Full Stack Development with MERN

Project Documentation

1. Introduction

- **Project Title:** Online Complaint Registration
- **Team Members:**
  - Alvin Roy A
  - Shaun Eliot Alex Nicholas
  - Darwin Jeffry S
  - Sakthi Manikandan I

2. Project Overview

- **Purpose:**
  - The purpose of this application is to create a digital platform that simplifies the process of registering, tracking and resolving complaints.
  - The system is designed to improve the overall efficiency and transparency of complaint management while enhancing user satisfaction.
- **Goals:**
  - Provide a user-friendly platform for individuals and organizations to register complaints conveniently.
  - Enable real-time tracking of complaints and updates.
  - Ensure users receive updates about their complaints.
  - Allow direct interaction between users and agents for effective communication.
  - Deliver a smooth and efficient complaint resolution experience.
- **Features:**
  - **User Registration:** Secure account creation with recovery options.
  - **Complaint Submission:** Easy to use complaint registration form**,** fields to capture detailed complaint information.
  - **Real-Time Complaint Tracking:** Users can view the status of their complaints on a dedicated dashboard.
  - **Agent Interaction:** Built-in messaging system for users to communicate with assigned agents. Real-time updates and clarification of issues through the platform.
  - **Admin Control Panel:** View and manage all registered complaints.
  - **Feedback Mechanism:** Users can rate their experience and provide feedback on the resolution process.

3. Architecture

- **Frontend:** The frontend is built using React.

  Key Features:

  - **Component-Based Design:** Reusable components like Header, Dashboard, ComplaintForm, AdminPanel, and ChatWindow. Components are organized into folders for better maintainability.]
  - **Routing:** React Router handles client-side routing to navigate between pages
  - **State Management:** React Context API or Redux is used to manage global states like authentication, user data and complaint statuses.
  - **API Integration:** Axios is used to communicate with the backend via RESTful APIs.
  - **Styling:** Material-UI and Bootstrap for responsive and modern design, Custom CSS for unique components where necessary.

- **Backend:** The backend uses Node.js with Express.js.

  Key Features:

  - **Middleware:** Express Middleware for request parsing, logging and error handling. Authentication Middleware for user verification using JWT (JSON Web Token).
  - **Real-Time Communication:** Socket.io for real-time chat and live status updates.
  - **Error Handling:** Centralized error-handling middleware for consistent and secure responses.
  - **Scalability:** Modular routing for separating user, admin, and agent functionalities. Designed to handle a growing number of users and requests efficiently.

- **Database:** MongoDB is used for database management

  Schema:

  - **Users Collection:**

    ```
    {
      "_id": "ObjectId",
      "name": "string",
      "email": "string",
      "password": "hashed_string",
      "role": "user | admin | agent",
    ```

```
  "createdAt": "Date",
  "updatedAt": "Date"
}
```

- **Complaints Collection:**

```
{
  "_id": "ObjectId",
  "userId": "ObjectId (ref: Users)",
  "description": "string",
  "status": "Submitted | In Progress | Resolved | Closed",
  "attachments": ["file_paths"],
  "assignedAgent": "ObjectId (ref: Users)",
  "createdAt": "Date",
  "updatedAt": "Date"
}
```

- **Messages Collection**:

```
{
  "_id": "ObjectId",
  "complaintId": "ObjectId (ref: Complaints)",
  "senderId": "ObjectId (ref: Users)",
  "message": "string",
  "timestamp": "Date"
}
```

Database Interactions:

- **Create:** Add new users, complaints, or messages.
- **Read:** Fetch complaints based on status, user, or time.
- **Update:** Modify complaint statuses, user details, or message threads.
- **Delete:** Remove old data based on retention policies.

4. Setup Instructions

- **Prerequisites:**
  - **Operating System:** Windows 8 or later / macOS / Linux.
  - **Software Dependencies:**
    - **Node.js:** Runtime environment for running the backend.
    - **MongoDB:** NoSQL database for storing data.
    - **Git:** Version control system for cloning the project repository.
    - **npm (Node Package Manager):** Comes with Node.js for managing project dependencies.
  - **Web Browsers:** Any modern web browser (e.g. Google Chrome, Firefox).
  - **Additional Tools:** Visual Studio Code or any other preferred IDE.
- **Installation:**
  - Open your terminal or command prompt and run:
    ```
    git clone
    https://github.com/shaunnicholas/complaint_registration_NaanMudhalvan.git
    cd complaint_registration_NaanMudhalvan
    ```
  - Install Frontend Dependencies
  - Navigate to the frontend directory
    ```
    cd frontend
    npm install
    ```
  - This command installs all necessary React dependencies
  - Install Backend Dependencies
  - Navigate to the backend directory
    ```
    cd backend

    npm install
    ```
  - This will install dependencies like Express.js, Mongoose, Socket.io and JWT.
  - Set Up Environment Variables
    ```
    PORT=5000

    MONGO_URI=mongodb://localhost:27017/details

    JWT_SECRET=your_secret_key

    CLIENT_URL=http://localhost:3000
    ```

- Start the MongoDB service locally or connect to a cloud database:
  Mongod
- Start the Backend Server. Open a new terminal, navigate to the backend directory and run
  npm start
- The backend server should now be running at http://localhost:5000.
- Start the Frontend Server.
- Open a new terminal, navigate to the frontend directory, and run

  npm start

- This will start the React development server at http://localhost:3000.
- Open your browser and navigate to http://localhost:3000.
- Register a new user or log in with existing credentials.
- Test the complaint registration and management features.

5. Folder Structure

- **Client:**

```
frontend/
│
├── public/              # Static files
│   ├── index.html        # Main HTML file
│   └── assets/           # Images, icons, and other static assets
│
├── src/                 # Source code
│   ├── components/       # Reusable UI components
│   │   ├── Header.js      # Header/navigation bar
│   │   ├── Footer.js      # Footer
│   │   ├── Dashboard.js    # User dashboard
│   │   ├── ComplaintForm.js # Complaint registration form
│   │   ├── ChatWindow.js   # Chat interface
│   │   └── AdminPanel.js   # Admin-specific features
│   │
│   ├── pages/           # Page-level components
│   │   ├── Login.js        # Login page
│   │   ├── Register.js     # Registration page
│   │   ├── Home.js         # Home page
│   │   └── NotFound.js     # 404 error page
│   │
│   ├── context/          # Context API for state management
│   │   └── AuthContext.js  # User authentication context
│   │
│   ├── hooks/            # Custom React hooks
│   │   └── useAuth.js      # Authentication-related utilities
│   │
│   ├── services/         # API service functions
│   │   ├── api.js          # Axios instance and API calls
│   │   └── authService.js  # Authentication-specific APIs
│   │
│   ├── styles/           # CSS and styled-components
│   │   └── global.css      # Global styles
│   │
│   ├── App.js            # Main application component
│   ├── index.js          # Entry point
│   └── routes.js         # App routes configuration
│
├── package.json          # Project dependencies and scripts
└── README.md               # Frontend-specific documentation
```

- **Server:**

```
backend/
│
├── src/                    # Source code
│   ├── controllers/        # Business logic and API handling
│   │   ├── authController.js # Authentication logic
│   │   ├── complaintController.js # Complaint-related APIs
│   │   └── adminController.js # Admin-specific features
│   │
│   ├── models/             # MongoDB schemas
│   │   ├── User.js         # User schema
│   │   ├── Complaint.js    # Complaint schema
│   │   └── Message.js      # Message schema
│   │
│   ├── routes/             # API routes
│   │   ├── authRoutes.js   # Routes for login, registration
│   │   ├── complaintRoutes.js # Routes for complaint handling
│   │   └── adminRoutes.js  # Admin-related routes
│   │
│   ├── middleware/         # Middleware functions
│   │   ├── authMiddleware.js # JWT authentication
│   │   └── errorHandler.js  # Error handling middleware
│   │
│   ├── config/             # Configuration files
│   │   ├── db.js           # MongoDB connection setup
│   │   └── env.js          # Environment variable configuration
│   │
│   ├── utils/              # Utility functions
│   │   ├── tokenUtils.js    # JWT token generation/validation
│   │   ├── logger.js        # Logging utility
│   │
│   ├── app.js              # Express app setup
│   ├── server.js            # Server entry point
│   └── socket.js            # Socket.io integration
│
├── .env                    # Environment variables
├── package.json            # Project dependencies and scripts
└── README.md               # Backend-specific documentation
```

6. Running the Application

- Starting the Frontend
  - Navigate to the frontend directory:
    cd frontend
  - Install dependencies
    npm install
  - Start the React development server
    npm start
  - Open your browser and navigate to
    http://localhost:3000
- Starting the Backend
  - Navigate to the backend directory
    cd backend
  - Install dependencies
    npm install
  - Start the Node.js server
    npm start
  - The backend server should now be running at
    http://localhost:5000

7. API Documentation

- User Registration
    - Endpoint: /api/auth/register
    - Method: POST
    - Description: Allows a new user to register.
    - Request Body:

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

Response:

201 Created

```
{
    "message": "User registered successfully!",
    "userId": "64abc123def456"
}
```

- User Login
    - Endpoint: /api/auth/login
    - Method: POST
    - Description: Authenticates a user and provides a JWT token.
    - Request Body:

```
{
    "email": "john.doe@example.com",
    "password": "password123"
}
```

Response:

200 OK

```
{
   "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...",
   "user": {
      "id": "64abc123def456",
      "name": "John Doe",
      "email": "john.doe@example.com"
   }
}
```

- Register a Complaint
  - Endpoint: /api/complaints
  - Method: POST
  - Description: Submits a new complaint.
  - Headers:
    Authorization: Bearer <JWT token>
  - Request Body:

```
{
   "title": "Product Defect",
   "description": "The product I received has a manufacturing defect.",
   "category": "Product Issue",
   "address": "123, Main Street, City, Country"
}
```

Response:

201 Created

```
{
   "message": "Complaint registered successfully!",
   "complaintId": "64xyz789uvw123"
}
```

- Get All Complaints (Admin)
    - Endpoint: /api/complaints
    - Method: GET
    - Description: Retrieves all registered complaints (Admin access required).
    - Headers:
      Authorization: Bearer <JWT token>
    - Response:
      200 OK

```
[
  {
    "id": "64xyz789uvw123",
    "title": "Product Defect",
    "status": "Pending",
    "user": {
      "id": "64abc123def456",
      "name": "John Doe"
    }
  }
]
```

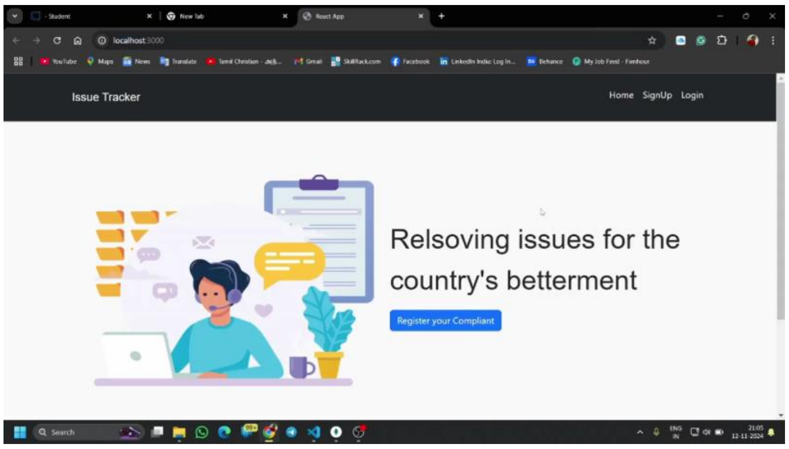
- Get User Complaints
    - Endpoint: /api/complaints/my
    - Method: GET
    - Description: Retrieves complaints submitted by the logged-in user.
    - Headers:
      Authorization: Bearer <JWT token>
    - Response:
      200 OK

```
[

  {

    "id": "64xyz789uvw123",

    "title": "Product Defect",

    "description": "The product I received has a manufacturing defect.",

    "status": "Pending",

    "createdAt": "2024-11-10T12:00:00Z"

  }

]
```

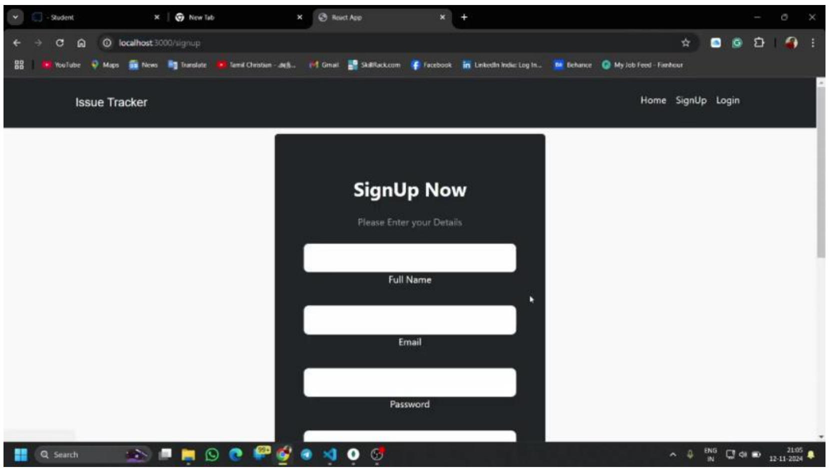
8. Authentication

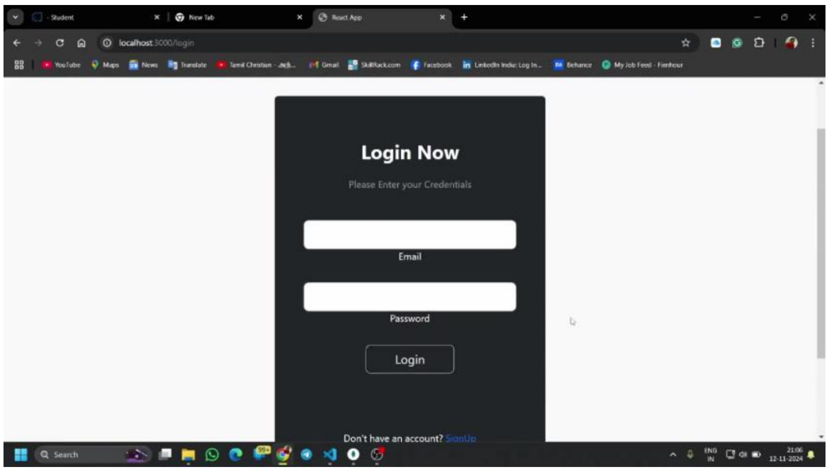
- Token Expiry:
  Tokens include an expiration time to limit their lifespan. After expiry, users must log in again to receive a new token.
- Secure Secret Key:
  The JWT secret key is stored in environment variables and never exposed in the codebase.
- CSRF Protection:
  If tokens are stored in cookies, CSRF tokens can be implemented to prevent cross-site request forgery attacks.

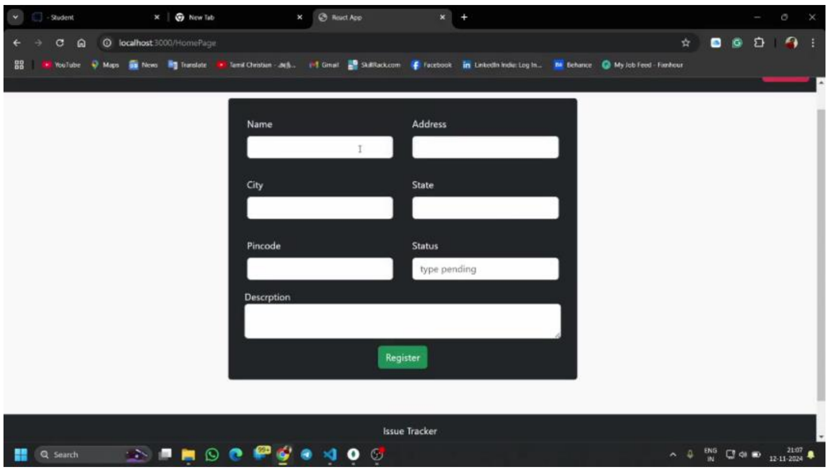
9. User Interface

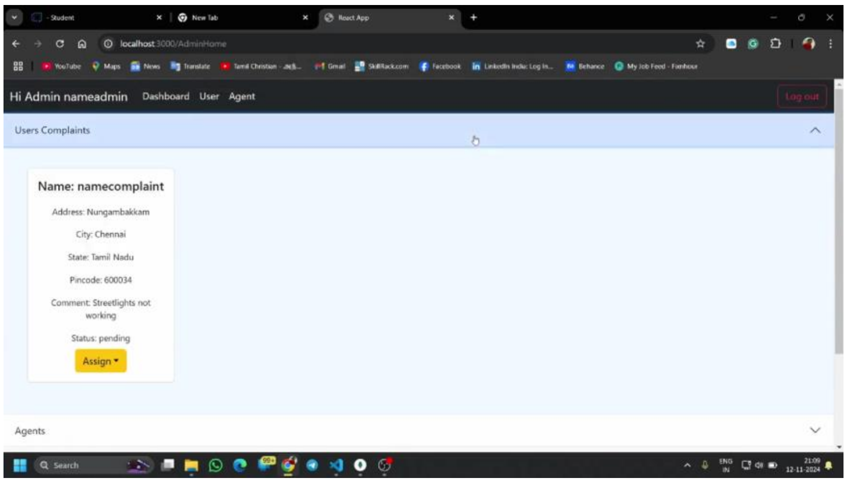


Relsoving issues for the country's betterment


localhost:3000/signup
Issue Tracker
Home
SignUp
Login
SignUp Now
Please Enter your Details
Full Name
Email
Password


localhost:3000/login
Login Now
Please Enter your Credentials
Email
Password
Login
Don't have an account? SignUp


localhost:3000/HomePage
Name
Address
City
State
Pincode
Status
type pending
Descrption
Register
Issue Tracker

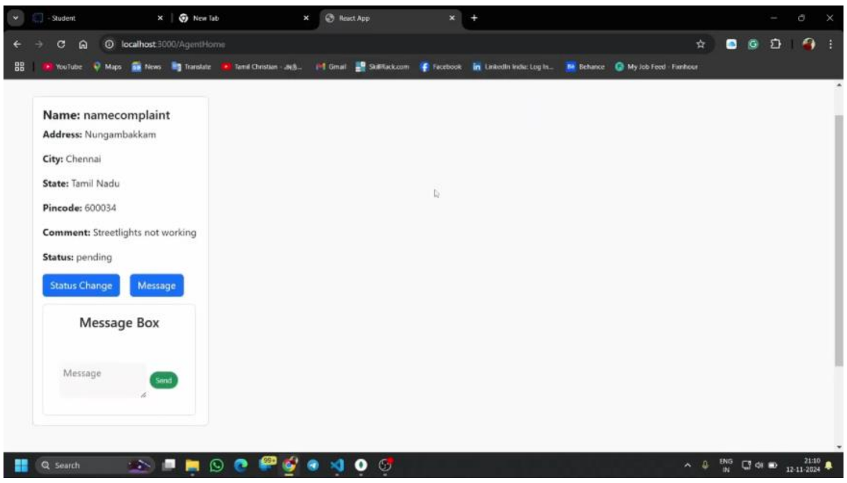

localhost:3000/AdminHome

Hi Admin nameadmin Dashboard User Agent Log out

Users Complaints

Name: namecomplaint
Address: Nungambakkam
City: Chennai
State: Tamil Nadu
Pincode: 600034
Comment: Streetlights not working
Status: pending
Assign

Agents

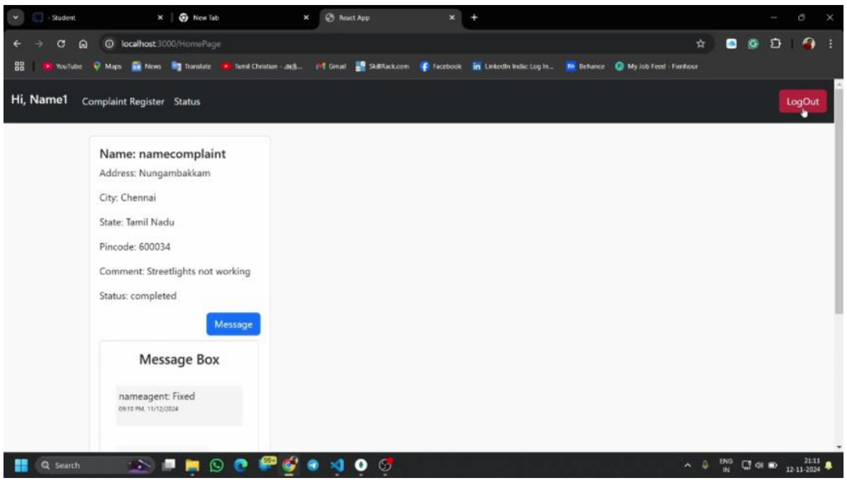

localhost:3000/AgentHome

Name: namecomplaint
Address: Nungambakkam
City: Chennai
State: Tamil Nadu
Pincode: 600034
Comment: Streetlights not working
Status: pending

Status Change Message

Message Box

Message Send


localhost:3000/HomePage

Hi, Name1 Complaint Register Status LogOut

Name: namecomplaint
Address: Nungambakkam
City: Chennai
State: Tamil Nadu
Pincode: 600034
Comment: Streetlights not working
Status: completed

Message

Message Box

nameagent: Fixed
09:10 PM, 11/12/2024

10. Testing

- **Manual Testing:** Manual testing is performed to catch any edge cases that automated tests may miss.
  Test Cases:
    - User Registration: Verify that a new user can successfully register.
    - Complaint Submission: Test that users can submit complaints and receive appropriate notifications.

    - Admin Role: Test that admins can view all complaints and change their status.
    - Authentication: Ensure that JWT tokens work properly for accessing protected routes.

- **Performance Testing:**
    - Test how the system performs when many users submit complaints simultaneously.
    - Measure the response times of critical API endpoints.

  Tools: JMeter or Artillery

11. Screenshots or Demo

https://youtu.be/cK0tUaYTKTE

12. Known Issues

- Inconsistent Layout on Mobile Devices
- Slow Form Validation
- Missing Tooltip or Help Text for Some Form Fields
- Duplicate Complaints

13. Future Enhancements

- Real-time Complaint Updates with WebSockets
- AI-Based Complaint Categorization
- User Profile Management