Nav className="text-white me-auto my-2 my-lg-0" style={{ maxHeight: '100px' }} navbarScroll>

<NavLink

className={`nav-link text-light ${activeComponent === 'dashboard' ? 'active' : ''}`}

onClick={() => handleNavLinkClick('dashboard')}

>

Dashboard

</NavLink>

<NavLink

className={`nav-link text-light ${activeComponent === 'UserInfo' ? 'active' : ''}`}

onClick={() => handleNavLinkClick('UserInfo')}

>

User

</NavLink>

<NavLink

className={`nav-link text-light ${activeComponent === 'Agent' ? 'active' : ''}`}

onClick={() => handleNavLinkClick('Agent')}

>

Agent

</NavLink>

</Nav>

<Button onClick={LogOut} variant="outline-danger">

Log out

</Button>

</Navbar.Collapse>

</Container>

</Navbar>

<div className="content">

{activeComponent === 'Agent' ? <AgentInfo /> : null}

{activeComponent === 'dashboard' ? <AccordionAdmin /> : null}

{activeComponent === 'UserInfo' ? <UserInfo /> : null}

</div>

</>

)

};

export default AdminHome;

* Agentinfo.jsx

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

import Button from 'react-bootstrap/Button';

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

import Table from 'react-bootstrap/Table';

import Alert from 'react-bootstrap/Alert';

import { Container } from 'react-bootstrap';

import Collapse from 'react-bootstrap/Collapse';

import Form from 'react-bootstrap/Form';

import Footer from '../common/FooterC'

import axios from 'axios';

const AgentInfo = () => {

const navigate = useNavigate();

const [ordinaryList, setOrdinaryList] = useState([]);

const [toggle, setToggle] = useState({})

const [updateAgent, setUpdateAgent] = useState({

name: '',

email: '',

phone: '',

})

const handleChange = (e) => {

setUpdateAgent({ ...updateAgent, [e.target.name]: e.target.value })

}

const handleSubmit = async (user\_id) => {

if (updateAgent === "") {

alert("atleast 1 fields need to be fill")

}

else {

window.confirm("Are you sure you want to update the agent?");

axios.put(`http://localhost:8000/user/${user\_id}`, updateAgent)

.then((res) => {

alert(`Agent updated successfully`)

JSON.stringify(res.data)

})

.catch((err) => {

console.log(err)

})

}

}

useEffect(() => {

const getOrdinaryRecords = async () => {

try {

const response = await axios.get('http://localhost:8000/agentUsers');

const ordinary = response.data;

setOrdinaryList(ordinary)

} catch (error) {

console.log(error);

}

};

getOrdinaryRecords();

}, [navigate]);

const deleteUser = async (userId) => {

try {

const confirmed = window.confirm("Are you sure you want to delete the user?");

if (confirmed) {

await axios.delete(`http://localhost:8000/OrdinaryUsers/${userId}`);

setOrdinaryList(ordinaryList.filter((user) => user.\_id !== userId));

}

} catch (error) {

console.log(error);

}

}

const handleToggle = (complaintId) => {

setToggle((prevState) => ({

...prevState,

[complaintId]: !prevState[complaintId],

}));

};

return (

<>

<div className="body">

<Container>

<Table striped bordered hover>

<thead>

<tr>

<th>Name</th>

<th>Email</th>

<th>Phone</th>

<th>Action</th>

</tr>

</thead>

<tbody>

{ordinaryList.length > 0 ? (

ordinaryList.map((agent) => {

const open = toggle[agent.\_id] || false;

return (

<tr key={agent.\_id}>

<td>{agent.name}</td>

<td>{agent.email}</td>

<td>{agent.phone}</td>

<td><Button onClick={() => handleToggle(agent.\_id)}

aria-controls={`collapse-${agent.\_id}`}

aria-expanded={open}

className='mx-2'

variant="outline-warning">

Update

</Button>

<Collapse in={open}>

<Form onSubmit={() => handleSubmit(agent.\_id)} className='p-5'>

<Form.Group className="mb-3" controlId="formBasic">

<Form.Label>Full Name </Form.Label>

<Form.Control type="text" name='name' value={updateAgent.name} onChange={handleChange} placeholder="Enter name" />

</Form.Group>

<Form.Group className="mb-3" value controlId="formBasicEmail">

<Form.Label>Email address</Form.Label>

<Form.Control type="email" name='email' value={updateAgent.email} onChange={handleChange} placeholder="Enter email" />

</Form.Group>

<Form.Group className="mb-3" value controlId="formBasicTel">

<Form.Label>Phone</Form.Label>

<Form.Control type="tel" name='phone' value={updateAgent.phone} onChange={handleChange} placeholder="Enter Phone no." />

</Form.Group>

<Button size='sm' variant="outline-success" type="submit">

Submit

</Button>

</Form>

</Collapse>

<Button onClick={() => deleteUser(agent.\_id)} className='mx-2' variant="outline-danger">Delete</Button></td>

</tr>

)

})

) : (

<Alert variant="info">

<Alert.Heading>No Agents to show</Alert.Heading>

</Alert>

)}

</tbody>

</Table>

</Container>

</div>

<Footer />

</>

)

}

export default AgentInfo

* Userinfo.jsx

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

import Button from 'react-bootstrap/Button';

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

import Table from 'react-bootstrap/Table';

import Alert from 'react-bootstrap/Alert';

import { Container } from 'react-bootstrap';

import Collapse from 'react-bootstrap/Collapse';

import Form from 'react-bootstrap/Form';

import Footer from '../common/FooterC'

import axios from 'axios';

const UserInfo = () => {

const navigate = useNavigate();

const [ordinaryList, setOrdinaryList] = useState([]);

const [toggle, setToggle] = useState({})

// const [count, setCount] = useState(0)

const [updateUser, setUpdateUser] = useState({

name: '',

email: '',

phone: '',

})

const handleChange = (e) => {

setUpdateUser({ ...updateUser, [e.target.name]: e.target.value })

}

const handleSubmit = async (user\_id) => {

if (updateUser === "") {

alert("atleast 1 fields need to be fill")

}

else {

window.confirm("Are you sure you want to Update the user?");

axios.put(`http://localhost:8000/user/${user\_id}`, updateUser)

.then((res) => {

alert(`user updated successfully`)

JSON.stringify(res.data)

})

.catch((err) => {

console.log(err)

})

}

}

useEffect(() => {

const getOrdinaryRecords = async () => {

try {

const response = await axios.get('http://localhost:8000/OrdinaryUsers');

const ordinary = response.data;

setOrdinaryList(ordinary)

} catch (error) {

console.log(error);

}

};

getOrdinaryRecords();

}, [navigate]);

const deleteUser = async (userId) => {

try {

const confirmed = window.confirm("Are you sure you want to delete the user?");

if (confirmed) {

await axios.delete(`http://localhost:8000/OrdinaryUsers/${userId}`);

setOrdinaryList(ordinaryList.filter((user) => user.\_id !== userId));

}

} catch (error) {

console.log(error);

}

}

const handleToggle = (complaintId) => {

setToggle((prevState) => ({

...prevState,

[complaintId]: !prevState[complaintId],

}));

};

return (

<>

<div className="body">

<Container>

<Table striped bordered hover>

<thead>

<tr>

<th>Name</th>

<th>Email</th>

<th>Phone</th>

<th>Action</th>

</tr>

</thead>

<tbody>

{ordinaryList.length > 0 ? (

ordinaryList.map((user) => {

const open = toggle[user.\_id] || false;

return (

<tr key={user.\_id}>

<td>{user.name}</td>

<td>{user.email}</td>

<td>{user.phone}</td>

<td><Button onClick={() => handleToggle(user.\_id)}

aria-controls={`collapse-${user.\_id}`}

aria-expanded={open}

className='mx-2'

variant="outline-warning">

Update

</Button>

<Collapse in={open}>

<Form onSubmit={() => handleSubmit(user.\_id)} className='p-5'>

<Form.Group className="mb-3" controlId="formBasic">

<Form.Label>Full Name </Form.Label>

<Form.Control name='name' value={updateUser.name} onChange={handleChange} type="text" placeholder="Enter name" />

</Form.Group>

<Form.Group className="mb-3" controlId="formBasicEmail">

<Form.Label>Email address</Form.Label>

<Form.Control name='email' value={updateUser.email} onChange={handleChange} type="email" placeholder="Enter email" />

</Form.Group>

<Form.Group className="mb-3" controlId="formBasicTel">

<Form.Label>Phone</Form.Label>

<Form.Control name='phone' value={updateUser.phone} onChange={handleChange} type="tel" placeholder="Enter Phone no." />

</Form.Group>

<Button size='sm' variant="outline-success" type="submit">

Submit

</Button>

</Form>

</Collapse>

<Button onClick={() => deleteUser(user.\_id)} className='mx-2' variant="outline-danger">Delete</Button></td>

</tr>

)

})

) : (

<Alert variant="info">

<Alert.Heading>No Users to show</Alert.Heading>

</Alert>

)}

</tbody>

</Table>

</Container>

</div>

<Footer />

</>

)

}

export default UserInfo

* AGENT:
* AgentHome.jsx

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

import Button from 'react-bootstrap/Button';

import Container from 'react-bootstrap/Container';

import Nav from 'react-bootstrap/Nav';

import Navbar from 'react-bootstrap/Navbar';

import Card from 'react-bootstrap/Card';

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

import axios from 'axios';

import Alert from 'react-bootstrap/Alert';

import Collapse from 'react-bootstrap/Collapse';

import ChatWindow from '../common/ChatWindow';

import Footer from '../common/FooterC'

const AgentHome = () => {

const style = {

marginTop: '66px',

}

const navigate = useNavigate();

const [userName, setUserName] = useState('');

const [toggle, setToggle] = useState({})

const [agentComplaintList, setAgentComplaintList] = useState([]);

useEffect(() => {

const getData = async () => {

try {

const user = JSON.parse(localStorage.getItem('user'));

if (user) {

const { \_id, name } = user;

setUserName(name);

const response = await axios.get(`http://localhost:8000/allcomplaints/${\_id}`);

const complaints = response.data;

setAgentComplaintList(complaints);

} else {

navigate('/');

}

} catch (error) {

console.log(error);

}

};

getData();

}, [navigate]);

const handleStatusChange = async (complaintId) => {

try {

await axios.put(`http://localhost:8000/complaint/${complaintId}`, { status: 'completed' });

setAgentComplaintList((prevComplaints) =>

prevComplaints.map((complaint) =>

complaint.\_doc.complaintId === complaintId ? { ...complaint, \_doc: { ...complaint.\_doc, status: 'completed' } } : complaint

)

);

} catch (error) {

console.log(error);

}

};

const handleToggle = (complaintId) => {

setToggle((prevState) => ({

...prevState,

[complaintId]: !prevState[complaintId],

}));

};

const LogOut = () => {

localStorage.removeItem('user');

navigate('/');

};

return (

<>

<div className="body">

<Navbar className="text-white" bg="dark" expand="lg">

<Container fluid>

<Navbar.Brand className="text-white">

Hi Agent {userName}

</Navbar.Brand>

<Navbar.Toggle aria-controls="navbarScroll" />

<Navbar.Collapse id="navbarScroll">

<Nav className="text-white me-auto my-2 my-lg-0" style={{ maxHeight: '100px' }} navbarScroll>

<NavLink style={{ textDecoration: 'none' }} className="text-white">

View Complaints

</NavLink>

</Nav>

<Button onClick={LogOut} variant="outline-danger">

Log out

</Button>

</Navbar.Collapse>

</Container>

</Navbar>

<div className="container" style={{ display: 'flex', flexWrap: 'wrap', margin: '20px' }}>

{agentComplaintList && agentComplaintList.length > 0 ? (

agentComplaintList.map((complaint, index) => {

const open = toggle[complaint.\_doc.complaintId] || false;

return (

<Card key={index} style={{ width: '18rem', margin: '15px' }}>

<Card.Body>

<Card.Title><b>Name:</b> {complaint.name}</Card.Title>

<Card.Text><b>Address:</b> {complaint.address}</Card.Text>

<Card.Text><b>City:</b> {complaint.city}</Card.Text>

<Card.Text><b>State:</b> {complaint.state}</Card.Text>

<Card.Text><b>Pincode:</b> {complaint.pincode}</Card.Text>

<Card.Text><b>Comment:</b> {complaint.comment}</Card.Text>

<Card.Text><b>Status:</b> {complaint.\_doc.status}</Card.Text>

{complaint.status !== 'completed' && (

<Button onClick={() => handleStatusChange(complaint.\_doc.complaintId)} variant="primary">

Status Change

</Button>

)}

<Button onClick={() => handleToggle(complaint.\_doc.complaintId)}

aria-controls={`collapse-${complaint.\_doc.complaintId}`}

aria-expanded={!open} className='mx-3' variant="primary">

Message

</Button>

<div>

<Collapse in={!open} dimension="width">

<div id="example-collapse-text">

<Card body style={{ width: '250px', marginTop: '12px' }}>

<ChatWindow key={complaint.\_doc.complaintId} complaintId={complaint.\_doc.complaintId} name={userName} />

</Card>

</div>

</Collapse>

</div>

</Card.Body>

</Card>

);

})

) : (

<Alert variant="info">

<Alert.Heading>No complaints to show</Alert.Heading>

</Alert>

)}

</div>

</div>

<Footer style={style}/>

</>

);

};

export default AgentHome;

* COMMON:
* ChatWindow.jsx

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

import axios from 'axios';

const ChatWindow = (props) => {

const [messageInput, setMessageInput] = useState('');

const messageWindowRef = useRef(null);

const [messageList, setMessageList] = useState([]);

const fetchMessageList = async () => {

try {

const response = await axios.get(`http://localhost:8000/messages/${props.complaintId}`);

setMessageList(response.data);

} catch (error) {

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

}

};

useEffect(() => {

fetchMessageList(props.complaintId, setMessageList);

}, [props.complaintId]);

useEffect(() => {

scrollToBottom();

}, [messageList]);

const sendMessage = async () => {

try {

let data = {

name: props.name,

message: messageInput,

complaintId: props.complaintId

}

const response = await axios.post('http://localhost:8000/messages', data)

setMessageList([...messageList, response.data]);

setMessageInput('');

fetchMessageList();

} catch (error) {

console.error('Error sending message:', error);

}

}

const scrollToBottom = () => {

if (messageWindowRef.current) {

messageWindowRef.current.scrollTop = messageWindowRef.current.scrollHeight;

}

};

return (

<>

<div className="chat-container">

<h1>Message Box</h1>

<div className="message-window" ref={messageWindowRef}>

{messageList.slice().reverse().map((msg) => (

<div className="message" key={msg.\_id}>

<p>{msg.name}: {msg.message}</p>

<p style={{ fontSize: '10px', marginTop: '-15px' }}>{new Date(msg.createdAt).toLocaleTimeString([], { hour: '2-digit', minute: '2-digit' })}, {new Date(msg.createdAt).toLocaleDateString()}</p>

</div>

))}

</div>

<div className="input-container">

<input required type="text" placeholder="Message" value={messageInput} onChange={(e) => setMessageInput(e.target.value)} />

<button className='btn btn-success' onClick={sendMessage}>Send</button>

</div>

</div>

</>)

}

export default ChatWindow

* FooterC.jsx

import React from 'react';

import { MDBFooter } from 'mdb-react-ui-kit';

export default function FooterC() {

return (

<MDBFooter style={{height: '112px', marginTop: '101px'}} bgColor='dark' className='text-center text-lg-left'>

<div className='text-center p-3'>

<p className='text-light'>

ComplaintCare

</p>

<p className='text-light'>&copy; {new Date().getFullYear()}</p>

</div>

</MDBFooter>

);

}

* Home.jsx

import React from 'react'

import Navbar from 'react-bootstrap/Navbar';

import Container from 'react-bootstrap/Container';

import Image1 from '../../Images/Image1.png'

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

import Button from 'react-bootstrap/Button';

import Footer from './FooterC'

const Home = () => {

return (

<>

<Navbar bg="dark" variant="dark">

<Container>

<Navbar.Brand>ComplaintCare </Navbar.Brand>

<ul className="navbar-nav">

<li className="nav-item mb-2">

<Link to={'/'}

className={`nav-link text-light `}

>

Home

</Link>

</li>

{/\* <li className="nav-item mb-2">

<Link

to={'/About'}

className={`nav-link text-light `}

>

About

</Link>

</li> \*/}

<li className="nav-item mb-2">

<Link

to={'/signup'}

className={`nav-link text-light `}

>

SignUp

</Link>

</li>

<li className="nav-item mb-2">

<Link

to={'/login'}

className={`nav-link text-light `}

>

Login

</Link>

</li>

</ul>

</Container>

</Navbar>

<Container className='home-container'>

<div className="left-side">

<img src={Image1} alt="" />

</div>

<div className="right-side">

<p>

<span className='f-letter'>Empower Your Team,</span><br />

<span className='s-letter'> Exceed Customer Expectations: Discover our</span> <br />

<span className='t-letter'>Complaint Management Solution</span><br />

<Link to={'/Login'}><Button className='mt-3 register'>Register your Compliant</Button></Link>

</p>

</div>

</Container>

<Footer/>

</>

)

}

export default Home

* Login.jsx

import axios from 'axios';

import React, { useState } from 'react';

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

import Container from 'react-bootstrap/Container';

import Navbar from 'react-bootstrap/Navbar';

import Footer from './FooterC'

const Login = () => {

const navigate = useNavigate();

const [user, setUser] = useState({

email: "",

password: ""

});

const handleChange = (e) => {

const { name, value } = e.target;

setUser({ ...user, [name]: value });

};

const handleSubmit = async (e) => {

e.preventDefault();

await axios.post("http://localhost:8000/Login", user)

.then((res) => {

alert("Successfully logged in");

localStorage.setItem("user", JSON.stringify(res.data));

const isLoggedIn = JSON.parse(localStorage.getItem("user"));

const { userType } = isLoggedIn

switch (userType) {

case "Admin":

navigate("/AdminHome")

break;

case "Ordinary":

navigate("/HomePage")

break;

case "Agent":

navigate("/AgentHome")

break;

default:

navigate("/Login")

break;

}

})

.catch((err) => {

if (err.response && err.response.status === 401) {

alert("User doesn`t exists");

}

navigate("/Login");

});

};

return (

<>

<Navbar bg="dark" variant="dark">

<Container>

<Navbar.Brand>ComplaintCare </Navbar.Brand>

<ul className="navbar-nav">

<li className="nav-item mb-2">

<Link to={'/'}

className={`nav-link text-light `}

>

Home

</Link>

</li>

{/\* <li className="nav-item mb-2">

<Link

to={'/about'}

className={`nav-link text-light `}

>

About

</Link>

</li> \*/}

<li className="nav-item mb-2">

<Link

to={'/signup'}

className={`nav-link text-light `}

>

SignUp

</Link>

</li>

<li className="nav-item mb-2">

<Link

to={'/login'}

className={`nav-link text-light `}

>

Login

</Link>

</li>

</ul>

</Container>

</Navbar>

<section className="vh-100 gradient-custom">

<div className="container py-5 h-100">

<div className="row d-flex justify-content-center align-items-center h-100">

<div className="col-12 col-md-8 col-lg-6 col-xl-5">

<div className="card bg-dark text-white">

<div className="card-body p-5 text-center">

<div className="mb-md-5 mt-md-4 pb-5">

<h2 className="fw-bold mb-4">Login For Registering the Complaint</h2>

<p className="text-white-50 mb-5">Please enter your Credentials!</p>

<form onSubmit={handleSubmit}>

<div className="form-outline form-white mb-4">

<input type="email" name="email" value={user.email} onChange={handleChange} className="form-control form-control-lg" required />

<label className="form-label" htmlFor="email">Email</label>

</div>

<div className="form-outline form-white mb-4">

<input type="password" name="password" value={user.password} onChange={handleChange} className="form-control form-control-lg" autoComplete="off" required />

<label className="form-label" htmlFor="password">Password</label>

</div>

<button className="btn btn-outline-light btn-lg px-5" type="submit">Login</button>

</form>

</div>

<div>

<p className="mb-0">Don't have an account? <Link to="/SignUp">SignUp</Link></p>

</div>

</div>

</div>

</div>

</div>

</div>

</section>

<Footer/>

</>

);

};

export default Login;

* SignUp.jsx

import axios from 'axios'

import React, { useState } from 'react'

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

import Dropdown from 'react-bootstrap/Dropdown';

import Container from 'react-bootstrap/Container';

import Navbar from 'react-bootstrap/Navbar';

import Footer from './FooterC'

const SignUp = () => {

const [title, setTitle] = useState("Select User")

const [user, setUser] = useState({

name: "",

email: "",

password: "",

phone: "",

userType: ""

})

const handleChange = (e) => {

setUser({ ...user, [e.target.name]: e.target.value })

}

const handleTitle = (select) => {

setTitle(select)

setUser({ ...user, userType: select });

}

const handleSubmit = async (e) => {

e.preventDefault()

const updatedUser = { ...user, userType: title };

axios.post("http://localhost:8000/SignUp", updatedUser)

.then((res) => {

alert("record submitted")

JSON.stringify(res.data.user)

})

.catch((err) => {

console.log(err)

})

setUser({

name: "",

email: "",

password: "",

phone: "",

userType: ""

})

}

return (

<>

<Navbar bg="dark" variant="dark">

<Container>

<Navbar.Brand>ComplaintCare </Navbar.Brand>

<ul className="navbar-nav">

<li className="nav-item mb-2">

<Link to={'/'}

className={`nav-link text-light `}

>

Home

</Link>

</li>

<li className="nav-item mb-2">

<Link

to={'/signup'}

className={`nav-link text-light `}

>

SignUp

</Link>

</li>

<li className="nav-item mb-2">

<Link

to={'/login'}

className={`nav-link text-light `}

>

Login

</Link>

</li>

</ul>

</Container>

</Navbar>

<section className="gradient-custom">

<div className="container">

<div className="row d-flex justify-content-center align-items-center h-100">

<div className="col-12 col-md-8 col-lg-6 col-xl-5">

<div className="card bg-dark text-white">

<div className="card-body p-5 text-center">

<div className="mb-md-5 mt-md-4 pb-5">

<h2 className="fw-bold mb-4 ">SignUp For Registering the Complaint</h2>

<p className="text-white-50 mb-4">Please enter your Details</p>

<form onSubmit={handleSubmit}>

<div className="form-outline form-white mb-4">

<input type="name" name="name" value={user.name} onChange={handleChange} className="form-control form-control-lg" required />

<label className="form-label" htmlFor="name">Full Name</label>

</div>

<div className="form-outline form-white mb-2">

<input type="email" name="email" value={user.email} onChange={handleChange} className="form-control form-control-lg" required />

<label className="form-label" htmlFor="email">Email</label>

</div>

<div className="form-outline form-white mb-2">

<input type="password" name="password" value={user.password} onChange={handleChange} className="form-control form-control-lg" required />

<label className="form-label" htmlFor="password">Password</label>

</div>

<div className="form-outline form-white mb-2">

<input type="phone" name="phone" value={user.phone} onChange={handleChange} className="form-control form-control-lg" required />

<label className="form-label" htmlFor="mobile">Mobile No.</label>

</div>

<div className="form-outline form-white mb-2">

<Dropdown>

<Dropdown.Toggle variant="secondary" id="dropdown-basic">

{title}

</Dropdown.Toggle>

<Dropdown.Menu>

<Dropdown.Item onClick={() => handleTitle("Ordinary")}>Ordinary</Dropdown.Item>

<Dropdown.Item onClick={() => handleTitle("Admin")}>Admin</Dropdown.Item>

<Dropdown.Item onClick={() => handleTitle("Agent")}>Agent</Dropdown.Item>

</Dropdown.Menu>

</Dropdown>

<label className="form-label" htmlFor="mobile">Select User Type</label>

</div>

<button className="btn btn-outline-light btn-lg px-5 mt-3" type="submit">Register</button>

</form>

</div>

<div>

<p className="mb-0">Had an account?<Link to={"/Login"}>Login</Link></p>

</div>

</div>

</div>

</div>

</div>

</div>

</section>

<Footer/>

</>

)

}

export default SignUp

* USER:
* Complaint.jsx

import axios from 'axios'

import React, { useState } from 'react'

const Complaint = () => {

const user = JSON.parse(localStorage.getItem('user'))

const [userComplaint, setUserComplaint] = useState({

userId: user.\_id,

name: '',

address: '',

city: '',

state: '',

pincode: '',

status: '',

comment: ''

})

const handleChange = (e) => {

const { name, value } = e.target

setUserComplaint({ ...userComplaint, [name]: value })

}

const handleClear = () => {

setUserComplaint({

userId: '',

name: '',

address: '',

city: '',

state: '',

pincode: '',

status: '',

comment: ''

})

}

const handleSubmit = async (e) => {

e.preventDefault()

const user = JSON.parse(localStorage.getItem('user'))

const { \_id } = user

axios.post(`http://localhost:8000/Complaint/${\_id}`, userComplaint)

.then(res => {

JSON.stringify(res.data.userComplaint)

alert("Your Complaint has been send!!")

handleClear()

})

.catch(err => {

console.log(err)

alert("Something went wrong!!")

})

}

return (

<>

<div className="text-white complaint-box">

<form onSubmit={handleSubmit} className="compliant-form row bg-dark ">

<div className="col-md-6 p-3 p-3">

<label htmlFor="name" className="form-label">Name</label>

<input name="name" onChange={handleChange} value={userComplaint.name} type="text" className="form-control" id="name" required />

</div>

<div className="col-md-6 p-3">

<label htmlFor="address" className="form-label">Address</label>

<input name="address" onChange={handleChange} value={userComplaint.address} type="text" className="form-control" id="address" required />

</div>

<div className="col-md-6 p-3">

<label htmlFor="city" className="form-label">City</label>

<input name="city" onChange={handleChange} value={userComplaint.city} type="text" className="form-control" id="city" required />

</div>

<div className="col-md-6 p-3">

<label htmlFor="state" className="form-label">State</label>

<input name="state" onChange={handleChange} value={userComplaint.state} type="text" className="form-control" id="state" required />

</div>

<div className="col-md-6 p-3">

<label htmlFor="pincode" className="form-label">Pincode</label>

<input name="pincode" onChange={handleChange} value={userComplaint.pincode} type="text" className="form-control" id="pincode" required />

</div>

{/\* <div className="col-md-6 p-3">

<label htmlFor="file" className="form-label">Document</label>

<input name="file" type="file" className="form-control" id="file" required />

</div> \*/}

<div className="col-md-6 p-3">

<label htmlFor="status" className="form-label">Status</label>

<input placeholder='type pending' name="status" onChange={handleChange} value={userComplaint.status} type="text" className="form-control" id="pincode" required />

</div>

<label className=" p-3form-label text-light" htmlFor="comment">Descrption</label>

<div className="form-floating">

<textarea name="comment" onChange={handleChange} value={userComplaint.comment} className="form-control" required></textarea>

</div>

<div className="text-center p-1 col-12">

<button type="submit" onClick={handleSubmit} className="mt-2 btn btn-success">Register</button>

</div>

</form>

</div>

</>

)

}

export default Complaint

* HomePage.jsx

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

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

import Footer from '../common/FooterC'

import Complaint from '../user/Complaint';

import Status from '../user/Status';

const HomePage = () => {

const navigate = useNavigate();

const [activeComponent, setActiveComponent] = useState('Complaint');

const [userName, setUserName] = useState('');

useEffect(() => {

const getData = async () => {

try {

const user = JSON.parse(localStorage.getItem('user'));

if (user) {

const { name } = user;

setUserName(name);

} else {

navigate('/');

}

} catch (error) {

console.log(error);

}

};

getData();

}, [navigate]);

const handleNavLinkClick = (componentName) => {

setActiveComponent(componentName);

};

const Logout = () => {

localStorage.removeItem('user');

navigate('/');

};

return (

<>

<nav className="navbar navbar-expand-lg bg-dark">

<div className="container-fluid">

<h1 className="navbar-brand text-light">Hi, {userName}</h1>

<div className="mt-2 navbar-collapse text-light" id="navbarSupportedContent">

<ul className="navbar-nav me-auto mb-lg-0">

<li className="nav-item mb-2">

<NavLink

className={`nav-link text-light ${activeComponent === 'Complaint' ? 'active' : ''}`}

onClick={() => handleNavLinkClick('Complaint')}

>

Complaint Register

</NavLink>

</li>

<li className="nav-item mb-2">

<NavLink

className={`nav-link text-light ${activeComponent === 'Status' ? 'active' : ''}`}

onClick={() => handleNavLinkClick('Status')}

>

Status

</NavLink>

</li>

</ul>

</div>

<button className="btn btn-danger" onClick={Logout}>

LogOut

</button>

</div>

</nav>

<div className="body">

<div className="container">

{activeComponent === 'Complaint' ? <Complaint /> : null}

{activeComponent === 'Status' ? <Status /> : null}

</div>

</div>

<Footer />

</>

);

};

export default HomePage;

* Status.jsx

import axios from 'axios'

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

import Card from 'react-bootstrap/Card';

import Alert from 'react-bootstrap/Alert';

import { Button } from 'react-bootstrap';

import ChatWindow from '../common/ChatWindow';

import Collapse from 'react-bootstrap/Collapse';

const Status = () => {

const [toggle, setToggle] = useState({})

const [statusCompliants, setStatusCompliants] = useState([]);

useEffect(() => {

const user = JSON.parse(localStorage.getItem('user'));

const { \_id } = user;

axios.get(`http://localhost:8000/status/${\_id}`)

.then((res) => {

setStatusCompliants(res.data);

})

.catch((err) => {

console.log(err);

});

}, []);

const handleToggle = (complaintId) => {

setToggle((prevState) => ({

...prevState,

[complaintId]: !prevState[complaintId],

}));

};

return (

<>

<div style={{ display: "flex", flexWrap: "wrap", margin: "20px" }}>

{statusCompliants.length > 0 ? (

statusCompliants.map((complaint, index) => {

const open = toggle[complaint.\_id] || false;

return (

<Card key={index} style={{ width: '18.5rem', margin: '0 15px 15px 0' }}>

<Card.Body>

<Card.Title>Name: {complaint.name}</Card.Title>

<Card.Text>Address: {complaint.address}</Card.Text>

<Card.Text>City: {complaint.city}</Card.Text>

<Card.Text>State: {complaint.state}</Card.Text>

<Card.Text>Pincode: {complaint.pincode}</Card.Text>

<Card.Text>Comment: {complaint.comment}</Card.Text>

<Card.Text>Status: {complaint.status}</Card.Text>

<Button onClick={() => handleToggle(complaint.\_id)}

aria-controls={`collapse-${complaint.\_id}`}

aria-expanded={open} variant="primary">

Message

</Button>

<div style={{ minHeight: '100%' }}>

<Collapse in={open} dimension="width">

<div id="example-collapse-text">

<Card body style={{ width: '250px', marginTop: '12px' }}>

<ChatWindow key={complaint.complaintId} complaintId={complaint.\_id} name={complaint.name} />

</Card>

</div>

</Collapse>

</div>

</Card.Body>

</Card>

)

})

) : (

<Alert variant="info">

<Alert.Heading>No complaints to show</Alert.Heading>

</Alert>

)}

</div>

</>

)

}

export default Status;

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

// const Status = () => {

// const [city, setCity] = useState('');

// const [state, setState] = useState('');

// const [complaint, setComplaint] = useState("")

// // useEffect(()=>{

// // const id = localStorage.getItem("user")

// // console.log(id)

// // // axios.get(`http://localhost:8000/status${id}`)

// // // .then((res)=>{

// // // const { city, state, complaint } = res.data;

// // // console.log(city,state,complaint)

// // // setState(state);

// // // setCity(city);

// // // setComplaint(complaint)

// // // })

// // // .catch((err)=>{

// // // console.log(err)

// // // })

// // },[])

// useEffect(() => {

// const user = JSON.parse(localStorage.getItem('user'));

// const { \_id } = user;

// console.log(\_id);

// axios.get(`http://localhost:8000/status/${\_id}`)

// .then((res) => {

// axios.get('http://localhost:8000/Complaint')

// .then((res) => {

// const { city, state, complaint } = res.data;

// console.log(city, state, complaint)

// setState(state);

// setCity(city);

// setComplaint(complaint)

// })

// .catch((err) => {

// console.log(err)

// })

// })

// .catch((err) => {

// console.log(err)

// })

// }, []);

// return (

// <>

// <div className="row">

// <div className="status col-sm-6 mb-sm-0">

// <div className="card status-card">

// <div className="card-body">

// <h5 className="card-title">City:{city}</h5>

// <p className="card-text">State:{state} </p>

// <p className="card-text">Complaint:{complaint} </p>

// </div>

// </div>

// </div>

// <div className="status col-sm-6 mb-sm-0">

// <div className="card status-card">

// <div className="card-body">

// <h5 className="card-title">h</h5>

// <p className="card-text">Lorem ipsum dolor sit amet, consectetur adipisicing elit. <br />In, voluptatibus!</p>

// </div>

// </div>

// </div>

// </div>

// </>

// )

// }

// export default Status

* AppCSS:

\* {

margin: 0;

padding: 0;

box-sizing: border-box;

}

.App{

max-width: 100%;

max-width: 100%;

background-color: #B0BEC5;

}

ul {

display: flex;

list-style: none;

flex-direction: row;

justify-content: space-evenly;

}

.navigate p {

cursor: pointer;

}

.container {

margin: 10px;

}

.home-container {

width: 100vw;

height: 88vh;

display: flex;

}

.left-side,

.right-side {

width: 50%;

height: 100%;

display: flex;

flex-direction: column;

justify-content: center;

align-items: center;

}

.left-side img {

width: 120%;

height: 70%;

}

.right-side {

display: flex;

align-items: center;

justify-content: center;

}

span {

font-family: Arial, Helvetica, sans-serif;

}

.f-letter {

font-size: 3em;

}

.s-letter {

font-size: 1.5em;

}

.t-letter {

font-size: 1.2em;

}

.register {

border-radius: 30px;

background-color: #00796B;

}

.container .form {

height: 100%;

width: auto;

}

.data {

width: auto;

height: auto;

display: flex;

justify-content: center;

align-items: center;

border: 1px solid lightgray;

}

.complaint-box {

position: relative;

margin-top: 10px;

width: 100%;

height: auto;

}

.compliant-form {

position: absolute;

left: 50%;

top: 50%;

transform: translate(-50%);

padding: 15px;

border-radius: 5px;

margin-top: 38px;

}

.nav-item p {

cursor: pointer;

font-size: 15px;

margin-top: 2px;

/\* Change this to the desired color for the active link \*/

}

.nav-item p:hover {

color: red;

}

.grid-container {

display: grid;

grid-template-columns: 4fr 1fr;

gap: 2rem;

}

.chat-container {

display: flex;

flex-direction: column;

height: 100%;

}

.chat-container h1{

font-size: 22px;

text-align: center;

margin-bottom: 20px;

}

.message-window {

flex: 1;

overflow-x: auto;

padding: 10px;

width: 100%;

max-height: 200px;

}

.message {

background-color: #f5f5f5;

padding: 5px;

margin-bottom: 10px;

}

.input-container {

display: flex;

align-items: center;

padding: 10px;

}

.input-container input[type="text"] {

width: 20px;

flex: 1;

padding: 5px;

margin-right: 10px;

border: none;

border-radius: 5px;

padding-left: 10px;

background-color: rgb(250, 247, 247);

}

.input-container button {

padding: 5px 10px;

color: white;

font-size: 12px;

border: none;

cursor: pointer;

border-radius: 15px;

}

.accordian, .body{

min-height: 70vh;

}

@media screen and (max-width: 480px) {

.chat-container {

height: auto;

}

}

* App.js

\* {

margin: 0;

padding: 0;

box-sizing: border-box;

}

.App{

max-width: 100%;

max-width: 100%;

background-color: #B0BEC5;

}

ul {

display: flex;

list-style: none;

flex-direction: row;

justify-content: space-evenly;

}

.navigate p {

cursor: pointer;

}

.container {

margin: 10px;

}

.home-container {

width: 100vw;

height: 88vh;

display: flex;

}

.left-side,

.right-side {

width: 50%;

height: 100%;

display: flex;

flex-direction: column;

justify-content: center;

align-items: center;

}

.left-side img {

width: 120%;

height: 70%;

}

.right-side {

display: flex;

align-items: center;

justify-content: center;

}

span {

font-family: Arial, Helvetica, sans-serif;

}

.f-letter {

font-size: 3em;

}

.s-letter {

font-size: 1.5em;

}

.t-letter {

font-size: 1.2em;

}

.register {

border-radius: 30px;

background-color: #00796B;

}

.container .form {

height: 100%;

width: auto;

}

.data {

width: auto;

height: auto;

display: flex;

justify-content: center;

align-items: center;

border: 1px solid lightgray;

}

.complaint-box {

position: relative;

margin-top: 10px;

width: 100%;

height: auto;

}

.compliant-form {

position: absolute;

left: 50%;

top: 50%;

transform: translate(-50%);

padding: 15px;

border-radius: 5px;

margin-top: 38px;

}

.nav-item p {

cursor: pointer;

font-size: 15px;

margin-top: 2px;

/\* Change this to the desired color for the active link \*/

}

.nav-item p:hover {

color: red;

}

.grid-container {

display: grid;

grid-template-columns: 4fr 1fr;

gap: 2rem;

}

.chat-container {

display: flex;

flex-direction: column;

height: 100%;

}

.chat-container h1{

font-size: 22px;

text-align: center;

margin-bottom: 20px;

}

.message-window {

flex: 1;

overflow-x: auto;

padding: 10px;

width: 100%;

max-height: 200px;

}

.message {

background-color: #f5f5f5;

padding: 5px;

margin-bottom: 10px;

}

.input-container {

display: flex;

align-items: center;

padding: 10px;

}

.input-container input[type="text"] {

width: 20px;

flex: 1;

padding: 5px;

margin-right: 10px;

border: none;

border-radius: 5px;

padding-left: 10px;

background-color: rgb(250, 247, 247);

}

.input-container button {

padding: 5px 10px;

color: white;

font-size: 12px;

border: none;

cursor: pointer;

border-radius: 15px;

}

.accordian, .body{

min-height: 70vh;

}

@media screen and (max-width: 480px) {

.chat-container {

height: auto;

}

}

* Index.js

import "./App.css";

import "../node\_modules/bootstrap/dist/css/bootstrap.min.css";

import { BrowserRouter as Router, Routes, Route } from "react-router-dom";

import HomePage from "./components/user/HomePage";

import Login from "./components/common/Login";

import SignUp from "./components/common/SignUp";

import Complaint from "./components/user/Complaint";

import Status from "./components/user/Status";

import AdminHome from "./components/admin/AdminHome";

import AgentHome from "./components/agent/AgentHome";

import UserInfo from "./components/admin/UserInfo";

import Home from "./components/common/Home";

import AgentInfo from "./components/admin/AgentInfo";

function App() {

const isLoggedIn = !!localStorage.getItem("user");

return (

<div className="App">

<Router>

<Routes>

<Route exact path="/" element={<Home />} />

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

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

{isLoggedIn ? (

<>

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

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

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

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

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

<Route path="/Homepage" element={<HomePage />} />

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

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

</>

) : (

<Route to="/Login" />

)}

</Routes>

</Router>

</div>

);

}

export default App;

* .gitignore:

# See https://help.github.com/articles/ignoring-files/ for more about ignoring files.

# dependencies

/node\_modules

/.pnp

.pnp.js

# testing

/coverage

# production

/build

# misc

.DS\_Store

.env.local

.env.development.local

.env.test.local

.env.production.local

npm-debug.log\*

yarn-debug.log\*

yarn-error.log\*

* BACKEND:
* Config.js

const mongoose = require("mongoose")

mongoose.connect("mongodb://127.0.0.1:27017/details")

.then(()=>{

console.log("connected to mongodb")

})

* Index.js

const express = require("express");

const cors = require("cors");

require("./config");

const {

ComplaintSchema,

UserSchema,

AssignedComplaint,

MessageSchema,

} = require("./Schema");

const app = express();

const PORT = 8000;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

app.use(express.json());

app.use(cors());

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

/\*\*\*\*\*\*\*message \*\*\*\*\*\*\*\*\*\*\*\*/

app.post("/messages", async (req, res) => {

try {

const { name, message, complaintId } = req.body;

const messageData = new MessageSchema({

name,

message,

complaintId,

});

const messageSaved = await messageData.save();

res.status(200).json(messageSaved);

} catch (error) {

res.status(500).json({ error: "Failed to send message" });

}

});

app.get("/messages/:complaintId", async (req, res) => {

try {

const { complaintId } = req.params;

const messages = await MessageSchema.find({ complaintId }).sort(

"-createdAt"

);

res.json(messages);

} catch (error) {

res.status(500).json({ error: "Failed to retrieve messages" });

}

});

/\*\*\*\*for signup user\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

app.post("/SignUp", async (req, res) => {

const user = new UserSchema(req.body);

try {

const resultUser = await user.save();

res.send(resultUser);

} catch (error) {

res.status(500).send(error);

}

});

//////////////////////for login user///////////////////

app.post("/Login", async (req, res) => {

const { email, password } = req.body;

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

if (!user) {

return res.status(401).json({ message: "User doesn`t exists" });

}

if (user.email === email && user.password === password) {

res.json(user);

} else {

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

}

});

//////////////////////////for fetching agent in admin portal///////////////

app.get("/AgentUsers", async (req, res) => {

try {

const { userType } = req.params;

const users = await UserSchema.find({ userType: "Agent" });

if (users.length === 0) {

return res.status(404).json({ error: "User not found" });

} else {

return res.status(200).json(users);

}

} catch (error) {

console.log(error);

res.status(500).json({ error: "Internal Server Error" });

}

});

//////////////////////////for fetching ordinary user in admin portal///////////////

app.get("/OrdinaryUsers", async (req, res) => {

try {

const users = await UserSchema.find({ userType: "Ordinary" });

if (users.length === 0) {

return res.status(404).json({ error: "User not found" });

} else {

return res.status(200).json(users);

}

} catch (error) {

console.log(error);

res.status(500).json({ error: "Internal Server Error" });

}

});

//////////////////////////for fetching ordinary user in admin portal///////////////

app.get("/AgentUsers", async (req, res) => {

try {

// const { userType } = req.params;

const agentUsers = await UserSchema.find({ userType: "Agent" });

if (agentUsers.length === 0) {

return res.status(404).json({ error: "User not found" });

} else {

return res.status(200).json(agentUsers);

}

} catch (error) {

console.log(error);

res.status(500).json({ error: "Internal Server Error" });

}

});

//////////////////displaying agent with id/////////////////

app.get("/AgentUsers/:agentId", async (req, res) => {

try {

const { agentId } = req.params;

const user = await UserSchema.findOne({ \_id: agentId });

if (user.userType === "Agent") {

return res.status(200).json(user);

} else {

return res.status(404).json({ error: "User not found" });

}

} catch {

return res.status(500).json({ error: "Internal Server Error" });

}

});

////////////for deleting the user from admin portal////////////////

app.delete("/OrdinaryUsers/:id", async (req, res) => {

try {

const { id } = req.params;

const user = await UserSchema.findOne({ \_id: id });

if (!user) {

return res.status(404).json({ error: "User not found" });

} else {

await UserSchema.deleteOne({ \_id: id });

await ComplaintSchema.deleteOne({ userId: id });

return res.status(200).json({ message: "User deleted successfully" });

}

} catch (error) {

console.log(error);

res.status(500).json({ error: "Internal Server Error" });

}

});

///////////////complaint register by user and its status checking///////////////

app.post("/Complaint/:id", async (req, res) => {

const UserId = req.params.id;

try {

const user = await UserSchema.findById(UserId);

if (!user) {

return res.status(404).json({ error: "User not found" });

} else {

const complaint = new ComplaintSchema(req.body);

let resultComplaint = await complaint.save();

res.send(resultComplaint).status(200);

}

} catch (error) {

console.error(error);

res.status(500).json({ error: "Failed to register complaint" });

}

});

/////////////////for the all complaints made by the single user/////////////

app.get("/status/:id", async (req, res) => {

const userId = req.params.id;

try {

const user = await UserSchema.findById(userId);

if (!user) {

return res.status(404).json({ error: "User not found" });

} else {

const comment = await ComplaintSchema.find({ userId: userId });

res.json(comment);

}

} catch (error) {

console.error(error);

res.status(500).json({ error: "Failed to retrieve user" });

}

});

/////////////status of complaint in admin page/////////////////////////////////////////

app.get("/status", async (req, res) => {

try {

const complaint = await ComplaintSchema.find();

res.json(complaint);

} catch (error) {

console.error(error);

res.status(500).json({ error: "Failed to retrieve Complaints" });

}

});

////////////Assigned complaint by admin//////////////////

app.post("/assignedComplaints", (req, res) => {

try {

const assignedComplaint = req.body;

AssignedComplaint.create(assignedComplaint);

res.sendStatus(201);

} catch (error) {

console.error(error);

res.status(500).json({ error: "Failed to add assigned complaint" });

}

});

////////////////complaints in agent homepage////////////////////

app.get("/allcomplaints/:agentId", async (req, res) => {

try {

const agentId = req.params.agentId;

const complaints = await AssignedComplaint.find({ agentId: agentId });

// Fetch all complaintIds from the complaints

const complaintIds = complaints.map((complaint) => complaint.complaintId);

// Fetch the corresponding complaints with their names and cities

const complaintDetails = await ComplaintSchema.find({

\_id: { $in: complaintIds },

});

// Merge the complaint details into the complaints array

const updatedComplaints = complaints.map((complaint) => {

const complaintDetail = complaintDetails.find(

(detail) => detail.\_id.toString() === complaint.complaintId.toString()

);

return {

...complaint,

name: complaintDetail.name,

city: complaintDetail.city,

state: complaintDetail.state,

address: complaintDetail.address,

pincode: complaintDetail.pincode,

comment: complaintDetail.comment,

};

});

res.json(updatedComplaints);

} catch (error) {

console.log(error);

res.status(500).json({ error: "Failed to get complaints" });

}

});

////////////////////updating the user profile by admin/////////////////////////////

app.put("/user/:\_id", async (req, res) => {

try {

const { \_id } = req.params;

const { name, email, phone } = req.body;

const user = await UserSchema.findByIdAndUpdate(

\_id,

{ name, email, phone },

{ new: true }

);

if (!user) {

res.status(404).json({ error: "User not found" });

}

res.json(user);

} catch (error) {

res.status(500).json({ error: "Failed to update the user" });

}

});

////////////////updating the complaint from the agent/////////////////////////////

app.put("/complaint/:complaintId", async (req, res) => {

try {

const { complaintId } = req.params;

const { status } = req.body;

if (!complaintId || !status) {

return res.status(400).json({ error: "Missing complaintId or status" });

}

const updatedComplaint = await ComplaintSchema.findByIdAndUpdate(

complaintId,

{ status },

{ new: true }

);

const assigned = await AssignedComplaint.findOneAndUpdate(

{complaintId: complaintId},

{ status },

{ new: true }

);

if (!updatedComplaint && !assigned) {

return res.status(404).json({ error: "Complaint not found" });

}

res.json(updatedComplaint);

} catch (error) {

console.log(error);

res.status(500).json({ error: "Failed to update complaint" });

}

});

app.listen(PORT, () => console.log(server started at ${PORT}));

* Package.json

{

"name": "backend",

"version": "1.0.0",

"description": "",

"main": "index.js",

"scripts": {

"start": "nodemon index.js"

},

"keywords": [],

"author": "",

"license": "ISC",

"dependencies": {

"bcrypt": "^5.1.0",

"cors": "^2.8.5",

"express": "^4.18.2",

"express-session": "^1.17.3",

"mongoose": "^7.1.1",

"nodemon": "^2.0.22"

}

}

* Schema.js

const mongoose = require("mongoose");

// const bcrypt = require("bcrypt");

/////////////////user///////////////////////////////

const userSchema = mongoose.Schema({

name: { type: String, required: 'Name is require' },

email: { type: String, required: 'Email is require' },

password: { type: String, required: 'Password is require' },

phone: { type: Number, required: 'Phone is require' },

userType: { type: String, required: 'UserType is require' },

},

{

timestamps: true,

});

// userSchema.pre("save", async fnction (next) {

// try {

// if (!this.isModified("password")) {

// return next();

// }

// const hashedPassword = bcrypt.hash(this.password, 10);

// this.password = hashedPassword;

// next();

// } catch (error) {

// return next(error);

// }

// });

const UserSchema = mongoose.model("user\_Schema", userSchema);

///////////////complaint///////////////////

const complaintSchema = mongoose.Schema({

userId: {type: mongoose.Schema.Types.ObjectId, required: true, ref: "user\_Schema" },

name: {type: String, required: true },

address: { type: String, required: true },

city: { type: String, required: true },

state: { type: String, required: true },

pincode: { type: Number, required: true },

comment: { type: String, required: true },

status: { type: String, required: true },

});

const ComplaintSchema = mongoose.model("complaint\_schema", complaintSchema)

///////////assigned complaint schema////////////////////////

const assignedComplaint = mongoose.Schema({

agentId : {type: mongoose.Schema.Types.ObjectId, required: true, ref: "user\_Schema" },

complaintId : {type: mongoose.Schema.Types.ObjectId, required: true, ref: "complaint\_schema" },

status: {type: String, required: true },

agentName: {type: String, required: true },

})

const AssignedComplaint = mongoose.model("assigned\_complaint",assignedComplaint)

////////////////////chatWindow schema/////////////////////////

const messageSchema = new mongoose.Schema({

name: {type: String, required: 'name is required'},

message: {type: String, required: 'message is required'},

complaintId: {type: mongoose.Schema.Types.ObjectId, ref: "assigned\_complaint"}

}, { timestamps: true });

const MessageSchema = mongoose.model('message', messageSchema);

module.exports = {

UserSchema,

ComplaintSchema,

AssignedComplaint,

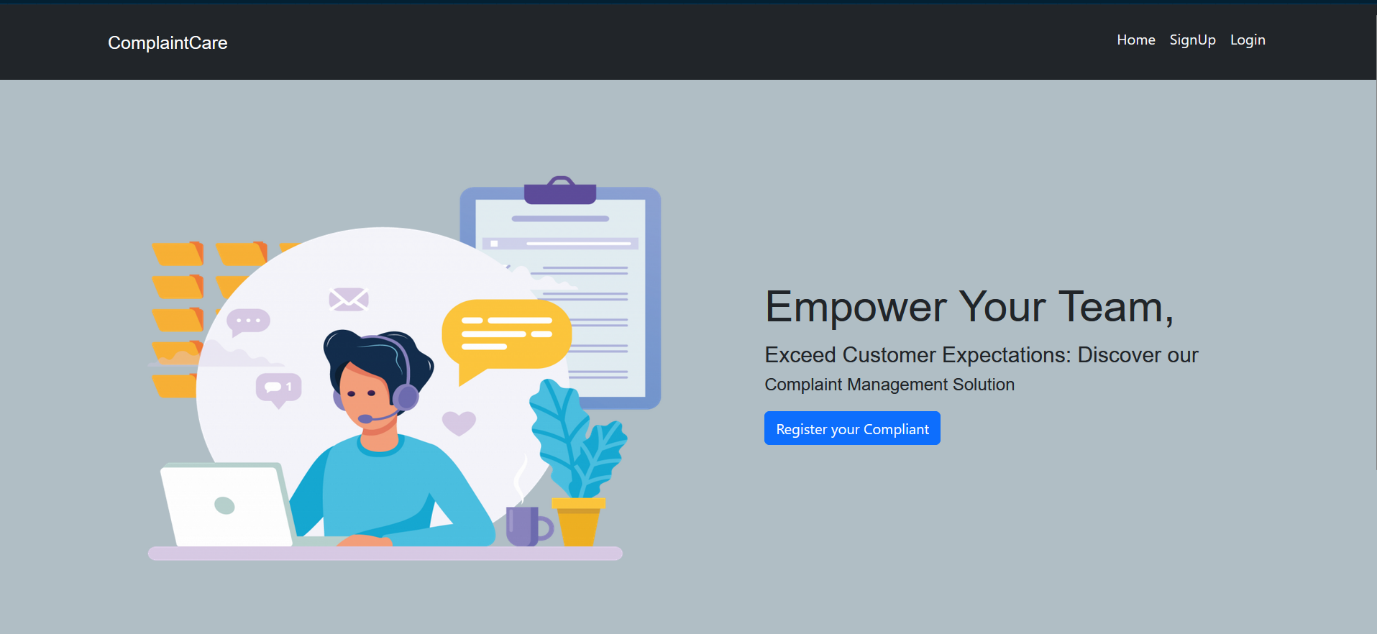
MessageSchema,

};

**Project Implementation:**

On completing the development part, we then run the application one last time to verify all the functionalities and look for any bugs in it. The user interface of the application looks a bit like the one’s provided below.

* Landing Page



* Login Page



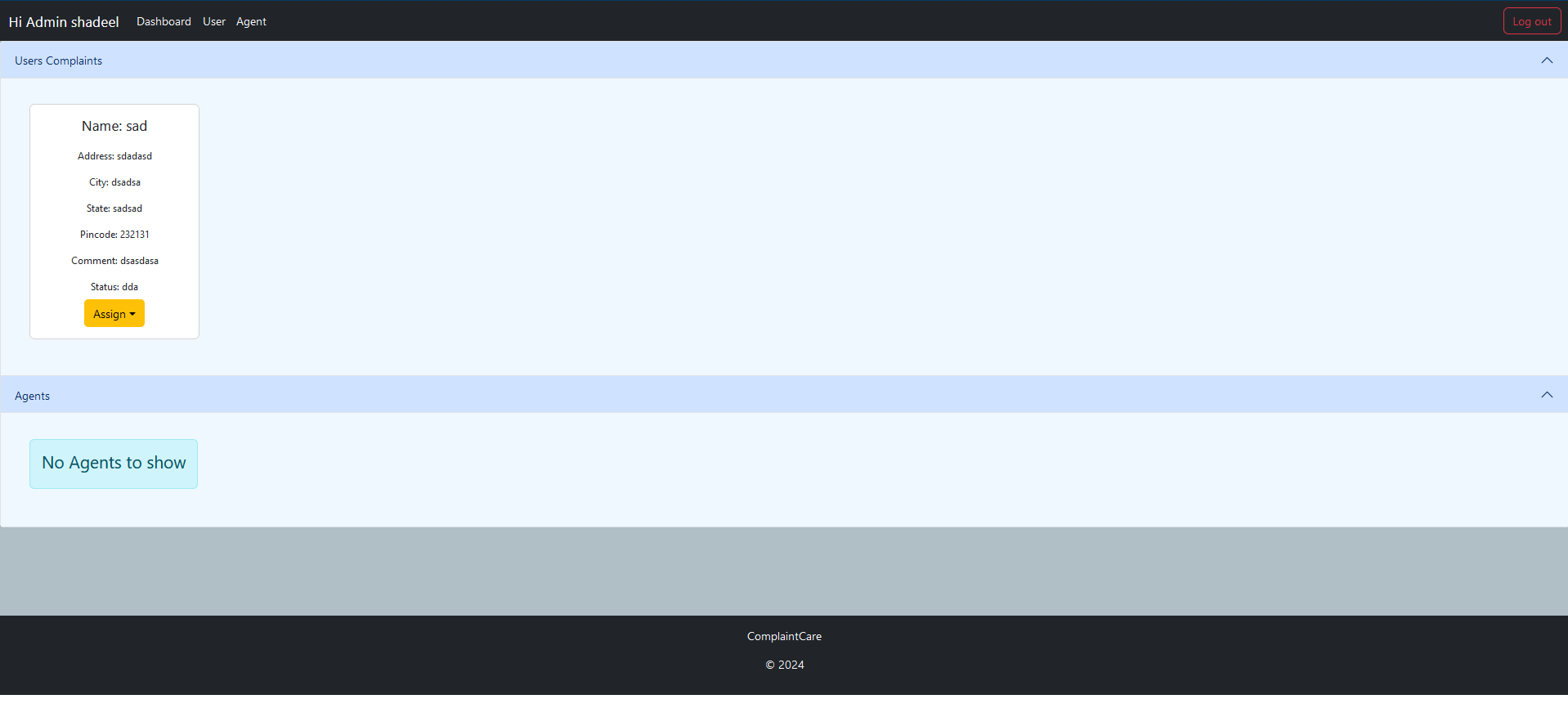
* Registration Page



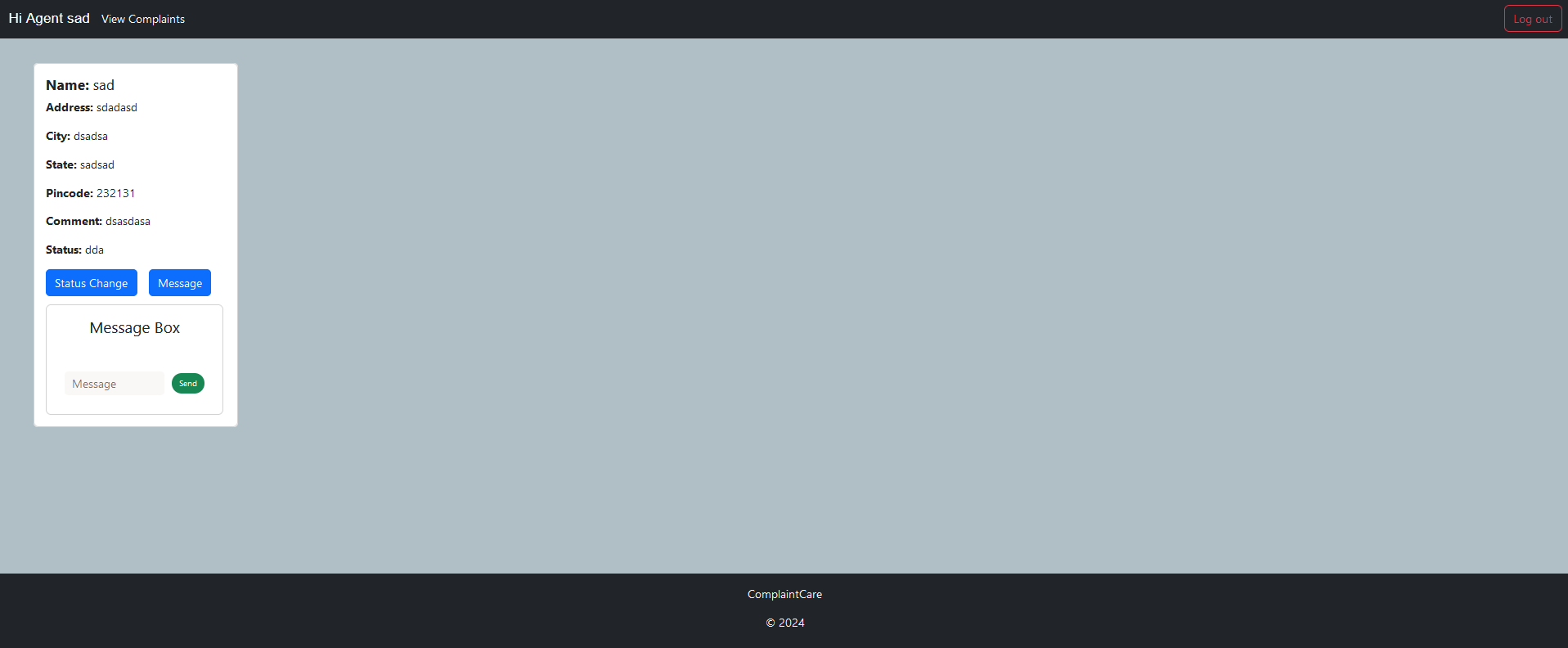
* Common Dashboard For Complaint



* Admin Dashboard



* Agent Dashboard



**Conclusion:**

The development and implementation of an online complaints system represent a significant step toward enhancing transparency, accountability, and customer satisfaction in both public and private sectors. This project demonstrated how technology can streamline the complaint submission process, reduce response times, and ensure that grievances are tracked and addressed efficiently. By creating a user-friendly interface, secure data handling mechanisms, and an organized complaint resolution workflow, the platform not only empowers users to voice their concerns but also enables organizations to identify recurring issues and improve their services. Moving forward, integrating features like real-time tracking, automated notifications, and data analytics can further elevate the effectiveness and impact of online complaint management systems.