ResolveNow : Your Platform For Online Complaints

TeamID : LTVIP2025TMID55215

***Member 1**:Kotte Yasaswini Prasanna Lakshmi*

***Member 2 :** Kunapareddy Srilakshmi*

***Member 3** : Kota Vismitha Chandra*

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.

It consists of some key features which include:

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.
6. 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.

Description:

The Online Complaint Registration and Management System is a user-friendly software solution designed to streamline the process of submitting, tracking, and resolving complaints or issues encountered by individuals or organizations. It provides a centralized platform for efficient

complaint management, allowing users to securely register complaints, track their progress in real-time, and interact with assigned agents for issue resolution. With features such as automatic notifications, intelligent complaint routing, and robust security measures, this system ensures timely and effective handling of complaints while prioritizing user Details.

Scenario Based Case Study:

Scenario: John, a customer, recently encountered a problem with a product he purchased online. He notices a defect in the item and decides to file a complaint using the Online Complaint Registration and Management System.

User Registration and Login:

- John visits the complaint management system's website and clicks on the "Sign Up" button to create a new account.
- He fills out the registration form, providing his full name, email address, and a secure password.
- After submitting the form, John receives a verification email and confirms his account.
- He then logs into the system using his email and password.

Complaint Submission:

- Upon logging in, John is redirected to the dashboard where he sees options to register a new complaint.
- He clicks on the "Submit Complaint" button and fills out the complaint form.
- John describes the issue in detail, attaches relevant documents or images showcasing the defect, and provides additional information such as his contact details and the product's purchase date.
- After reviewing the information, John submits the complaint.

Tracking and Notifications:

- After submitting the complaint, John receives a confirmation message indicating that his complaint has been successfully registered.
- He navigates to the "My Complaints" section of the dashboard, where he can track the status of his complaint in real-time.
- John receives email notifications whenever there is an update on his complaint, such as it being assigned to an agent or its resolution status.

Interaction with Agent:

- A customer service agent, Sarah, is assigned to handle John's complaint.
- Sarah reviews the details provided by John and contacts him through the system's built-in messaging feature.
- John receives a notification about Sarah's message and accesses the chat window to communicate with her.
- They discuss the issue further, and Sarah assures John that the company will investigate and resolve the problem promptly.

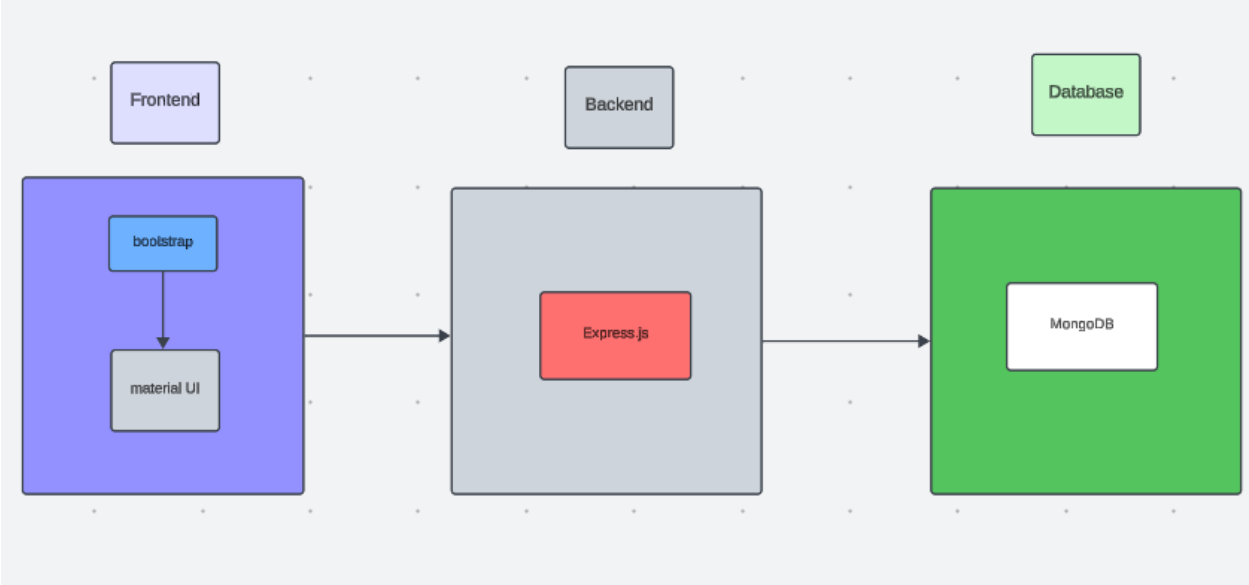
Resolution and Feedback:

- After investigating the complaint, the company identifies the defect in the product and offers John a replacement or refund.
- John receives a notification informing him of the resolution, along with instructions on how to proceed.
- He provides feedback on his experience with the complaint handling process, expressing his satisfaction with the prompt resolution and courteous service provided by Sarah.

Admin Management:

- Meanwhile, the system administrator monitors all complaints registered on the platform.
- The admin assigns complaints to agents based on their workload and expertise.
- They oversee the overall operation of the complaint management system, ensuring compliance with platform policies and 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.

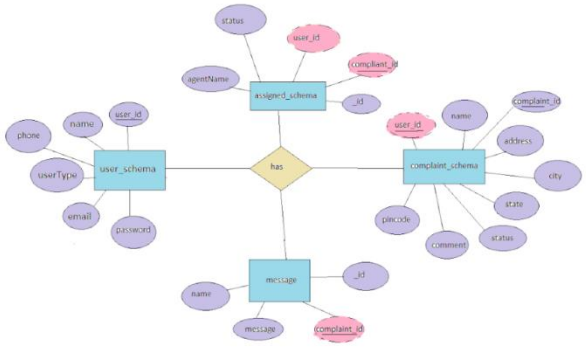
The frontend utilizes the bootstrap and material UI library to establish real-time and better UI experience for any user whether it is agent, admin or ordinary user working on it.

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.

Together, the frontend and backend components, along with socket.io, Express.js, WebRTC API, and MongoDB, form a comprehensive technical architecture for our video conference app. This architecture enables real-time communication, efficient data exchange, and seamless integration, ensuring a smooth and immersive video conferencing experience for all users.

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.

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.

• Download: https://nodejs.org/en/download/

•Installation instruction : https://nodejs.org/en/download/package-manager/

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 hotel and booking information. Install MongoDB locally or use a cloud-based MongoDB service.

• Download: https://www.mongodb.com/try/download/community

• Installation instructions:https://docs.mongodb.com/manual/installation/

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 : **npm install express**

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.
Follow the installation guide: https://reactjs.org/docs/create-a-new-react-app.html

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:

https://www.section.io/engineering-education/nodejs- mongoosejs-mongodb/

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.

Git:

Download and installation instructions can be found at:

https://git-scm.com/downloads

Development Environment:

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

- **Visual Studio Code: Download from https://code.visualstudio.com/download**

To run the existing Video Conference App project downloaded from GitHub:

Follow below steps:

Clone the Repository:

Open your terminal or command prompt.

Navigate to the directory where you want to store the e-commerce app.

Execute the following command to clone the repository:

git clone: https://github.com/awdhesh-student/complaint-registery.git

Install Dependencies:

• Navigate into the cloned repository directory:
cd complaint-registery

• Install the required dependencies by running the following commands:

cd frontend
npm install
cd ../backend
npm install

Start the Development Server:

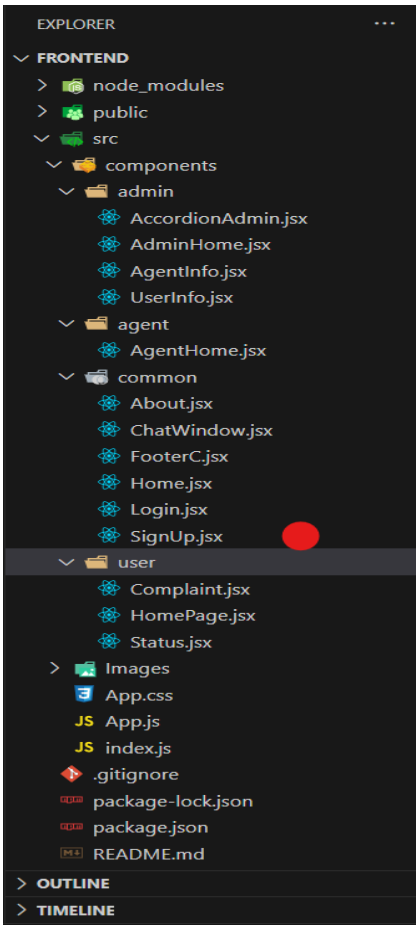
• To start the development server, execute the following command:
npm start

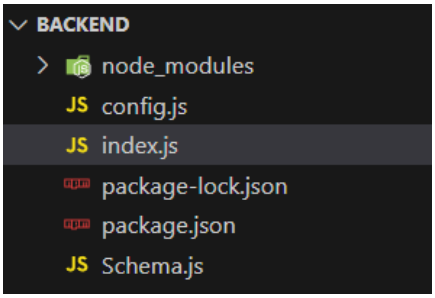
• The online complaint registration and management app will be accessible at

http://localhost:3000

You have successfully installed and set up the online complaint registration and management app on your local machine. You can now proceed with further customization, development, and testing as needed.

Project Structure :





The first image is of frontend part which is showing all the files and folders that have been used in UI development

The second image is of Backend part which is showing all the files and folders that have been used in backend development

Roles and Responsibility:

Ordinary (Customer/User):

Role:
Submit and manage complaints, communicate with assigned agents, and maintain personal profile information.

Responsibilities:

- Register and log in to the platform securely.
- Submit detailed complaints with supporting documents or images.
- Track complaint status and receive real-time notifications.
- Communicate with the assigned agent for issue resolution.
- Manage personal profile information such as name, contact details, and address.
- Provide feedback after complaint resolution.

Agent:

Role:
Handle complaints assigned by the admin, communicate with customers, and update the status of each complaint.

Responsibilities:

- Log in to access the agent dashboard.
- View and manage complaints assigned by the admin.
- Review complaint details and investigate the issue.
- Communicate with the customer via built-in messaging/chat.
- Provide timely updates and mark complaint statuses (e.g., In Progress, Resolved).
- Ensure clear and courteous communication to resolve issues effectively.

Admin:

Role:
Oversee the entire platform operation, manage users and agents, assign complaints, and enforce system policies.

Responsibilities:

- Monitor all registered complaints and their resolution status.
- Assign complaints to agents based on workload and expertise.
- Manage registration, login, and profile details of users and agents.
- Enforce platform policies, data privacy, and security compliance.

- Oversee real-time platform performance and user activity.
- Address system-wide issues and implement improvements for better functionality and user experience

Project Flow:-

Before starting to work on this project, let's see the demo.

Project
Demo
https://drive.google.com/file/d/1YwXaHRBZJL_V7dcEK8SOmtPWZasAxccm/view?usp=drive_link
Use the code in:
https://github.com/awdhesh-student/complaint-registery.git
or follow the videos below for better understanding.

Milestone 1: Project Setup and Configuration:

1. Install required tools and software:

- Node.js.
- MongoDB.
- Create-react-app.

2. Create project folders and files:

- Client folders
- Server fodlders

3. Install Packages:

Frontend Packages

- React Js.
- Material UI.
- Bootstrap.
- Axios.

Backend Packages

- Node.js.
- MongoDB.
- Bcrypt.
- Body-parser

After the installation of all the libraries, the package.json files for the frontend looks like the one mentioned below.

```
{
  "name": "task1",
  "version": "0.1.0",
  "proxy": "http://localhost:8000",
  "private": true,
  "dependencies": {
    "@emotion/react": "^11.11.1",
    "@emotion/styled": "^11.11.0",
    "@testing-library/jest-dom": "^5.16.5",
    "@testing-library/react": "^13.4.0",
    "@testing-library/user-event": "^13.5.0",
    "axios": "^1.4.0",
    "bootstrap": "^5.2.3",
    "mdb-react-ui-kit": "^6.1.0",
    "react": "^18.2.0",
    "react-bootstrap": "^2.7.4",
    "react-dom": "^18.2.0",
    "react-router-dom": "^6.11.2",
    "react-scripts": "5.0.1",
    "web-vitals": "^2.1.4"
  },
  "scripts": {
    "start": "react-scripts start",
    "build": "react-scripts build",
    "test": "react-scripts test",
    "eject": "react-scripts eject"
  },
  "eslintConfig": {
    "extends": [
      "react-app",
      "react-app/jest"
    ]
  },
  "browserslist": {
    "production": [
      ">0.2%",
      "not dead",
      "not op_mini all"
    ],
    "development": [
      "last 1 chrome version",
      "last 1 firefox version",
      "last 1 safari version"
    ]
  }
}
```

After the installation of all the libraries, the package.json files for the backend looks like the one mentioned below.

```
{
  "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"
  }
}
```

Milestone 2: Backend Development:

Setup Project Structure:

- Create a new directory for your project and set up a package.json file using npm init command.
- Install necessary dependencies such as Express.js, Mongoose, and other required packages.

Set Up Project Structure:

- Create a new directory for your project and set up a package.json file using npm init command.
- Install necessary dependencies such as Express.js, Mongoose, and other required packages.

Create Express.js Server:

- Set up an Express.js server to handle HTTP requests and serve API endpoints.
- Configure middleware such as body-parser for parsing request bodies and cors for handling cross-origin requests.

Define API Routes:

- Create separate route files for different API functionalities such as authentication, stock actions, and transactions.
- Implement route handlers using Express.js to handle requests and interact with the database.

Implement Data Models:

- Define Mongoose schemas for the different data entities like products, users, and orders.
- Create corresponding Mongoose models to interact with the MongoDB database.
- Implement CRUD operations (Create, Read, Update, Delete) for each model to perform database operations.

User Authentication:

- Implement user authentication using strategies like JSON Web Tokens (JWT) or session-based authentication.
- Create routes and middleware for user registration, login, and logout.
- Set up authentication middleware to protect routes that require user authentication.

Handle new transactions:

- Allow users to make transactions to other users using the user's account id.
- Update the transactions and account balance dynamically in real-time.

Admin Functionality:

- Implement routes and controllers specific to admin functionalities such as fetching all the data regarding users, transactions, stocks and orders.

Error Handling:

- Implement error handling middleware to catch and handle any errors that occur during the API requests.
- Return appropriate error responses with relevant error messages and HTTP status codes.

Milestone 3: Database Development:

User Schema:

- The user schema defines the structure of user data stored in the database. It includes fields such as name, email, password, phone, and userType.
- Each user must provide a name, email, password, phone number, and userType (e.g., customer, agent, admin).
- User data is stored in the "user_Schema" collection in the MongoDB database.

Complaint Schema:

- The complaint schema specifies the format of complaint data registered by users.
- It contains fields like userId, name, address, city, state, pincode, comment, and status.

- Complaints are associated with users through the userId field, and each complaint must have a name, address, city, state, pincode, comment, and status.
- Complaint data is stored in the "complaint_schema" collection in the MongoDB database.

Assigned Complaint Schema:

- The assigned complaint schema defines how complaints are assigned to agents for resolution.
- It includes fields such as agentId, complaintId, status, and agentName.
- Each assigned complaint is linked to a specific agent (identified by agentId) and complaint (identified by complaintId).
- The status field indicates the current status of the assigned complaint.
- Assigned complaint data is stored in the "assigned_complaint" collection in the MongoDB database.

Chat Window Schema:

- The chat window schema governs the structure of messages exchanged between users and agents regarding specific complaints.
- It comprises fields like name, message, and complaintId.
- Messages are associated with a complaint through the complaintId field, allowing for easy tracking and retrieval of chat history for each complaint.
- Message data is stored in the "message" collection in the MongoDB database

Milestone 4: Frontend Development:

Setup React Application:

Bringing Customer Care Registry to life involves a three-step development process. First, a solid foundation is built using React.js. This includes creating the initial application structure, installing necessary libraries, and organizing the project files for efficient development. Next, the user interface (UI) comes to life. To start the development process for the frontend, follow the below steps.

- Install required libraries.
- Create the structure directories.

Design UI components:

Reusable components will be created for all the interactive elements you'll see on screen, from stock listings and charts to buttons and user profiles. Next, we'll implement a layout and styling scheme to define the overall look and feel of the application. This ensures a visually-appealing and intuitive interface. Finally, a navigation system will be integrated, allowing you to effortlessly explore different sections of Customer Care Registry, like making specific complaints or managing your Product complaints.

Implement frontend logic:

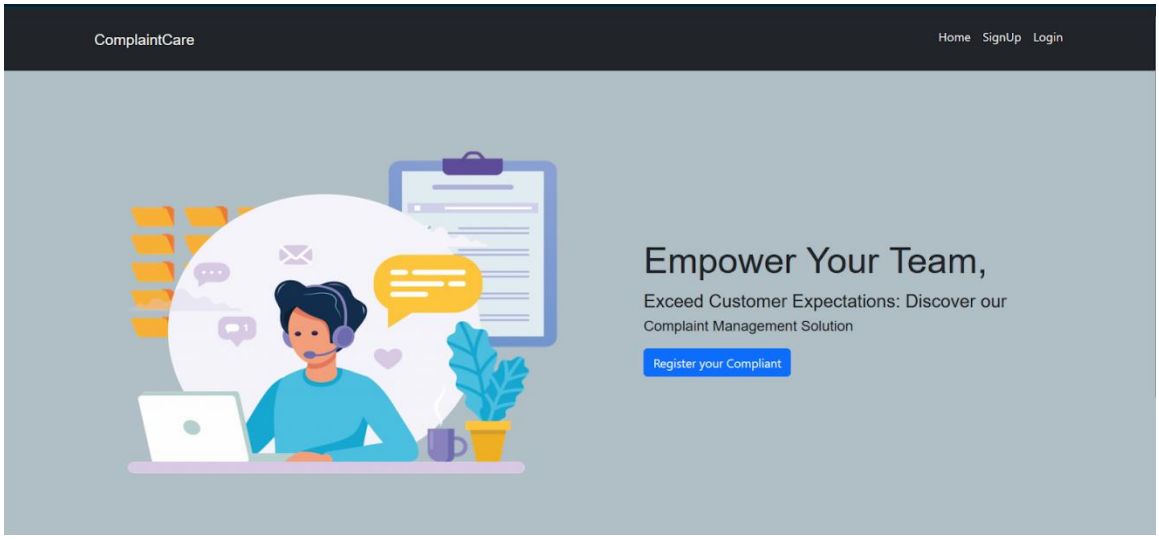
In the final leg of the frontend development, we'll bridge the gap between the visual interface and the underlying data. It involves the below stages.

- Integration with API endpoints.
- Implement data binding.

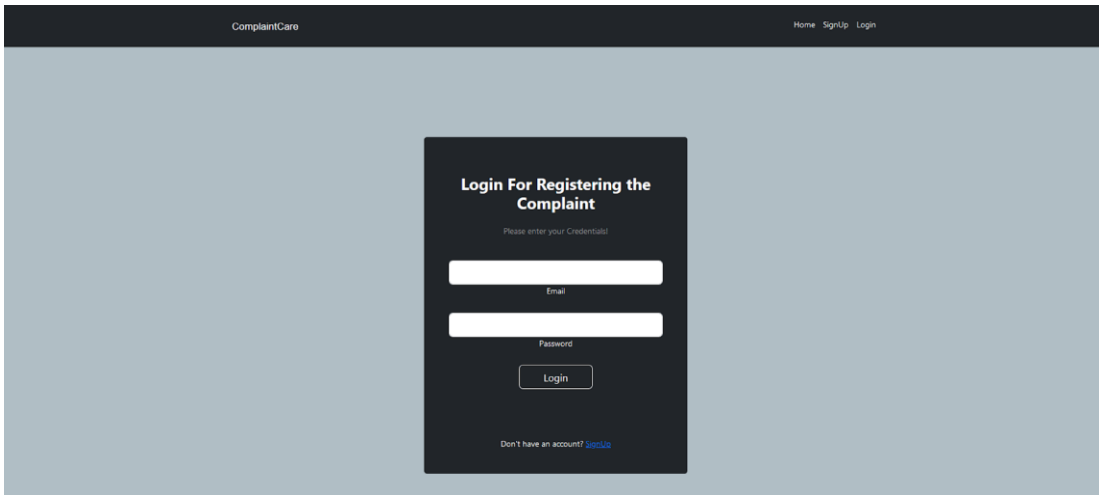
Milestone 5: Project Implementation:

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.

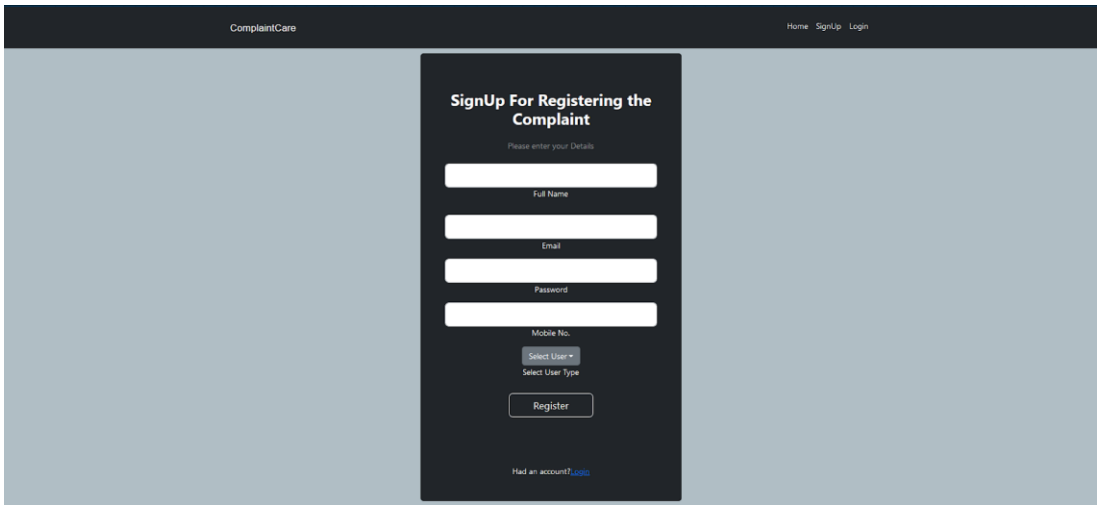
Landing page:-



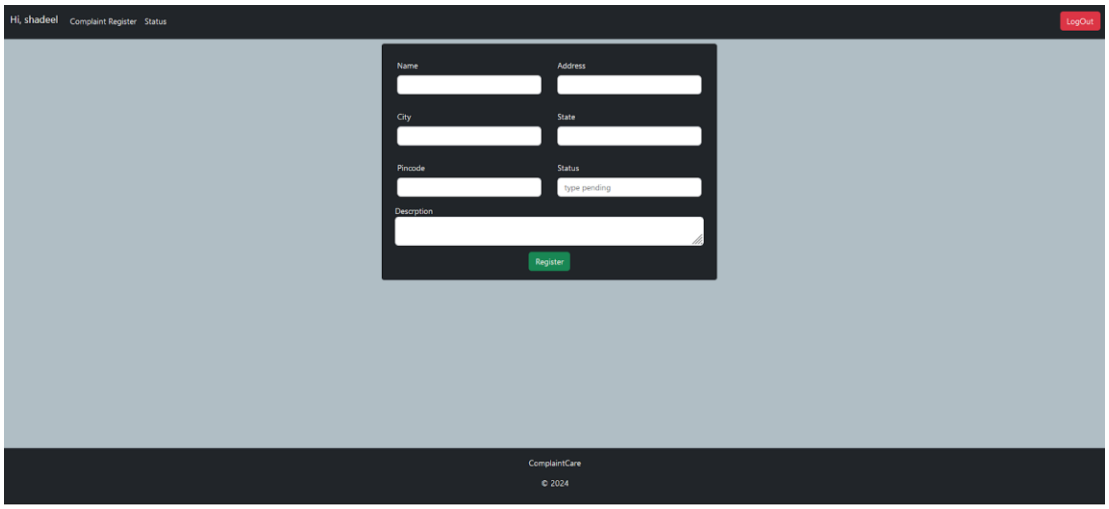
Login Page:-



Registration Page:-

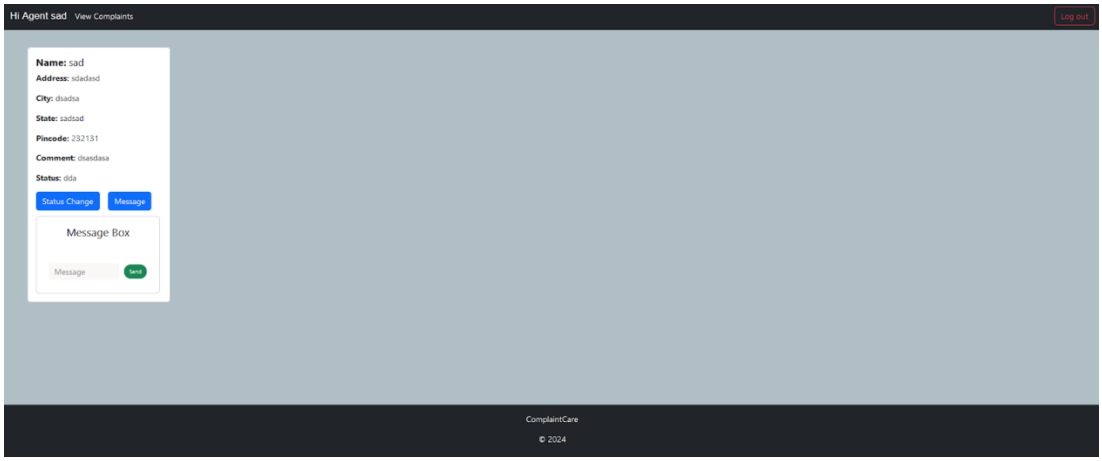


Common Dashboard For Complaint:-



Admin Dashboard :-

Agent Dashboard :-



Conclusion

The Online Complaint Registration and Management System presents a comprehensive and efficient solution for addressing customer grievances in a structured, secure, and user-friendly manner. By integrating modern web technologies like React.js, Node.js, Express.js, and MongoDB, the platform enables seamless complaint submission, tracking, and resolution for users while empowering agents and administrators to manage and resolve issues effectively. With features such as real-time notifications, intelligent complaint routing, secure user communication, and role-based access control, the system not only enhances user satisfaction but also promotes operational transparency and accountability. Overall, this platform stands as a scalable and reliable model for any organization seeking to improve its customer support and grievance redressal process.

For any further doubts or help, please consider the code from the google drive,

https://drive.google.com/drive/folders/1uGwb-keRJCab88xNFCD4EoXzZsDChZe3

The demo of the app is available at:

https://drive.google.com/drive/folders/1uGwb-keRJCab88xNFCD4EoXzZsDChZe3

*** Happy Coding ***