ONLINE COMPLAINT REGISTRATION

Project Title:

ResolveNow : Your Platform for Online Complaints

1.INTRODUCTION

ResolveNow is an online platform built to bridge the gap between dissatisfied customers and businesses, making complaint resolution faster, easier, and more transparent.
Whether it's a product issue, poor service, billing error, or delayed delivery, ResolveNow provides a centralized space for users to file complaints, communicate with companies, and track progress—all in one place. By empowering individuals to speak up and enabling businesses to respond efficiently, ResolveNow helps create a more accountable and customer-focused marketplace.With ResolveNow, your voice is heard—and your concerns are resolved.

Team Members

1.Allaka Hari Prakash

2.Challa Bhargav

3.Amarthi Satya

4.Bade Navya

2.PROJECT OVERVIEW

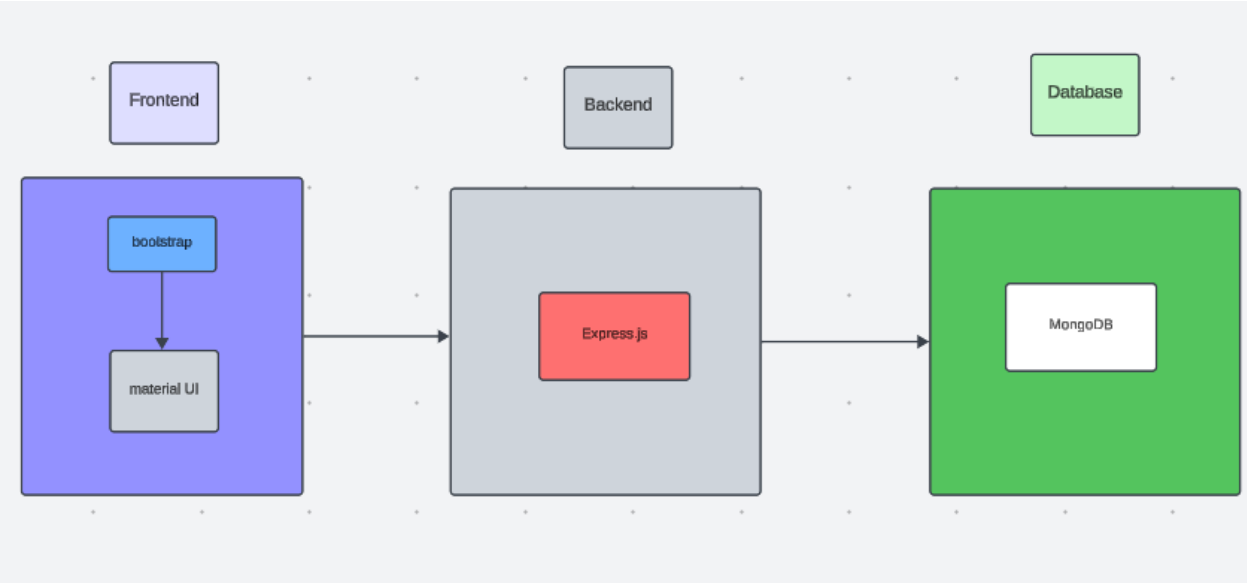
Purpose

The Online Complaint Registration and Management System is a user-friendly platform designed to streamline complaint handling by enabling users to register, track, and interact with agents for resolution.

Features

1. User registration and login
2. Complaint submission with file attachments
3. Real-time complaint tracking and notifications
4. Interaction with agents via chat
5. Complaint routing to relevant departments
6. Secure user data and system access control

3.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 back end 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.

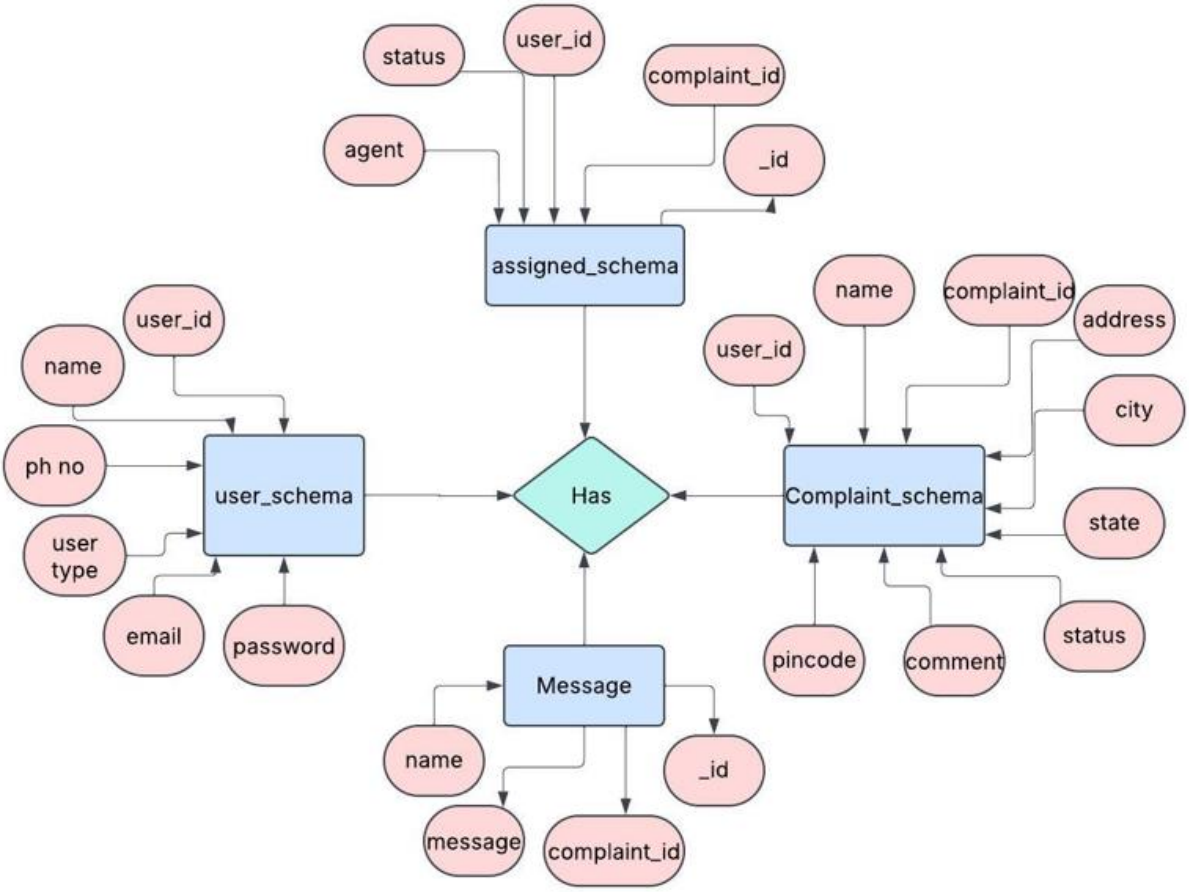
The front end utilizes the bootstrap and material UI library to establish a real-time and better UI experience for any user whether it is an agent, admin, or ordinary user working on it.

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 userId and complaintId from other schemas.

4.SETUP INSTRUCTIONS

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/satya-amarthi04/ResolveNow-Platform-For-Online-Complaints/tree/main

INSTALLATION

• Navigate into the cloned repository directory:
    cd ComplaintCare-MERN
• Install the required dependencies by running the following commands:
    cd frontend
    npm install

cd ../backend
npm install

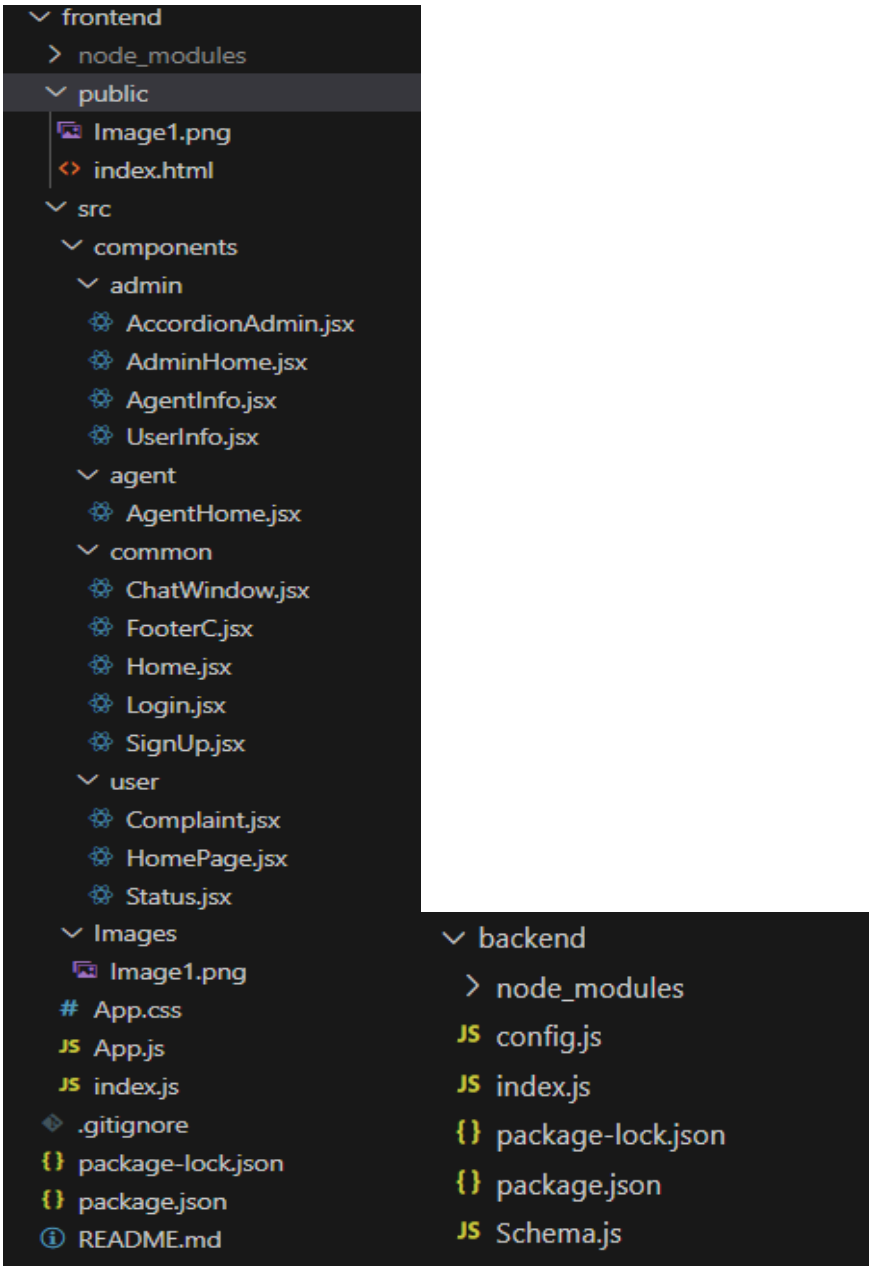
Start the Development Server:

• 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

5.FOLDER STRUCTURE



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

6.RUNNING THE APPLICATION

To run the Online Complaint Registration and Management System, follow the instructions below to set up and launch both the frontend and backend servers locally.

Step 1: Start the Backend Server

1. Open a terminal and navigate to the backend directory:
   cd server

2. Install backend dependencies:
   npm install

3. Set up environment variables by creating a .env file in the root of the server folder. Example:
   PORT=5000
   MONGO_URI=mongodb://localhost:27017/complaintDB
   JWT_SECRET=your_secret_key

4. Start the backend server:
   npm start

The backend server should now be running at: http://localhost:5000

Step 2: Start the Frontend (React Application)

1. Open a new terminal window and navigate to the frontend directory:
   cd client

2. Install frontend dependencies:
   npm install

3. (Optional) Add a .env file in the client folder if you're configuring environment variables:
   REACT_APP_API_URL=http://localhost:5000/api

4. Start the frontend application:
   npm start

The frontend will now be available at: http://localhost:3000

Final Output

Once both servers are running:

- The frontend application will interact with the backend API through RESTful services using Axios.
- You can access the full functionality — including user registration, complaint tracking, agent interaction, and admin management — from the UI.

7.API DOCUMENTATION

The following table outlines the core REST API endpoints used in the Online Complaint Registration and Management System. These APIs are used for user authentication, complaint handling, agent assignment, and messaging.

Authentication APIs

Endpoint	Method	Description	Request Body / Params	Response
/api/register	POST	Register a new user	{ name, email, password, userType }	201 Created / 400 Error
/api/login	POST	User login & JWT generation	{ email, password }	{ token, userType, user }
/api/user/profile	GET	Get logged-in user's profile	JWT Token in headers	User object

Complaint Management APIs

Endpoint	Method	Description	Request Body / Params	Response
/api/complaints	POST	Create a new complaint	{ name, address, state, pincode, comment }	201 Created
/api/complaints	GET	Get all complaints (Admin)	JWT Token	Array of complaints
/api/complaints/:id	GET	Get complaint by ID	:id in URL	Complaint object

/api/complaints/user/:userId	GET	Get all complaints by a user	:userId in URL	Array of complaints

Agent Assignment APIs

Endpoint	Method	Description	Request Body / Params	Response
/api/assignments	POST	Assign a complaint to an agent	{ agentId, complaintId }	201 Created
/api/assignments/agent/:id	GET	Get assigned complaints by agent	:id agentId in URL	List of complaints

Messaging APIs

Endpoint	Method	Description	Request Body / Params	Response
/api/messages	POST	Send message in chat	{ name, message, complaintId }	201 Created
/api/messages/:complaintId	GET	Get messages for a complaint	:complaintId in URL	Array of messages

8.AUTHENTICATION

The application uses JSON Web Tokens (JWT) for secure authentication and authorization. JWTs ensure that only authenticated users can access protected resources by verifying each request with a token.

Authentication Workflow

1. A user logs in by submitting their email and password to the /api/login endpoint.
2. If credentials are valid, the server responds with a JWT token and user details.
3. The client stores this token (typically in localStorage or sessionStorage).
4. For all future API requests to protected routes, the token must be included in the request headers:

Authorization: Bearer <token>

5. The backend middleware validates the token before processing the request.

Protected Routes

Protected routes include operations like:

- Submitting a complaint
- Retrieving user-specific complaints
- Chat messaging
- Admin or agent-specific functionalities

Access to these routes is restricted unless a valid JWT is provided.

Token Expiry & Security

Tokens have a limited lifespan to ensure security. On expiry, the user must log in again to receive a new token.
It is essential to secure the JWT in the frontend and prevent XSS attacks to protect user sessions.

Middleware Usage

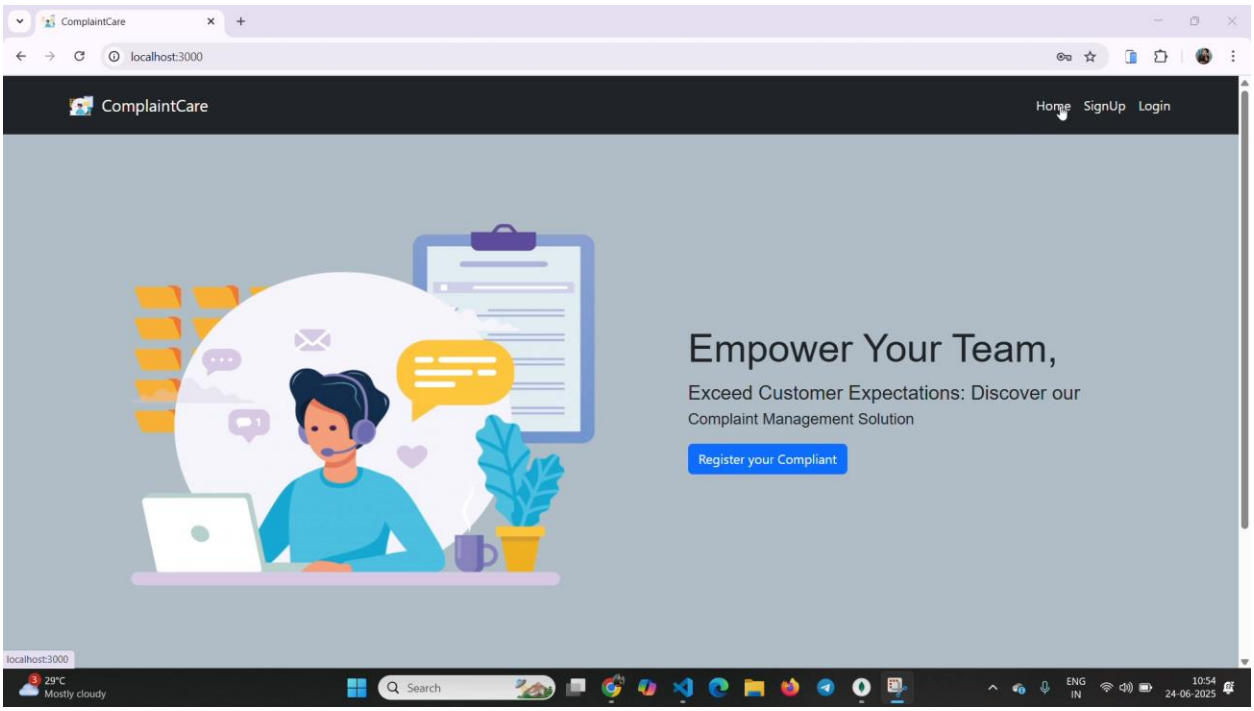
In the backend, middleware functions are implemented to check the presence and validity of tokens for every protected route.
Unauthorized requests without a token or with an invalid token are rejected with a 401 Unauthorized status code.

9.USER INTERFACE

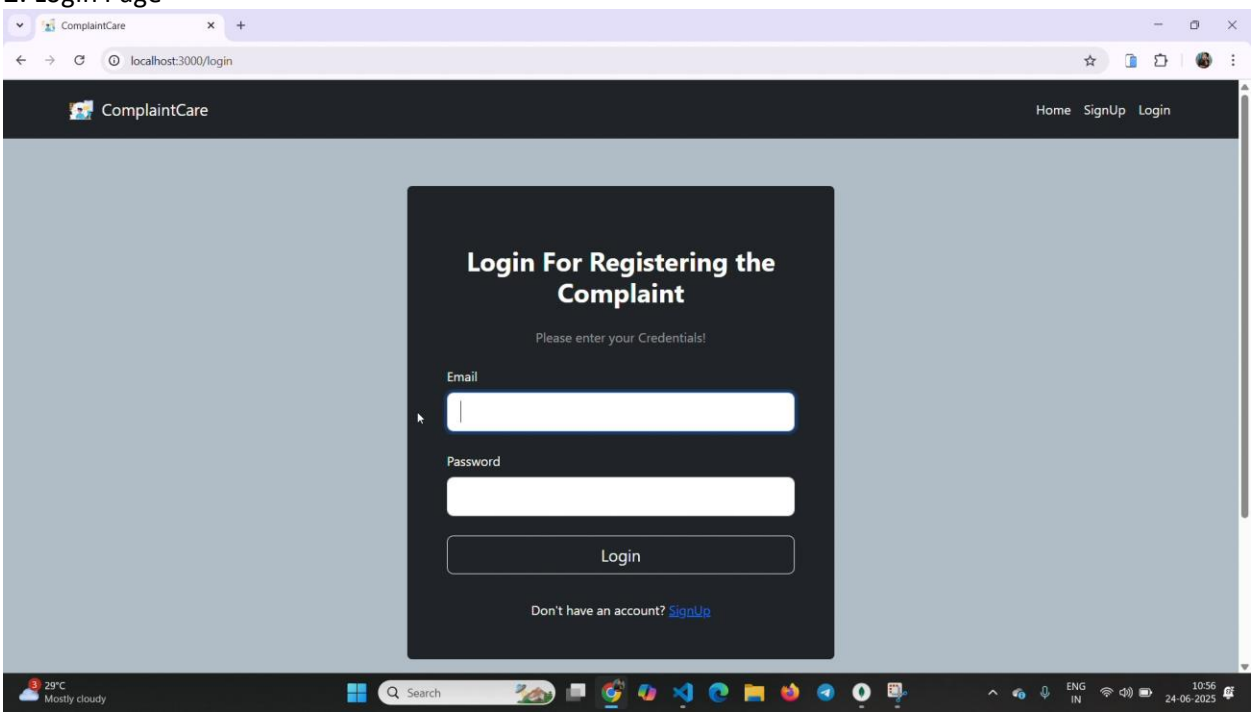
The user interface (UI) of the **Online Complaint Registration and Management System** is designed to be responsive, user-friendly, and role-specific. It is built using **React.js** for component-based development and styled with **Material UI** and **Bootstrap** for modern design consistency.

Screenshots

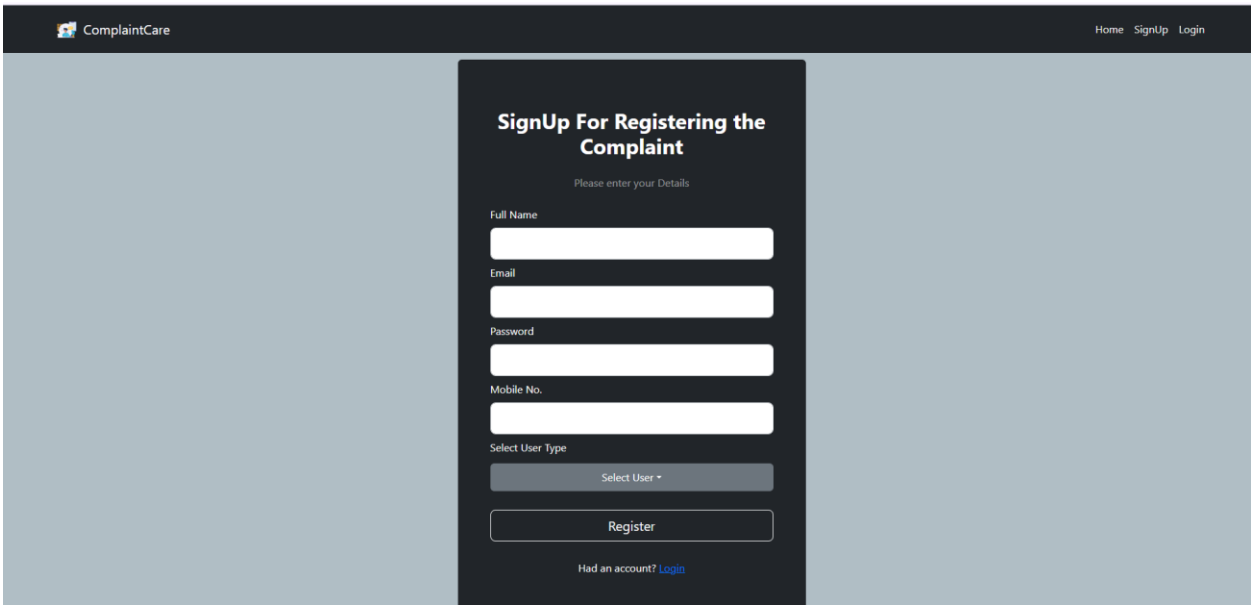
1.Landing page



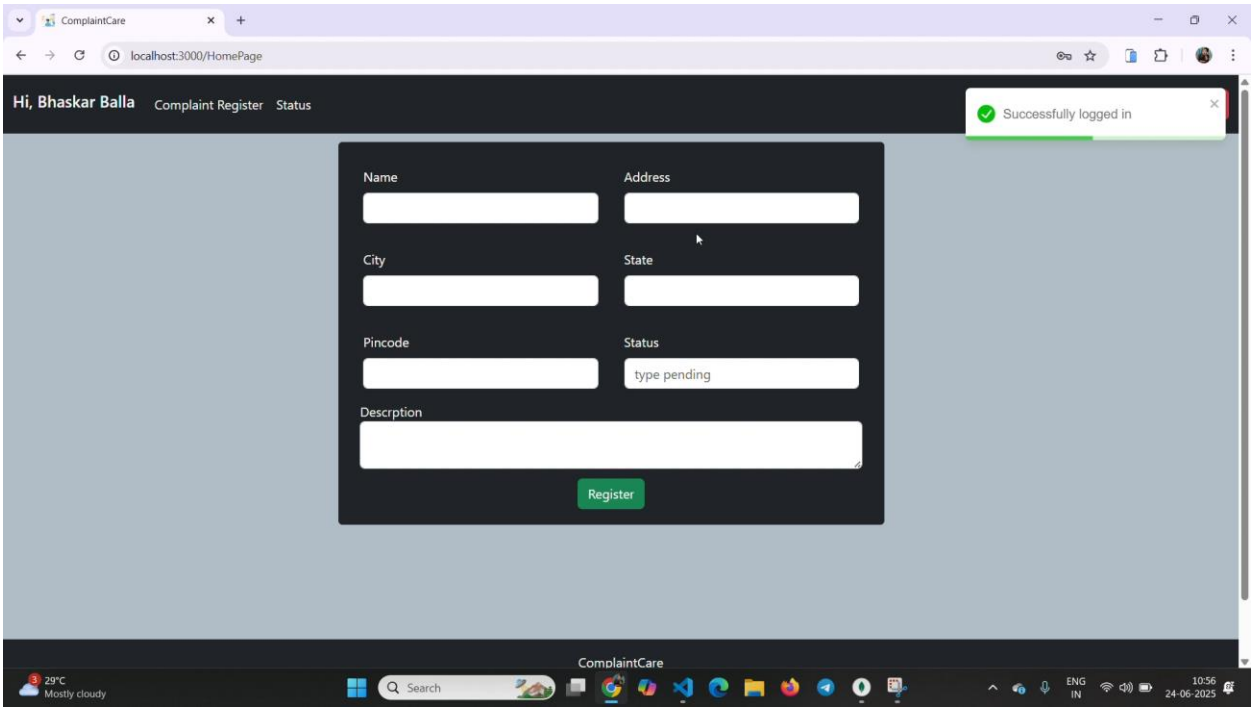
2. Login Page



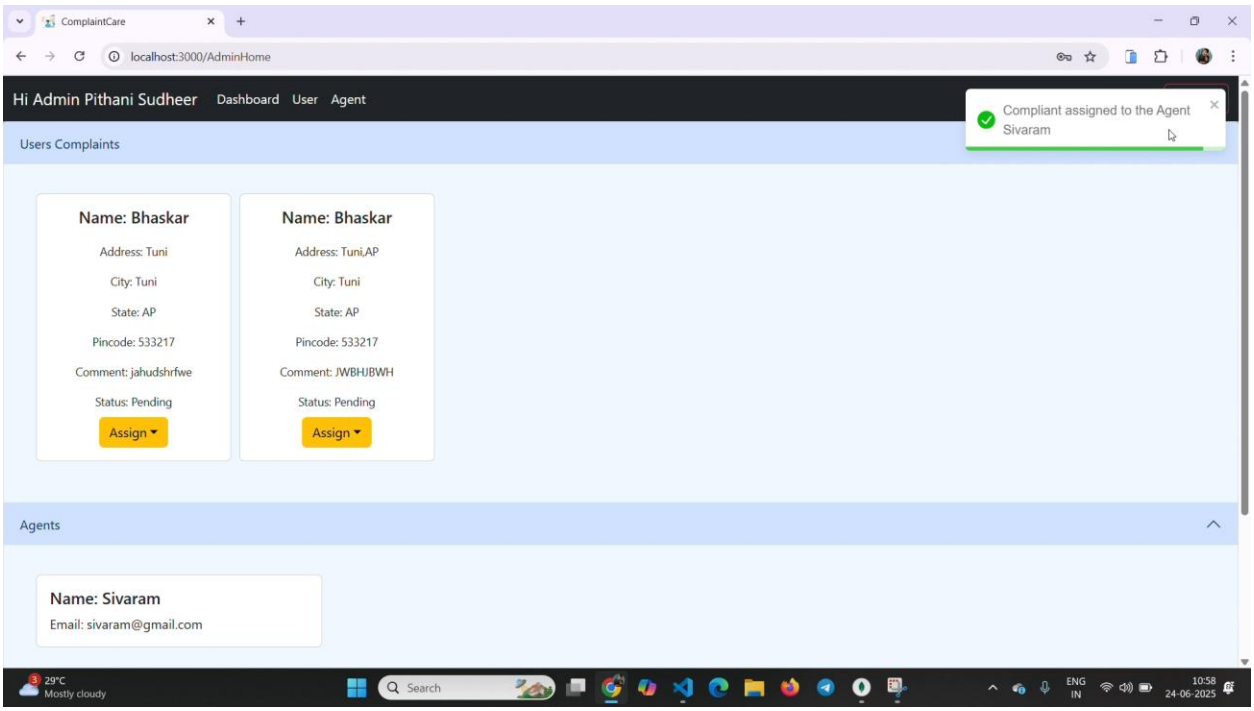
3. Registration Page



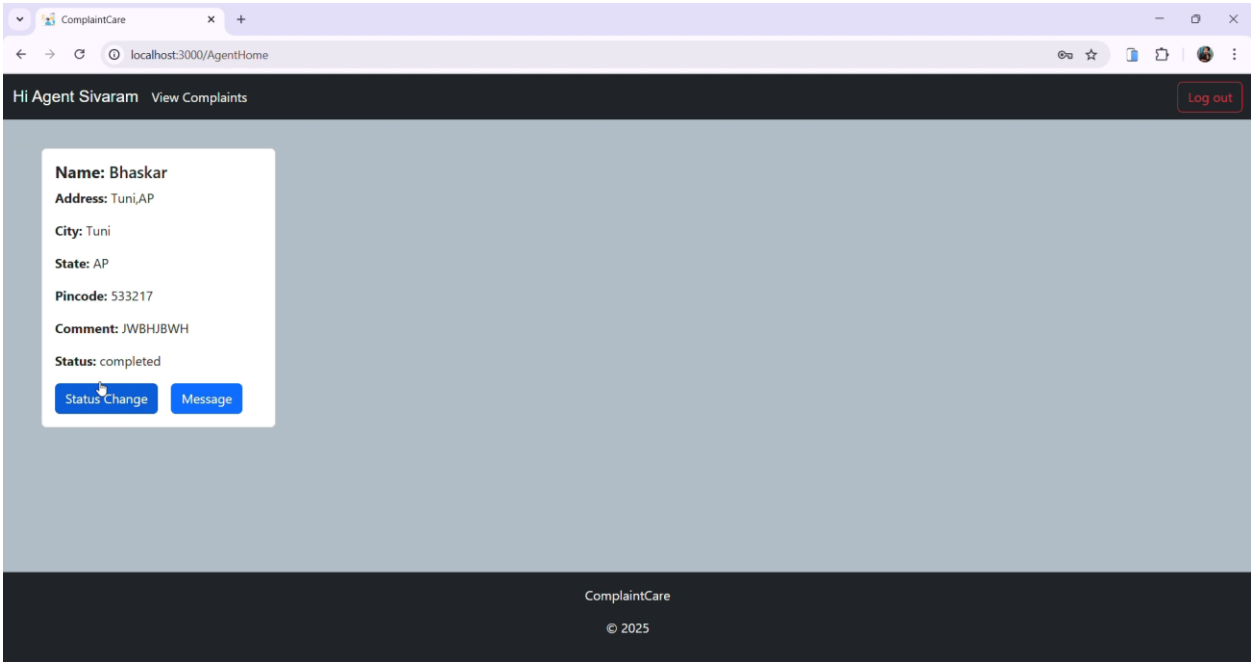
4. Common Dashboard For Complaint



5.Admin Dashboard



6.Agent Dashboard



10.TESTING

Testing is a critical phase of the development process, ensuring that each component of the application functions as expected and delivers a seamless user experience. The Online Complaint Registration and Management System underwent manual and planned automated testing phases.

Manual Testing

Manual testing was performed across different user roles (Customer, Agent, Admin) to validate core functionalities

Tested Functionalities

- User Registration and Login
- Complaint Submission and Form Validation
- Complaint Status Tracking
- Chat Functionality (User ↔ Agent)
- Admin Dashboard Controls
- Role-Based Access Restrictions
- Frontend responsiveness across screen sizes

Tools Used

- **Postman**: For testing REST API endpoints
- **Browser Console & DevTools**: For inspecting UI behavior and network requests
- **MongoDB Compass**: To verify backend data integrity

Example Postman Tests

- POST /api/register – Register a user with valid/invalid data
- POST /api/login – Token response and role detection
- POST /api/complaints – Complaint creation and retrieval
- GET /api/messages/:complaintId – Message history retrieval

11.SCREENSHOTS OR DEMO

Demo: https://drive.google.com/file/d/182pw1nHkkDfUmISlZDOB8FFXbqc1_XJq/view?usp=sharing

12.KNOWN ISSUES

While the **Online Complaint Registration and Management System** is fully functional for its core features, there are several known issues and limitations that have been identified during development and testing. These will be addressed in future updates.

Current Limitations

- **No File Type Validation**
  Users can upload files (such as complaint evidence) without any restrictions on file type or size, which may lead to potential security or storage issues.
- **No Analytics or Dashboard Insights**
  The admin dashboard currently lacks analytics, such as visual summaries of complaint trends, agent performance metrics, or user activity charts.
- **Chat Lacks Media Upload**
  The messaging system supports plain text communication only. There is no option to share images, files, or audio for better context during complaint resolution.

13.FUTURE ENHANCEMENTS

To continually improve user experience and system functionality, the **Online Complaint Registration and Management System** includes several planned enhancements. These improvements are based on feedback from users, testing observations, and evolving application goals.

Planned Features

- **Admin Analytics Dashboard**
  Implement visual data representations such as graphs, charts, and tables to monitor complaint trends, resolution time, agent workload, and system usage.
- **Media & Voice Chat Support**
  Extend the chat functionality to allow users and agents to exchange images, documents, and voice notes for richer and clearer communication.
- **Complaint Categorization & Search Filters**
  Enable complaint tagging and categorization (e.g., “Delivery Issue”, “Product Defect”), along with filters to easily sort or search through complaints based on status, date, or type.

- **User Rating and Feedback System**
  Allow users to rate their experience after a complaint is resolved, helping admins assess agent performance and customer satisfaction.