ResolveNow :
Online Complaint Registration and Management System

1. INTRODUCTION

1.1 **Project Overview**

ResolveNow is an online platform enabling users to register, track, and resolve complaints efficiently. The application offers real-time status updates and interaction with agents for problem resolution.

1.2 **Purpose**

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.

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 as 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. Users 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.

2. IDEATION PHASE

2.1 **Problem Statement**

Manual complaint handling is slow, lacks transparency, and leads to customer dissatisfaction. There is a need for a digital system to streamline and automate the process.

2.2 **Empathy Map Canvas**

- Users: Citizens/customers reporting issues
- Needs: Transparency, fast resolution, communication
- Pains: Delays, no updates, unresponsive support
- Gains: Real-time updates, secure platform, agent support

2.3 **Brainstorming**

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.

1. **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.
2. **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.
3. **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.
4. **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.

5. **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.
6. **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.

3. REQUIREMENT ANALYSIS

3.1 **Customer Journey Map**

From registration → complaint submission → status tracking → agent interaction → resolution.

3.2 **Solution Requirement**

- User roles: User, Agent, Admin - Complaint form - Chat module - Notification system - Admin panel

3.3 **Data Flow Diagram**

Visualizes data between UI, backend APIs, and MongoDB.

3.4 **Technology Stack**

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:

Github: https://github.com/SwaroopSai1123/Online-Complaints/tree/main

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

4. PROJECT DESIGN

4.1 **Problem Solution Fit**

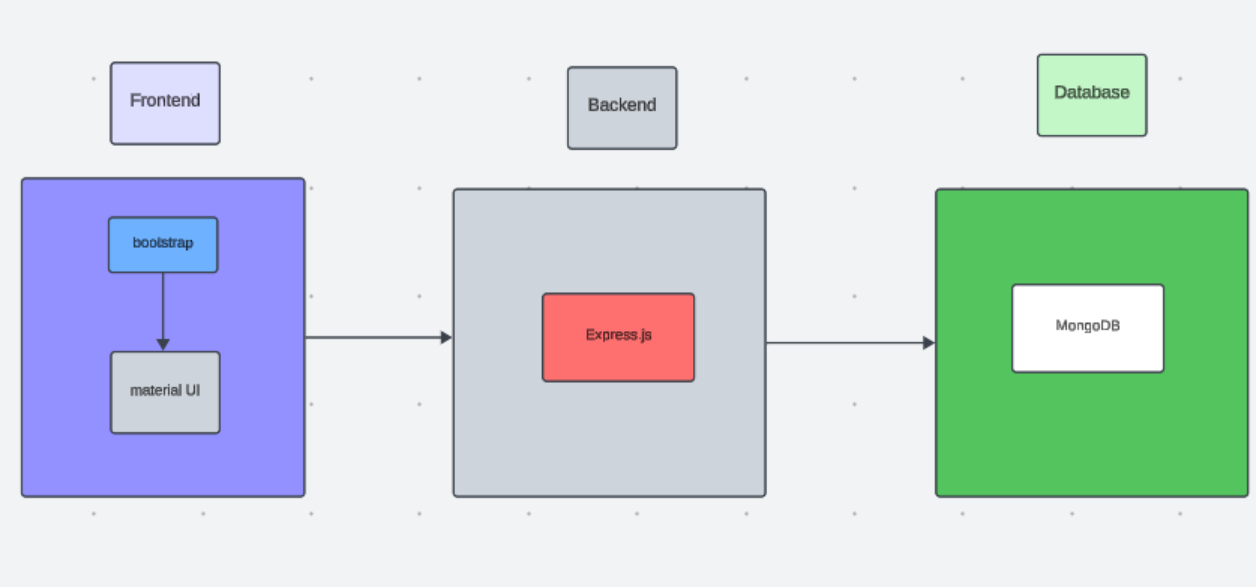
The platform directly solves delays and lack of updates in traditional systems by digitizing the complaint lifecycle.

4.2 **Proposed Solution**

A full-stack MERN application allowing users to register complaints and communicate with support.

4.3 **Solution Architecture**

- Client-server model - REST APIs - MongoDB for schema-driven storage - Socket.io for chat integration



5. PROJECT PLANNING & SCHEDULING

5.1 Project Planning

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.

```
{
  "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.

```
{
  "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:

- **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.

- **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, creating, assigning complaints, and chat window.
  - 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.

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.

- 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.

3. **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.
4. **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:

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.

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.

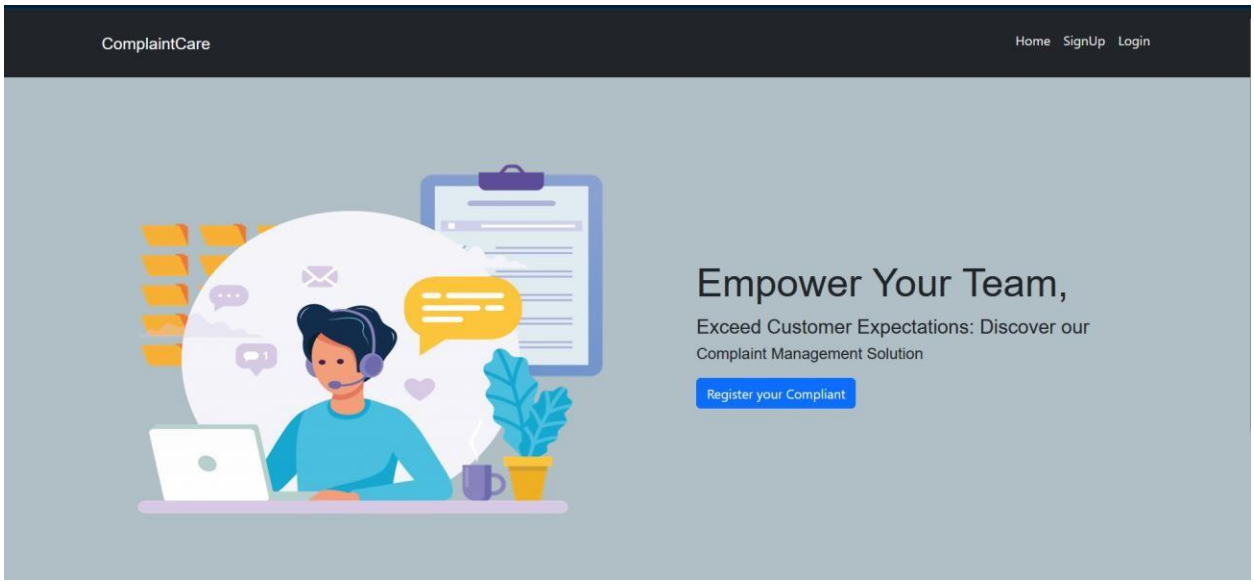
6. FUNCTIONAL AND PERFORMANCE TESTING

6.1 **Performance Testing** - Load tested chat and complaint modules - Verified API response times - Frontend optimized for responsiveness

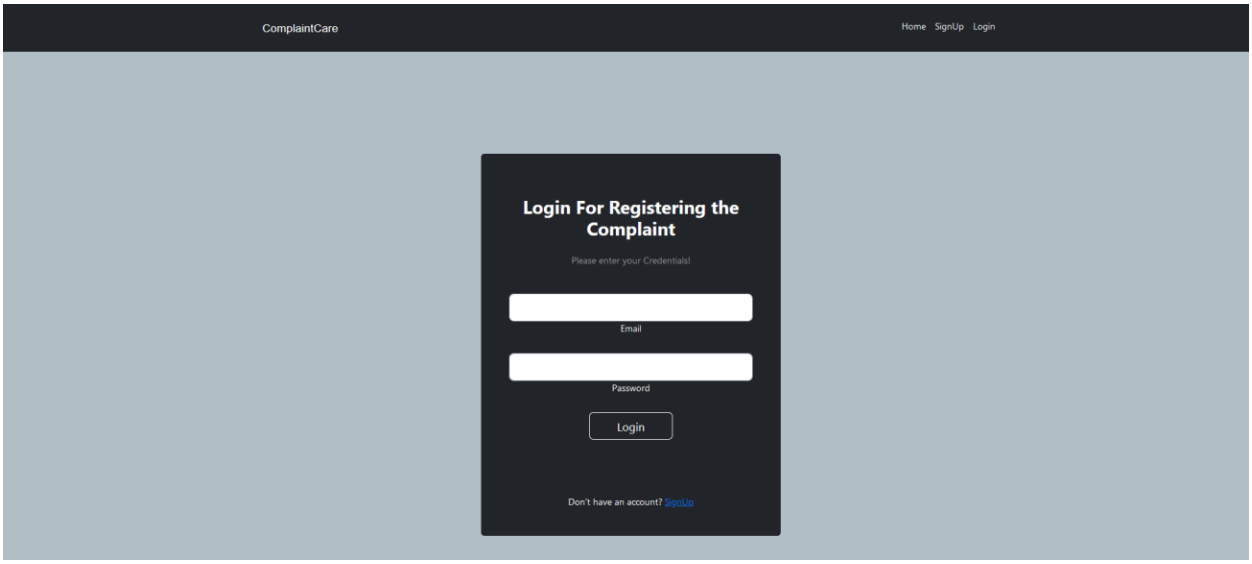
7. RESULTS

7.1 **Output Screenshots**

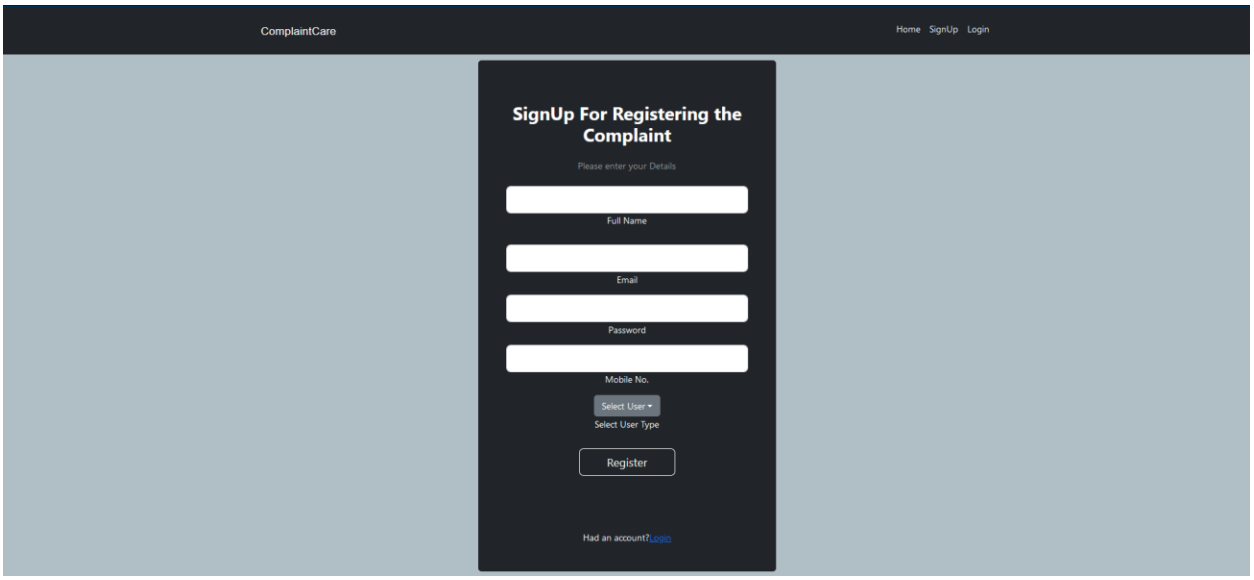
- Loanding Page



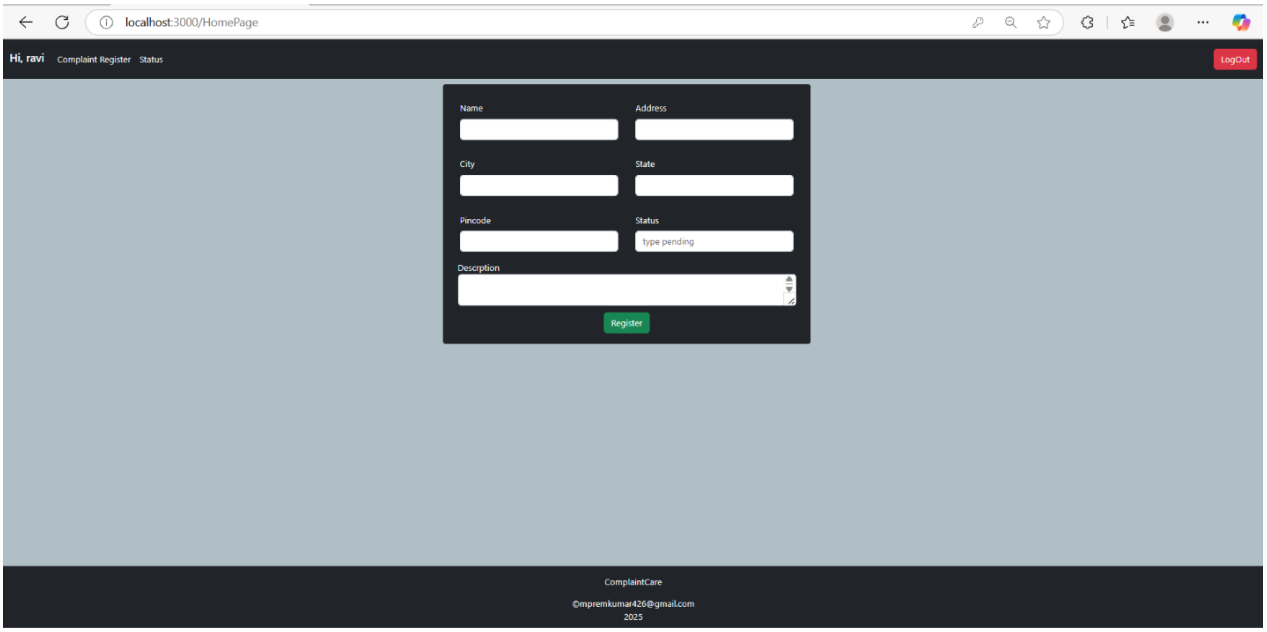
- Login Page



- Registration Page

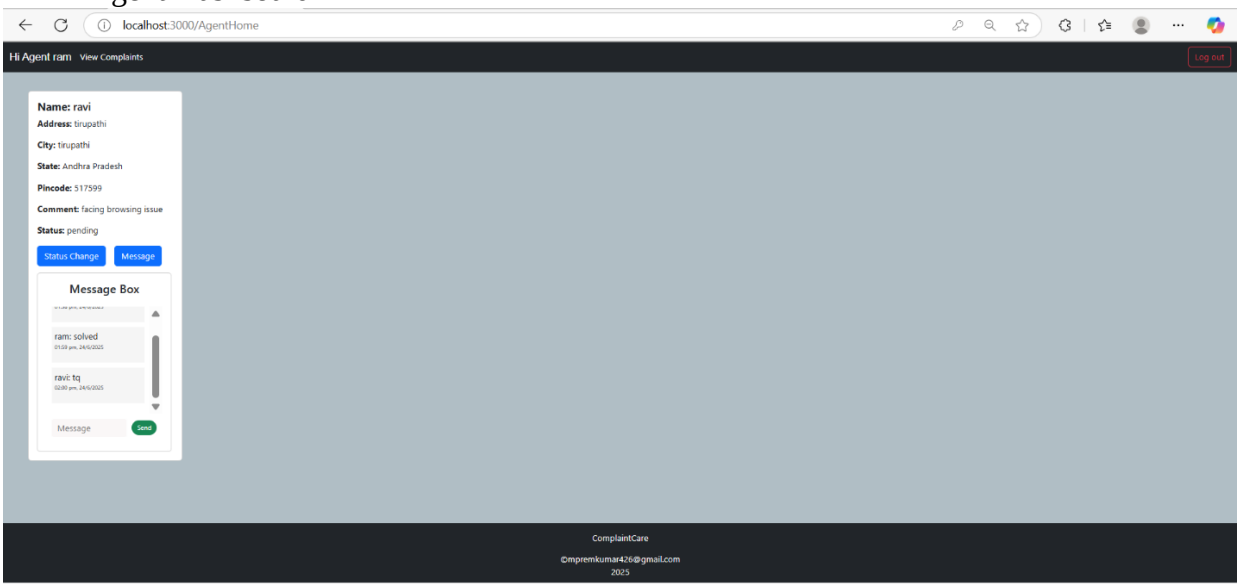


- Common Dashboard For Complaint



- Admin Dashboard

- Agent Dashboard



8. ADVANTAGES & DISADVANTAGES

Advantages:

 Improved Efficiency
Automates the entire complaint lifecycle—from registration to resolution—reducing manual effort and turnaround time.

 Real-time Tracking
Users can monitor complaint status updates in real time, enhancing transparency and trust.

 User-Agent Interaction
The integrated chat module allows direct communication, enabling quick clarification and faster problem-solving.

 Role-based Access Control
Separate interfaces and permissions for Admins, Agents, and Users ensure data integrity and security.

 Centralized Management
All complaints are stored in a centralized database, making it easier for administrators to manage and analyze reports.

 Scalable Architecture
Built using the MERN stack, the system can easily be scaled to support more users and modules in the future.

 Data Security
Incorporates encryption, secure login, and access control measures to protect user data.

 Eco-friendly
Reduces the need for paper-based complaint submissions, supporting digital transformation and sustainability.

Disadvantages:

1. **Internet Dependency**
   The system requires a stable internet connection; users in remote or underdeveloped areas may face access issues.

2. **Initial Setup Cost**
   Organizations may incur costs related to hosting, infrastructure, and employee training.

3. **Learning Curve**
   Users unfamiliar with digital platforms may require onboarding and support to use the system effectively.

4. **Data Breach Risks**
   Though secure, any online system faces potential cybersecurity threats if not regularly updated or maintained.

5. **Limited to Digital Users**
   The platform excludes users who are not tech-savvy or do not have access to digital devices.

9. CONCLUSION

"ResolveNow" is a full-stack web application designed to streamline the process of submitting, tracking, and resolving complaints efficiently. The platform empowers users to register issues, monitor their status, and communicate directly with agents. Admins can manage complaints and assign them to relevant personnel, creating a smooth and organized resolution pipeline.

Key Takeaways:

- Built using the **MERN stack** (MongoDB, Express.js, React.js, Node.js).
- Supports **user registration, complaint submission, real-time tracking, and agent-user interaction**.
- Prioritizes **data security** through authentication and role-based access.
- Offers a clean **frontend UI with Bootstrap/Material UI** and REST API-driven backend.
- Can be extended to include **real-time notifications, charts, analytics, and video support**.

Project Benefits:

- Reduces manual complaint handling.
- Improves customer satisfaction through transparency.

- Scales easily for both small and large organizations.
- Enhances accountability and traceability within teams.

10. FUTURE SCOPE

- Mobile App

- AI-based complaint categorization

- Analytics Dashboard

- Multi-language support

- Integration with SMS/Email services

11. APPENDIX

- **Source Code**: https://github.com/SwaroopSai1123/Online-Complaints/tree/main

- **Demo Link**: https://drive.google.com/file/d/1LV2ParnHkWJJ-IUgObupmU3ZwzSzmwpV/view?usp=sharing