# FULL STACK DEVELOPMENT WITH MERN

## PROJECT DOCUMENTATION

**1. Introduction**

**Project Title:**

**Resolve Now: Your Platform for Online Complaints**

**Team Members:**

This project has been individually developed and maintained by MS **Sumathi Adimulam**, serving as the sole developer and project lead. The responsibilities included frontend and backend development, database design, API integration, testing, and deployment.

**Project Summary:**

Resolve Now is a full-stack MERN (MongoDB, Express.js, React, Node.js) web application designed to streamline the process of filing, managing, and resolving public complaints in a systematic and efficient manner. It provides a centralized platform where users can easily report issues they face in their localities (such as civic problems, public grievances, etc.) and track their resolution status through a responsive web interface.

This system bridges the gap between the public, complaint management teams (agents), and administrators. It ensures transparency, accountability, and improved service delivery by allowing proper communication and complaint tracking.

**Motivation Behind the Project:**

In many organizations or municipalities, the traditional way of handling complaints is slow, unorganized, and lacks visibility. Citizens often feel unheard, and administrators find it difficult to assign and track the resolution process. This inspired the creation of a digital complaint management system that is intuitive, efficient, and user-friendly.

Resolve Now simplifies complaint registration by allowing users to:

* Submit a complaint online with all relevant details.
* View their complaint status in real time.
* Communicate with complaint handlers (agents).

At the same time, it helps administrators:

* View all submitted complaints.
* Assign complaints to agents.
* Monitor complaint status and take action as required.

Agents have their dedicated dashboard to:

* View assigned complaints.
* Update the resolution status.
* Track resolved and pending issues.

**Why Resolve Now?**

* Promotes transparency in complaint handling.
* Reduces response time through digital automation.
* Provides dashboards tailored to each role (User, Agent, Admin).
* Can be extended to various public and private organizations.

**2. Project Overview**

**Purpose:**

The primary purpose of **Resolve Now** is to provide an online platform that simplifies and streamlines the complaint registration and resolution process for users, administrators, and field agents. It is designed to eliminate the need for manual complaint handling by digitizing the entire workflow—from complaint filing to final resolution.

In many municipalities, institutions, or organizations, complaint handling is often inefficient due to paperwork, poor tracking systems, and lack of transparency. ResolveNow aims to bridge this gap by offering a centralized system that allows citizens to report grievances and track them, while enabling administrators and agents to manage and respond to complaints in a structured and timely manner.

This platform empowers users, ensures accountability from the authorities, and creates a transparent system that keeps everyone informed and engaged in the complaint resolution process.

**Features:**

Resolve Now offers a rich set of features, tailored for three primary roles: **User**, **Admin**, and **Agent**. Below is a breakdown of the core functionalities:

🔹 **For Users:**

* Register and log in to the platform securely.
* File new complaints with details such as title, description, address, city, and pin code.
* View and track the status of all submitted complaints.
* Edit user profile or log out when needed.
* User-friendly dashboard for complaint history and updates.

🔹 **For Admin:**

* View all complaints submitted by users.
* Assign each complaint to a specific agent for resolution.
* Monitor complaint progress (Pending, Resolved, Rejected).
* Manage agent accounts and system users.
* Centralized admin dashboard with complaint statistics and filtering.

🔹 **For Agents:**

* View only the complaints assigned to them.
* Update the status of complaints to “Pending”, “Resolved”, or “Rejected”.
* Track the history of resolved and pending complaints.
* Agent dashboard showing task queue and complaint details.

🔹 **Technical Features:**

* Authentication using JWT Tokens.
* Role-based access control for User, Admin, and Agent.
* RESTful API design using Express.js.
* Fully responsive frontend with React and Bootstrap.
* MongoDB schema for scalable data storage and relationship management.
* Secure communication between frontend and backend using Axios.

This module-based structure makes Resolve Now a powerful and flexible system that can be scaled and adapted to different institutions, organizations, or government bodies in need of a public complaint management solution.

**3. System Architecture**

The architecture of **Resolve Now** follows a standard **MERN Stack** model, leveraging **MongoDB**, **Express.js**, **React.js**, and **Node.js** to ensure scalability, modularity, and performance. The system is designed using a **client-server architecture** where the frontend and backend communicate via REST APIs.

**Components:**

🟢 **Frontend – React.js**

* Built using **React.js** with components for login, dashboard, complaint filing, and role-based views.
* Integrated with **Axios** to make API requests.
* Uses **Bootstrap/React-Bootstrap** for responsive and clean UI design.
* Handles routing using **React Router**.

🟢 **Backend – Node.js with Express**

* Implements RESTful API endpoints to handle user authentication, complaint operations, and agent/admin workflows.
* Uses **Express.js** for routing and middleware integration.
* Handles JWT-based authentication and error handling.

🟢 **Database – MongoDB with Mongoose**

* Stores user data, complaints, and agent assignments in collections.
* Designed using **Mongoose schemas** that reflect relationships like:
  + User–Complaint (One-to-Many)
  + Complaint–Agent (Many-to-One)
* Flexible document-based storage to accommodate scaling.

🟢 **Authentication – JWT (JSON Web Token)**

* Each user, upon login, receives a JWT stored in localStorage.
* Tokens are used to authorize users for protected routes and APIs.
* Role-based middleware ensures only authorized roles access relevant data.

**Data Flow Summary:**

1. **User logs in** → Token is stored locally.
2. **User submits a complaint** → Sent to backend with token → Stored in MongoDB.
3. **Admin views all complaints**, assigns to agents.
4. **Agent views assigned complaints** → Updates status accordingly.
5. **User can view complaint status updates** on their dashboard.

This clean separation of concerns makes the system easy to maintain, test, and expand. It ensures security through role-based authentication and improves performance using async requests and modular architecture.

**4.SetUp Instructions**

This section explains how to set up and run the **Resolve Now** complaint management system on a local machine using the MERN stack (MongoDB, Express, React, Node.js).

**✅ 1. Prerequisites**

Before setting up the project, ensure the following software is installed:

* **Node.js** and **npm**  
  Download from: <https://nodejs.org>
* **MongoDB**  
  Use [MongoDB Atlas](https://www.mongodb.com/cloud/atlas) or install it locally.
* **Git**  
  Download from: <https://git-scm.com>

**📁 2. Folder Structure**

After downloading/cloning the project, the folder structure will look like:

complaint-registery/

│

├── backend/ ← Express server and MongoDB logic

├── frontend/ ← React frontend interface

└── README.md

**⚙️ 3. Backend Setup (Express + MongoDB)**

cd complaint-registery/backend

npm install # Install server dependencies

* Create a .env file inside the backend folder with the following content:

env

PORT=5000

MONGO\_URI=your\_mongodb\_connection\_string

JWT\_SECRET=your\_jwt\_secret\_key

Replace your\_mongodb\_connection\_string with a valid MongoDB URI.

* Start the backend server:

npm start

You should see:  
✅ Server running on port 5000  
✅ MongoDB connected

**💻 4. Frontend Setup (React)**

cd ../frontend

npm install # Install client dependencies

npm start

This will open the frontend in your browser at:  
http://localhost:3000

**🧪 5. Testing the Setup**

* **User Role**: Register, login, and file a complaint.
* **Admin Role**: Login and view all complaints, assign agents.
* **Agent Role**: Login and update the status of assigned complaints.

**📌 Notes:**

* Ensure MongoDB is running or the Atlas URI is correct.
* Token-based authentication is implemented using JWT.
* Roles (user, admin, agent) are assigned during user creation or manually in the database.

**📁 Project Folder Structure — Resolve Now**

Resolve Now/

│

├── 📁 backend/ # Express.js server-side code

│ ├── 📁 config/ # MongoDB configuration

│ │ └── db.js

│ ├── 📁 controllers/ # Route handler logic (users, complaints, auth)

│ │ ├── authController.js

│ │ ├── complaintController.js

│ │ └── userController.js

│ ├── 📁 middleware/ # Custom middleware (auth protection, error handling)

│ │ └── authMiddleware.js

│ ├── 📁 models/ # Mongoose schemas and models

│ │ ├── User.js

│ │ └── Complaint.js

│ ├── 📁 routes/ # API routes

│ │ ├── authRoutes.js

│ │ ├── complaintRoutes.js

│ │ └── userRoutes.js

│ ├── .env # Environment variables

│ ├── server.js # Entry point for backend

│ └── package. Json # Backend dependencies

│

├── 📁 frontend/ # React.js client-side code

│ ├── 📁 public/ # Static files and HTML template

│ ├── 📁 src/ # Source code

│ │ ├── 📁 components/ # Reusable components (Login, Dashboards, File Complaint etc.)

│ │ │ ├── Login.js

│ │ │ ├── UserDashboard.js

│ │ │ ├── AdminDashboard.js

│ │ │ ├── AgentDashboard.js

│ │ │ └── FileComplaint.js

│ │ ├── App.js # Main React app

│ │ ├── index.js # Entry point for React

│ │ └── App.css # Global styles

│ ├── package. Json # Frontend dependencies

│ └── .env # Frontend environment variables

│

├── 📄 README.md # Project readme

**6.Running the Application**

To run the **Resolve Now** project, you need to start both the **backend server** (Node.js + Express + MongoDB) and the **frontend client** (React.js).

**✅ Prerequisites**

Make sure the following are installed on your system:

* **Node.js** and **npm**
* **MongoDB** (local or cloud like MongoDB Atlas)
* **Git** (optional, for cloning repository)

**🛠️ Step 1: Start the Backend Server**

cd backend

npm install # Install backend dependencies

# Create a .env file in backend folder

# Example .env content:

MONGO\_URI=your\_mongodb\_uri

JWT\_SECRET=your\_jwt\_secret

PORT=5000

npm start # Start the backend server

Once started, it should show:

✅ MongoDB Connected

✅ Server running on port 5000

**💻 Step 2: Start the Frontend (React Client)**

Open a new terminal and run:

cd frontend

npm install # Install frontend dependencies

npm start # Start React app

The frontend will run on:  
[**http://localhost:3000**](http://localhost:3000)

**🧪 Test the App**

1. Go to http://localhost:3000
2. Register and Login as:
   * **User** (to file and view complaints)
   * **Agent** (to resolve assigned complaints)
   * **Admin** (to assign agents)
3. Try features like:
   * Filing complaints
   * Assigning agents
   * Updating status

**📝 Notes**

* If you're using **MongoDB Atlas**, ensure you whitelist your IP address.
* Make sure the backend is running before starting the frontend.
* Use **Postman** to test APIs if needed.

7. **API Documentation — Resolve Now**

This section describes the RESTful API endpoints used in the **Resolve Now** platform. All requests that require authentication must include a valid JWT token in the header:

Authorization: Bearer <token>

**🔐 Authentication APIs**

**✅ POST /API/auth/register**

Register a new user (Admin/User/Agent).

**Request Body:**

json

{

"name": "John Doe",

"email": "john@example.com",

"password": "123456",

"role": "user" // or "agent", "admin"

}

**Response:**

{

"token": "...",

"user": { "name": "John Doe", "email": "john@example.com", "role": "user" }

}

**✅ POST /api/auth/login**

Login an existing user.

**Request Body:**

json

{

"email": "john@example.com",

"password": "123456"

}

**Response:**

{

"token": "...",

"user": { "name": "John Doe", "role": "user" }

}

**📝 Complaint APIs**

**✅ POST /api/complaints**

Create a new complaint (User only).

**Request Header:**

Authorization: Bearer <token>

**Request Body:**

json

{

"title": "Water Leakage",

"description": "Pipe leakage in street 4",

"city": "Hyderabad",

"pin code": "500001",

"address": "Street 4, ABC Nagar"

}

**Response:**

{

"\_id": "...",

"title": "Water Leakage",

"status": "Pending",

"user": "...",

...

}

**✅ GET /api/complaints/user**

Get complaints submitted by the **logged-in user**.

**Response:**

{

"\_id": "...",

"title": "Water Leakage",

"status": "Pending",

...

}

**✅ GET /api/complaints**

Get **all complaints** — Admin only.

**Response:**

{

"\_id": "...",

"title": "...",

"description": "...",

"user": { "name": "John Doe" },

"agent": { "name": "Agent A" },

"status": "Pending"

}

**✅ PUT /api/complaints/assign/:id**

Assign a complaint to an agent — Admin only.

**Request Body:**

{ "agentId": "agent\_id\_here" }

**✅ PUT /api/complaints/status/:id**

Update complaint status — Agent only.

**Request Body:**

{ "status": "Resolved" }

**👨‍💼 User Management APIs**

**✅ GET /api/users/agents**

Get all users with role "agent" — Admin only.

**Response:**

{ "\_id": "...", "name": "Agent A", "email": "..." }

**🔒 Protected Routes**

All complaint-related routes are protected and require the user to be authenticated. Admin-only and agent-only routes should be guarded using middleware (protect, adminOnly, agentOnly).

8. **Authentication – Resolve Now**

Authentication is a critical component of the **ResolveNow: Online Complaint Management System**, ensuring secure access control for different types of users, including **Admin**, **User**, and **Agent**. The system uses **JSON Web Tokens (JWT)** for secure and stateless user authentication.

**🧩 Key Concepts**

**✅ 1. Role-Based Access**

The system supports three user roles:

* **User** – Can file complaints and view their own complaint history.
* **Agent** – Can view assigned complaints and update their status (Pending, Resolved, Rejected).
* **Admin** – Has access to all complaints, can view all users/agents, and assign complaints to agents.

Each role is stored in the user’s database record and checked using middleware during API access.

**✅ 2. Registration**

Users can register via the /api/auth/register endpoint by providing:

* Name
* Email
* Password
* Role (user, agent, or admin)

Password is hashed before storing in the database using **bcrypt.js** for security.

**✅ 3. Login**

Users log in using the /api/auth/login endpoint. Upon successful login:

* A **JWT token** is generated
* Token and user data (name, role) are sent back in the response
* Token is stored in browser localStorage on the frontend

The token is used for all future API requests.

**✅ 4. Token Authentication (JWT)**

The backend uses JWT to:

* Authenticate each request via middleware (protect)
* Identify the user from the token
* Attach user details (like \_id, role) to req.user

If the token is invalid or missing, access is denied.

**✅ 5. Middleware Functions**

* protect: Verifies token and attaches user to request
* adminOnly: Allows only admin to proceed
* agentOnly: Allows only agent to proceed

These middleware functions enforce strict role-based permissions.

**✅ 6. Frontend Handling**

* On login, the token and user details are stored in local storage
* On logout, these values are removed
* Protected routes are conditionally rendered based on role

// Example

if (user.role === "admin") navigate("/admindashboard");

**✅ Security Summary**

* ✅ Passwords hashed with bcrypt
* ✅ Tokens secured with secret keys
* ✅ Middleware-based access control
* ✅ Role-based redirection in frontend

**9.User Interface (UI) – Resolve Now**

The **User Interface (UI)** of Resolve Now has been carefully designed to offer a user-friendly, intuitive experience for all roles – **User**, **Agent**, and **Admin**. Built using **React.js** and styled with **Bootstrap**, the UI focuses on ease of complaint filing, role-based dashboard navigation, and real-time status tracking.

**✅ Key UI Screens**

| **Page** | **Description** |
| --- | --- |
| **Landing Page** | Entry point of the platform with options to Login/Register |
| **Login/Register** | Secure authentication with role-based access |
| **User Dashboard** | Allows users to file complaints and view complaint status |
| **File Complaint** | Complaint form with fields: title, description, city, pin code, address |
| **Agent Dashboard** | View assigned complaints and update statuses (Pending, Resolved, Rejected) |
| **Admin Dashboard** | View all complaints and assign them to agents |
| **Logout** | Clears local storage and redirects to login |

**🎯 UI Technologies Used**

* **React.js** – Component-based frontend structure
* **React Router DOM** – Role-based navigation
* **Axios** – API requests and responses
* **Bootstrap** – Responsive design and styling
* **React Icons** – Clean, modern icon set
* **Local Storage** – Token & user info storage

**10. Testing – Resolve Now**

Resolve Now was tested manually by simulating real user interactions and by verifying backend responses for every API.

**🔍 Test Scenarios & Results**

| **Test Case** | **Expected Result** | **Status** |
| --- | --- | --- |
| User Registration | New user account created successfully | ✅ Pass |
| User Login | Token and role info stored in local storage | ✅ Pass |
| File Complaint | Complaint stored in database with user ID | ✅ Pass |
| View User Complaints | User sees only their own complaints | ✅ Pass |
| Admin View All Complaints | Admin sees all complaints with user & agent | ✅ Pass |
| Assign Agent to Complaint (Admin) | Complaint updates with agent ID | ✅ Pass |
| Agent View Assigned Complaints | Agent sees only their assigned complaints | ✅ Pass |
| Agent Update Complaint Status | Complaint status updates in real-time | ✅ Pass |
| Unauthorized access without token | User receives 401 Unauthorized | ✅ Pass |
| Role-Based Redirection After Login | Navigated to correct dashboard | ✅ Pass |

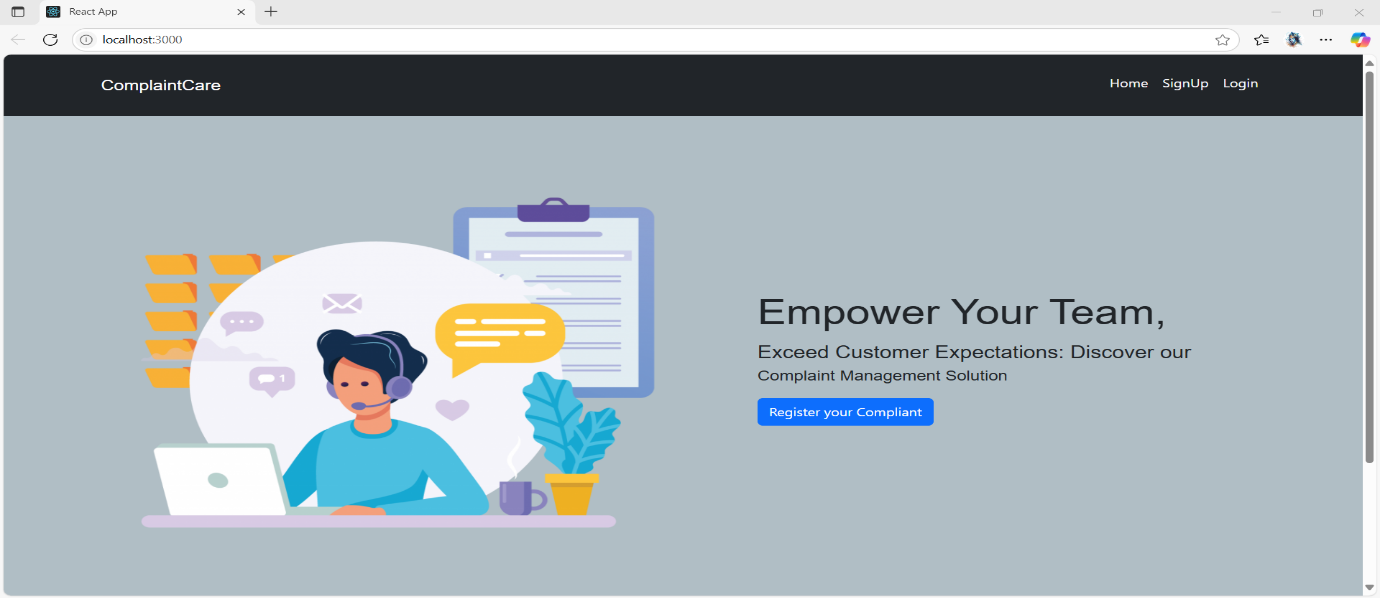
**✅ Browser Compatibility**

Tested on:

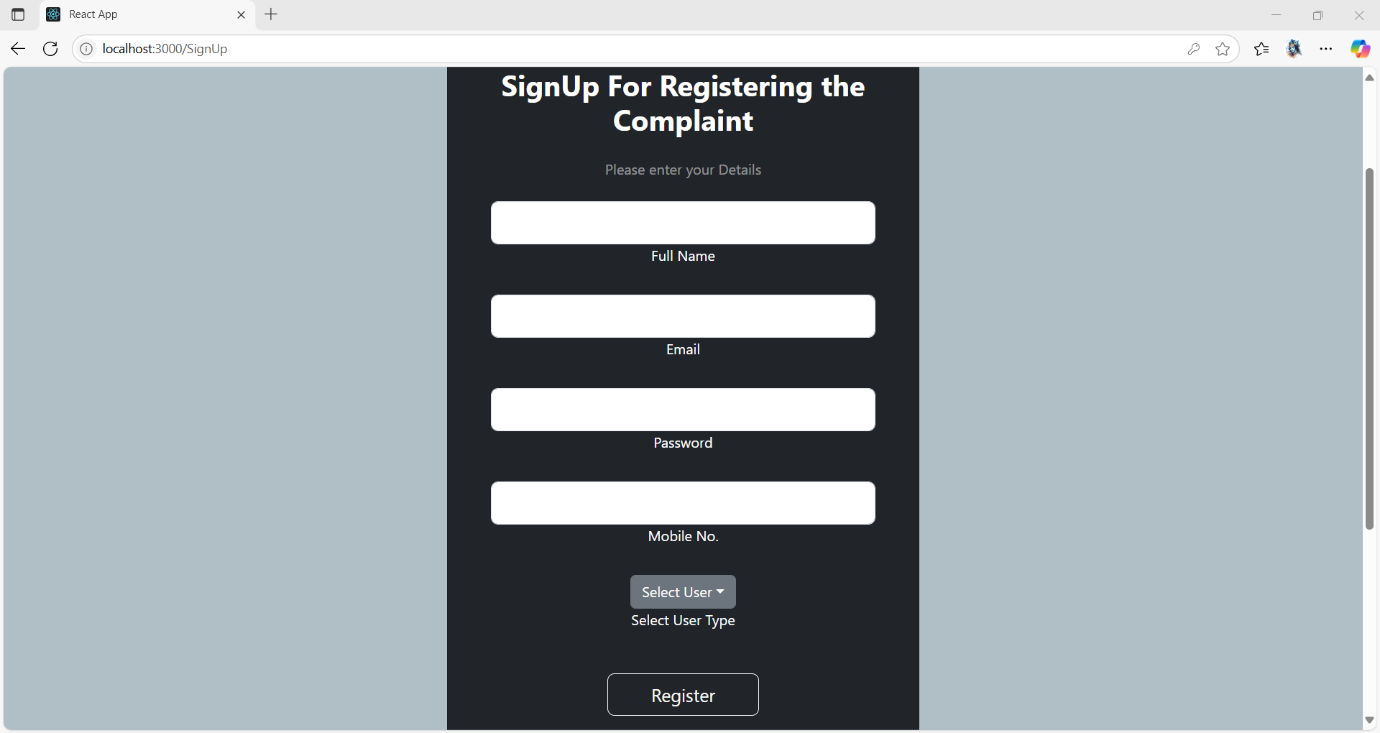
* ✅ Google Chrome
* ✅ Microsoft Edge
* ✅ Mozilla Firefox

11.Screenshots

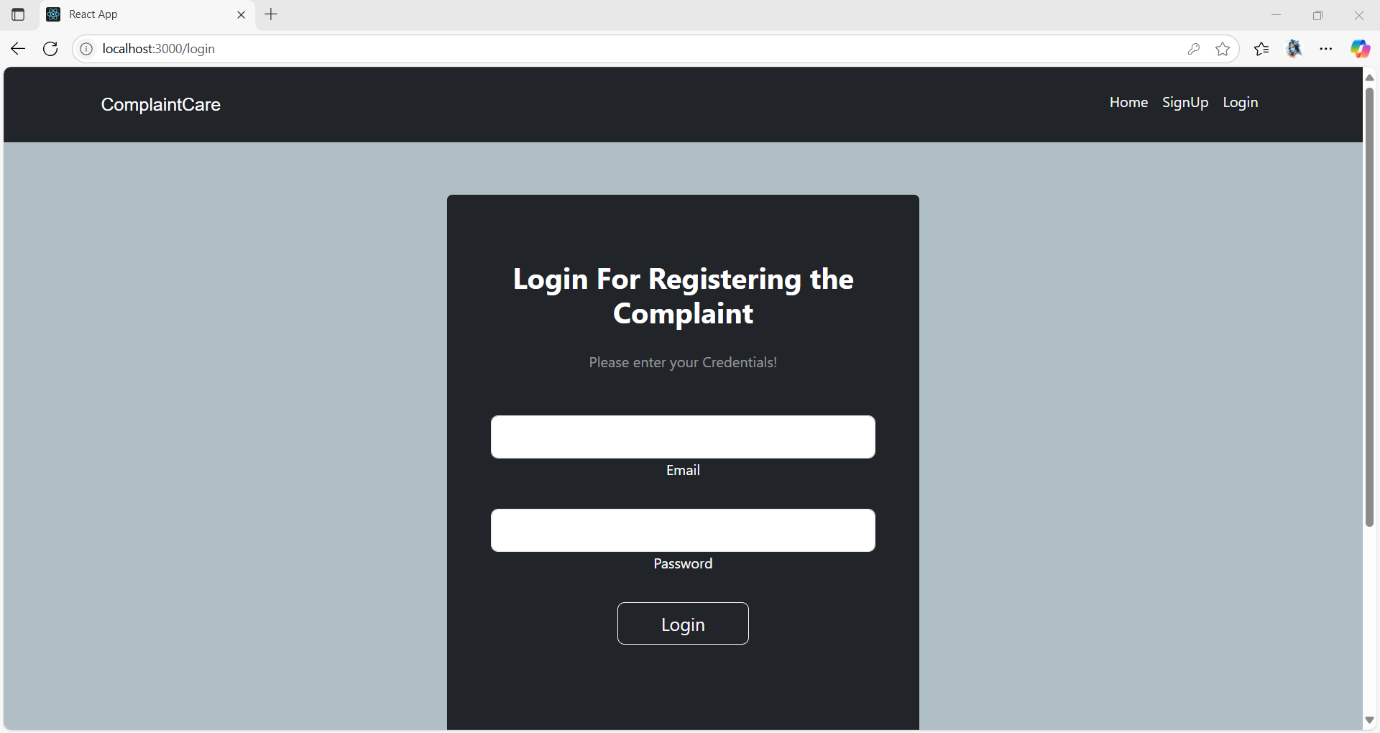
Home Page:



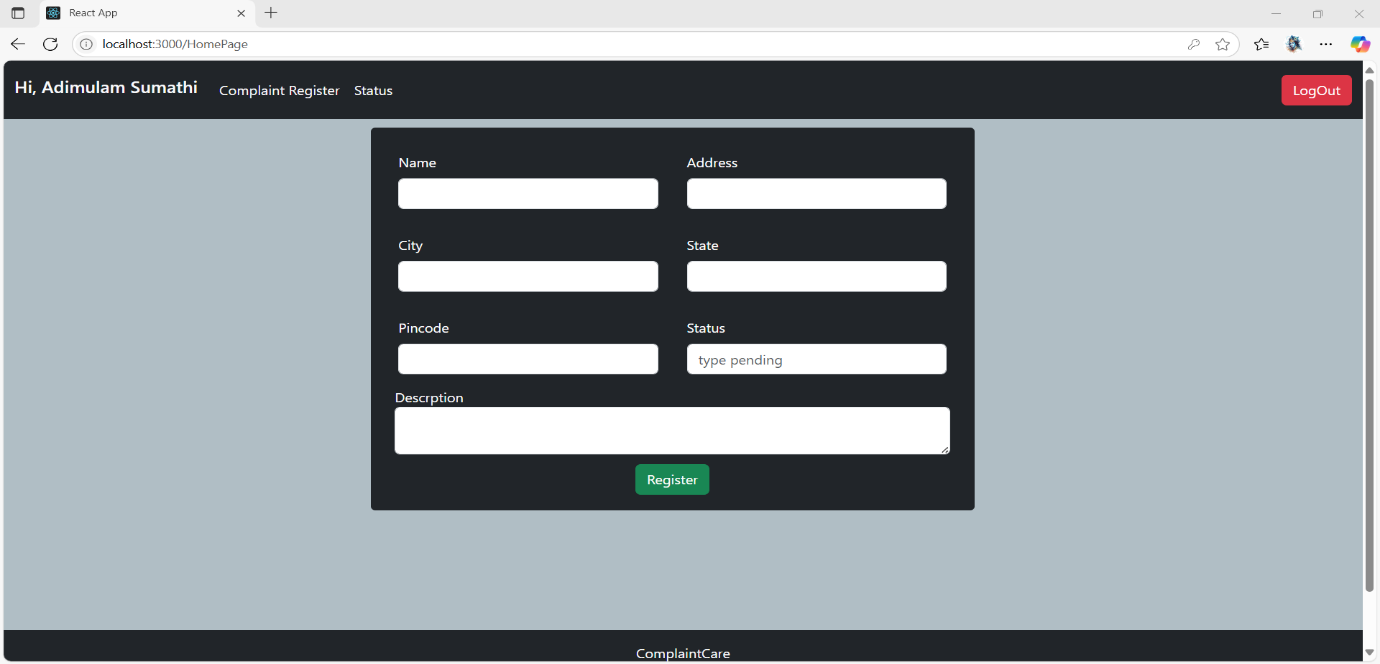
Register Page :

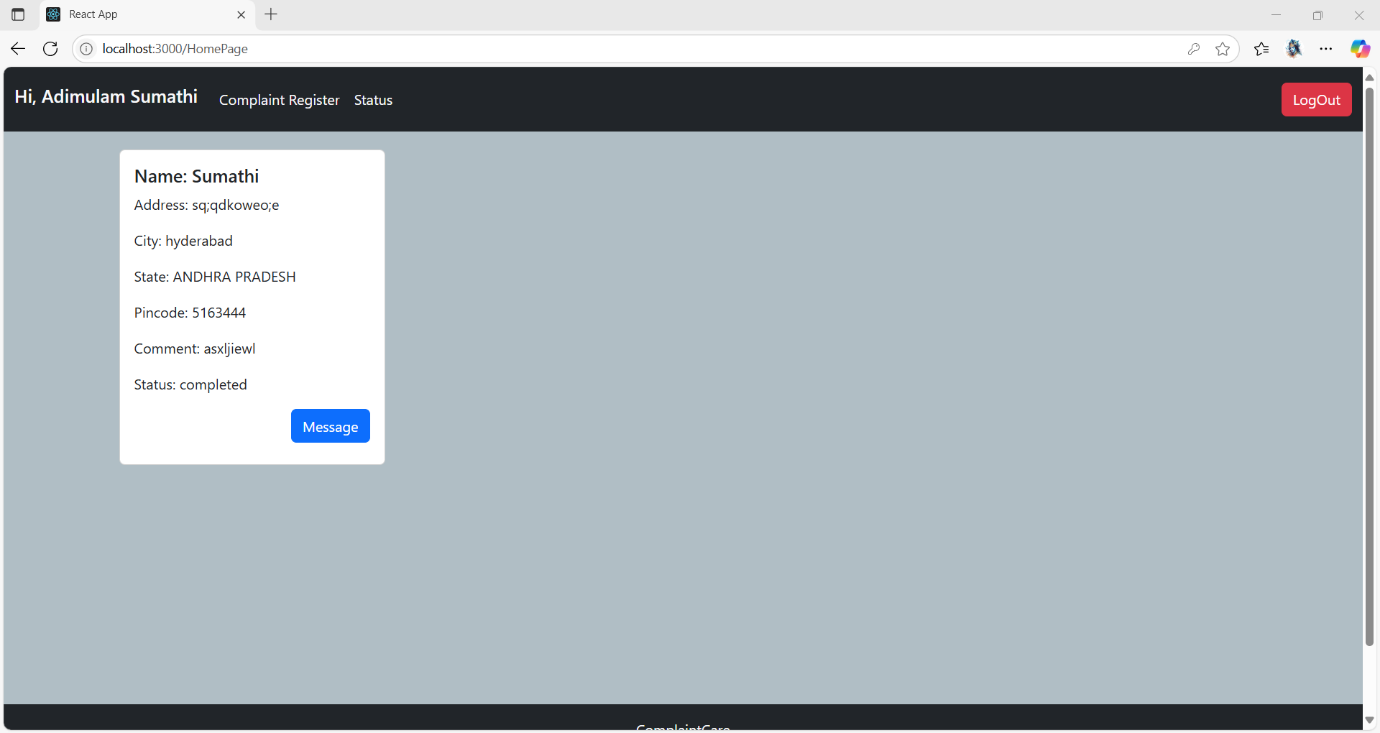


Login Page :

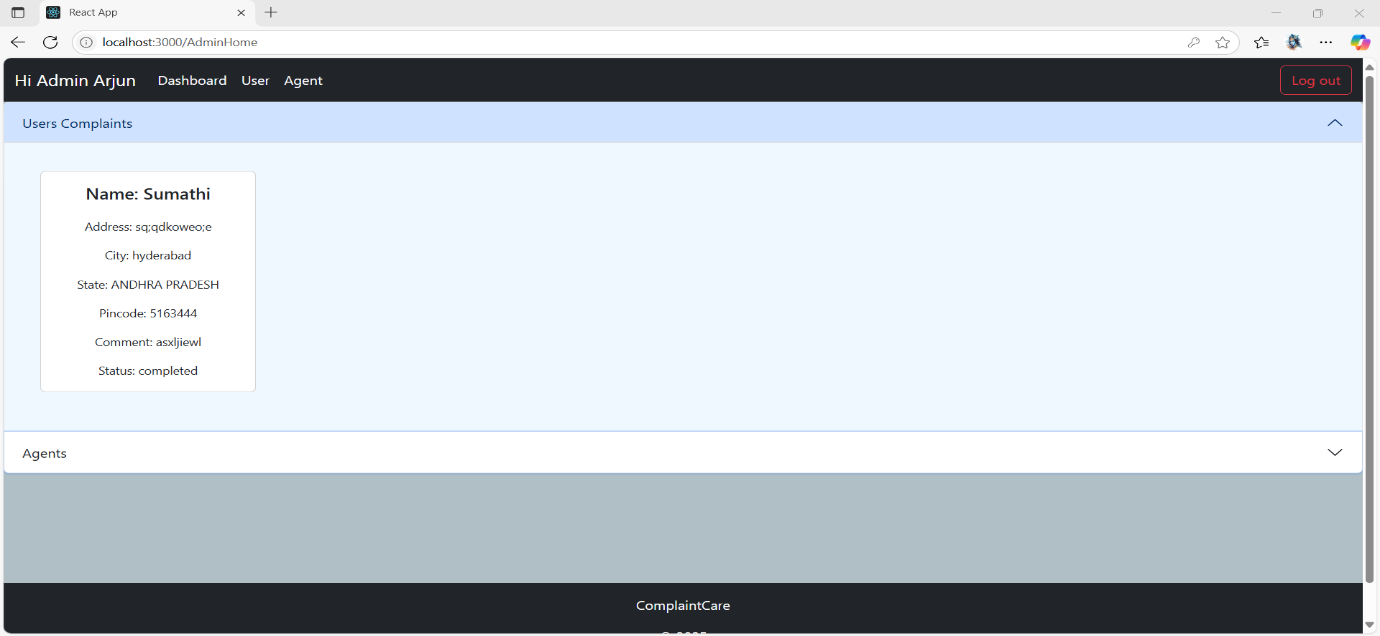


User Dashboard:

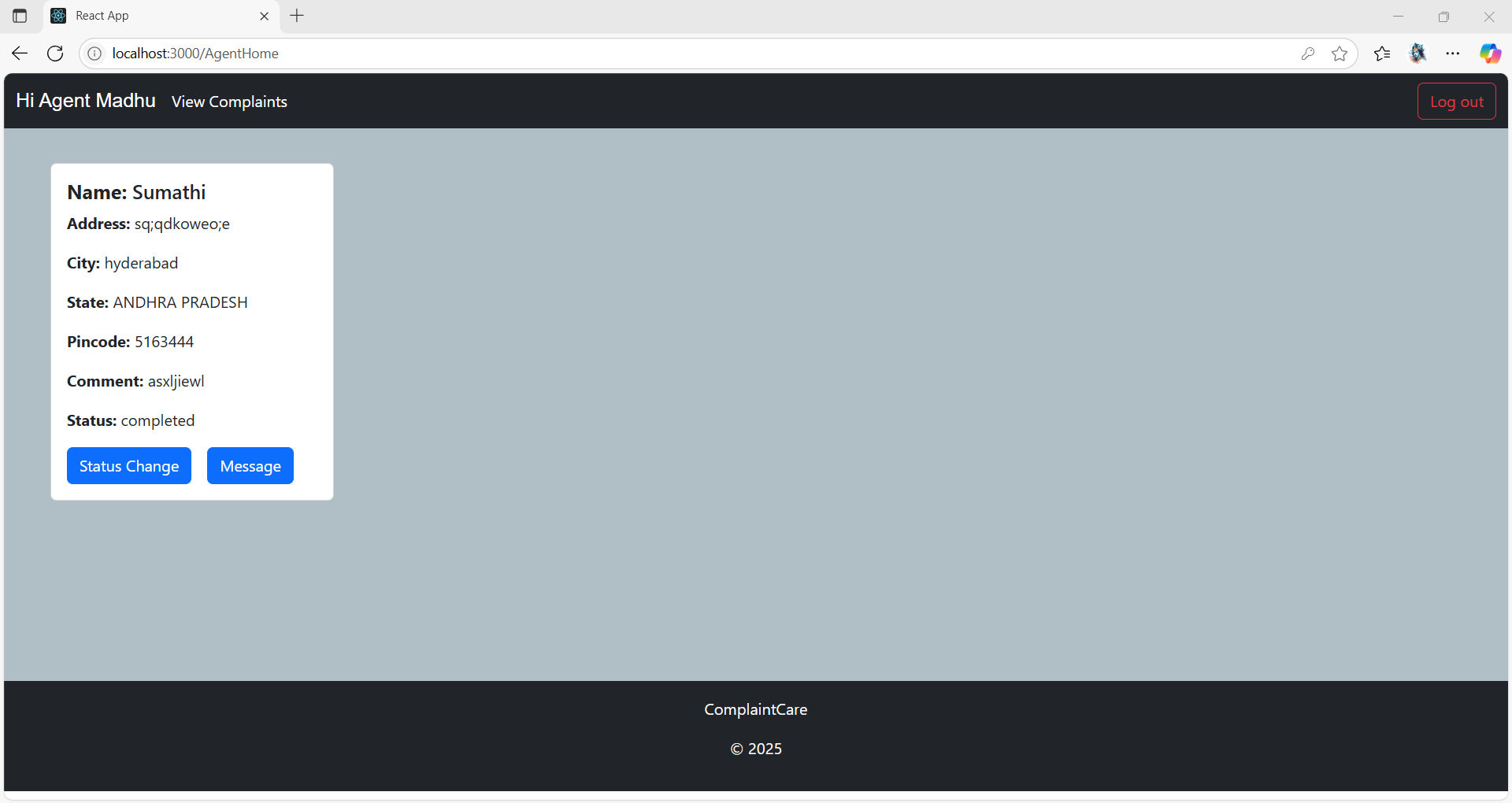




Admin Dashboard:



Agent Dashboard:



**12.Known Issues**

| **Issue No.** | **Description** | **Status** |
| --- | --- | --- |
| 1 | No email notifications sent to users or agents on complaint updates | Known Issue |
| 2 | Form validations are basic; users can enter invalid pincode or empty fields | Known Issue |
| 3 | No image/file upload support in complaint form | Known Issue |
| 4 | Agent dashboard doesn't support sorting/filtering of complaints | Known Issue |
| 5 | Admin has no search functionality to find specific complaints quickly | Known Issue |
| 6 | No pagination implemented in complaints list | Known Issue |
| 7 | Token expiration is not handled gracefully (no auto-logout or refresh) | Known Issue |

**🌟 Future Enhancements**

| **Enhancement** | **Description** |
| --- | --- |
| Email Notifications | Notify users and agents via email when complaint status changes or a new complaint is filed |
| Complaint Attachments | Allow users to upload images, videos, or files as proof or evidence of complaints |
| Role-Based Analytics | Add dashboards with charts showing resolved, pending, and rejected complaints |
| Feedback Mechanism | Let users rate or provide feedback after complaint resolution |
| Chat Support | Enable real-time chat between users and assigned agents |
| Mobile Responsiveness | Improve UI for mobile and tablet users |
| Search & Filter Functionality | Allow filtering of complaints by city, status, or date |
| Auto Logout on Token Expiry | Enhance security by automatically logging out on JWT expiration |
| Admin Reports & Export | Allow admin to download complaint data in Excel/PDF format |
| Push Notifications | Use browser notifications for status updates and alerts |