**Online Complaint Registration and Management System – Report**

**1.Introduction**

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

**Team Members**

* Boobeash M
* John Alfred
* Jean PJ
* Joel John Thomas

**2. Project Overview**

Purpose:

The **Online Complaint Registration Management System** is designed to provide a seamless, efficient, and transparent process for submitting, tracking, and resolving complaints in various sectors such as customer service, government, and healthcare. This system aims to simplify the complaint management workflow by allowing users to easily lodge complaints online, while providing administrators and service providers with tools to manage and resolve these complaints promptly.

In traditional complaint registration systems, users often face difficulties such as long wait times, lack of transparency, and no easy way to track the status of their complaints. The Online Complaint Registration Management System addresses these issues by providing an accessible platform for complaint submission, real-time tracking, and timely resolutions.

**Features**

* **Streamline** the complaint registration and resolution process.
* **Transparent system** where users can track the progress of their complaints.
* **Automate** the complaint management workflow, ensuring quicker responses.
* **Dashboard** for administrators to monitor complaints and manage resources effectively.

**3. System Design and Architecture**

**3.1. System Overview**

The Online Complaint Registration Management System consists of the following key components:

* **User Interface (UI)**: Enables users to submit complaints, track their progress, and receive notifications.
* **Admin Panel**: Allows administrators to manage complaints, assign them to relevant departments, and monitor progress.
* **Backend Services**: Handles complaint processing, user management, and communication between frontend and database.
* **Database**: Stores complaint records, user information, and complaint status updates.

**3.2. Technology Stack**

* **Frontend**: HTML5, CSS3, JavaScript (React.js or Angular for interactive UI).
* **Backend**: Node.js/Java (Spring Boot) for processing and managing data.
* **Database**: Mongodb atlas for complaint and user data storage..
* **Security**: JWT-based authentication for secure user logins

**4. Setup Instructions**

1. **Clone the Repository -** git clone

https://github.com/boobeash22/smartinternz-

1. **Install Dependencies**
   * Frontend:

cd frontend && npm install

* + Backend:

cd backend && npm install

1. **Set Up Database**
   * Create a database (e.g., complaint\_system\_db).
   * Update the database connection in the backend (config.js or application.properties).
2. **Configure** **Environment Variables**
   * Set up .env file with necessary keys (e.g., DATABASE\_URL, JWT\_SECRET, EMAIL\_SERVICE\_API\_KEY).
3. **Run the Application**
   * Backend:

node index.js

* + Frontend:

npm start

1. **Access the App**
   * Frontend: http://localhost:3000
   * Backend: <http://localhost:8000>

**5.Folder Structure**

**Client**

frontend/

├── src/

│ ├── components/ # Reusable components (e.g., Navbar, ProductCard, CartItem)

│ ├── pages/ # Page components (e.g., Home, Cart, Checkout, AdminPanel)

│ ├── services/ # API interactions (fetching and posting data)

│ ├── redux/ # State management (cart items, user auth, order status)

│ └── App.js # Main App component

├── public/ # Static files (index.html, images, etc.)

├── .env # Environment variables (e.g., API URL)

├── package.json # Frontend dependencies and scripts

└── README.md # Frontend documentation

**Server**

backend/

├── routes/ # Defines RESTful API routes (users, products, orders)

├── controllers/ # Logic for handling API requests and responses

├── models/ # Database schemas (Users, Products, Orders)

├── middleware/ # Handles authentication (JWT) and error management

├── config/ # Database configuration and environment variables

├── utils/ # Helper functions (e.g., validation, error handling)

├── .env # Environment variables (e.g., DB credentials, JWT secret)

├── server.js # Main server file to start the app

├── package.json # Backend dependencies and scripts

└── README.md # Backend documentation

**6. Running the Application**

**Frontend:**

Navigate to the client directory:

cd frontend

Start the React application:

npm start

**Backend:**

Navigate to the server directory:

cd backend

Start the Node.js server:

**7.API Documentation**

**Request Body:**

json

{

"user\_id": "string",

"title": "string",

"description": "string",

"category": "string",

"status": "string",

}

**Response:**

json

{

"complaint\_id": "string",

"message": "Complaint registered successfully"

}

**8. Authentication**

**JWT-based Authentication:**

● **Login**: Issues a JWT token upon successful authentication.

● **Token Validation**: Protects private routes like /complaints and /admin.

● **Logout**: Invalidates the token on the client side.

**9.User Interface:**

* Dashboard
* Registration Form
* Complaint List
* Complaint Details page
* Admin panel

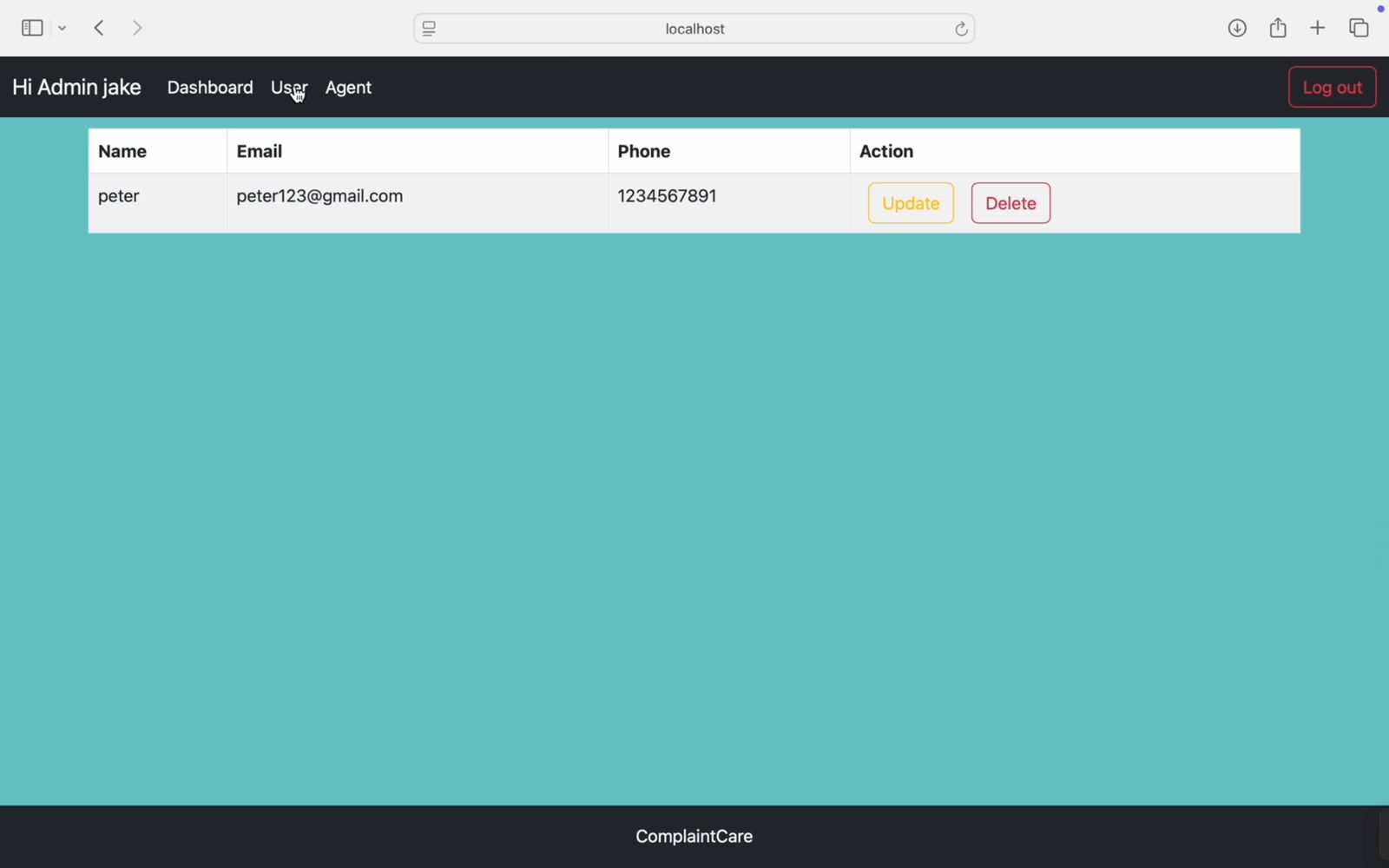
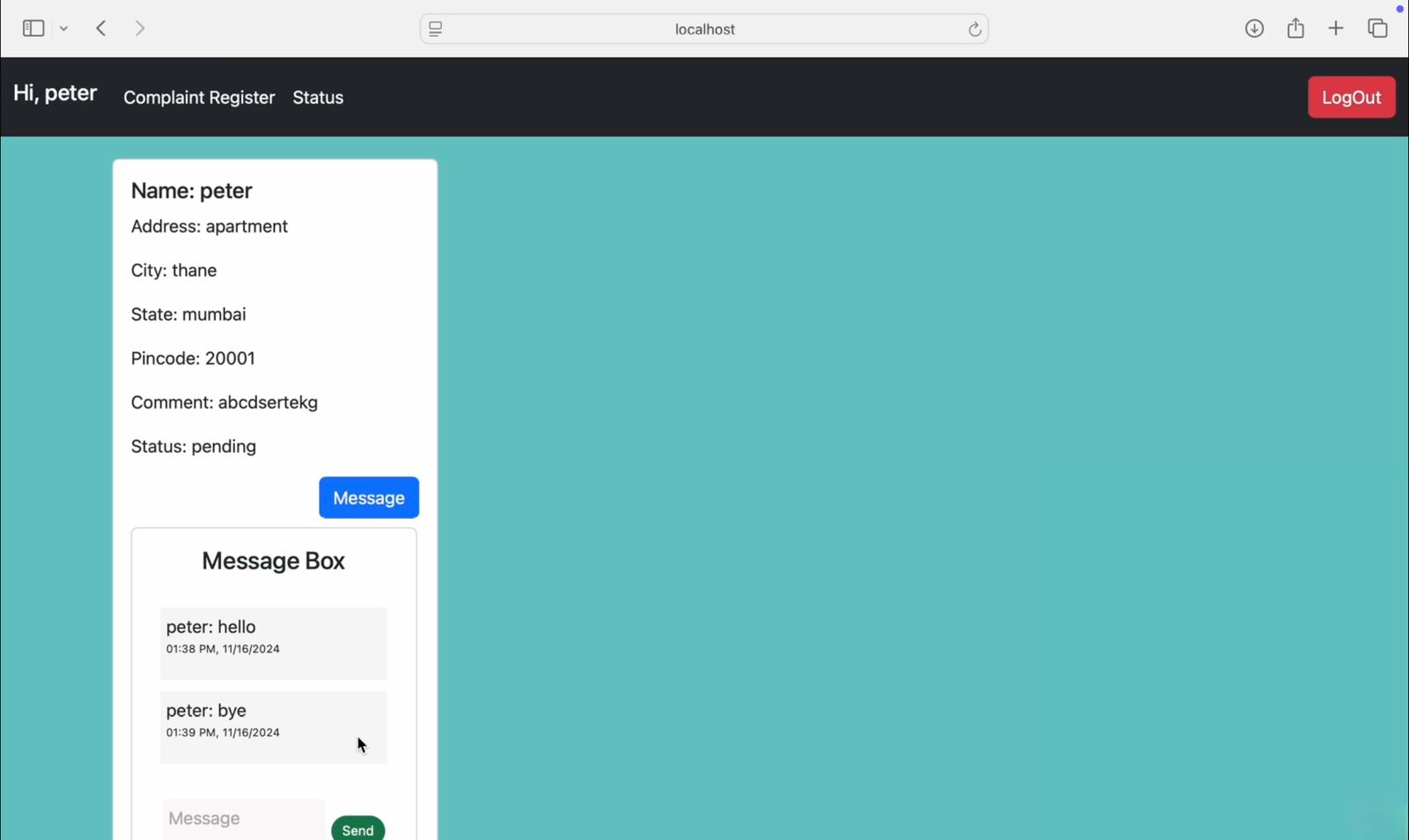
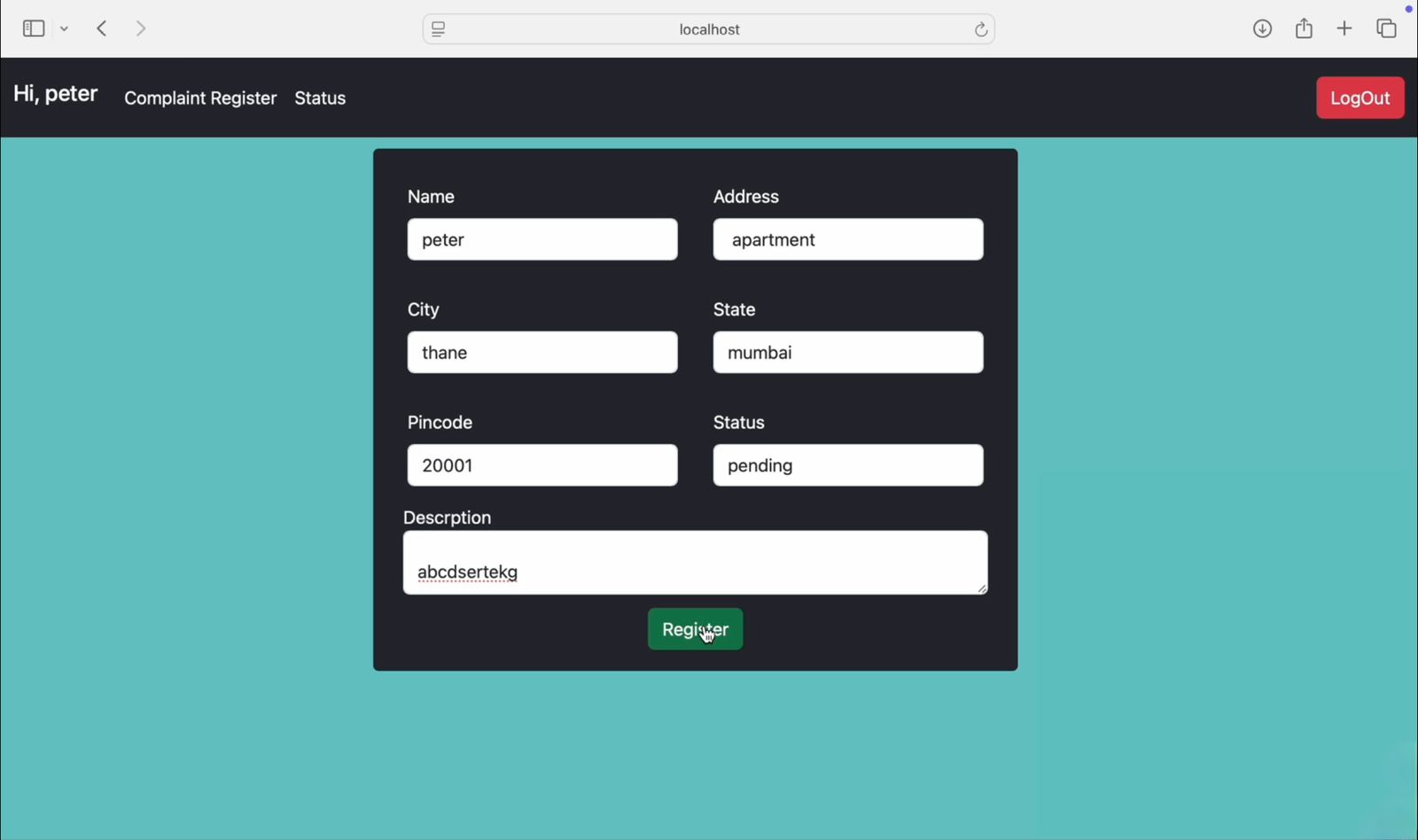
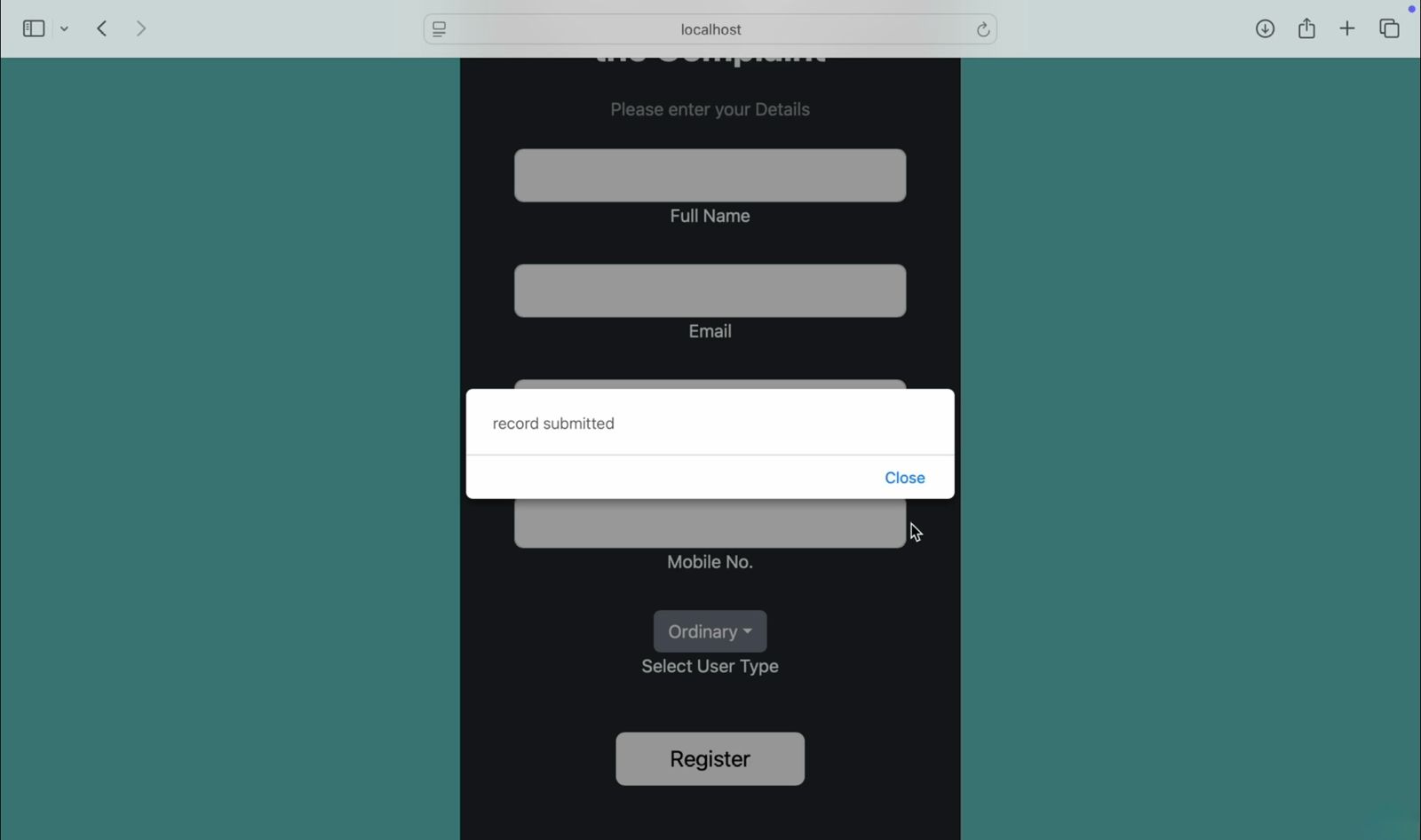
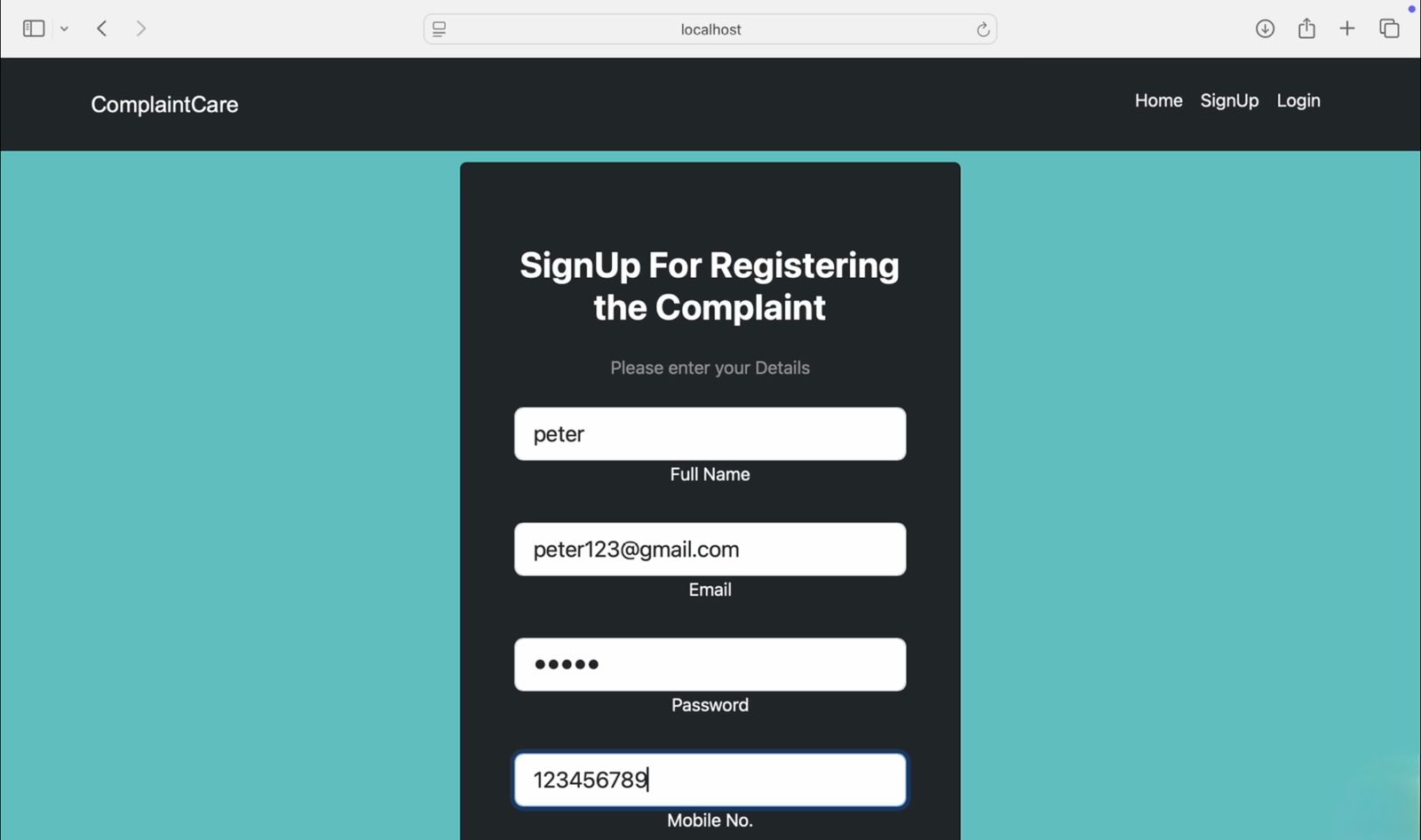
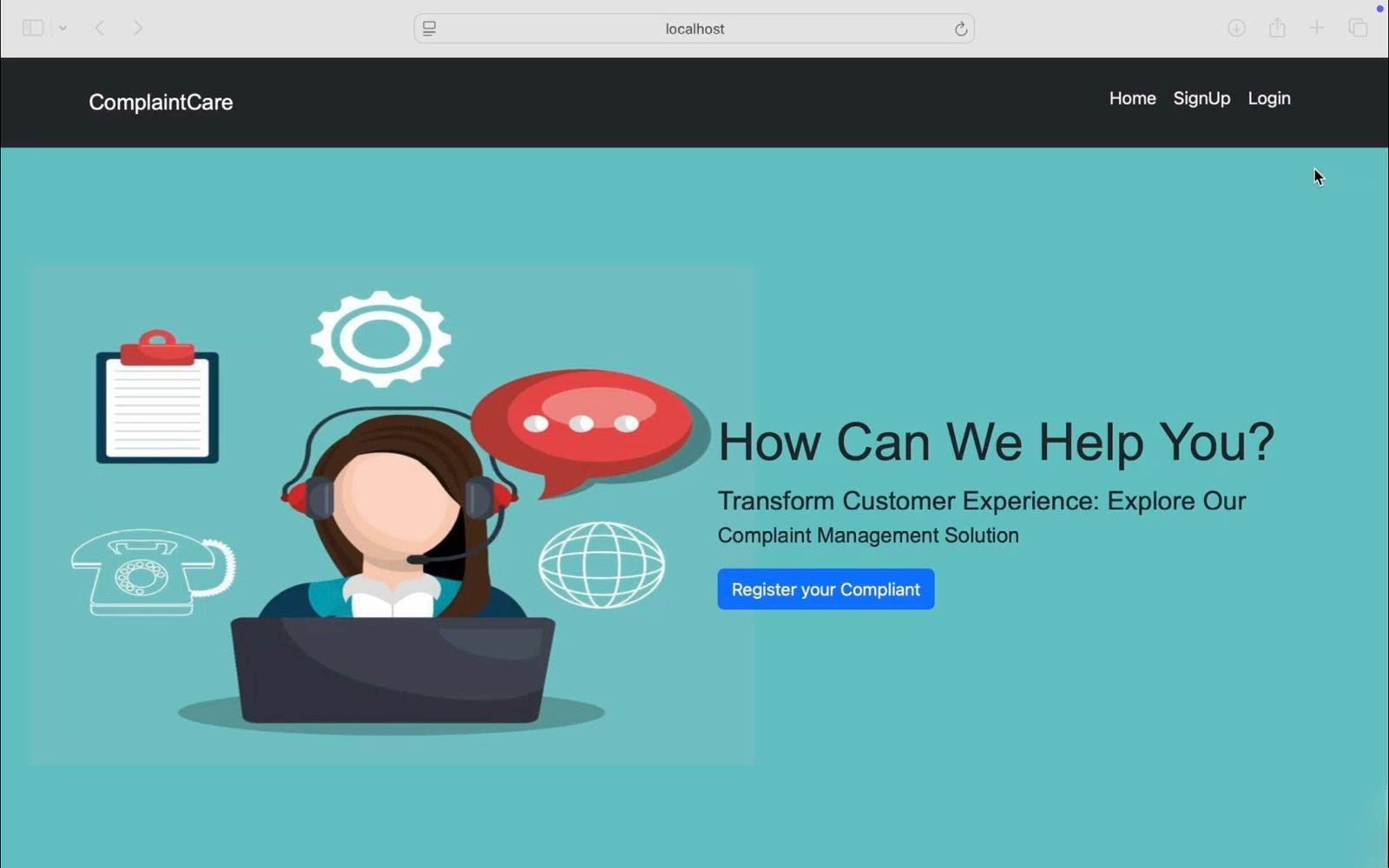
**10. Testing**

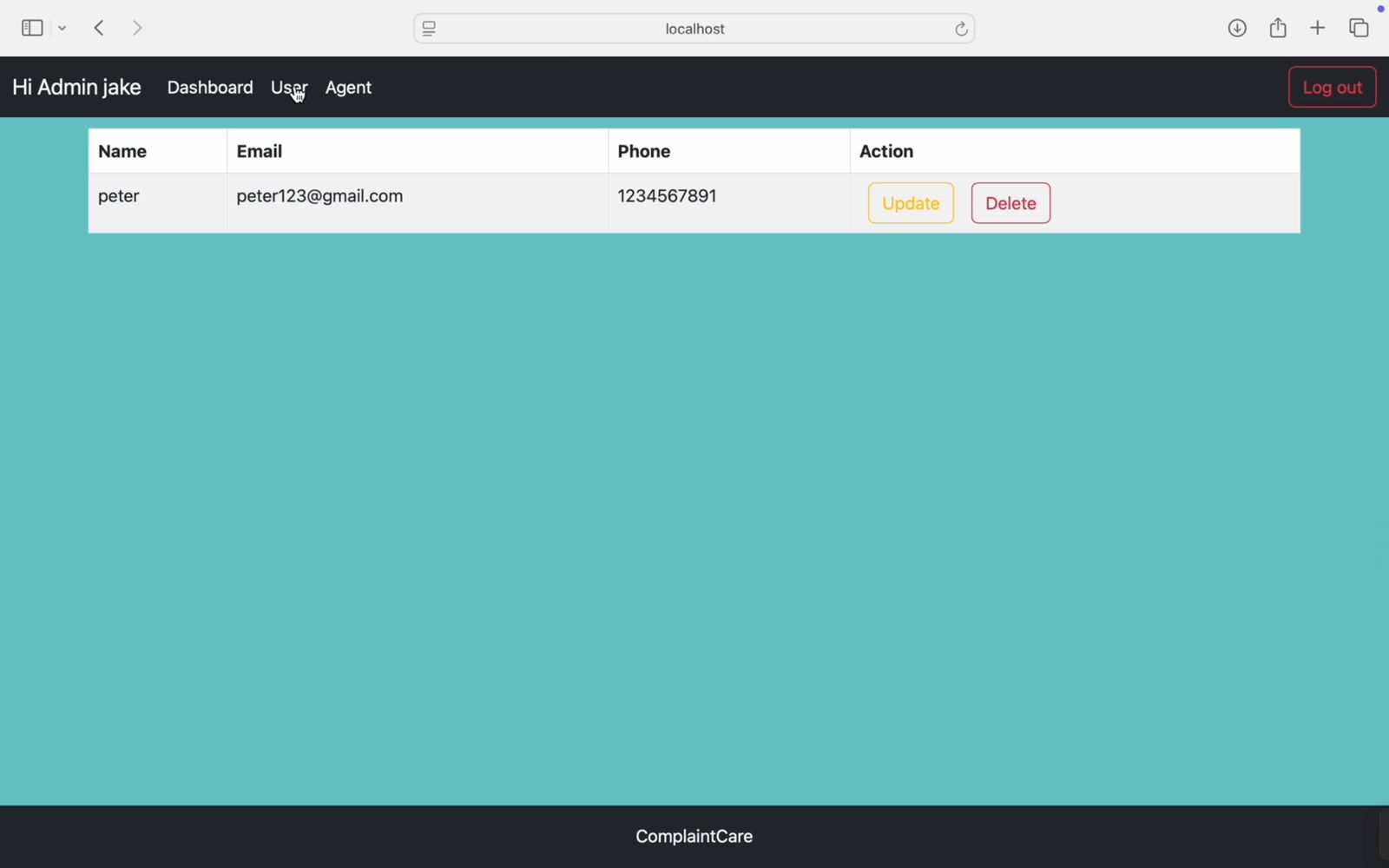
● Unit Testing: Conducted using Jest for components and backend logic.

● API Testing: Performed using Postman to validate endpoints.

● Usability Testing: Evaluates the ease of use and user experience.

● Load Testing: Tests the system's behavior under high traffic conditions

**11.Screenshots or Demo**

****

**12. Known Issues**

● User Authentication & Authorization Issues

● User Experience (UX) Issues

● Spam and Fake Complaints

**13. Future Enhancements**

● **Chatbot Integration**: Provides instant support and helps users register complaints.

● **Priority-Based System**: Classifies complaints by urgency for efficient resolution.

● **Enhanced Security (MFA)**: Adds multi-factor authentication for improved account protection.