ONLINE COMPLAINT REGISTRATION AND MANAGEMENT SYSTEM

TEAM MEMBERS:

1. Geesala Ramya   -   geesalaramya6@gmail.com

2. Kasireddy Jashwanth   -   jashwanthvarma07@gmail.com

3. Koppisetti Rohith   -   Bunnyrohith619@gmail.com

4. Nalam Ramya   -   Ramyanalam2004@gmail.com

INTRODUCTION

The **Online Complaint Registration and Management System** is a web-based platform designed to streamline the process of lodging, tracking, and resolving complaints in an efficient and transparent manner. It serves as a digital bridge between the public and the authorities or organization responsible for handling grievances.

Traditionally, complaint registration has been a time-consuming and manual process, often leading to delays, loss of records, and lack of accountability. This system addresses those challenges by offering users a simple and intuitive interface to file complaints online, monitor their status in real-time, and receive timely updates on actions taken.

The system is equipped with features such as user authentication, category-wise complaint filing, automated complaint tracking, priority tagging, and administrative dashboards. It ensures that every complaint is properly recorded, routed to the appropriate department, and handled within a stipulated timeframe.

This project can be deployed in various domains such as municipal corporations, educational institutions, customer service departments, and more — wherever grievance redressal is essential.

In summary, the Online Complaint Registration and Management System improves user satisfaction, enhances operational transparency, and fosters better communication between complainants and authorities.

PROJECT OVERVIEW

Purpose:

The purpose of the Online Complaint Registration and Management System is to provide a digital platform where users can conveniently lodge complaints, report issues, or raise concerns, and receive timely resolutions. This system aims to eliminate the inefficiencies of manual complaint handling by offering an automated, organized, and user-friendly solution.

Goals of the Project:

- To enable users to register complaints from anywhere, at any time.

- To ensure systematic recording, categorization, and tracking of complaints.

- To provide a transparent mechanism for monitoring the status and progress of each complaint.

- To assist administrators in managing, prioritizing, and resolving issues more efficiently.

- To enhance user satisfaction and trust by ensuring timely feedback and resolution.

By implementing this system, organizations can improve their grievance redressal process, strengthen communication with users, and maintain better accountability across departments.

Features

- **User Authentication** – Secure login and registration functionality for users and admins.

- **Complaint Submission** – Users can submit complaints through a simple online form.

- **Complaint Tracking** – Track the status of submitted complaints in real-time.

- **Search & Filter** – Easily find complaints based on ID, category, or date.

- **Admin Dashboard** – Admins can view, manage, and update complaint statuses.

TECHNICAL ARCHITECTURE
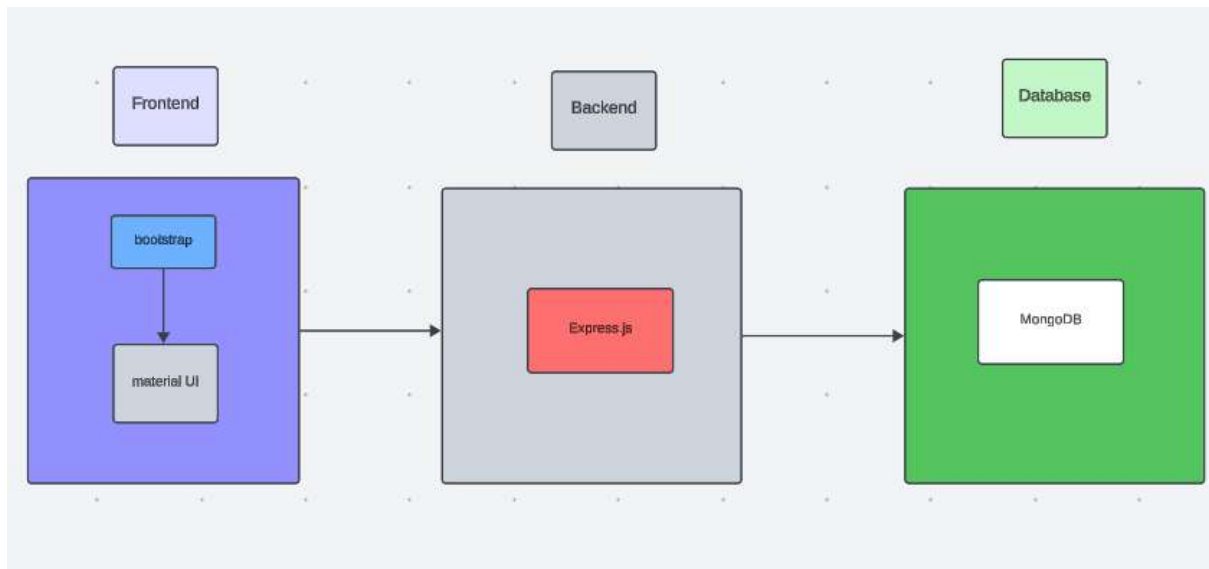
The technical architecture of our online complaint registration and management app follows a client-server model, where the front end serves as the client and the backend acts as the server. The front end encompasses the user interface and presentation and incorporates the Axios library to connect with the backend easily by using RESTful Apis.

The front end utilizes the bootstrap and material UI library to establish a real-time and better UI experience for any user whether it is an 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, complaints registration, etc. It ensures reliable and quick access to the necessary information during registration of users 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
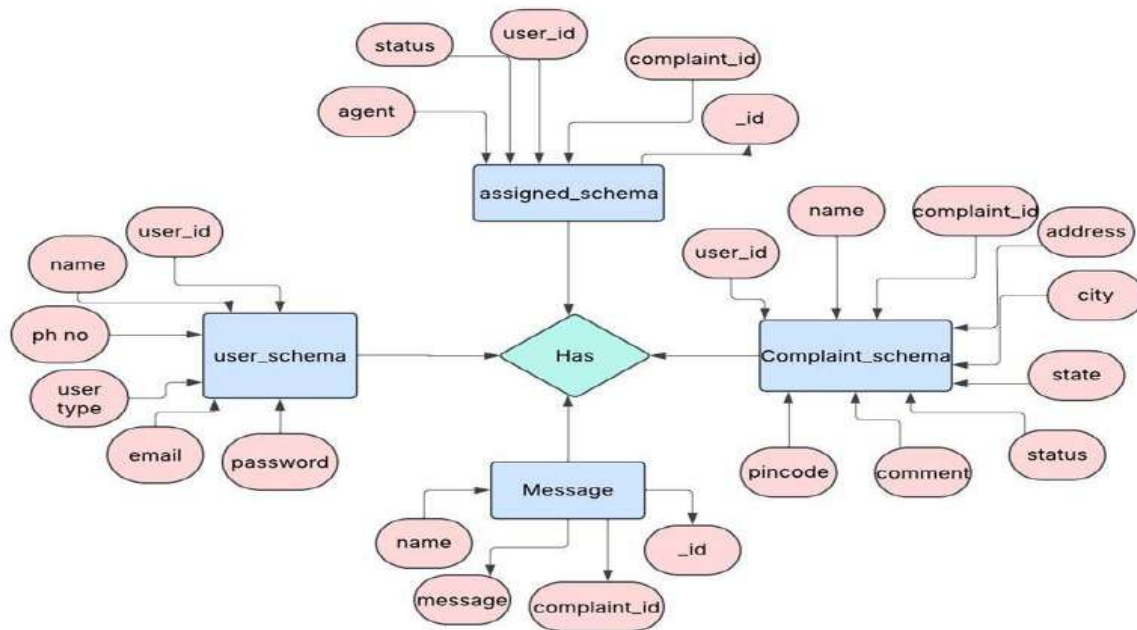
This is the ER diagram of the project which shows the relationship between the user and the agent. It shows how users who have required fields can raise a complaint by filling required fields.

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 a chat window that follows the message schema which uses user Id and complaint Id from other schemas.

PRE-REQUISITES:

Here are the key prerequisites for developing a full-stack application using Node.js, Express.js, MongoDB, and 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 instructions: https://nodejs.org/en/download/package-manager/

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

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

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

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, the status of the complaints, and user interfaces for the admin dashboard.
To make better UI we have also used some libraries like Material UI and Bootstrap.

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 the 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/ramyageesala/ResolveNow-Your-Platform-for-Online-Complaints

Install Dependencies:

• Navigate into the cloned repository directory:
      cd complaint-register
• Install the required dependencies by running the following commands:
      cd frontend
      npm install
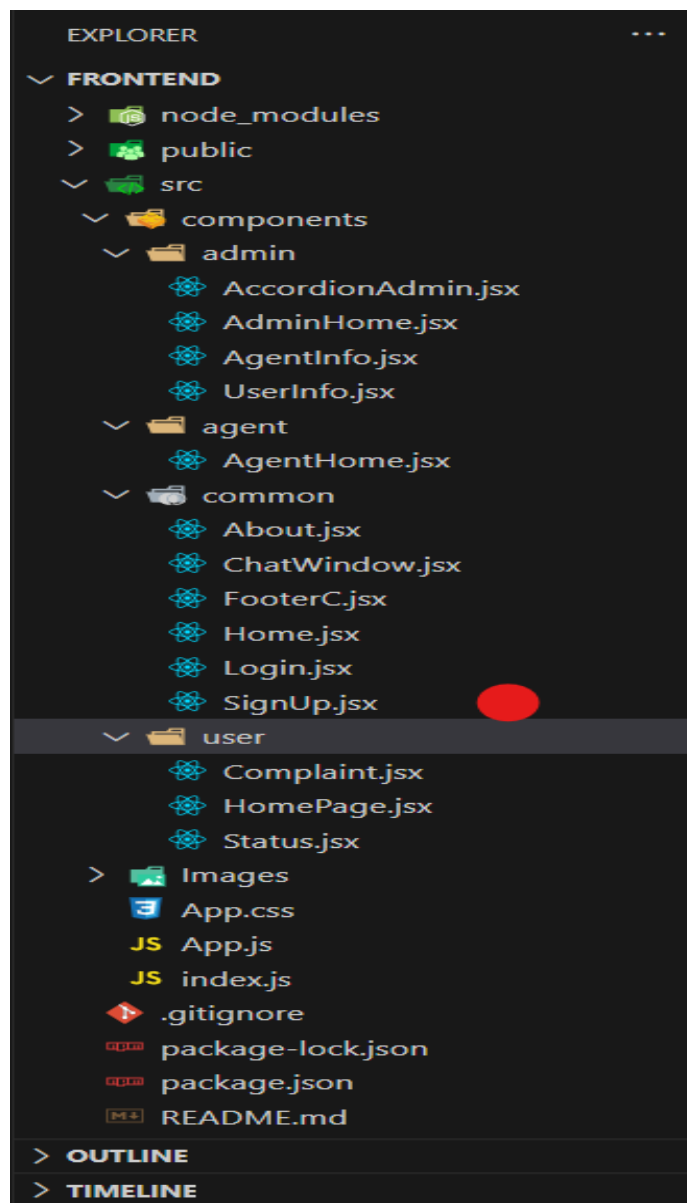   cd ../backend
      npm install
Start the Development Server:
      • To start the development server, execute the following command:
      npm start
• The online complaint registration and management app will be accessible at
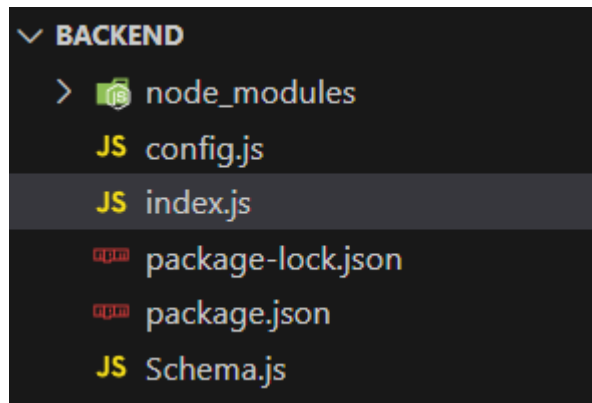http://localhost:3000

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:

FRONTEND:

BACKEND:

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

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

APPLICATION FLOW:

Online Complaint Registration and Management System

1. **Customer/Ordinary User:**
   - **Role:** Create and manage complaints, interact with agents, and manage profile information.
   - **Flow:**
   1. **Registration and Login:**
      - Create an account by providing the necessary information such as email and password.
      - Log in using the registered credentials.
   2. **Complaint Submission:**
      - Fill out the complaint form with details of the issue, including description, contact information, and relevant attachments.
      - Submit the complaint for processing.
   3. **Status Tracking:**
      - View the status of submitted complaints in the dashboard or status section.
      - Receive real-time updates on the progress of complaints.
   4. **Interaction with Agents:**
      - Connect with assigned agents directly using the built-in messaging feature.
      - Discuss complaints further and provide additional information or clarification.
   5. **Profile Management:**
      - Manage personal profile information, including details and addresses.
2. **Agent:**
   - **Role:** Manage complaints assigned by the admin, communicate with customers, and update complaint statuses.
   - **Flow:**

1. **Registration and Login:**
    - Create an account using email and password.
    - Log in using the registered credentials.
2. **Complaint Management:**
    - Access the dashboard to view and manage complaints assigned by the admin.
    - Communicate with customers regarding their complaints through the chat window.
3. **Status Update:**
    - Change the status of complaints based on resolution or progress.
    - Provide updates to customers regarding the status of their complaints.
4. **Customer Interaction:**
    - Respond to inquiries, resolve issues, and address feedback from customers.
3. **Admin:**
    - **Role:** Oversee the overall operation of the complaint registration platform, manage complaints, users, and agents, and enforce platform policies.
    - **Flow:**
1. **Management and Monitoring:**
    - Monitor and moderate all complaints submitted by users.
    - Assign complaints to agents based on workload and expertise.
2. **Complaint Assignment:**
    - Assign complaints to the desired agents for resolution.
    - Ensure timely and efficient handling of complaints.

3. **User and Agent Management:**
    - Manage user and agent accounts, including registration, login, and profile information.
    - Enforce platform policies, terms of service, and privacy regulations.
4. **Continuous Improvement:**
    - Implement measures to improve the platform's functionality, user experience, and security measures.
    - Address any issues or concerns raised by users or agents for better service delivery.

Project Flow:

Before starting to work on this project, let's see the demo.
Project demo:
https://drive.google.com/file/d/1n5Yt0L0871n9IDbF_O5zPBzHqMnBorLW/view?usp=sharing
Use the code at: https://github.com/ramyageesala/ONLINE-COMPLAINT-REGISTRATION-AND-MANAGEMENT-SYSTEM.git
or follow the videos below for a better understanding.

Milestone 1:
Project Setup and Configuration:

1. **Create project folders and files:**

Now, firstly create the folders for frontend and backend to write the respective code and install the essential libraries.

- Client folders.

- Server folders

2. **Install required tools and software:**

For the backend to function well, we use the libraries mentioned in the prerequisites. Those libraries include

- Node.js.

- MongoDB.

- Bcrypt

- Body-parser

Also, for the frontend we use the libraries such as

- React Js.

- Material UI

- Bootstrap

- Axios

After the installation of all the libraries, the package.json files for the front end look like the one mentioned below.

```json
{
  "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 look like the one mentioned below.

```
 1    {
 2      "name": "backend",
 3      "version": "1.0.0",
 4      "description": "",
 5      "main": "index.js",
 6      "scripts": {
 7        "start": "nodemon index.js"
 8      },
 9      "keywords": [],
10      "author": "",
11      "license": "ISC",
12      "dependencies": {
13        "bcrypt": "^5.1.0",
14        "cors": "^2.8.5",
15        "express": "^4.18.2",
16        "express-session": "^1.17.3",
17        "mongoose": "^7.1.1",
18        "nodemon": "^2.0.22"
19      }
20    }
```

Milestone 2:
Backend Development:

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

- **Set Up Project Structure:**
  - o Create a new directory for your project and set up a package.json file using npm init command.
  - o Install necessary dependencies such as Express.js, Mongoose, and other required packages.
- **Create Express.js Server:**
  - o Set up an Express.js server to handle HTTP requests and serve API endpoints.
  - o Configure middleware such as body-parser for parsing request bodies and cors for handling cross-origin requests.
- **Define API Routes:**
  - o Create separate route files for different API functionalities such as authentication, creating, assigning complaints, and chat window.
  - o Implement route handlers using Express.js to handle requests and interact with the database.

- **Implement Data Models:**

- 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).
  - Create routes and middleware for user registration, login, and logout.
  - Set up authentication middleware to protect routes that require user authentication.

- **Admin Functionality:**
  - Implement routes and controllers specific to admin functionalities such as fetching all the data regarding users, complaints, agents.

- **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.

Reference video for backend code:

https://drive.google.com/file/d/19UO9XUotRVeJASTdmNIIPaYzrhlrkTAx/view?usp=drive_link

Milestone 3:

Database Development

1. **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.

2. **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.

o Complaints are associated with users through the user Id field, and each complaint must have a name, address, city, state, pincode, comment, and status.

o Complaint data is stored in the "complaint_ schema" collection in the MongoDB database.

3. **Assigned Complaint Schema:**

o The assigned complaint schema defines how complaints are assigned to agents for resolution.

o It includes fields such as agent Id, complaint Id, status, and agent Name.

o Each assigned complaint is linked to a specific agent (identified by agent Id) and complaint (identified by complaint Id).

o The status field indicates the current status of the assigned complaint.

o Assigned complaint data is stored in the "assigned_ complaint" collection in the MongoDB database.

4. **Chat Window Schema:**

o The chat window schema governs the structure of messages exchanged between users and agents regarding specific complaints.

o It comprises fields like name, message, and complaint Id.

o Messages are associated with a complaint through the complaint Id field, allowing for easy tracking and retrieval of chat history for each complaint.

o Message data is stored in the "message" collection in the MongoDB database.

Reference video for Database code:

https://drive.google.com/file/d/1CQil5KzGnPvkVOPWTLP0h-Bu2bXhq7A3/view

Milestone 4:

Frontend Development:

1. **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 front end, follow the below steps.

- Install required libraries.

- Create the structure directories.

2.Design UI components:

Reusable components will be created for all the interactive elements you'll see on the 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 the Customer Care Registry, like making specific complaints or managing your Product complaints.

3.Implement frontend logic:

In the final leg of the front-end development, we'll bridge the gap between the visual interface and the underlying data. It involves the following stages.

- Integration with API endpoints.
- Implement data binding.

Reference video for frontend code:
https://drive.google.com/file/d/1k5Ww7arbOk-uWC3_ebx7X0q6dD-u-wQg/view?usp=drive_link

Milestone 5:

Project Implementation:

On completing the development part, we then ran 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 provided below.

- Landing Page
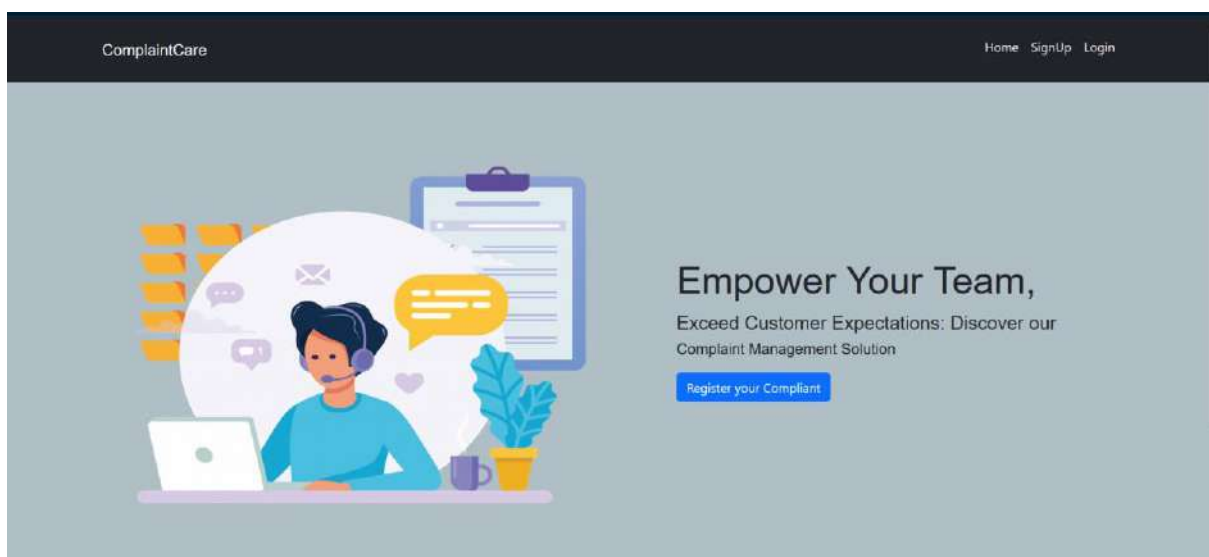
- Login Page

- Registration Page

- Common Dashboard For Complaint

- Admin Dashboard

- Agent Dashboard

The demo of the app is available at:

https://drive.google.com/file/d/1n5Yt0L0871n9IDbF_O5zPBzHqMnBorLW/view?usp=sharing