Full Stack Development with MERN

Project Documentation

1. Introduction

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

• **Team ID :** LTVIP2025TMID49547

Team Size : 4

Team Leader : Nali Kusuma

Team member : S Saikumar Reddy

Team member : Sibyala Preethi

Team member : Sankavarapu Lakshmi Prasanna

ResolveNow is a modern and efficient solution designed to tackle the issues in traditional complaint management systems by providing a unified digital platform. This project emphasizes transparency, quick resolutions, and user-friendly interactions through a responsive web application.

2. Project Overview

• Purpose:

The primary goal of this project is to provide a robust, secure, and scalable web platform for registering and managing complaints online. This solution ensures timely updates to users, structured complaint handling for administrators, and effective resolution workflows to improve user satisfaction and operational efficiency.

• Features:

- User Registration and Secure Login System
- Role-based Authentication (User and Admin roles)
- Real-time Complaint Submission and Tracking
- Admin Dashboard with Analytical Overview
- Status Update Notifications
- Integrated Messaging System
- Data Encryption and Token-based Authentication
- Clean and Interactive UI with Mobile Responsiveness

- REST API based modular architecture

3. Architecture

• Frontend:

Developed using React.js, the frontend follows a component-based architecture ensuring code reusability and modular design. React Router DOM facilitates navigation while Axios manages API interactions. TailwindCSS is used to ensure a consistent design language and responsiveness.

• Backend:

Node.js powers the backend with Express.js managing the routing. Backend handles REST API requests, authentication flows, validation, and business logic implementation. Express middlewares are utilized for error handling and token verification.

• Database:

MongoDB acts as a flexible and scalable NoSQL database. Using Mongoose, models and schemas are defined to represent users, complaints, and messages. Database queries are optimized with indexing where applicable.

4. Setup Instructions

• Prerequisites:

- Node.js (v18 or later)
- MongoDB Community Edition or MongoDB Atlas
- Git for version control

- Express js
- React

• Installation:

- Clone repository using: git clone <repo-url>
- Install React dependencies: cd client && npm install
- Install server dependencies: cd server && npm install
- Setup .env in server with environment variables:
  - MONGO_URI
  - JWT_SECRET
  - PORT
- Run MongoDB locally or configure cloud database
- Run frontend and backend using respective commands

5. Folder Structure

• Client Structure:

```
client/
|-- public/
|-- src/
   |-- components/      # Reusable UI components
   |-- pages/           # Functional pages like Login, Dashboard
   |-- services/        # API services
   |-- App.js
   |-- index.js
```

• Server Structure:

```
server/
|-- config/            # Database connection files
|-- controllers/       # Business logic handling
|-- middleware/        # Authentication middleware
|-- models/            # Mongoose schemas
|-- routes/            # API routes
|-- index.js           # Entry point
```

6. Running the Application

Commands to run application:

- Frontend:

```
cd client
npm run dev
```

- Backend:

```
cd server
node index.js /nodemon index.js
```

7. API Documentation

• Auth Routes:

- POST /api/users → User registration
- POST /api/userlog → User authentication

• Complaint Routes:

- POST /api/complaint→ Submit complaint
- GET /api/complaint/user → Get user complaints
- PATCH /api/complaint/update/:complaintId → Update complaint status
- DELETE /api/complaint/delete/:complaintId → Delete complaint (Admin only)

• Messaging Routes:
 - POST /api/message→ Send message to admin
 - GET /api/message/admin → Retrieve all messages for user

8. Authentication

Authentication is handled using JWT. Upon successful login, the backend generates a token signed with a secret key and valid for a limited period. Token is stored in local storage and attached to subsequent API calls. Protected routes use middleware for token verification to ensure secure data access.

9. User Interface

The application follows a minimalist yet functional UI approach:
- Registration and Login with form validations
- Dashboard with dynamic complaint cards
- Real-time updates on complaint status
- Admin dashboard with filters (Pending, Resolved, Closed)
- Notification badges
- Interactive form submission with feedback alerts

Technologies used: ReactJS, React Router.

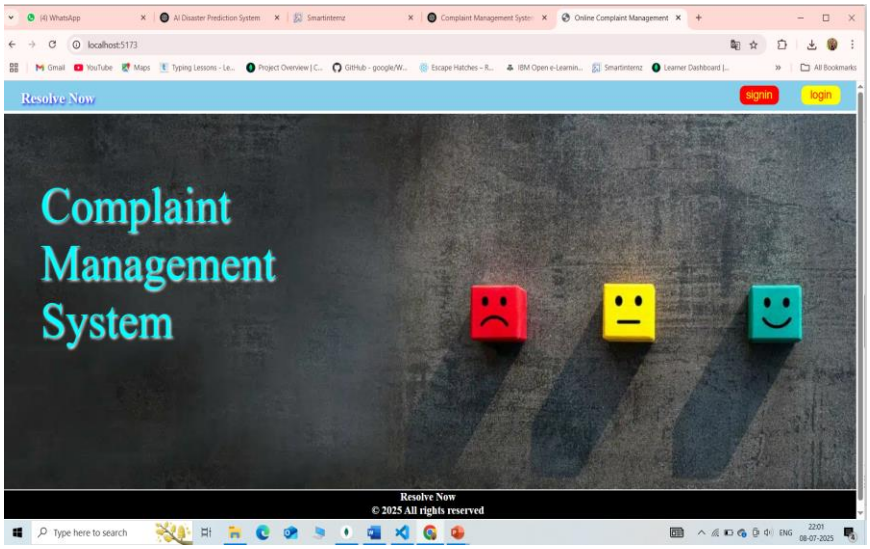
10. Testing

Testing includes:
- Unit Testing of API endpoints using Postman
- Manual frontend testing on multiple browsers
- Mobile responsiveness tests on Chrome Developer Tools
- Integration testing for complete user flows

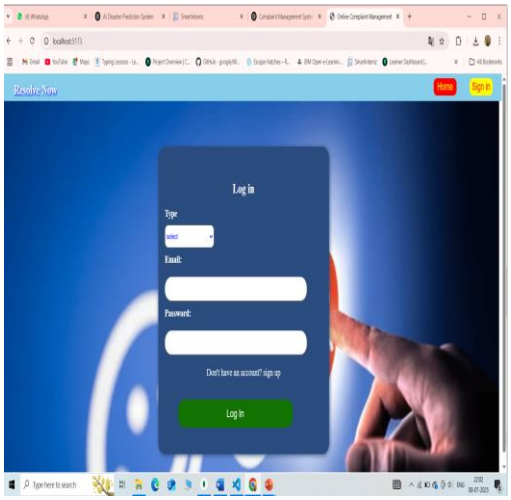
Future Testing Scope:
- Integration of Jest for backend unit testing
- React Testing Library for frontend component testing
- CI/CD pipeline with GitHub Actions for automated testing

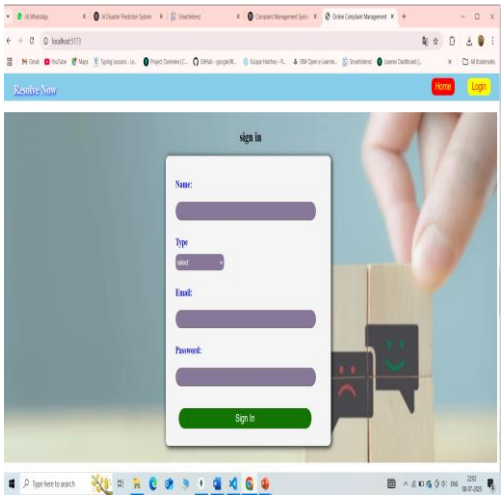
11. Screenshots or Demo

Screenshots:
- Landing Page

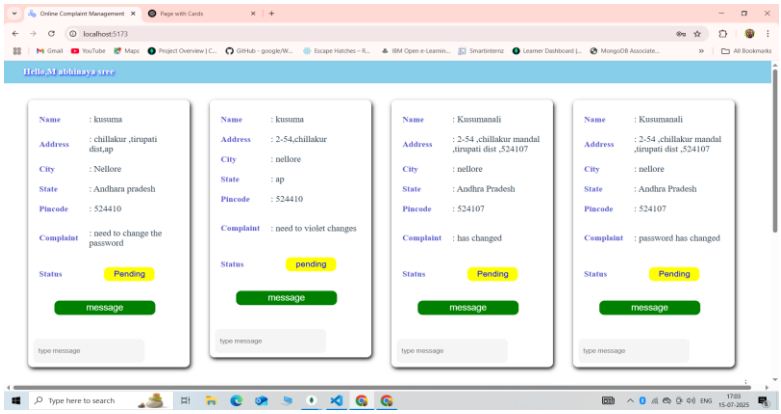


- Login/Registration Flow

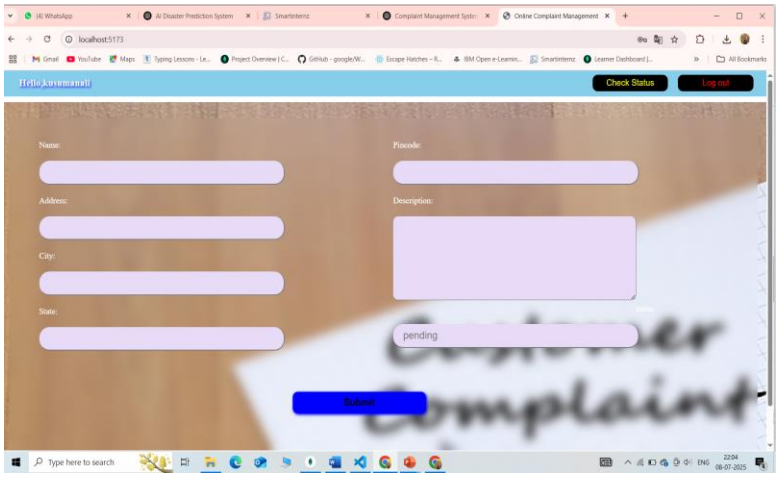




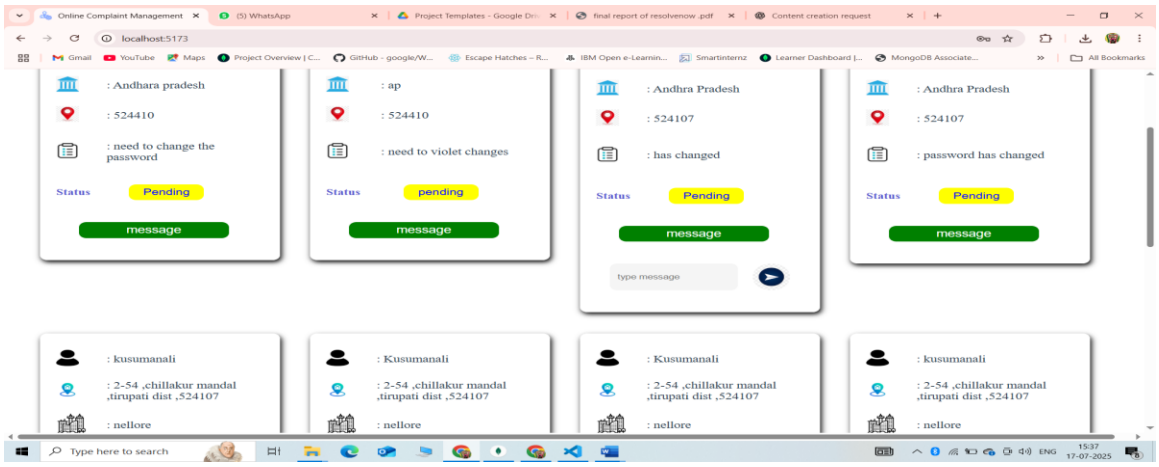
- User Dashboard



- Complaint Submission Modal



- Admin Complaint Management Interface



Demo Link: https://drive.google.com/file/d/1gzD-2wCSwVEJ_U_XLn2zOWBunt4Vhx5I/view?usp=drivesdk

12. Known Issues

- Absence of password reset functionality
- Admin role currently assigned manually
- No email verification process

13. Future Enhancements

- Two-Factor Authentication (2FA)
- Email and SMS notifications integration
- Analytics dashboard for admins (Graphs, Pie charts)
- Mobile application via React Native
- Admin role assignment through UI
- Enhanced filtering and search capabilities
- Push notifications for complaint updates