ResolveNow: Your Platform For Online Complaints

1. Introduction
Project Title: ResolveNow: Your Platform For Online Complaints
Team Members:

1. Kode Devi Sri Vallika(lead)

 2.Tupakula Yoshika Yadav

 3.Shakina Dasari

 4.Lukka Sanjay

2. Project Overview
Purpose: To provide a centralized, transparent digital platform for users to register
service complaints and track their resolution in real-time, eliminating the need for manual
paperwork and physical office visits.
Features: Role-based dashboards (Customer, Admin, Agent), dynamic complaint
lifecycle tracking (Pending -> Assigned -> Resolved), responsive "Glassmorphism" UI,
and secure user authentication.

3. Architecture
Frontend: Built using React.js with React Router for navigation. The UI utilizes
Bootstrap and custom CSS for a modern, tech-themed aesthetic.
Backend: Powered by Node.js and Express.js, handling RESTful API requests, business
logic for task routing, and data processing.
Database: MongoDB is used to store unstructured document data, including user profiles,
role assignments, and detailed complaint logs.

4. Setup Instructions
Prerequisites: Node.js installed on the system, MongoDB Server (Compass) running
locally, and a code editor like VS Code.
Installation:
Clone the repository or extract the project folder.
Navigate to the backend folder and run npm install to install Express and Mongoose
dependencies.
Navigate to the frontend folder and run npm install to install React and Axios
dependencies.
Configure environment variables (if any) such as the MongoDB connection URI in a .env
file.

5. Folder Structure

Client: Organized into /src/components containing separate folders for common (Login/Register), user, admin, and agent dashboards. Static assets are kept in the /public folder.

Server: Structured using the MVC pattern with folders for /models (Mongoose schemas), /routes (Express API endpoints), and the main entry point index.js.

6. Running the Application

Backend: Open a terminal in the server directory and run npm start (or npx nodemon index.js) to initialize the server on port 5000.

Frontend: Open a separate terminal in the client directory and run npm start to launch the React app on localhost:3000.

7. API Documentation

POST /api/auth/register: Registers a new user with email, password, and role.

POST /api/auth/login: Authenticates user credentials and returns user details.

POST /api/complaints: Submits a new complaint document to the database.

GET /api/complaints: Retrieves a list of complaints (Admin sees all, Agent sees assigned, User sees their own).

PUT /api/complaints/update: Modifies the status of a complaint (e.g., to "Assigned" or "Resolved").
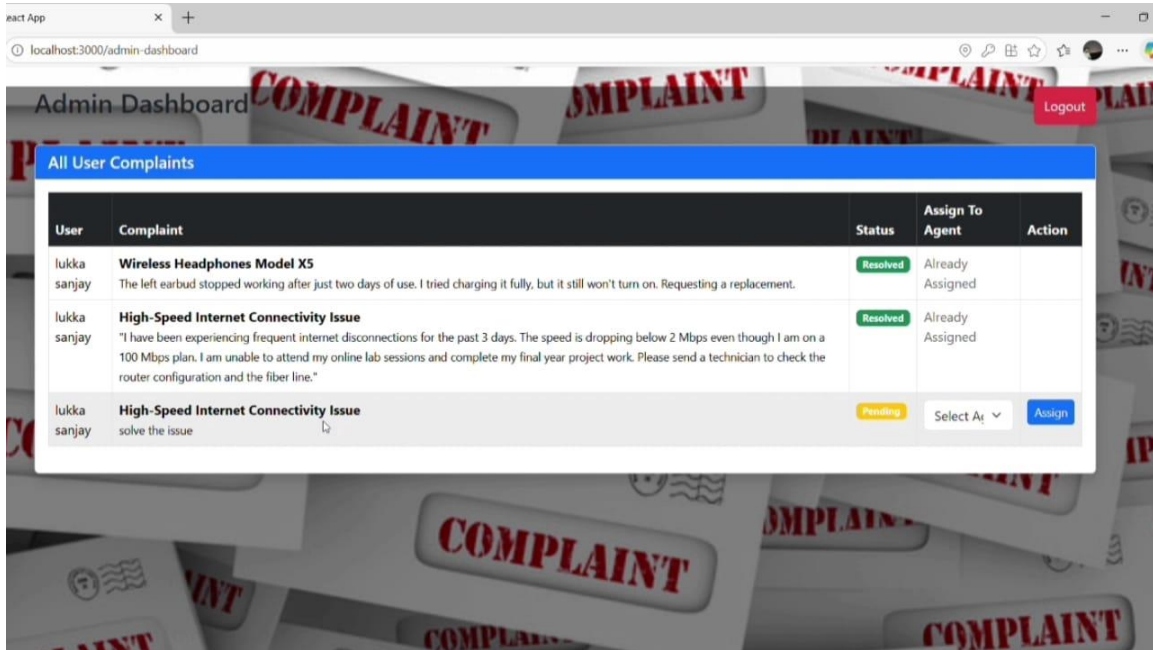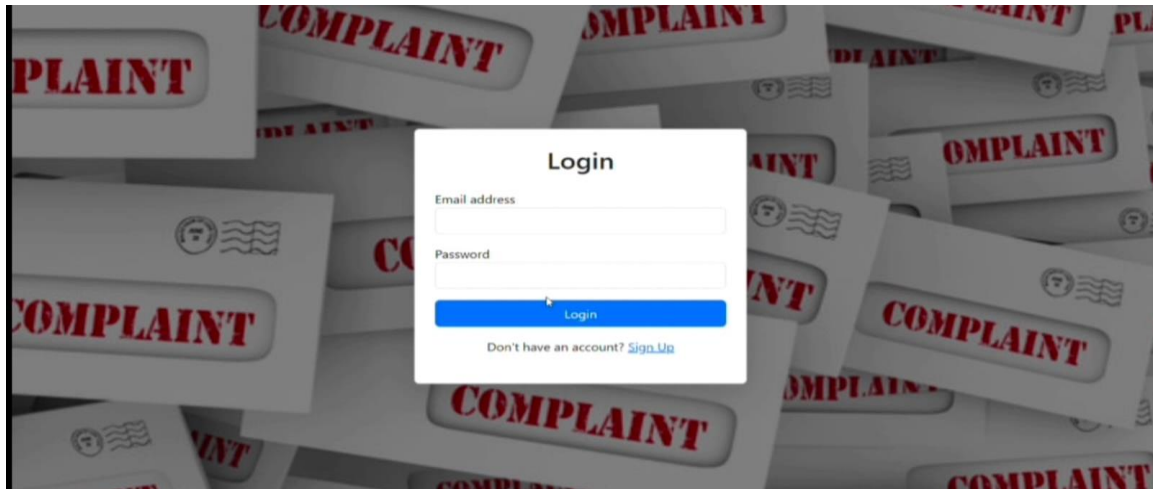
8. Authentication

The system uses secure role-based authentication. Upon a successful login via the backend API, the user's ID, Name, and Role (Customer, Admin, or Agent) are stored locally.

React Router uses this stored role to authorize access, ensuring an Agent cannot access the Admin dashboard, and vice versa.

9. User Interface

The UI features a dark, professional tech-operations background with semi-transparent "Glass" cards for forms.

10. Testing

Manual User Acceptance Testing (UAT) was conducted to verify the end-to-end data flow.

This included registering test accounts for all three roles and verifying that a complaint successfully transitions from User Creation -> Admin Assignment -> Agent Resolution.

11. Screenshots or Demo

Demo Video Link:

https://drive.google.com/drive/folders/1IsfoLoByEv0VklPzHOBKAreumnf9p850

User Dashboard:

Agent Dashboard:

12. Known Issues
The application currently relies on active internet access to render external background images.
Users must manually refresh the dashboard to see the latest status updates assigned by the Admin.

13. Future Enhancements
Integration of automated Email or SMS notifications when a complaint status changes.

Implementation of an AI chatbot to help users categorize their complaints automatically before submission.
Exporting monthly complaint resolution reports as PDF/Excel files for Admin analytics.