Full Stack Development with MERN

Project Documentation

1. Introduction

- **Project Title:** Resolve Now: Your Platform for Online Complaints
- **Team Members:** List of Team Members and their Roles.

Team Members	Roles
Jetti Jaya Keerthi	Team Leader & Backend Lead
Gangineni Surekha	Frontend Developer
Garikapati Tulasi	Admin Panel & Agent Dashboard Developer
Gera Bhumika	Testing, Documentation & Deployment Engineer

2. Project Overview

- **Purpose:**
  The *Online Complaint Registration and Management System* is designed to streamline the process of lodging and resolving complaints by citizens. The goal is to provide a centralized digital platform where users can submit complaints, track their status, and receive timely updates, while enabling administrators to manage, monitor, and resolve those complaints efficiently. This system aims to replace traditional manual processes with a responsive, secure, and user-friendly interface that enhances transparency, accountability, and ease of use for both citizens and government/organization authorities.

- **Features:**
  1. **User Authentication –** Secure Registration and login using JWT – based authentication.
  2. **Role Management –** Support for both regular users and admin roles.
  3. **Complaint Registration –** Users can file complaints with titles, descriptions, and categories.
  4. **Complaint Dashboard –** Users can view the status of their complaints.
  5. **Admin Panel –** Admins can view, update, and delete any complaints.
  6. **Status Tracking –** Complaints go through various statuses like "Pending", "In Progress", and "Resolved".
  7. **Responsive UI –** Built with React, Material UI, and Bootstrap for a mobile- friendly interface.
  8. **MongoDB Integration –** Persistent data storage with flexible schema for user and complaint data.
  9. **RESTful API –** Node.js and Express.js backend providing well-structured endpoints for frontend communication.

3. Architecture

Frontend (React.js):

- Built using React.js to provide a responsive and dynamic user interface.
- Utilizes functional components and React Hooks for efficient state management.
- Routing is managed using React Router DOM for smooth navigation across pages (login, register, dashboard, admin panel).
- Styled using Bootstrap and Material UI to maintain a modern and consistent look and feel.
- Handles API calls via Axios to communicate with the backend.

Backend (Node.js + Express.js):

- Developed with Node.js and the Express.js framework to handle HTTP requests and implement RESTFUL API's.
- Contains routes for user registration, login, complaint management, and admin actions.
- Implements JWT (JSON Web Token) based middleware for authentication and authorization.
- Structured in a modular way with separate folders for routes, controllers, middleware, and models.

Database(MongoDB):

- MongoDB is used as the NoSQL database to store data such as user information and complaint records.
- Managed using Mongoose, which provides schema definitions and object data modelling (ODM).
- Schema design includes:
  1. **User Schema:** Name, email, password (hashed), phone, userType (user/admin).
  2. **Complaint Schema:** Title, description, status, user reference, timestamps

4. Setup Instructions

- **Prerequisites:**

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

Front-end Framework:

- Utilize React.js to build the user-facing part of the application, including complaint registration, complaint status tracking, and user interfaces for the admin dashboard.
- **UI Libraries:** For creating a better user interface, we have also used 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 be used to host your repository.
- **Git Installation:** Download and installation instructions can be found at: https://git-scm.com/downloads
- **Installation:**
  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

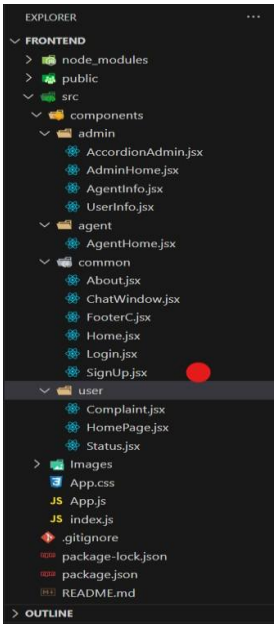
  npm install

  cd ../backend

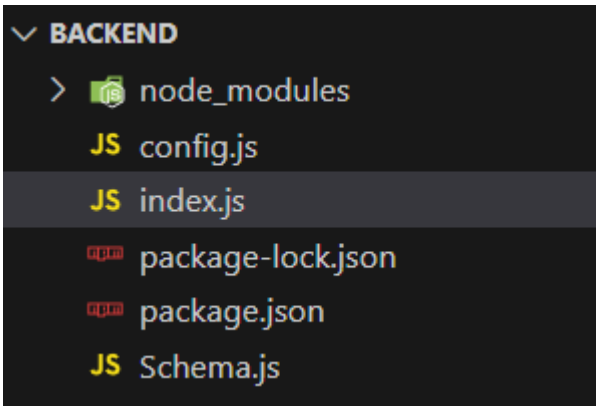
  npm install
- Start the Development Server:
- To start the development server, execute the following command:
- npm start
- The online complaint registration and management app will be **accessible at http://localhost:3000**

5. Folder Structure

- **Client: Describe the structure of the React frontend.**



- **Server:** Explain the organization of the Node.js backend.



6. Running the Application

Provide commands to start the frontend and backend servers locally.

- **Frontend:** npm start in the client directory.
- **Backend:** npm start in the server directory.

7. API Documentation

- The backend exposes RESTful API endpoints to manage complaints, users, and authentication. Below are key endpoints:
- **User Routes**

  POST /api/register – Register a new user

  Body: { name, email, password, phone, userType }

  POST /api/login – Authenticate a user

  Body: { email, password }
- **Complaint Routes**

  POST /api/complaints – Register a new complaint

  Headers: { Authorization: Bearer <token> }

  Body: { title, description, category }

  GET /api/complaints – Get all complaints (admin only)

  GET /api/complaints/user – Get complaints of logged-in user

  PUT /api/complaints/:id – Update complaint status

  DELETE /api/complaints/:id – Delete a complaint (admin)
- **Example response for POST /api/complaints:**

  {
  "message": "Complaint registered successfully", "complaint": {
  "_id": "60f...",
  "title": "Broken street light", "status": "Pending"
  }
  }

8. Authentication

- We use **JWT (JSON Web Token)** based authentication.
- After login, a JWT is issued and stored in localStorage.
- Protected routes require the token in the Authorization header.

- Middleware on the backend validates the token to allow access to resources.
- **Roles Supported:**

  User: Can register/login and create/view their complaints.

  Admin: Can view/update/delete all complaints.

9. User Interface

- The user interface is built using **React.js** and styled with **Bootstrap & Material-UI**.
- Key UI Components:

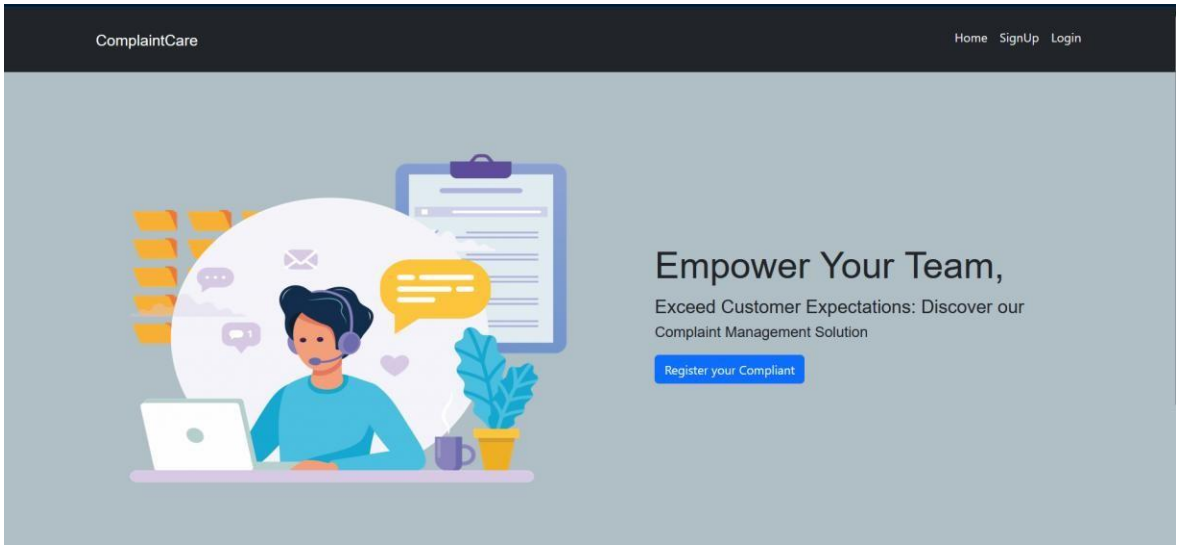
  Register/Login Forms

  Complaint Submission Form

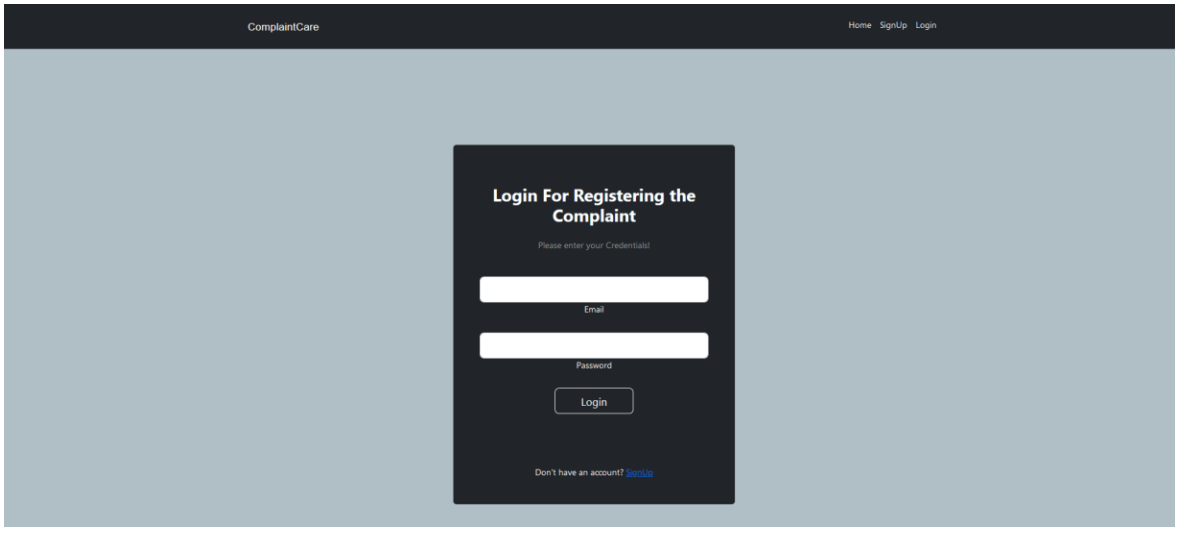
  User Complaint Dashboard

  Admin Dashboard for Complaint Management

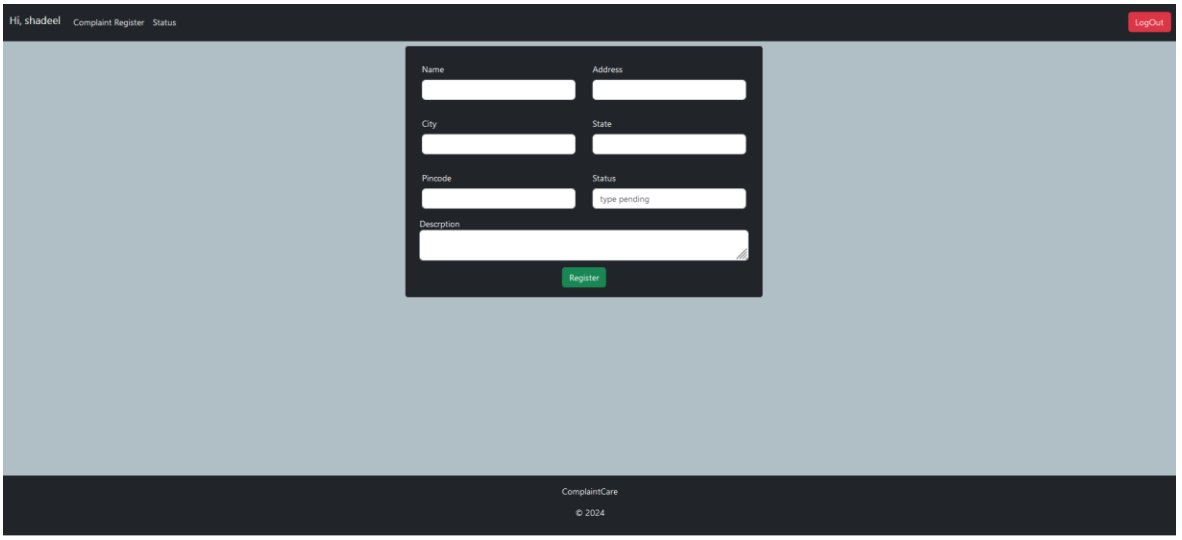
- Home Page



- Login Page:



- Common Dashboard For Complaint



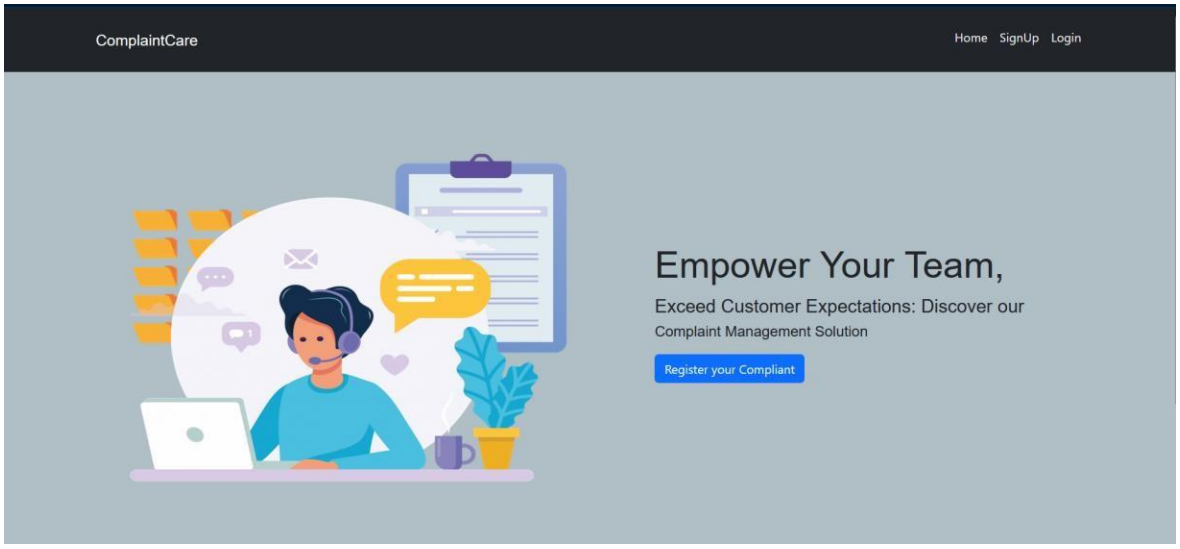
- Admin Dashboard

10. Testing

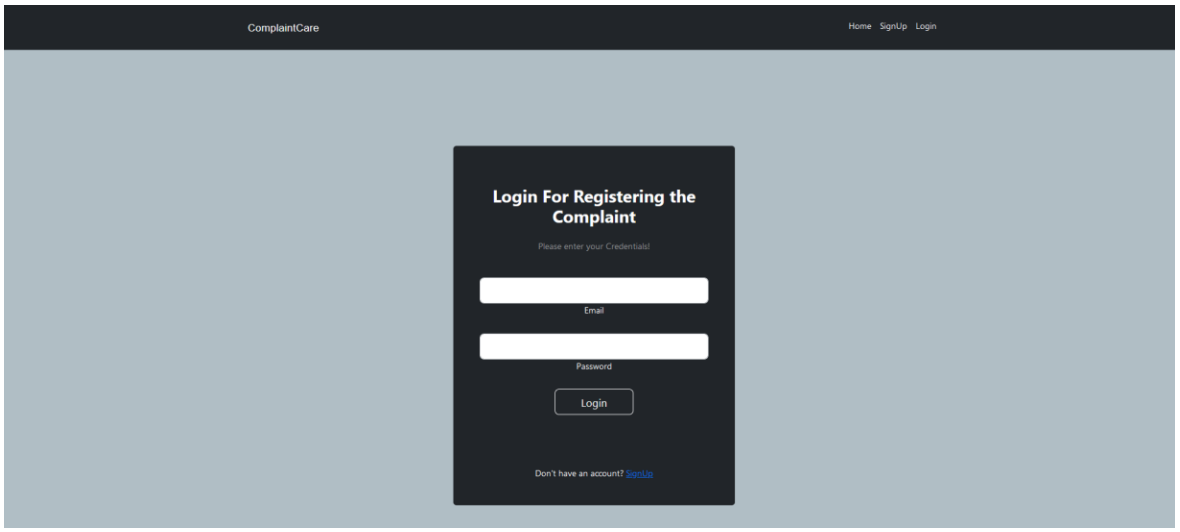
- We followed manual and component-level testing for both frontend and backend.
- **Backend Testing:** Used Postman to test REST APIs for user and complaint modules.
- **Frontend Testing:** React components tested with sample data and checked for state changes and UI response.

11. Screenshots or Demo

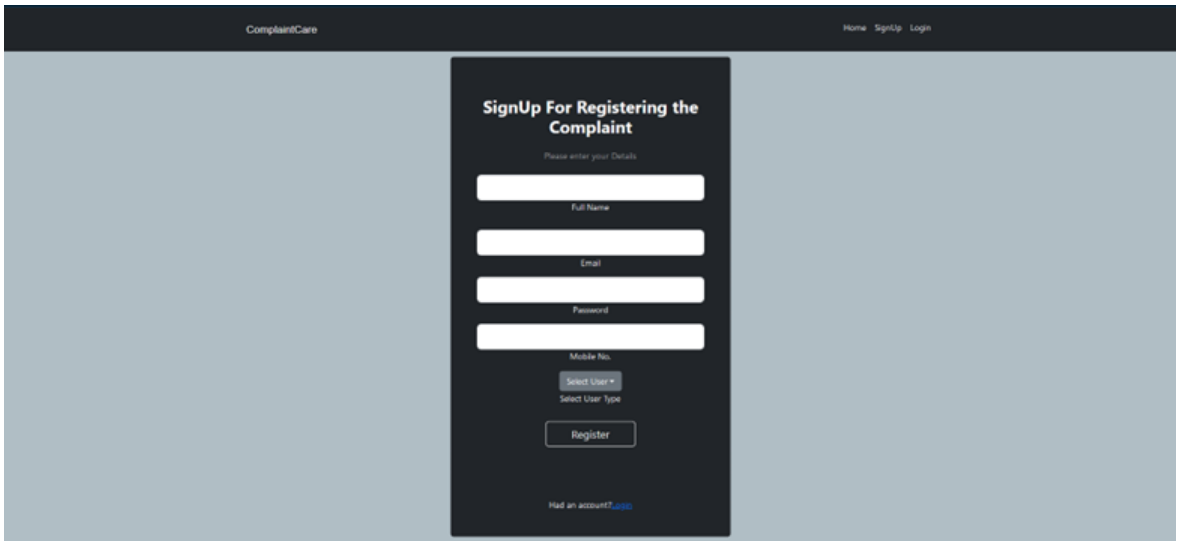
- Home Page



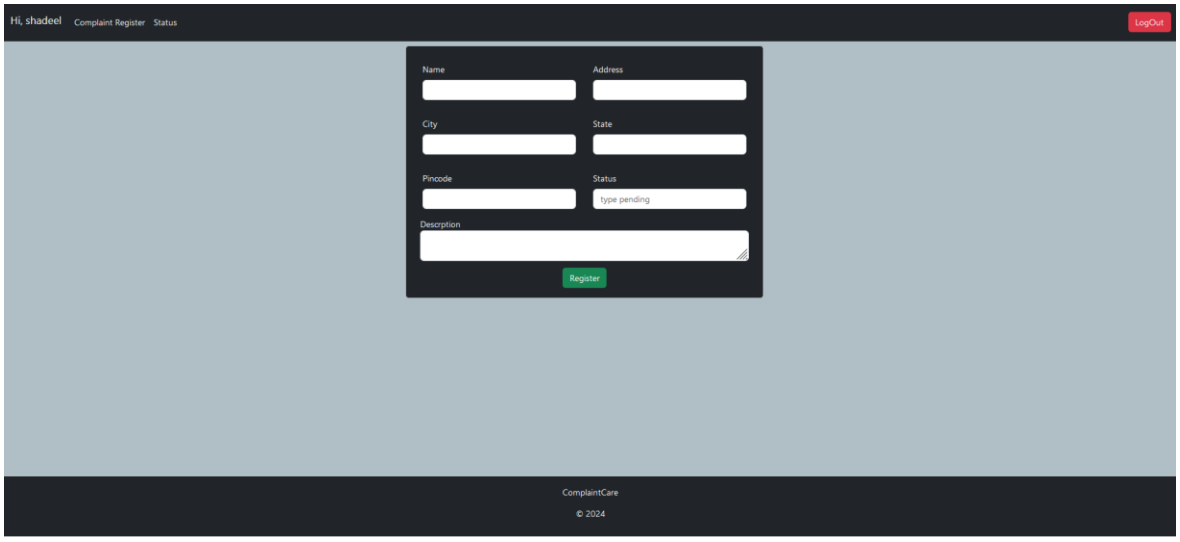
- Login Page



- Registration Page

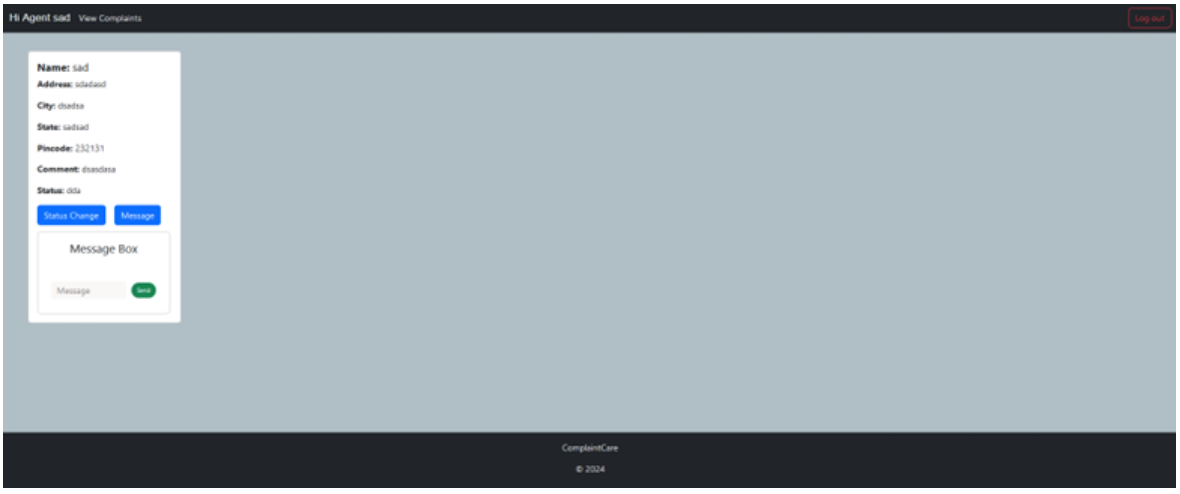


- Common Dashboard For Complaint



- Admin Dashboard

- Agent Dashboard



12. Known Issues

- Session does not persist across hard refresh without re-login.
- No email verification after registration.
- Some admin actions lack confirmation prompts.
- Error handling on frontend needs refinement.

13. Future Enhancements

- Add **email/SMS notifications** for complaint updates.
- Integrate **chat support** or **real-time updates** using Socket.IO.
- Role-based dashboard with analytics for complaint trends.
- Mobile-responsive PWA version for users.
- Implement advanced search/filter by status, date, or category.