Full Stack Development with MERN

Project Documentation format

1. INTRODUCTION:

ResolveNow is a user-centric digital platform designed to streamline the process of lodging and resolving complaints online across various sectors—be it consumer products, public services, or institutional grievances. It empowers individuals to raise their voice, document issues, and seek resolution in an organized, transparent, and efficient manner**.**

❖ PROJECT TITLE:

Resolve now : your platform for online complaints

❖ TEAM MEMBERS:

- 1.Pusuluri.Ajith – Frontend Development
- 2. Shaik.Reshma – Backend Development
- 3. Seelam.Pallavi – Testing & Deployment
- 4. Mundru.Venkatesh– Final Documentation
- 5. Kotapati.SaiRam – UI/UX Design & Survey Analysis

2. PROJECT OVERVIEW:

ResolveNow is a digital complaint management platform aimed at empowering citizens to file, track, and resolve grievances across sectors—including government services, consumer products, public utilities, and institutional issues. Designed with transparency, speed, and user convenience in mind, it transforms the traditional complaint process into a more accessible and accountable experience.

❖ PURPOSE:

To create a centralized, easy-to-use platform where users can:

- Lodge complaints from anywhere, anytime
- Track complaint status in real time

- Engage with relevant authorities and service providers directly

❖ FEATURES:

- **Simplified Complaint Submission**: Users can register complaints through a guided interface that ensures all necessary details are captured without hassle.

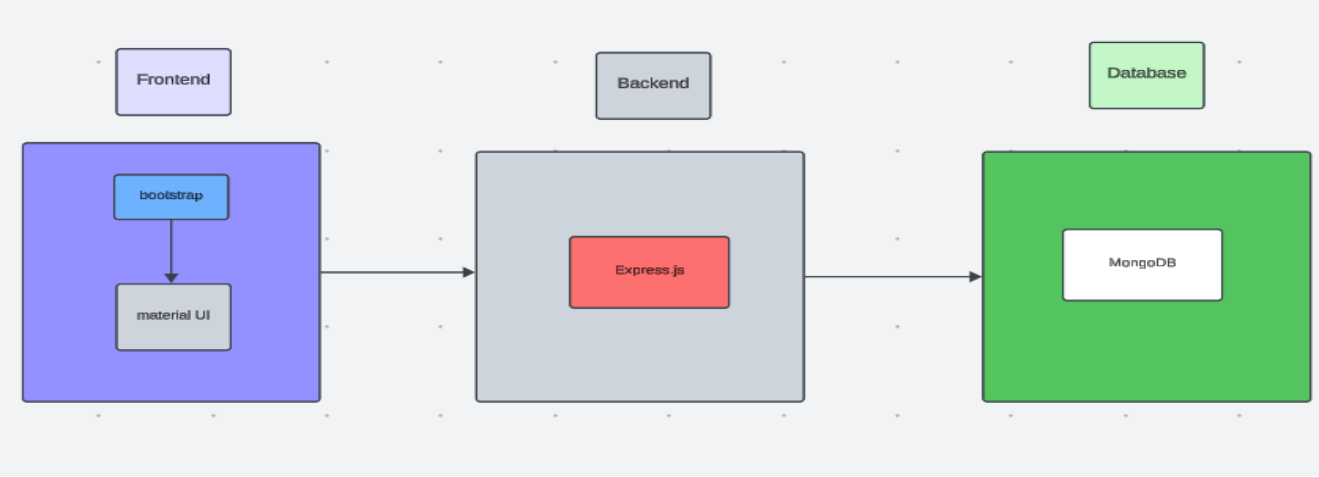
- **Real-Time Tracking**: The platform allows complainants to monitor the status of their case with timely updates.

- **Multi-Channel Integration**: Connects with government portals, service providers, and customer care systems to ensure wider reach and accountability.

- **Secure Documentation**: All user submissions are encrypted and stored safely for future reference and legal validity.

- **Feedback Loop**: Once resolved, users can provide feedback and rate the resolution experience, ensuring continuous improvement.

3. 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.

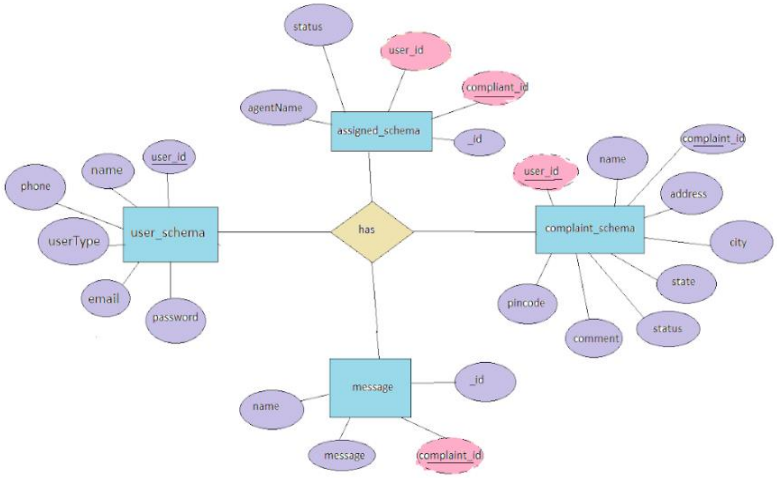
The frontend utilizes the bootstrap and material UI library to establish real-time and better UI experience for any user whether it is agent, admin or ordinary user working on it.

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 DIAGRAM:



4. SETUP INSTRUCTIONS:

To set up the ResolveNow complaint platform, first ensure that you have Node.js, MongoDB, and Git installed on your system, along with a code editor like VS Code. Start by organizing your project into two main folders: one for the backend and one for the frontend. In the backend folder, initialize a Node.js project, install essential packages like Express, Mongoose, and dotenv, then create routes and models for users and complaints, connecting everything to MongoDB. For the frontend, use React by creating a React app and install additional libraries like Axios and React Router. Develop pages for adding complaints, tracking their status, and managing feedback. Link the frontend and backend using API calls, and securely store configuration details in a .env file. Finally, test the system using tools like Postman, and once everything works smoothly, deploy the frontend and backend using platforms like Netlify and Render. This gives you a working, user-friendly online complaint platform that's ready to serve real users.

❖ PRE-REQUISTIC:

Here are the key prerequisites for developing a full-stack application using Express Js, MongoDB, React.js:

✓Node.js and npm:

Node.js is a powerful JavaScript runtime environment that allows you to run JavaScript code on the server-side. It provides a scalable and efficient platform for building network applications.

Install Node.js and npm on your development machine, as they are required to run JavaScript on the server-side.

✓Express.js:

Express.js is a fast and minimalist web application framework for Node.js. It simplifies the process of creating robust APIs and web applications, offering features like routing, middleware support, and modular architecture.

Install Express.js, a web application framework for Node.js, which handles server-side routing, middleware, and API development.

Installation: Open your command prompt or terminal and run the following command:

npm install express

✓MongoDB:

MongoDB is a flexible and scalable NoSQL database that stores data in a JSON-like format. It provides high performance, horizontal scalability, and seamless integration with Node.js, making it ideal for handling large amounts of structured and unstructured data.

Set up a MongoDB database to store your application's data.

✓React.js:

React.js is a popular JavaScript library for building user interfaces. It enables developers to

create interactive and reusable UI components, making it easier to build dynamic and responsive web applications.
Install React.js, a JavaScript library for building user interfaces.
✓**HTML, CSS, and JavaScript**: Basic knowledge of HTML for creating the structure of your app, CSS for styling, and JavaScript for client-side interactivity is essential.
✓**Database Connectivity**: Use a MongoDB driver or an Object-Document Mapping (ODM) library like Mongoose to connect your Express Js server with the MongoDB database and perform CRUD (Create, Read, Update, Delete) operations
✓**Front-end Framework**: Utilize React Js to build the user-facing part of the application, including entering booking room, status of the booking, and user interfaces for the admin dashboard. For making better UI we have also used some libraries like material UI and bootstrap.
✓**Version Control**: Use Git for version control, enabling collaboration and tracking changes throughout the development process. Platforms like GitHub or Bitbucket can host your repository.
✓**Development Environment**: Choose a code editor or Integrated Development Environment (IDE) that suits your preferences, such as Visual Studio Code, Sublime Text, or WebStorm.
Install Dependencies:
• Navigate into the cloned repository directory:
cd freelancer-app-MERN
• Install the required dependencies by running the following commands:
cd client
npm install
../cd server
npm install

Start the Development Server:
• To start the development server, execute the following command:
npm start
You have successfully installed and set up the SB Works application on your local machine. You can now proceed with further customization, development, and testing as needed.

5. PROJECT STRUCUTRE:

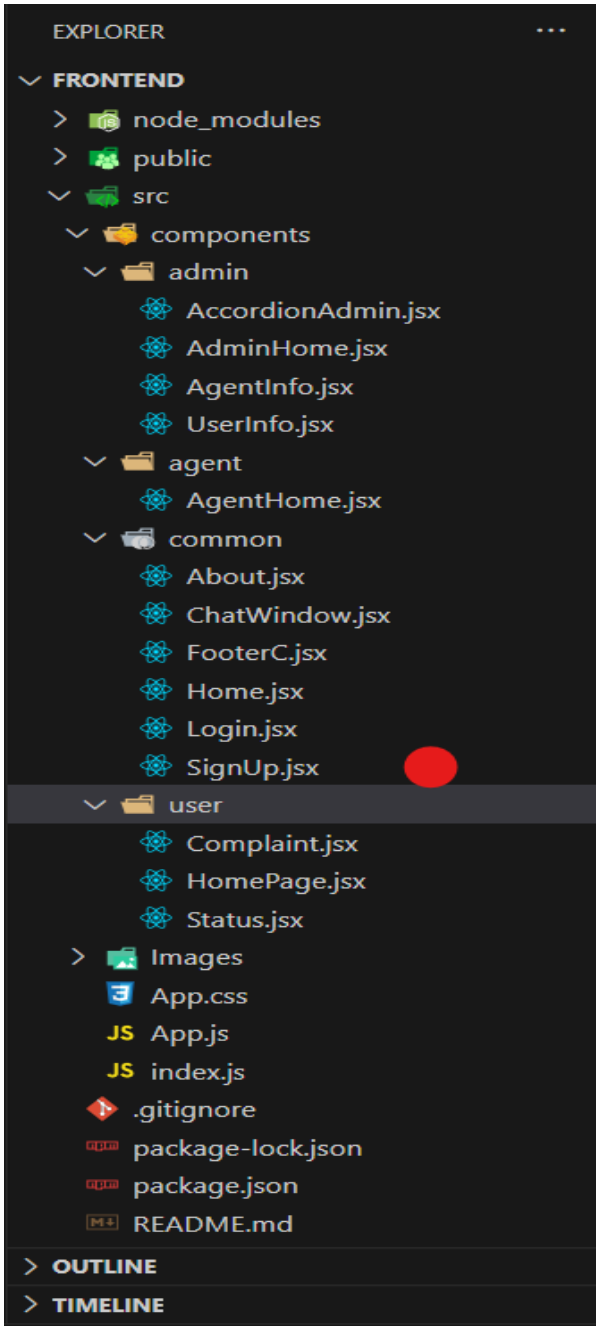
APPLICATION FLOW:
Online Complaint Registration and Management System

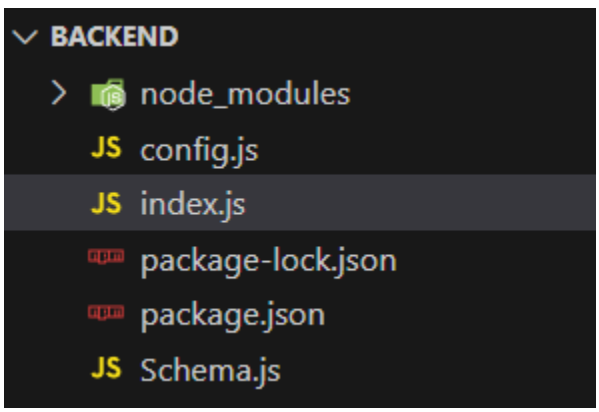
1. Customer/Ordinary User:
    - Role: Create and manage complaints, interact with agents, and manage profile information.

- Flow:
    1. Registration and Login:
        - Create an account by providing necessary information such as email and password.
        - Log in using the registered credentials.
    2. Complaint Submission:
        - Fill out the complaint form with details of the issue, including description, contact information, and relevant attachments.
        - Submit the complaint for processing.
    3. Status Tracking:
        - View the status of submitted complaints in the dashboard or status section.
        - Receive real-time updates on the progress of complaints.
    4. Interaction with Agents:
        - Connect with assigned agents directly using the built-in messaging feature.
        - Discuss complaints further and provide additional information or clarification.
    5. Profile Management:
        - Manage personal profile information, including details and addresses.

2. Agent:
    - Role: Manage complaints assigned by the admin, communicate with customers, and update complaint statuses.
    - Flow:
        1. Registration and Login:
            - Create an account using email and password.
            - Log in using the registered credentials.
        2. Complaint Management:
            - Access the dashboard to view and manage complaints assigned by the admin.
            - Communicate with customers regarding their complaints through the chat window.
        3. Status Update:
            - Change the status of complaints based on resolution or progress.
            - Provide updates to customers regarding the status of their complaints.
        4. Customer Interaction:
            - Respond to inquiries, resolve issues, and address feedback from customers.
3. Admin:

- Role: Oversee the overall operation of the complaint registration platform, manage complaints, users, and agents, and enforce platform policies.
- Flow:
    1. Management and Monitoring:
        - Monitor and moderate all complaints submitted by users.
        - Assign complaints to agents based on workload and expertise.
    2. Complaint Assignment:
        - Assign complaints to the desired agents for resolution.
        - Ensure timely and efficient handling of complaints.
    3. User and Agent Management:
        - Manage user and agent accounts, including registration, login, and profile information.
        - Enforce platform policies, terms of service, and privacy regulations.
    4. Continuous Improvement:
        - Implement measures to improve the platform's functionality, user experience, and security measures.
        - Address any issues or concerns raised by users or agents for better service delivery.


EXPLORER
FRONTEND
node_modules
public
src
components
admin
AccordionAdmin.jsx
AdminHome.jsx
AgentInfo.jsx
UserInfo.jsx
agent
AgentHome.jsx
common
About.jsx
ChatWindow.jsx
FooterC.jsx
Home.jsx
Login.jsx
SignUp.jsx
user
Complaint.jsx
HomePage.jsx
Status.jsx
Images
App.css
App.js
index.js
.gitignore
package-lock.json
package.json
README.md
OUTLINE
TIMELINE


BACKEND
node_modules
config.js
index.js
package-lock.json
package.json
Schema.js

- The first image is of frontend part which is showing all the files and folders that have been used in UI development

The second image is of Backend part which is showing all the files and folders that have been used in backend development

6.RUNNING THE APPLICATION

❖ 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.

❖ 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.

2.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.

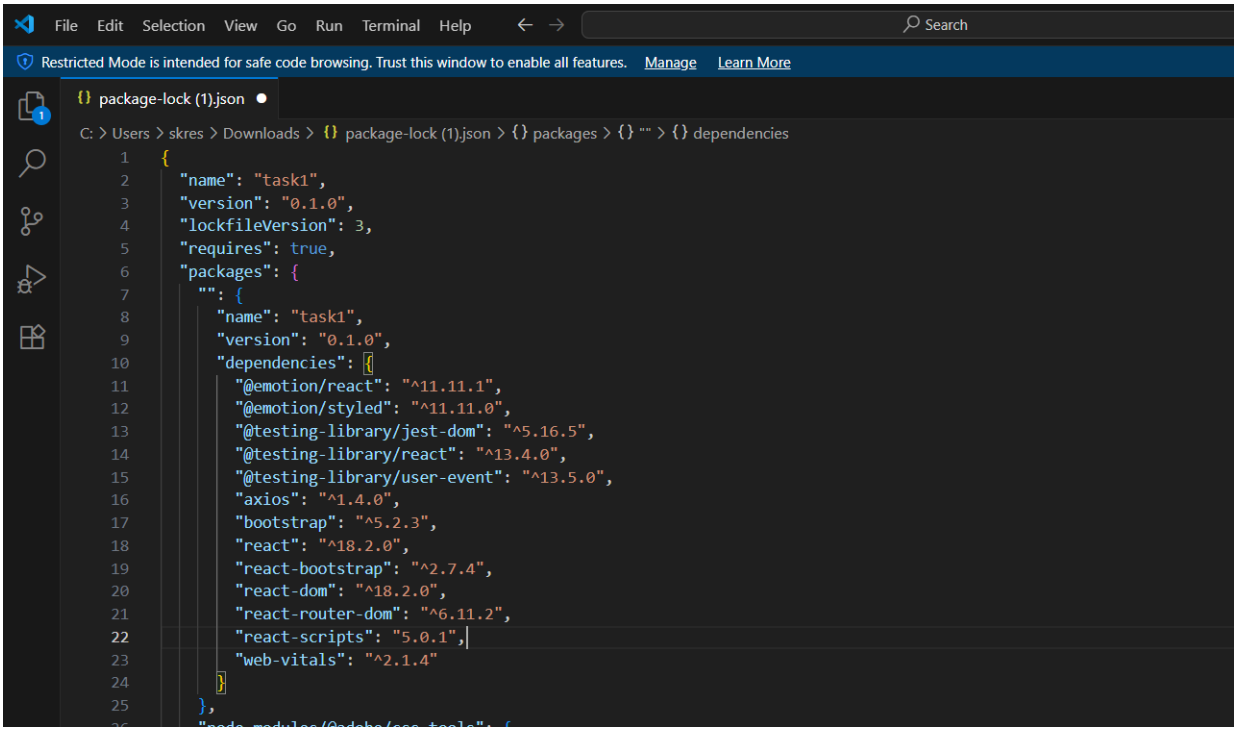
3.Implement frontend logic:

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.

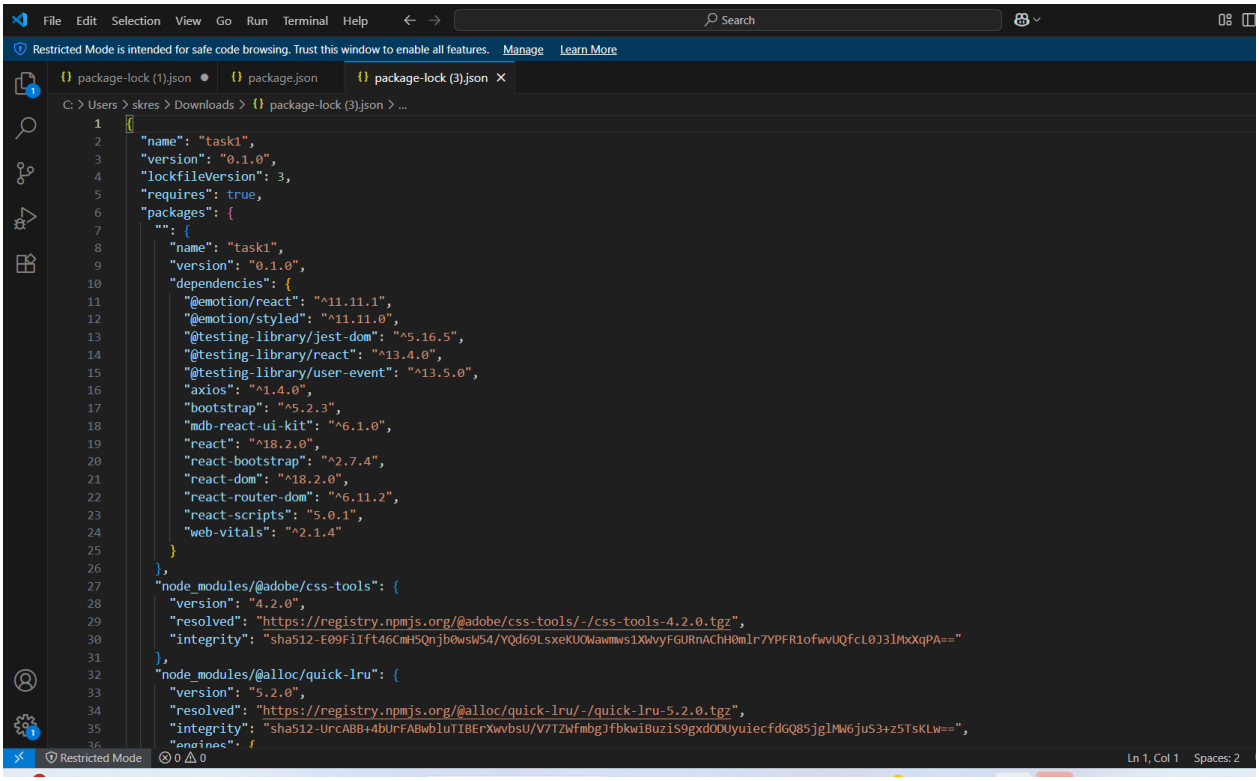
- Integration with API endpoints.
- Implement data binding.

7.API DOCUMENTATION:

🔐 Authentication APIs



After the installation of all the libraries, the package. Json files for the backend looks like the one mentioned below.



The backend of the ResolveNow platform is the engine that powers all the logic behind user actions, complaint processing, data storage, and communication with the frontend. Built typically with Node.js and Express, it handles everything that happens behind the scenes when a user files a complaint, an admin updates a case, or feedback is submitted.
It connects to a database like MongoDB to store and retrieve structured data such as user profiles, complaint records, feedback, and progress updates. The backend includes RESTful APIs that expose endpoints (like /submitComplaint, /getStatus, /login) which the frontend uses to send and receive information. It also manages user authentication and roles—ensuring only authorized users (like administrators or complainants) can access or modify certain data.

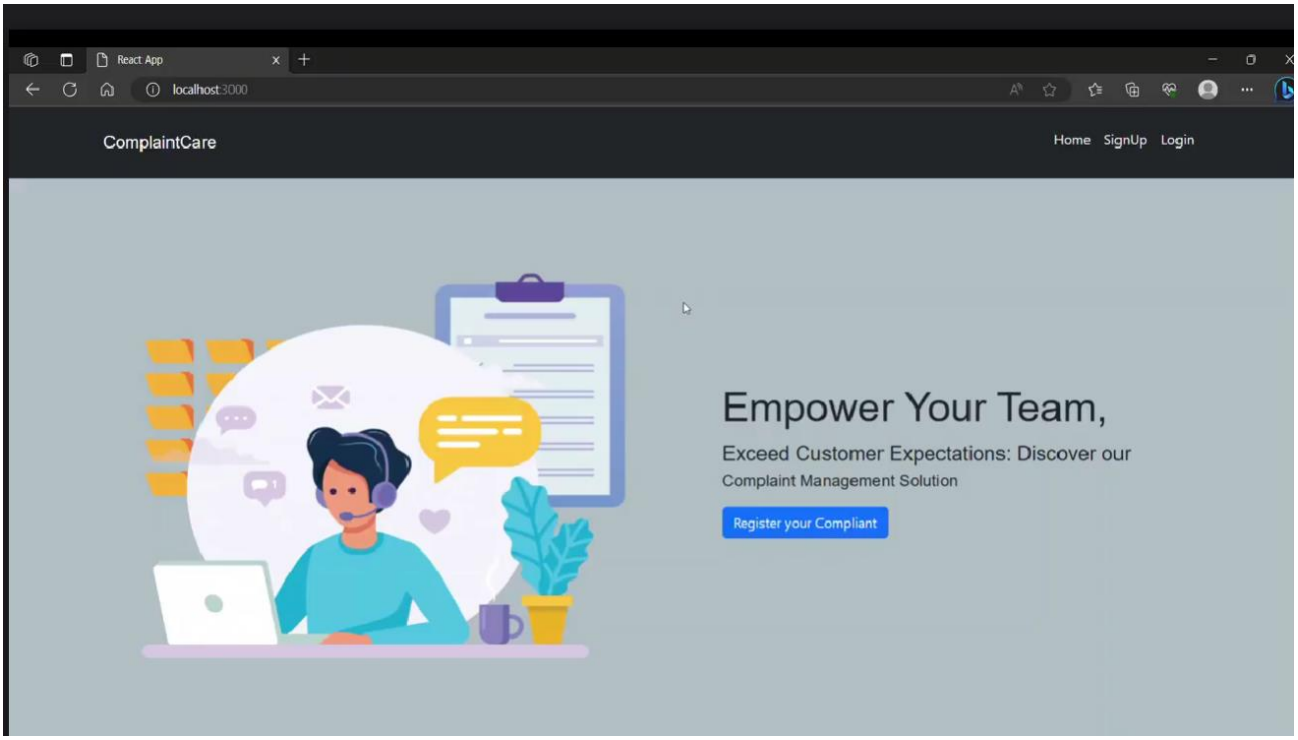
8.AUTHENTICATION:

Authentication in the ResolveNow platform ensures that only authorized users can access specific features—like filing complaints, managing them, or giving feedback. It plays a crucial role in protecting user data and ensuring privacy.
In a typical setup, authentication is handled using **JWT (JSON Web Tokens)**. When a user signs up or logs in, the server verifies their credentials (like email and password). If correct, it creates a token and sends it back to the user's browser. This token acts like a digital ID card—it's included in future requests to prove the user's identity.
The backend checks the validity of the token before allowing access to protected routes. For example, only logged-in users can file complaints, and only admins can update statuses or respond to users. Tokens can also carry **roles** (like "user" or "admin") so the server knows what permissions each person has.

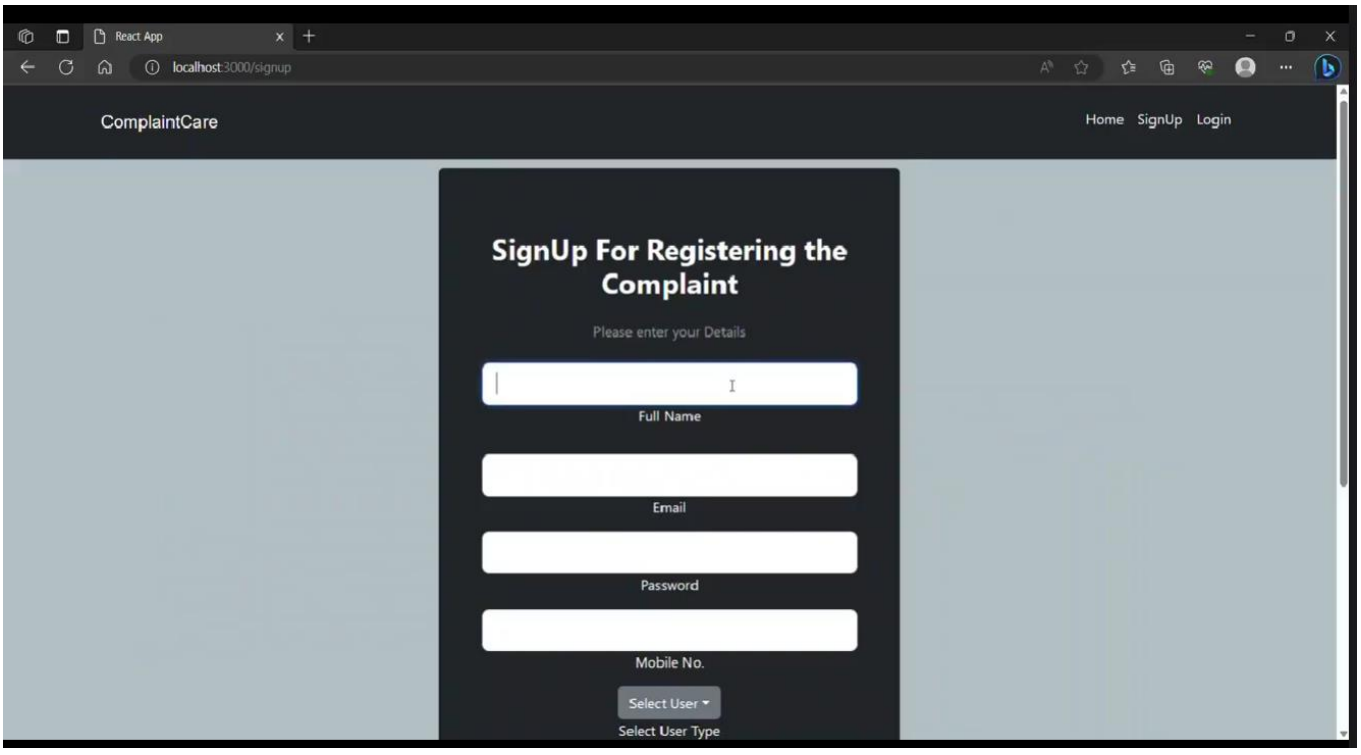
9. USER INTERFACE:

- Providing screenshots or GIFs showcasing different UI features.

Landing page:



Authentication:



10.TESTING:

• Describe the testing strategy and tools used.

□ Testing Strategy:

- **Frontend:** The part of the platform users interact with directly—like forms, buttons, and pages.
- **Backend:** The behind-the-scenes code that handles logic, data storage, and communication with the frontend.
- **API (Application Programming Interface):** A set of rules that allows the frontend and backend to talk to each other.
- **Database:** Where all the information (like user data and complaints) is stored securely.
- **Authentication:** A way to check and confirm a user's identity when they log in.
- **JWT (JSON Web Token):** A small, secure digital "badge" given to users when they log in to prove who they are.
- **Hosting:** Putting your platform on the internet so others can access it.
- **React:** A tool for building dynamic web pages in the frontend.
- **Express:** A tool for creating routes and handling server-side logic in the backend.
- **MongoDB:** A popular database used to store information in the form of flexible documents.

✓ **Installation of required tools**:

1. Open the frontend folder to install necessary tools

For frontend, we use:

- React
- Bootstrap
- Material UI
- Axios
- react-bootstrap

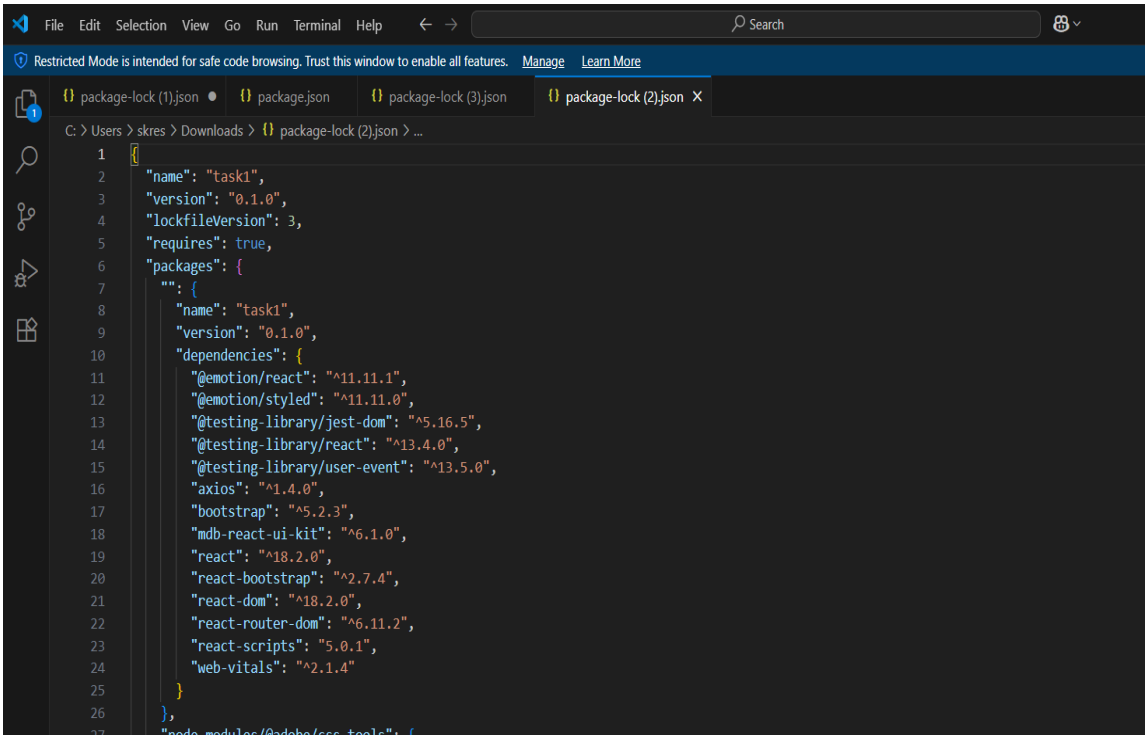
2. Open the backend folder to install necessary tools

For backend, we use:

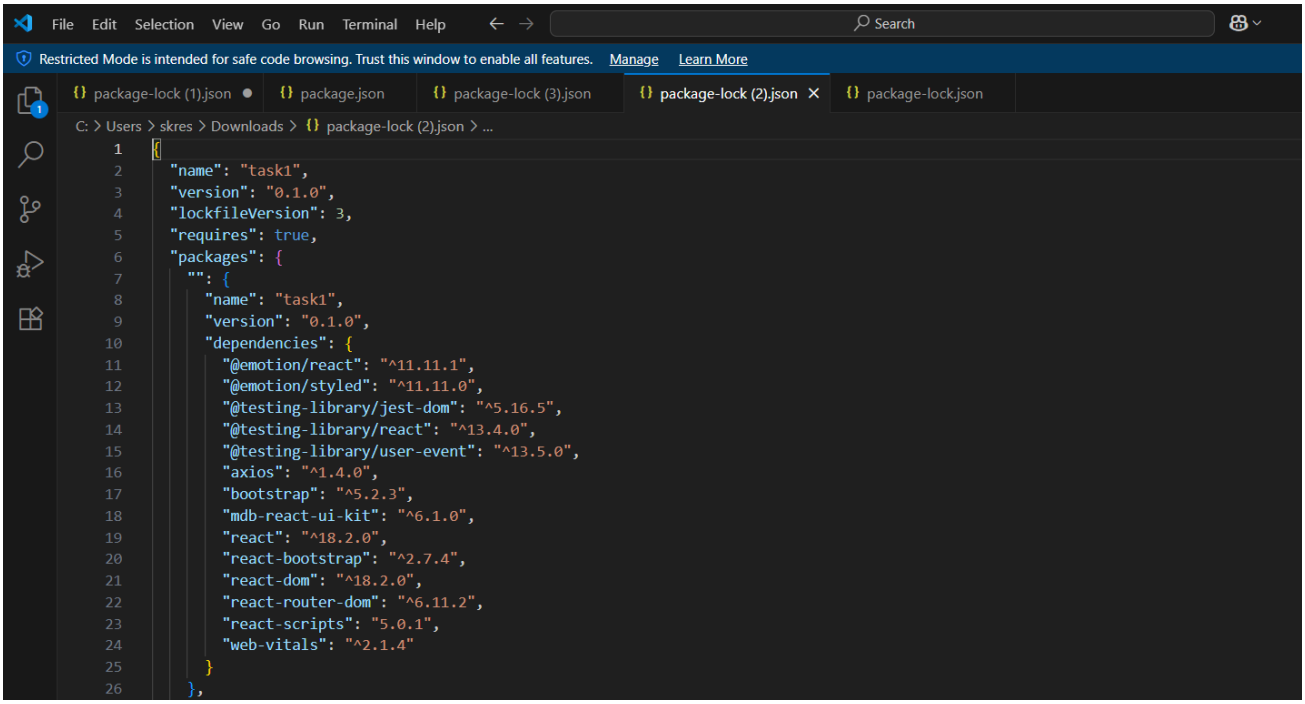
- Express Js
- Node JS
- MongoDB
- Mongoose

- Cors
- Bcrypt

After the installation of all the libraries, the package.json files for the frontend looks like the one mentioned below.



After the installation of all the libraries, the package.json files for the backend looks like the one mentioned below.

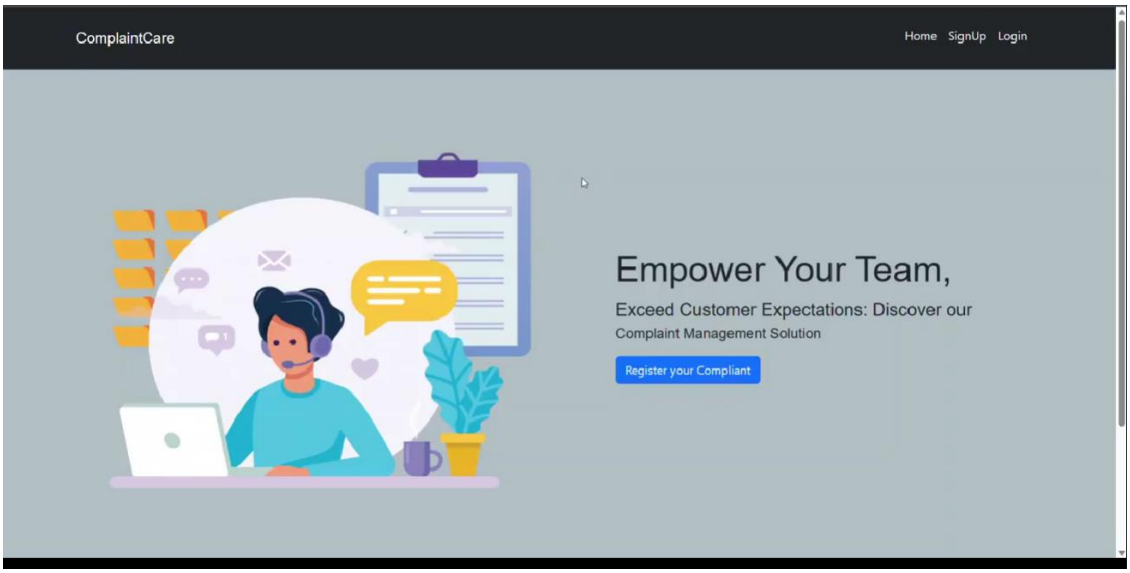


11. SCREENSHOTS OR DEMO:

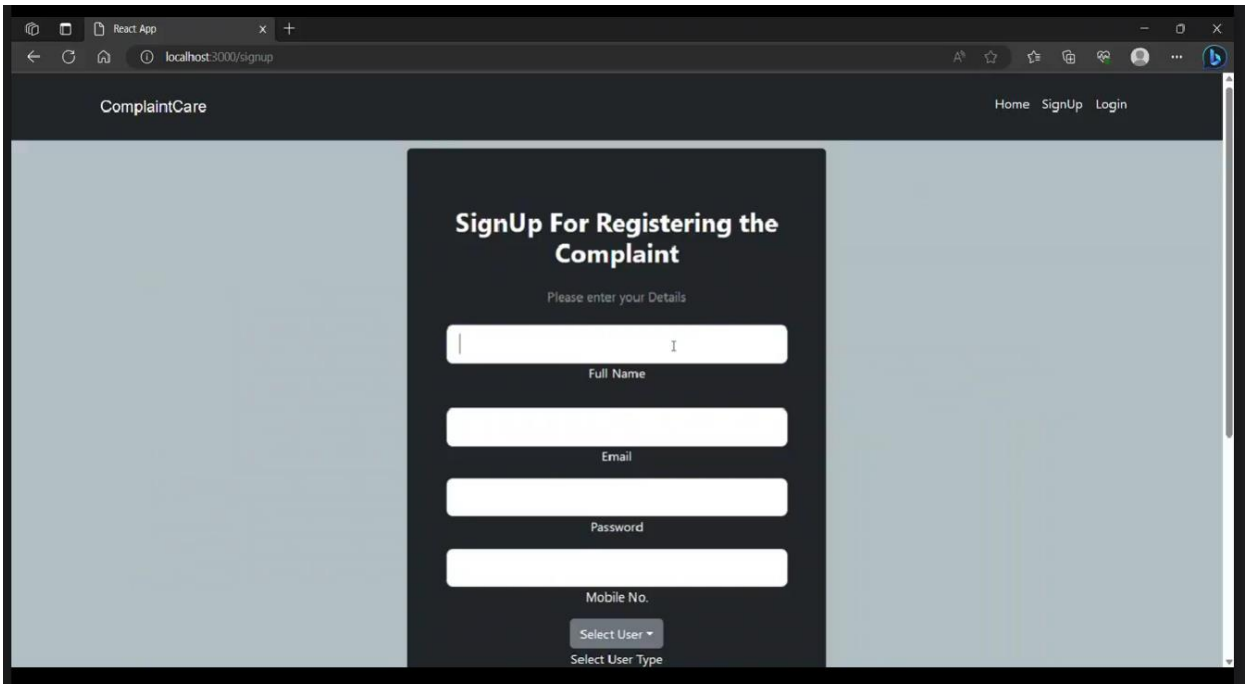
• Providing screenshots or a link to a demo to showcase the application.

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 images provided below.

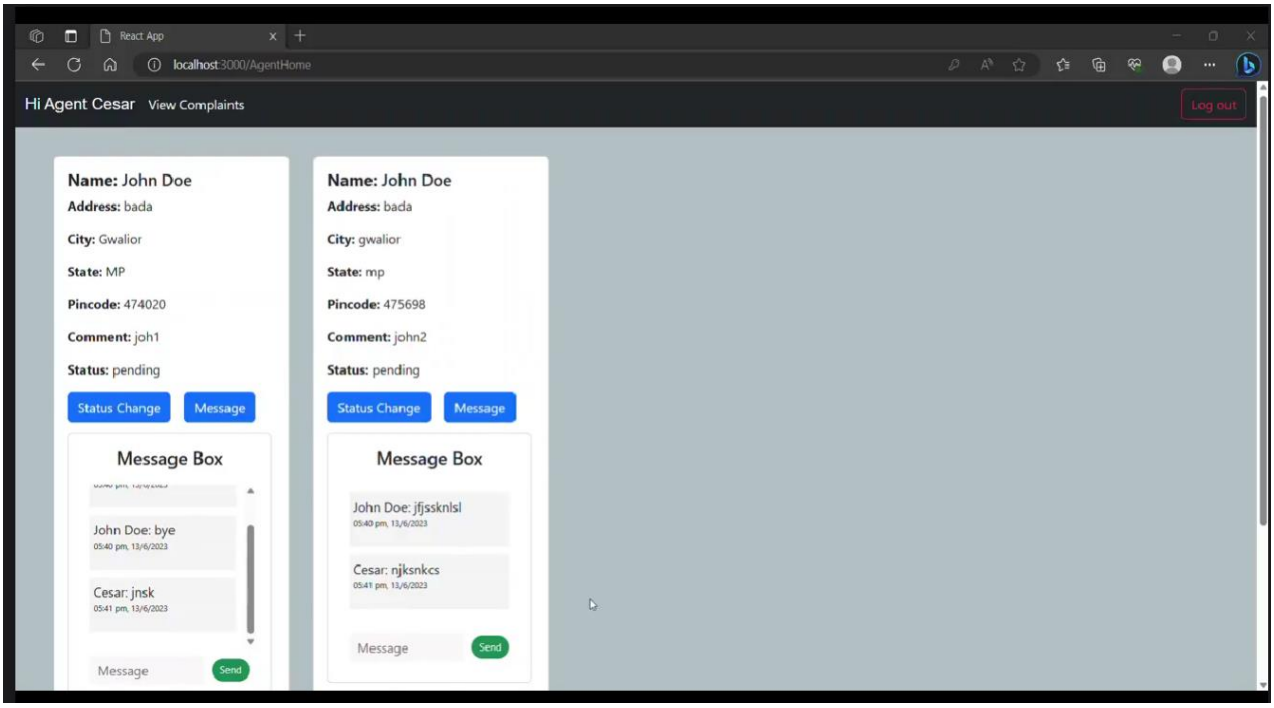
Landing page:



Authentication:



dashboard:



12. KNOWN ISSUES:

- ✓ API errors or timeouts when backend services are overloaded or misconfigured
- ✓ Incorrect role permissions where users or admins access unintended features
- ✓ Token expiration or invalidation causing sudden logouts or access denials
- ✓ Frontend not syncing with backend due to mismatched routes or inconsistent data formats
- ✓ MongoDB connection failures if environment variables are misconfigured or the database is unreachable

13. FUTURE ENHANCEMENTS:

🌐 **1. Multilingual Support**
Enable users to file and track complaints in multiple regional languages, making the platform more accessible and inclusive.

📲 **2. Mobile App Development**
Create dedicated Android and iOS apps with push notifications to improve usability and increase reach, especially in mobile-first regions.

🤖 **3. Chatbot Integration**
Introduce an AI-powered chatbot to guide users through filing complaints, answering FAQs, and providing real-time updates.

📊 **4. Analytics Dashboard**
Provide real-time data visualization for admins—showing trends, resolution times, complaint categories, and overall performance metrics.

🔄 **5. Auto-Routing Engine**
Implement smart routing that directs each complaint to the right department or authority based on category, location, and urgency.