

# ***FULL STACK DEVELOPMENT WITH MERN – PROJECT DOCUMENTATION***

## **1. INTRODUCTION**

**Project Title:** ResolveNow – Online Complaint Registration and Management System

**Team Members:**

**Team ID:** LTVIP2025TMID53123

1. **Team Leader:** Ratna Hadhassa – Documentation
2. **Team Member:** Rajulapati Anil Kumar – No contribution recorded during implementation
3. **Team Member:** Ramisetty Bala Sai Venkat – No participation in project execution
4. **Team Member:** **Rajakumar Varshitha** –Handled the complete frontend and backend development, database setup, authentication, real-time chat, testing, and full project documentation.

## **2. PROJECT OVERVIEW**

**ResolveNow: Your Platform for Online Complaints** is a centralized and user-friendly complaint registration and management system. It enables individuals or organizations to register complaints, monitor their progress, and communicate with agents in real time. This solution is designed to streamline complaint handling, improve transparency, and ensure faster and more satisfactory resolutions.

The platform is built to optimize the entire complaint lifecycle — from submission to resolution — while aligning with compliance and service standards. It ensures data security, user privacy, and operational efficiency across all roles: users, agents, and administrators.

## **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.

## Purpose

The main goal of **ResolveNow** is to provide a reliable and transparent system where users can:

- Quickly register complaints online
- Track complaint status and progress
- Interact directly with the assigned support agent
- Receive timely updates and resolution notifications
- Submit feedback on the quality of service

## Features

1. **User Registration & Login**  
Users can register using email credentials or OAuth methods (e.g., Gmail). A verification process ensures data authenticity.
2. **Complaint Submission & Tracking**  
Users can file complaints by providing relevant details, including the issue, location, and supporting documents. Each complaint can be tracked through a live dashboard with real-time status updates.
3. **Real-Time Agent Interaction**  
Once a complaint is assigned, users can chat directly with the assigned agent through an integrated messaging system, enabling better issue clarification and faster resolution.
4. **Admin Dashboard for Complaint Assignment**  
Admins monitor all complaints and assign them to agents based on workload and expertise. They oversee operations to ensure system efficiency and policy compliance.
5. **Feedback & Rating System**  
After the resolution of complaints, users can provide ratings and feedback, helping improve service quality and accountability.

### 3.ARCHITECTURE

#### Frontend Architecture (React.js)

The frontend of **ResolveNow** is developed using **React.js**, a powerful JavaScript library for building user interfaces. It follows a **component-based architecture**, allowing reusable and maintainable code. Major UI libraries like **Material UI** and **Bootstrap** are used to ensure responsiveness and modern design.

- **Axios** handles API communication between frontend and backend.
- Components are structured by role: UserDashboard, AdminPanel, AgentView, etc.
- Form validation, conditional rendering, and localStorage are used for session handling and role-specific access.

#### Backend Architecture (Node.js + Express.js)

The backend is built on **Node.js** with the **Express.js** framework, following a modular and RESTful API structure. It handles:

- **Routing** for user, admin, agent, and complaint-related endpoints
- **JWT-based Authentication** and middleware for access control
- **Real-time communication** via **Socket.IO** for user-agent chat
- Middleware like cors, body-parser, and custom error handlers for smooth request handling

#### Database Architecture (MongoDB + Mongoose)

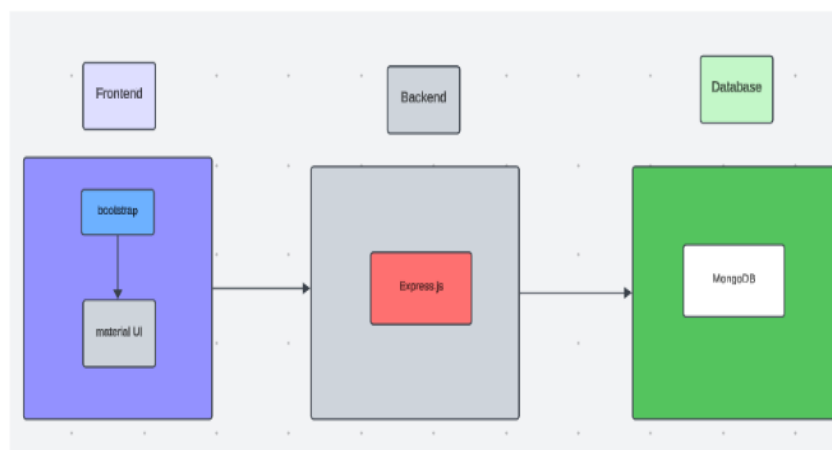
MongoDB is used as the NoSQL database to store application data. Mongoose ORM is used to define clear schemas and interact with the database efficiently.

#### Key Schemas:

- **User Schema:** Stores user details (name, email, password, role – user/admin/agent)
- **Complaint Schema:** Stores complaint info including userId, details, status, timestamps
- **Message Schema:** Stores messages between users and agents with complaintId references
- **Feedback Schema:** Captures user ratings and comments post-resolution

Data is stored in collections and linked using ObjectIds, enabling relational behavior in a document-based model.

### 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.

## **4. SETUP INSTRUCTIONS**

### **PRE-REQUISITES**

Here are the key prerequisites for developing a full-stack application using Node.js, 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.

Download: <https://nodejs.org/en/download/>

Installation instructions: <https://nodejs.org/en/download/package-manager/>

#### **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:

### **INSTALLATION**

**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.

Download: <https://www.mongodb.com/try/download/community>

Installation instructions: <https://docs.mongodb.com/manual/installation/>

## **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.

Follow the installation guide: <https://reactjs.org/docs/create-a-new-react-app.html>

**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 Node.js server with the MongoDB database and perform CRUD (Create, Read, Update, Delete) operations. To Connect the Database with Node JS go through the below provided link:

<https://www.section.io/engineering-education/nodejs-mongoosejs-mongodb/>

**Front-end Framework:** Utilize Reactjs to build the user-facing part of the application, including entering complaints, status of the complaints, 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.

Git: Download and installation instructions can be found at: <https://git-scm.com/downloads>

**Development Environment:** Choose a code editor or Integrated Development Environment (IDE) that suits your preferences, such as Visual Studio Code, Sublime Text, or WebStorm.

- Visual Studio Code: Download from <https://code.visualstudio.com/download>

To run the existing Video Conference App project downloaded from GitHub:

Follow below steps:

Clone the Repository:

- Open your terminal or command prompt.
- Navigate to the directory where you want to store the e-commerce app.
- Execute the following command to clone the repository:

**git clone:** <https://github.com/awdhesh-student/complaint-registry.git>

## Install Dependencies:

- Navigate into the cloned repository directory:  
cd complaint-registry
- Install the required dependencies by running the following commands:  
cd frontend  
npm install  
cd ../backend  
npm install

## Start the Development Server:

- To start the development server, execute the following command:  
npm start
- The online complaint registration and management app will be accessible at <http://localhost:3000>

You have successfully installed and set up the online complaint registration and management app on your local machine. You can now proceed with further customization, development, and testing as needed.

## 5. FOLDER STRUCTURE

### PROJECT STRUCTURE:

### 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.

### 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.

### 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:

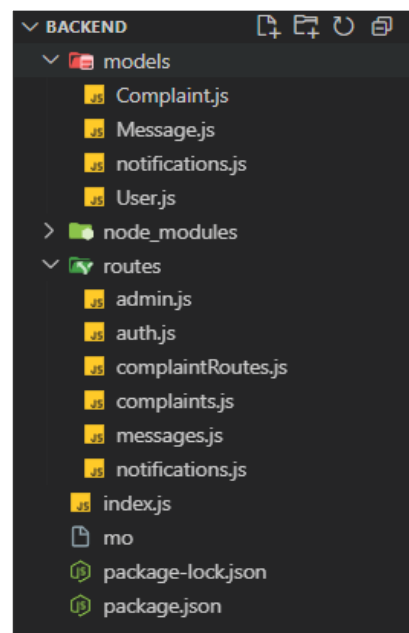
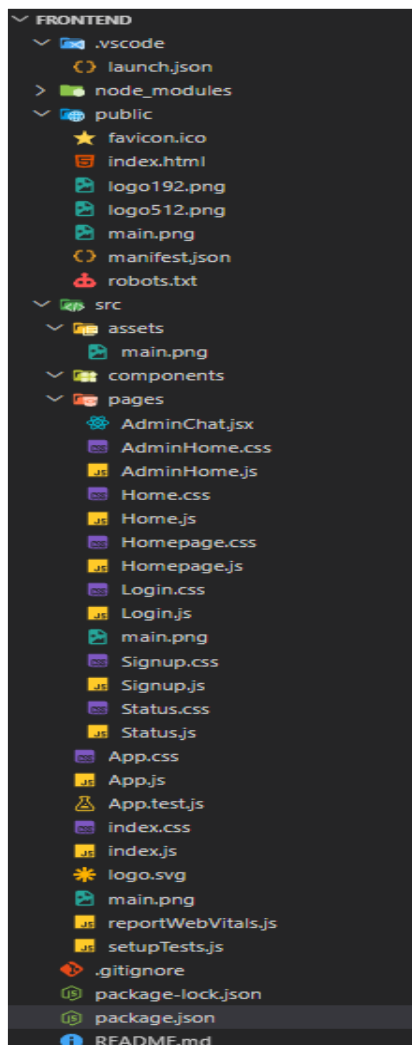
- Assign complaints to the desired agents for resolution.
- Ensure timely and efficient handling of complaints.

## 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.



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

To run the **ResolveNow** application locally, follow these steps:

### **Start the Frontend Server**

```
Copy code
cd frontend
npm install    # Install all required frontend dependencies
npm start     # Starts the React development server
```

The frontend will be available at:

<http://localhost:3000>

### **Start the Backend Server**

```
Copy code
cd backend
npm install    # Install all backend dependencies
node index.js # Starts the Node.js + Express server
```

- The backend will be running at: <http://localhost:8000>
- MongoDB will automatically connect using the configured URI (e.g.,  
mongodb://localhost:27017/complaintDB)
- A success message like "**MongoDB connected**" or "**Database connected successfully**" will confirm a working connection in the terminal.

## **7. API DOCUMENTATION**

### **Base URL**

<http://localhost:8000/api>

### **Project Setup and Configuration**

#### **1. Create project folders and files:**

Now, firstly create the folders for frontend and backend to write the respective code and install the essential libraries.

- Client folders.
- Server folders

#### **2. 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 looks like the one mentioned below.

```
package.json x
package.json > private
3  "version": "0.1.0",
4  "private": true,
5  "dependencies": {
6    "@emotion/react": "^11.14.0",
7    "@emotion/styled": "^11.14.0",
8    "@mui/material": "^7.1.2",
9    "@testing-library/dom": "^10.4.0",
10   "@testing-library/jest-dom": "^6.6.3",
11   "@testing-library/react": "^16.3.0",
12   "@testing-library/user-event": "^13.5.0",
13   "axios": "^1.10.0",
14   "bootstrap": "^5.3.7",
15   "react": "^19.1.0",
16   "react-dom": "^19.1.0",
17   "react-router-dom": "^7.6.2",
18   "react-scripts": "5.0.1",
19   "uuid": "^11.1.0",
20   "web-vitals": "^2.1.4"
21 },
22 "scripts": {
23   "start": "react-scripts start",
24   "build": "react-scripts build",
25   "test": "react-scripts test",
26   "eject": "react-scripts eject"
27 },
28 "eslintConfig": {
29   "extends": [
30     "react-app",
31     "react-app/jest"
32   ]
33 },
34 "browserslist": {
35   "production": [
36     ">0.2%",
37     "not dead",
38     "not op_mini all"
39   ],
40   "development": [
41     "last 1 chrome version",
42     "last 1 firefox version",
43     "last 1 safari version"
44   ]
45 }
46 }
47
```

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

```

package.json x
package.json > ...
1  {
2    "name": "task1",
3    "version": "0.1.0",
4    "proxy": "http://localhost:8000",
5    "private": true,
6    "dependencies": {
7      "@emotion/react": "^11.11.1",
8      "@emotion/styled": "^11.11.0",
9      "@testing-library/jest-dom": "^5.16.5",
10     "@testing-library/react": "^13.4.0",
11     "@testing-library/user-event": "^13.5.0",
12     "axios": "^1.4.0",
13     "bootstrap": "^5.2.3",
14     "mdb-react-ui-kit": "^6.1.0",
15     "react": "^18.2.0",
16     "react-bootstrap": "^2.7.4",
17     "react-dom": "^18.2.0",
18     "react-router-dom": "^6.11.2",
19     "react-scripts": "5.0.1",
20     "web-vitals": "^2.1.4"
21   },
22   "scripts": {
23     "start": "react-scripts start",
24     "build": "react-scripts build",
25     "test": "react-scripts test",
26     "eject": "react-scripts eject"
27   },
28   "eslintConfig": {
29     "extends": [
30       "react-app",
31       "react-app/jest"
32     ]
33   },
34   "browserslist": {
35     "production": [
36       ">0.2%",
37       "not dead",
38       "not op_mini all"
39     ],
40     "development": [
41       "last 1 chrome version",
42       "last 1 firefox version",
43       "last 1 safari version"
44     ]
45   }
46 }
47

```

## Authentication Routes

### 1. User Signup

- **Endpoint:** /auth/signup
- **Method:** POST
- **Body Parameters:**

```

json
{
  "name": "Varshitha",
  "email": "user@example.com",
  "password": "Password123",
  "userType": "user"
}

```

- **Response:**

```

json
{
  "message": "User registered successfully",
  "user": { "_id": "abc123", "email": "user@example.com" }
}

```

```
}
```

## 2. User Login

- **Endpoint:** /auth/login

- **Method:** POST

- **Body Parameters:**

```
json
{
  "email": "user@example.com",
  "password": "Password123"
}
```

- **Response:**

```
json
{
  "token": "jwt_token_here",
  "user": { "_id": "abc123", "role": "user" }
}
```

## Complaint Routes

### 3. Register a Complaint

- **Endpoint:** /complaints/register

- **Method:** POST

- **Headers:** Authorization: Bearer <token>

- **Body Parameters:**

```
json
{
  "name": "Varshitha",
  "address": "Hyderabad",
  "city": "Hyderabad",
  "state": "Telangana",
  "pincode": "500001",
  "comment": "Streetlight not working"
}
```

- **Response:**

```
json
{
  "message": "Complaint registered successfully",
  "complaintId": "cmp123"
}
```

### 4. Get Complaints by User

- **Endpoint:** /complaints/user/:userId

- **Method:** GET

- **Headers:** Authorization: Bearer <token>

- **Response:**

```
json
[
  {
    "_id": "cmp123",
```

```
    "comment": "Streetlight not working",
    "status": "Pending"
  }
]
```

## 5. Update Complaint Status (Admin)

- **Endpoint:** /complaints/status/:id
- **Method:** PATCH
- **Headers:** Authorization: Bearer <admin\_token>
- **Body Parameters:**

```
json
{
  "status": "In Progress"
}
```

- **Response:**
- ```
json
{
  "message": "Status updated successfully"
}
```

## Chat Routes

### 6. Send a Message

- **Endpoint:** /messages
  - **Method:** POST
  - **Body Parameters:**
- ```
json
{
  "name": "User",
  "message": "Any update?",
  "complaintId": "cmp123"
}
```
- **Response:**
- ```
json
{
  "message": "Message sent"
}
```

## 8. AUTHENTICATION

The **ResolveNow** platform uses **JWT (JSON Web Token)** based authentication to securely verify and manage user sessions across different roles (user, agent, admin).

### How It Works:

1. **User Login:**
  - On successful login, the server generates a **JWT token** containing the user's ID and role.
  - The token is sent back to the client and stored in **localStorage**.
2. **Token Format:**
  - Encodes user ID and role.

- Signed with a secret key defined in the .env file (JWT\_SECRET).

### 3. **Token Usage:**

- The token is attached to all protected API requests in the Authorization header:

Authorization: Bearer <token>

### 4. **Protected Routes:**

- Backend uses middleware to verify the token before granting access to sensitive routes.
- Role-based access control is enforced:
  - **Users** can register complaints and view their own.
  - **Agents** can view and respond to assigned complaints.
  - **Admins** can manage users, assign complaints, and update statuses.

## **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.

### • **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:**

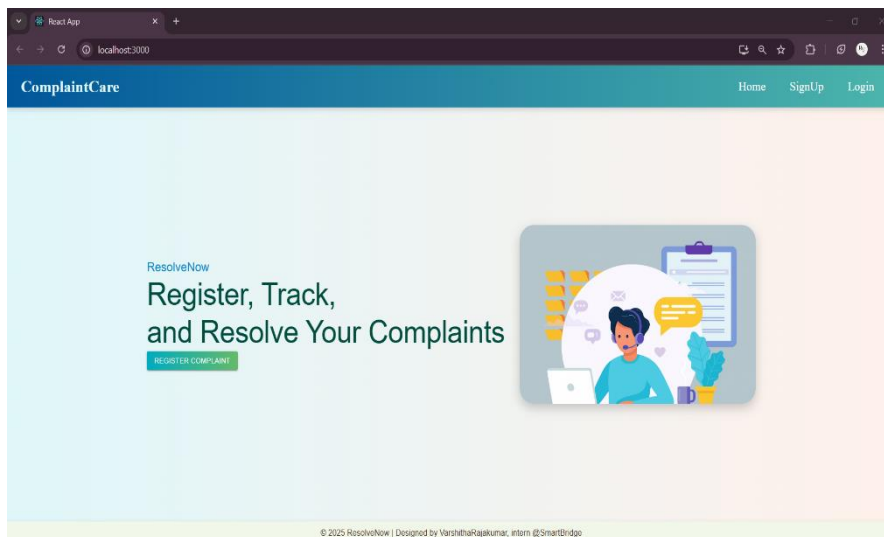
- Implement user authentication using strategies like JSON Web Tokens (JWT) or session-based authentication.
- Create routes and middleware for user registration, login, and logout.
- Set up authentication middleware to protect routes that require user authentication.

- 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.

## 9. USER INTERFACE

The **ResolveNow** user interface is built with **React.js**, styled using **Material UI** and **Bootstrap** to ensure a clean, modern, and responsive experience across devices.

### HOME:



### SIGNUP :

## LOGIN :

React App

localhost:3000/login

ComplaintCare

Home SignUp Login

Login For Registering the Complaint

Email\* v1@gmail.com

Password\* v1

LOGIN

## USER-DASHBOARD :

React App

localhost:3000/homepage

Hi, VARSHITHA RAJAKUMAR

Status

REGISTER

Name: VARSHITHA RAJAKUMAR

Address: JALAPETA

City: Machilipatnam

State: Andhra Pradesh

Pincode: 521001

Complaint Description: WATER PROBLEM

## USER-STATUS :

Description: WATER PROBLEM

Address: JALAPETA

City: Machilipatnam

State: Andhra Pradesh

Pincode: 521001

Status: RESOLVED

★ Rate the service: ★★★★★ SUBMIT RATING

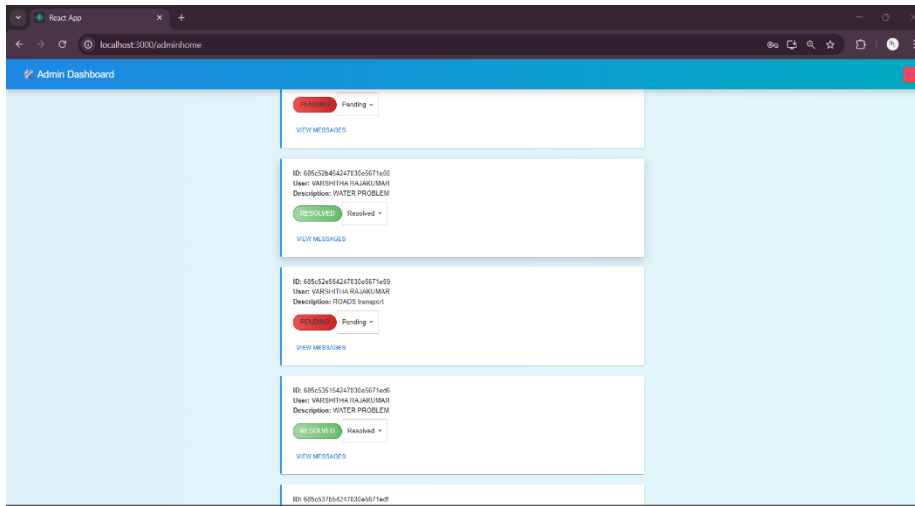
HIDE MESSAGES

user: hello...Cam you solve my issue as soon as possible 6/27/2025, 8:57:12 AM

THANKYOU....BYE

SEND

## ADMIN-DASHBAORD:



The UI of **ResolveNow** is built using **React.js**, styled with **Material UI** and **Bootstrap** for a responsive and clean layout.

### Key Screens:

- **Login/Signup:** Simple, validated forms with role selection
- **User Dashboard:** Submit complaints, track status, chat with agents
- **Admin Panel:** Assign complaints, manage users, update statuses
- **Agent Panel:** View assigned complaints, update status, chat
- **Feedback Form:** Users rate service after complaint is resolved

## 10. TESTING

### Testing Strategy

The testing approach for **ResolveNow** focused on verifying that all core functionalities worked as intended across different user roles (user, agent, admin). The strategy included:

- **Manual Testing** of user flows such as:
  - Registration & Login
  - Complaint Submission & Status Tracking
  - Agent-User Messaging
  - Admin Assignment & Status Update
  - Feedback submission after resolution
- **Positive and Negative Scenarios** were covered to ensure:
  - Proper field validations
  - Unauthorized access is restricted
  - Functional correctness under real conditions

### Tools Used

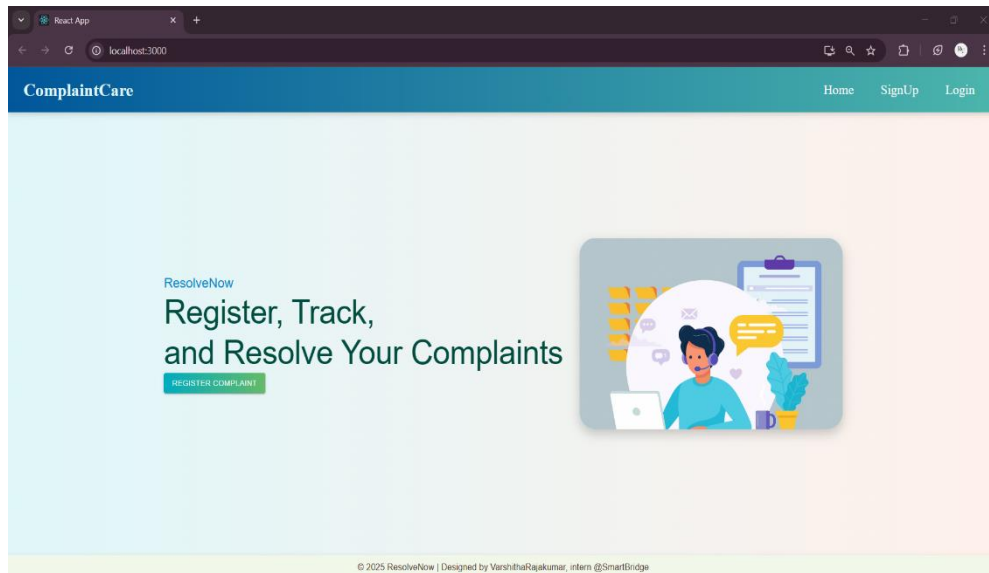
- **Postman** – For testing backend RESTful APIs (GET, POST, PATCH)



- **Browser DevTools** – For inspecting frontend behavior, component rendering, and network requests
- **React Developer Tools** – To test React components and props/state updates
- **Console Logging** – To debug application logic during development

## 11. SCREENSHOTS OR DEMO

**HOME :**



**SIGNUP :**

### Sign Up to Register Your Complaint

Full Name \*
 Varshitha Rajakumar

Email \*
 V1@gmail.com

Password \*
 ...

Mobile No \*
 1234567890

User Type \*
 User

REGISTER

Already have an account? [Login](#)

### Sign Up to Register Your Complaint

Full Name \*
 Nikhila

Email \*
 nna@gmail.com

Password \*
 ...

Mobile No \*
 0901234567

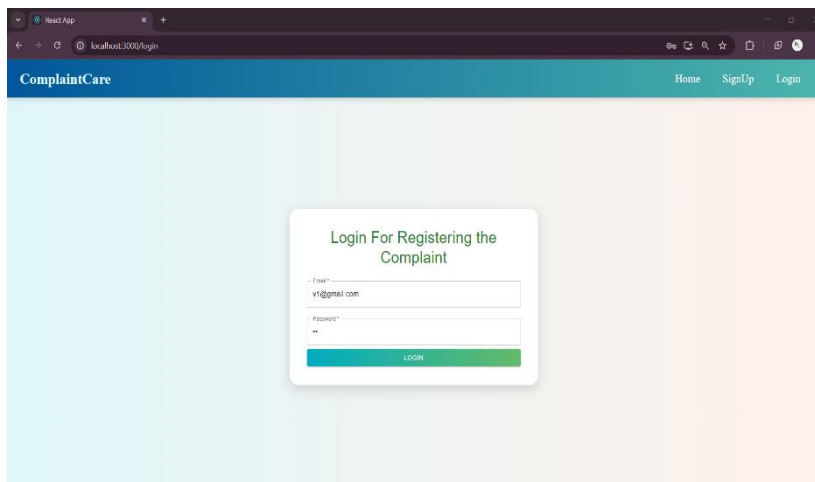
User Type \*
 Admin

REGISTER

Already have an account? [Login](#)

## LOGIN :

## USER-LOGIN



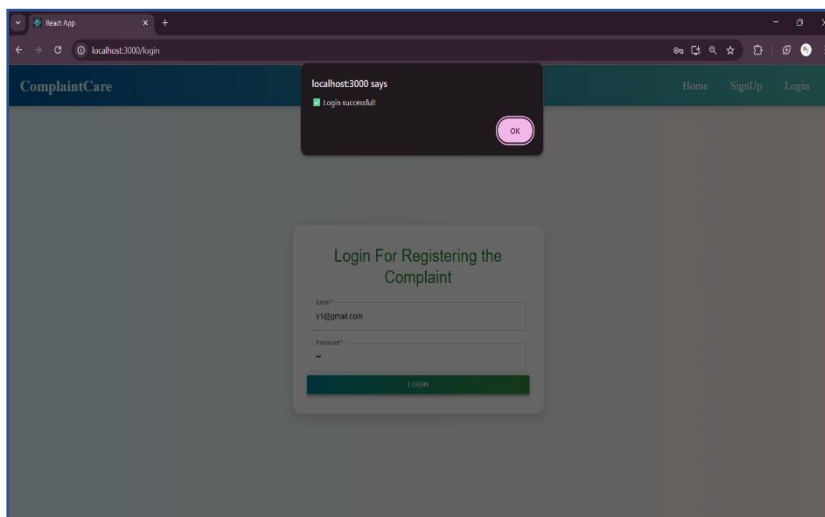
ComplaintCare Home SignUp Login

Login For Registering the Complaint

Email \*  
v1@gmail.com

Password \*  
\*\*

LOGIN



ComplaintCare Home SignUp Login

localhost:3000 says  
Login successful OK

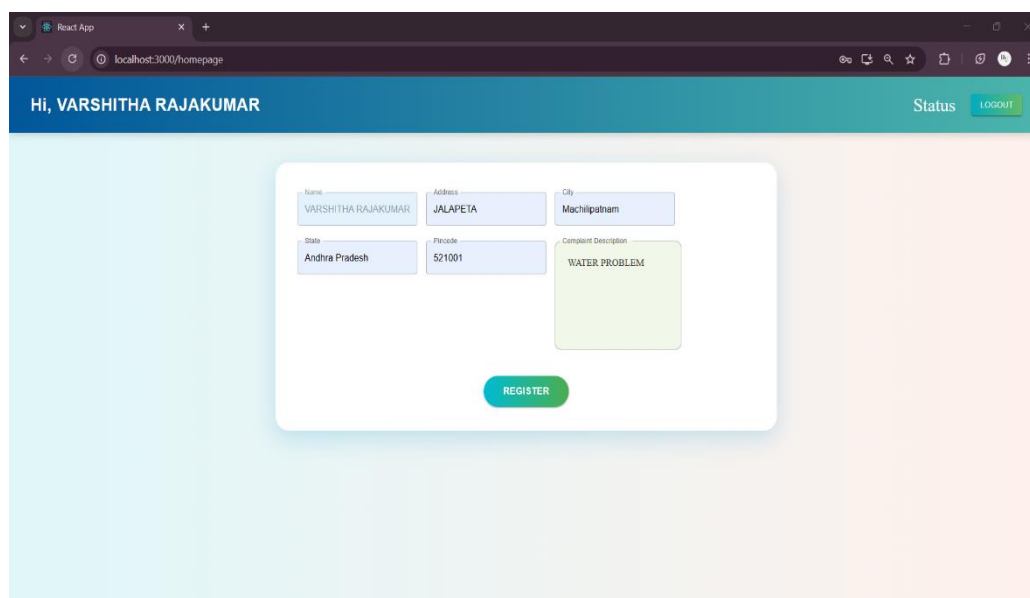
Login For Registering the Complaint

Email \*  
v1@gmail.com

Password \*  
\*\*

LOGIN

## USER DASHBOARD :



Hi, VARSHITHA RAJAKUMAR Status LOGOUT

Name  
VARSHITHA RAJAKUMAR

Address  
JALAPETA

City  
Machilipatnam

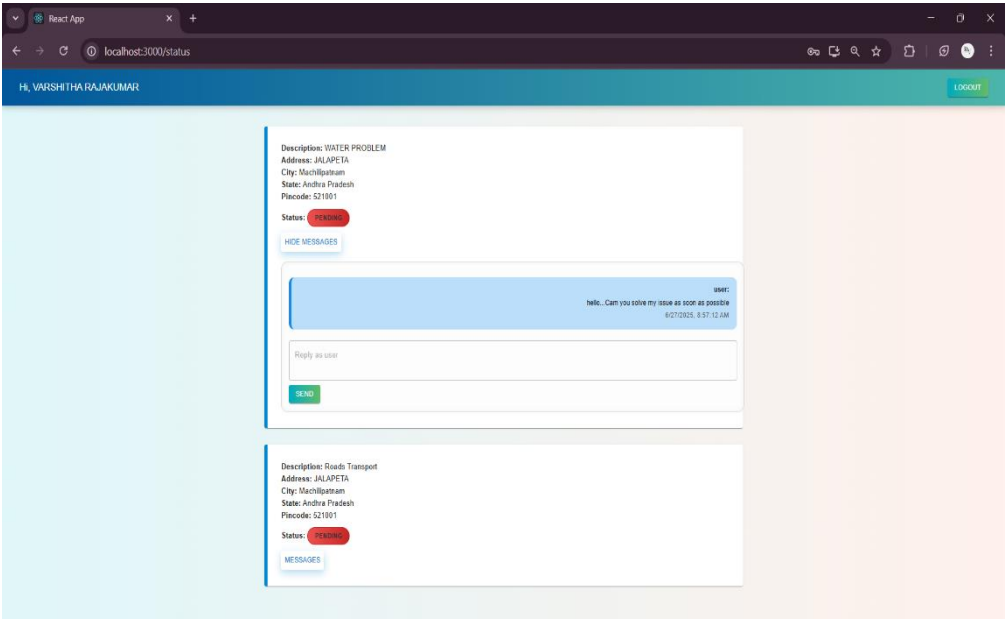
State  
Andhra Pradesh

Pincode  
521001

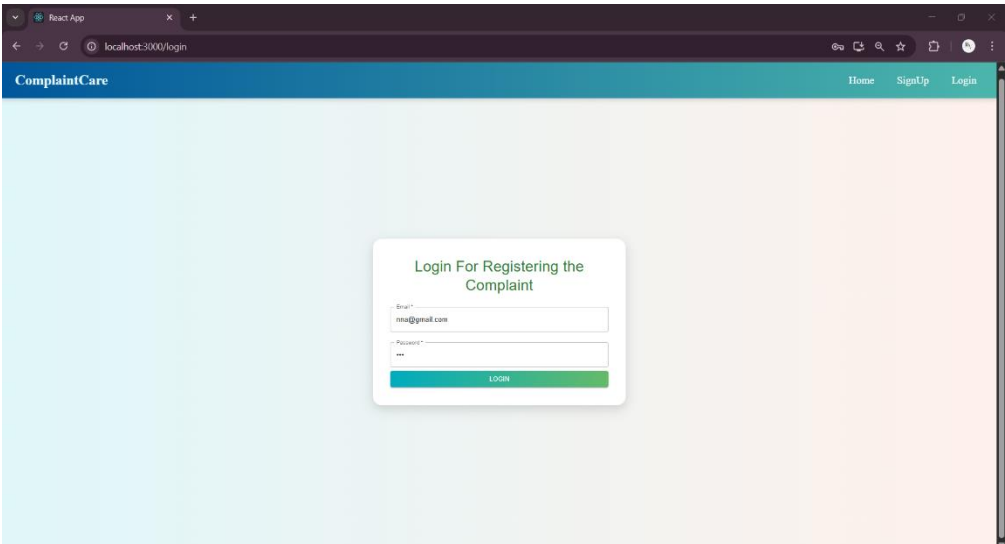
Complaint Description  
WATER PROBLEM

REGISTER

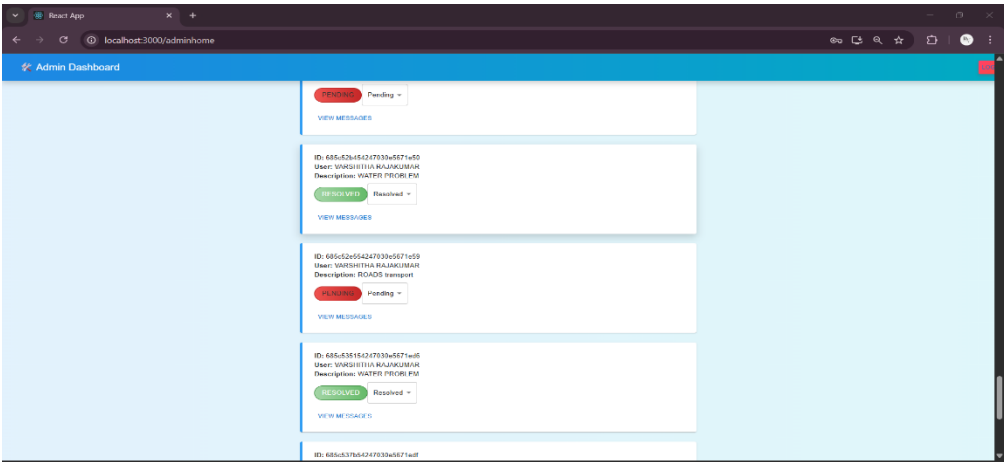
USER-STATUS:



ADMIN-LOGIN:



ADMIN-DASHBOARD:



SOLVING COMPLAINTS

ID: 685e0f6ddfa2d55f4cf94ad  
User: VARSHITHA RAJAKUMAR  
Description: Roads Transport

IN PROGRESS

In Progress

HIDE MESSAGES

user:  
requesting ypu to please visit our area  
6/27/2025, 9:14:04 AM

admin:  
can you give sometime to solve the issue  
6/27/2025, 9:17:09 AM

Type your message to user...

SEND

ID: 685e1396ddfa2d55f4cf954a  
User: VARSHITHA RAJAKUMAR  
Description: WATER PROBLEM

PENDING

Pending

VIEW MESSAGES

UPDATED COMPLAINT STATUS:

Description: Roads Transport  
Address: JALAPETA  
City: Machilipatnam  
State: Andhra Pradesh  
Pincode: 521001

Status: IN PROGRESS

HIDE MESSAGES

user:  
requesting ypu to please visit our area  
6/27/2025, 9:14:04 AM

admin:  
can you give sometime to solve the issue  
6/27/2025, 9:17:09 AM

okay...Thankyou

SEND

RATING :

Description: WATER PROBLEM  
Address: JALAPETA  
City: Machilipatnam  
State: Andhra Pradesh  
Pincode: 521001

Status: RESOLVED

★ Rate the service: ★★★★★

SUBMIT RATING

HIDE MESSAGES

user:  
hello...Cam you solve my issue as soon as possible  
6/27/2025, 8:57:12 AM

THANKYOU...BYE

SEND

**Demo Link:**

<https://drive.google.com/file/d/1WhxFHVgHrlz46jyluNsEOqS1nsG9it5-/view?usp=drivesdk>

## **12. KNOWN ISSUES**

- Minor delay in real-time chat messages under slow network conditions
- No visual alert if complaint form is submitted with empty fields
- Chat history not saved if the page is refreshed without submission
- UI layout may slightly break on smaller mobile screens
- Feedback form currently lacks input validation

## **13. FUTURE ENHANCEMENTS**

- Add **push notifications** for real-time status updates
- Implement **multi-language support** for wider accessibility
- Integrate **AI-based complaint categorization**
- Enable **file uploads** (images/videos) for complaint evidence
- Develop **analytics dashboard** for admin (reports, charts)
- Add **dark mode** and enhanced mobile responsiveness
- Improve **chat system** with message history and read receipts