Full Stack Development

1. Introduction

- **Project Title:** Complaint Resolver – Online Complaint Registration and Management System
- **Team Members:** Bandaru Phani Venkata Ram Kumar
  Boddu Amrutha
  M Jayasree
  Yuva Sree Mamidisetty

2. Project Overview

RightVoice serves as a digital platform for logging and managing user grievances. It simplifies the complaint process, supports status tracking, and provides seamless interaction between users and service personnel. It also helps organizations address concerns effectively while meeting compliance standards and improving user trust.

🏆 Key Aims:

- Streamline the process of registering complaints
- Allow complaint routing to the right support personnel
- Facilitate real-time updates and transparency
- Maintain secure and centralized complaint records

3. Architecture

- **Frontend:** Built with React.js, styled using Bootstrap and Material UI. Axios is used for backend interaction.
- **Backend:** Developed in Node.js using Express. RESTful APIs handle logic, with real-time communication powered by Socket.IO and WebRTC.
- **Database:** MongoDB stores complaint details, users, messages, and logs using Mongoose schemas.

4. Setup Instructions

🔧 Step 1: Requirements

- Ensure you have the following:
- Node.js
- MongoDB (local or Atlas)
- Git

📥 Step 2: Clone the Project

⚙️ Step 3: Backend Configuration

- Navigate to backend/
- Run npm install
- Add environment variables in a .env file
- Start the backend server

💻 Step 4: Frontend Setup

- Navigate to frontend/
- Run npm install
- Start the frontend

🌐 Step 5: Launch the App

- Visit: http://localhost:3000
  Register with a role and begin submitting, managing, or resolving complaints

5. Folder Structure

```
Complaint Resolver/
│
├── backend/            # Node.js + Express backend
│   ├── controllers/     # Logic for complaints, users, etc.
│   ├── models/          # Database schemas (e.g., User, Complaint)
│   ├── routes/          # API routes (e.g., /api/complaints)
│   ├── middleware/      # Auth & error handlers
│   ├── config/          # DB connection, environment setup
│   └── server.js        # Entry point for backend server

├── frontend/           # React frontend
│   ├── public/          # Static files
│   ├── src/
│   │   ├── components/    # UI components (Navbar, Form, Cards)
│   │   ├── pages/         # Pages like Login, Dashboard
│   │   ├── services/      # API call helpers
│   │   ├── App.js         # Main app
│   │   └── index.js       # Entry point
│
├── .env                  # Environment variables
├── package.json          # For managing dependencies (root)
└── README.md             # Project documentation
```

6. Running the Application

➢ Install Dependencies

- Go to root, then install both client and server dependencies:

```
npm install
cd backend && npm install
cd ../frontend && npm install
```

➢ Environment Setup

- Create a .env file in backend/:

➢ Start the App

- From the root directory:
- This runs both backend and frontend using concurrently.

7. API Documentation

🔐 Authentication APIs

- **Register**
  Allows a new user, agent, or admin to create an account by providing their name, email, password, and role. On successful registration, the system responds with a confirmation message and a login token.

- **Login**
  Enables existing users to access their accounts by verifying their credentials. Upon successful login, they receive a token to authenticate future requests.

📝 Complaint Handling APIs

- **File a Complaint**
  Users can submit a new complaint with a title, description, address, and any supporting files such as images or documents.

- **View Personal Complaints**
  Users can see all the complaints they have submitted and track the status of each.

- **View All Complaints (Admin)**
  Admins can view every complaint submitted on the platform, including which agent (if any) is handling it and what stage it is at.

- **Assign Complaint to Agent**
  Admins can allocate complaints to specific agents for investigation and resolution based on availability or specialization.

- **View Assigned Complaints (Agent)**
  Agents can view a list of complaints that have been assigned to them along with all relevant details.

💬 Chat/Messaging APIs

- **Send a Message**
  Users and agents can exchange messages about a complaint. This supports real-time or near real-time communication between the two parties.

- **View Complaint Conversation**
  Retrieves the full message history for a particular complaint so either the user or agent can refer back to previous messages.

8. Authentication

1. User Registration

- **Purpose:** Allows anyone to sign up and create an account on the platform.
- **Details Collected:** Name, email, password, and role selection (user, admin, or agent).
- **Access Level:** Open to all – no prior login required.
- **Outcome:** A new account is created, and a secure token is provided for future use.

2. User Login

- **Purpose:** Enables registered users to log in with their credentials.
- **Details Required:** Registered email and password.
- **Access Level:** Public – login required to access further features.
- **Outcome:** Upon successful login, the system issues a secure token (used to verify the user's identity during each action in the app).

3. Role-Based Access

- **Users** can view and manage their own complaints.
- **Agents** are restricted to handling only complaints assigned to them.
- **Admins** have full privileges to view and manage all users, agents, and complaints.

9.User Interface

🧑 For Users

- Sign Up/Login
- Dashboard to lodge and follow complaints
- Chat window for direct communication
- Complaint tracking and logout option

🛠 For Admins

- View all complaints
- Assign to agents
- Filter/search complaints
- User/agent overview and logout

🧑‍💼 For Agents

- View assigned complaints
- Check details and files
- Chat with complainant
- Update resolution status

10.Testing :

- Manual testing for workflows like login, complaint entry, assignment, and chat
- Postman used to test API responses and authorization
- Jest tests used for authentication and server routes
- Cross-device and screen-size checks performed for responsiveness

11.Screenshots or Demo

Demo Link: https://drive.google.com/file/d/1rXU0uoTu7kZj1HjFx9D2wa9UEHQfFRRG/view?usp=drivesdk

12.Known Issues :

- SMS notifications pending (email only)
- UI not fully responsive on small screens
- Chat does not auto-refresh
- Some API calls lack loading indicators
- App supports only one language

13. Future Enhancements

- Mobile version using React Native
- Smart replies using AI for repetitive issues
- Real-time admin analytics dashboard