Your Platform for Online Complaints

INTRODUCTION:

The Online Complaint Registration and Management System is a web-based full-stack application designed to streamline the complaint lodging, monitoring, and resolution process. It enables ordinary users, agents, and admins to interact efficiently in resolving issues, offering features like complaint tracking, real-time messaging, and status updates. The system supports secure login, role-based access, and dynamic dashboards tailored to each user type.

DESCRIPTION:

This digital platform simplifies the handling of complaints by offering users a structured interface to submit their issues, communicate with responsible personnel, and track progress in real-time. The frontend is developed in React.js, while the backend is powered by Node.js and Express.js, with MongoDB used as the database.

Users can register, log in, file complaints, and track updates. Admins oversee all operations and assign complaints to agents, while agents manage and respond to the complaints via an integrated messaging system.

Features:

1. User Authentication:

- Secure registration and login system using bcrypt for password hashing.
- Role-based access for users, agents, and admins.

2 Complaint Submission:

- Users submit complaints with full details (name, address, city, state, pin code, and description).
- Each complaint is linked to the user and stored with a default status of "Pending".

3. Complaint Tracking:

- Users can view the status of their complaints from their dashboard.
- Admins and agents have access to view and update complaints assigned to them.

4. Messaging System:

- A built-in chat feature allows users and staff to exchange messages on a complaint.
- Messages include sender, senderName, content, and a timestamp.

5. User Authentication:

- Secure registration and login system using bcrypt for password hashing.
- Role-based access for users, agents, and admins.

6. Complaint Submission:

- Users submit complaints with full details (name, address, city, state, pin code, and description).
- Each complaint is linked to the user and stored with a default status of "Pending".

7. Complaint Tracking:

- Users can view the status of their complaints from their dashboard.

- Admins and agents have access to view and update complaints assigned to them.

8. Messaging System:

- A built-in chat feature allows users and staff to exchange messages on a complaint.
- Messages include sender, senderName, content, and a timestamp.

SCENARIO-BASED CASE STUDY:

Scenario: Olivia, a customer, recently encountered an issue with a product she purchased through an online retailer. Upon inspecting the item, she discovers a visible defect. In order to report the issue and seek a resolution, Olivia decides to file a complaint using the Online Complaint Registration.

1. User Registration and Onboarding:-

- **Account Creation:** Olivia visits the complaint management system's website and clicks on the "Sign Up" button to create a new account. She provides her full name, email address, and a secure password.
- **Email Verification:** After submitting her details, Olivia receives a verification email. She clicks on the verification link to confirm her account.
- **Login:** Olivia logs into the system using her email and password, gaining access to her personalized dashboard.

2. Complaint Submission:-

- **Initiating a Complaint:** Upon logging in, Olivia is directed to her dashboard, which displays an option to "Submit Complaint." She clicks on this option to begin the process.
- **Filling the Complaint Form:** Olivia completes the complaint form by:
  - Describing the issue in detail.
  - Attaching relevant documents or images showcasing the defect.
  - Providing additional information such as her contact details and the product's purchase date.
- **Review and Submission:** After reviewing the information, Olivia submits the complaint. The system generates a unique case reference number for tracking purposes.

3. Tracking and Notifications

- **Real-Time Tracking:** Olivia navigates to the "My Complaints" section of the dashboard, where she can track the status of her complaint in real-time.
- **Email Notifications:** Olivia receives email notifications whenever there is an update on her complaint, such as it being assigned to an agent or its resolution status.
- **Dashboard Updates:** The dashboard reflects the current status of the complaint, including stages like "Under Review," "Assigned to Agent," and "Resolved."

4. Interaction with Customer Service Agent:-

- **Assignment of Agent:** A customer service agent, Sarah, is assigned to handle Olivia's complaint. Sarah reviews the details provided by Olivia and contacts her through the system's built-in messaging feature.

- **Communication:** Olivia receives a notification about Sarah's message and accesses the chat window to communicate with her. They discuss the issue further, and Sarah assures Olivia that the company will investigate and resolve the problem promptly.
- **Documentation:** All communications between Olivia and Sarah are securely stored within the case record for easy retrieval and future reference.

5. Resolution and Feedback:-

- **Investigation and Resolution:** After investigating the complaint, the company identifies the defect in the product and offers Olivia a replacement or refund.
- **Notification:** Olivia receives a notification informing her of the resolution, along with instructions on how to proceed.
- **Feedback:** Olivia provides feedback on her experience with the complaint handling process, expressing her satisfaction with the prompt resolution and courteous service provided by Sarah.

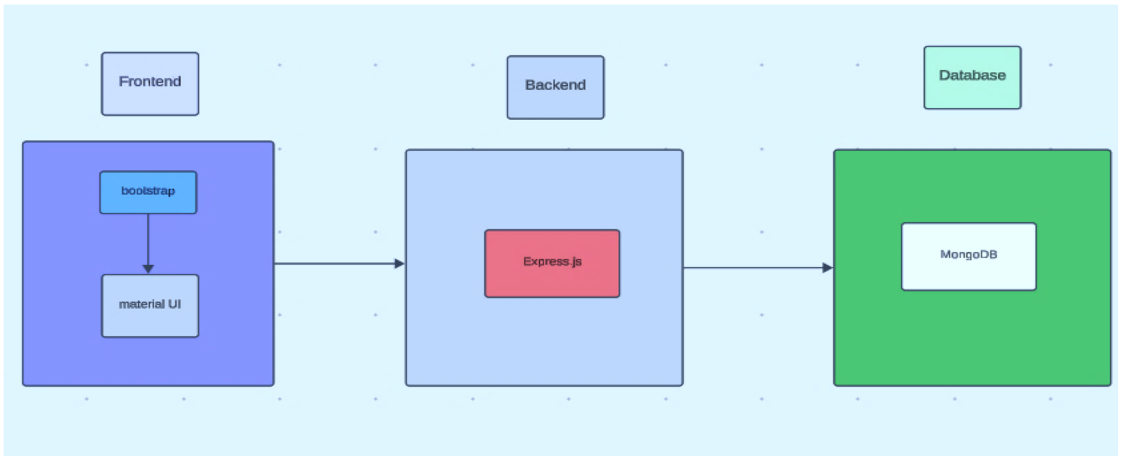
6. Admin Management:-

- **Monitoring Complaints:** The system administrator monitors all complaints registered on the platform. They oversee the overall operation of the complaint management system, ensuring compliance with platform policies and regulations.
- **Assigning Complaints:** The admin assigns complaints to agents based on their workload and expertise.
- **Reporting and Analytics:** The admin generates reports on metrics like resolution times, customer satisfaction, and complaint trends to identify areas for improvement.

TECHNICAL ARCHITECTURE:

Our online complaint registration and management system is designed using a **three-tier client-server architecture**, ensuring scalability, maintainability, and efficient data handling. The architecture is divided into three distinct layers:

1. **Presentation Layer (Frontend)**
2. **Application Layer (Backend)**
3. **Data Layer (Database)**



1. Presentation Layer (Frontend)

- **Frameworks & Libraries**: The frontend is developed using **React.js**, a popular JavaScript library for building user interfaces. It leverages **Bootstrap** and **Material-UI** for responsive and modern design components, ensuring a seamless user experience across devices.
- **State Management**: **Redux** is utilized for managing application state, providing a predictable state container that facilitates debugging and testing.
- **API Communication**: The frontend communicates with the backend via **Axios**, a promise-based HTTP client, to make RESTful

API calls. This setup allows for efficient data exchange between the client and server.

- **Real-Time Communication**: For real-time updates and notifications, **Socket.IO** is integrated, enabling bi-directional communication between the client and server over WebSockets.

2. Application Layer (Backend)

- **Framework**: The backend is built using **Node.js** with the **Express.js** framework. This combination provides a lightweight and flexible environment for handling HTTP requests and building RESTful APIs.
- **Authentication & Authorization**: User authentication is managed using **JWT (JSON Web Tokens)**, ensuring secure and stateless sessions. Role-based access control (RBAC) is implemented to differentiate permissions between users, agents, and admins.
- **Real-Time Communication**: **Socket.IO** is also employed on the server side to handle real-time events, such as new complaint submissions and status updates, ensuring users receive timely notifications.
- **Video Conferencing**: For agent-user interactions requiring face-to-face communication, **WebRTC** is integrated, allowing peer-to-peer video and audio communication directly within the browser.

3. Data Layer (Database)

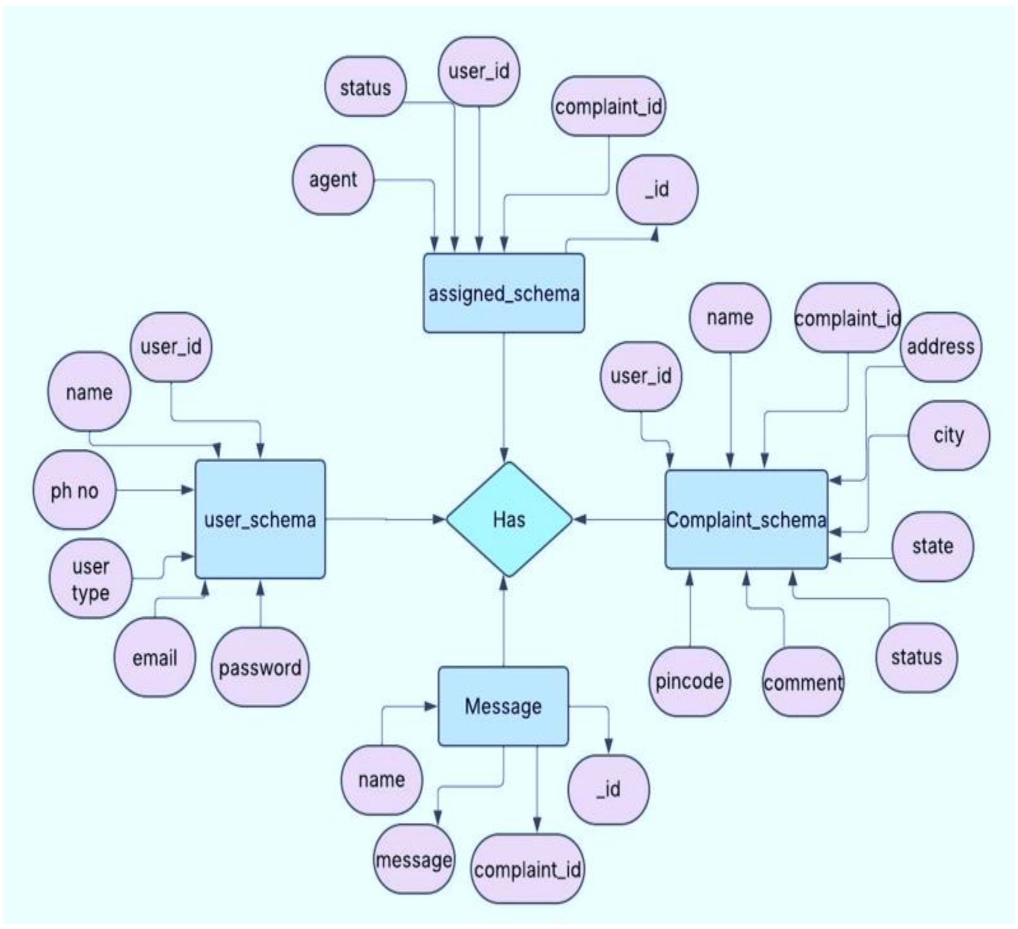
- **Database**: The system utilizes **MongoDB**, a NoSQL database, for storing user profiles, complaint records, and communication logs. MongoDB's flexible schema design allows for efficient storage and retrieval of diverse data types.

- **Data Modelling**: Collections are structured to represent entities such as Users, Complaints, Messages, and Feedback, with appropriate indexing to optimize query performance.
- **Data Security**: Sensitive data is encrypted using industry-standard algorithms, and access controls are enforced to protect user privacy and comply with data protection regulations.
- **Integration and Workflow:**
- **Complaint Submission**: Users submit complaints through the frontend, which are sent to the backend via RESTful APIs. Upon submission, the complaint is stored in MongoDB, and a real-time notification is broadcasted to relevant agents using Socket.IO.
- **Agent Interaction**: Assigned agents receive real-time notifications of new complaints. They can communicate with users via the integrated messaging system, and if necessary, initiate video calls using WebRTC for more in-depth support.
- **AdminOversight**: Administrators have access to a comprehensive dashboard that provides insights into complaint statuses, agent performance, and user feedback. They can assign or reassign complaints, monitor real-time interactions, and generate reports.

4.Key Benefits

- **Scalability**: The modular architecture allows for horizontal scaling, accommodating increased user load and data volume.
- **Maintainability**: Clear separation of concerns between frontend, backend, and database layers simplifies maintenance and future enhancements.
- **Real-Time Interactions**: Integration of Socket.IO and WebRTC ensures users and agents can communicate promptly and effectively.

- **Security**: Robust authentication, authorization, and data encryption mechanisms safeguard user data and system integrity.



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 userId and complaintId from other schemas.

PRE-REQUISITES:

To develop and run the **Online Complaint Registration and Management System**, the following tools, libraries, and platforms are required:

1.Node.js and npm:

- Node.js: JavaScript runtime for server-side code.
- npm: Node Package Manager to install dependencies.

2.Express.js

- **Express.js** is a **minimalist, unopinionated web framework** designed for **Node.js**, enabling rapid creation of RESTful APIs and web applications by abstracting away low-level logic while maintaining full flexibility.
- Installation: Open your command prompt or terminal and run the following command: **npm install express.**

3.MongoDB & Mongoose:

- **MongoDB:** NoSQL database for storing complaints, users, and messages.
- **Mongoose:** ODM for schema modeling and CRUD operations

4.React.js:

- **React.js** is an open-source **JavaScript library**—not a full framework—designed for building user interfaces using **components**. It was developed by Meta (formerly Facebook) and released in 2013. React allows developers to create reusable UI components and efficiently manage complex UIs.

5.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.

6.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

7.Front-end (User Interface):

We use **React.js** to build the parts of the application users see and interact with, such as forms for entering complaints, checking complaint status, and the admin dashboard. To improve the look and feel, we also use UI libraries like **Material UI** and **Bootstrap**.

8.Axios:

Promise-based HTTP client to make API calls between frontend and backend.

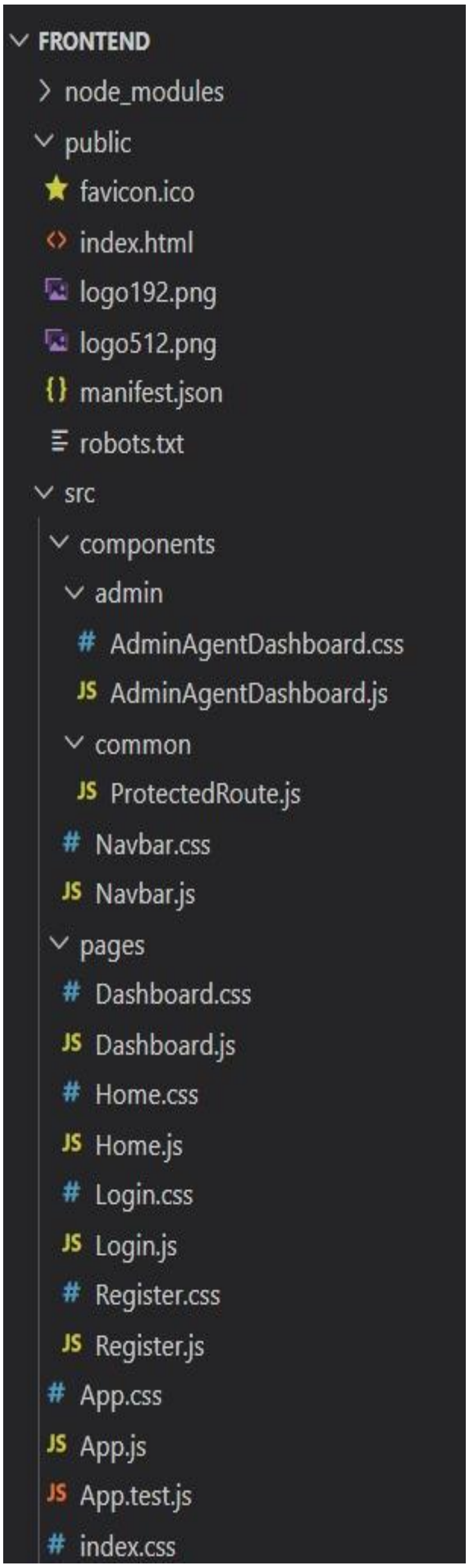
9.bcryptjs:

For secure password hashing and verification.

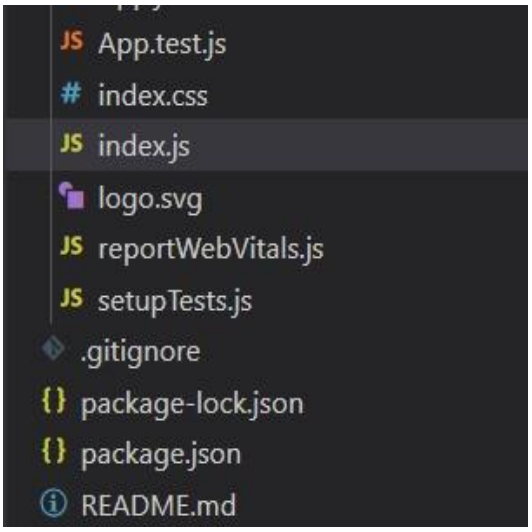
10. CORS & Middleware

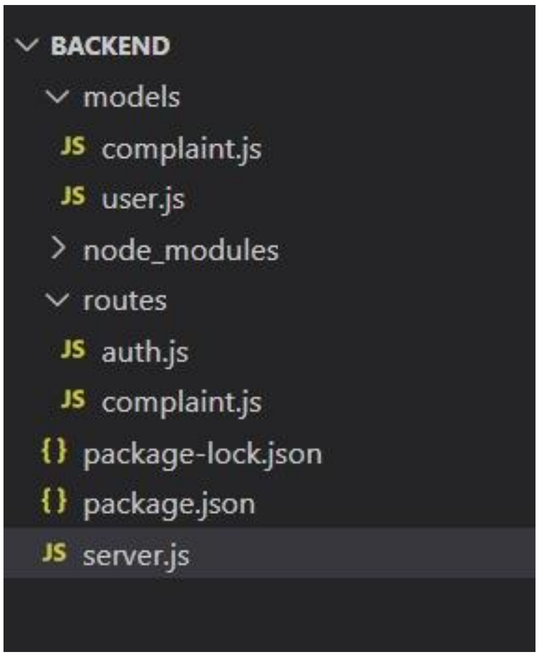
- **CORS**: Handles cross-origin requests.
- Middleware like express.json() handles JSON payloads.

PROJECT STRUCTURE:



```
FRONTEND
  > node_modules
  v public
    favicon.ico
    index.html
    logo192.png
    logo512.png
    manifest.json
    robots.txt
  v src
    v components
      v admin
        AdminAgentDashboard.css
        AdminAgentDashboard.js
      v common
        ProtectedRoute.js
      Navbar.css
      Navbar.js
    v pages
      Dashboard.css
      Dashboard.js
      Home.css
      Home.js
      Login.css
      Login.js
      Register.css
      Register.js
    App.css
    App.js
    App.test.js
    index.css
```


```
JS App.test.js
#  index.css
JS index.js
   logo.svg
JS reportWebVitals.js
JS setupTests.js
   .gitignore
{} package-lock.json
{} package.json
ⓘ  README.md
```


```
∨ BACKEND
  ∨ models
    JS complaint.js
    JS user.js
  > node_modules
  ∨ routes
    JS auth.js
    JS complaint.js
  {} package-lock.json
  {} package.json
  JS server.js
```

The first and second images are of frontend part which shows all the files and folders that have been used in UI development.

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

APPLICATION FLOW:

1. User (Ordinary Customer) Flow:

- **Registration:**
  - User signs up by providing name, email, password, phone, and selects role as "user".
  - Password is hashed and stored securely in MongoDB.
- **Login:**
  - User logs in with email and password.
  - On success, userId and userType are stored in localStorage.
- **Complaint Submission:**
  - User fills complaint form (name, address, city, state, pincode, description).
  - Complaint is submitted via POST /api/complaints.
- **Complaint Tracking:**
  - User clicks “Status” button to view submitted complaints.
  - Each complaint shows current status (Pending or Completed) and message history.
- **Message Communication:**
  - User sends messages related to the complaint via chat box.
  - Messages include sender: 'user' and senderName.

2. Agent Flow

- **Login:**
  - Agent logs in and is redirected to /agent-home.
- **Complaint Access:**
  - Agent can view all complaints assigned (if using assignment logic) or all open complaints.

- **Message Communication:**
    - Agent replies to user messages using the chat window.
    - Messages are saved with sender: 'agent' and their actual name.
- **Status Update:**
    - Agent may mark the complaint as Completed.

3. Admin Flow

- **Login:**
    - Admin logs in and is redirected to /admin-home.
- **View All Complaints:**
    - Admin sees all user complaints on dashboard.
    - Can view user details and full complaint information.
- **Assign or Respond:**
    - Admin can reply to messages or assist agents.
- **Change Status:**
    - Admin can click "Mark as Completed" to close the complaint.
    - Triggers a PUT /api/complaints/:id/status request (you may need to implement this in backend).

System Requirements:

1.User Interface: The system shall provide an intuitive and user-friendly interface for customers, agents, and admins to interact with the platform.

2.Complaint Management: The system shall be able to manage complaints efficiently, including assignment, tracking, and resolution.

3.Communication: The system shall facilitate communication between customers, agents, and admins via built-in messaging features.

4.Security: The system shall ensure the security and integrity of user data, including authentication, authorization, and data encryption.

Project Flow:

The development of the Online Complaint Registration and Management System was organized into five major milestones, each covering critical phases from setup to deployment.

Milestone 1: Project Setup & Environment Configuration

- **Folder Structure Created:**
    - Client folder for React frontend
    - Server folder for Express backend
- **Essential Tools Installed:**
    - Node.js and npm
    - MongoDB
    - Visual Studio Code (as IDE)
    - Git for version control
- **Library Setup:**
    - Backend: Express, Mongoose, bcryptjs, cors
    - Frontend: React, Axios, React Router DOM

Milestone 2: Backend Development

- **API Development:**
  - Authentication Routes: Register, Login, Get User
  - Complaint Routes: Submit, Get All, Get by User, Post Message, Update Status
- **Database Modelling:**
  - **User Schema:** Stores name, email, hashed password, phone, user Type (user/agent/admin)
  - **Complaint Schema:** Includes userId, address, complaint details, status, and embedded messages
  - **Message Schema:** Embedded in complaints with fields like sender, senderName, content, timestamp
- **Secure Authentication:**
  - Passwords hashed with bcryptjs
  - Role-based logic handled in frontend via userType
- **Database Connection:**

  Connected to mongodb://localhost:27017/complaintSystem

Milestone 3: Frontend Development

- **React App Structure:**
  - Components for login, registration, complaint form, dashboards
  - Role-based views: Ordinary User, Agent, Admin
- **User Flow Implemented:**
  - Register → Login → Submit Complaint → View Status → Chat
- **Admin & Agent Features:**

    - View all complaints
    - Reply to messages
    - Mark complaint as completed
- **Real-Time-Like Interaction:**
    - Message list updates upon new replies
    - Sender name and type shown with each message

Milestone 4: Integration and Testing

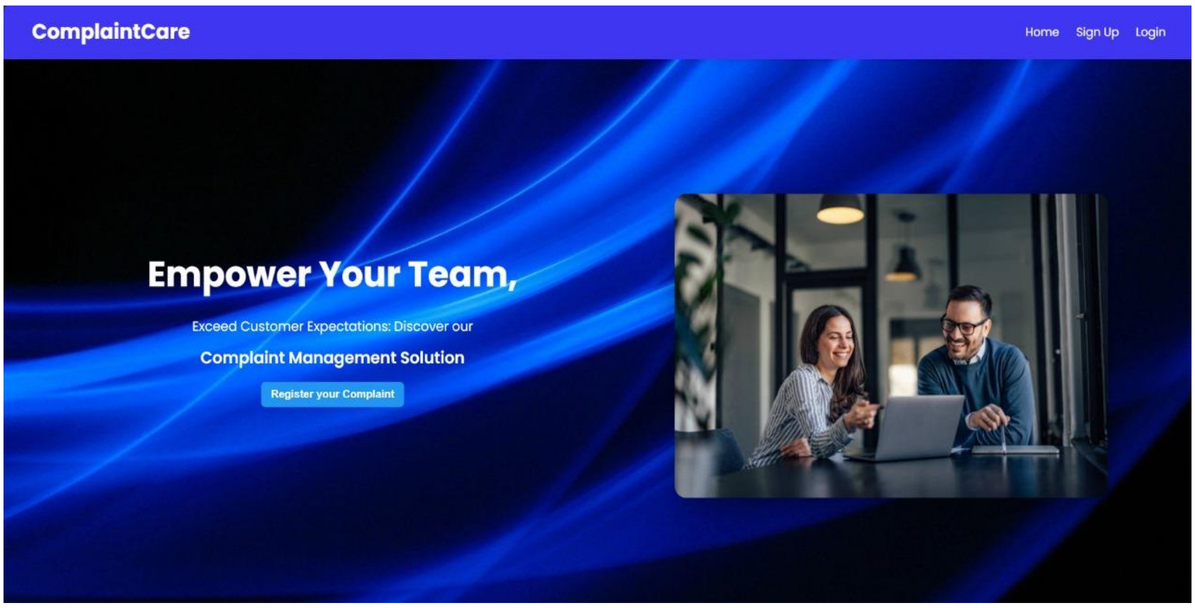
- **Frontend-Backend Integration:**
    - Axios used for REST API communication
    - Message system dynamically updates complaint document in MongoDB
- **Testing Scenarios:**
    - User can submit and track complaints
    - Admin/Agent can reply and close complaints
    - Chat flow verified between user and staff roles
- **Bug Fixes & Validation:**
    - Form validation
    - Error handling for empty inputs, failed logins, etc.

Milestone 5: Final Implementation and Demo

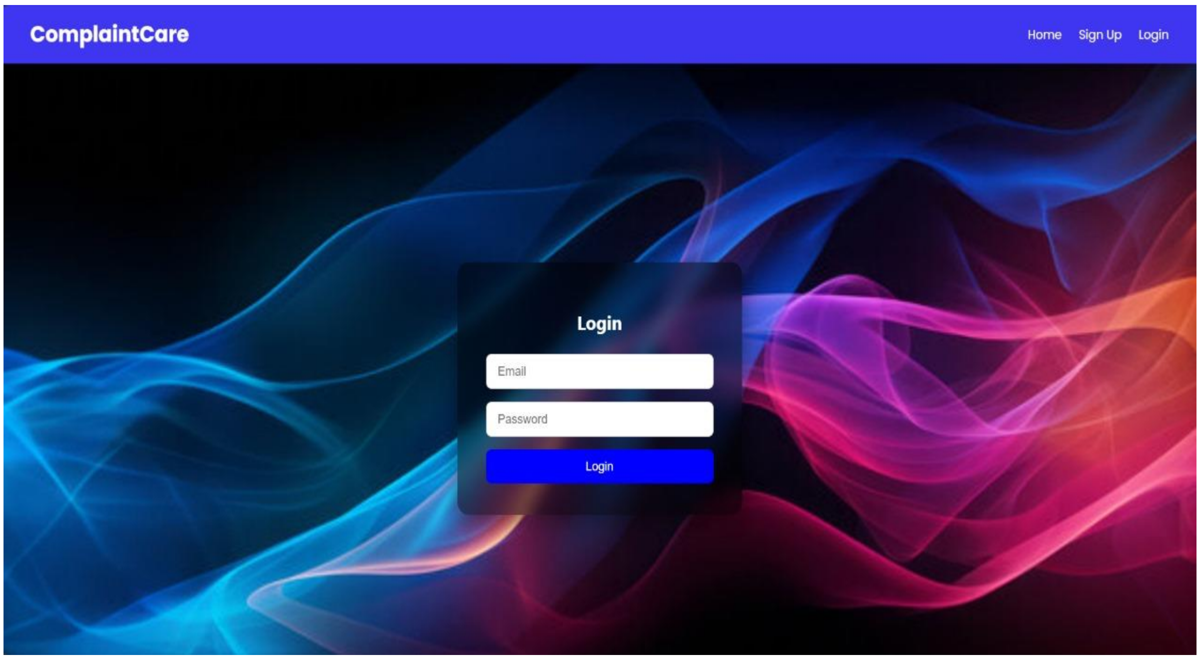
- **Deployment (Local):**
    - Frontend: http://localhost:3000
    - Backend API: http://localhost:5000
- **Demo Recorded:**
    - Walkthrough of all roles and their respective flows

- Shared via Google Drive for review

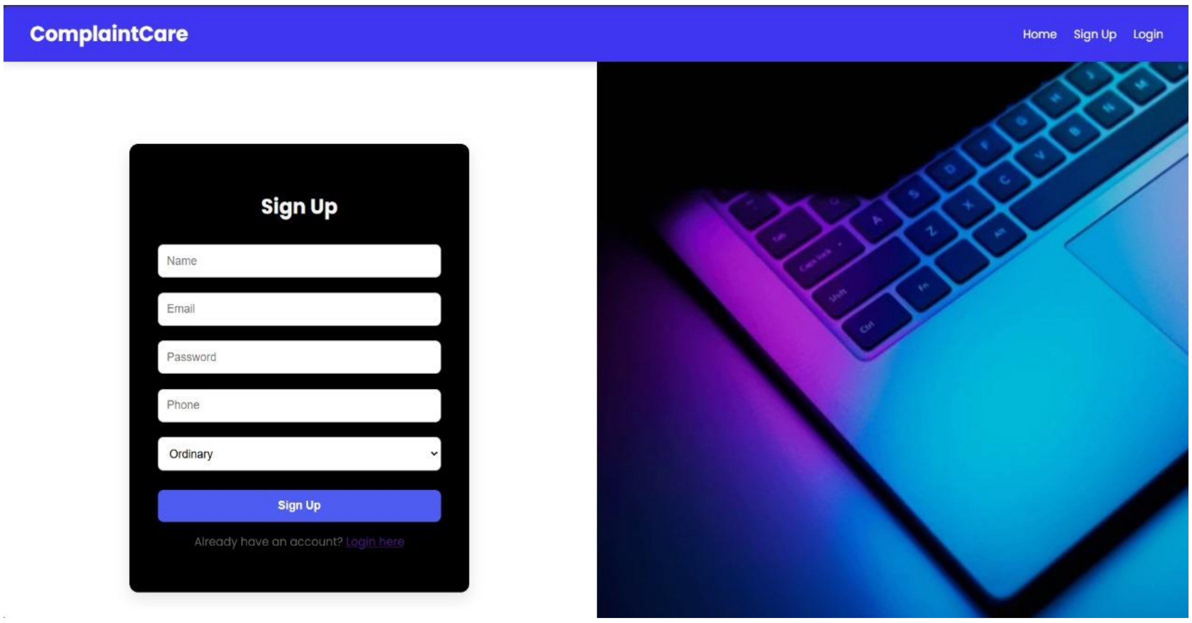
- Landing Page


ComplaintCare
Home
Sign Up
Login
Empower Your Team,
Exceed Customer Expectations: Discover our
Complaint Management Solution
Register your Complaint

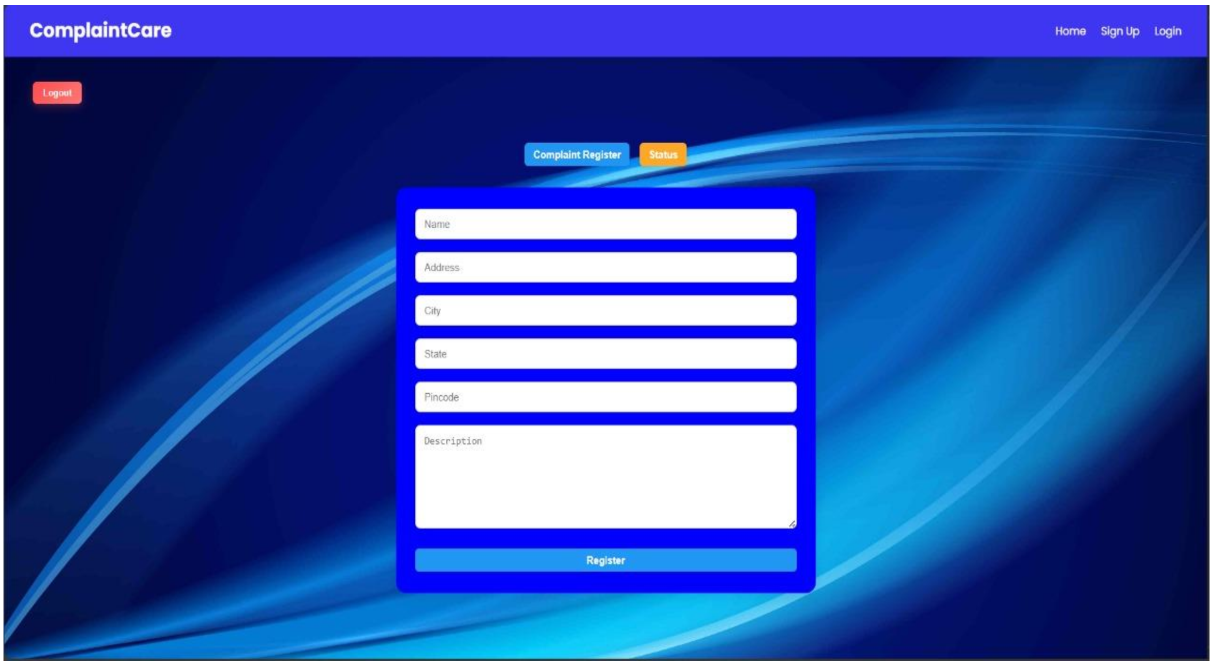
- Login Page



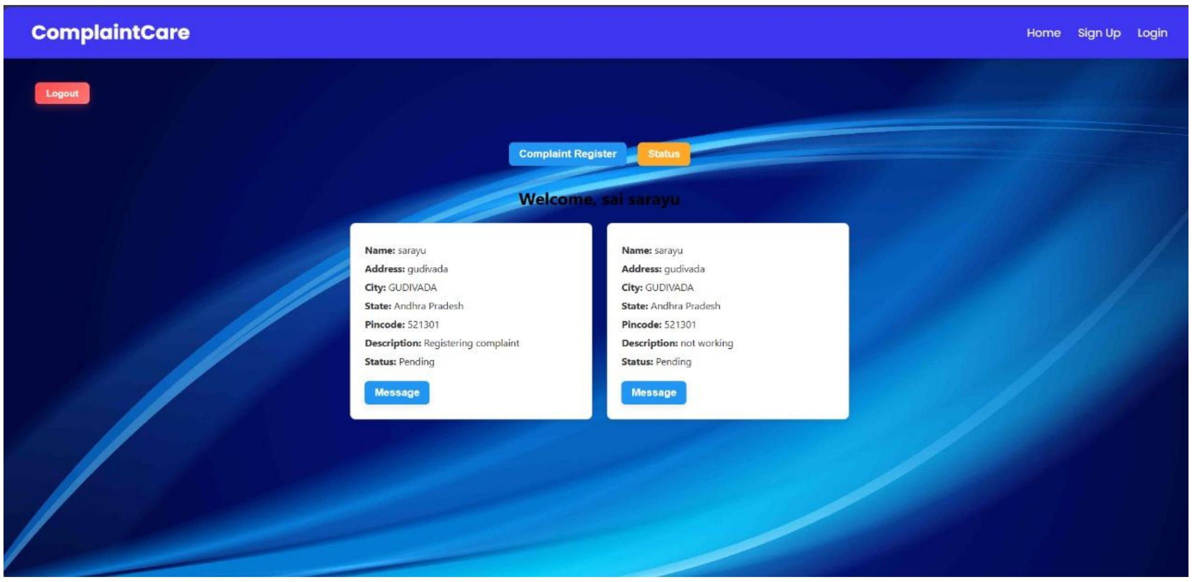
- Registration Page



- Common Dashboard For Complaint



- Admin Dashboard

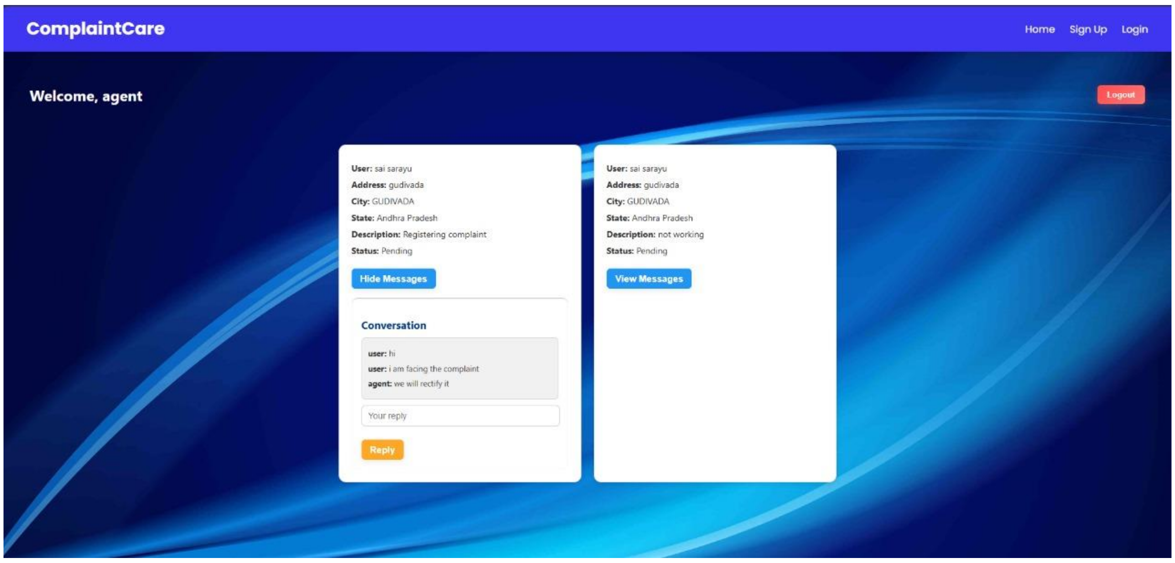


Welcome, sai sarayu

Name: sarayu
Address: gudivada
City: GUDIVADA
State: Andhra Pradesh
Pincode: 521301
Description: Registering complaint
Status: Pending

Name: sarayu
Address: gudivada
City: GUDIVADA
State: Andhra Pradesh
Pincode: 521301
Description: not working
Status: Pending

- Agent Dashboard



Welcome, agent

User: sai sarayu
Address: gudivada
City: GUDIVADA
State: Andhra Pradesh
Description: Registering complaint
Status: Pending

Conversation

user: hi
user: i am facing the complaint
agent: we will rectify it

User: sai sarayu
Address: gudivada
City: GUDIVADA
State: Andhra Pradesh
Description: not working
Status: Pending