

**NM1042-MERN Stack Powered by Mongo DB:**

ONLINE COMPLAINT REGISTRATION AND MANAGEMENT SYSTEM

**A PROJECT REPORT**

**SUBMITTED BY**

SIVARANJINI. S- 310821104091

SURESHKUMAR.B- 310821104096

NITYA POOJAA. T- 310821104065

VISHAL ESHWAR.D-310821104115

***in partial fulfillment for the award of the degree***

***of***

BACHELOR OF ENGINEERING

**IN**

COMPUTER SCIENCE AND ENGINEERING

**JEPPIAAR ENGINEERING COLLEGE**

**ANNA UNIVERSITY : CHENNAI 600 025**

**2021-2025**



**ANNA UNIVERSITY : CHENNAI 600 025**

**BONAFIDE CERTIFICATE**

Certified that this project report **“ONLINE COMPLAINT REGISTRATION AND MANAGEMENT SYSTEM** **”** is the bonafide work of **“---------------------------------------------------”** who carried out the project work under my supervision.

## **SIGNATURE SIGNATURE**

**HEAD OF THE DEPARTMENT SUPERVISOR**

TABLE OF CONTENTS

1. **Introduction**
2. **Project Title**
3. **Team Members**
4. **Project Overview**
5. **Purpose**
6. **Features**
7. **Architecture**
   * Frontend
   * Backend
   * Database
8. **Setup Instructions**
   * Prerequisites
   * Installation
   * Folder Structure
     + Client
     + Server
   * Running the Application
     + Starting Frontend
     + Starting Backend
9. **API Documentation**
   * Endpoints
   * Request Methods, Parameters, and Example Responses
10. **Authentication**
    * Authentication and Authorization Details
    * Tokens, Sessions, or Other Methods
11. **User Interface**
    * Screenshots of Features
12. **Testing**
    * Testing Strategy and Tools
    * Screenshots or Demo
13. **Application Screenshots and Demo Link**
14. **Known Issues**
    * Bugs or Issues
15. **Future Enhancements**
    * Potential Features or Improvements

### **Introduction**

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

The **Online Complaint Registration and Management System** is designed to streamline the process of submitting and managing complaints. It aims to provide an efficient way for customers to submit complaints, track their status, and interact with agents, improving customer service and response times.

### 

### **Team Members and Their Roles**

* **SIVARANJINI.S** –**Backend Developer** Developed the backend using Express.js and MongoDB, focusing on APIs for complaint submission, management, and routing.
* **SURESHKUMAR.B** – **Frontend Developer** Designed and implemented the user interface using React.js and Material UI, ensuring a responsive and user-friendly system.
* **NITYA POOJA.T** – **Database Administrator** Managed the MongoDB database, ensuring efficient data storage and handling CRUD operations for users, complaints, and agents.
* **VISHAL ESHWAR.D** – **Project Manager & QA Lead** Coordinated the project’s progress and led QA efforts, ensuring the system met quality standards through testing and bug fixes.

### 

### **Project Overview**

The Online Complaint Registration and Management System is designed to enable customers to submit complaints and track their resolution progress. It serves as a central hub for managing complaints, automating task assignments, and maintaining a transparent and efficient complaint resolution process. The system is built with a client-server architecture, using modern technologies like React, Node.js, and MongoDB, ensuring a seamless user experience for both customers and administrators.

### 

### 

### 

### **Purpose**

The purpose of this project is to provide a user-friendly platform for individuals or organizations to report issues and track their resolution in real-time. The system aims to streamline the complaint handling process, ensuring faster resolution, reducing administrative overhead, and improving overall customer satisfaction.

### **Features**

* **User Registration:** Allows users to create an account and log in to submit complaints.
* **Complaint Submission:** Users can submit complaints with necessary details, including attachments like images and documents.
* **Tracking & Notifications:** Users can track the status of their complaints and receive updates via email or SMS.
* **Agent Interaction:** A direct communication channel between users and assigned agents through a messaging system.
* **Complaint Assignment:** Complaints are automatically assigned to appropriate agents based on predefined criteria.
* **Admin Management:** Admins can monitor complaints, assign tasks, and manage users and agents.

### **Architecture**

The system follows a **Client-Server Architecture**:

* **Frontend:** Developed using React.js for an interactive user interface.
* **Backend:** Built with Node.js and Express.js to handle API requests and manage data.
* **Database:** MongoDB stores user and complaint data in a NoSQL format for scalability and flexibility.

### **Frontend**

The frontend of the system is built using **React.js**, a popular JavaScript library for creating user interfaces. React enables the development of reusable UI components, ensuring the application is responsive and efficient. The frontend also integrates libraries like **Material UI** and **Bootstrap** to ensure a polished and responsive design for various user roles (customers, agents, admins).

### **Backend**

The backend uses **Node.js** and **Express.js** to handle HTTP requests, perform server-side logic, and manage API routes. The backend is responsible for handling user authentication, complaint management, and communication between the frontend and database.

### 

### 

### **Database**

The system utilizes **MongoDB** as the database. MongoDB is a NoSQL database that stores data in a flexible, JSON-like format. This is ideal for the system as it allows efficient handling of diverse complaint data, including attachments and messages.

### **Setup Instructions**

To set up and run the system locally, follow the instructions below:

### **Prerequisites**

* **Node.js**: JavaScript runtime to run the backend server.
* **MongoDB**: Database for storing application data.
* **React.js**: Frontend library for building the user interface.
* **Git**: For version control and repository management.

Ensure that **Node.js**, **npm**, and **MongoDB** are installed on your machine before proceeding.

### **Installation**

Clone the repository to your local machine:  
  
 git clone https://github.com/awdhesh-student/complaint-registery.git

1. Navigate to the project directory:  
     
    cd complaint-registry
2. Install dependencies for both frontend and backend:  
     
    cd frontend
3. npm install

### 

### **Folder Structure**

The project is divided into two main directories: **Frontend** and **Backend**.

#### **Client (Frontend) Structure**

* src/ — Contains all the React components and related files.
* public/ — Contains static files like images and index.html.
* package.json — Lists dependencies for the frontend.

#### **Server (Backend) Structure**

* controllers/ — Contains logic for handling HTTP requests.
* models/ — Contains Mongoose models for database schemas.
* routes/ — Defines the API routes.
* config/ — Contains configuration files, including database setup.
* server.js — Main entry point for the backend server.

### **Running the Application**

#### **Starting Frontend**

Navigate to the **frontend** folder:  
  
 cd frontend

1. Start the development server

:

1. npm start
2. The frontend will be available at [**http://localhost:3000**](http://localhost:3000/).

#### **Starting Backend**

Navigate to the **backend** folder:  
  
 cd backend

1. Start the backend server:
2. npm start
3. The backend will be available at[**http://localhost:5000**](http://localhost:5000).

I apologize for the confusion earlier. Here's a revised version of the **API Documentation** section based on the specific details you have shared:

# 

# API Documentation

The **Online Complaint Registration and Management System** exposes a set of RESTful APIs that allow users to interact with the system. These APIs manage complaints, user roles, and authentication processes. Below are the details of the API endpoints:

### **Endpoints**

#### **1. User Registration Endpoint**

* **URL:** /api/register
* **Method:** POST

**Description:** Registers a new user (Customer, Agent, or Admin).

**Request Body:** {

"username": "SIVARANJINI",

"password": "Password123",

"role": "Customer"

}

**Response:** {

"status": "success",

"message": "User registered successfully"

}

#### **2. User Login Endpoint**

* **URL:** /api/login
* **Method:** POST
* **Description:** Authenticates a user and returns a token for further requests.

**Request Body:** {

"username": "SIVARANJANI",

"password": "Password123"

}

**Response:** {

"status": "success",

"token": "abcd1234efgh5678ijkl9012mnop3456"

}

#### **3. Complaint Submission Endpoint**

* **URL:** /api/complaint
* **Method:** POST
* **Description:** Allows a customer to submit a new complaint.

**Request Body:** {

"customerId": "SIVARANJINI",

"title": "Issue with product",

"description": "The product delivered is damaged.",

"attachments": ["image1.jpg"]

}

**Response:** {

"status": "success",

"complaintId": "complaint12345"

}

#### **4. Complaint Status Update Endpoint**

* **URL:** /api/complaint/{complaintId}/status
* **Method:** PUT
* **Description:** Updates the status of a submitted complaint.

**Request Body:** {

"status": "resolved"

}

**Response:** {

"status": "success",

"message": "Complaint status updated successfully"

}

#### **5. Get All Complaints Endpoint**

* **URL:** /api/complaints
* **Method:** GET
* **Description:** Retrieves a list of all complaints. Accessible by customers, agents, and admins, with different access levels.

**Response:** {

"status": "success",

"complaints": [

{

"complaintId": "complaint12345",

"title": "Product Issue",

"status": "pending"

}

]

}

### **Request Methods, Parameters, and Example Responses**

* **Request Methods:** The API supports HTTP methods like GET, POST, PUT, and DELETE.
* **Parameters:**
  + **URL Parameters:** e.g., complaintId to refer to a specific complaint.
  + **Query Parameters:** e.g., status=pending to filter complaints based on status.
  + **Request Body:** Typically in JSON format, used for creating and updating resources (e.g., complaints).

### 

### 

### 

### **Authentication**

The **Online Complaint Registration and Management System** uses **token-based authentication** to secure its endpoints.

#### **Authentication and Authorization Details**

* **Authentication:** Users log in with their username and password to obtain a token. This token must be included in the Authorization header for subsequent API calls.
* **Authorization:**
  + **Customer Role:** Can view and submit complaints.
  + **Agent Role:** Can view, resolve, and update complaint statuses.
  + **Admin Role:** Can manage users, view all complaints, and perform administrative actions.

### 

### 

### 

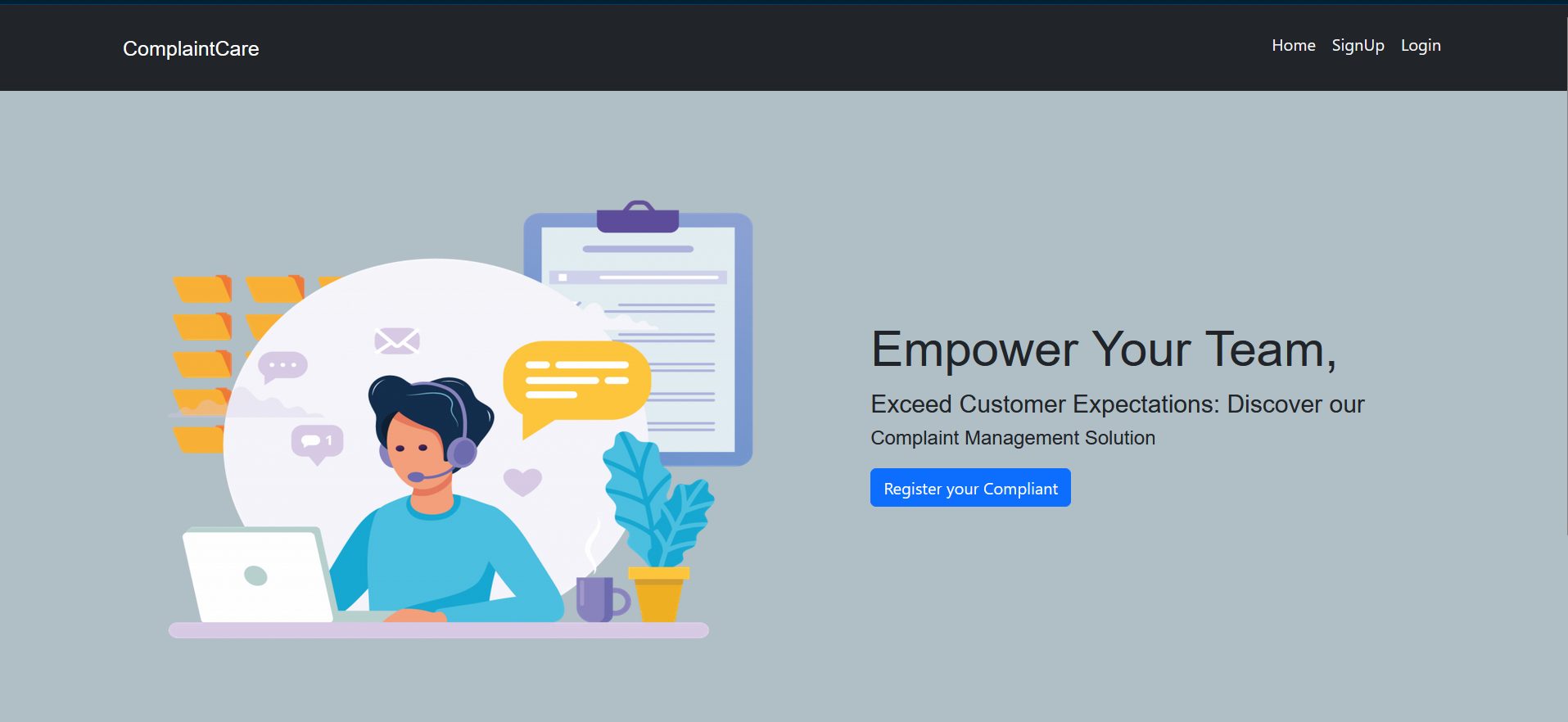
### **Tokens, Sessions, or Other Methods**

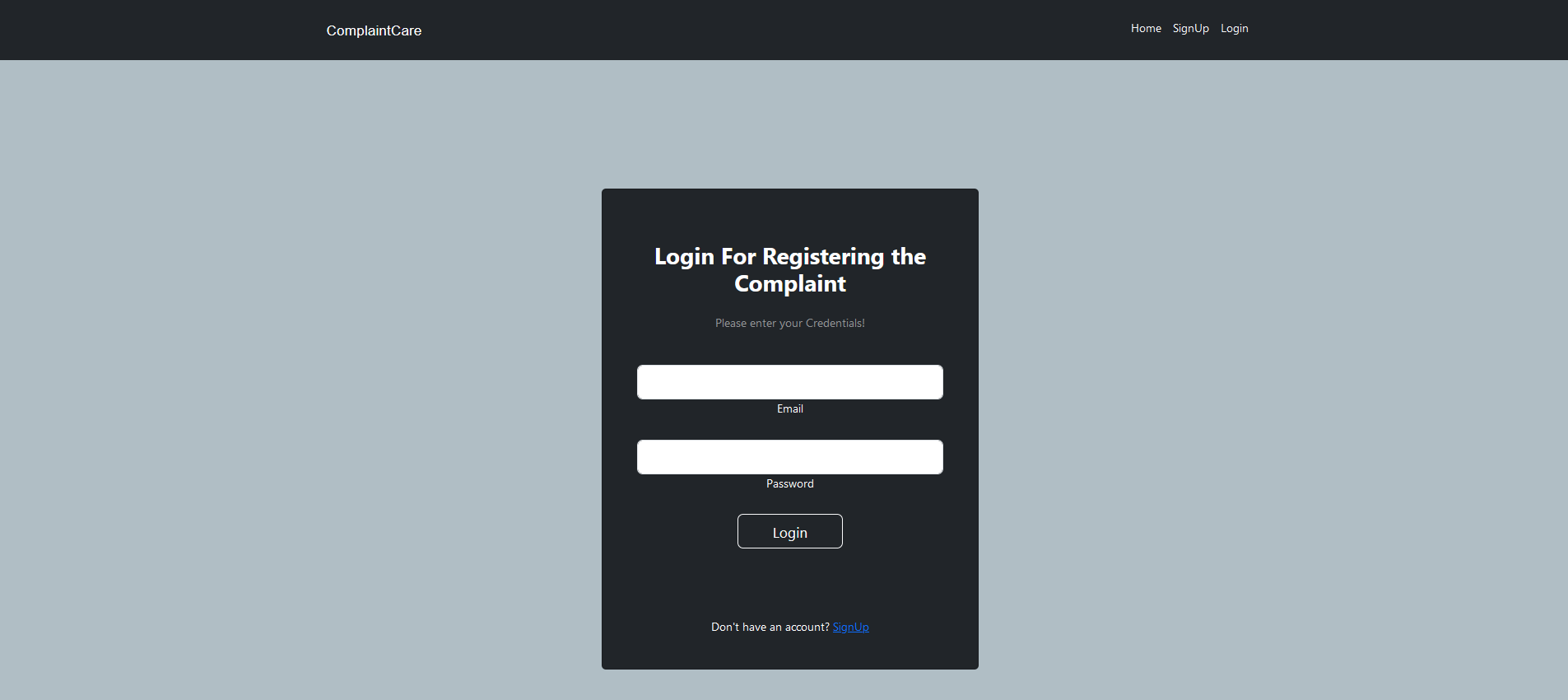
* **JWT Tokens:** Upon successful login, the system returns a JSON Web Token (JWT). This token must be included in the Authorization header of requests to access protected endpoints.
* **Session Management:** The system uses secure HTTP cookies to maintain user sessions on the client side.

### **User Interface**

The system provides a user-friendly interface for different user roles, offering distinct views and functionalities.

**Screenshots of Features**





### 

### 

### 

### 

### **Customer Dashboard**

The Customer Dashboard is the primary interface for end-users to manage their complaints. It provides an easy-to-navigate experience to submit new complaints, view existing ones, and track the progress. The dashboard includes:

* Complaint Submission Form: A form for users to submit their complaints, attach relevant files, and provide detailed information.
* Complaint Status Overview: A section displaying the current status of all complaints, along with timestamps of any updates.
* Search and Filter Options: Allows users to search and filter complaints based on various criteria (e.g., status, date, type).

**Agent Dashboard:** Allows agents to view and resolve complaints assigned to them.

The Agent Dashboard is designed for agents who handle and resolve complaints. It includes features to manage assigned complaints and communicate with customers. Key elements include:

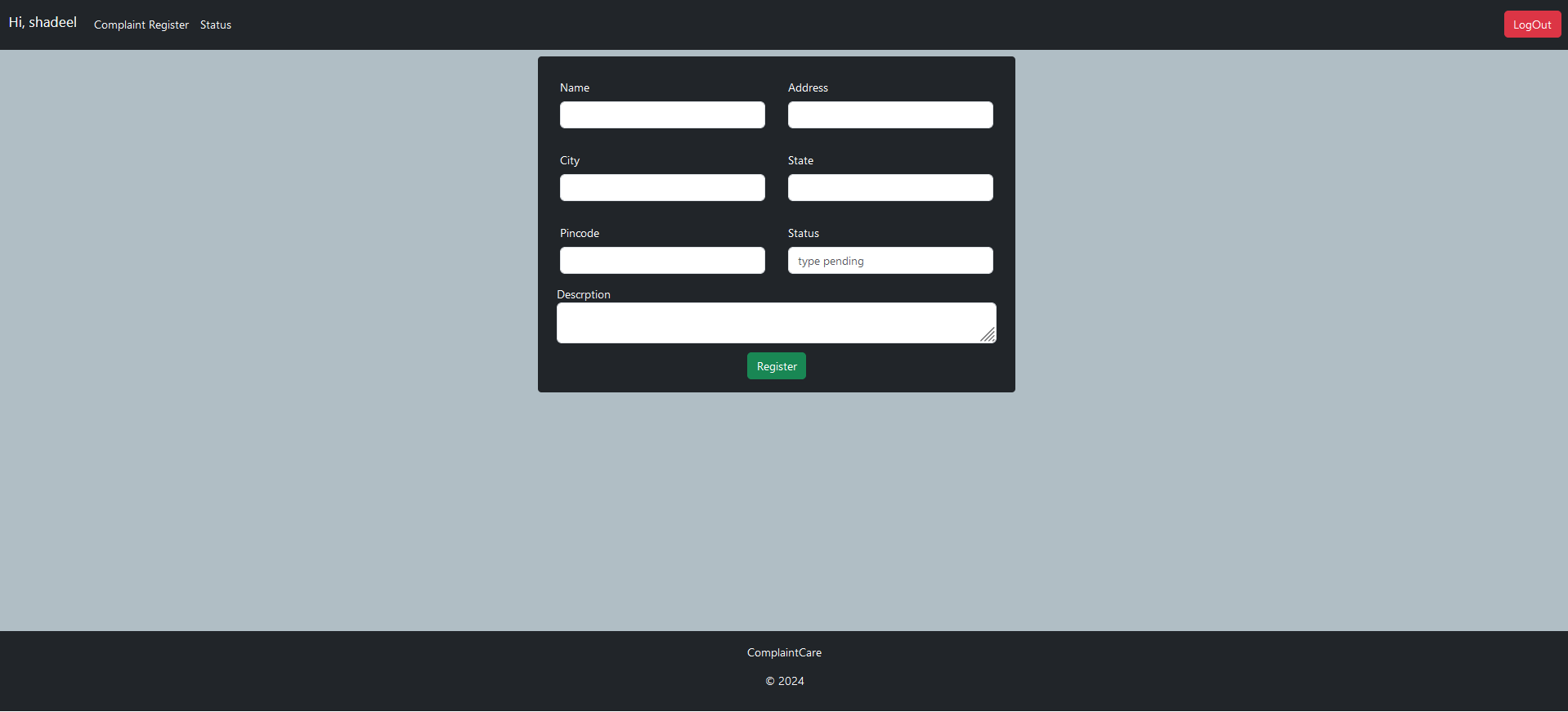
* Complaint Assignment List: Displays a list of complaints assigned to the agent with filters for status, priority, and date.
* Messaging System: A chat interface for agents to communicate directly with customers and resolve issues.
* Complaint Resolution Status: A section to update and track the status of complaints being handled by the agent.

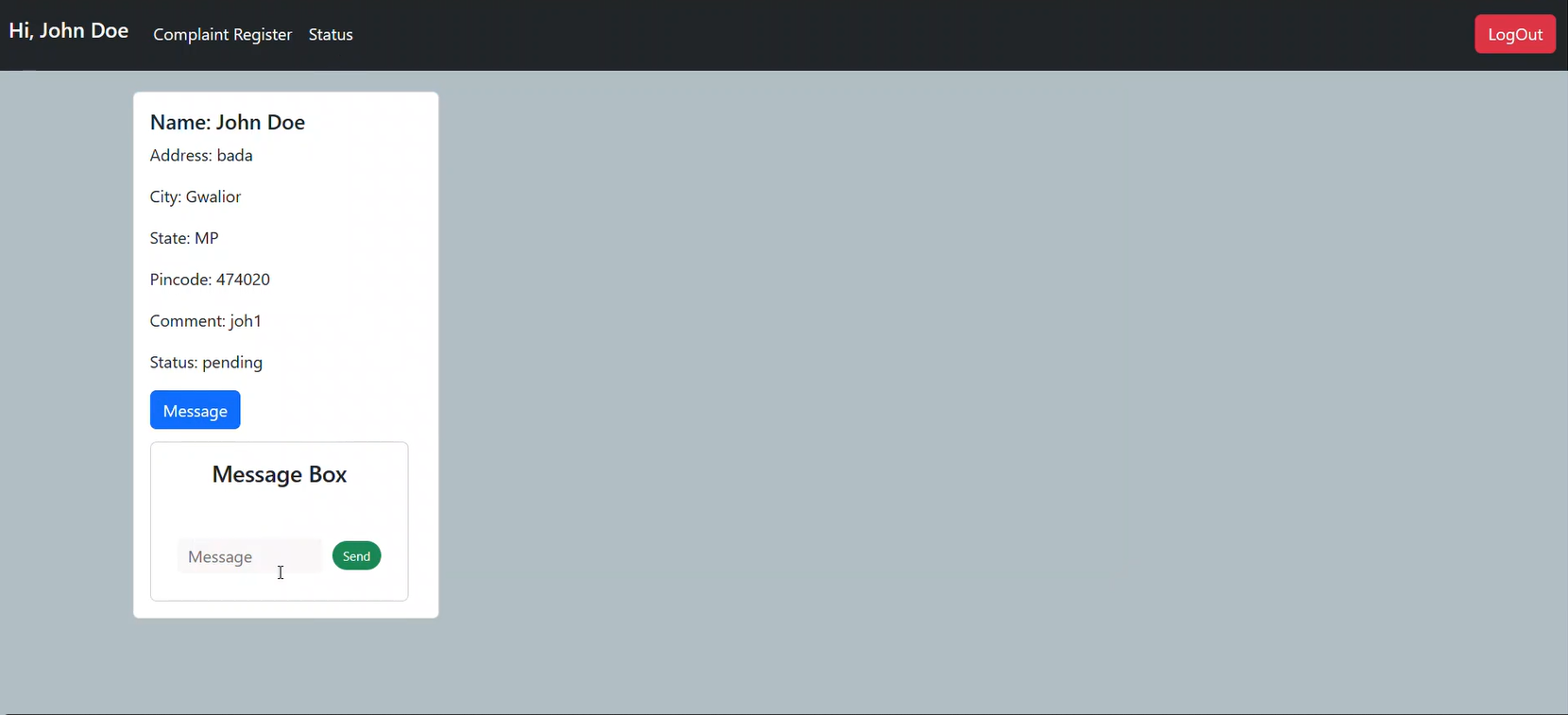
**Admin Dashboard:** Displays a comprehensive list of complaints and user management options.

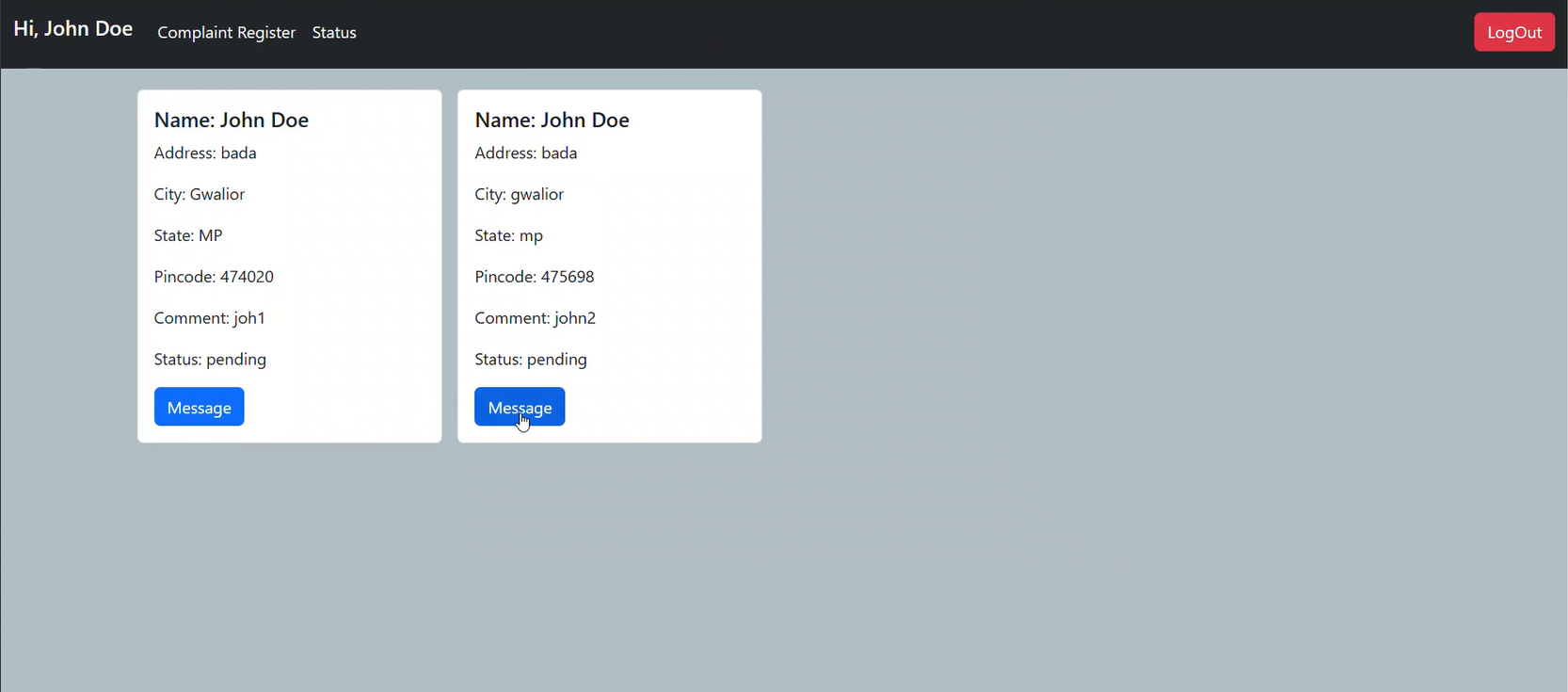
The Admin Dashboard provides system administrators with complete control over user management, complaint tracking, and data analytics. Features include:

* User Management: The ability to view, add, or remove users, agents, and customers.
* Complaint Overview: A comprehensive view of all complaints submitted, including filtering by status and priority.
* Reports and Analytics: Admins can generate reports on complaint trends, resolution times, and overall system performance.

**Screenshots or Demo**







**Testing of the system is carried out through the following strategies:**

* **Unit Testing:** Jest is used for unit testing individual functions and API endpoints.
* **Integration Testing:** Mocha is used to verify interactions between the API and database.
* **End-to-End Testing:** Cypress is used to test the user interface and simulate user actions.

#### 

#### 

#### 

#### **Test Results and Logs**

All tests conducted, both on APIs and frontend, passed successfully during the testing phase. Here are the logs for successful tests:

* API response times are within the acceptable range (<500ms).
* All test cases for complaint registration, user login, and complaint updates passed.
* The UI components render correctly on both desktop and mobile devices, ensuring full responsiveness.

#### 

#### 

#### 

#### 

#### 

#### 

#### 

**API Testing with Postman:**  
Postman is a powerful tool used for testing APIs by simulating HTTP requests and checking the responses from the server. It helps ensure that the API is functioning as expected, validating endpoints, and handling data correctly.

#### 

#### 

#### **Purpose of Using Postman**

* API Endpoint Testing: Postman allows us to send requests (GET, POST, PUT, DELETE, etc.) to the API and examine the responses, including status codes, headers, and body content.
* Automation and Scripting: Postman enables writing tests that can automatically verify the correctness of responses for each API request.
* Environment and Variables: We can define environments and variables (e.g., base URL, authorization tokens) that can be reused across multiple requests, streamlining testing processes.

Testing User Registration Endpoint

* Request:
  + Method: POST
  + URL: /api/register
  + Headers:
    - Content-Type: application/json

Body:{

"username": "SIVARANJANI",

"password": "Password123",

"role": "Customer"

}

Expected Response:

{

"status": "success",

"message": "User registered successfully"

}

Postman Test Script Example:

pm.test("Status code is 200", function () {

pm.response.to.have.status(200);

});

pm.test("Response body contains success message", function () {

pm.response.to.have.body("User registered successfully");

});

Testing Complaint Submission Endpoint

* Request:
  + Method: POST
  + URL: /api/complaint
  + Headers:
    - Content-Type: application/json

Copy code

* + { "customerId": "SIVARANJINI", "title": "Product Issue", "description": "The product delivered is damaged.", "attachments": ["image1.jpg"] }

Expected Response:

Copy code

{ "status": "success", "complaintId": "complaint12345" }

Postman Test Script Example:

Copy code

pm.test("Complaint ID is generated", function () { pm.response.to.have.jsonBody('complaintId'); }); pm.test("Status code is 201", function () { pm.response.to.have.status(201); });

Test Result: The test checks that the complaint ID is returned in the response and that the status code is 201.

* **Live Demo:**https://drive.google.com/file/d/1yjLPJ7dclGxi5k7LH3hGEnuys5-U2Kwv/view

**Known Issues**

1. UI Responsiveness
   * Issue: Some components may not display correctly on smaller devices, such as mobile phones or tablets.
   * Status: This issue is planned for future improvement. The UI design will be revisited to ensure compatibility and responsiveness across various screen sizes.
   * Potential Impact: Users on smaller screens might experience difficulty in navigating the application, which can affect their experience.
2. Session Timeout
   * Issue: Occasionally, users may experience an unexpected session timeout, causing them to be logged out of the application without warning.
   * Status: This is currently under investigation. Developers are looking into potential server-side issues or session management configurations that could be causing these timeouts.
   * Potential Impact: Users may lose progress in complaint registration or management due to session expiration.

### **Future Enhancements**

1. Real-Time Notifications
   * Enhancement: Integrating real-time push notifications to keep users informed about complaint status changes.
   * Expected Benefit: Users will receive instant updates when their complaint status changes (e.g., resolved, in progress), leading to improved engagement and user satisfaction.
   * Implementation Plan: Integrating with services like Firebase or implementing WebSockets for real-time communication.
2. AI Integration
   * Enhancement: Implementing AI-driven suggestions for resolving common complaints.
   * Expected Benefit: The system will intelligently suggest solutions or troubleshooting steps based on common complaints, improving response times and resolution rates.
   * Implementation Plan: Using AI/ML algorithms to analyze complaint data and recommend predefined solutions for frequently reported issues.
3. Multilingual Support
   * Enhancement: Expanding the application to support multiple languages to cater to a broader audience.
   * Expected Benefit: Users from different linguistic backgrounds will be able to interact with the system in their preferred language, enhancing accessibility and user experience.
   * Implementation Plan: Adding language options in the settings and using internationalization (i18n) libraries to manage multiple language translations.

4.Advanced Search Functionality

* Enhancement: Implementing a robust search feature allowing users to filter and search complaints by various parameters such as status, date, category, or priority.
* Expected Benefit: Users will be able to quickly find specific complaints, improving navigation and efficiency in managing large numbers of complaints.
* Implementation Plan: Develop a search interface with filtering options and integrate it with the backend to support fast query execution.

5.Voice Command Integration

* Enhancement: Adding voice command functionality to allow users to register or track complaints using voice recognition technology.
* Expected Benefit: Users will have an alternative, hands-free method to interact with the application, which can improve accessibility and user experience.
* Implementation Plan: Integrating voice recognition APIs (like Google Speech-to-Text or Amazon Alexa) to capture user commands and process them accordingly.

6.Admin Dashboard for Analytics

* Enhancement: Creating an admin dashboard to provide real-time analytics on complaints, such as the number of open complaints, average resolution time, and user feedback.
* Expected Benefit: Admins and managers will have better insights into the performance and health of the complaint management system, leading to data-driven decisions.
* Implementation Plan: Develop an interactive dashboard with charts and graphs, using tools like Chart.js or D3.js, and integrate it with backend analytics.

7.Integration with Third-Party Services

* Enhancement: Expanding the system to integrate with external third-party services like CRM tools or issue tracking systems to streamline complaint resolution workflows.
* Expected Benefit: Users and support agents will benefit from more comprehensive solutions, as complaints can be automatically forwarded to other systems for faster resolution.
* Implementation Plan: Implement APIs to allow seamless communication between the complaint management system and popular third-party tools like Salesforce, Jira, or Zendesk.

8.Offline Mode Support

* Enhancement: Enabling users to access and register complaints while offline and sync data when an internet connection is available.
* Expected Benefit: Users in areas with limited connectivity will still be able to interact with the application, improving usability in regions with poor network coverage.
* Implementation Plan: Implement offline storage and background sync capabilities using technologies like service workers or local storage.

9.Automated Ticket Routing

* Enhancement: Introducing automation to route complaints to the appropriate department or team based on keywords, categories, or severity.
* Expected Benefit: This will streamline the process of complaint handling and reduce manual intervention, ensuring faster response times.
* Implementation Plan: Develop an intelligent routing mechanism based on predefined rules or machine learning models trained on past complaint data.

10.User Feedback and Rating System

* Enhancement: Adding a feedback and rating system that allows users to rate their complaint resolution experience.
* Expected Benefit: This will provide valuable insights into the quality of service, helping to identify areas of improvement.
* Implementation Plan: Implement a simple feedback form post-resolution, with a rating scale and optional comments section..