**RESLOVE NOW (ONLINE COMPLAINT)**

**Introduction:**

Project Name: **Customer Registry App**

**Team Members:**

1. D SWATHI (Backend Developer)
2. K TULASI(Supporter)
3. D UDAY KIRAN (Front End Developer)
4. A SHIVA(Testing)

**Description:**

Welcome to RESLOVE NOW, The Resolve Now Project is an online platform designed to help patients, healthcare providers, and other stakeholders resolve complaints related to healthcare services in a fast and efficient manner. It provides a user-friendly interface that allows individuals to lodge complaints or express concerns about the quality of care they received, issues with healthcare facilities, or any administrative hurdles they encountered.

Collating and analyzing data from diverse customer interactions, the registry becomes a treasure trove of insights. These insights guide strategic decisions, aiding in the development of targeted training programs, refined service protocols, and streamlined processes. Consequently, customer care teams can proactively address recurring issues, offer personalized assistance, and curate a more satisfying experience for each customer.

In a world where customer satisfaction can make or break a brand, the Customer Care Registry emerges as an indispensable asset. It epitomizes a commitment to not only resolving issues but understanding the customer journey holistically. As businesses continue to evolve, the registry remains a steadfast ally, fostering lasting customer relationships built on trust, efficiency, and genuine care.

**Features:**

1. Centralized Complaint Management System

A unified platform to log, track, and resolve patient complaints efficiently.

Easy access for healthcare providers to review and act upon complaints in real time.

2. Automated Complaint Acknowledgement

Immediate confirmation to patients when a complaint is filed.

Provides a sense of transparency and reassurance for the patient that their concern is being addressed.

3. Real-Time Tracking and Updates

Patients can track the status of their complaints through an online portal or mobile app.

Automatic notifications when there are updates on their complaint status.

4. Root Cause Analysis

Identifying the underlying causes of recurring issues, not just addressing the symptoms.

Helps improve patient care processes and eliminate common complaints.

5. Integrated Feedback System

Collecting feedback from patients after their complaint is resolved to ensure satisfaction with the outcome.

Ensures that healthcare providers continuously improve based on real patient experiences.

6. Multichannel Support

Complaints can be filed via multiple channels like phone, email, web forms, or even through social media.

A flexible system that ensures patients can easily raise concerns in a way that's most convenient for them.

7. Personalized Responses and Solutions

Addressing each complaint in a personalized manner, offering tailored resolutions to the patient’s specific issue.

Avoiding generic responses that may frustrate the customer further.

8. Escalation Process

Clear pathways for escalating more serious complaints to senior management or relevant authorities.

Ensures complex issues are dealt with promptly and appropriately.

9. Data Analytics and Reporting

Collecting data from complaints to identify patterns, frequently mentioned issues, or areas that need improvement.

Provides healthcare administrators with insights to refine policies and practices.

10. Patient Education and Communication

Educating patients about the complaint process and how it can be used to improve their healthcare experience.

Ensuring transparent communication about what will happen next in the process.

11. Resolution Timeliness and Accountability

Defining clear timelines for resolving complaints.

Holding healthcare providers accountable for addressing issues within set timeframes

**Purpose:**

Efficient Resolution of Complaints

**Purpose:** The primary goal of the Resolve Now project is to provide a fast, effective, and transparent process for resolving patient complaints.

Why it's important: Healthcare systems often face challenges with addressing complaints in a timely manner, leading to frustration for patients. A structured project helps ensure that complaints are acknowledged quickly and resolved in a way that meets or exceeds patient expectations.

2. Enhance Patient Experience

**Purpose:** To improve the overall patient experience by addressing their concerns in a timely and empathetic manner.

Why it's important: Healthcare is an inherently stressful experience for many patients, and negative experiences can have long-term effects on patient trust and loyalty. By addressing complaints online in a responsive way, the project works to improve patient satisfaction.

3. Transparency and TrustBuilding

**Purpose:** The project aims to create transparency in the complaint-handling process, ensuring that patients feel their voices are heard and that their concerns are taken seriously.

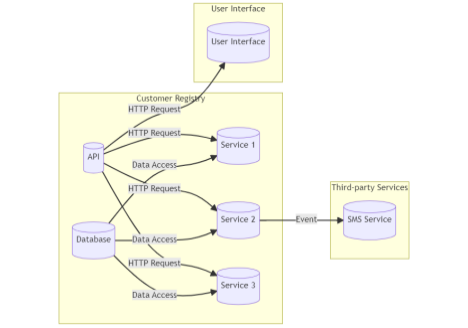
Why it's important: Transparency helps to build trust with patients, showing them that their healthcare provider values their feedback and is committedto resolving issues**.**

4. Improvement in Healthcare Services

**Purpose:** To identify recurring problems and patterns in complaints that can be used to improve the quality of care and services provided.

Why it's important: Healthcare providers can gain insights from complaint data that can lead to improved policies, procedures, and training, reducing the frequency of similar complaints in the future**.**

**Architecture:**

****

A customer registry is a system that stores and manages information about customers or clients. It typically includes details such as names, contact information, addresses, purchase history, preferences, and other relevant data. The technical architecture of a

customer registry involves the components and technologies used to build and operate the system effectively.

Here are some key elements commonly found in a customer registry technical architecture:

**Database:** The database is the central storage component that holds all customer data. It can be a relational database management system (RDBMS) like MySQL or PostgreSQL, a NoSQL database like MongoDB or Cassandra, or a combination of multiple databases depending on the requirements.

**API (Application Programming Interface):** The API serves as an interface for external systems to interact with the customer registry. It exposes a set of endpoints that allow authorized applications or services to perform operations such as creating, updating, retrieving, or deleting customer records. The API handles data validation, authentication, and authorization.

**Services:** Services are responsible for performing specific business logic and operations related to the customer registry. They can include services like customer registration,

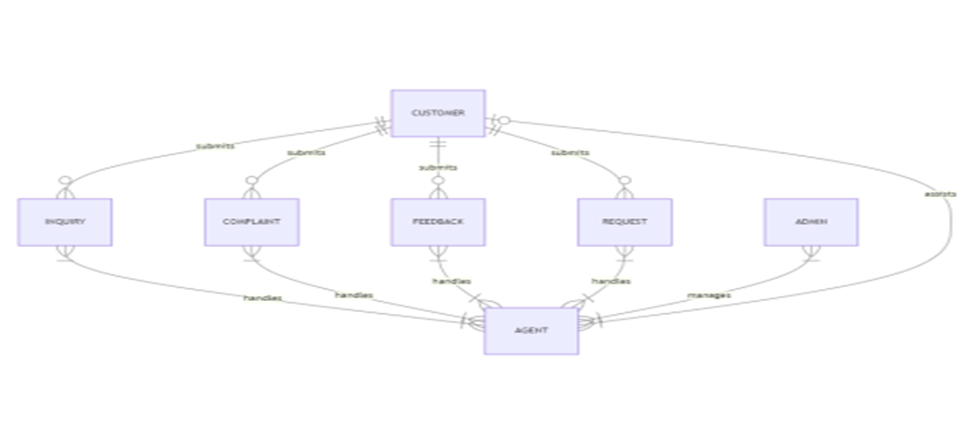
authentication, customer profile management, data validation, and more. These services interact with the database and may communicate with each other as needed.

**User Interface:** The user interface component provides an interface for human users to interact with the customer registry system. It can be a web-based application, a mobile app, or even a command-line interface (CLI). The user interface communicates with the

API to perform actions such as searching for customers, updating information, or generating reports.

**Integration with Third-party Services:** A customer registry may integrate with various third party services to enhance its functionality. For example, it can integrate with email services to send notifications or marketing emails to customers. It can also integrate with SMS gateways for sending text messages. These integrations are typically based on event driven architecture, where certain actions or events trigger interactions with the third party services.

**Security and Authentication:** Security is a crucial aspect of a customer registry. It includes mechanisms for authentication and authorization to ensure that only authorized users or systems can access or modify customer data. Techniques like encryption, secure communication protocols (HTTPS), and user permissions play a vital role in securing the system.

**ERDiagram**

**Installation:**

**Install Node.js and npm**

First, you need to install Node.js, which includes npm (Node Package Manager).

* Download and install Node.js from here.

**Set Project Directory**

Create a new directory for your project:

mkdir my-mern-app

cd my-mern-app

**Initialize your Project**

Initialize a new Node.js project with npm:

npm init -y

**Install Backend Dependencies**

Install Express.js and other necessary backend packages:

npm install express mongoose

**Create Backend Files**

Create a basic structure for your backend:

* server.js: Main server file
* models/: Directory for Mongoose models
* routes/: Directory for Express routes

**Install Frontend Dependencies**

Install Create React App to set up the frontend:

npx create-react-app client

cd client

**Run the React Application**

Start the React development server:

npm start

**Connect Frontend to Backend**

In the React app, use Axios or Fetch to make API calls to your Express server. For example, create a new file api.js in your React app:

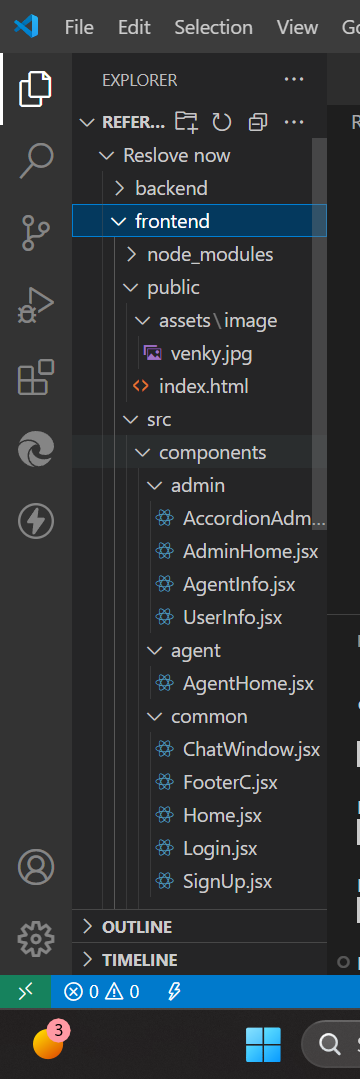
**Start Both Servers**

Run both servers (backend and frontend) simultaneously. You can use concurrently:

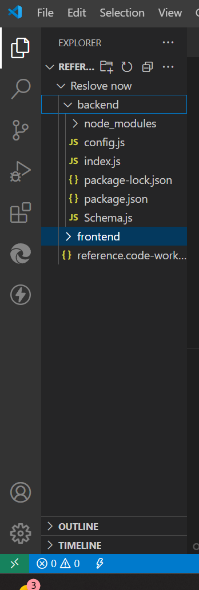
npm start

**Folder Structure:**

**Front end:**



**Backend:**



**Running the Application:**

**Frontend (Client) Server:**

Navigate to the client directory in your terminal.

Run the following command to start the frontend server:

npm start

This will start the frontend application, usually at <http://localhost:3000>.

**Backend (Server) Server:**

Navigate to the server directory in your terminal.

Run the following command to start the backend server:

npm start

This will start the backend API server, usually at http://localhost:5000 (or another port if specified).

Ensure that you have all the dependencies installed by running npm install in both the client and server directories before starting the servers.

**API Documentation:**

API (Application Programming Interface): The API serves as an interface for external systems to interact with the customer registry. It exposes a set of endpoints that allow authorized applications or services to perform operations such as creating, updating, retrieving, or deleting customer records. The API handles data validation, authentication, and authorization.

Services: Services are responsible for performing specific business logic and operations related to the customer registry. They can include services like customer registration,

authentication, customer profile management, data validation, and more. These services interact with the database and may communicate with each other as needed.

User Interface: The user interface component provides an interface for human users to interact with the customer registry system. It can be a web-based application, a mobile app, or even a command-line interface (CLI). The user interface communicates with the

API to perform actions such as searching for customers, updating information, or generating reports.

Integration with Third-party Services: A customer registry may integrate with various third party services to enhance its functionality. For example, it can integrate with email services to send notifications or marketing emails to customers. It can also integrate with SMS gateways for sending text messages. These integrations are typically based on event driven architecture, where certain actions or events trigger interactions with the third party services.

Security and Authentication: Security is a crucial aspect of a customer registry. It includes mechanisms for authentication and authorization to ensure that only authorized users or systems can access or modify customer data. Techniques like encryption, secure communication protocols (HTTPS), and user permissions play a vital role in securing the system**.**

**Authentication:**

**User Authentication:** User provides credentials (username and password) to the server.

**Session Creation:** If the credentials are correct, the server creates a session and stores user information in it.

**Session ID Issuance:** The server sends a session ID to the user.

**Session ID Storage:** The user stores the session ID (typically in cookies).

**Access Protected Routes:** The user includes the session ID in subsequent requests to access protected routes.

**User Interface:**

**Simplicity and Clarity**

* **Minimalist Design:** Use a clean and simple design to avoid overwhelming users with too much information. Focus on essential elements and remove any unnecessary clutter.
* **Clear Navigation:** Ensure that navigation is intuitive and straightforward. Use clear labels and icons to guide users through the app.

**User-Centric Design**

* **Personalization:** Tailor the user experience based on individual preferences and behaviors. Provide personalized recommendations and insights.
* **Accessibility:** Design the UI to be accessible to all users, including those with disabilities. Use appropriate color contrasts, font sizes, and screen reader compatibility.

**Security and Trust**

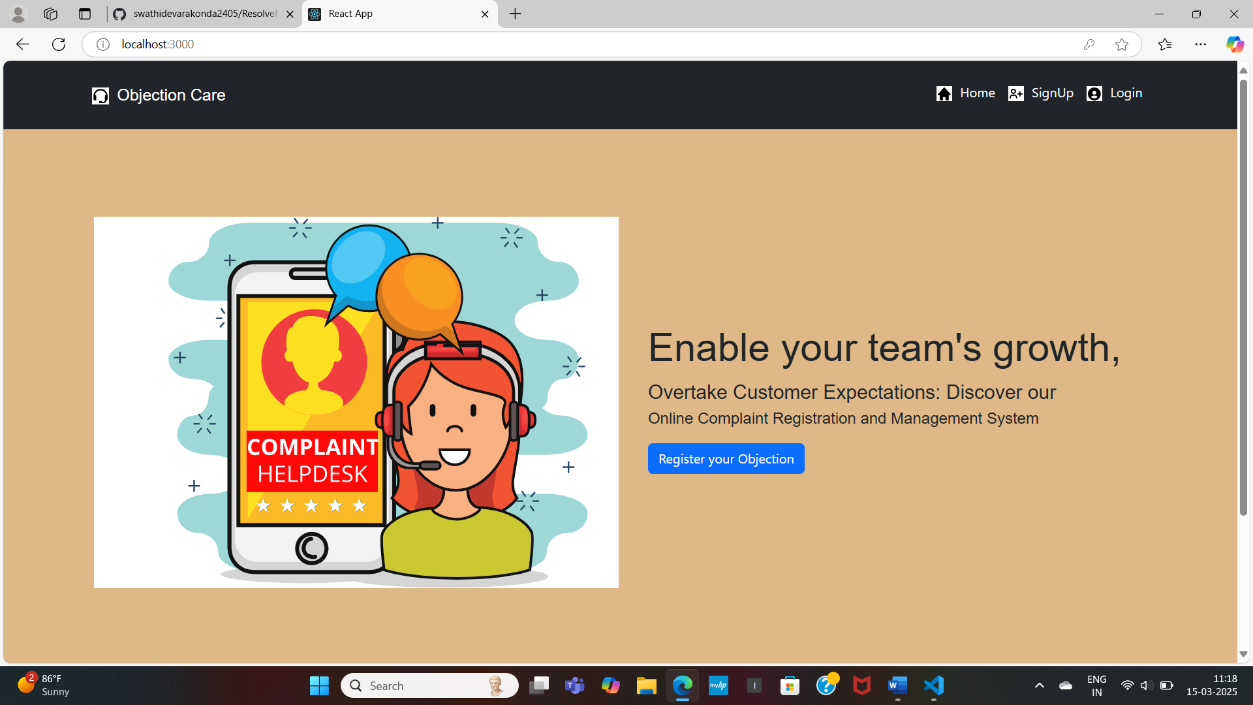
* **Secure Authentication:** Implement strong authentication methods such as two-factor authentication (2FA) or biometric authentication (fingerprint, facial recognition) to ensure user security.
* **Transparency:** Clearly communicate security measures and privacy policies to build trust with users.

**Consistency**

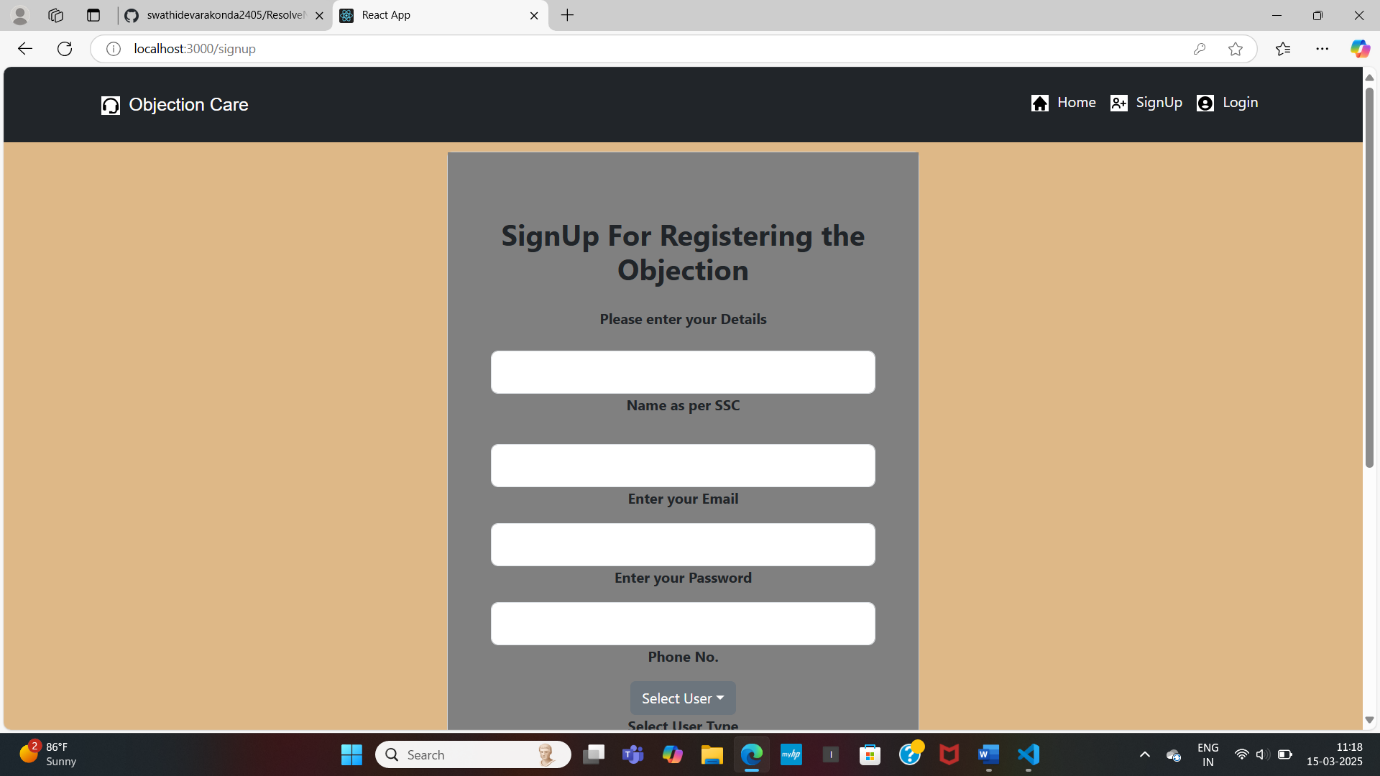
* **Uniform Design Language:** Maintain consistency in design elements such as colors, fonts, and button styles throughout the app. This helps users feel familiar and comfortable with the interface.
* **Consistent User Experience:** Ensure that the user experience is consistent across different devices and platforms (e.g., mobile, web).

**Demo:**

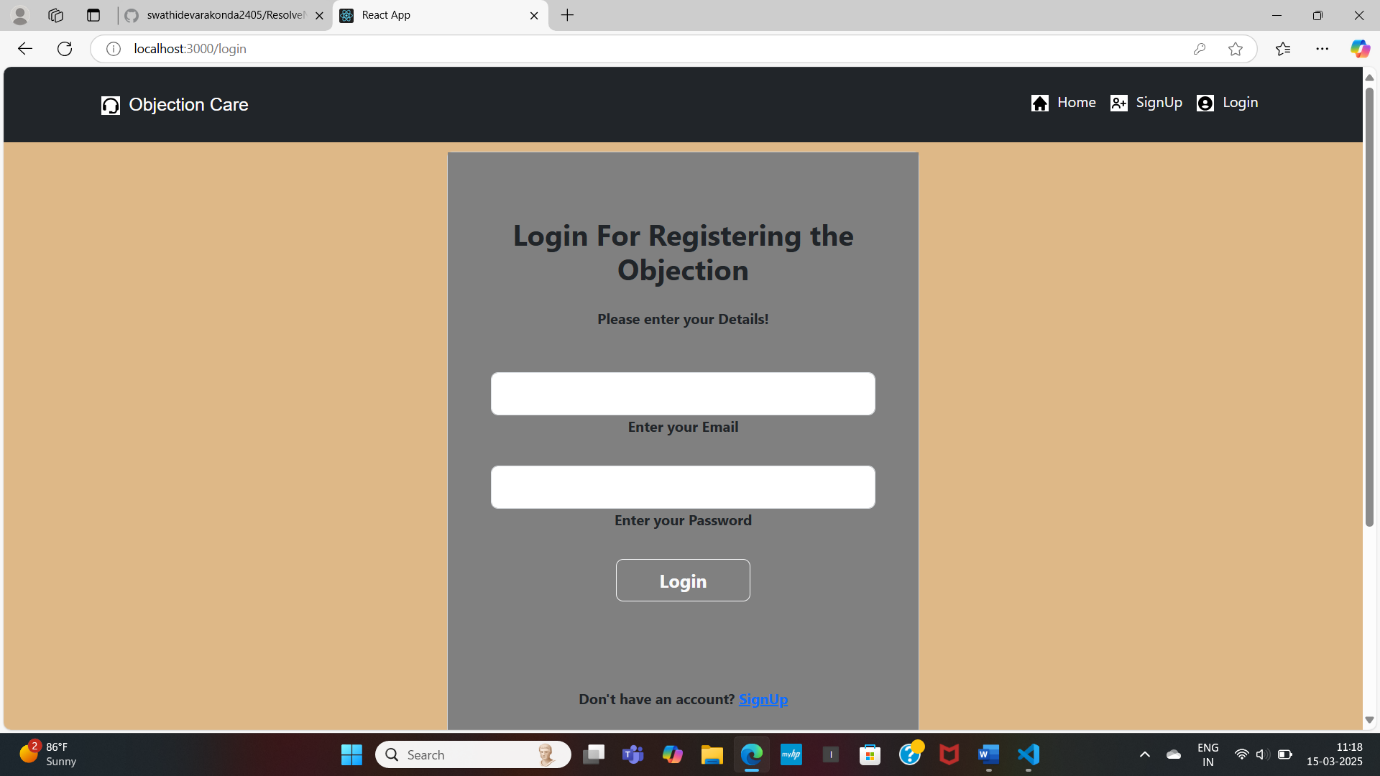
**Home:**



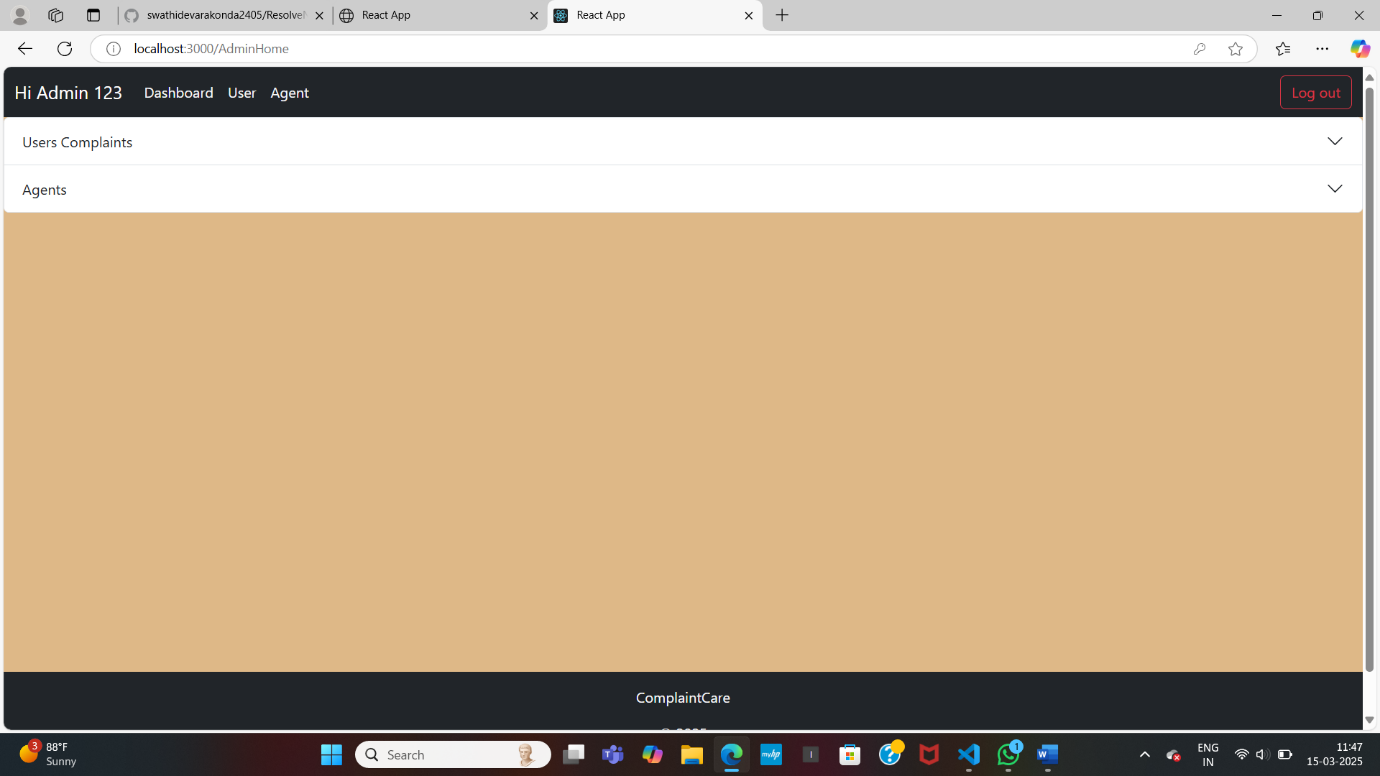
**Signup:**

****

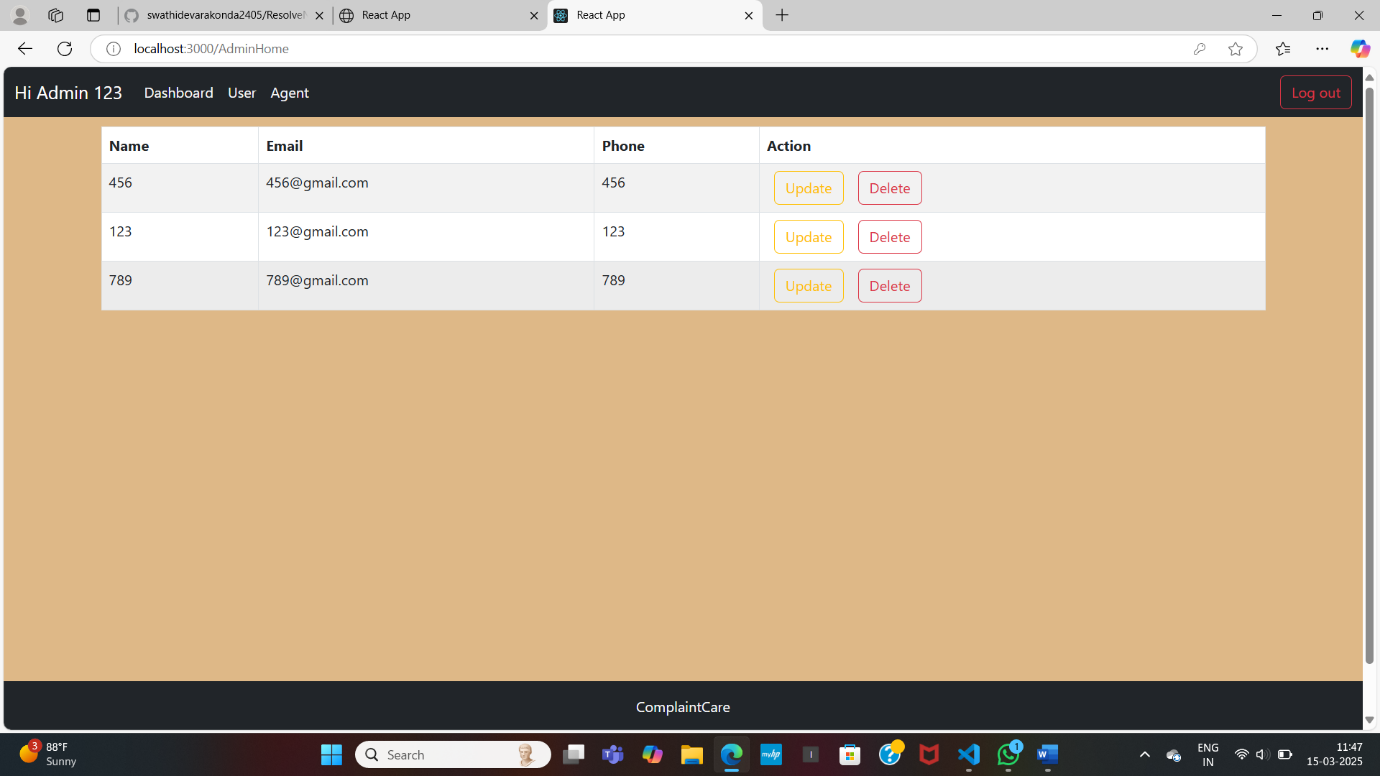
**Login:**



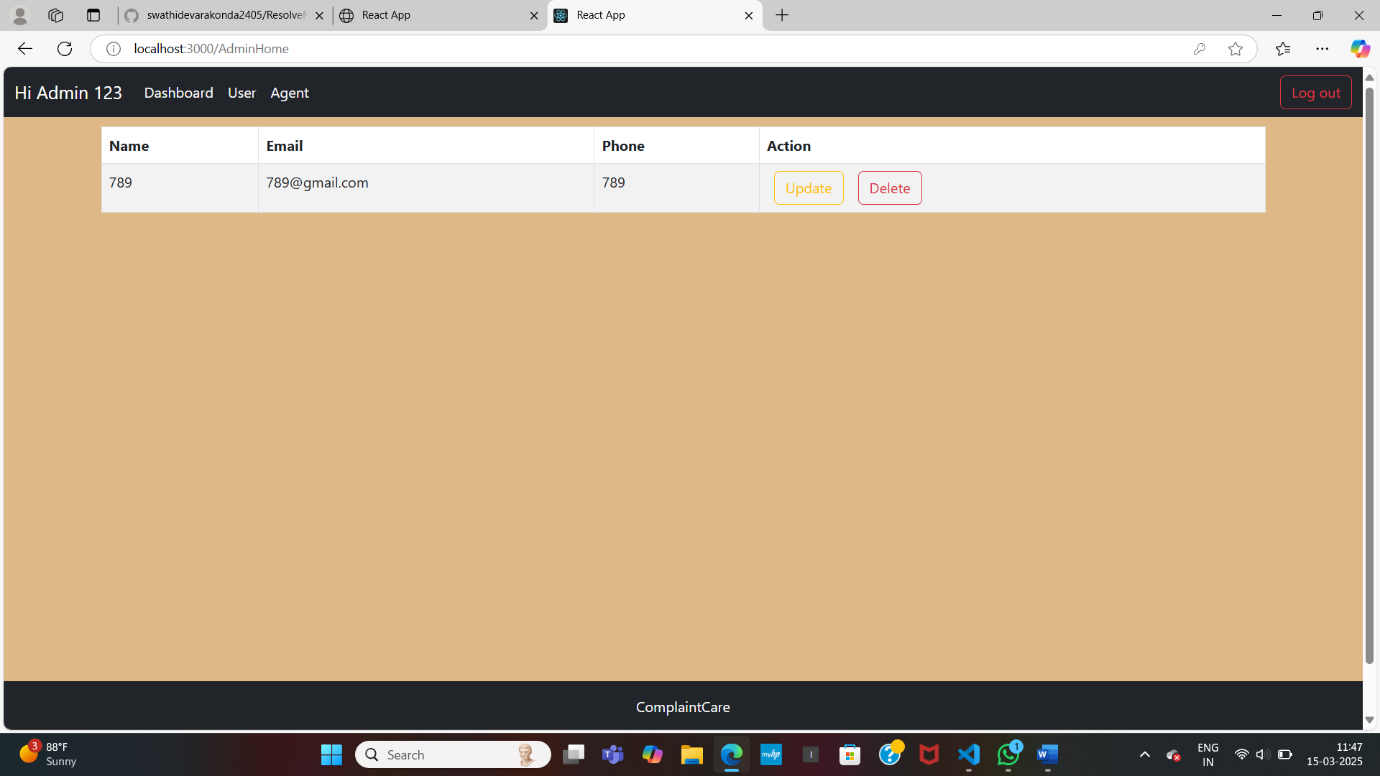
**Admin Home**



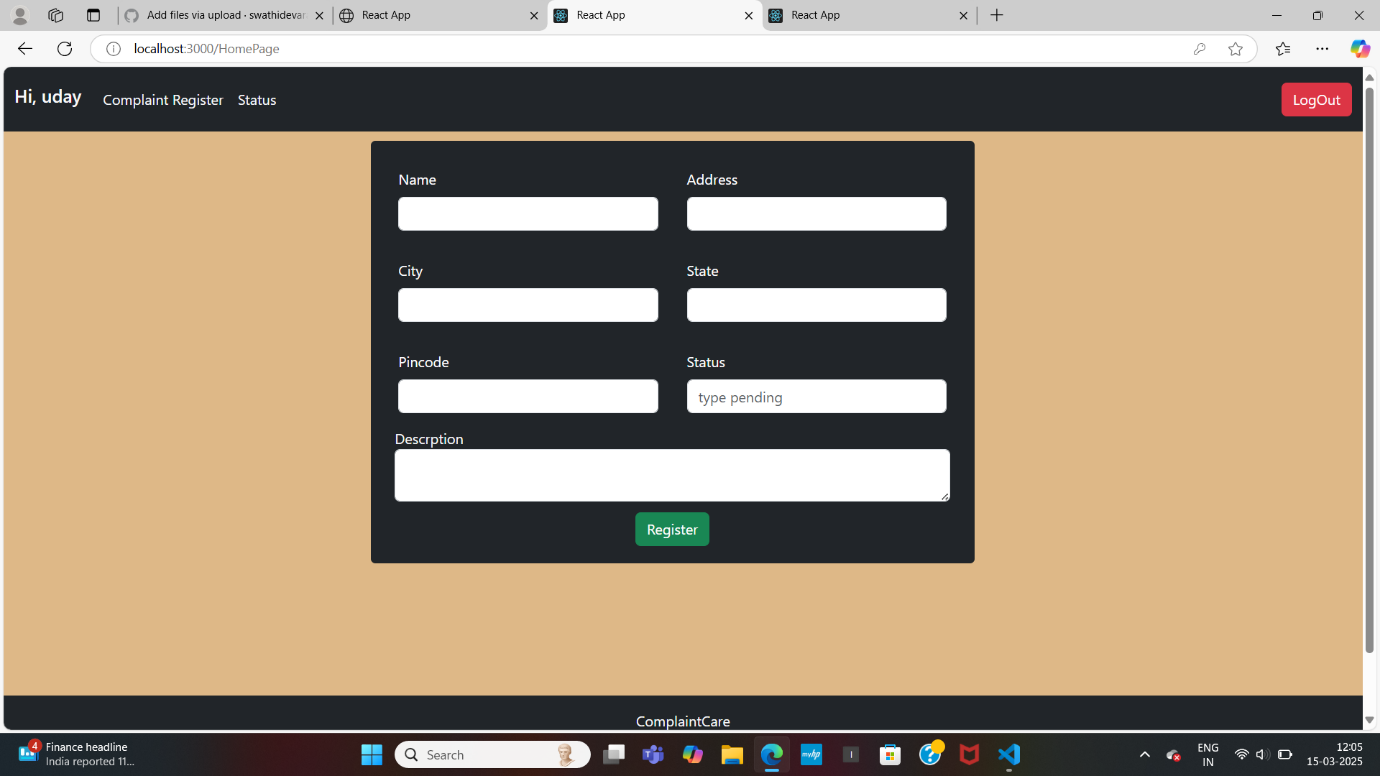
**Admin Users**



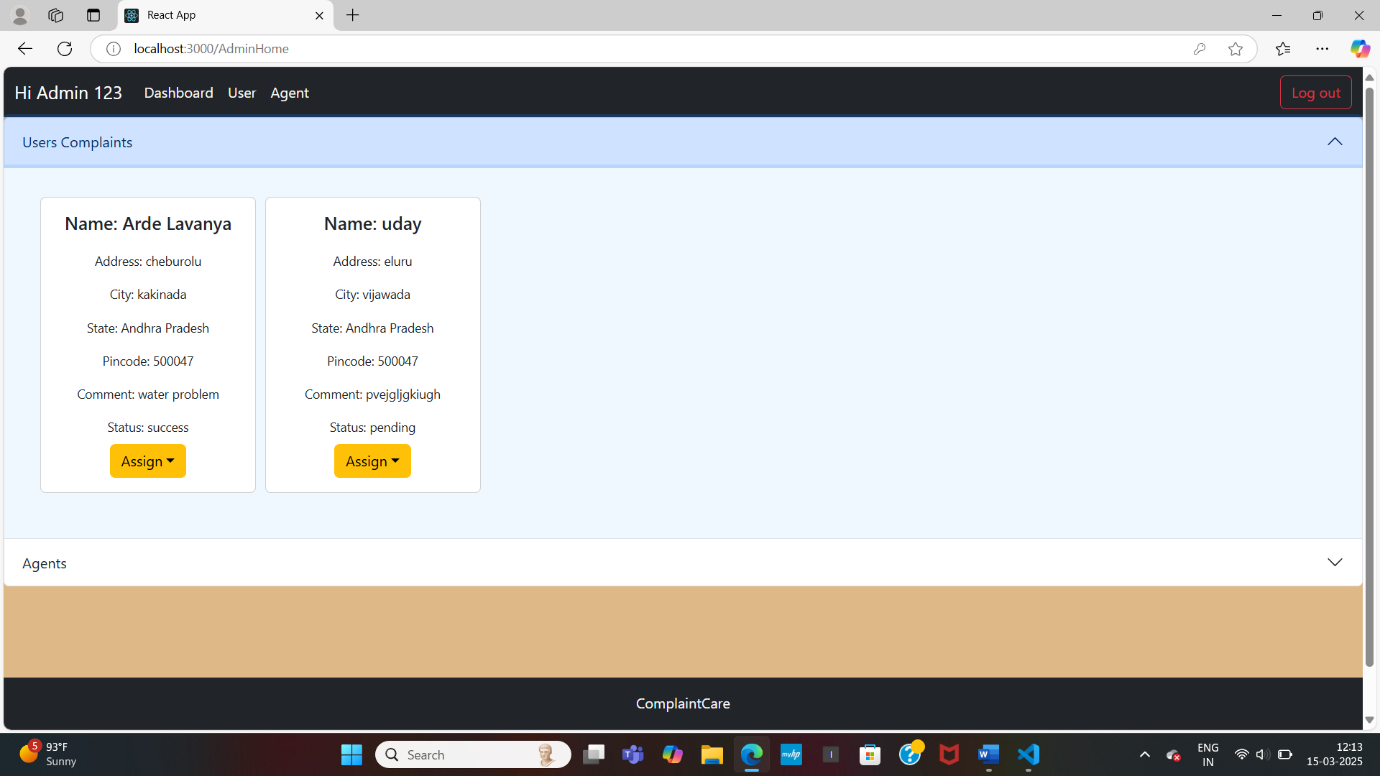
**Admin Agent:**



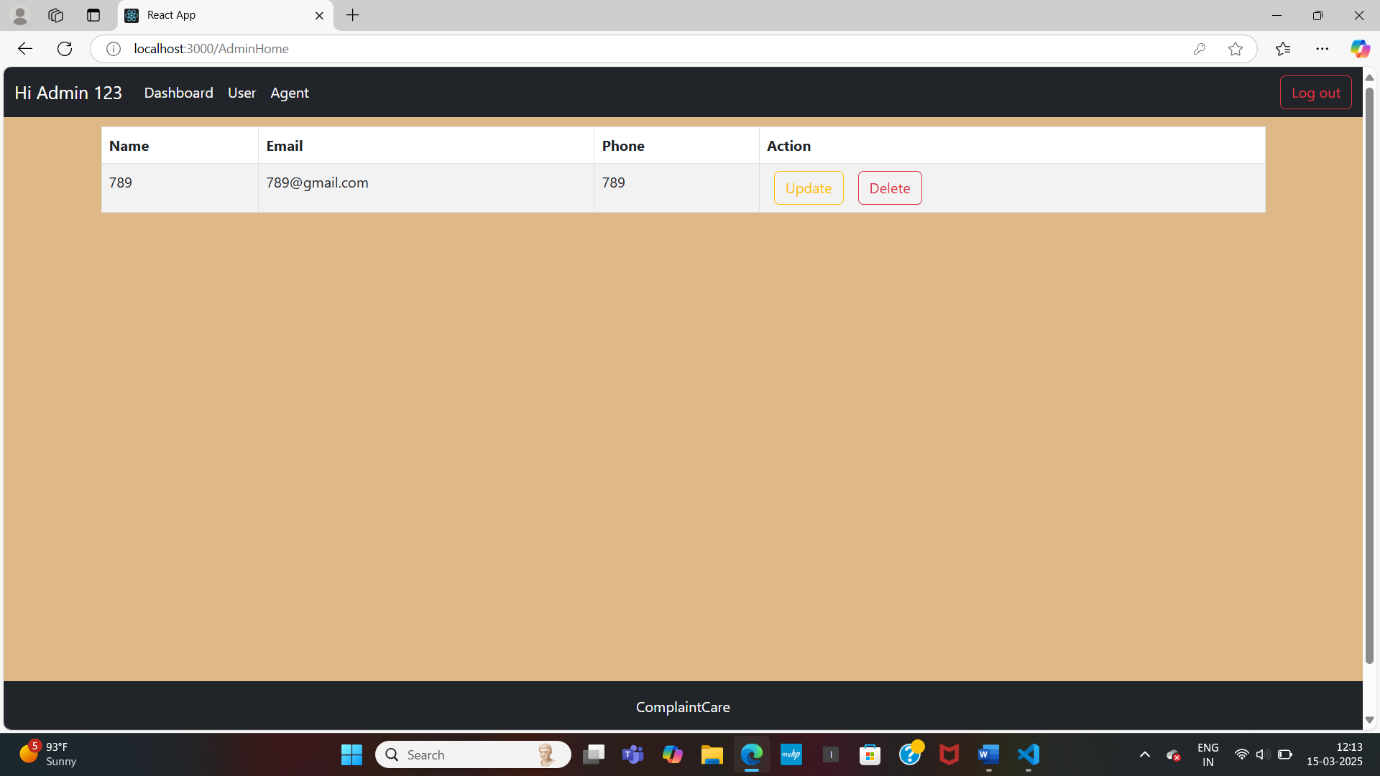
**User Complaints Register:**



**User Status:**

****

**Agent Home:**

****

**Testing:**

**Purpose:** To validate that the application meets user requirements and expectations.

* Users test the entire application to ensure it meets their needs and provides a good user experience.
* Gathering feedback from users and stakeholders to make improvements.

**Known Issues:**

**Database Connection Issues**

* **Problem:** Difficulty connecting to MongoDB, especially in production environments.
* **Solution:** Ensure your MongoDB URI is correct and includes the username, password, and database name. If using MongoDB Atlas, make sure your IP address is whitelisted. Check that Problem your MongoDB server is running if it's a local database.

**Project structure:**

* **Problem:** Poorly structured project architecture leading to maintainability issues.
* **Solution:** Follow a structured project layout with separate folders for models, controllers, routes, and services. Use the MVC (Model-View-Controller) architecture for better code maintainability.

**Future Enhancement:**

. Real-Time Resolution

Live Tracking & Transparency: Develop real-time resolution dashboards where customers can track the status of their complaint at each stage (submitted, under review, resolved) and communicate directly with healthcare providers or administrators.

Feedback Loops: Allow users to rate their experience with the complaint resolution process, helping healthcare organizations track satisfaction and improve customer service.

Here's a full-stack description of an online complaint care project using HTML, CSS, JavaScript, and Node.js, with Visual Studio Code as the IDE, along with future enhancement plans:

1. Integration with Social Media: Integrate the complaint care system with social media platforms to allow customers to submit complaints directly from social media.

2. Mobile App: Develop a mobile app for customers to submit complaints and track their status on-the-go.

3. Predictive Analytics: Use predictive analytics to identify patterns and trends in customer complaints and provide insights to improve the complaint resolution process.

4. Complaint Categorization: Develop a complaint categorization system to automatically categorize complaints based on their type.

**Project Overview:**

The Online Complaint Care Project is a web-based application designed to facilitate the submission, tracking, and resolution of customer complaints.