**Full Stack Development With Mern**

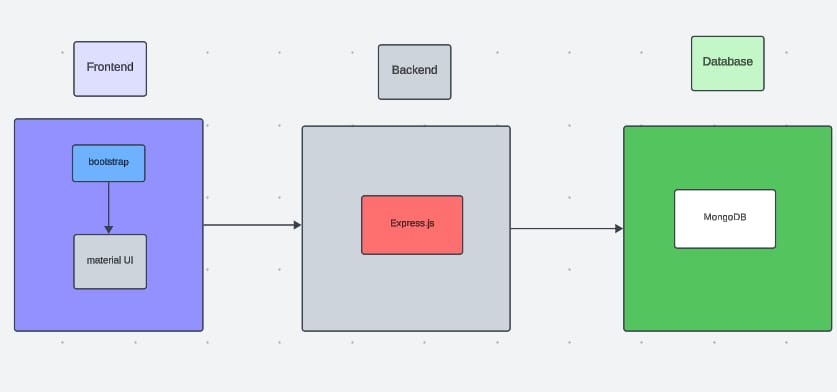
**1.Introduction**

* **Project Title: Online Complaint Registration And Management System**
* **Team Members:Blessy,Deepana,Priyadharshini,Preetika**

**2. Project Overview**

* **Purpose: The purpose of the Online Complaint Registration and Management System is to provide a platform for users to easily submit, track, and manage complaints, while enabling administrators to efficiently handle and resolve these issues.**
* **Features:**
* **User Registration & Login: Secure sign-up and login for users.**
* **Complaint Submission: Submit complaints with details and attachments.**
* **Complaint Tracking: Track the status of complaints in real-time.**
* **Admin Dashboard: Manage and resolve complaints from the admin panel.**
* **Complaint Categorization: Organize complaints by type (e.g., service, infrastructure).**
* **Email Notifications: Notify users and admins about status updates.**
* **Search & Filter: Find complaints by status, category, or date.**
* **Complaint History: View past complaints and resolutions.**
* **Role-based Access: Different access levels for users and admins.**

**3. Architecture**

****

* **Frontend:** **The frontend is powered by React, with React Router for navigation and Axios for making API requests.**
* **Backend:** **The backend uses Node.js and Express.js to handle API requests, authentication, and database operations with MongoDB.**
* **Database:** **The database uses MongoDB to store user and complaint data, with collections for users, complaints, and complaint statuses, managed through Mongoose schemas.**

**4. Setup Instructions**

* **Prerequisites:** **Prerequisites include Node.js, MongoDB (local or Atlas), and npm for managing dependencies.**
* **Installation:**

**1. Clone the repository: `git clone <repo-url>`**

**2. Install backend dependencies: `cd server && npm install`**

**3. Install frontend dependencies: `cd client && npm install`**

**4. Set up environment variables in the `.env` file for both frontend and backend.**

**5. Run the backend: `npm start` inside the server directory.**

**6. Run the frontend: `npm start` inside the client directory.**

**5. Folder Structure**

* **Client:** **The React frontend is organized into components, pages for different routes, and uses context or state management to handle global application data.**
* **Server:** **The Node.js backend is structured with routes for API endpoints, controllers for business logic, models for database schemas, and middleware for authentication and error handling.**

**6. Running the Application**

* **Frontend:** **Run `npm start` in the client folder to start the React app and open it in the browser.**
* **Backend:** **Run `npm start` in the server directory to start the Node.js backend server.**

**7. API Documentation**

* **Document all endpoints exposed by the backend**.

**API Endpoints:**

**1. POST /api/auth/register - Register a new user.**

**2. POST /api/auth/login - Login and get JWT token for authentication.**

**3. GET /api/complaints - Get all complaints for the logged-in user.**

**4. POST /api/complaints - Create a new complain**

**5. GET /api/complaints/:id - Get details of a specific complaint.**

**6. PUT /api/complaints/:id- Update a complaint's status (admin only).**

**7. DELETE /api/complaints/:id- Delete a complaint (admin only).**

* **Include details about tokens, sessions, or any other methods used.**
* **Authentication is handled using \*\*JWT tokens\*\*, which are issued after login and stored in the client's local storage, then sent in the Authorization header for subsequent requests.**

**8. Authentication**

* **Authentication is managed with \*\*JWT tokens\*\* issued at login, while authorization is handled by checking user roles to grant access to specific routes for users and admins.**
* **The project uses \*\*JWT tokens\*\* for authentication, which are stored in local storage and included in the Authorization header for protected API requests.**

**9.User Interface**

* **The user interface includes login/registration forms, a dashboard to view and track complaints, a submission form for new complaints, real-time status updates, and admin features for managing complaints and users.**

**10.** **Testing**

* **Unit Testing: Test individual functions and components (e.g., form validation,**
* **Integration Testing: Ensure that frontend and backend interact correctly (e.g., complaint submission from frontend to backend).**
* **API Testing: Use tools like Postman to test all API endpoints (e.g., user registration, complaint creation, status updates).**
* **UI Testing: Use tools like Jest and React Testing Library to test React components for proper rendering and functionality.**
* **End-to-End Testing: Simulate user flows (e.g., registration, login, complaint submission) using tools like Cypress.**
* **Security Testing: Test for vulnerabilities such as SQL injection, XSS, and authentication flaws.**
* **Performance Testing: Ensure the system handles a high volume of complaints and users smoothly**

**11.Screenshots or Demo**

**12.Known Issues**

* **Email Delays: Email notifications may be delayed.**
* **Mobile UI Issues: Some elements may not display properly on mobile devices.**
* **Login Problems: Occasional issues with token expiration or session management.**
* **File Upload Issues: Problems with large file sizes or certain formats.**
* **Admin Dashboard Lag: Slow loading when displaying many complaints.**
* **Search Inaccuracy: Search may not always return accurate results.**

1. **Future Enhancements**

* **Future enhancements for an Online Complaint Registration and Management System could involve improved user authentication, mobile app integration, chatbot assistance, automated complaint prioritization, and enhanced data analytics for better reporting and decision-making.**