# RESOLVENOW- A COMPLAINT SYSTEM USING MERN

## INTRODUCTION:

ResolveNow is a web-based platform designed to help users easily register and manage complaints online. It provides a simple and user-friendly interface for users to submit their issues, track the status, and receive updates. The system also helps administrators and concerned departments view, assign, and resolve complaints efficiently.

This project aims to improve communication between users and authorities, reduce manual work, and ensure timely resolutions. By using ResolveNow, the entire complaint process becomes faster, more transparent, and better organized.

The platform supports role-based access for users, admins, and agents, ensuring that each user has access to the features relevant to their role. It also maintains a proper complaint history, sends notifications for status updates, and allows for better monitoring and reporting. Overall, ResolveNow helps create a more responsive and efficient complaint management system suitable for institutions, municipalities, or service-based organizations.

## Key Features :

#### User Registration and Login :

- New users can securely sign up using their email and password.
- Existing users and admins can log in with authentication to access their respective dashboards.

### Complaint Submission :

- Users can file complaints by providing necessary details like title, description, category, and location.
- Users can also attach images or documents to support their complaint.

### Complaint Tracking :

- Each complaint has a unique ID for easy tracking.
- Users can log in anytime to view the current status and updates of their submitted complaints.

### Role-Based Access :

- Users, admins, and department staff have different permissions and dashboard views.

- This ensures data privacy and allows each role to focus on their specific tasks.

### Admin Dashboard

- Admins can view all complaints, filter them based on status or category, and assign them to relevant staff.
- The dashboard provides summary statistics to monitor system performance.

### Complaint Status Management :

- Admins or staff can update the status of each complaint as it progresses (e.g., In Progress, Resolved).
- Users are notified when a change is made, ensuring transparency in the process.

### Notifications and Alerts :

- Users receive email or system notifications when a complaint is registered, updated, or resolved.
- This keeps users informed and reduces the need for manual follow-ups.

### Complaint History :

- All complaints submitted by a user are stored and can be accessed from their dashboard.
- This helps maintain a record of grievances for future reference or analysis.

### Responsive Web Design :

- The application adapts well to all devices including mobiles, tablets, and desktops.
- Users can submit and track complaints on-the-go without requiring a dedicated app.

### Data Security :

- User information and complaint data are securely stored using best practices.
- Authentication and access control prevent unauthorized access or tampering.

### Feedback and Ratings :

- After a complaint is resolved, users can give feedback or rate the service quality.
- This feature helps improve accountability and gather user satisfaction metrics.

**ResolveNow** is an online complaint management system that allows users to easily register and manage complaints related to public services or institutional issues. It provides a user-friendly platform where individuals can report problems by filling in details like complaint type, description, and location, along with optional attachments such as images. This system helps eliminate the need for physical visits or paperwork, making the complaint process faster and more convenient.

The platform includes role-based access for three types of users: regular users, admins, and department staff. Users can submit and track their complaints, while admins and staff can view, assign, and resolve issues based on category or priority. Each role has its own dashboard, ensuring that everyone sees only the information relevant to them. This separation of responsibilities helps improve efficiency and keeps the system organized.

ResolveNow also includes features like real-time complaint tracking, status updates, and notifications. Users receive alerts when their complaint status changes, which builds trust and transparency in the system. Additionally, the platform maintains a complaint history for each user, helping them refer back to past issues and their resolutions. Admins can monitor performance through filters, search options, and dashboard analytics.

Overall, ResolveNow offers a complete solution to manage complaints in a digital format. It ensures faster response times, better accountability, and improved communication between users and authorities. With added features like feedback and rating systems, it also encourages continuous improvement in service quality. The project is ideal for municipalities, colleges, and organizations looking to simplify and modernize their complaint handling process.

## SCENARIO-BASED CASE STUDY :

## Scenario: Complaint about a Streetlight Not Working

1. **User Registration**
   - A resident, Anjali, signs up on the ResolveNow platform using her email and creates a secure password.
2. **Complaint Submission**
   - Anjali notices a non-functional streetlight in her area and submits a complaint by selecting the "Streetlight" category.
   - She adds a brief description, selects her location on the map, and attaches a photo.
3. **Complaint ID Generation & Acknowledgment**
   - The system generates a unique complaint ID and sends an acknowledgment to Anjali via email and dashboard notification.
4. **Admin Review & Assignment**
   - The admin logs in, reviews the complaint, and assigns it to the Electrical Department staff.

5. **Department Action**
    - The assigned staff marks the complaint status as “In Progress” after scheduling the inspection.
    - After fixing the issue, the status is updated to “Resolved.”
6. **User Notification**
    - Anjali receives notifications at each stage: when the complaint is assigned, when work begins, and when it is resolved.
7. **Feedback Collection**
    - Once resolved, Anjali is prompted to give feedback and rate the resolution process.
8. **Complaint History Maintenance**
    - Anjali can view the complaint and status anytime from her dashboard as part of her complaint history.
9. **Admin Reporting**
    - The admin uses the dashboard to view performance stats such as number of complaints resolved, pending, and average resolution time.
10. **Improved Service**

- The timely resolution and feedback help the system improve its service quality and build trust among users.

**TECHNICAL ARCHITECTURE :**

The technical architecture of **ResolveNow** is based on a robust and scalable MERN stack (MongoDB, Express.js, React.js, Node.js). The frontend is built using React.js for dynamic user interaction, while the backend uses Express.js with Node.js to handle server-side logic and API communication. MongoDB is used as the database for storing complaint and user data, and JSON Web Token (JWT) is used for secure authentication. The architecture ensures a smooth, secure, and responsive user experience across all roles—user, admin, and department staff.

## Frontend Technologies :

- Built using React.js to create a fast and interactive user interface.
- React Router DOM is used for seamless navigation between pages.
- Bootstrap or React-Bootstrap ensures responsive and mobile-friendly design.

## Backend Framework :

- Node.js and Express.js handle API requests, logic, and routing.
- Modular structure with clean endpoints for better maintenance and scalability.
- Middleware functions used for validation, authentication, and error handling.

**Database :**

- MongoDB is used to store user, complaint, and feedback data.
- Mongoose provides schema definition and easy interaction with MongoDB.
- Collections are designed to support flexible, scalable data storage.

**Authentication :**

- JWT (JSON Web Token) ensures secure and stateless authentication.
- Bcrypt is used to hash passwords for enhanced security.
- Role-based access control restricts users, admins, and staff to their permitted actions.

**Admin Panel and Governance :**

- Admin dashboard allows viewing, filtering, and assigning complaints.
- Staff can manage complaint statuses and track department performance.
- Admin can monitor system statistics, user feedback, and reports.

**Scalability and Performance :**

- Modular code structure and RESTful APIs allow easy scaling of features.
- Asynchronous operations and optimized database queries improve performance.
- Can handle growing users and complaints without affecting speed.

**Time Management and Scheduling :**

- Complaints include timestamps to track submission and resolution time.
- Admins and staff can prioritize complaints based on urgency and date.
- Helps in analyzing service response times and improving turnaround.

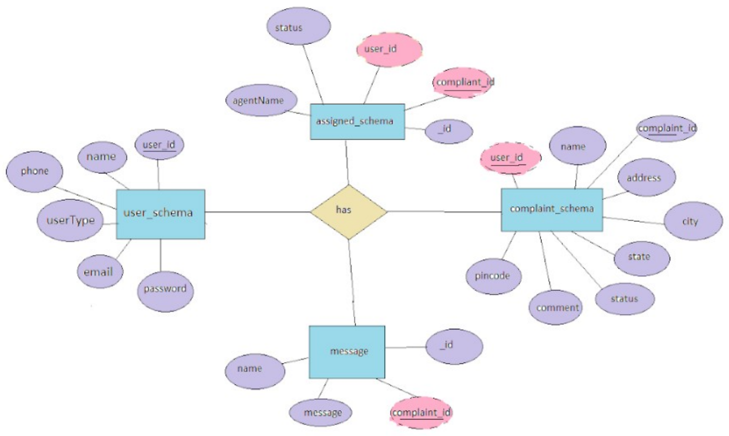
**Security Features :**

- Secure password hashing, token-based login, and input sanitization.
- Protected API routes to prevent unauthorized access.
- Environment variables hide sensitive credentials and keys.

**Notifications and Reminders :**

- Users receive notifications on complaint updates via email or dashboard.
- Admin/staff alerted on new or pending complaints to ensure timely action.
- Feedback prompts sent to users once a complaint is resolved

## ER DIAGRAM :



This is the er diagram of the project which shows the relationship between user and agent

It shows how user which have required fields can raise a complaint by fillings required fields.
It illustrates how these entities relate to each other, helping us understand the underlying database structure and the flow of information within the app. He / She can also communicate with the agent with chat window which follows the message schema which uses userId and complaintId from other schemas

## PRE REQUISITES :

## NODE.JS AND NPM:

- Node.js is a JavaScript runtime that allows you to run JavaScript code on the server-side. It provides a scalable platform for network applications.
- npm (Node Package Manager) is required to install libraries and manage dependencies.
- Download Node.js: Node.js Download
- Installation instructions: Installation Guide
- Run npm init to set up the project and create a package.json file.

## EXPRESS.JS:

- Express.js is a web application framework for Node.js that helps you build APIs and web applications with features like routing and middleware.
- Install Express.js to manage backend routing and API endpoints.
- Install Express:

- Run npm install express

## MONGODB:

- MongoDB is a NoSQL database that stores data in a JSON-like format, making it suitable for storing data like user profiles, doctor details, and appointments.
- Set up a MongoDB database for your application to store data.
- Download MongoDB: MongoDB Download
- Installation instructions: MongoDB Installation Guide

## MOMENT.JS:

- Moment.js is a JavaScript package for handling date and time operations, allowing easy manipulation and formatting.
- Install Moment.js for managing date-related tasks, such as appointment scheduling.

## REACT.JS:

- React.js is a popular JavaScript library for building interactive and reusable user interfaces. It enables the development of dynamic web applications.
- Install React.js to build the frontend for your application.
- React.js Documentation: Create a New React App

## HTML, CSS, AND JAVASCRIPT:

- Basic knowledge of HTML, CSS, and JavaScript is essential to structure, style, and add interactivity to the user interface.

## DATABASE CONNECTIVITY (MONGOOSE):

- Use Mongoose, an Object-Document Mapping (ODM) library, to connect your Node.js backend to
- MongoDB for managing CRUD operations.
- Learn Database Connectivity: Node.js + Mongoose + MongoDB

## FRONT-END FRAMEWORKS AND LIBRARIES:

- React.js will handle the client-side interface for managing complaints, viewing complaint statuses, and providing an feedback.
- You may use Material UI and Bootstrap to enhance the look and feel of the application.

## SETUP AND INSTALLATION INSTRUCTIONS :

## CLONE THE PROJECT REPOSITORY:

- Download the project files from GitHub or clone the repository using Git.
- **INSTALL DEPENDENCIES:**
- Navigate to the frontend and backend directories and install all required dependencies for both parts of the application.
- Frontend:
- Navigate to the frontend directory and run npm install.
- Backend:
- Navigate to the backend directory and run npm install.

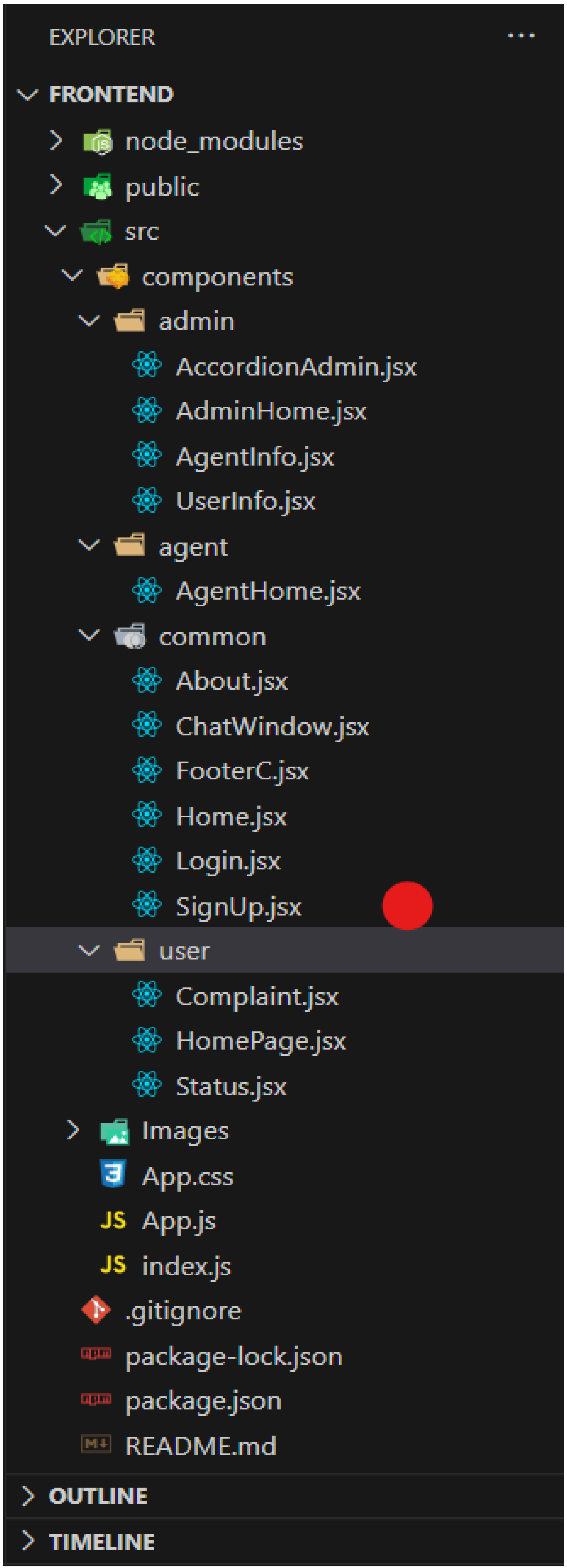
### START THE DEVELOPMENT SERVER:

- After installing the dependencies, start the development server for both frontend and backend.
- Frontend will run on http://localhost:3000.
- Backend will run on http://localhost:8001 or the specified port.

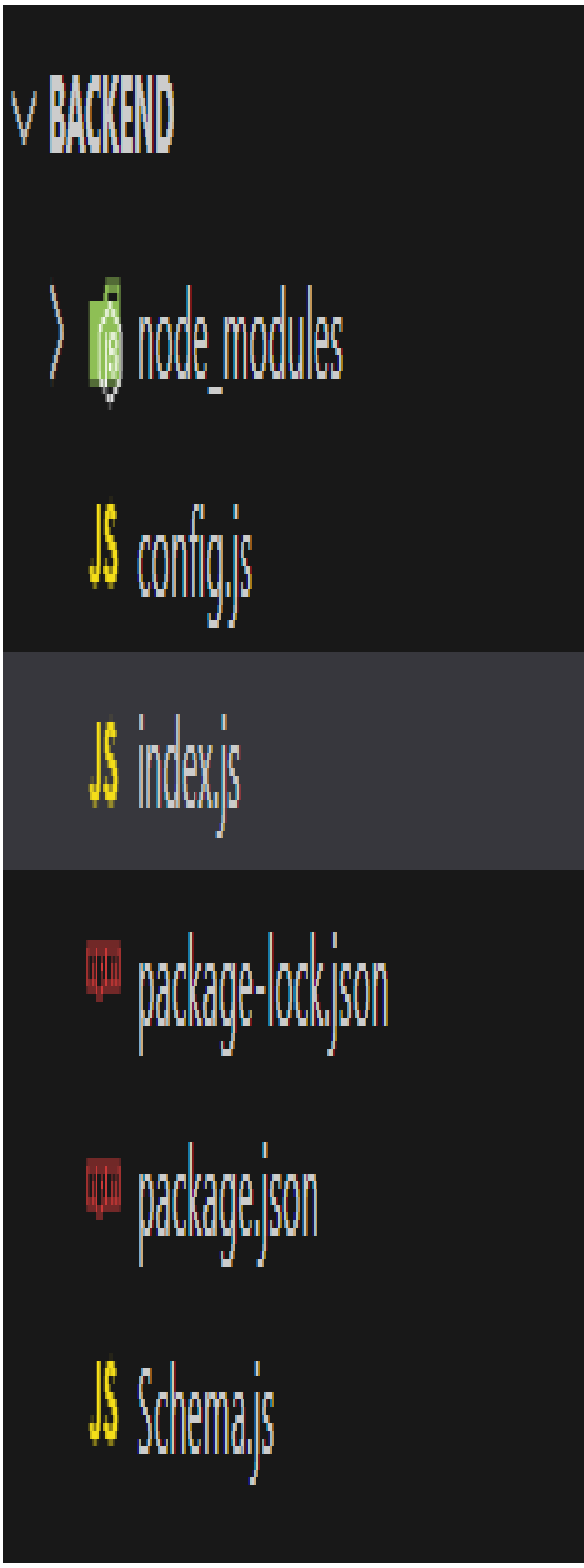
### ACCESS THE APPLICATION:

- After running the servers, access the Doctor Appointment Webpage in your browser at http://localhost:3000 for the frontend interface and http://localhost:8001 for backend API services.

**PROJECT STRUCTURE :**



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



```
v 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

## APPLICATION FLOW:

## Online Complaint Registration and Management System :

### Customer/Ordinary User:

Role: Create and manage complaints, interact with agents, and manage profile information.

### Registration and Login:

- Create an account by providing necessary information such as email and password.
- Log in using the registered credentials.

### Complaint Submission:

- Fill out the complaint form with details of the issue, including description, contact information, and relevant attachments.
- Submit the complaint for processing.

### Status Tracking:

- View the status of submitted complaints in the dashboard or status section.
- Receive real-time updates on the progress of complaints.

### Interaction with Agents:

- Connect with assigned agents directly using the built-in messaging feature.
- Discuss complaints further and provide additional information or clarification.

### 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.

### Registration and Login:

- Create an account using email and password.
- Log in using the registered credentials.

### Complaint Management:

- Access the dashboard to view and manage complaints assigned by the admin.
- Communicate with customers regarding their complaints through the chat window.

### Status Update:

- Change the status of complaints based on resolution or progress.
- Provide updates to customers regarding the status of their complaints.

### 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.

### Management and Monitoring:

- Monitor and moderate all complaints submitted by users.
- Assign complaints to agents based on workload and expertise.

### Complaint Assignment:

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

### User and Agent Management:

- Manage user and agent accounts, including registration, login, and profile information.

- Enforce platform policies, terms of service, and privacy regulations.

**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.

**SETUP AND CONFIGURATION :**

- Setting up the Online Complaint Management system Webpage involves configuring both the Frontend (React.js) and Backend(Node.js, Express.js, MongoDB) to ensure the application runs smoothly. Below are the steps to set up and configure the environment for your project.

**FRONTEND CONFIGURATION :**

**INSTALLATION :**

**Clone the Repository:**

- Clone the project from GitHub to your local machine:
- bash
- Copy code
- git clone <your-repository-url>
- Replace <your-repository-url> with the URL of your project repository.

**Navigate to Frontend Directory:**

- After cloning, navigate to the frontend folder where the React.js app is located:
- bash
- Copy code
- cd complaintregistry/frontend

**Install Dependencies:**

- Use npm (Node Package Manager) to install the necessary dependencies:

- bash
- Copy code
- npm install
- This will install all the required libraries, such as React, React Router, Ant Design, and others.

**Run the React Development Server:**

- To start the frontend server and run the React application:
- bash
- Copy code
- npm start
- The application will be available at http://localhost:3000 in your browser

**BACKEND CONFIGURATION :**

**INSTALLATION :**

**Navigate to Backend Directory:**

- Move to the backend folder of your project:
- bash
- Copy code
- cd complaintregistry/backend
- Install Dependencies:
- Install the necessary backend dependencies using npm:
- bash
- Copy code
- npm install

- This will install all required libraries for Node.js and Express.js, such as express, mongoose, bcryptjs, jsonwebtoken, etc.

**Configure MongoDB :**

- Ensure you have MongoDB installed on your local machine or use a cloud-based MongoDB service like MongoDB Atlas.
- In the backend, open the configuration file (e.g., config/db.js or .env) and configure the connection URL for MongoDB:
- bash
- Copy code
- MONGO_URI=mongodb://localhost:27017/complaintregistry
- If you are using MongoDB Atlas, replace the localhost part with your MongoDB Atlas connection string.

**Set Up Environment Variables:**

- Create a .env file in the backend directory to store environment-specific variables such as:
- bash
- Copy code
- PORT=5000
- JWT_SECRET=your_jwt_secret
- Make sure to replace your_jwt_secret with a strong secret key for JWT authentication.

**Run the Backend Server:**

- Start the backend server by running:
- bash
- Copy code
- npm start
- The backend server will be running at http://localhost:5000

## DATABASE CONFIGURATION (MONGODB) :

### Install MongoDB (Local Installation):

- If you are using a local MongoDB instance, download and install it from the official MongoDB website:Download MongoDB.

### Set Up MongoDB Database:

- After installation, start the MongoDB service:
- bash
- Copy code
- mongod
- This will run MongoDB on the default port 27017.

### MongoDB Atlas (Cloud-based MongoDB):

- If you prefer to use MongoDB Atlas, create an account on MongoDB Atlas and create a cluster.
- Once the cluster is set up, get the connection string and replace it in the backend’s .env file:
- bash
- Copy code
- MONGO_URI=<your-mongodb-atlas-connection-string>

## FINAL CONFIGURATION & RUNNING THE APP

### Run Both Servers:

- The React frontend and Node.js backend servers should run simultaneously for the app to function correctly.
- You can open two terminal windows or use tools concurrently to run both servers together.
- To install concurrently, run:
- bash

- Copy code
- npm install concurrently --save-dev

In your package.json file, add a script to run both servers:

json

Copy code

"scripts": {

 "start": "concurrently \"npm run server\" \"npm run client\"",

 "server": "node backend/server.js",

 "client": "npm start --prefix frontend"

}

**Start Both Servers:**

- Now you can run the following command in the root directory to start both the backend and frontend:
- bash
- Copy code
- npm start
- The backend will be available at http://localhost:5000, and the frontend at http://localhost:3000.

**VERIFYING THE APP :**

**Check Frontend:**

- Open your browser and go to http://localhost:3000. The React.js application should load with the list of users,complaints,agents.

**Check Backend:**

- Open Postman or any API testing tool to verify if the backend endpoints are working correctly, such as user login, registration, and complaint creation.

## ADDITIONAL SETUP :

- Version Control:
- If you haven’t already done so, initialise a Git repository in the root of your project:
- bash
- Copy code
- git init

## Add your project files and commit them:

- bash
- Copy code
- git add .
- git commit -m "Initial commit"
- Deployment (Optional):
- If you want to deploy your app, consider using cloud platforms like Heroku, AWS, or Vercel for frontend hosting and backend API deployment.

## FOLDER SETUP :

- The folder structure for your Resolve now online complaint management system Webpage project will include separate folders for the frontend and backend components to keep the code organised and modular. Here’s how to set it up:

## PROJECT ROOT STRUCTURE :

## Create the Main Folders:

- In your project’s root directory, create two main folders: frontend and backend.
- plaintext
- Copy code
- project-root/

- ├── frontend/
- └── backend/

## BACKEND SETUP :

- Install Necessary Packages in the Backend Folder:
- Navigate to the backend folder and install the following essential packages:
- plaintext
- Copy code
- backend/
- ├── config/
- ├── controllers/
- ├── models/

  ├── routes/
- ├── middleware/
- ├── uploads/
- ├── server.js
- ├── .env

## Packages to Install :

- cors: To enable cross-origin requests.
- bcryptjs: For securely hashing user passwords.
- express: A lightweight framework to handle server-side routing and API management.
- dotenv: For loading environment variables.
- mongoose: To connect and interact with MongoDB.
- multer: To handle file uploads.

- nodemon: A utility to auto-restart the server upon code changes (for development).
- jsonwebtoken: To manage secure, stateless user authentication.
- Installation Commands
- Run these commands in the backend folder:
- bash
- Copy code
- npm init -y
- npm install cors bcryptjs express dotenv mongoose multer jsonwebtoken
- npm install --save-dev nodemon

**FRONTEND SETUP :**

- React Project Initialization:
- Navigate to the frontend folder and initialise a new React application:
- plaintext
- Copy code
- frontend/

```
├── public/
├── src/
│ ├── components/
│ ├── pages/
│ ├── services/
│ └── App.js
├── .env
└── package.json
```

- Setting Up the Frontend Project

- In the frontend folder, run the following commands to set up and install any initial dependencies:
- bash

## PROJECT FLOW :

## Project Demo :

- Before diving into development, you can view a demo of the project to understand its functionality and user interactions.
- Project FlowDemo Video :

  https://drive.google.com/file/d/1OIlwCd2_HQeON0BMDgiypDDdh36ibkjA/view?usp=sharing
- Project Code Repository :

  https://github.com/sumathi255/Resolve-Now.git
- The source code for this project can be accessed and cloned from GitHub, providing a base structure and example code for further customization and understanding

### MILESTONE 1:PROJECT SETUP AND CONFIGURATION :

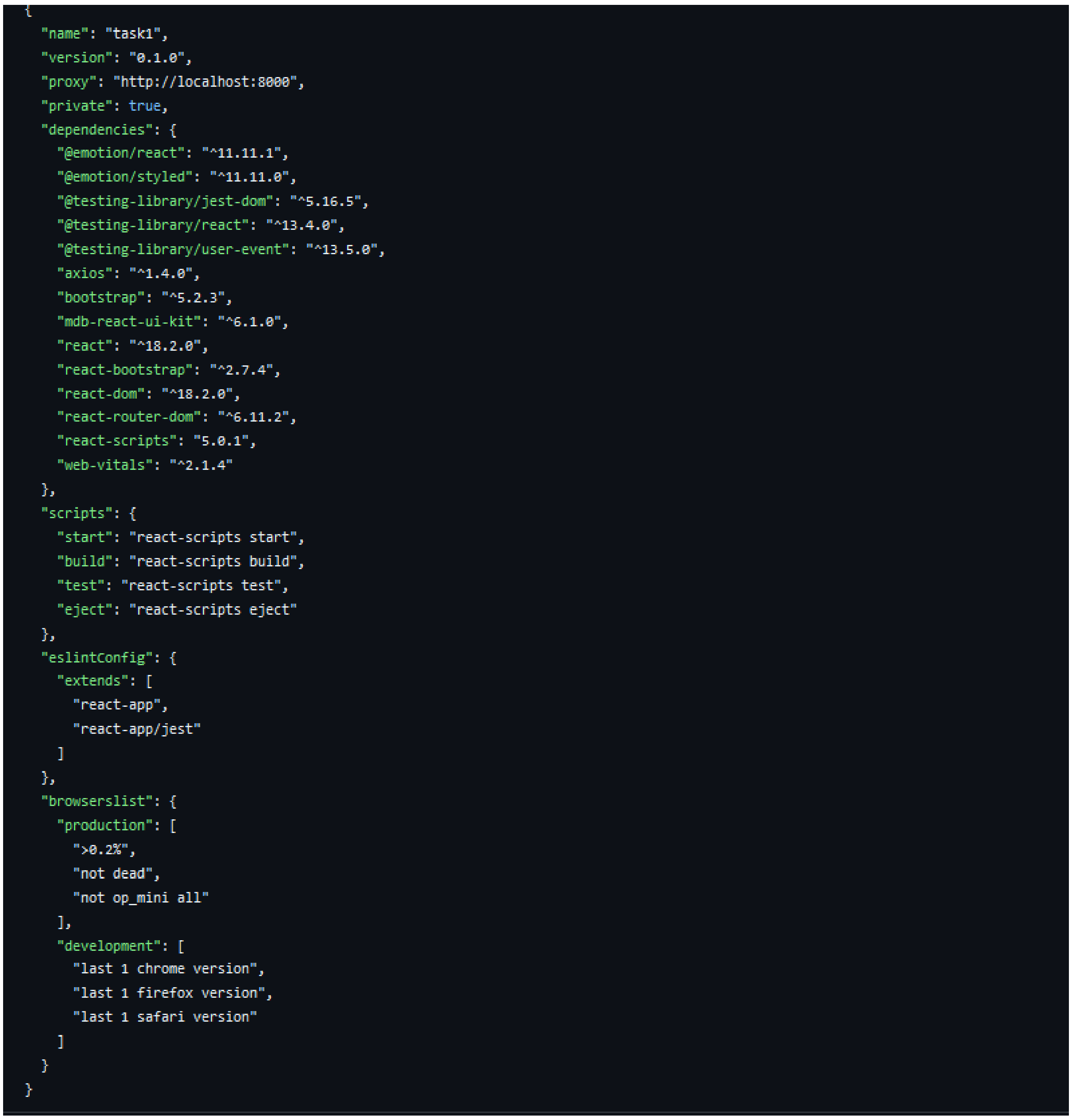
- Setting up a structured environment is crucial for any application.By organising the project into separate folders for the frontend and backend,we ensure that the codebase is manageable and scalable.

Create project folders and files:

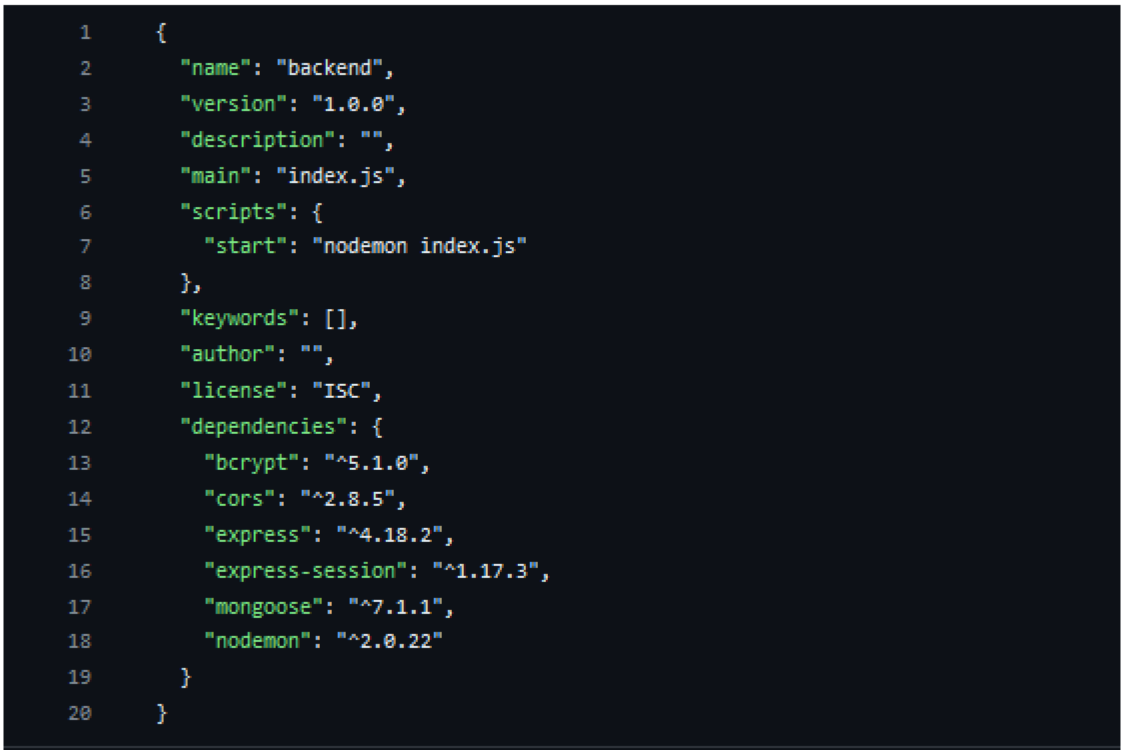
Now, firstly create the folders for frontend and backend to write the respective code and install the essential libraries.

•Client folders.

•Server folders



```json
{
  "name": "task1",
  "version": "0.1.0",
  "proxy": "http://localhost:8000",
  "private": true,
  "dependencies": {
    "@emotion/react": "^11.11.1",
    "@emotion/styled": "^11.11.0",
    "@testing-library/jest-dom": "^5.16.5",
    "@testing-library/react": "^13.4.0",
    "@testing-library/user-event": "^13.5.0",
    "axios": "^1.4.0",
    "bootstrap": "^5.2.3",
    "mdb-react-ui-kit": "^6.1.0",
    "react": "^18.2.0",
    "react-bootstrap": "^2.7.4",
    "react-dom": "^18.2.0",
    "react-router-dom": "^6.11.2",
    "react-scripts": "5.0.1",
    "web-vitals": "^2.1.4"
  },
  "scripts": {
    "start": "react-scripts start",
    "build": "react-scripts build",
    "test": "react-scripts test",
    "eject": "react-scripts eject"
  },
  "eslintConfig": {
    "extends": [
      "react-app",
      "react-app/jest"
    ]
  },
  "browserslist": {
    "production": [
      ">0.2%",
      "not dead",
      "not op_mini all"
    ],
    "development": [
      "last 1 chrome version",
      "last 1 firefox version",
      "last 1 safari version"
    ]
  }
}
```



```json
{
  "name": "backend",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "start": "nodemon index.js"
  },
  "keywords": [],
  "author": "",
  "license": "ISC",
  "dependencies": {
    "bcrypt": "^5.1.0",
    "cors": "^2.8.5",
    "express": "^4.18.2",
    "express-session": "^1.17.3",
    "mongoose": "^7.1.1",
    "nodemon": "^2.0.22"
  }
}
```

## PROJECT FOLDERS:

- Frontend Folder: Contains all code related to the user interface, written in JavaScript using frameworks and libraries like React, Material UI, and Bootstrap. This setup helps maintain a clear boundary between UI logic and server logic.
- Backend Folder: Manages the server, API routes, and database interactions, typically handled through Node.js and Express.js. Using separate folders enables a modular structure, allowing changes in one area without affecting the other.

## LIBRARY AND TOOL INSTALLATION:

### Backend Libraries:

- Node.js: Provides a runtime environment to run JavaScript code on the server side.
- MongoDB: A NoSQL database, perfect for flexible and schema-less data storage, ideal for applications needing frequent updates and various data types.
- Bcrypt: Encrypts passwords for secure authentication, helping protect user data from potential breaches.
- Body-parser: Parses incoming request bodies, making it easy to access data in various formats like JSON.
- Frontend Libraries:
- React.js: Manages component-based UI creation, providing the flexibility to build reusable UI components.
- Material UI & Bootstrap: Provides styling frameworks, ensuring a consistent, responsive, and visually appealing design.
- Axios: Facilitates easy HTTP requests, allowing the frontend to communicate with the backend effectively.

## MILESTONE 2: BACKEND DEVELOPMENT :

The backend forms the core logic and data management for the project.A well-structured backend ensures efficient data handling ,security and scalability.

### 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.

## MILESTONE 3: DATABASE DEVELOPMENT

**User Schema:**

- The user schema defines the structure of user data stored in the database. It includes fields such as name, email, password, phone, and userType.

- Each user must provide a name, email, password, phone number, and userType (e.g., customer, agent, admin).

- User data is stored in the "user_Schema" collection in the MongoDB database.

**Complaint Schema:**

- The complaint schema specifies the format of complaint data registered by users.

- It contains fields like userId, name, address, city, state, pincode, comment, and status.

- Complaints are associated with users through the userId field, and each complaint must have a name, address, city, state, pincode, comment, and status.

- Complaint data is stored in the "complaint_schema" collection in the MongoDB database.

**Assigned Complaint Schema:**

- The assigned complaint schema defines how complaints are assigned to agents for resolution.

- It includes fields such as agentId, complaintId, status, and agentName.

- Each assigned complaint is linked to a specific agent (identified by agentId) and complaint (identified by complaintId).

- The status field indicates the current status of the assigned complaint.

- Assigned complaint data is stored in the "assigned_complaint" collection in the MongoDB database.

**Chat Window Schema:**

- The chat window schema governs the structure of messages exchanged between users and agents regarding specific complaints.
- It comprises fields like name, message, and complaintId.
- Messages are associated with a complaint through the complaintId field, allowing for easy tracking and retrieval of chat history for each complaint.
- Message data is stored in the "message" collection in the MongoDB database

## MILESTONE 4: FRONTEND DEVELOPMENT :

**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.

**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.

**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.

## MILESTONE 5: PROJECT IMPLEMENTATION AND TESTING :

- After completing development,running a final set of tests is crucial for identifying any bugs or issues.

## VERIFY FUNCTIONALITY :

Running the entire application ensures that each part(frontend,backend,database) works cohesively.Testing various user flows help confirm that all processes are functioning as intended.

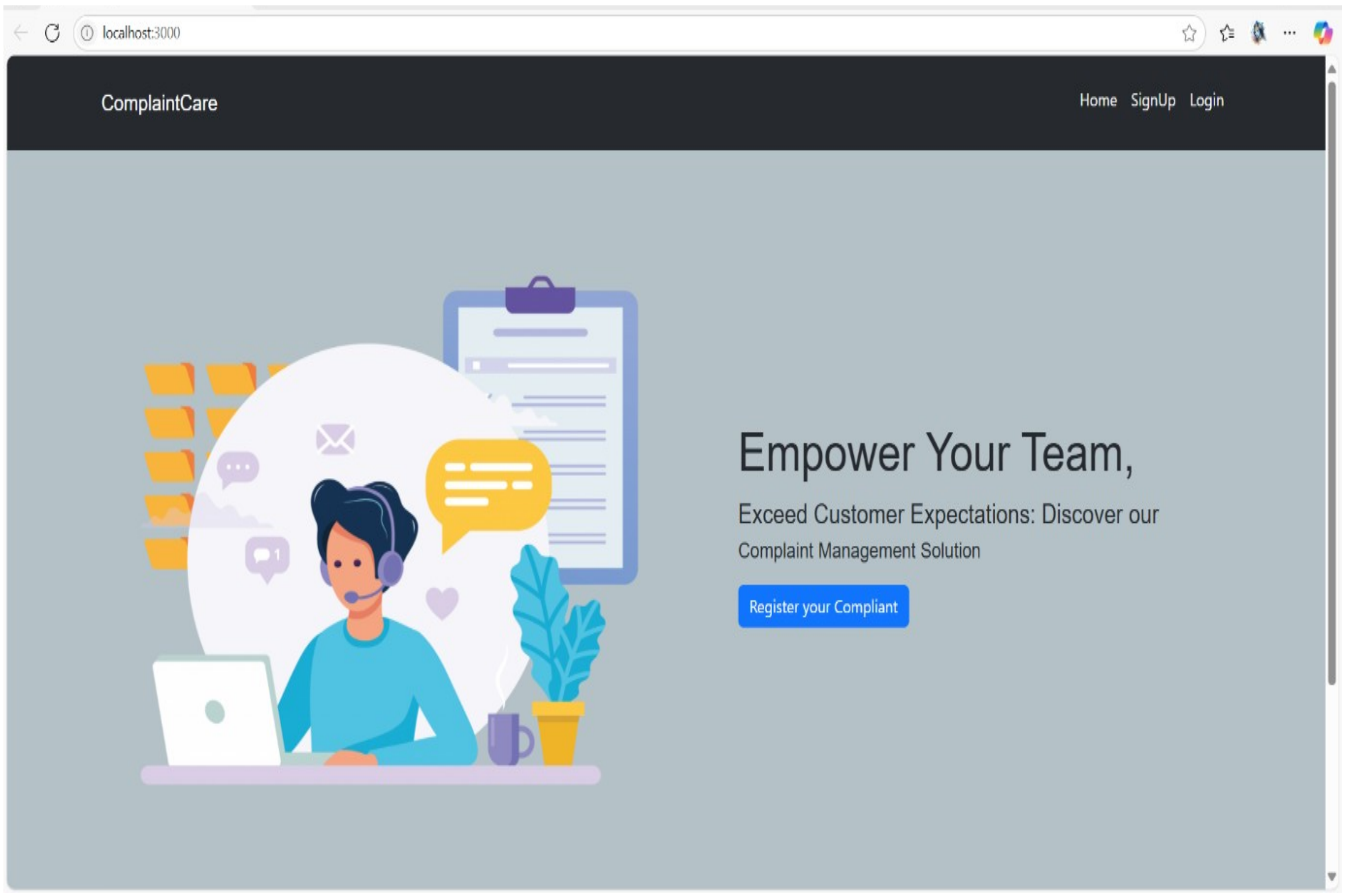
## USER INTERFACE ELEMENTS :

Testing the UI includes verifying the look and feel of each page-landing,login,registration and dashboards for different user types.

Ensuring responsive design and usability across devices and screen sizes is also essential FINAL DEPLOYMENT.

Once testing is complete,the application can be deployed to a production server,making it accessible to end-users.

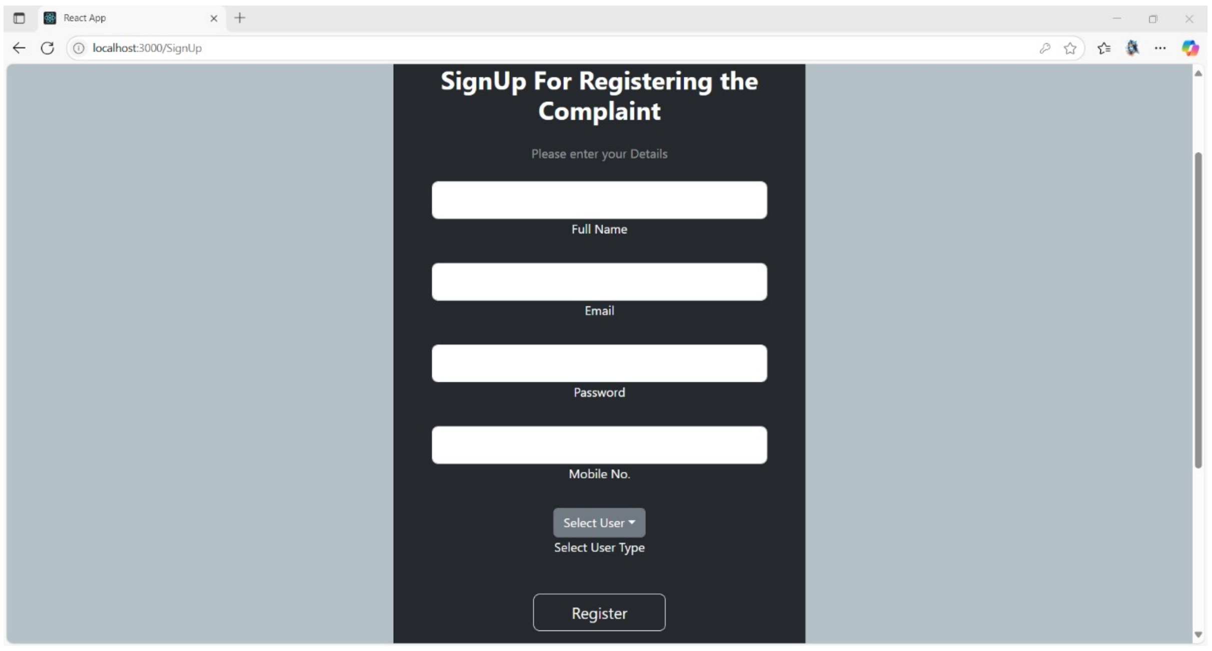
## Home PAGE :



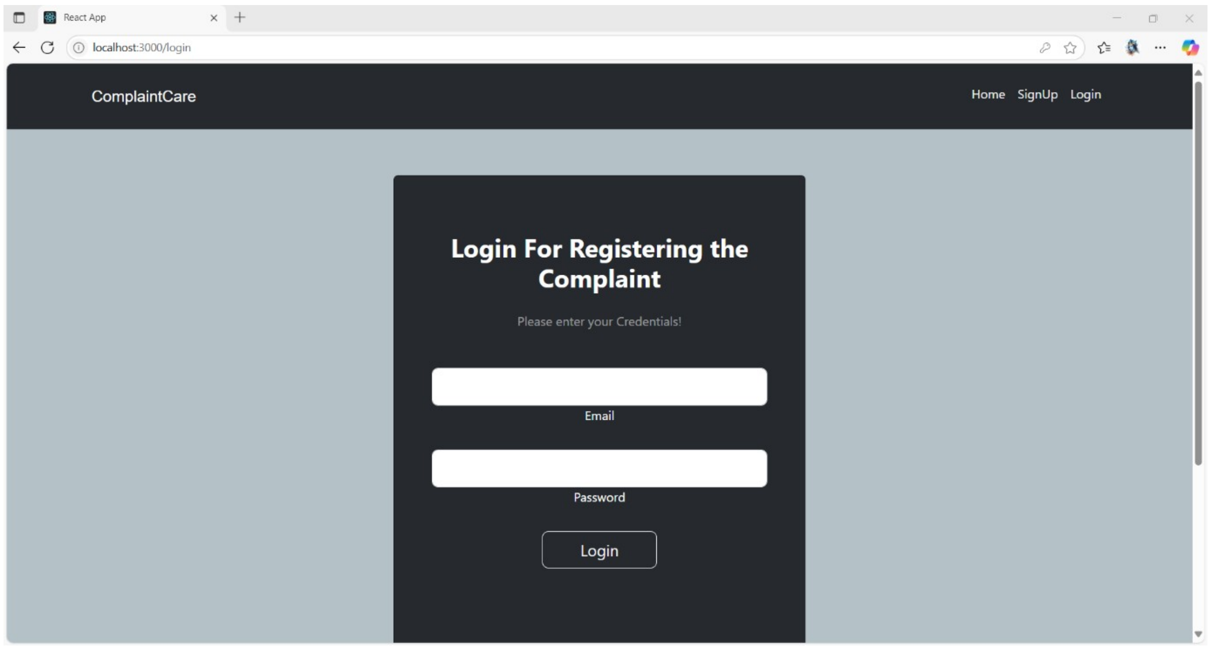
Empower Your Team,

Exceed Customer Expectations: Discover our Complaint Management Solution

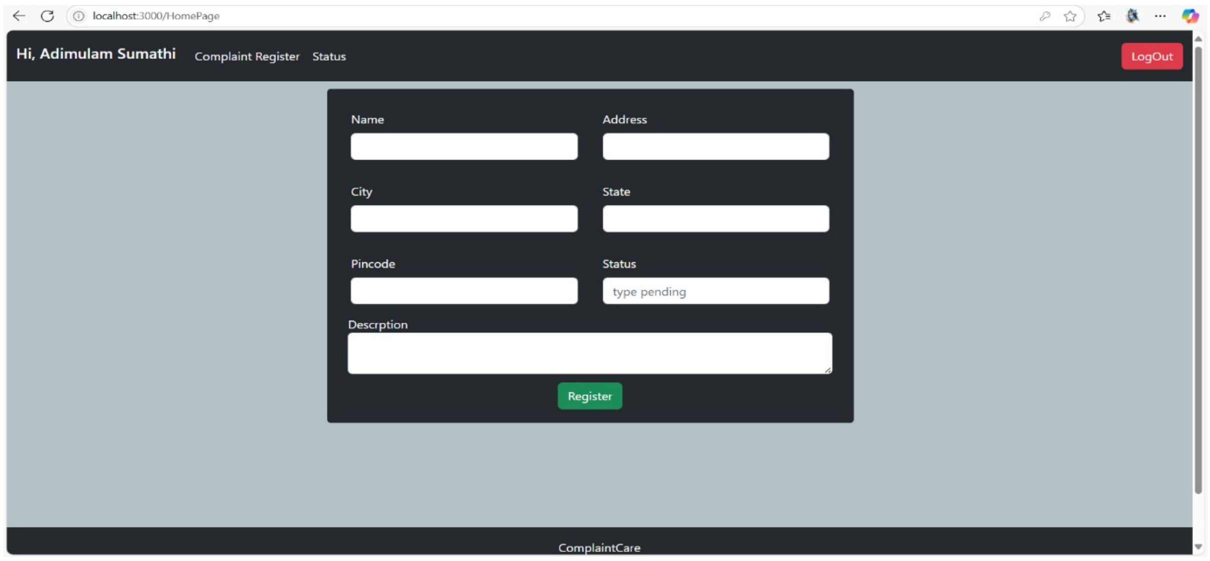
Register Page :



Login Page :



User Dashboard:



localhost:3000/HomePage

Hi, Adimulam Sumathi Complaint Register Status LogOut

Name

Address

City

State

Pincode

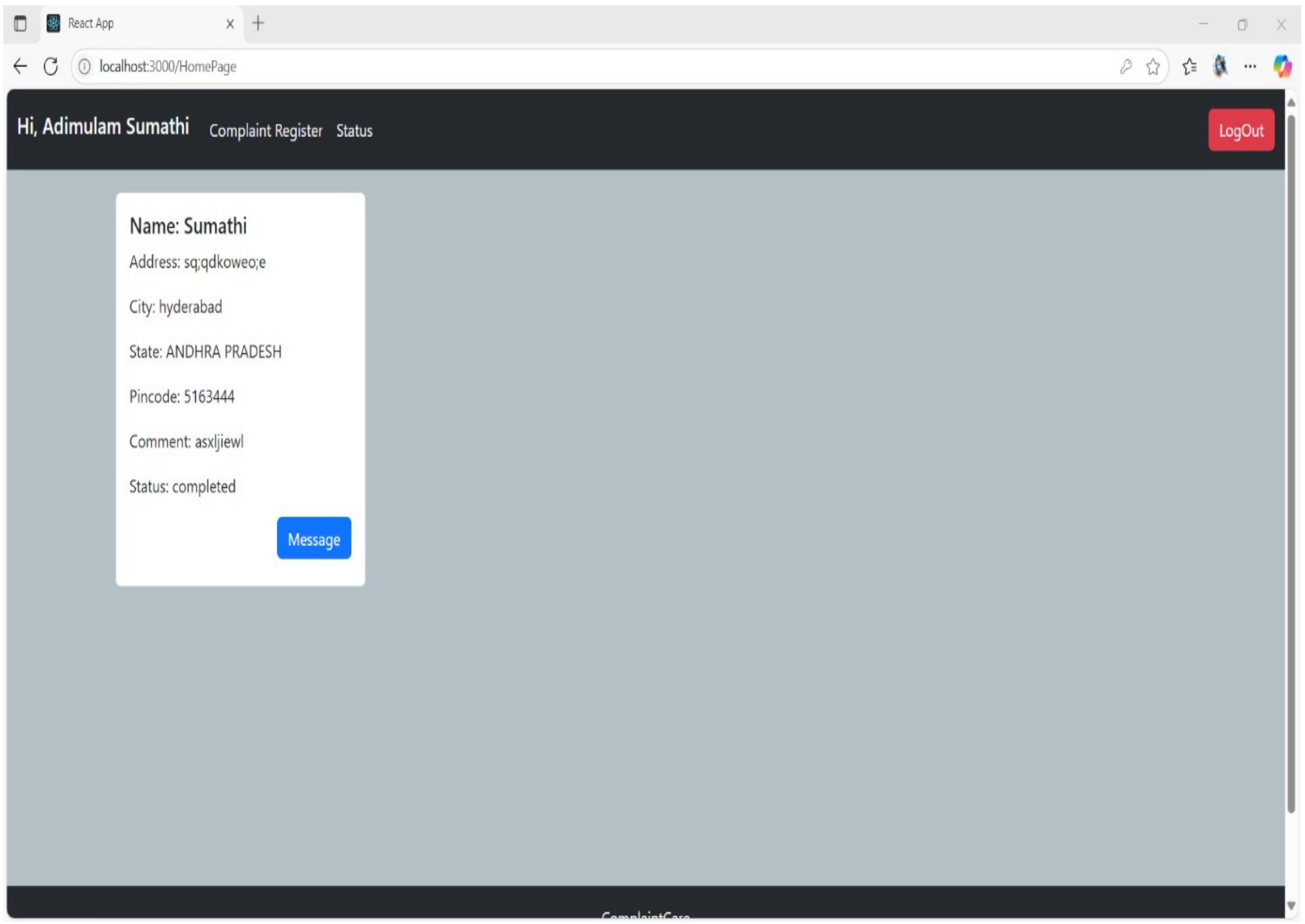
Status

type pending

Descrption

Register

ComplaintCare



React App

localhost:3000/HomePage

Hi, Adimulam Sumathi Complaint Register Status LogOut

Name: Sumathi

Address: sq;qdkoweo;e

City: hyderabad

State: ANDHRA PRADESH

Pincode: 5163444

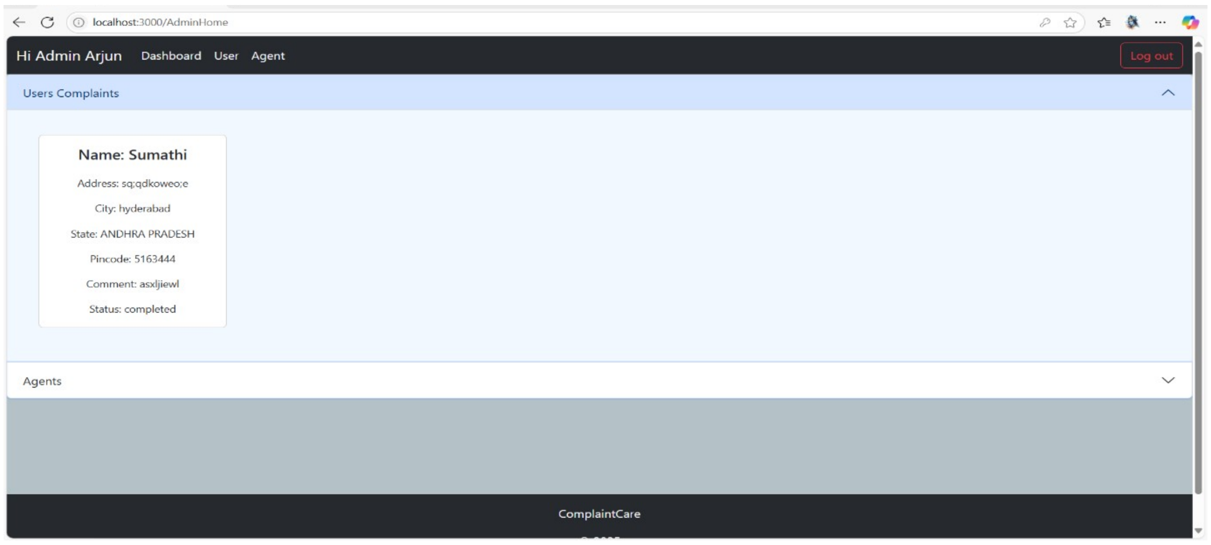
Comment: asxljiewl

Status: completed

Message

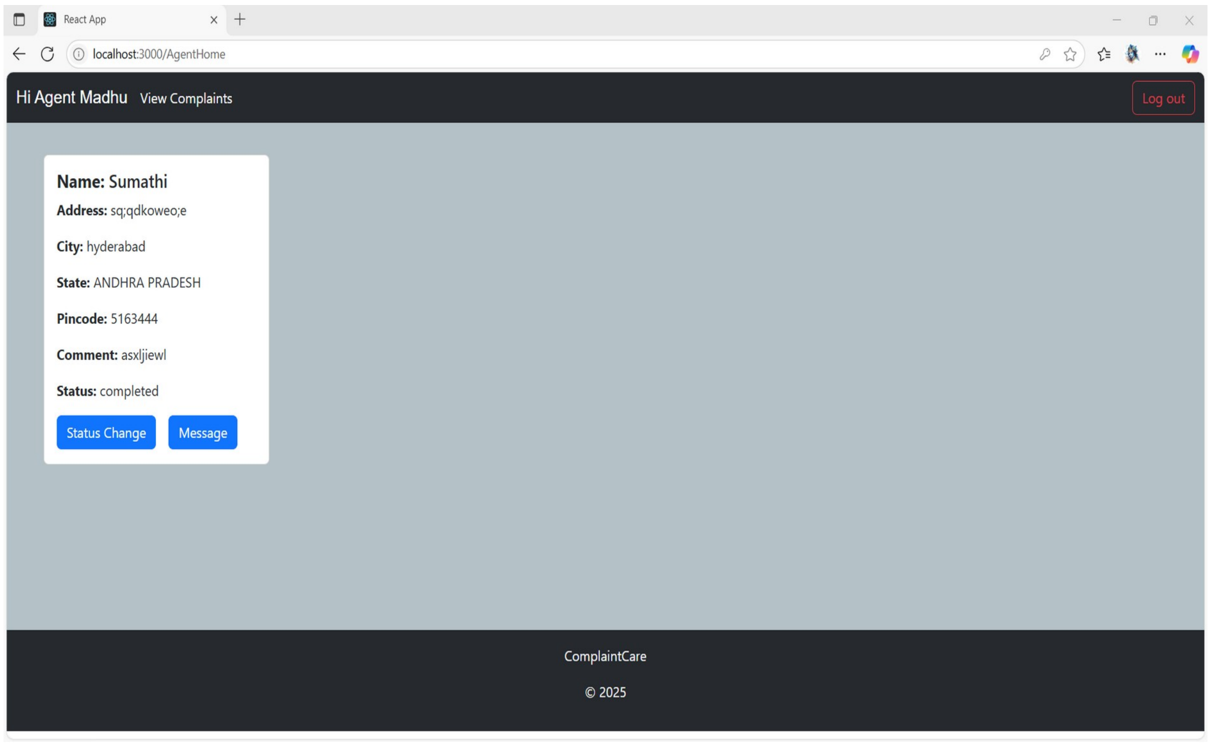
ComplaintCare

Admin Dashboard:



Name: Sumathi
Address: sq;qdkoweo;e
City: hyderabad
State: ANDHRA PRADESH
Pincode: 5163444
Comment: asxljiewl
Status: completed

Agent Dashboard:



**Name:** Sumathi
**Address:** sq;qdkoweo;e
**City:** hyderabad
**State:** ANDHRA PRADESH
**Pincode:** 5163444
**Comment:** asxljiewl
**Status:** completed

## CONCLUSION:

ResolveNow provides a smart, user-friendly, and efficient platform to manage complaints online. It simplifies the complaint process for users and improves issue resolution for administrators and departments. With features like real-time tracking, role-based access, secure authentication, and feedback collection, the system ensures transparency, accountability, and faster response times. Overall, ResolveNow is a practical solution for modernizing complaint handling in institutions, organizations, and public services.