ONLINE COMPLAINT REGESTRATION

AND

MANAGEMENT SYSTEM

1. Introduction

Project title:

Online Complaint Registration and Management System

Team Members:

Harini S

Eyarkai R

Kumaresh R

Ijaz Ahmed U

2. Project Overview

Purpose:

The **Online Complaint Registration and Management System** is designed to streamline the process of filing and managing complaints online. It enables users to submit complaints regarding various issues (e.g., municipal services, customer support) and track the status of their issues in real-time. The system aims to improve communication between the public and service providers, ensuring timely resolution of complaints and increased transparency.

Features:

Complaint Registration: Users can submit complaints with a description, category, and supporting files.

Complaint Tracking: Users can track the status of their complaints in real-time.

Admin Dashboard: Admins can view, update, and resolve complaints.

User Authentication: Secure login and registration process for users and administrators.

Complaint Categories: Users can categorize complaints (e.g., Service Quality, Maintenance Issues, etc.).

Email Notifications: Automated emails for complaint updates and status changes.

Search and Filter: Users and admins can search and filter complaints based on different criteria (e.g., date, status, category).

3. Architecture

Frontend:

The frontend of the application is built using React.js, a popular JavaScript library for creating user interfaces. React' s component-based architecture allows for easy reusability of UI elements like forms, buttons, and cards. The frontend interacts with the backend through RESTful API calls and updates the UI based on the data received from the server.

State Management: We use Redux for global state management to handle user authentication and complaint data.

Routing: React Router is used for navigating between different pages (Home, Login, Dashboard, etc.).

Backend:

The backend is built using Node.js and Express.JS to serve the RESTful APIs. The server handles client requests, communicates with the database, and returns data in JSON format.

Node.js provides a fast, event-driven server environment.

Express.js is used to simplify routing and middleware management.

JWT Authentication: The backend uses JSON Web Tokens (JWT) for secure authentication and authorization.

Database:

We use MongoDB for storing complaint data and user information due to its scalability and flexibility. The system uses a NoSQL schema, which allows for easy addition of new fields without affecting existing data.

Database Schema:

Users Collection: Stores user details (name, email, password, role).

Complaints Collection: Stores complaint information (complaint ID, user ID, description, category, status, date created).

Admin Collection: Stores admin credentials and privileges.

4. Setup Instructions

Prerequisites:

Node.js (v14.x or above)

MongoDB (local or MongoDB Atlas for cloud storage)

Git (for cloning the repository)

Installation:

1.Clone the repository:

```bash
git clone https://github.com/your-repo/complaint-management-system.git
cd complaint-management-system
```

2. Install dependencies:

Frontend:

```bash
cd client
npm install
```

Backend:

```bash
cd server
npm install
```

3.Set up environment variables:

- Create a `.env` file in both the `client` and `server` directories and configure them.

Server `.env` file example:

PORT=5000

MONGO_URI=mongodb://localhost:27017/complaintsDB

JWT_SECRET=your_jwt_secret

Client `.env` file example:

```

REACT_APP_API_URL=http://localhost:5000/api

```

4. Run the application:

Start the frontend server:

```bash

cd client

npm start

```

- Start the backend server:

```bash

cd server

npm start

```

5. Folder Structure

Client (Frontend - React):

- `client/`

- `src/`
- `components/` - Reusable UI components (Button, Modal, ComplaintCard)
- `pages/` - Different pages of the app (Home, Dashboard, Login)
- `redux/` - Redux state management (actions, reducers, store)
- `services/` - API calls to interact with the backend
- `App.js` - Main component for routing and structure
- `index.js` - Entry point for the React app
- `public/` - Static assets (images, icons)
- `package.json` - Frontend dependencies

Server (Backend - Node.js/Express):
- `server/`
  - `models/` - Database schemas (User, Complaint, Admin)
  - `routes/` - API routes for complaints and authentication
  - `controllers/` - Business logic for each route
  - `middleware/` - Authentication middleware for protecting routes
  - `server.js` - Main entry point for the backend
  - `package.json` - Backend dependencies

6. Running the Application

Frontend:

To run the frontend locally:

```bash
cd client
npm start
```

This will start the development server at `http://localhost:3000`.

Backend:

To run the backend locally:

```bash
cd server
npm start
```

This will start the backend server at `http://localhost:5000`.

7. API Documentation

Endpoint: `POST /api/users/register`

Description: Registers a new user (for both complainants and admins).

Request Body:

```json
{
    "name": "John Doe",
    "email": "john@example.com",
    "password": "password123",
    "role": "user"
}
```

Response:

```json
{
    "message": "User registered successfully",
    "userId": "123abc"
}
```

Endpoint: `POST /api/complaints`

Description: Creates a new complaint.

Request Body:

```Json
{
```

"description": "Leaky faucet in my apartment",

"category": "Maintenance",

"userId": "123abc"

}

Response:

```Json

{

"message": "Complaint submitted successfully",

"complaintId": "456def"

}

Endpoint: `GET /api/complaints/:userId`

Description: Retrieves all complaints submitted by a user.

Response:

```json

{

"complaintId": "456def",

"description": "Leaky faucet in my apartment",

"status": "Pending",

"category": "Maintenance",

"dateCreated": "2024-11-01T10:00:00Z"

},

...

]

**8. Authentication**

The system uses JWT (JSON Web Tokens) for authentication and authorization:

Login: After successful login, the server generates a JWT token for the user, which is sent to the client.

Authorization: The token is stored in the client (typically in `localStorage`), and it is sent in the `Authorization` header of API requests to protected routes.

Token Expiry: Tokens are set to expire after 1 hour, requiring users to log in again.

## 9. User Interface

Screenshots:

_Login Page:_ ![Login](link_to_screenshot_1)

_Dashboard:_ ![Dashboard](link_to_screenshot_2)

_Complaint Submission:_ ![Complaint Form](link_to_screenshot_3)

## 10. Testing

Testing is performed using Jest for unit tests and React Testing Library for frontend components. We also use Super test for integration testing of the API endpoints.

Unit tests are written for individual components and backend logic.

Integration tests ensure that frontend and backend communicate correctly.

End-to-end tests are run using Cypress for real-user scenarios

## 11. Screenshots or Demo

For a live demo or screenshots, refer to the following links:

- [Live Demo](link_to_demo)

- [Screenshots](link_to_screenshot_folder)

## 12. Known Issues

Issue #1: Complaints may not be assigned to an admin automatically (manual assignment required).

Issue #2: Pagination on the complaint dashboard is not yet implemented.

Issue #3: Email notifications occasionally delay after complaint submission.

## 13. Future Enhancements

Real-time Updates: Implement WebSockets for real-time complaint status updates.

Mobile App: Develop a mobile app using **React Native** for complaint management.

Advanced Search Filters: Add additional search filters, such as complaint priority, location, etc.

Admin Role Management:** Allow admins to manage user roles and permissions more granularly.