Full Stack Development with MERN

Online Complaint Registration and Management System

Introduction

Project Title: Online Complaint Registration and Management System
Team Members:

Name	Roles
1. Nikitha Raj - 311121104041	Frontend, Backend Dev
2. Estelle Charles - 311121104018	Frontend, Design
3. Harshavardhini - 311121104026	Frontend, Design
4. Austin Prince Roosevelt - 311121104010	Frontend, Backend Dev

---

Project Overview

Purpose:
The Online Complaint Registration and Management System provides a user-friendly platform for individuals or organizations to register, track, and resolve complaints seamlessly.

Features:

- User authentication and role-based access control.
- Complaint submission with detailed information and document uploads.
- Real-time complaint tracking with email/SMS notifications.
- Messaging feature for user-agent interactions.
- Secure backend for data handling with compliance to data protection regulations.

Architecture

Frontend:

- **Technologies**: React with Material-UI, Ant Design, and Bootstrap for a responsive interface.
- **Libraries**:
    - react-router-dom: Navigation.
    - axios: For RESTful API integration.

Backend:

- **Technologies**: Node.js with Express.js for RESTful API implementation.
- **Functionalities**:
    - Authentication using JWT.
    - Password encryption with bcrypt.js.
    - File handling with Multer for document uploads.

Database:

- **Technology**: MongoDB with Mongoose for schema management.
- **Schema Design**:
    - **Users**: Storing details like name, email, role, and password.
    - **Complaints**: Storing complaint details like title, description, status, and attachments.

Setup Instructions

Prerequisites:

- Node.js
- MongoDB

Installation:

1. **Clone the repository:**
   git clone https://github.com/yourusername/ComplaintSystem.git
   cd ComplaintSystem

2. **Frontend Setup**:

   cd frontend

   npm install

3. **Backend Setup**:

   ```
   cd ../backend
   npm install
   ```

4. Create a .env file in the backend directory:

   ```
   PORT=5000
   MONGO_URI=<your_mongodb_connection_string>
   JWT_SECRET=<your_jwt_secret>
   ```

Folder Structure

Frontend:

```
frontend/
├── public/            # Static assets
├── src/               # Main application code
│   ├── components/    # Reusable UI components
│   ├── modules/       # Features grouped by functionality
│   │   ├── admin/     # Admin-related features
│   │   ├── user/      # User-specific features
│   ├── App.js         # Application entry point
│   └── index.js       # React DOM renderer
├── package.json       # Frontend dependencies
└── README.md          # Project description
```

Backend:

```
backend/
├── models/            # Mongoose schemas
│   ├── User.js
│   ├── Complaint.js
├── routes/            # API route handlers
│   ├── auth.js
│   ├── complaint.js
├── controllers/       # Business logic
│   ├── authController.js
```

```
│   ├── complaintController.js
├── index.js           # Server entry point
└── package.json       # Backend dependencies
```

Running the Application

Frontend:

```
cd frontend
npm start
```

Backend:

```
cd backend
npm start
```

Access the application at:

- Frontend: http://localhost:3000
- Backend: http://localhost:5000

API Documentation

User Management:

- **POST /register**: Register a new user.
- **POST /login**: Login a user.
- **GET /getuserdata**: Fetch user data (auth required).

Complaint Management:

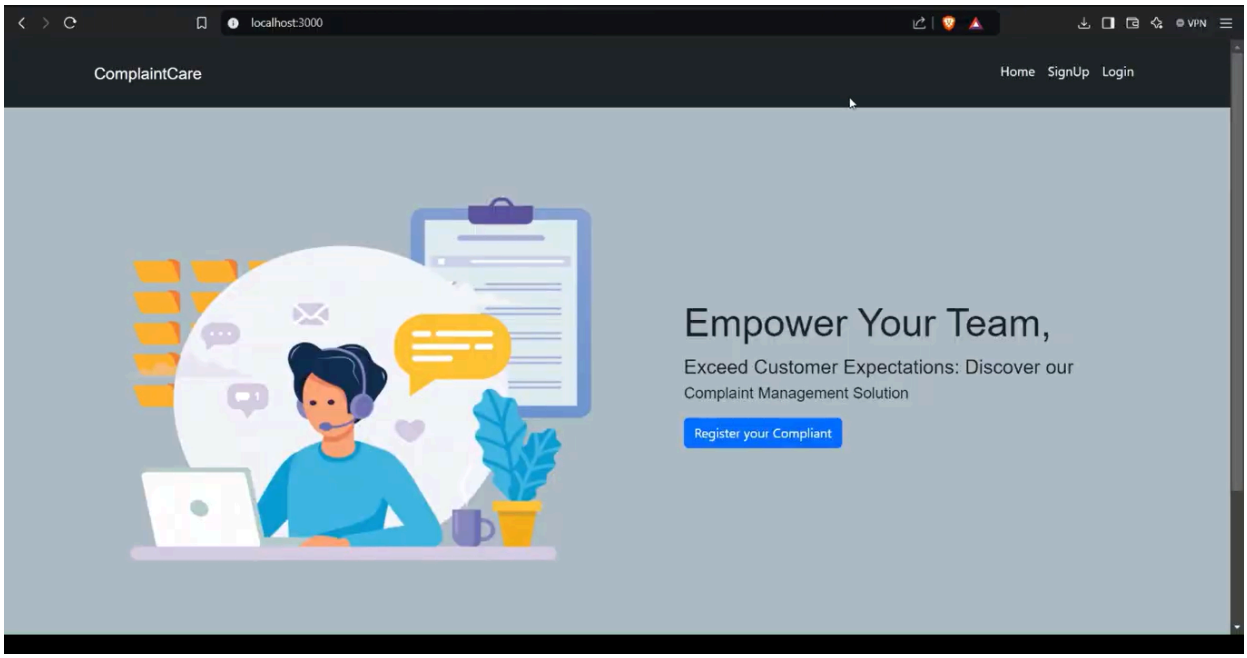
- **POST /submitcomplaint**: Register a complaint (auth required).
- **GET /getallcomplaints**: Fetch all complaints for admin (auth required).
- **PATCH /updatecomplaint/**
  : Update complaint status (auth required).

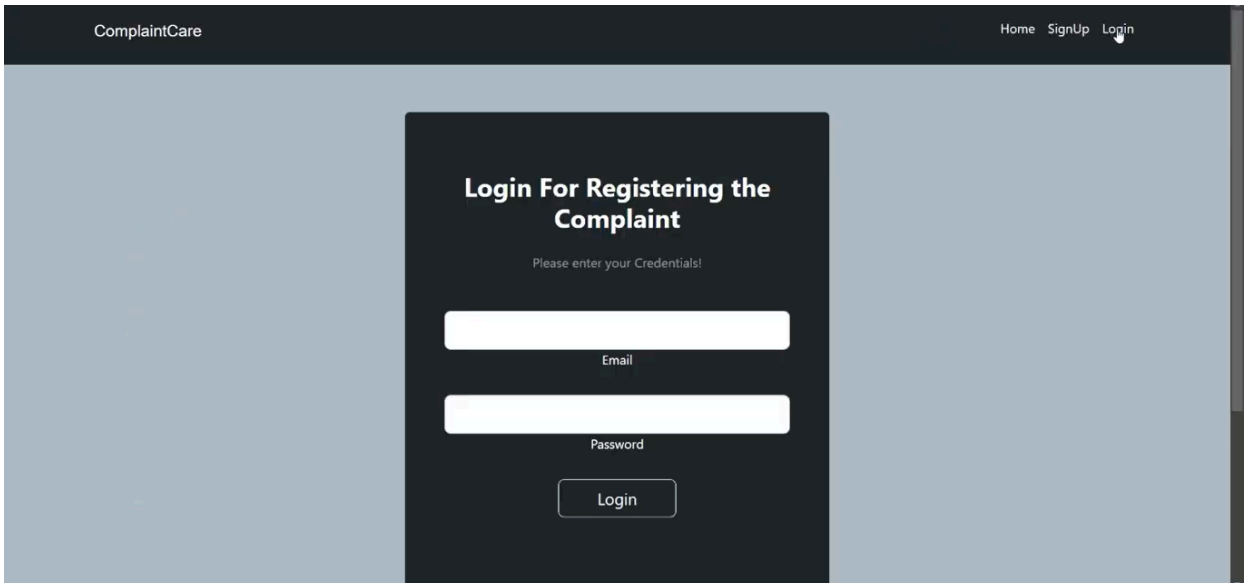
Authentication and Authorization

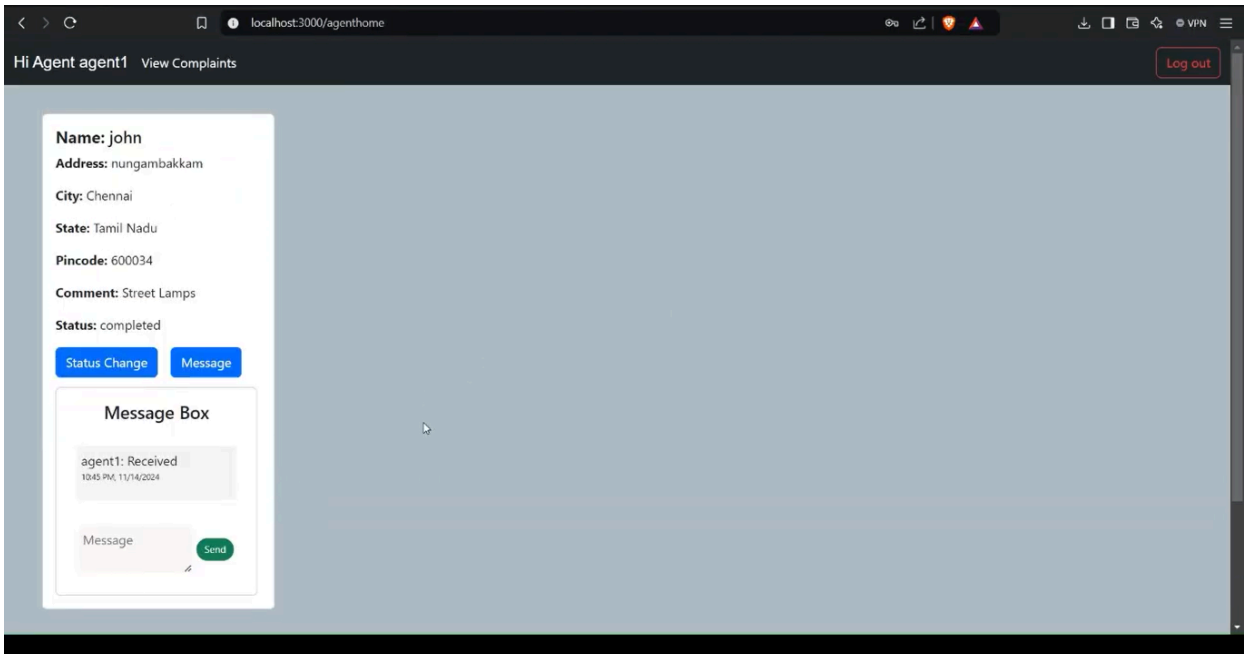
- **Authentication**
  - **JWT-Based Authentication**:
    The project uses **JSON Web Tokens (JWT)** for authentication. After a user logs in or registers, a token is generated using a secret key (process.env.JWT_KEY) and sent to the client.
  - The client includes this token in the Authorization header for subsequent API requests.
- **Authorization**
  - **Middleware Validation**:
    The middleware checks for the Authorization header and validates the token using jsonwebtoken.
    - If valid, the user's id is extracted and appended to req.body for use in controllers.
    - If invalid or missing, appropriate error responses (401 or 403) are sent.
- **Session Management**
  - This implementation is stateless as tokens do not require server-side storage, making it scalable and efficient.

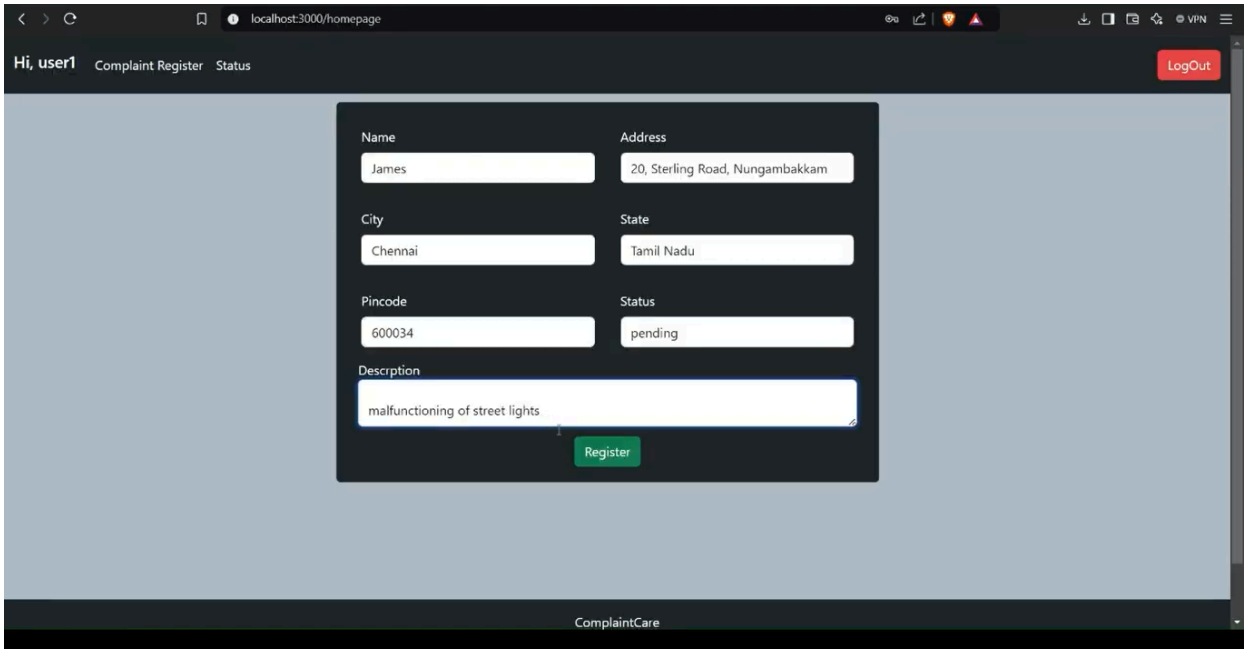
This setup ensures secure access to protected routes based on user identity.

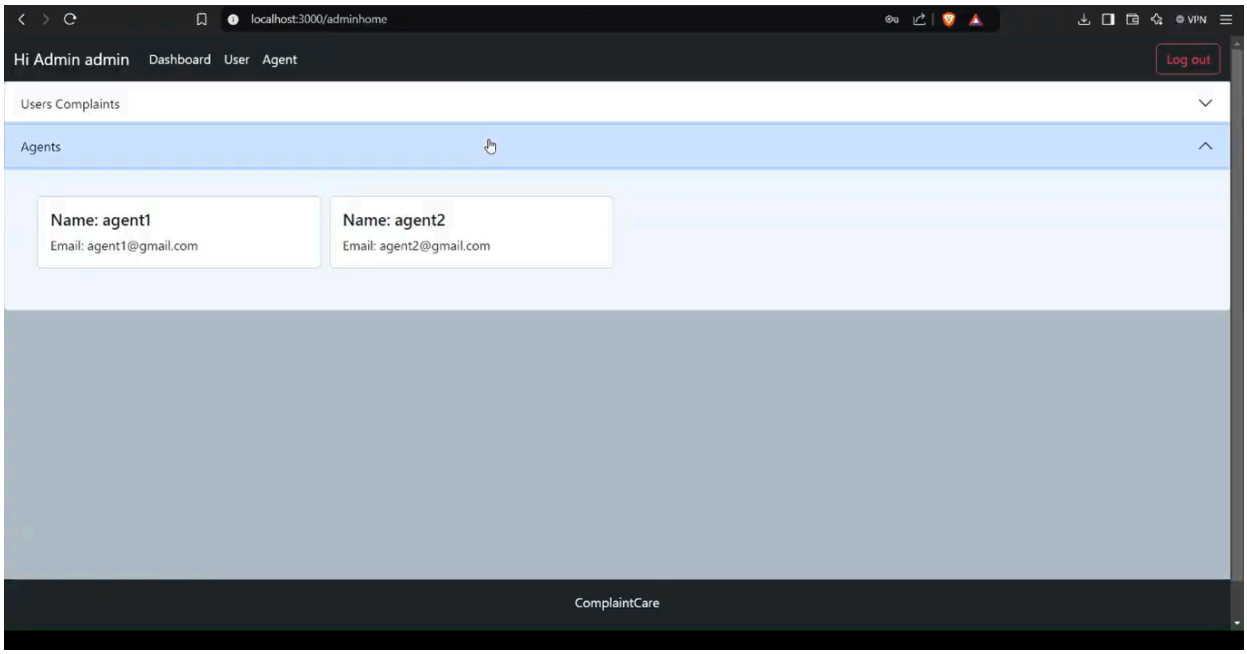
User Interface

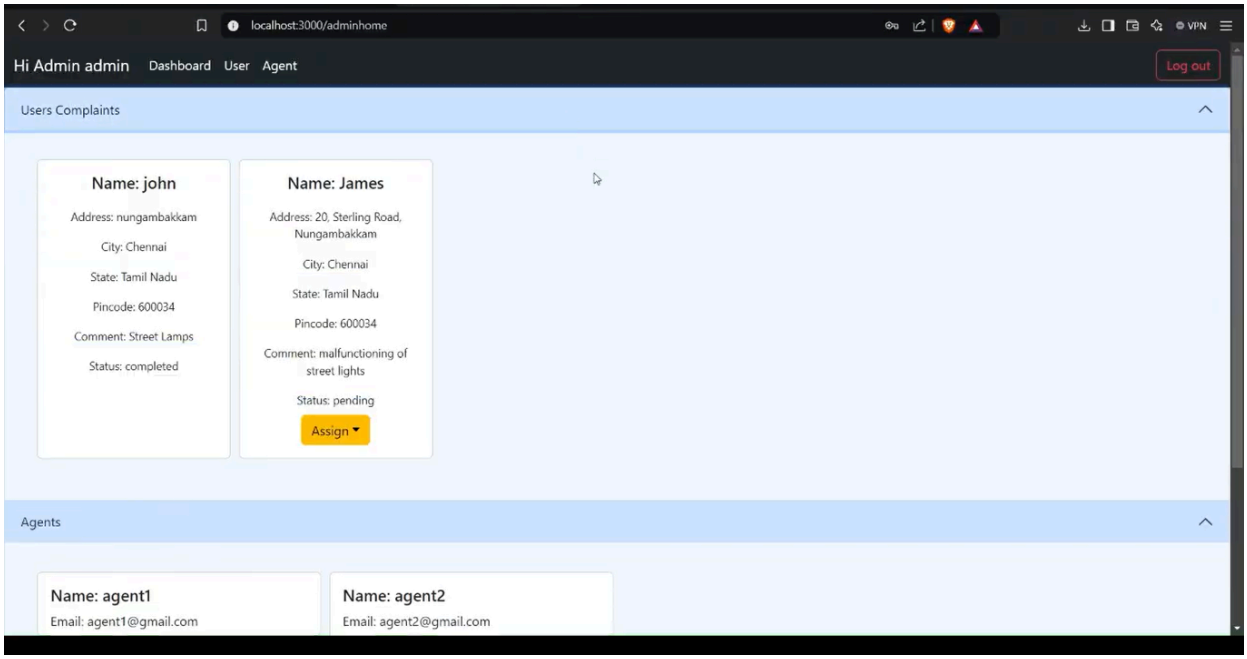


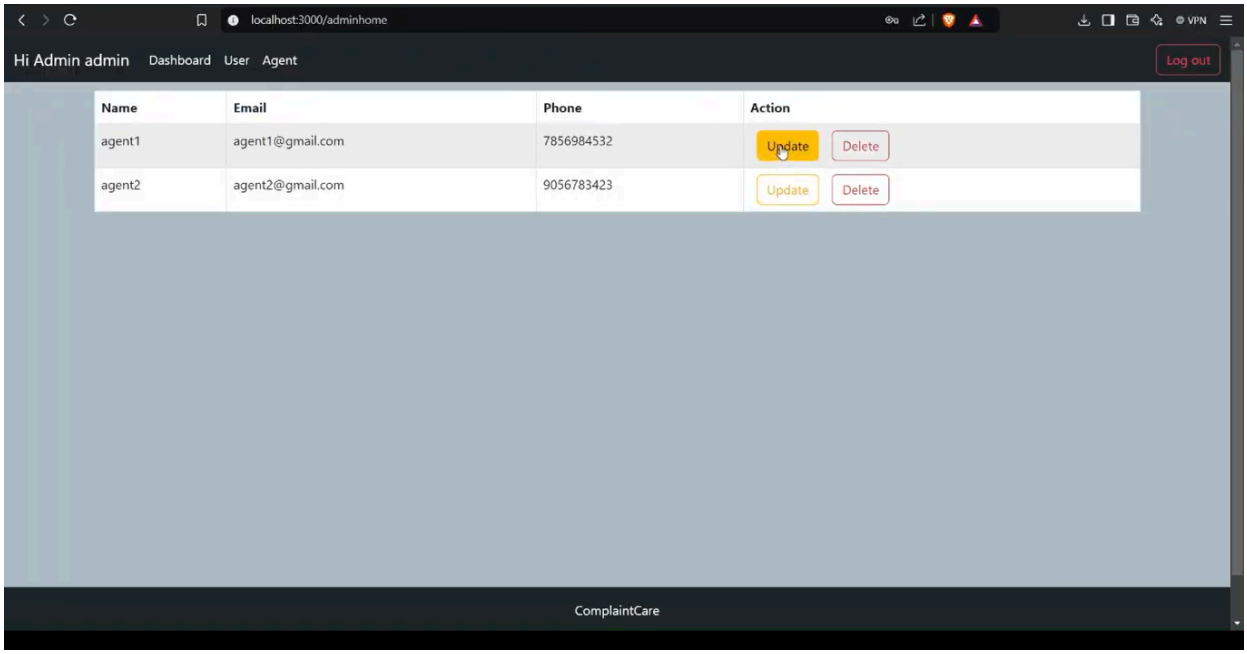

ComplaintCare
Home
SignUp
Login
Login For Registering the Complaint
Please enter your Credentials!
Email
Password
Login


localhost:3000/agenthome
Hi Agent agent1
View Complaints
Log out
Name: john
Address: nungambakkam
City: Chennai
State: Tamil Nadu
Pincode: 600034
Comment: Street Lamps
Status: completed
Status Change
Message
Message Box
agent1: Received
10:45 PM, 11/14/2024
Message
Send


localhost:3000/homepage
Hi, user1
Complaint Register
Status
LogOut
Name
James
Address
20, Sterling Road, Nungambakkam
City
Chennai
State
Tamil Nadu
Pincode
600034
Status
pending
Descrption
malfunctioning of street lights
Register
ComplaintCare


localhost:3000/adminhome
Hi Admin admin
Dashboard
User
Agent
Log out
Users Complaints
Agents
Name: agent1
Email: agent1@gmail.com
Name: agent2
Email: agent2@gmail.com
ComplaintCare





Testing

- **Frontend**:
  - **Tool**: React Testing Library to validate UI interactions and state changes.
- **Backend/API**:
  - **Tool**: Postman to test API endpoints like /login, /submitcomplaint.

- **Database**:
    - **Tool**: MongoDB Compass to verify data storage integrity.
- **Performance**:
    - **Tool**: Apache JMeter to simulate user load.

Demo

https://drive.google.com/file/d/1Z_HeQ0DpUuNTA68RYP4D_vb0fQQBDoAt/view?usp=drive_link