Project Documentation

1. Introduction

Project Title: Online Complaint Registration and Management System

Team Members:

- Allwin S – 211521243009
- Aravinth S – 211521243018
- Ashwin RS – 211521243024
- Balamurugan K - 211521243030

2. Project Overview

Purpose:

This project aims to provide a seamless and efficient platform for users to register complaints, track their progress, and interact with assigned agents for issue resolution. The goal is to enhance user experience and operational efficiency through a centralized system for complaint management.

Features:

- User registration and secure login
- JWT-based authentication and session handling
- Real-time status updates
- Admin and agent dashboards for efficient complaint management
- Role-based access control (User, Agent, Admin)

3. Architecture

Frontend:

The frontend is structured using React. It offers a responsive user interface for different user roles. The UI components are styled with Bootstrap for a consistent look and feel.

Technologies Used:

- React
- Bootstrap
- Axios for API requests
- React Router for navigation

Backend:

The backend is built using Node.js and Express.js. It handles API requests, user authentication, session management, and complaint processing.

Technologies Used:

- Node.js
- Express.js
- JWT for authentication
- Mongoose for database operations

Database:

MongoDB is used as the database. Mongoose ODM is used to define schemas and manage database operations. The primary collections include:

- **User Schema:** Stores user details (username, password, role).
- **Complaint Schema:** Manages complaint details, including status, assigned agent, and user references.

4. Setup Instructions

Prerequisites:

Software Requirements:

- **Node.js:** For running the backend and managing dependencies.
- **MongoDB:** NoSQL database for storing data.
- **Git:** Version control for managing code changes.

Libraries/Dependencies:

- **Express:** Backend framework for setting up routes and handling requests.
- **Mongoose:** For MongoDB data modeling.
- **React:** Frontend library for building the user interface.
- **Axios/Fetch:** For making HTTP requests from React to the backend.
- **CORS:** Middleware for handling cross-origin requests.

Installation:

i. **Clone the repository:**

"git clone https://github.com/Bala-273/Complaint-Registration-System.git"

ii. **Navigate to the backend directory and install the dependencies:**

"cd backend -> npm install"

iii. **Navigate to the frontend directory and install the dependencies:**

"cd frontend -> npm install"

iv. **Configure environment variables in a `.env` file:**

"MONGODB_URI=mongodb+srv://<db_username><db_password>:@cluster0.ohtes.mongodb.net"

v. **Start the server:**

"npm start"

5. Folder Structure

i. **Root Folder (Complaint-Registry/):** Contains the main project structure, including separate folders for backend and frontend code.

ii. **Backend Folder (backend/):**

- **schema.js:** Defines Mongoose schemas for MongoDB, including user details and complaint information.
- **config.js:** Contains configuration details, such as MongoDB connection settings.
- **index.js:** The main entry point for the backend server, initializing Express, connecting to MongoDB, and defining routes.
- **package.json:** Lists the backend dependencies such as Express, Mongoose, CORS, etc.
- **package-lock.json:** Locks specific versions of dependencies to ensure consistent installation across environments.

iii. **Frontend Folder (frontend/):**

- **public/index.html:** The main HTML file served by the React application, providing the root element for rendering the React app.
- **src/App.css:** Contains styles for the React components.
- **src/App.js:** Main React component that includes the application logic and UI structure.
- **src/index.js:** Entry point for the React app, rendering the main `App` component into the root element in `index.html`.
- **package.json:** Lists frontend dependencies such as React, React Router, Axios, etc.

- **package-lock.json:** Ensures consistent installation of frontend dependencies by locking specific versions.

6. Running the Application

Clone the Repository:

“git clone https://github.com/Bala-273/Complaint-Registration-System.git”

Navigate to the Project Directory:

“cd Complaint-Registry”

Running the Backend

Install Backend Dependencies and Start the Backend Server:

“cd../backend -> npm install”

“npm start”

The backend server will run on ‘http://localhost:3000’.

Running the Frontend

Install Frontend Dependencies and Start the Frontend Server:

“cd ../frontend -> npm install”

“npm start”

The frontend server will run on ‘http://localhost:3001’ (or another available port).

Access the Application:

Open your web browser and go to the port where the server is running to access the frontend interface.

7. API Documentation

- User Endpoints (‘/signup`, `/login’): For registering and authenticating users.
- Homepage Endpoint (‘/homepage’): Provides user-specific content based on the role.
- Admin Endpoint (‘/admin/home’): Provides an overview of complaint statistics for the admin dashboard.

- Agent Endpoint (‘/agent/home’): Shows the list of complaints assigned to the agent.

8. Authentication

The application uses JWT (JSON Web Token) for secure authentication:

- **Login:** On successful login, a JWT token is generated and stored in the user's session or cookies.
- **Authorization Middleware:** Protects routes to ensure only authenticated users can access them, based on their roles (User, Agent, Admin).

9. User Interface

- **Login and Signup Pages:** For user registration and login.
- **Dashboard:** Separate dashboards for users, agents, and admins.
- **Complaint Submission Form:** Allows users to file complaints
- **Admin Panel:** Admins can view and manage all complaints, assign agents, and update statuses.

10. Testing

Testing Strategy:

- **Unit Testing:** Focus on testing individual components, functions, and modules to ensure they work as expected.
- **Integration Testing:** Verify the interaction between different modules like backend API with the MongoDB database, and React components with the backend API.
- **End-to-End (E2E) Testing:** Simulate real user scenarios to test the complete application flow from signup/login to complaint submission and resolution.

Tools Used:

- Jest: For unit testing React components and Node.js functions.
- React Testing Library: To test React components, user interactions, and UI updates.
- Postman: For testing API endpoints manually and ensuring they respond as expected.

Test Cases Examples:

User Registration:

- Test: Check if a new user can register successfully.
- Expected Result: The user receives a confirmation response and is stored in the database.

Login Functionality:

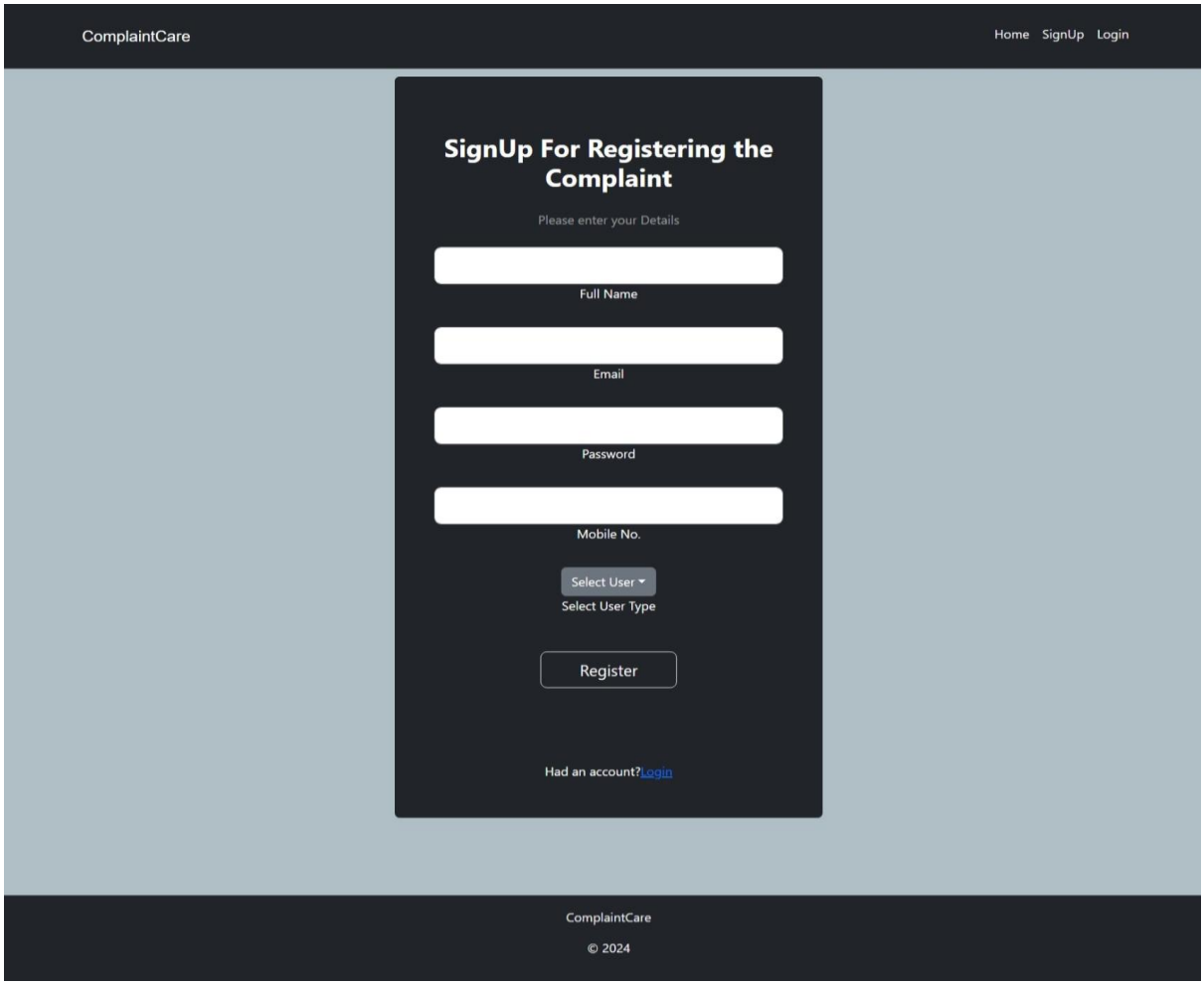
- Test: Validate that only registered users can log in with the correct credentials.
- Expected Result: The user receives a valid JWT token upon successful login.

Complaint Submission:

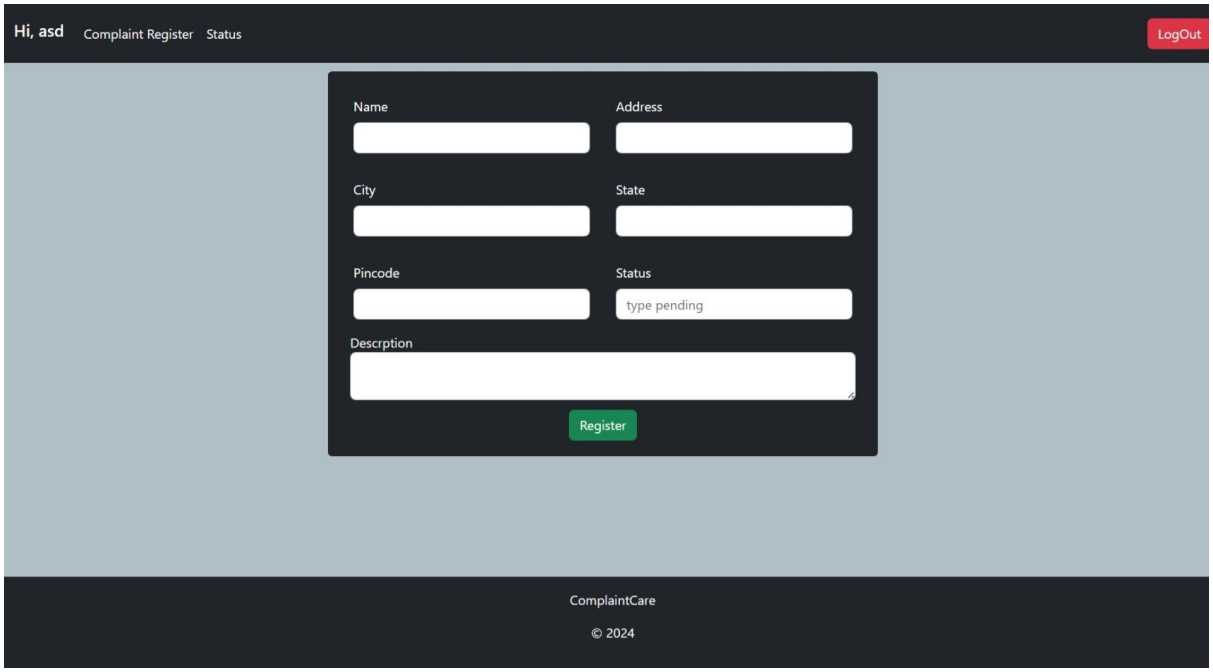
- Test: Verify that a user can submit a complaint with a valid form.
- Expected Result: The complaint is saved to the database, and the user sees a confirmation message.

11. Screenshots

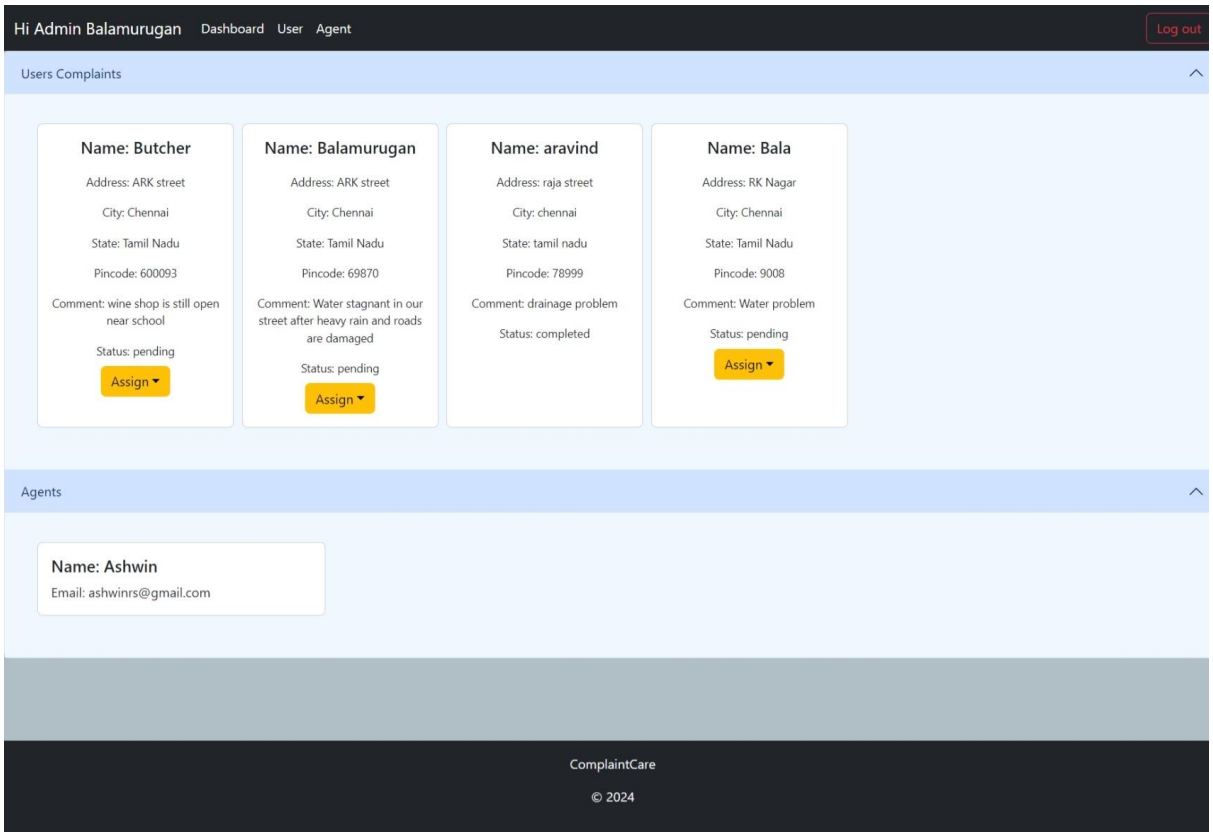
User registration page: Select user type and register the website



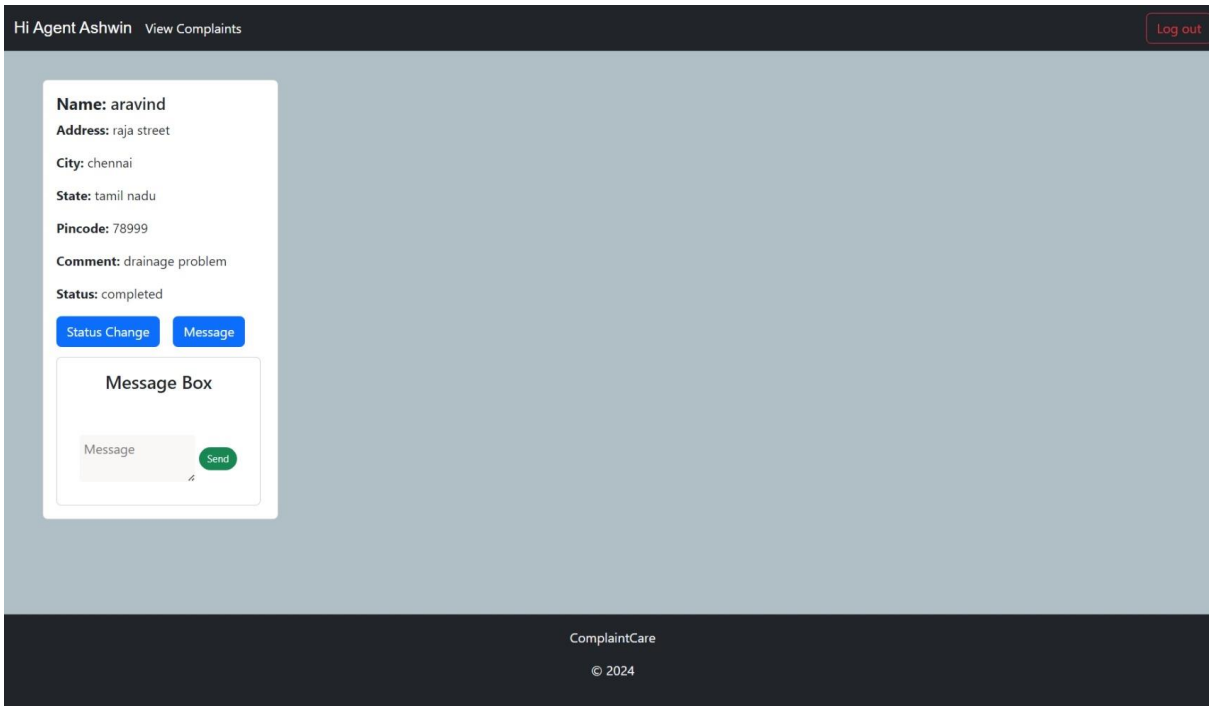
Complaint registration form: Users can register their complaint by providing the details



Admin dashboard: Admin can view the status of the user`s complaints and assign agents for the complaints



Agent dashboard: Agents can view their assigned complaints and communicate with the users and also update the users when the problem is resolved



12. Known Issues

- Currently, the application cannot upload files.
- When handling a large number of simultaneous complaints, there might be performance issues.

13. Future Enhancements

- **Mobile Application Development:** Develop a mobile version of the application using React Native or Flutter to make it accessible on smartphones and tablets, increasing user engagement and accessibility.
- **AI-Powered Complaint Categorization:** Implement Natural Language Processing (NLP) to automatically categorize complaints based on their content. This feature could prioritize high-severity issues and route them to the appropriate department quickly.
- **Chatbot Integration:** Add a chatbot feature using to assist users in registering complaints, providing updates, and answering common queries, enhancing the user experience and reducing response time.
- **Advanced Analytics Dashboard:** Create a comprehensive analytics dashboard for administrators to visualize trends in complaint data, monitor agent performance, and identify common issues.