**Full Stack Development with MERN**

**Project Documentation**

**1. Introduction**

**Project Title:** Online Complaint Registration and Management system

**Team Members**:

Vinoth.P - Full Stack Developer

Abishek.R-Frontend

Mohanarangan.G-Backend

SCR Vigneshwar Sah-MongoDB

**2. Project Overview**

**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.

**Features**:

- Users can register and log in securely using their credentials

- Users and admins receive updates and alerts regarding the progress of complaints

- Users can track the status of their submitted complaints in real-time

- Administrators have a centralized dashboard to view, assign, and manage complaints.

**3. Architecture**

**Frontend**:

The frontend is responsible for capturing the user’s input, displaying relevant information, and interacting with the backend via APIs or other means to perform tasks like registering complaints, viewing complaints, and tracking the status.

**Backend**:

The backend architecture of an online complaint registration and management system refers to the server-side components that handle the processing of complaint data, store it securely, and manage user interactions

**Database**:

MongoDB is used as the primary database. The database schema includes collections for users and complaints. Key interactions include creating and fetching user data, user details, and complaint records.

**4. Setup Instructions**

**Prerequisites:**

- Node.js (v14 or higher)

- MongoDB

**Installation**:

1. Clone the repository:

```bash

git clone [repository\_url]

```

2. Navigate to the project directory and install dependencies:

```bash

cd backend

npm install

```

3. Create a `.env` file in the `backend` directory with the following environment variables:

```plaintext

MONGO\_DB=<your\_mongodb\_connection\_string>

PORT=3000

**5. Folder Structure**

**Client:**

To build the Client-side for an Online Complaint Registration and Management System, the focus will be on creating a simple user interface where users can submit complaints, view their complaints, and track the status of their complaints. The client-side will typically consist of HTML, CSS, and JavaScript (or a framework like React or Angular) to interact with the backend API.

**Server:**

The Server-side for an Online Complaint Registration and Management System, we need to build a backend that handles the logic for complaint submission, management, and status tracking. In this implementation, we've created a simple Complaint Registration and Management System backend using Node.js, Express, and MongoDB.

**6. Running the Application**

**Frontend:**

Run `npm start` in the `client` directory to start the React development server.

**Backend:**

Run `npm start` in the `backend` directory to start the Express server.

**7. API Documentation**

**Endpoints**:

- POST /complaints-Register a complaint

- GET /complaints – View complaint details

- PATCH /complaints/status – Update status of complaint

- POST /complaints/comments – Add comments to complaint

-GET /complaints- List all complaints

**8. Authentication**

**Current Status:**

No authentication is implemented, per project requirements. All endpoints are accessible without restrictions.

**9. User Interface**

**Overview**:

The system allows users to file, track, and manage complaints online. It is designed to streamline the complaint management process, improving efficiency and communication between users and administrators.

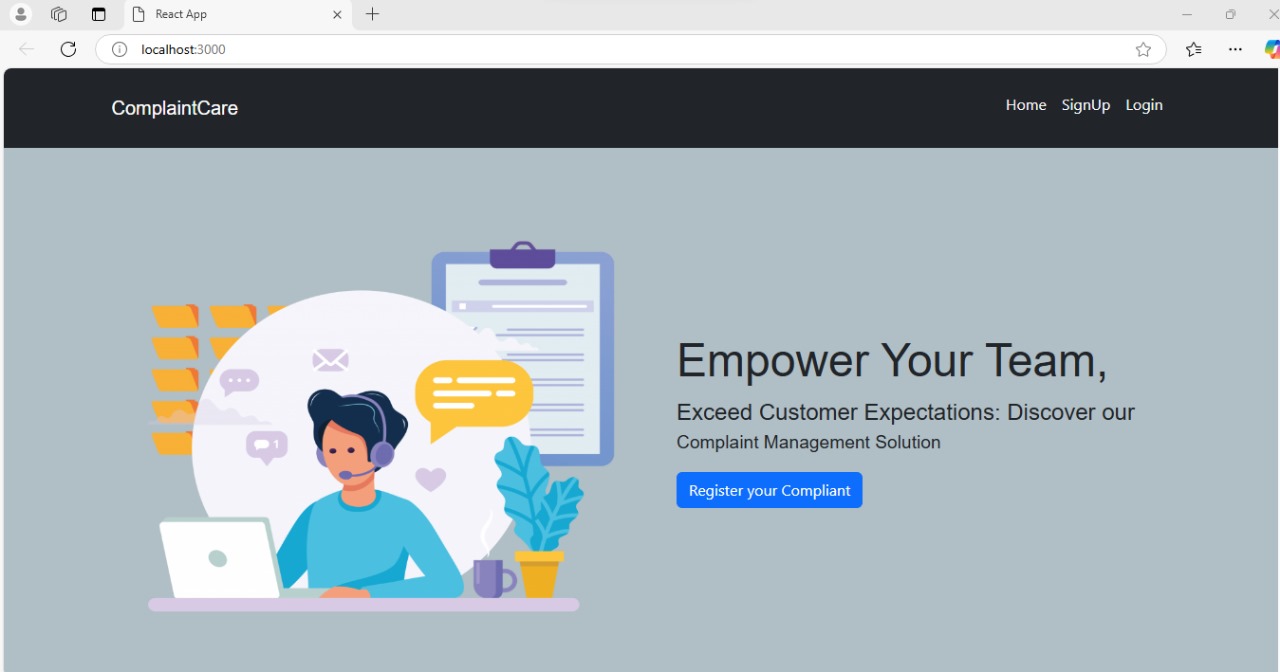
**10. Testing**

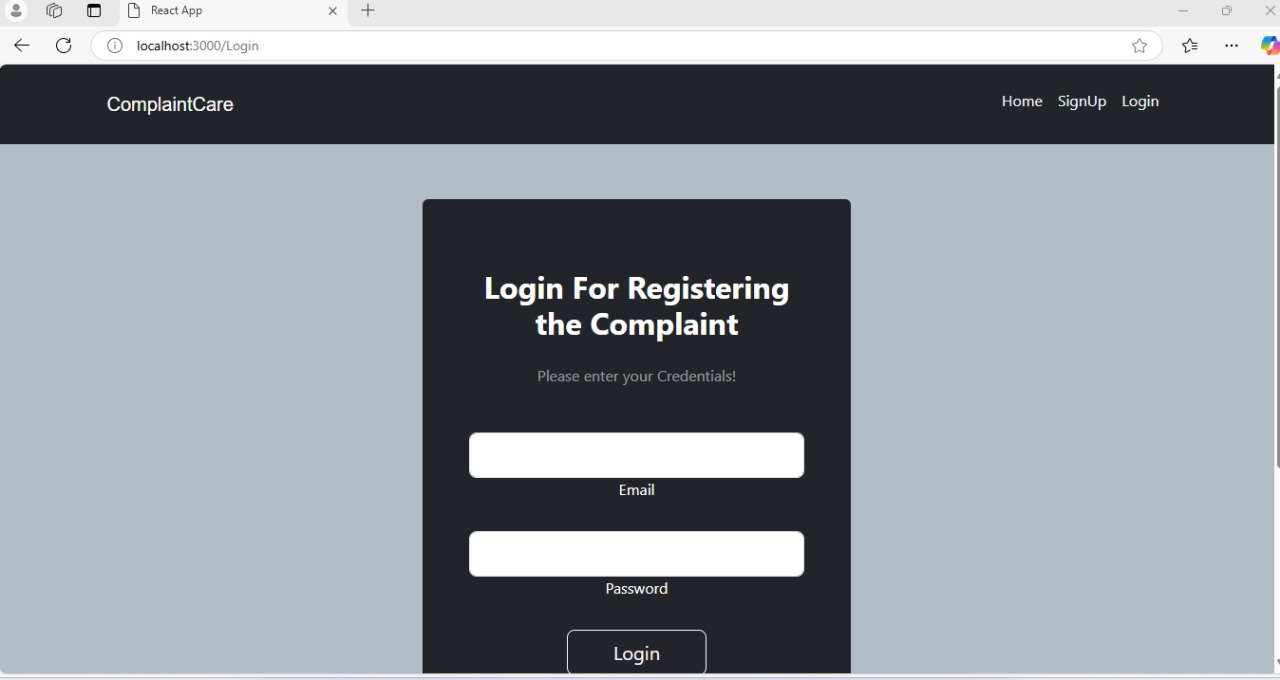
**Testing Strategy**:

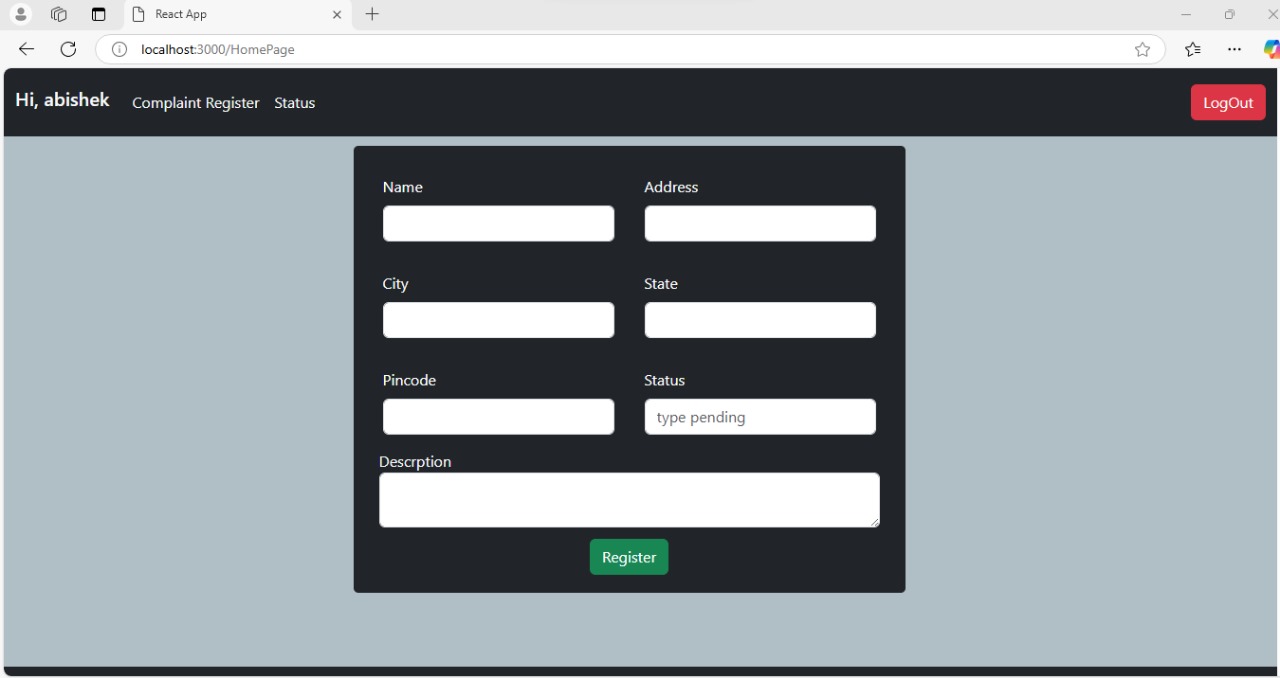
- Manual testing of all frontend and backend functionality.

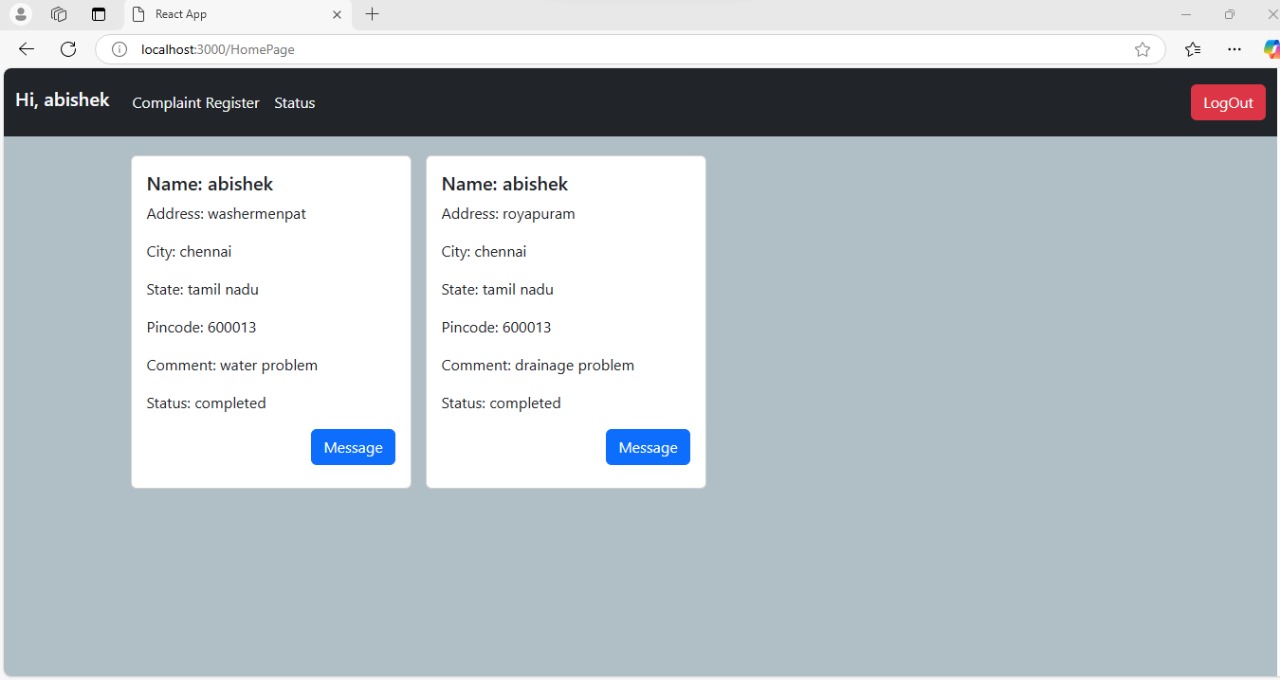
- Postman is used for API testing.

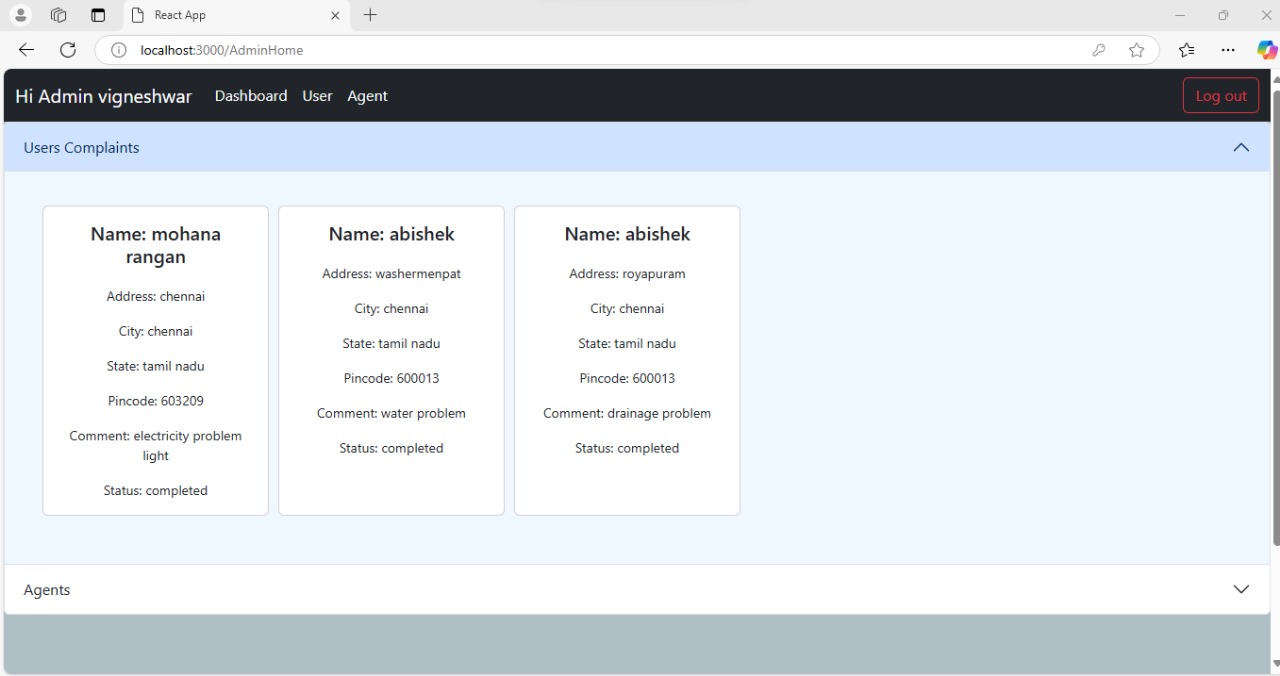
**11. Screenshots or Demo**

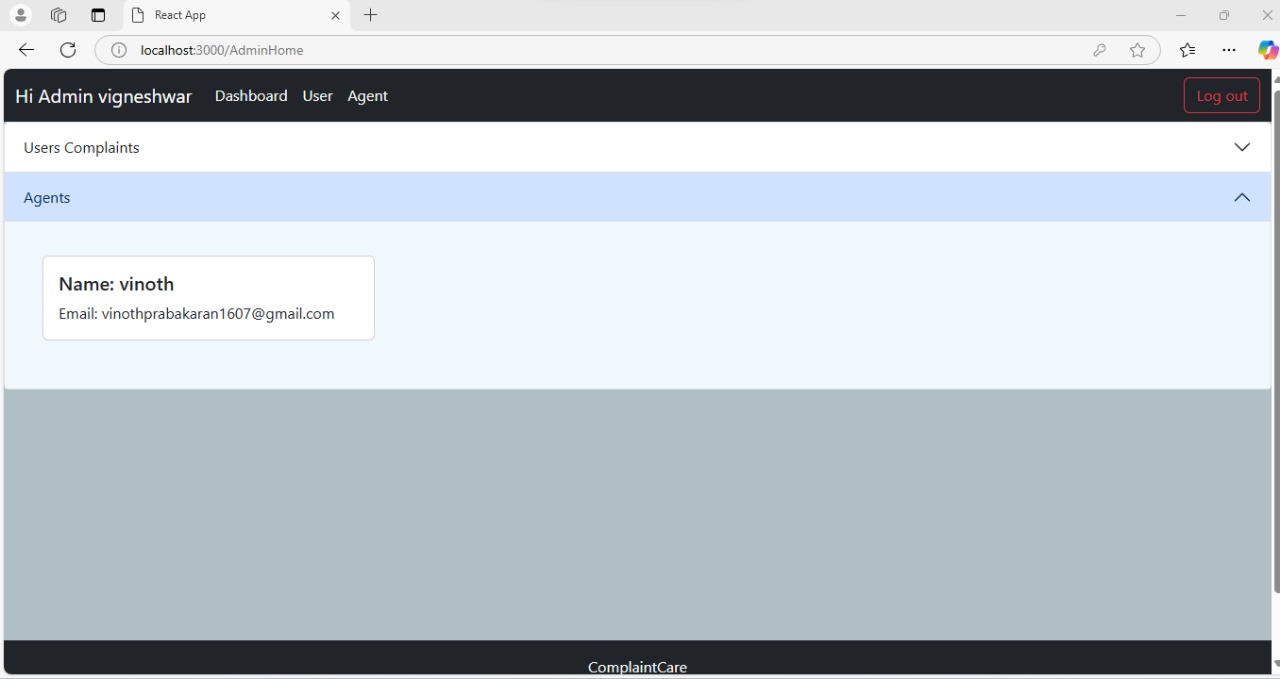
**Screenshots: **

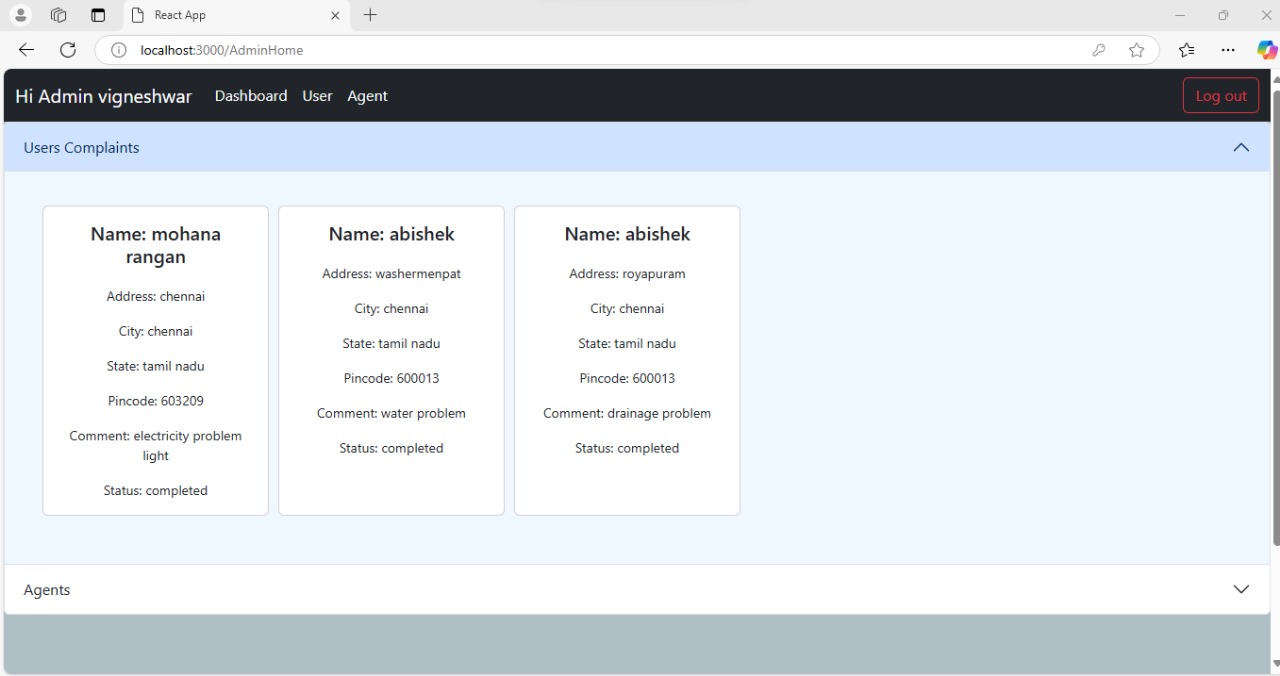
****



****

****

****

****

**12. Known Issues**

**Issues:**

- Page reload required after booking updates.

**13. Future Enhancements**

**Potential Features**:

** Enhanced Analytics**:  
Implement analytics to track user behavior, popular listings, and booking trends to help property owners improve their listings.

 **Multi-language Support**:  
Add multi-language functionality to accommodate users from different regions.

 **Dynamic Pricing Model**:  
Integrate a feature for dynamic pricing based on demand, availability, or seasonal trends.

 **Review and Rating System**:  
Allow users to review and rate properties, improving transparency and helping future users make informed decisions.

 **Property Comparison Feature**:  
Enable users to compare properties side-by-side based on features, prices, and availability.

 **Virtual Tours**:  
Offer virtual tours or 360° views for listed properties, providing an immersive experience before booking.