Project Title

Resolve – Online Complaint Registration and Management System

Team ID : LTVIP2026TMIDS78964

TABLE OF CONTENTS

1. INTRODUCTION

Project Title :

ResolveNow – Online Complaint Registration and Management System

Team ID : LTVIP2026TMIDS78964

Team Size : 4

Team Leader : Hema Durga Kokkirala

Team member : Jyothirmayi Ramisetti

Team member : Jaya Kanth Vemuluri

Team member : Jayasri Madem

2. PROJECT OVERVIEW

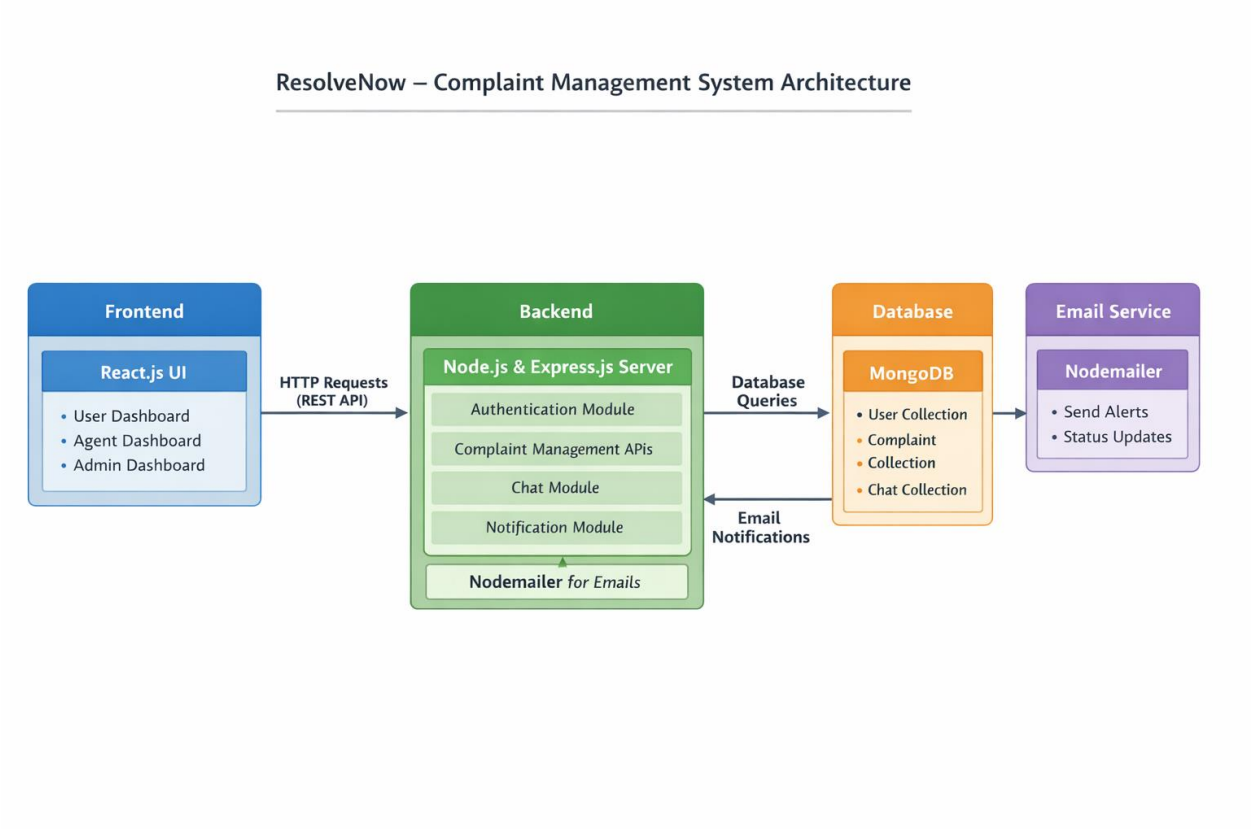
2.1 Purpose :

The purpose of the ResolveNow – Online Complaint Registration and Management System is to provide a centralized web-based platform that enables users to register and track complaints efficiently while allowing administrators and agents to manage and resolve issues through a structured workflow. The system aims to improve transparency, reduce response time, and enhance communication between users and support teams by implementing role-based dashboards, real-time complaint tracking, and automated email notifications. The project is developed using the MERN stack to ensure scalability, security, and efficient data handling.

2.2 Features :

- User Registration and Authentication with role-based access (Admin, Agent, User).
- Complaint Submission and Management through an interactive dashboard.
- Real-Time Complaint Tracking with status updates.
- Admin Panel for managing users, complaints, and agent assignments.
- Agent Dashboard for handling assigned complaints and updating resolution status.
- Chat Communication between users and agents.
- Automated Email Notifications for complaint updates and system events.
- Secure database integration using MongoDB for storing user and complaint data.

3. ARCHITECTURE



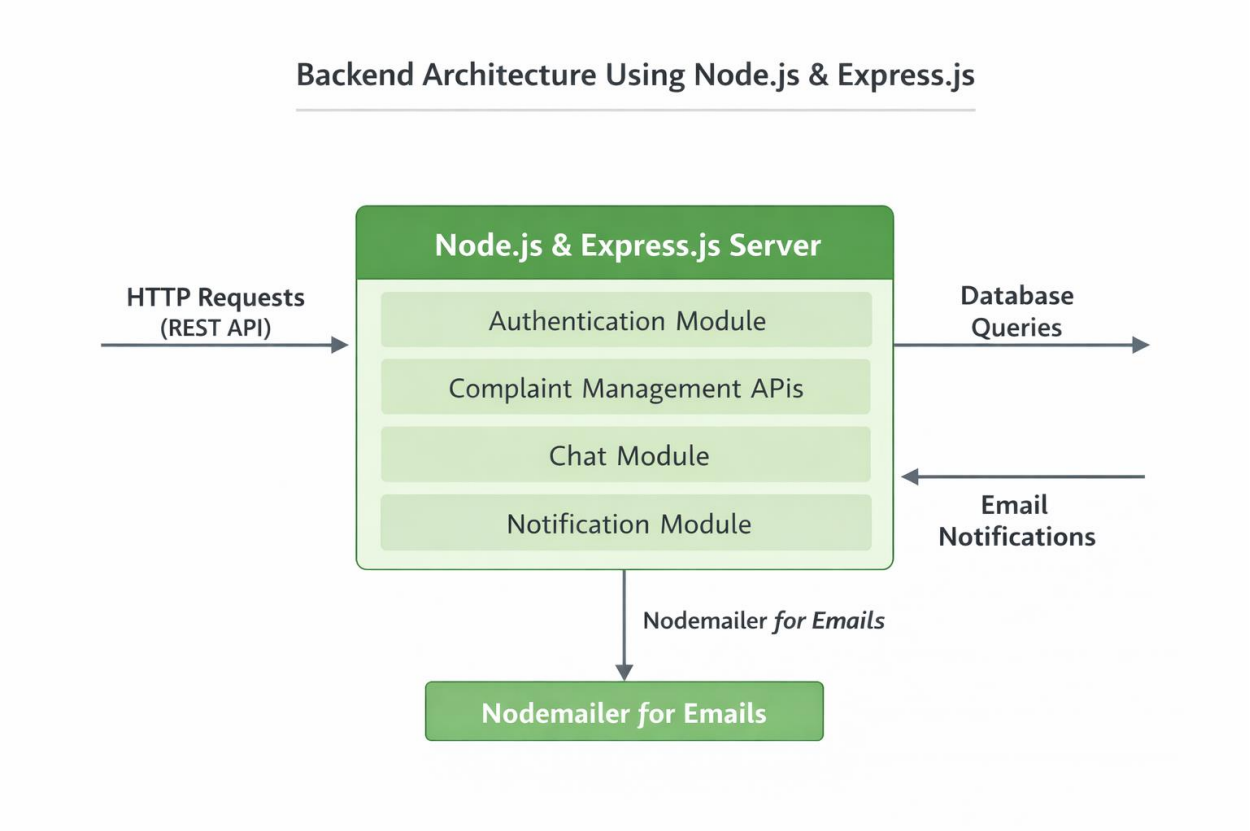
3.1 Frontend (React.js) :

The frontend of the ResolveNow system is developed using **React.js**, following a component-based architecture that improves reusability and maintainability. The user interface is structured into modular components such as Login, SignUp, User Dashboard, Admin Dashboard, Agent Dashboard, Complaint Form, and Status Tracking pages. React Router is used for navigation and role-based routing, ensuring that users are redirected to appropriate dashboards based on their login credentials. Axios is implemented to communicate with backend REST APIs, while Bootstrap and Material UI are used to provide responsive layouts and consistent styling. The frontend manages user sessions using local storage to maintain authentication state across the application.

3.2 Backend (Node.ja & Express.js) :

The backend is built using **Node.js** with the **Express.js** framework, which handles server-side logic and API request processing. The architecture follows a RESTful API structure where routes manage user authentication, complaint submission, assignment, status updates, chat messages, and email notifications. Middleware such as CORS and JSON parsing ensures secure and efficient data exchange between the frontend and backend. Role-based access logic is implemented to control

operations for Admin, Agent, and User roles. Additionally, Nodemailer is integrated to send automated email notifications whenever complaints are registered, assigned, or updated.



3.3 Database (MongoDB) :

The system uses **MongoDB** as a NoSQL database to store application data in structured collections. Mongoose schemas define the data models, including User Schema, Complaint Schema, Assigned Complaint Schema, and Message Schema. Each complaint is linked to a user through a unique userId, while assigned complaints connect agents to specific complaint records. Chat messages are stored with complaint references to maintain communication history. MongoDB enables flexible and scalable data storage, allowing efficient CRUD operations and seamless integration with the Node.js backend.

4. SETUP INSTRUCTIONS

4.1 Prerequisites :

Before running the application, ensure the following software and tools are installed on the system:

- Node.js (v16 or higher)
- npm (Node Package Manager)
- MongoDB (Local installation or MongoDB Atlas)
- Visual Studio Code (Recommended IDE)
- Git (for version control and cloning repository)
- Web Browser (Chrome/Edge recommended)

Additional Libraries Used:

- React.js
- Express.js
- MongoDB & Mongoose
- Axios
- Bootstrap / Material UI
- Nodemailer (for email notifications)
- dotenv (for environment variables)

4.2 Installation :

Step 1: Clone the Repository

Open terminal or PowerShell and run :

```
git clone https://github.com/awdhesh-student/complaint-registery.git
```

Navigate to project folder :

```
cd complaint-registery
```

Step 2: Install Backend Dependencies

```
cd backend
npm install
```

Create a .env file inside backend folder and add :

```
EMAIL_USER=yourgmailaccount@gmail.com
EMAIL_PASS=your_app_password
```

Start Backend Server :

```
npm start
```

Backend will run on :

```
http://localhost:8000
```

Step 3: Install Frontend Dependencies

Open a new terminal :

```
cd frontend
npm install
```

Start React Application :

```
npm start
```

Frontend will run on :

```
http://localhost:3000
```

Step 4: Database Setup

- Ensure MongoDB service is running locally or connected through MongoDB Atlas.
- The backend automatically connects using Mongoose configuration inside config.js.

Step 5: Run the Application

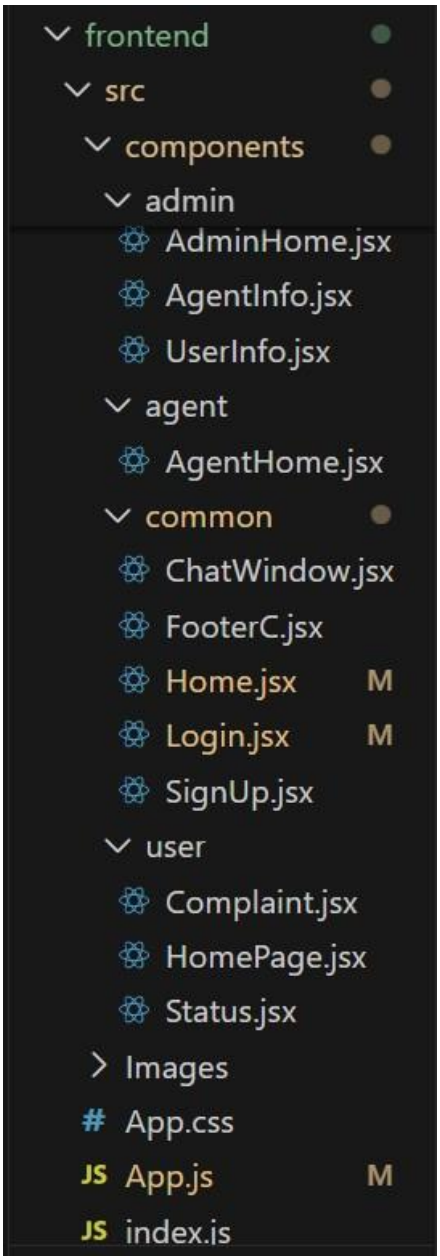
1. Start Backend Server.
2. Start Frontend Server.
3. Open browser and navigate to:

   ```
   http://localhost:3000
   ```

5. FOLDER STRUCTURE

5.1 Client (React Frontend Structure) :

The frontend of the ResolveNow application is developed using **React.js** and follows a component-based architecture. The project is organized into folders based on roles and common functionalities to improve scalability and maintainability.



Frontend Description

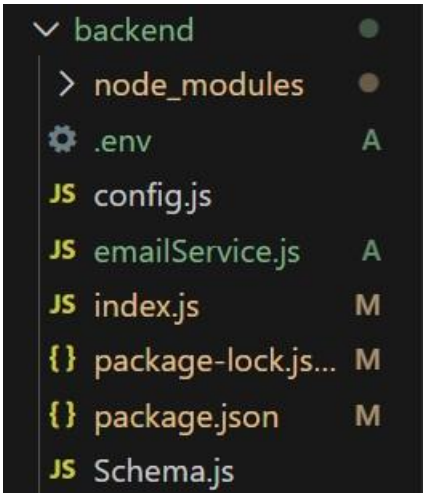
- **components/admin** – Contains dashboards and management pages for Admin role.
- **components/agent** – Includes Agent dashboard for handling assigned complaints.
- **components/user** – Handles complaint submission, tracking, and user dashboard.
- **components/common** – Contains shared pages like Login, Signup, Home, Footer.

- **App.js** – Defines routing using React Router.
- **Images** – Stores UI assets used in the interface.

This structure ensures separation of roles and improves readability of the application.

5.2 Server (Node.js Backend Structure) :

The backend of the application is developed using **Node.js and Express.js**. It follows a modular structure to manage API routes, database models, and configurations.



Backend Description

- **index.js** – Main server entry file containing Express routes, APIs, and server configuration.
- **config.js** – Handles MongoDB database connection setup.
- **Schema.js** – Defines MongoDB schemas including User, Complaint, AssignedComplaint, and Message models.
- **emailService.js** – Implements Nodemailer configuration for sending email notifications.
- **.env** – Stores sensitive credentials like email username and app password.
- **package.json** – Manages dependencies and project scripts.

The backend structure supports REST APIs, complaint processing logic, real-time updates, and email notification services.

6. RUNNING THE APPLICATION

The ResolveNow – Online Complaint Registration and Management System is developed using a MERN stack architecture. The frontend and backend servers must be started separately to run the application locally.

Starting the Backend Server

1. Open Visual Studio Code and navigate to the project root directory.
2. Move to the backend folder using the terminal.

```
cd backend
```

3. Install dependencies (only first time setup):

```
npm install
```

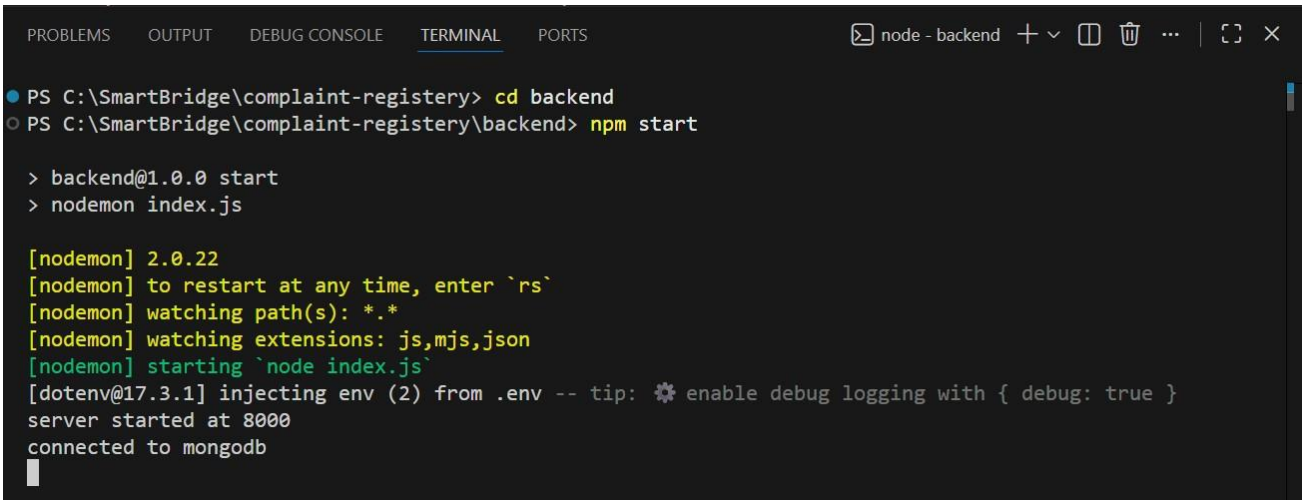
4. Start the backend server:

```
npm start
```

After successful execution, the server runs on:

http://localhost:8000

The backend connects to MongoDB and exposes REST APIs for authentication, complaint management, messaging, and email notifications.



Starting the Frontend Application

1. Open a new terminal and navigate to the frontend folder:

```
cd frontend
```

2. Install frontend dependencies:

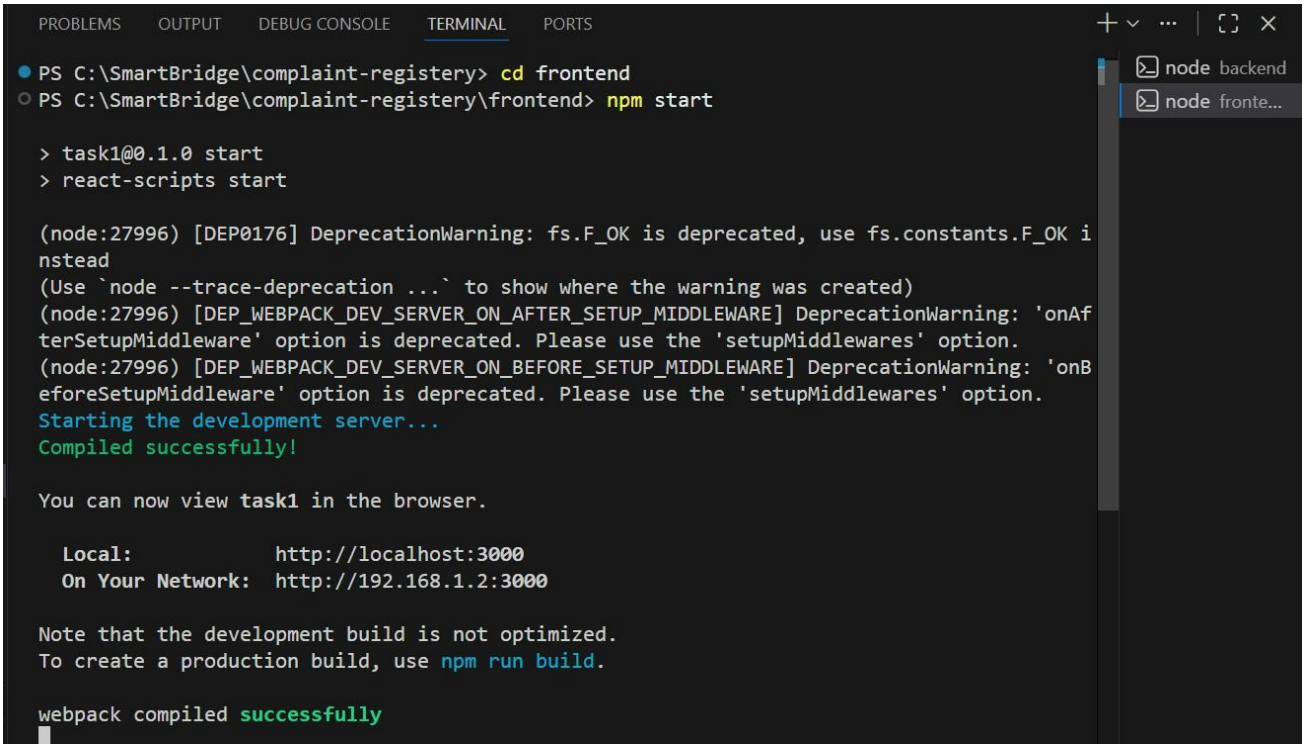
```
npm install
```

3. Run the React development server:

```
npm start
```

The application will launch automatically in the browser at:

http://localhost:3000



Application Execution Flow

- The React frontend sends API requests using Axios.
- Node.js and Express.js handle the request processing.
- MongoDB stores and retrieves application data.
- Nodemailer sends email notifications when user actions occur.

7. API DOCUMENTATION

The backend of the ResolveNow application exposes RESTful APIs developed using Node.js and Express.js. These APIs manage user authentication, complaint handling, messaging, agent assignment, and email notification workflows.

1. User Authentication APIs

User Registration

Endpoint: `/SignUp`

Method: `POST`

Description: Registers a new user in the system.

Request Body Example:

```
{
  "name": "Durga",
  "email": "Durga1234@gmail.com",
  "password": "123456",
  "phone": "9876543210",
  "userType": "Ordinary"
}
```

Response:

```
{
  "_id": "userId",
  "name": "Durga",
  "email": "Durga1234@gmail.com"
}
```

User Login

Endpoint: `/Login`

Method: `POST`

Description: Authenticates users and returns role-based data.

Request Body:

```
{
  "email": "Durga1234@gmail.com",
  "password": "123456"
}
```

Response:

```
{
  "_id": "userId",
  "userType": "Ordinary"
}
```

2. Complaint Management APIs

Submit Complaint

Endpoint: `/Complaint/:id`

Method: `POST`

Description: Allows users to register complaints.

Parameters:

- `id` – User ID

Request Body Example:

```
{
  "name": "John",
  "city": "Hyderabad",
  "comment": "Product issue",
  "status": "Pending"
}
```

Get User Complaint Status

Endpoint: `/status/:id`

Method: `GET`

Description: Fetch complaints created by a specific user.

Get All Complaints (Admin)

Endpoint: `/status`

Method: `GET`

Description: Retrieves all complaints for admin dashboard.

Update Complaint Status (Agent)

Endpoint: `/complaint/:complaintId`

Method: `PUT`

Description: Updates complaint resolution status.

Request Body:

```
{
  "status": "Resolved"
}
```

3. Admin Management APIs

Fetch Agent Users

Endpoint: `/AgentUsers`

Method: `GET`

Returns list of all agents.

Fetch Ordinary Users

Endpoint: `/OrdinaryUsers`

Method: `GET`

Returns list of all ordinary users.

Delete Ordinary User

Endpoint: `/OrdinaryUsers/:id`

Method: `DELETE`

Deletes user and related complaints.

Update User Profile

Endpoint: `/user/:_id`

Method: `PUT`

Updates user details.

4. Agent Complaint APIs

Assign Complaint

Endpoint: `/assignedComplaints`

Method: `POST`

Assigns complaint to agent.

Get Assigned Complaints

Endpoint: `/allcomplaints/:agentId`

Method: `GET`

Fetches complaints assigned to a specific agent.

5. Messaging APIs (Chat Feature)

Send Message

Endpoint: `/messages`

Method: `POST`

Used for user-agent communication.

Request Body:

```
{
  "name": "Agent",
  "message": "We are working on your issue",
  "complaintId": "123"
}
```

Fetch Messages

Endpoint: `/messages/:complaintId`

Method: `GET`

Retrieves chat history related to a complaint.

6. Email Notification Integration

Email notifications are triggered during:

- User Registration
- Complaint Submission
- Complaint Status Update

These APIs internally call the email service configured using Nodemailer.

8. AUTHENTICATION

Authentication and Authorization Overview

The ResolveNow system implements a role-based authentication mechanism to control access for different types of users, including Admin, Agent, and Ordinary Users. Authentication is handled through backend API validation using Node.js and Express.js, where user credentials are verified against records stored in the MongoDB database.

During login, the frontend sends the user's email and password to the backend `/Login` API. The server validates the credentials by matching them with the User Schema stored in MongoDB. Once authentication is successful, the backend returns user details including the `userType`, which determines role-based authorization within the application.

Authorization Mechanism

Authorization is implemented using **role-based routing** in the React frontend. After successful login:

- Admin users are redirected to the **Admin Dashboard**.
- Agent users are redirected to the **Agent Dashboard**.
- Ordinary users are redirected to the **User Homepage**.

The application stores user information in **localStorage**, which is used to maintain the login state and control access to protected routes.

Example:

```
localStorage.setItem("user", JSON.stringify(res.data));
```

The `App.js` routing logic checks if a user exists in localStorage before allowing access to dashboard pages.

Token and Session Handling

The current implementation does not use JWT tokens or server-side session management. Instead, it relies on client-side storage (localStorage) to maintain authentication state during the user session. This approach simplifies development while still enabling role-based access control within the application.

Security Considerations

- User credentials are validated through backend APIs.
- Access to dashboard routes is restricted based on login status.
- Email notifications are sent securely using environment variables stored in the `.env` file.

9. USER INTERFACE

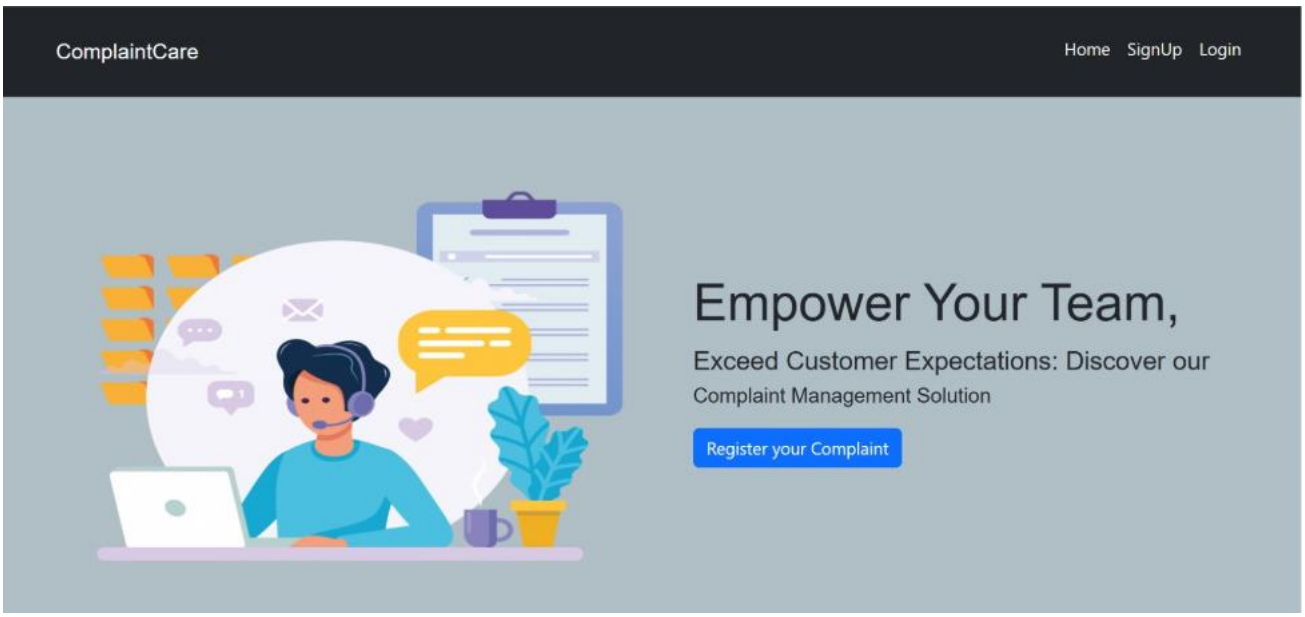


Fig 1: Home Page

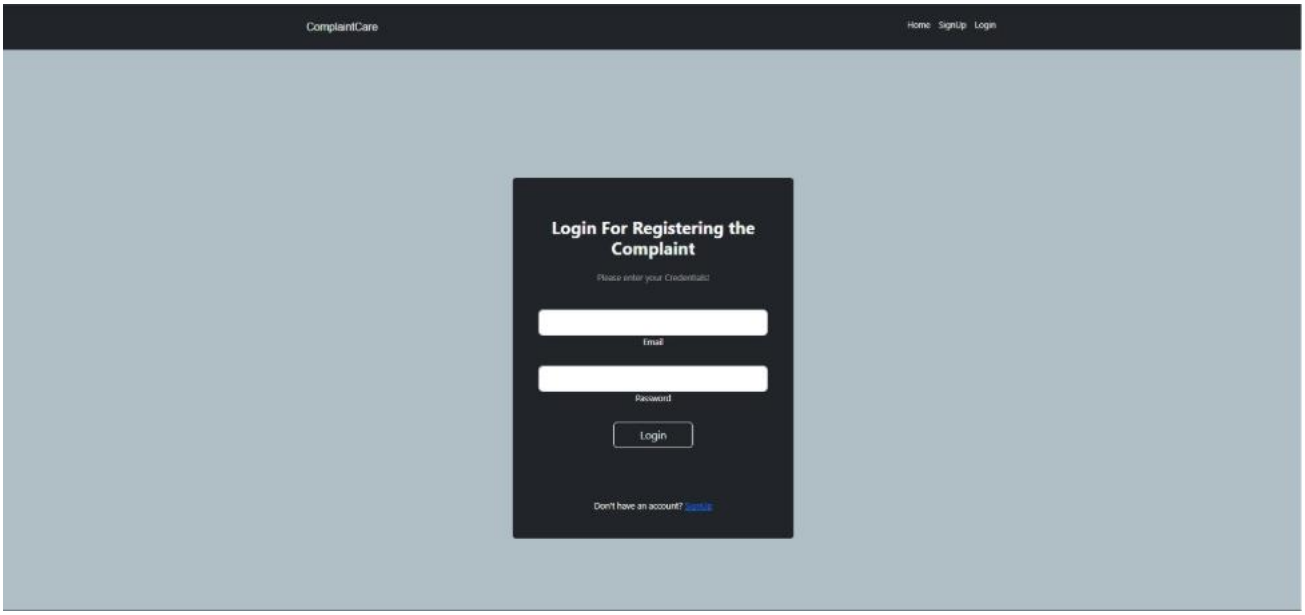


Fig 2: Login Page

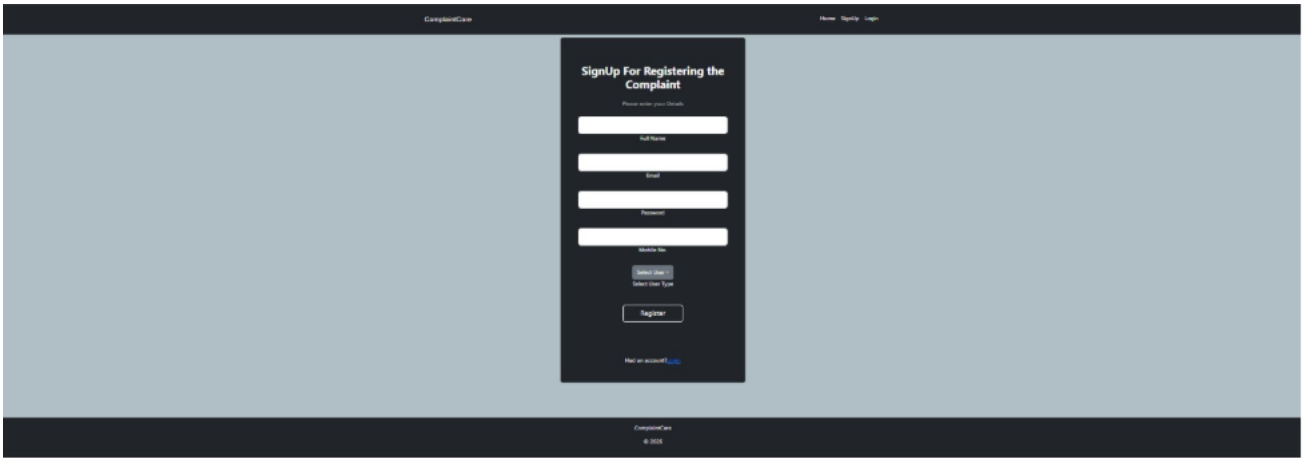


Fig 3: Registration Form

Fig 4: User Page

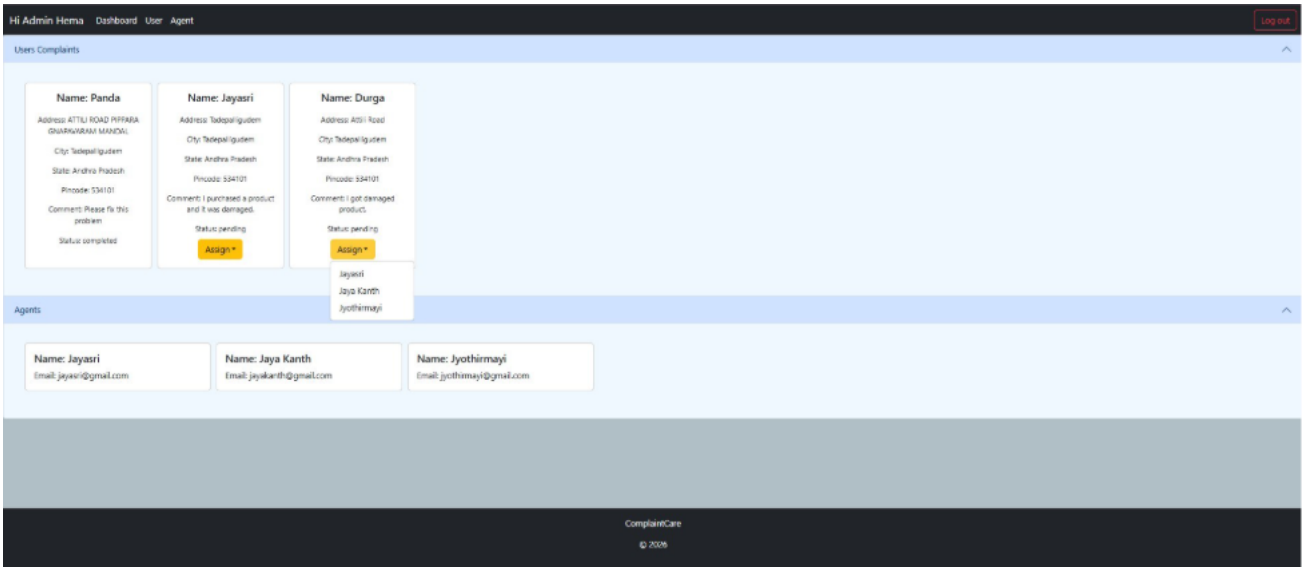


Fig 5: Admin Page

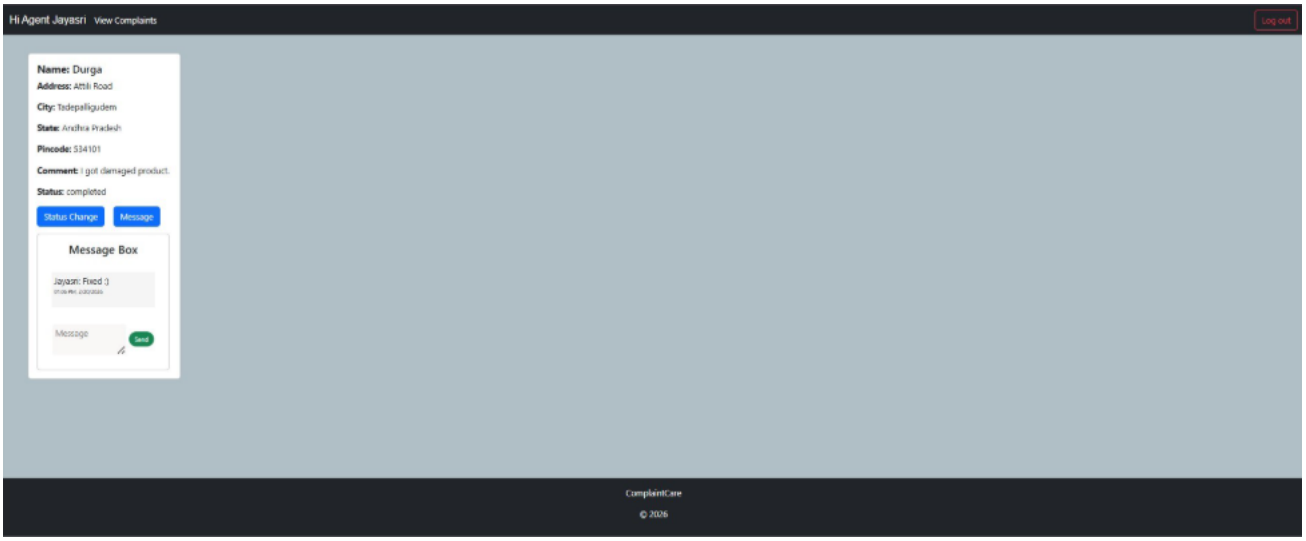


Fig 6: Agent Page

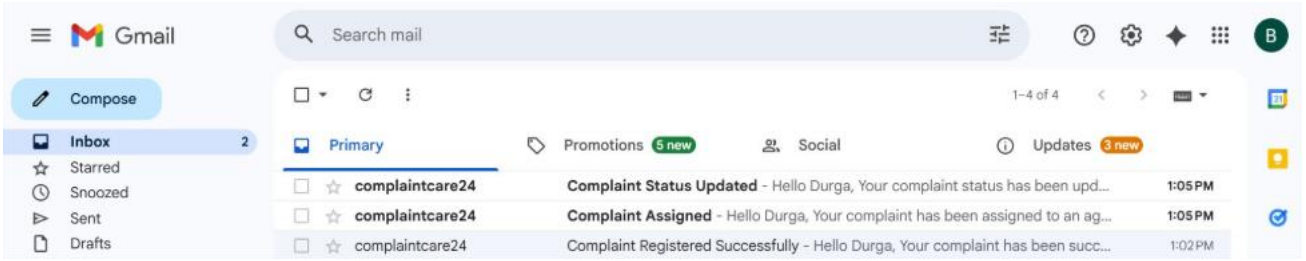


Fig 7: Email Notifications

10. TESTING

Testing Strategy

The ResolveNow – Online Complaint Registration and Management System was tested using a combination of functional testing and manual testing to ensure that all modules operate correctly. The testing process focused on validating user authentication, complaint submission, role-based dashboards, email notifications, chat functionality, and complaint status updates.

Each module was tested individually and then verified through end-to-end testing to ensure proper communication between the frontend, backend, and database. The testing approach ensured that all workflows — including user registration, complaint assignment, and resolution — functioned without errors.

Testing Tools and Methods

- **Manual Testing:**
  Used to verify user interface functionality, navigation flow, and role-based access control.
- **Browser Testing:**
  The application was tested on modern browsers such as Google Chrome to ensure responsive UI behavior.
- **API Testing:**
  Backend REST APIs were tested through frontend requests using Axios and monitored through terminal logs.
- **Database Testing:**
  MongoDB Compass was used to validate stored data such as users, complaints, assigned complaints, and messages.
- **Email Notification Testing:**
  Nodemailer integration was tested by triggering events such as user registration and complaint status updates to verify successful email delivery.

Test Scenarios Covered

- User registration and login validation.
- Admin assigning complaints to agents.
- Agent updating complaint status.

- User tracking complaint progress.
- Real-time chat communication.
- Email notifications triggered on system events.

11. SCREENSHOTS OR DEMO

Project Demo Link :

https://drive.google.com/file/d/1rvmn2YABKr0aLgoNstAIIIIgngnjjYnZ/view?usp=drivesdk

Project Github Link :

https://github.com/hemadurgakokkirala/ResolveNow-Online-Complaint-Registration-and-Management-System

12. KNOWN ISSUES

During development and testing of the ResolveNow – Online Complaint Registration and Management System, a few known limitations and minor issues were identified. These do not affect the core functionality of the application but may require improvement in future updates.

- The current authentication mechanism uses localStorage instead of token-based authentication such as JWT, which may require enhancement for higher security in production environments.
- Email notifications depend on Gmail App Password configuration; incorrect environment variable setup may prevent emails from being sent.
- Form validation is primarily handled on the frontend, and additional backend validation can be implemented for stronger data security.
- Real-time updates rely on manual refresh in some dashboard sections, which can be improved by integrating live socket-based updates.
- The application is optimized for desktop browsers, and mobile responsiveness can be further enhanced.
- Error handling messages are basic and can be improved to provide more user-friendly feedback.

Despite these minor issues, the system performs all required functionalities including user authentication, complaint management, role-based dashboards, chat communication, and email notifications successfully.

13. FUTURE ENHANCEMENTS

The ResolveNow – Online Complaint Registration and Management System has been designed with scalability in mind, allowing several enhancements to be implemented in future versions to improve performance, usability, and security.

- **Advanced Authentication:**
  Implement JWT-based authentication and role-based middleware to enhance application security and session management.
- **Mobile Application Support:**
  Develop a mobile-friendly version or dedicated Android/iOS application to allow users and agents to manage complaints on the go.
- **AI-Based Complaint Categorization:**
  Integrate machine learning algorithms to automatically categorize and prioritize complaints based on keywords and urgency.
- **Real-Time Notifications:**
  Enhance the notification system by adding push notifications and real-time alerts using WebSockets instead of manual refresh.
- **Analytics Dashboard:**
  Introduce reporting and analytics features for admins to monitor complaint trends, agent performance, and system usage.
- **Cloud Deployment:**
  Deploy the application using cloud platforms such as AWS or Azure to improve scalability, availability, and performance.
- **Improved UI/UX Design:**
  Enhance user experience with better responsiveness, accessibility improvements, and modern design components.

These enhancements will improve system efficiency, provide better user engagement, and prepare the platform for enterprise-level deployment.