**Full Stack Development with MERN Project**

# Introduction

**Project Title:** ResolveNow - An Online Complaint Registration and Management System  **Team ID :** LTVIP2025TMID55000

**Team Members:**

* **Team Leader :** Sankarasetty Madhan
* Role: Backend Developer
* Responsibilities: Builds RESTful APIs using Node.js and Express.js, manages authentication and server logic.
* **Team member :** S Sumitra
* Role: Frontend Developer
* Responsibilities: Works on the React-based UI, handles component design, page routing, and user interactions.
* **Team member :** Saila Kishore
* Role: Database Administrator
* Responsibilities: Designs and manages MongoDB schemas, handles CRUD operations and ensures data consistency.
* **Team member :** Sabhavathula Neeraj Nayak
* Role: Project Coordinator
* Responsibilities: Responsible for overall planning, coordination, GitHub management, and integration of frontend and backend.

# Project Overview

**Purpose:** The purpose of the ResolveNow project is to develop a full-stack web application that simplifies the process of registering and managing complaints online. It aims to provide users with a seamless experience through a modern and responsive web interface.

* Enable users to register complaints anytime.
* Allow users to track their complaints in real-time.
* Facilitate communication between users and agents assigned to handle their complaints.
* Provide an admin system to manage complaints and assign them to appropriate personnel.

**Features: For Users:**  o Sign Up / Log In – Create an account and access your complaint history.

* + Submit Complaints – Enter details of complaints including name, description, address, etc. o Track Complaints – View updates and receive notifications via email or SMS.
  + Communicate with Agents – Interact with assigned agents for issue resolution.**.**  o Order Confirmation **–** Get a message when your order is successfully placed**.**

**For Admin (Future Scope):**

* Assign Complaints – Route complaints to the appropriate department or personnel.
* Manage Complaints – View and update the status of all complaints.
* Monitor System – Ensure compliance with platform policies and regulations.

# Architecture

* Frontend (React.js)
* Built using React with multiple pages (Home, Dashboard,Complaint Submission, etc.)
* Uses React Router for navigation and Context API for managing state.
* Axios is used for API calls to the backend.
* User information and complaint status are stored in localStorage for persistence.
* Backend (Node.js + Express.js)
* Handles API routes like registration, login, submitting complaints, and tracking.
* Uses Express middleware for JSON handling and CORS.
* Connects to MongoDB using Mongoose for database operations.

Database (MongoDB)

* Stores user, complaint, and agent data.

Collections:

* users : name, email, password, contact details.
* complaints : user ID, description, date submitted, status, assigned agent.
* agents : name, assigned complaints.

# Setup Instructions

Prerequisites

* **Node.js & npm** – For running frontend and backend
* **MongoDB** – Local database (use Compass or terminal)
* **Git** – To clone the project
* **VS Code** – Recommended editor

Installation Steps:

**Clone the Project** <https://github.com/kishoresaila/online-complaint-project.git>

resolve\code >cd backend

1. **Install & Run Backend** npm install dotenv npm start
2. **Install & Run Frontend** Open a new terminal:

Resolve\code > cd frontend

Npm install

Npm start

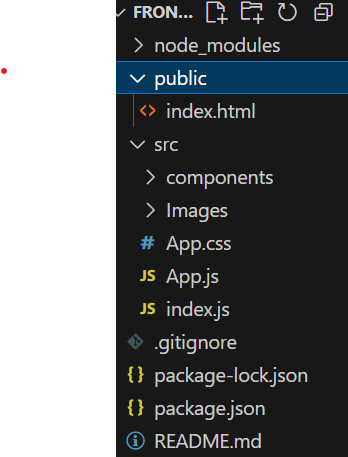
1. **Start MongoDB** : Use MongoDB Compass or run mongod in terminal.

Your app will run at:

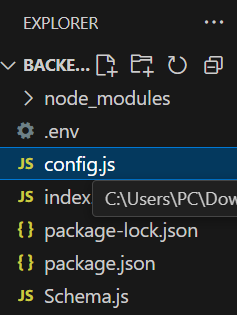
* Frontend: http://localhost:3000
* Backend API: http://localhost:5002

# Folder Structure

**Client (React frontend)**

****

**Server (Node.js backend)**

****

Running the Application

Frontend :

cd client npm start

Runs the chrome or React app at: [http://localhost:3000](http://localhost:3000/)

**Backend :**

cd server npm start # Or use: node server.js

Runs the Node.js server at: http://localhost:500[2](http://localhost:5000/)

# API Documentation

* POST /api/register : Registers a new user.
* POST /api/login : Logs in an existing user.
* GET /api/complaints : Retrieves a list of complaints for the logged-in user.
* POST /api/complaints : Submits a new complaint.
* PUT /api/complaints/:id : Updates the status of a complaint (Admin only**).**

# Authentication

How Authentication Works:

* Users register by providing their name, email, password, and contact details using the endpoint: POST /api/register.
* They log in with their email and password using: POST /api/login.

Method Used:

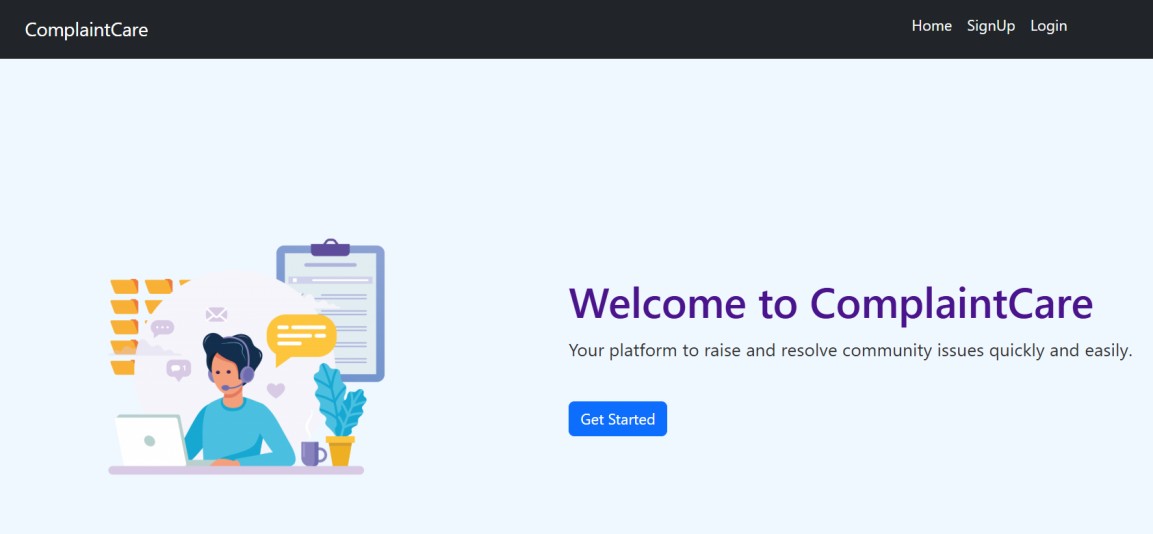
* The current setup uses basic email and password matching.
* There is no token-based authentication or sessions implemented at this stage.
* After login, the user’s details can be stored on the frontend (e.g., in localStorage) to maintain the login state. Recommendations for Improvement:

To enhance security in the future, it is recommended to:

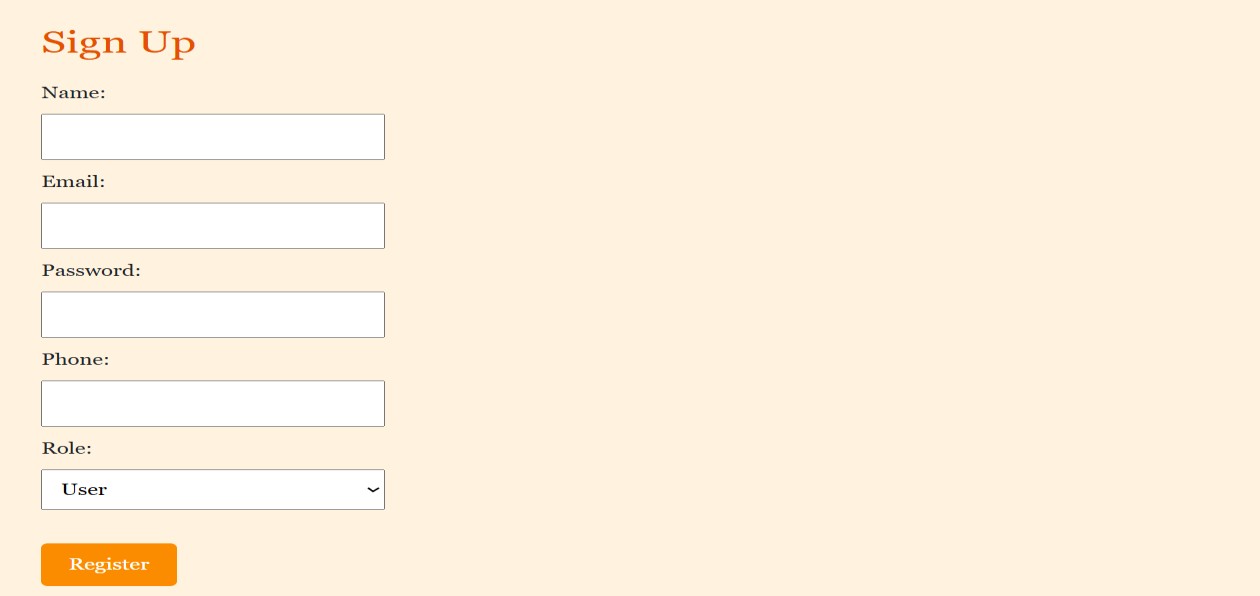
* Implement JWT (JSON Web Token) authentication.
* Use middleware to protect private API routes.
* Store tokens securely (e.g., in localStorage or HTTP-only cookies).

# User Interface

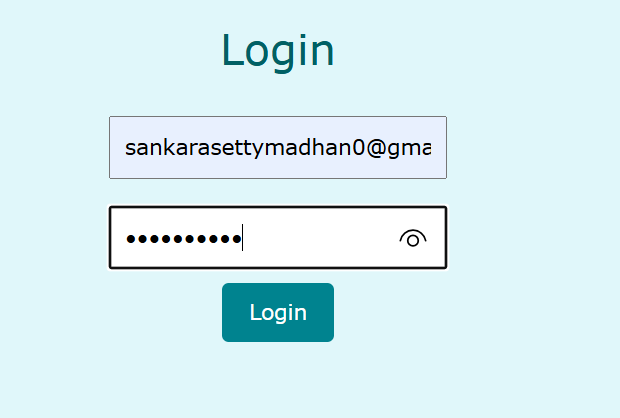
**Home page:**



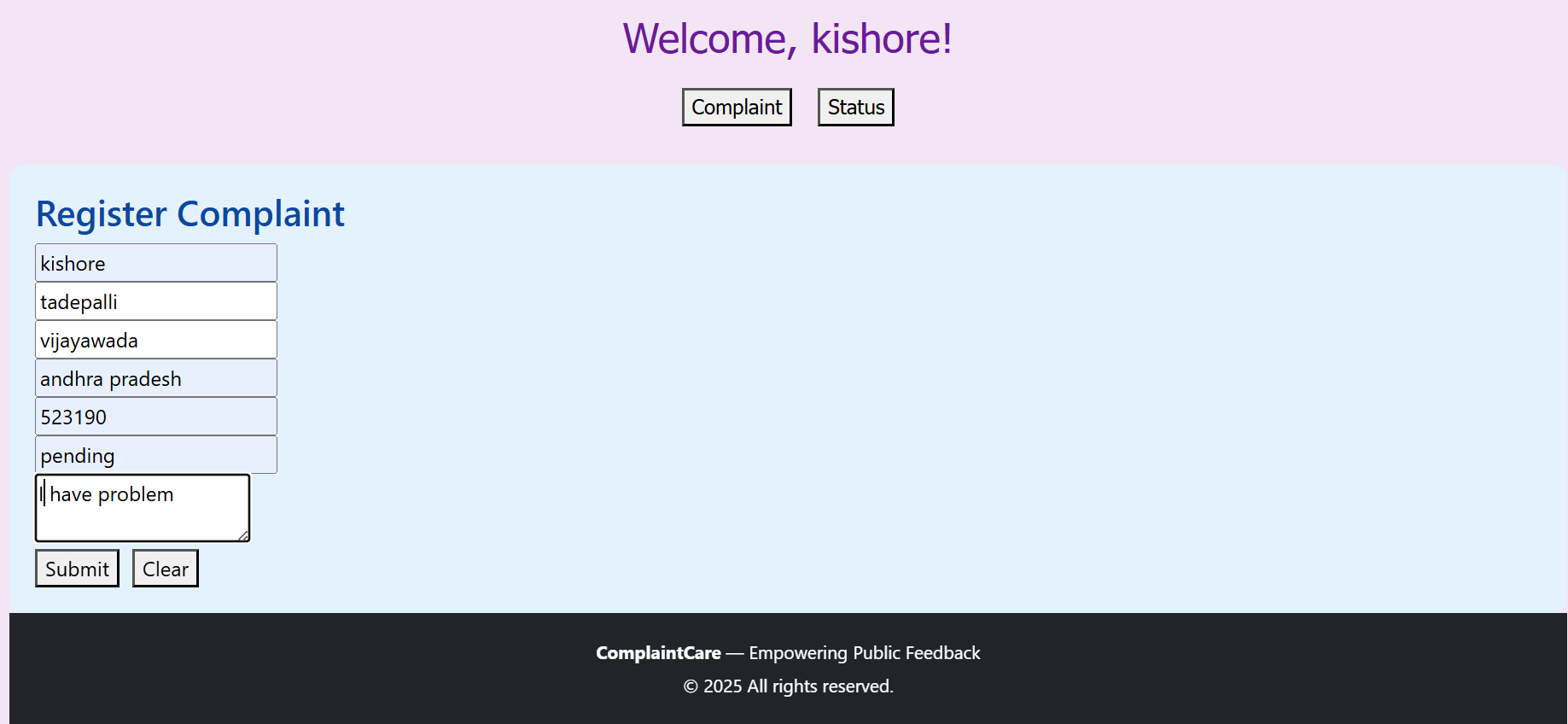
**Registration Page:**



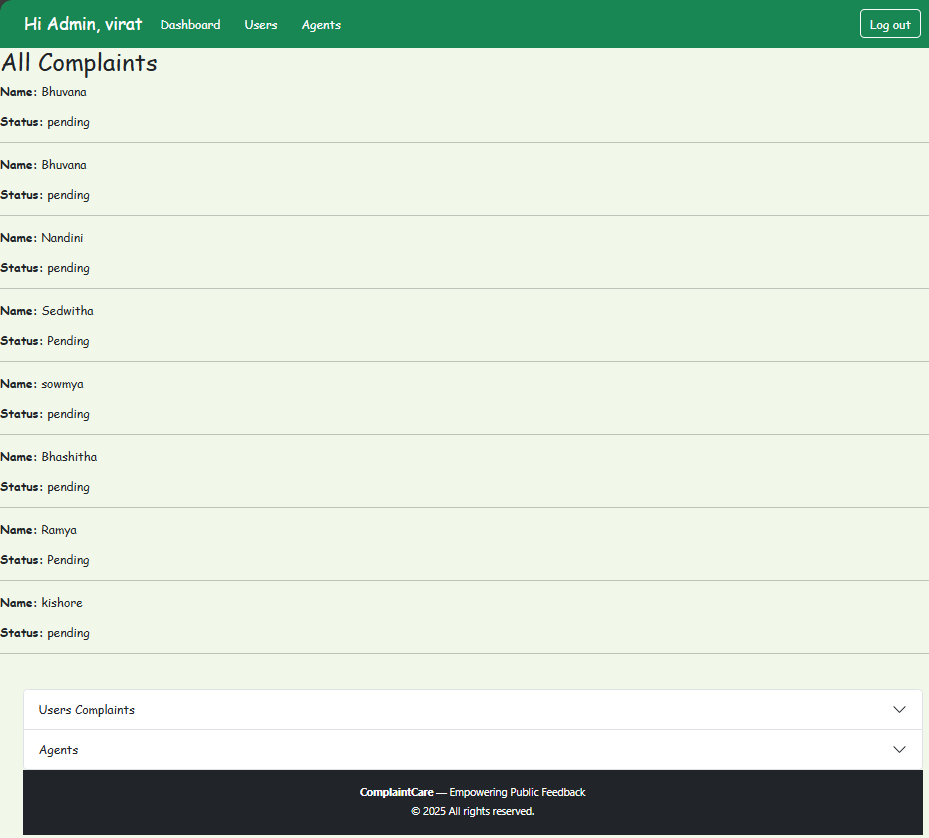
**Login Page:**

****

**Dashboard Page(To register Complaints):**



**ComplaintDetails Page:**

****

**Agent Page:**



**Project Website Link:**

# Testing

* Manual testing was done by using the app (register, login, complaint submission, tracking flow).
* Postman was used to test backend APIs.
* Browser DevTools helped inspect React components and API requests.

# Screenshots or Demo

Demo Video: Check out a quick demo of ResolveNow in action:

<https://youtu.be/tDJkiZ6lpwc>

# Known Issues

* No authentication tokens – Login does not use JWT or sessions, so user sessions are not fully secure.
* No complaint history – Users cannot view past complaints after resolving them.
* Data loss on logout – Complaint drafts or progress may reset when browser data is cleared or user logs out.
* No automated testing – All testing is manual; no test scripts are in place.
* No real-time updates – Status changes aren't reflected instantly on the user side without refreshing the page.

# Future Enhancements

* Use Jest for frontend tests. o Use Supertest for backend API testing. o Integrate video conferencing features using WebRTC API.
* Implement role-based access control for different user types (agent, admin, user).
* Enhance notification system with SMS and in-app alerts.