# **ResolveNow: Your Platform for Online Complaints**

Team Members:

TEAM ID : LTVIP2025TMID59501

|  |  |
| --- | --- |
| **MEMBER** | **RESPONSIBILITY** |
| 1. Bandalapati Sasi Kumar | Testing &Documentation |
| 2. Amirisetty Bhanu Prakash | Resource Gathering |
| 3. Annam Sneha Latha | Devops |
| 4. Alle Sreeja | Frontend , Backend & Database |

**1.Introduction**

**• Project Title:**

**ResolveNow: Your Platform for Online Complaints**

**2. Project Overview**

**• Purpose:**

An online complaint registration and management system is a software application or platform that allows individuals or organizations to submit and track complaints or issues they have encountered. It can help optimize the complaint handling process and empower organizations to develop a safety management system to efficiently resolve customer complaints, while staying in line with industry guidelines and regulatory compliance obligations. It provides a centralized platform for managing complaints, streamlining the complaint resolution process, and improving customer satisfaction.

**• Features:**

* User registration: Users can create accounts to submit complaints and track their progress.
* Complaint submission: Users can enter details of their complaints, including relevant information such name, description of the issue, address etc.
* Tracking and notifications: Users can track the progress of their complaints, view updates, and receive notifications via email or SMS when there are any changes or resolutions.
* User can interact with the agent who has assigned the complaint.
* Assigning and routing complaints: The system assigns complaints to the appropriate department or personnel responsible for handling them. It may use intelligent routing algorithms to ensure efficient allocation of resources.
* Security and confidentiality: The system ensures the security and confidentiality of user data and complaint information through measures such as user authentication, data encryption, access controls, and compliance with relevant data protection regulations.

**3. Architecture**

**SCENARIO:**

Scenario: John, a customer, recently encountered a problem with a product he purchased online. He notices a defect in the item and decides to file a complaint using the Online Complaint Registration and Management System.

1. **User Registration and Login:**

* + John visits the complaint management system's website and clicks on the "Sign Up" button to create a new account.
  + He fills out the registration form, providing his full name, email address, and a secure password.
  + After submitting the form, John receives a verification email and confirms his account.
  + He then logs into the system using his email and password.

1. **Complaint Submission:**
   * Upon logging in, John is redirected to the dashboard where he sees options to register a new complaint.
   * He clicks on the "Submit Complaint" button and fills out the complaint form.
   * John describes the issue in detail, attaches relevant documents or images showcasing the defect, and provides additional information such as his contact details and the product's purchase date.
   * After reviewing the information, John submits the complaint.
2. **Tracking and Notifications:**
   * After submitting the complaint, John receives a confirmation message indicating that his complaint has been successfully registered.
   * He navigates to the "My Complaints" section of the dashboard, where he can track the status of his complaint in real-time.
   * John receives email notifications whenever there is an update on his complaint, such as it being assigned to an agent or its resolution status.
3. **Interaction with Agent:**
   * A customer service agent, Sarah, is assigned to handle John's complaint.
   * Sarah reviews the details provided by John and contacts him through the system's built-in messaging feature.
   * John receives a notification about Sarah's message and accesses the chat window to communicate with her.
   * They discuss the issue further, and Sarah assures John that the company will investigate and resolve the problem promptly.
4. **Resolution and Feedback:**
   * After investigating the complaint, the company identifies the defect in the product and offers John a replacement or refund.
   * John receives a notification informing him of the resolution, along with instructions on how to proceed.
   * He provides feedback on his experience with the complaint handling process, expressing his satisfaction with the prompt resolution and courteous service provided by Sarah.
5. **Admin Management:**
   * Meanwhile, the system administrator monitors all complaints registered on the platform.
   * The admin assigns complaints to agents based on their workload and expertise.
   * They oversee the overall operation of the complaint management system, ensuring compliance with platform policies and regulations.

# **TECHNICAL ARCHITECTURE**



* The technical architecture of our online complaint registration and management app follows a client-server model, where the frontend serves as the client and the backend acts as the server. The frontend encompasses not only the user interface and presentation but also incorporates the axios library to connect with backend easily by using RESTful Apis.
* The frontend utilizes the bootstrap and material UI library to establish real-time and better UI experience for any user whether it is agent, admin or ordinary user working on it.
* On the backend side, we employ Express.js frameworks to handle the server-side logic and communication.
* For data storage and retrieval, our backend relies on MongoDB. MongoDB allows for efficient and scalable storage of user data, including user profiles, for complaints registration, etc. It ensures reliable and quick access to the necessary information during registration of user or any complaints.
* Together, the frontend and backend components, along with socket.io, Express.js, WebRTC API, and MongoDB, form a comprehensive technical architecture for our video conference app. This architecture enables real-time communication, efficient data exchange, and seamless integration, ensuring a smooth and immersive video conferencing experience for all users.

# **ER DIAGRAM**



This is the er diagram of the project which shows the relationship between user and agent

It shows how user which have required fields can raise a complaint by fillings required fields.

It illustrates how these entities relate to each other, helping us understand the underlying database structure and the flow of information within the app. He / She can also communicate with the agent with chat window which follows the message schema which uses userId and complaintId from other schemas.

**4. Setup Instructions**

# **PRE-REQUISITES:**

Here are the key prerequisites for developing a full-stack application using Node.js, Express.js, MongoDB, React.js:

**Node.js and npm:**

Node.js is a powerful JavaScript runtime environment that allows you to run JavaScript code on the server-side. It provides a scalable and efficient platform for building network applications.

Install Node.js and npm on your development machine, as they are required to run JavaScript on the server-side.

Download: https://nodejs.org/en/download/

Installation instructions: https://nodejs.org/en/download/package-manager/

**Express.js:**

Express.js is a fast and minimalist web application framework for Node.js. It simplifies the process of creating robust APIs and web applications, offering features like routing, middleware support, and modular architecture.

Install Express.js, a web application framework for Node.js, which handles server-side routing, middleware, and API development.

Installation: Open your command prompt or terminal and run the following command:

**npm install express**

**MongoDB:**

MongoDB is a flexible and scalable NoSQL database that stores data in a JSON-like format. It provides high performance, horizontal scalability, and seamless integration with Node.js, making it ideal for handling large amounts of structured and unstructured data.

Set up a MongoDB database to store your application's data.

Download: https://www.mongodb.com/try/download/community

Installation instructions: https://docs.mongodb.com/manual/installation/

**React.js:**

React.js is a popular JavaScript library for building user interfaces. It enables developers to create interactive and reusable UI components, making it easier to build dynamic and responsive web applications.

Install React.js, a JavaScript library for building user interfaces.

Follow the installation guide: <https://reactjs.org/docs/create-a-new-react-app.html>

**HTML, CSS, and JavaScript**: Basic knowledge of HTML for creating the structure of your app, CSS for styling, and JavaScript for client-side interactivity is essential.

**Database Connectivity**: Use a MongoDB driver or an Object-Document Mapping (ODM) library like Mongoose to connect your Node.js server with the MongoDB database and perform CRUD (Create, Read, Update, Delete) operations. To Connect the Database with Node JS go through the below provided link:

https://www.section.io/engineering-education/nodejs- mongoosejs-mongodb/

**Front-end Framework**: Utilize Reactjs to build the user-facing part of the application, including entering complaints, status of the complaints, and user interfaces for the admin dashboard.

For making better UI we have also used some libraries like material UI and boostrap.

**Version Control**: Use Git for version control, enabling collaboration and tracking changes throughout the development process. Platforms like GitHub or Bitbucket can host your repository.

Git: Download and installation instructions can be found at: https://git-scm.com/downloads

**Development Environment**: Choose a code editor or Integrated Development Environment (IDE) that suits your preferences, such as Visual Studio Code, Sublime Text, or WebStorm.

• Visual Studio Code: Download from <https://code.visualstudio.com/download>

To run the existing Video Conference App project downloaded from GitHub:

Follow below steps:

Clone the Repository:

* Open your terminal or command prompt.
* Navigate to the directory where you want to store the e-commerce app.
* Execute the following command to clone the repository:

**git clone**: <https://github.com/awdhesh-student/complaint-registery.git>

Install Dependencies:

• Navigate into the cloned repository directory:

cd complaint-registery

• Install the required dependencies by running the following commands:

cd frontend

npm install

cd ../backend

npm install

Start the Development Server:

• To start the development server, execute the following command:

npm start

• The online complaint registration and management app will be accessible at <http://localhost:3000>

You have successfully installed and set up the online complaint registration and management app on your local machine. You can now proceed with further customization, development, and testing as needed.

**5.PROJECT STRUCTURE:**

The first image is of frontend part which is showing all the files and folders that have been used in UI development

The second image is of Backend part which is showing all the files and folders that have been used in backend development





# **APPLICATION FLOW:**

**Online Complaint Registration and Management System**

1. **Customer/Ordinary User:**
   * **Role:** Create and manage complaints, interact with agents, and manage profile information.
   * **Flow:**
     1. **Registration and Login:**
        + Create an account by providing necessary information such as email and password.
        + Log in using the registered credentials.
     2. **Complaint Submission:**
        + Fill out the complaint form with details of the issue, including description, contact information, and relevant attachments.
        + Submit the complaint for processing.
     3. **Status Tracking:**
        + View the status of submitted complaints in the dashboard or status section.
        + Receive real-time updates on the progress of complaints.
     4. **Interaction with Agents:**
        + Connect with assigned agents directly using the built-in messaging feature.
        + Discuss complaints further and provide additional information or clarification.
     5. **Profile Management:**
        + Manage personal profile information, including details and addresses.
2. **Agent:**
   * **Role:** Manage complaints assigned by the admin, communicate with customers, and update complaint statuses.
   * **Flow:**
     1. **Registration and Login:**
        + Create an account using email and password.
        + Log in using the registered credentials.
     2. **Complaint Management:**
        + Access the dashboard to view and manage complaints assigned by the admin.
        + Communicate with customers regarding their complaints through the chat window.
     3. **Status Update:**
        + Change the status of complaints based on resolution or progress.
        + Provide updates to customers regarding the status of their complaints.
     4. **Customer Interaction:**
        + Respond to inquiries, resolve issues, and address feedback from customers.
3. **Admin:**
   * **Role:** Oversee the overall operation of the complaint registration platform, manage complaints, users, and agents, and enforce platform policies.
   * **Flow:**
     1. **Management and Monitoring:**
        + Monitor and moderate all complaints submitted by users.
        + Assign complaints to agents based on workload and expertise.
     2. **Complaint Assignment:**
        + Assign complaints to the desired agents for resolution.
        + Ensure timely and efficient handling of complaints.
     3. **User and Agent Management:**
        + Manage user and agent accounts, including registration, login, and profile information.
        + Enforce platform policies, terms of service, and privacy regulations.
     4. **Continuous Improvement:**
        + Implement measures to improve the platform's functionality, user experience, and security measures.
        + Address any issues or concerns raised by users or agents for better service delivery

**5.API Documentation**

**Overview**

The system uses RESTful APIs to manage authentication, complaint registration, assignments, messaging, and admin controls. All requests and responses are in JSON format.

**1. Format**

* All requests and responses use **JSON** format.
* Data is sent in the request body and received in JSON structure.

**2. Authentication APIs**

* **POST /register** – Register a new user
* **POST /login** – User login and receive token

**3. Complaint APIs**

* **POST /complaints** – Submit a new complaint
* **GET /complaints/:userId** – Get all complaints for a user
* **PUT /complaints/:id** – Update complaint details
* **DELETE /complaints/:id** – Delete a complaint

**4. Assignment APIs (Admin only)**

* **GET /assignments** – View assigned complaints
* **POST /assignments** – Assign a complaint to an agent
* **PUT /assignments/:id** – Change complaint status

**5. Chat APIs**

* **POST /messages** – Send a message
* **GET /messages/:complaintId** – Get all messages for a complaint

**6. User Management (Admin only)**

* **GET /users** – List all users and agents
* **PUT /users/:id** – Edit a user profile or role
* **DELETE /users/:id** – Delete a user

**7. Error Handling**

* **400** – Bad Request (missing or invalid data)
* **404** – Not Found (endpoint or data missing)
* **500** – Server Error (unexpected issue)
* All errors include a short message to explain the problem

**Milestone 1:**

**Project Setup and Configuration:**

1. **Create project folders and files:**

Now, firstly create the folders for frontend and backend to write the respective code and install the essential libraries.

* + Client folders.
  + Server folders.

1. **Install required tools and software:**

For the backend to function well, we use the libraries mentioned in the prerequisites. Those libraries includes

* + Node.js.
  + MongoDB.
  + Bcrypt
  + Body-parser

Also, for the frontend we use the libraries such as

* + React Js.
  + Material UI
  + Bootstrap
  + Axios

After the installation of all the libraries, the package.json files for the frontend looks like the one mentioned below.



After the installation of all the libraries, the package.json files for the backend looks like the one mentioned below.



**Milestone 2:**

**Backend Development**

* **Set Up Project Structure:**
  + Create a new directory for your project and set up a package.json file using npm init command.
  + Install necessary dependencies such as Express.js, Mongoose, and other required packages.
* **Create Express.js Server:**
  + Set up an Express.js server to handle HTTP requests and serve API endpoints.
  + Configure middleware such as body-parser for parsing request bodies and cors for handling cross-origin requests.
* **Define API Routes:**
  + Create separate route files for different API functionalities such as authentication, stock actions, and transactions.
  + Implement route handlers using Express.js to handle requests and interact with the database.

#### **Implement Data Models:**

* + Define Mongoose schemas for the different data entities like Bank, users, transactions, deposits and loans.
  + Create corresponding Mongoose models to interact with the MongoDB database.
  + Implement CRUD operations (Create, Read, Update, Delete) for each model to perform database operations.

#### **User Authentication:**

* + Implement user authentication using strategies like JSON Web Tokens (JWT) or session-based authentication.
  + Create routes and middleware for user registration, login, and logout.
  + Set up authentication middleware to protect routes that require user authentication.

#### **Handle new transactions:**

* + Allow users to make transactions to other users using the user’s account id.
  + Update the transactions and account balance dynamically in real-time.

#### **Admin Functionality:**

* + Implement routes and controllers specific to admin functionalities such as fetching all the data regarding users, transactions, stocks and orders.

#### **Error Handling:**

* + Implement error handling middleware to catch and handle any errors that occur during the API requests.
  + Return appropriate error responses with relevant error messages and HTTP status codes.

**Milestone 3:**

**Database Development**

1. **User Schema:**
   * The user schema defines the structure of user data stored in the database. It includes fields such as name, email, password, phone, and userType.
   * Each user must provide a name, email, password, phone number, and userType (e.g., customer, agent, admin).
   * User data is stored in the "user\_Schema" collection in the MongoDB database.
2. **Complaint Schema:**
   * The complaint schema specifies the format of complaint data registered by users.
   * It contains fields like userId, name, address, city, state, pincode, comment, and status.
   * Complaints are associated with users through the userId field, and each complaint must have a name, address, city, state, pincode, comment, and status.
   * Complaint data is stored in the "complaint\_schema" collection in the MongoDB database.
3. **Assigned Complaint Schema:**
   * The assigned complaint schema defines how complaints are assigned to agents for resolution.
   * It includes fields such as agentId, complaintId, status, and agentName.
   * Each assigned complaint is linked to a specific agent (identified by agentId) and complaint (identified by complaintId).
   * The status field indicates the current status of the assigned complaint.
   * Assigned complaint data is stored in the "assigned\_complaint" collection in the MongoDB database.
4. **Chat Window Schema:**
   * The chat window schema governs the structure of messages exchanged between users and agents regarding specific complaints.
   * It comprises fields like name, message, and complaintId.
   * Messages are associated with a complaint through the complaintId field, allowing for easy tracking and retrieval of chat history for each complaint.
   * Message data is stored in the "message" collection in the MongoDB database.

**Milestone 4:**

**Frontend Development:**

1. **Setup React Application:**

Bringing Customer Care Registry to life involves a three-step development process. First, a solid foundation is built using React.js. This includes creating the initial application structure, installing necessary libraries, and organizing the project files for efficient development. Next, the user interface (UI) comes to life. To start the development process for the frontend, follow the below steps.

* + Install required libraries.
  + Create the structure directories.

1. **Design UI components:**

Reusable components will be created for all the interactive elements you'll see on screen, from stock listings and charts to buttons and user profiles. Next, we'll implement a layout and styling scheme to define the overall look and feel of the application. This ensures a visually-appealing and intuitive interface. Finally, a navigation system will be integrated, allowing you to effortlessly explore different sections of Customer Care Registry, like making specific complaints or managing your Product complaints.

1. **Implement frontend logic:**

In the final leg of the frontend development, we'll bridge the gap between the visual interface and the underlying data. It involves the below stages.

* + Integration with API endpoints.
  + Implement data binding.

**Milestone 5:**

**Project Implementation:**

On completing the development part, we then run the application one last time to verify all the functionalities and look for any bugs in it. The user interface of the application looks a bit like the one’s provided below.

**7. Running the Application**

**Frontend:** npm start -in the client directory  **Backend:** npm run debug- in the server directory.

**8.User Interface**

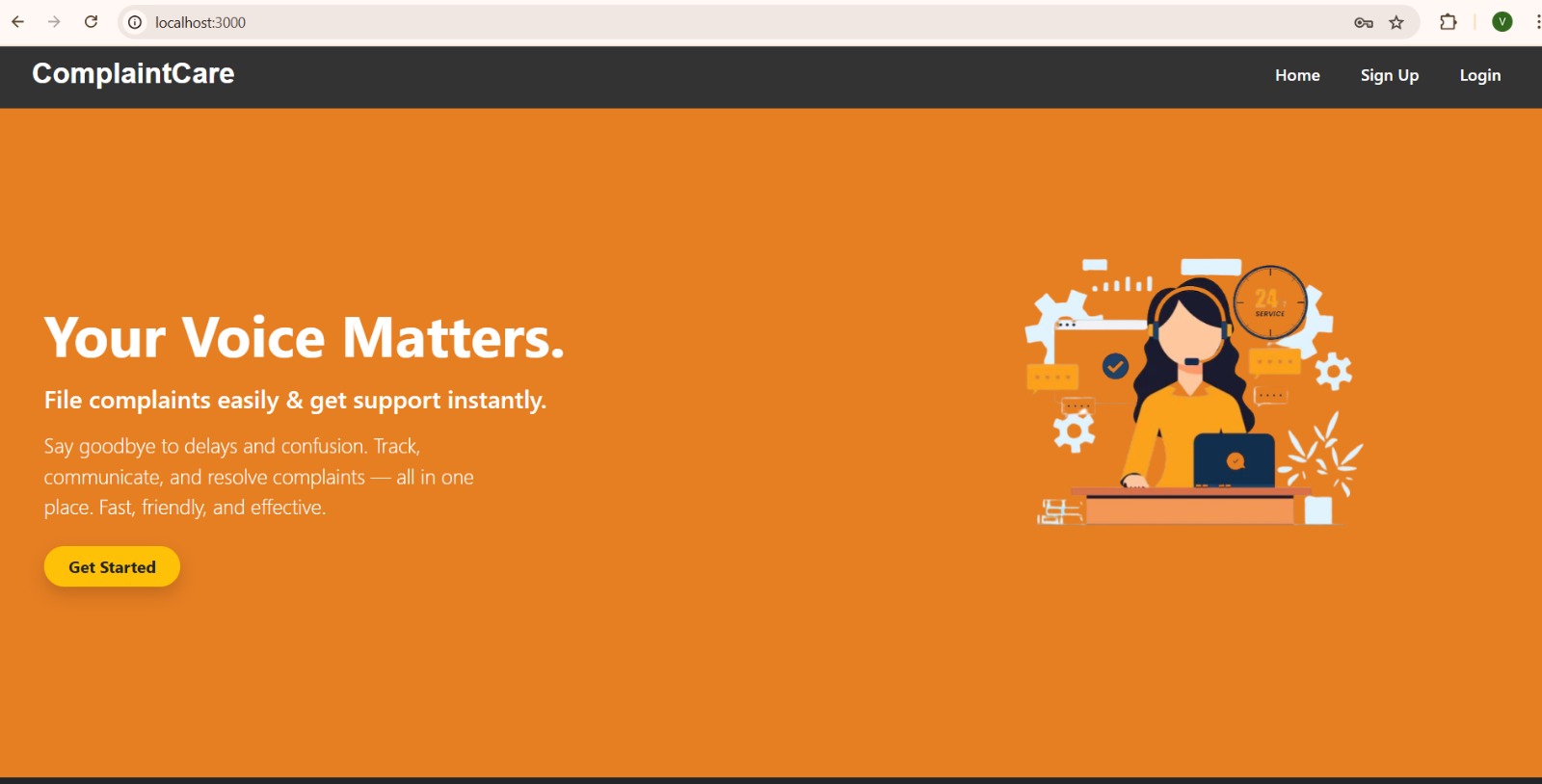
****

Fig: Landing Page(Home Page)

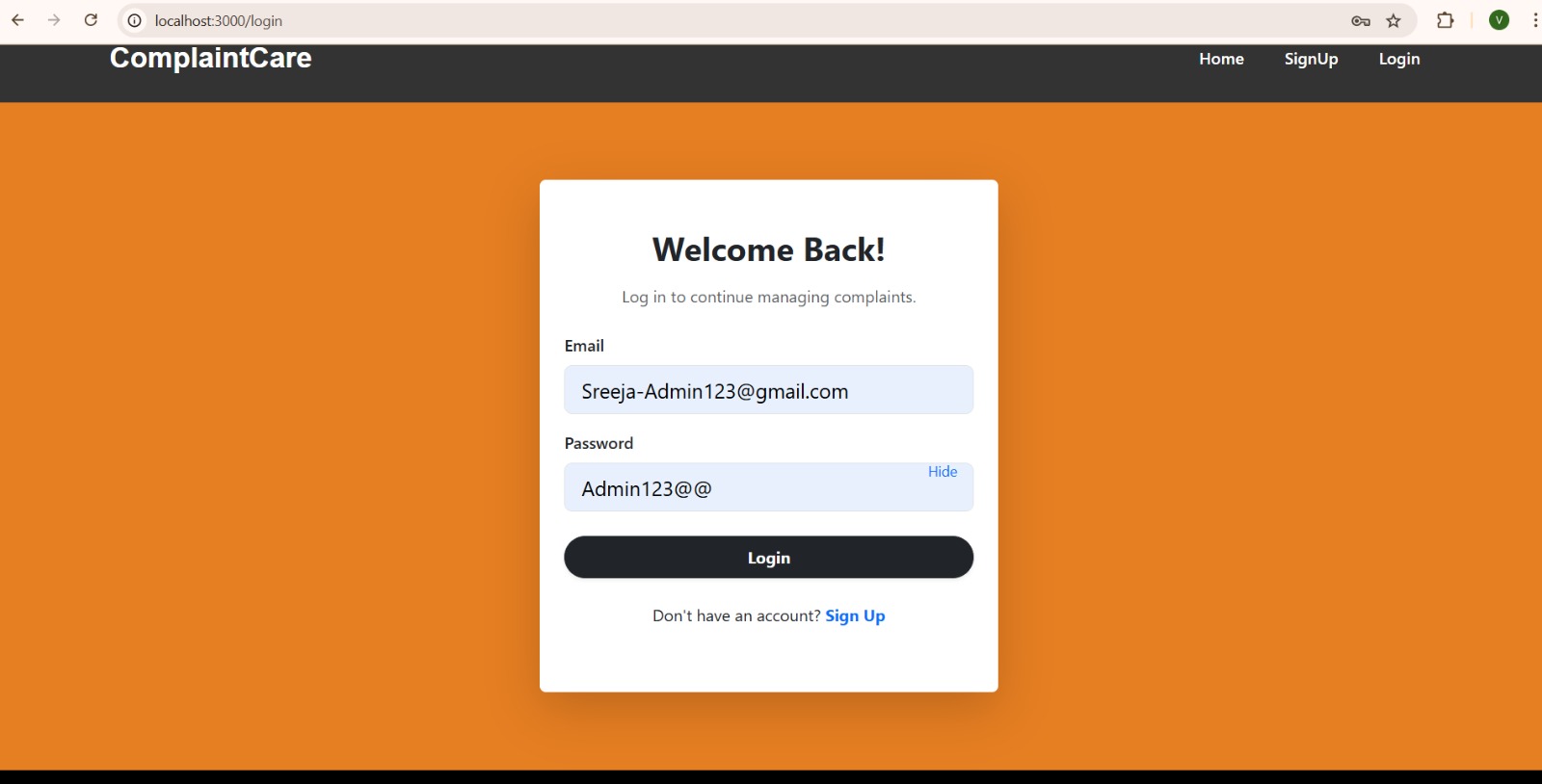


Fig: Login Page

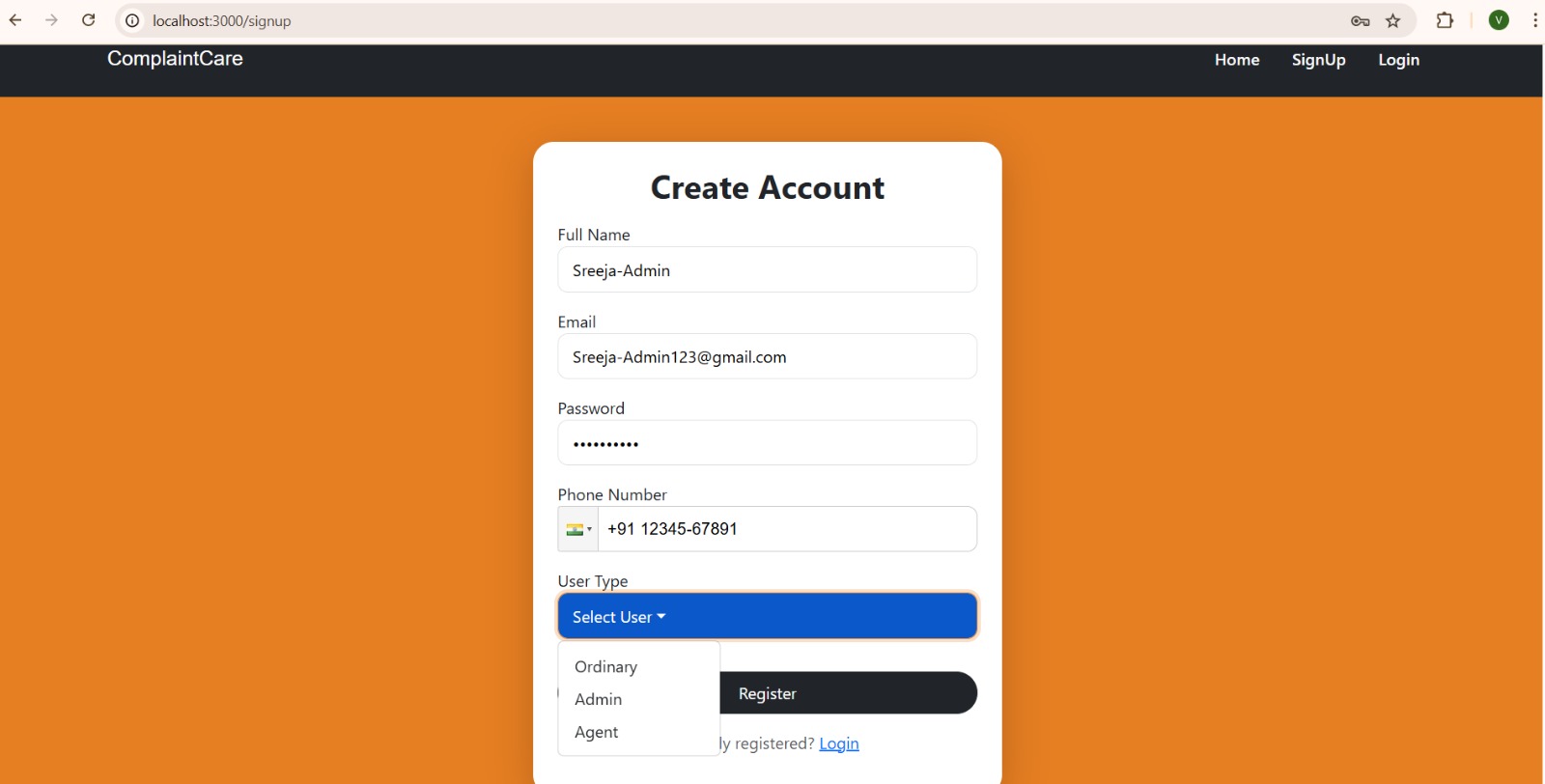


Fig: Sign up Page

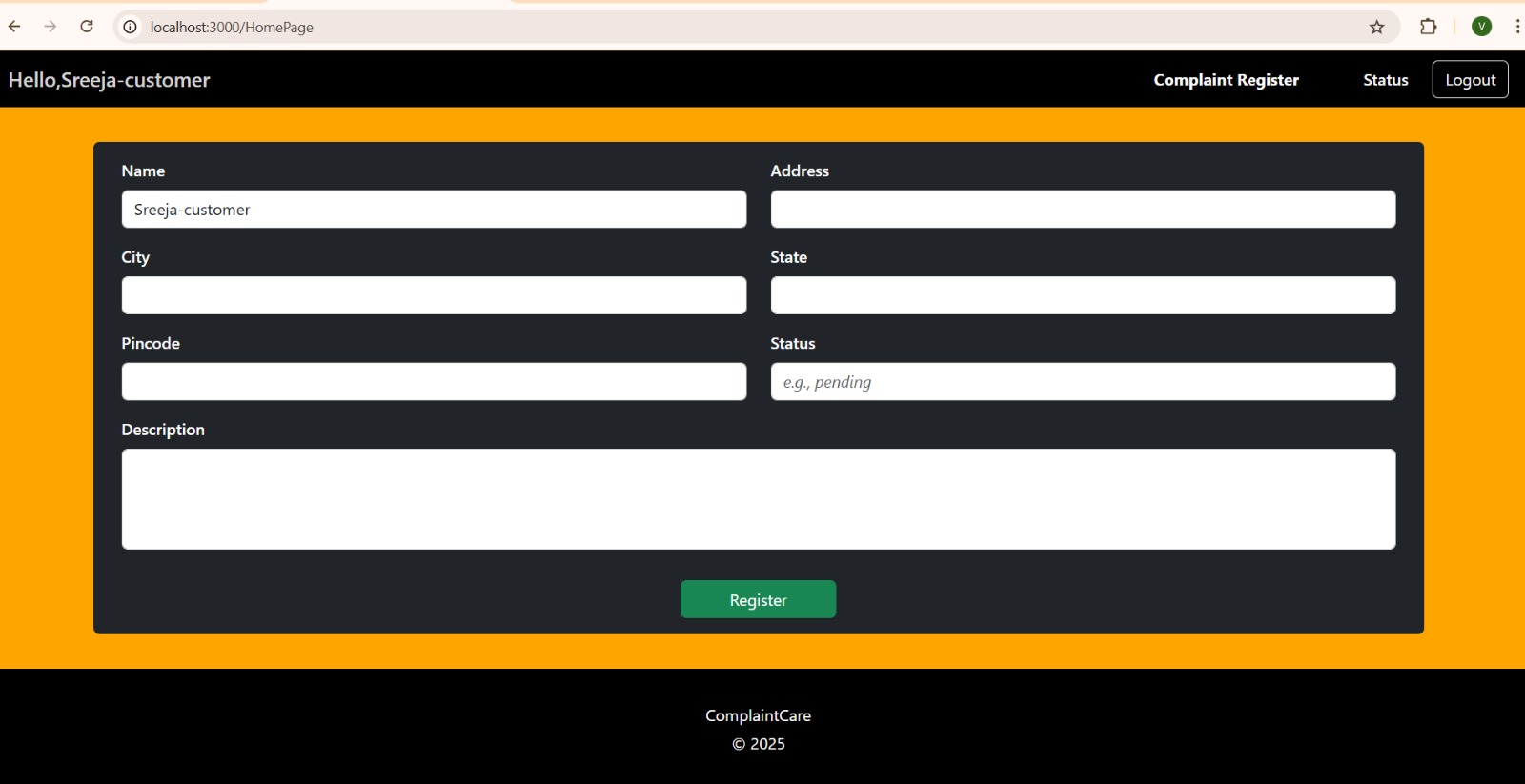


Fig: Customer Complaint Registry Page

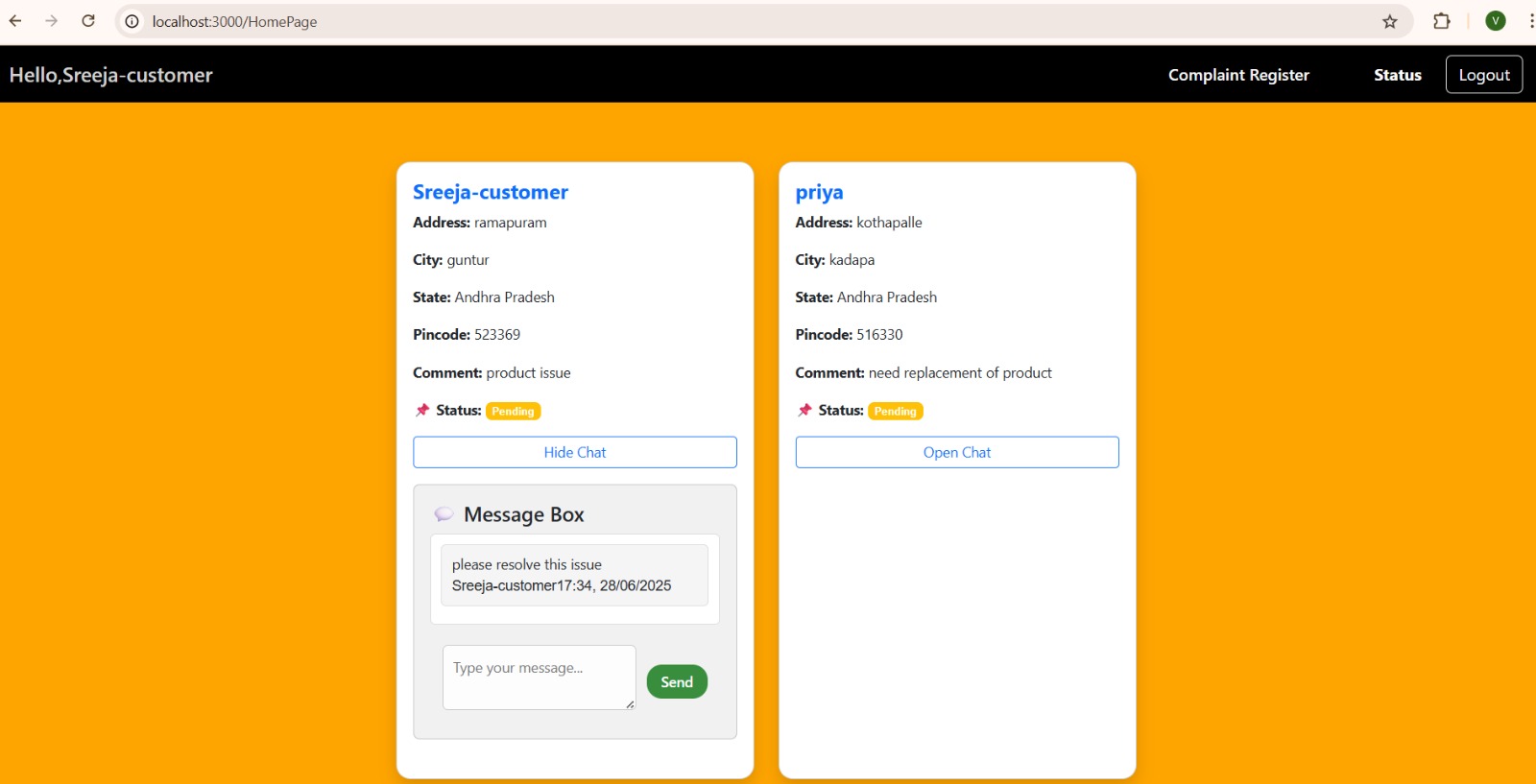


Fig: Customer Complaint Status Page

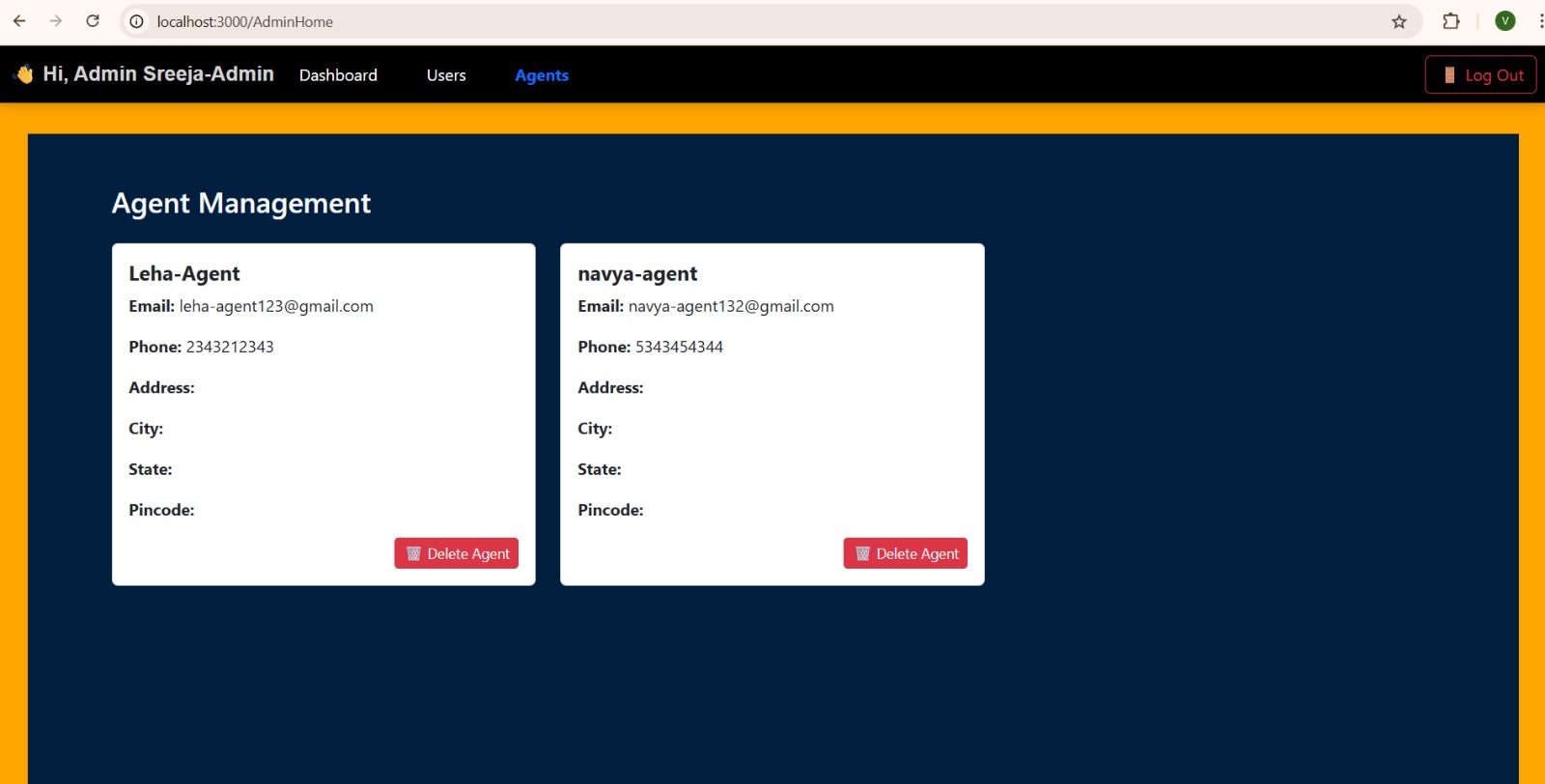


Fig: Admin Agent Management Page

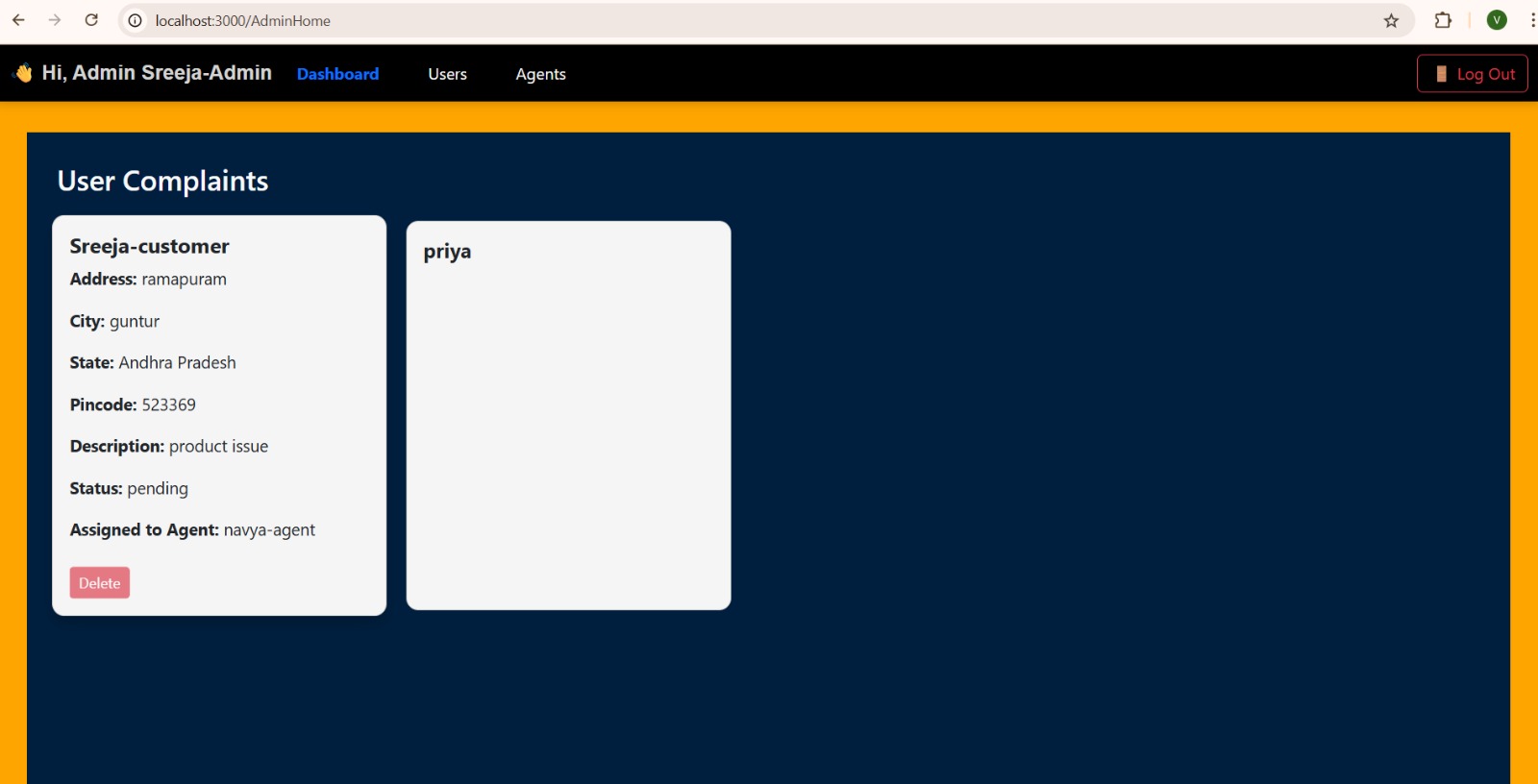
****

Fig: Admin Home Page

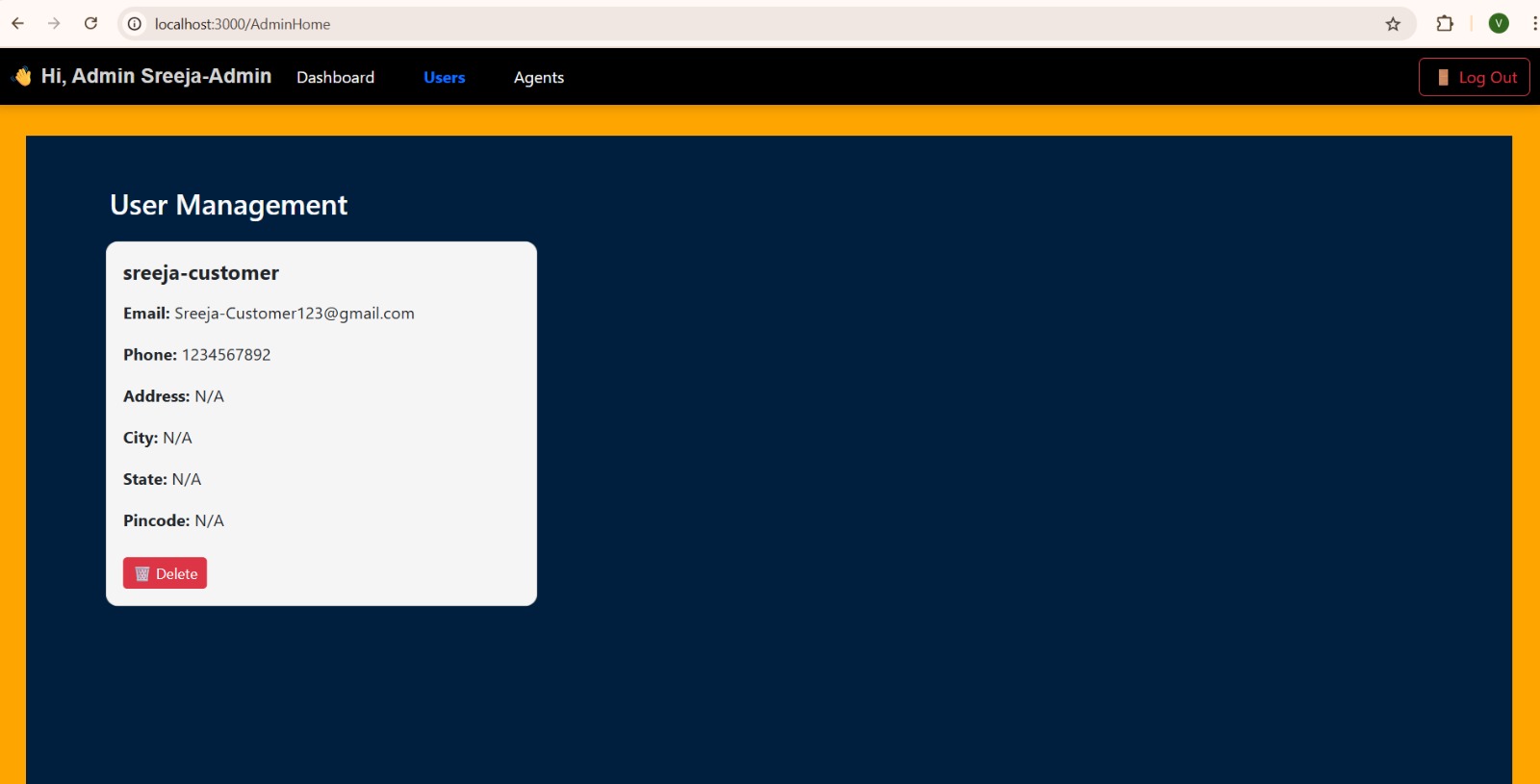
****

Fig: Admin User Management Page

****

Fig: Agent Home Page(Complaint completed status)

****

Fig: Agent Home Page(Complaint pending status)

**9. Authentication**

**Authentication:**  
Users (customers, agents, and admins) are required to register with an email and password. During login, the system verifies these credentials to authenticate the user. Upon successful login, the user is issued a token that securely maintains their session and identity within the system.

**Authorization:**  
Each authenticated user is assigned a role—such as customer, agent, or admin—which determines their level of access. The system enforces permissions accordingly:

* Customers can submit complaints, track their status, and communicate with agents.
* Agents can manage complaints assigned to them, update complaint status, and message users.
* Admins can assign complaints, manage users and agents, and oversee system-wide operations.

This role-based approach ensures secure, role-specific access and helps prevent unauthorized use of system features.

**10. Testing Strategy**

• **Unit Testing**:  
We check each part of the code separately, like login logic or complaint submission, to make sure each one works by itself.

• **Integration Testing**:  
We test how different parts of the system work together. For example, we check if submitting a complaint also saves it properly in the database.

• **End-to-End Testing**:  
We follow the full user journey—like registering, submitting a complaint, chatting with an agent—to make sure everything works as expected from start to finish.

• **Acceptance Testing**:  
We test the system from the user's point of view to confirm it meets all the required features and is easy to use.

**11.Demo**

The demo of the app is available at:

[**https://drive.google.com/file/d/1643Pm2Qn3DJqQMF5yl8IBbZiG91qqCeD/view?usp=drivesdk**](https://drive.google.com/file/d/1643Pm2Qn3DJqQMF5yl8IBbZiG91qqCeD/view?usp=drivesdk)

**12. Known Issues**

**Backend (Node.js + Express)**

* Uncaught Exceptions: Some server errors may not be gracefully handled, risking service disruption
* MongoDB Connectivity: Intermittent disconnects without automatic reconnection mechanisms
* Lack of Rate Limiting: APIs are open to brute force and abuse due to missing throttling logic

**Frontend (React.js)**

* Cross-Browser Compatibility: UI elements may not render properly on legacy browsers (e.g., Internet Explorer)
* Responsiveness Issues: Layout inconsistencies on small mobile screens and tablets
* State Overload: Complex complaint flows can lead to buggy UI due to poor state isolation

**Database (MongoDB)**

* Inconsistent Transactions: Absence of multi-document transactions might lead to partial complaint or message saves
* Slow Data Retrieval: Unindexed or nested queries reduce performance on complaint lists or dashboards
* Scalability Challenges: Without proper sharding or schema refactoring, large datasets may throttle read/write speeds

**Security & Access Control**

* Weak Hashing Configuration: Use of basic bcrypt settings may not be sufficient under targeted attacks
* RBAC Gaps: Role-based access control may not fully restrict agent/admin user scopes
* Input Validation Risks: Inadequate sanitization exposes components to cross-site scripting (XSS) and injection attacks

**13. Future Enhancements**

**Mobile Optimization**

* Launch Android & iOS apps for complaint submission and status tracking
* Optimize UI for smartphones with responsive design and intuitive navigation
* Integrate mobile authentication and real-time push/SMS notifications

**Smart Complaint Management**

* Add AI chatbot for instant user support and guided complaint filing
* Implement advanced filters and keyword search across complaints and chats
* Use analytics to detect complaint trends and automate status prioritization