**ONLINE COMPLAINT REGISTRATION AND MANAGEMENT SYSTEM**

**Project Title: [Online Complaint Registration And Management System]**

Team Members:

Abarajitha - Project Manager: Responsible for overseeing the project development, coordinating with team members, and ensuring project milestones are met. Leads backend development, including setting up the server, creating API endpoints, and integrating with MongoDB.

Vishwa Manikandan - Backend Developer: Leads backend development, including setting up the server, creating API endpoints, and integrating with MongoDB. APIs allow different software systems to communicate with each other, enabling the backend developer to create endpoints that can be accessed by frontend applications or other services.

Madhumitha - Frontend Developer: Handles the development of the React frontend, including creating UI components, implementing user flows, and ensuring responsiveness. Collaborates with the backend team to integrate API endpoints and data fetching.

Mohammed Taufeeq - UI/UX Designer: Focuses on designing an intuitive and user-friendly interface for the application. Responsible for creating wireframes, mockups, and implementing design elements using tools like Tailwind CSS for a polished user experience.

Karthick - Quality Assurance & Tester: Ensures the application meets quality standards by creating and executing test cases for both frontend and backend. Responsible for identifying bugs, documenting issues, and conducting final testing before deployment.

**ABSTRACT**

The Online Complaint Registration and Management System is a web-based platform designed to streamline the process of submitting, tracking, and resolving complaints efficiently. The system provides a centralized interface where users, including customers, employees, or the public, can register their complaints, monitor their progress, and interact with support agents. It aims to improve customer satisfaction by ensuring timely responses and resolutions to issues, while also helping organizations manage and track complaints more effectively.

The system features a user registration module, allowing individuals to create accounts and submit complaints with detailed descriptions, supporting documents, and contact information. Complaints are routed to appropriate agents or departments based on predefined rules, ensuring that issues are handled by the right personnel. Real-time tracking and notifications keep users informed of their complaint status, while a built-in messaging system facilitates direct communication between users and agents.

The platform employs a robust backend architecture using Node.js with Express.js to handle server-side logic and API requests. MongoDB is used for scalable and efficient data storage, and Socket.io enables real-time updates and interactions. The frontend is built with React.js, Bootstrap, and Material UI, ensuring a responsive and user-friendly interface. Security measures, including encryption, authentication, and role-based access control, ensure that user data and complaint information are kept secure and confidential.

This system is designed to enhance operational efficiency, improve transparency in the complaint resolution process, and provide a positive user experience by ensuring that complaints are addressed promptly and effectively.

**TABLE OF CONTENTS**

* Introduction.
* Scenario.
* ER Diagram.
* Pre – Requisites.
* Project structure.
* Project Flow.
* Project Setup And Configuration.
* Backend Development.
* Database Development.
* Frontend Development.
* Project Implementation.
* Conclusion

**INTRODUCTION**

In today’s competitive business landscape, the ability to effectively handle customer complaints and feedback is critical to maintaining a positive reputation and building lasting customer relationships. Organizations that prioritize efficient complaint management not only resolve issues quickly but also demonstrate a commitment to customer satisfaction and service excellence. However, traditional complaint management methods—often based on manual processes—are prone to errors, delays, and inefficiencies, which can result in customer frustration and loss of business.

The **Online Complaint Registration and Management System** is designed to address these challenges by providing a modern, automated solution to streamline the entire complaint management process. This system offers businesses a comprehensive platform that enables users to submit complaints, track their status, and communicate with support agents in real-time. For businesses, it serves as a powerful tool to centralize complaint data, automate complaint routing, and facilitate efficient communication between agents and customers. Ultimately, the goal of the system is to improve operational efficiency, enhance customer service, and foster stronger customer relationships.

The system is structured around several core features that work together to automate and improve complaint resolution. Users can easily register complaints by providing detailed descriptions, uploading relevant documents or images, and sharing their contact details. Once a complaint is submitted, the system assigns it to the appropriate agent or department based on the nature of the issue. Agents can then manage and resolve the complaints while interacting with customers via built-in messaging or communication tools, ensuring that the complaint is resolved to the customer’s satisfaction.

A major advantage of this system is its ability to provide real-time complaint tracking. Once a complaint is submitted, users can monitor its progress from submission to resolution. Automated notifications and status updates keep customers informed, reducing the need for follow-up calls or emails. Additionally, the platform ensures transparency by allowing users to track every stage of their complaint’s lifecycle, thus enhancing trust in the system and the organization. This transparency, combined with prompt communication, contributes to a positive customer experience, even in situations where complaints arise.

One of the most important aspects of the **Online Complaint Registration and Management System** is its ability to optimize the complaint routing process. Complaints are automatically categorized and routed to the appropriate agent or department based on predefined rules such as issue type, urgency, or product category. This intelligent routing system ensures that complaints are handled by the most qualified agent, which improves the speed and quality of issue resolution. Furthermore, by automating the routing process, the system helps prevent complaints from being lost or delayed in the system, ensuring that they are promptly addressed.

For customer service teams, the platform offers tools that simplify complaint management, enhance communication with customers, and streamline issue resolution. Agents are provided with all relevant details about the complaint, such as the customer’s information, the complaint description, any attachments (e.g., images), and the complaint’s current status. The platform also includes messaging features that allow agents to communicate directly with users, ensuring that responses are timely and that any additional information needed to resolve the issue is quickly gathered.

The **Online Complaint Registration and Management System** is powered by modern web technologies designed for scalability, flexibility, and security. The system’s backend is built using **Node.js** and **Express.js**, which provide a fast, lightweight, and scalable framework for handling the server-side operations and business logic. **MongoDB** serves as the database, allowing for flexible, high-performance storage and retrieval of data such as user profiles, complaint details, and complaint histories. This NoSQL database allows the system to scale efficiently and handle large amounts of data, making it suitable for both small businesses and large enterprises.

The frontend is designed using **React.js**, a popular JavaScript library for building user interfaces, ensuring a smooth and interactive experience for users. To make the platform accessible on a variety of devices, it uses **Bootstrap** and **Material UI**, which enable responsive design and a clean, intuitive layout. With this user-friendly interface, customers, agents, and administrators can easily navigate the system, whether they are submitting a complaint, managing cases, or generating reports.

For real-time communication, the system integrates **Socket.io**, which allows agents and users to interact live within the platform. This capability ensures that users receive immediate assistance or updates when necessary, reducing the time taken to resolve complaints. The inclusion of **WebRTC** can also enable video or voice calls, allowing for more personalized support in cases where a more detailed explanation or face-to-face communication is required.

In addition to handling complaints, the system provides robust security features to protect user data and ensure compliance with data privacy regulations. Authentication mechanisms such as **OAuth** or **JSON Web Tokens (JWT)** are implemented to verify user identities, while **SSL encryption** ensures that data is transmitted securely. Role-based access control (RBAC) ensures that only authorized users (such as administrators or agents) can access sensitive complaint data. This level of security helps prevent unauthorized access to the system and ensures that user data remains confidential.

The system also provides powerful reporting and analytics features. Administrators can access real-time dashboards that provide insights into key performance indicators (KPIs), such as complaint resolution time, agent productivity, customer satisfaction, and complaint volume trends. These reports can be used to monitor system performance, identify areas for improvement, and make data-driven decisions that enhance the overall complaint management process. By tracking common complaint trends and agent performance metrics, businesses can identify recurring issues, streamline their operations, and implement corrective actions to improve customer satisfaction.

Moreover, the system is highly scalable and customizable, allowing organizations to adjust it according to their specific needs. Whether for a small business with a handful of complaints or a large corporation with thousands of customer interactions, the platform can handle high volumes of data and users without compromising on performance. As businesses grow and customer demands increase, the system’s architecture ensures that it can scale seamlessly, making it an ideal solution for organizations of all sizes.

The **Online Complaint Registration and Management System** not only serves as a tool for resolving customer complaints but also acts as a valuable resource for businesses to improve their products, services, and overall customer experience. By analyzing the data generated from complaints, businesses can identify patterns, make improvements, and ultimately prevent recurring issues. This proactive approach to customer feedback fosters trust and builds long-term customer loyalty.

In summary, the **Online Complaint Registration and Management System** is an innovative and efficient platform that automates and streamlines the complaint handling process, making it easier for users to report issues and track their resolution, while also enabling businesses to enhance their customer service operations. By centralizing complaint management, automating routing, and providing real-time communication features, the system ensures that complaints are resolved quickly and effectively, leading to improved customer satisfaction and stronger business outcomes. Through its combination of modern technologies, security measures, and scalability, this system represents the future of complaint management for businesses seeking to improve both operational efficiency and customer satisfaction.

**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.
2. **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.
3. **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.
4. **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.
5. **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.
6. **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 DAIGRAM**

The ER diagram for the Online Complaint Registration and Management System visually represents the structure and interconnections between different entities within the database. The key entities involved include **User**, **Complaint**, **Agent**, **Message**, and **Admin**. The diagram helps outline the relationships and data flow, ensuring that the system’s database is effectively designed for operational efficiency.

**1. User Entity:**

* **Attributes:** UserID, Name, Email, Password, ContactDetails
* **Relationships:** Each **User** can raise multiple complaints, represented by a one-to-many relationship with the **Complaint** entity.
* **Description:** Users are individuals or organizations that register in the system to submit and track complaints.

**2. Complaint Entity:**

* **Attributes:** ComplaintID, UserID, Description, Status, DateSubmitted
* **Relationships:** Linked to the **User** entity through UserID and associated with the **Agent** entity, as each complaint is assigned to an agent for resolution.
* **Description:** The **Complaint** entity holds the details of each registered complaint, including a unique identifier and the current status.

**3. Agent Entity:**

* **Attributes:** AgentID, Name, Specialization, ContactInfo
* **Relationships:** An agent can be assigned to handle multiple complaints, forming a one-to-many relationship with the **Complaint** entity.
* **Description:** The **Agent** entity represents customer service personnel responsible for managing and resolving complaints.

**4. Message Entity:**

* **Attributes:** MessageID, ComplaintID, UserID, AgentID, Content, Timestamp
* **Relationships:** The **Message** entity references both the **User** and **Agent** entities, establishing communication between them. It uses ComplaintID to ensure the messages are associated with the relevant complaint.
* **Description:** This entity supports the chat feature, allowing users and agents to interact within the system.

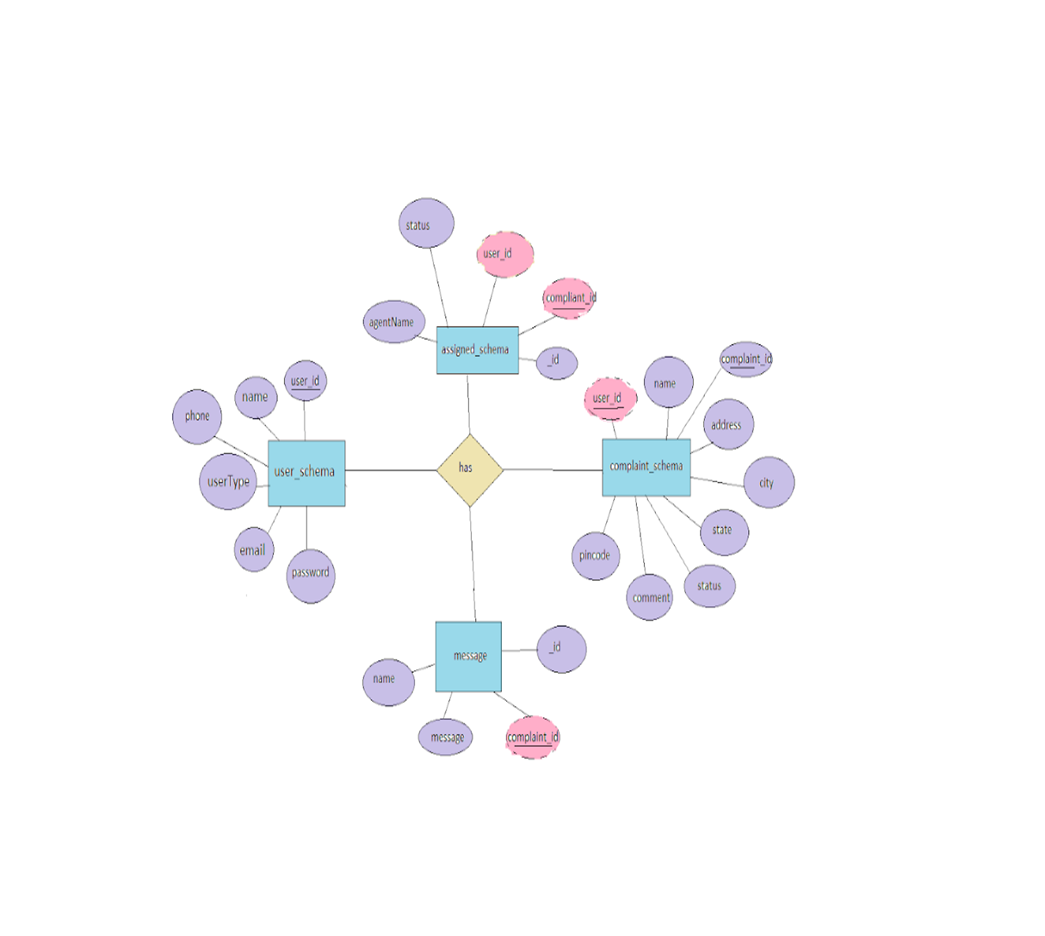
**5. Admin Entity:**

* **Attributes:** AdminID, Name, Role, ContactInfo
* **Relationships:** The **Admin** monitors the overall complaint system, assigning complaints and overseeing agent workload.
* **Description:** The **Admin** entity ensures proper system management and compliance.

**Relationship Overview:**

* **User-Complaint Relationship:** One **User** can submit multiple **Complaints** (1).
* **Complaint-Agent Relationship:** One **Complaint** is assigned to one **Agent**, but one **Agent** can handle multiple **Complaints** (1).
* **Complaint-Message Relationship:** Each **Complaint** can have multiple **Messages** associated with it (1), facilitating user-agent interaction.
* **User-Message Relationship:** Each **Message** has a UserID and AgentID, linking it back to the **User** and **Agent** involved in the conversation.

This ER diagram framework ensures data consistency, supports real-time tracking and notifications, and maintains the integrity of interactions between users, agents, and admins.



**PRE-REQUISITES**

*Key prerequisites for developing a full-stack application using Node.js, Express.js, MongoDB, and React.js:*

**1. Node.js and npm:**

**Node.js** is a powerful JavaScript runtime environment that runs JavaScript code on the server-side, providing a scalable platform for network applications.

* **Installation**: Ensure Node.js and npm are installed on your development machine.
* **Download**: [Node.js Download](https://nodejs.org/en/download/)
* **Installation Guide**: [Node.js Package Manager Installation](https://nodejs.org/en/download/package-manager/)

**2. Express.js:**

**Express.js** is a minimalist web application framework for Node.js, simplifying API creation, routing, and middleware.

* **Installation Command**:

npm install express

**3. MongoDB:**

**MongoDB** is a scalable NoSQL database storing data in a JSON-like format, ideal for handling structured and unstructured data.

* **Setup**: Ensure MongoDB is installed and configured.
* **Download**: [MongoDB Community Edition](https://www.mongodb.com/try/download/community)
* **Installation Guide**: [MongoDB Manual Installation](https://docs.mongodb.com/manual/installation/)

**4. React.js:**

**React.js** is a popular library for building dynamic user interfaces with reusable components.

* **Installation Guide**: [Create React App](https://reactjs.org/docs/create-a-new-react-app.html)

**5. HTML, CSS, and JavaScript:**

Basic understanding of **HTML** for structure, **CSS** for styling, and **JavaScript** for interactivity is essential.

**6. Database Connectivity:**

Connect your Node.js server to MongoDB using a MongoDB driver or an ODM like **Mongoose**.

* **Guide**: Node.js and Mongoose Connection

**7. Front-End Frameworks:**

Use **React.js** for building user interfaces like complaint entry forms, status tracking, and admin dashboards.

* Utilize libraries such as **Material UI** and **Bootstrap** for enhanced UI.

**8. Version Control:**

Use **Git** for version control to collaborate and track changes.

* **Download Git**: [Git Installation](https://git-scm.com/downloads)

**9. Development Environment:**

Choose an IDE like **Visual Studio Code**, **Sublime Text**, or **WebStorm**.

* **VS Code Download**: [Download VS Code](https://code.visualstudio.com/download)

**Running the Existing Project**

To run the existing Video Conference App project from GitHub:

1. **Clone the Repository**:

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

1. **Navigate to the repository**:

cd complaint-registery

1. **Install Dependencies**:

cd frontend

npm install

cd ../backend

npm install

1. **Start the Development Server**:

npm start

The application will be available at <http://localhost:3000>.

You have now set up the online complaint registration and management app on your local machine, ready for development and testing.

**PROJECT STRUCTURE**

The Online Complaint Registration and Management System is designed with a clear and logical project structure to ensure seamless development and ease of maintenance. Below is the detailed project structure encompassing both the frontend and backend components.

**Project Structure**

**1. Root Directory**

* **/complaint-management-system**
  + Contains the main project folder with subdirectories for both frontend and backend development.

**2. Frontend Directory (/frontend)**

The frontend is developed using React.js, Material-UI, and Bootstrap to create an interactive and user-friendly interface.

* **/public**
  + index.html: The main HTML file used to render the React app.
  + Assets like images and icons.
* **/src**
  + **/components**
    - Contains reusable React components for UI elements (e.g., Header.js, Footer.js, ComplaintForm.js).
  + **/pages**
    - Contains main pages for routing (e.g., HomePage.js, Dashboard.js, ProfilePage.js).
  + **/services**
    - Contains service files for handling API calls (e.g., apiService.js).
  + **/context**
    - Includes context API files for managing global state (e.g., UserContext.js).
  + **App.js**
    - The main component that holds routing logic.
  + **index.js**
    - The entry point of the React application.
  + **styles.css**
    - Custom styling for the application.

**3. Backend Directory (/backend)**

The backend is built with Node.js and Express.js, facilitating RESTful API development. MongoDB is used for database management.

* **/config**
  + Contains configuration files such as dbConfig.js for database connectivity.
* **/controllers**
  + Contains controller files that handle business logic (e.g., complaintController.js, userController.js).
* **/models**
  + Contains Mongoose schemas and models (e.g., Complaint.js, User.js).
* **/middlewares**
  + Middleware files for tasks like authentication (authMiddleware.js).
* **/routes**
  + Contains Express route files for different modules (e.g., complaintRoutes.js, userRoutes.js).
* **/utils**
  + Utility functions (e.g., errorHandler.js, emailService.js).
* **server.js**
  + The main server file that sets up the Express app and routes.

**Database**

* **MongoDB**
  + Collections:
    - **Users**: Stores user details such as name, email, and password.
    - **Complaints**: Stores complaints with fields like description, attachments, and status.
    - **Agents**: Stores agent information responsible for handling complaints.

**PROJECT FLOW**

The Project Flow for the Online Complaint Registration and Management System follows a well-defined sequence of steps, ensuring a smooth and organized approach to complaint handling:

**Application Flow**

**1. Customer/Ordinary User:**

* **Registration and Login**: Register and log in to the platform.
* **Complaint Submission**: Submit complaints and attach relevant documents.
* **Status Tracking**: Monitor the complaint status and receive updates.
* **Interaction with Agents**: Use the chat feature for follow-ups.

**2. Agent:**

* **Registration and Login**: Access and log in to the platform.
* **Complaint Management**: View assigned complaints and communicate with users.
* **Status Updates**: Update the progress of each complaint.

**3. Admin:**

* **Management and Monitoring**: Oversee the entire system.
* **Complaint Assignment**: Assign complaints to agents.
* **User and Agent Management**: Manage user profiles and ensure policy adherence.

**PROJECT SETUP AND CONFIGURATION**

**1. Create Project Folders and Files:**

To start building the **Online Complaint Registration and Management System**, you first need to create the necessary folders for both the frontend and backend. This will help organize your code and ensure that all components are structured properly.

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

After the installation of all the libraries, the package.json files for the backend

**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.

**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

**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.

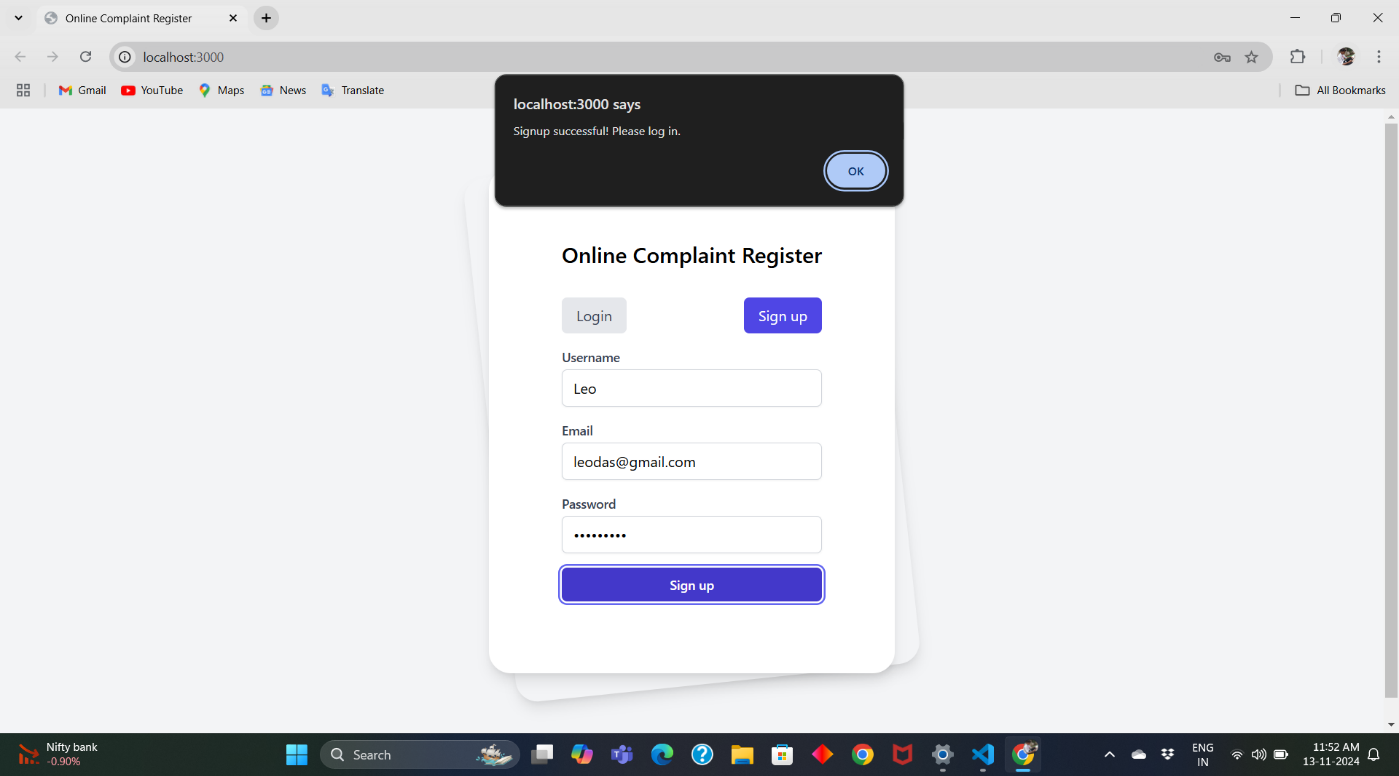
1. **Implement frontend logic:**

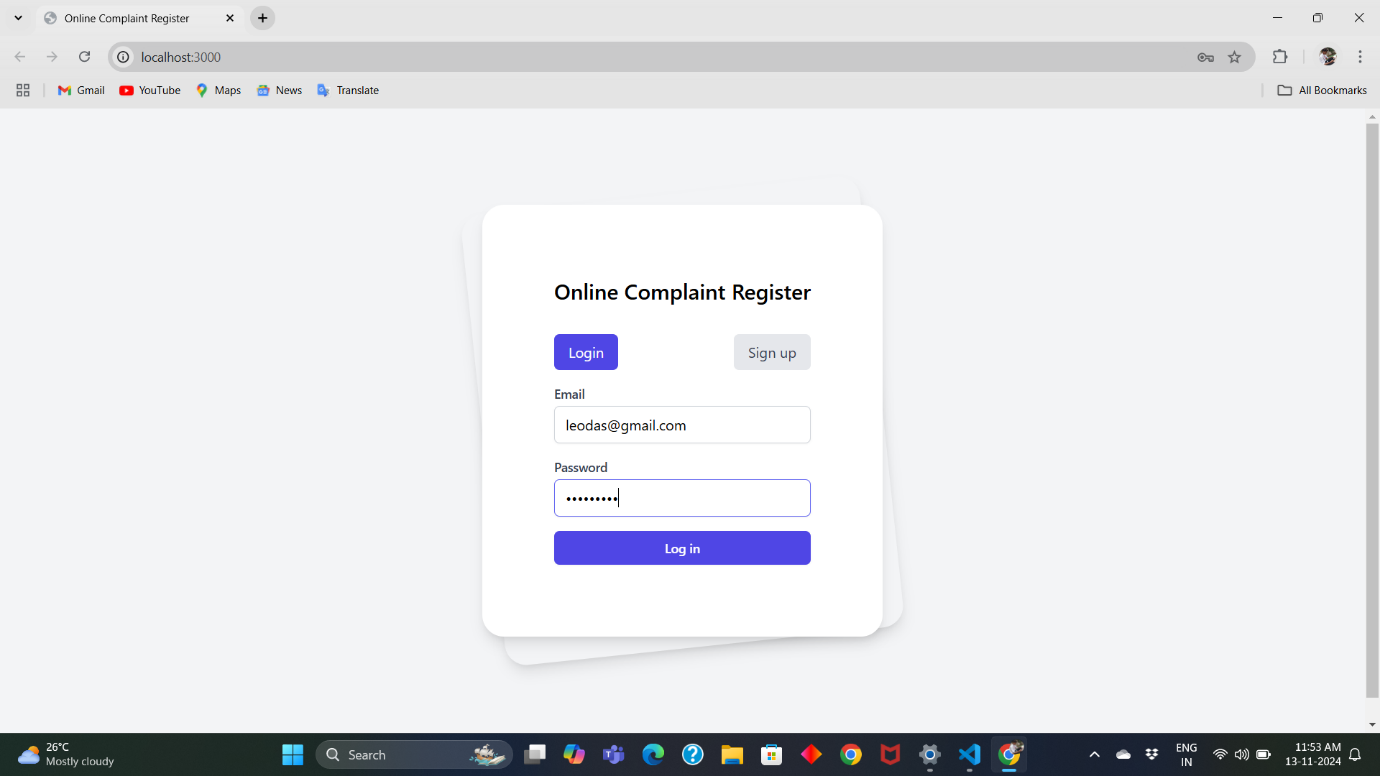
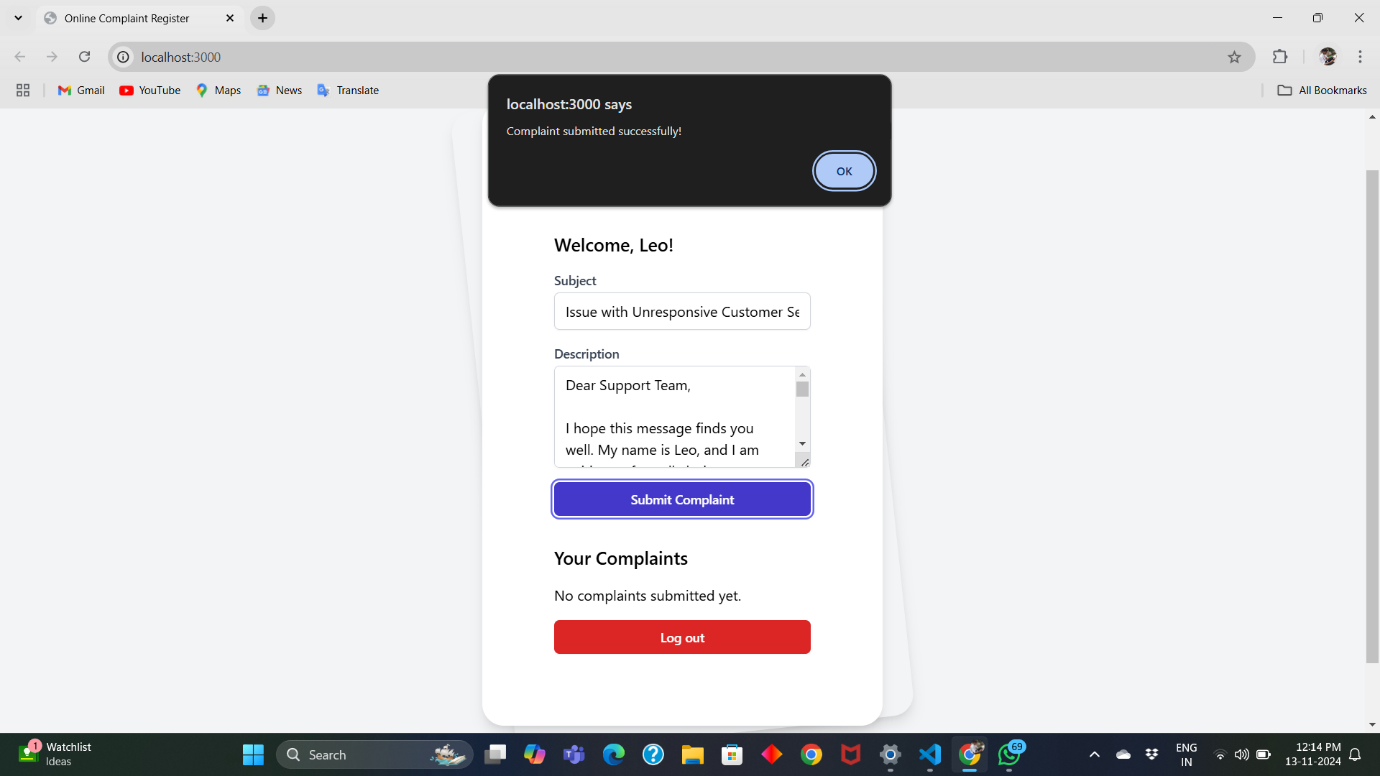
In the final leg of the frontend development, we'll bridge the gap between the visual interface and the underlying data. It involves the below stages.

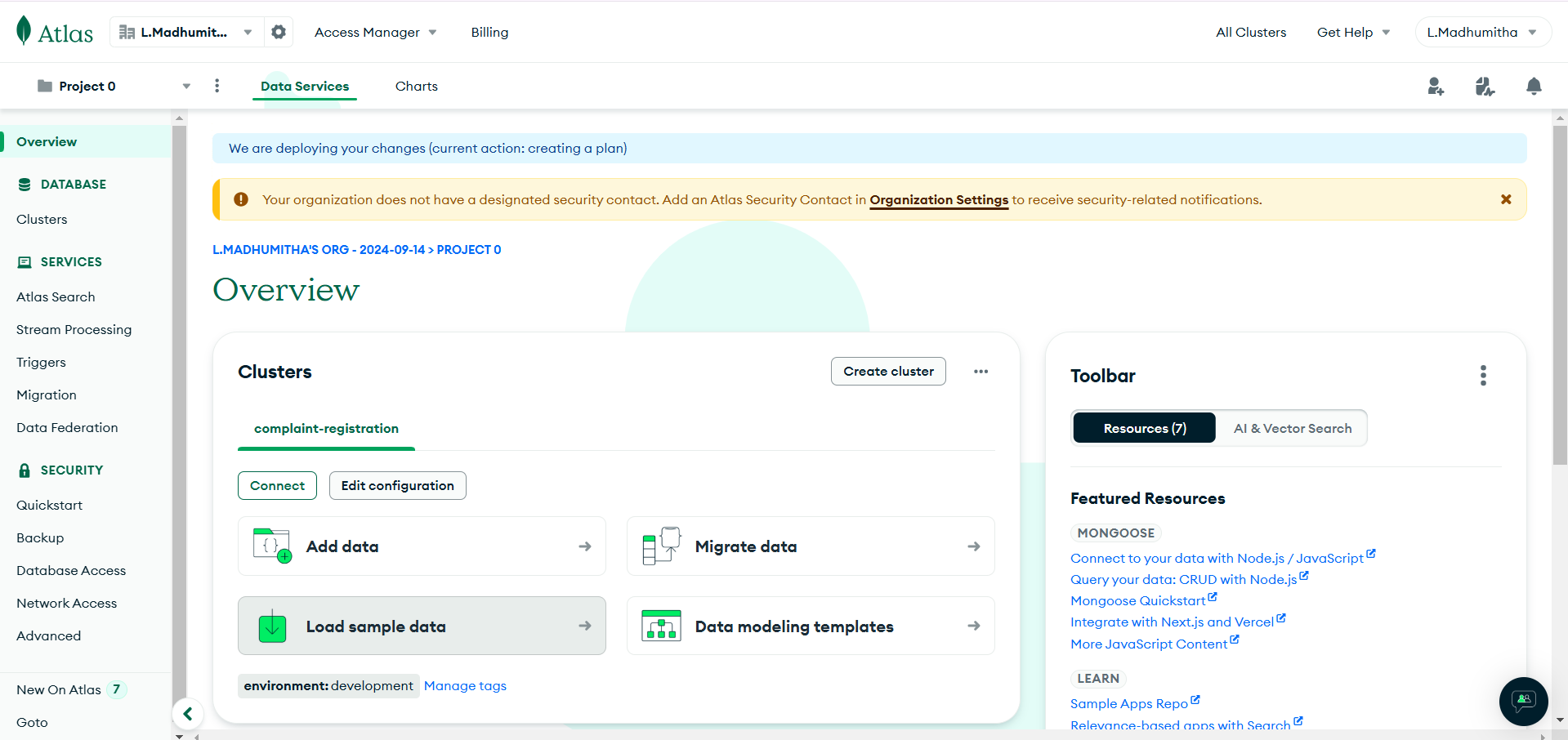
* Integration with API endpoints.
* Implement data binding.

**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.





**CONCLUSION**

In this project, we successfully developed an **Online Complaint Registration System** that streamlines the process of submitting and managing complaints, making it more accessible and efficient for users. The system was built using a combination of **HTML**, **CSS**, **ReactJS**, **JavaScript**, **NodeJS**, **MongoDB**, and **Bootstrap**, allowing us to create a responsive and user-friendly platform with robust functionality.

The **frontend** was developed using **ReactJS**, enabling dynamic and interactive interfaces for users to submit complaints, track their status, and receive notifications. The design was made responsive using **Bootstrap**, ensuring compatibility across different screen sizes and devices. **HTML** and **CSS** were used to structure and style the pages, providing a clean and intuitive user experience.

For the **backend**, we leveraged **NodeJS** with **Express.js** to handle API requests, ensuring smooth communication between the client and the server. **MongoDB** was chosen as the database due to its flexibility and scalability, enabling efficient storage and retrieval of complaint data.

As a team, we worked collaboratively, dividing tasks based on individual strengths, and integrating each part of the project to ensure a seamless end-to-end solution. The use of modern technologies and frameworks not only improved the performance of the system but also enhanced the user experience by providing fast and real-time feedback.

This project has not only improved our technical skills but also strengthened our ability to work in a team, solve complex problems, and meet deadlines. The system is now ready for deployment, providing a solid foundation for further improvements and scalability.In conclusion, the Online Complaint Registration System offers an effective, modern solution for managing complaints, and with the technologies used, we have ensured that it is both scalable and maintainable for future enhancements.