**Full Stack Development with MERN**

**Project Documentation**

1.Introduction

Project Title: Online Complaint Registration System

Team Lead: Srinath

Team Members: Nithish, Vedhathiri, Deepak

2. Project Overview

Purpose:

The purpose of this project is to provide an online platform for users to register complaints and track their status. It aims to streamline the process of complaint management, enabling users to report issues efficiently and allowing administrators to manage and resolve complaints effectively.

Features:

1. User Registration & Login: Users can sign up, log in, and manage their profiles.

2.Complaint Submission: Users can submit complaints along with descriptions, attachments, and categories.

3.Complaint Tracking: Users can track the status of their complaints.

4.Admin Panel: Admins can view, update, and resolve complaints.

5. Email Notifications: Users and admins receive email notifications on complaint updates and status changes.

6.Search and Filter: Users and admins can search and filter complaints based on category, status, or date.

3. Architecture

Frontend:

The frontend is built using React.js. It includes the following components.

1.User Interface: Forms for complaint registration, user login, and complaint tracking.

2. State Management: Managed using React's useState and Context API for global state management.

3.API Integration: React communicates with the backend through RESTful API endpoints to submit and fetch data.

Backend:

The backend is developed using Node.js and Express.js, providing RESTful APIs for the frontend to interact with,

1.Authentication: JWT (JSON Web Tokens) is used for secure user authentication.

2.Complaint Management: Express routes handle CRUD operations for complaints, user registration, and status updates.

Database:

MongoDB is used to store user and complaint data. The schema for complaints includes fields such as title, description, category, status, and timestamps. The user schema includes fields for email, password (hashed), and role (user or admin).

4. Setup Instructions

Prerequisites:

1. Node.js (v14 or above)

2. MongoDB (Local or Atlas)

3.npm or yarn (Package manager)

Installation:

1. Clone the repository:

```bash

git clone https://github.com/your-repository/online-complaint-registration.git

```

2. Navigate to the project directory:

```bash

cd online-complaint-registration

```

3. Set up the client (frontend):

```bash

cd client

npm install

```

4. Set up the server (backend):

```bash

cd server

npm install

```

5. Set up environment variables in both the client and server directories:

6. For server, create a `.env` file with necessary configurations (e.g., MongoDB URI, JWT secret).

7.For client, set up a `.env` file with the backend API URL.

**5.Folder Structure**

Client:

client/

├── src/

├── components/

├── pages/

├── services/

├── context/

AuthContext)

└── App.js

Server:

server/

├── models/

├── routes/

├── controllers/

├── middleware/

authentication)

├── config/

connection)

└── server.js

6. Running the Application

Frontend:

To start the frontend, navigate to the client directory and run:

bash

npm start

Backend:

To start the backend, navigate to the server directory and run:

bash

npm start

7. API Documentation

Description: Register a new user.

Request Body:

json

{

"email": "user@example.com",

"password": "password123"

}

Response:

json

{

"message": "User registered successfully"

}

Description: Authenticate a user and return a JWT token.

Request Body:

json

{

"email": "user@example.com",

"password": "password123"

}

Response:

json

{

"token": "JWT\_TOKEN\_HERE"

}

Description: Submit a new complaint.

Request Body:

json

{

"title": "Internet is down",

"description": "The internet is not working since morning.",

"category": "Internet",

"status": "Pending"

}

Response:

json

{

"message": "Complaint registered successfully"

}

Description: Fetch all complaints.

Response:

json

[

{

"title": "Internet is down",

"status": "Pending",

"category": "Internet",

"createdAt": "2024-11-12T12:34:56Z"

}

]

**8. Authentication**

JWT Authentication is used for secure login and protected routes.

- When a user logs in, a JWT token is generated and sent to the client.

- The token is stored in the browser (localStorage or cookies) and used in the header of subsequent requests.

- The backend verifies the token to authorize the user for specific actions (e.g., submitting complaints, viewing complaints).

9. User Interface

- Screenshots or GIFs showcasing key UI features:

- Login Page

- Complaint Submission Form

- Complaint Tracking Dashboard

10. Testing

Unit Testing:

Jest is used for testing frontend components and backend API routes.

Integration Testing:

Supertest is used for testing the API endpoints.

Tools:

- Jest

- Supertest

11. Screenshots or Demo

- Link to live demo: [Demo link]

- Screenshots of key pages such as the login page, complaint submission form, and complaint dashboard.

12. Known Issues

Issue 1: Sometimes the complaint status does not update instantly due to caching issues.

Issue 2: Some users experience delay in email notifications.

13. Future Enhancements

Feature 1: Add a priority level (High, Medium, Low) for complaints.

Feature 2: Implement a real-time chat feature for users and admins.

Feature 3: Enhance the search feature by adding more filters (e.g., date range, status).

THANK YOU