**Full Stack Development with MERN**

**Project Documentation format**

**1. Introduction**

* **Project Title:** Online Complaint Registration And Management System
* **Team Members:** Bhavitha.G, Sharan.K,Guru Prasaath.D,Janani.RS

**2. Project Overview**

**Purpose:**  
The purpose of this project is to provide a streamlined platform for users to register complaints and for administrators to manage these complaints effectively. The goal is to enhance user satisfaction by ensuring that complaints are handled efficiently and transparently.

**Features:**

* User registration and authentication
* Complaint submission with detailed descriptions
* Viewing and updating complaint statuses
* Admin dashboard for managing complaints
* Role-based access control
* Notifications for status updates

**3. Architecture**

**Frontend:**  
The frontend is developed using React.js. It is responsible for rendering the user interface and handling user interactions. Key libraries include Redux for state management and Axios for HTTP requests.

**Backend:**  
The backend is built with Node.js and Express.js. It provides RESTful APIs for the frontend to interact with, handling business logic, and data validation.

**Database:**  
MongoDB is used as the database to store user data and complaints. Mongoose is utilized to define schemas and interact with the database.

**4. Setup Instructions**

**Prerequisites:**

* Node.js
* MongoDB
* npm or yarn
* vs code

**Installation:**

* 1. **Clone the repository**
  2. **Install dependencies(npm install)**
  3. **Set up environment variables:**
* Create a .env file in the root directory
* Add the following variables:

MONGO\_URI=your\_mongodb\_connection\_string

JWT\_SECRET=your\_jwt\_secret

PORT=5000

.

**5. Folder Structure**

**Client:**

* src/: Main source code directory
  + components/: React components
  + api/: Axios instances for API calls
  + App.js: Main application component
  + index.js: Entry point for the React application

**Server:**

* controllers/: Handles request logic
* models/: Mongoose schemas
* routes/: Defines API endpoints
* middleware/: Custom middleware for authentication
* server.js: Entry point for the Node.js application
* config.js/:connects database

**6. Running the Application**

* Provide commands to start the frontend and backend servers locally.
  + **Frontend:** npm run dev in the client directory.
  + **Backend:** npm start in the server directory.

**7. API Documentation**

**Endpoints:**

* **User Registration:**
  + POST /api/users/register
  + Request: { "name": "John Doe", "email": "john@example.com", "password": "password123" }
  + Response: { "token": "jwt\_token" }
* **User Login:**
  + POST /login
  + Request: { "email": "john@example.com", "password": "password123" }
  + Response: { "token": "jwt\_token" }
* **Create Complaint:**
  + POST /api/complaints
  + Request: { "description": "Description of the complaint", "address": "Address" }
  + Response: {"\_id": "complaint\_id", "description": "Description of the complaint", "address": "Address", "status": "Pending", "assignedTo": null, "createdAt": "timestamp", "updates": []}
* **Get Complaints:**
  + GET /api/complaints
  + Response: [{ "description": "Description of the complaint", "address": "Address", "status": "Pending", "assignedTo": null, "createdAt": "timestamp", "updates": [] }]
* **Update Complaint:**
  + PUT /api/complaints/:id
  + Request: { "status": "In Progress", "assignedTo": "user\_id", "updates": "New update message" }
  + Response: {"\_id": "complaint\_id", "description": "Description of the complaint", "address": "Address", "status": "In Progress", "assignedTo": "user\_id", "createdAt": "timestamp", "updates": [{"message": "New update message", "date": "timestamp"}]}

**8. Authentication**

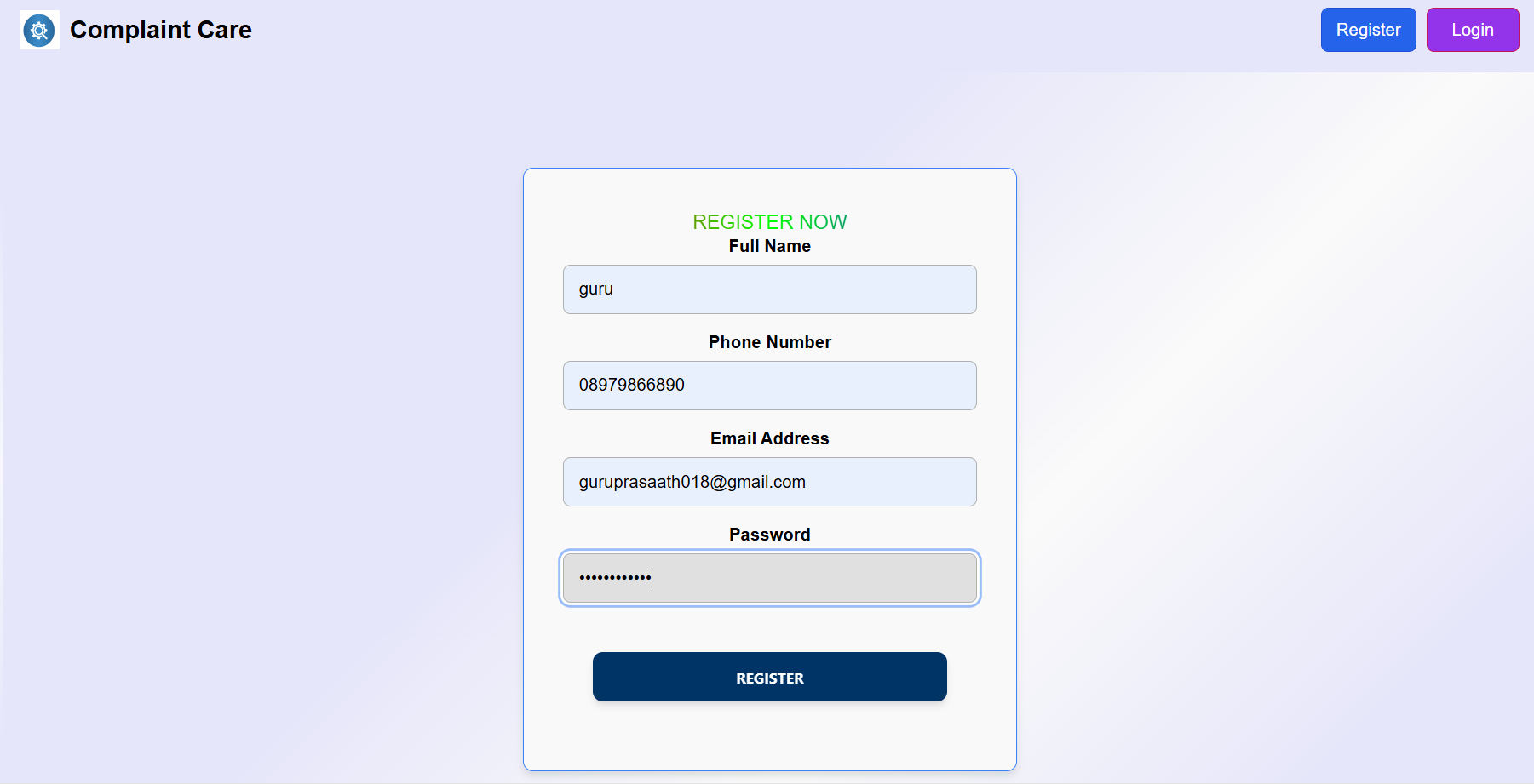
* JWT (JSON Web Token) is used for authentication.
* Upon registration or login, a token is generated and sent to the client.
* This token must be included in the Authorization header of protected routes.
* Middleware (auth.js) verifies the token and grants access to protected resources.

**9. User Interface**

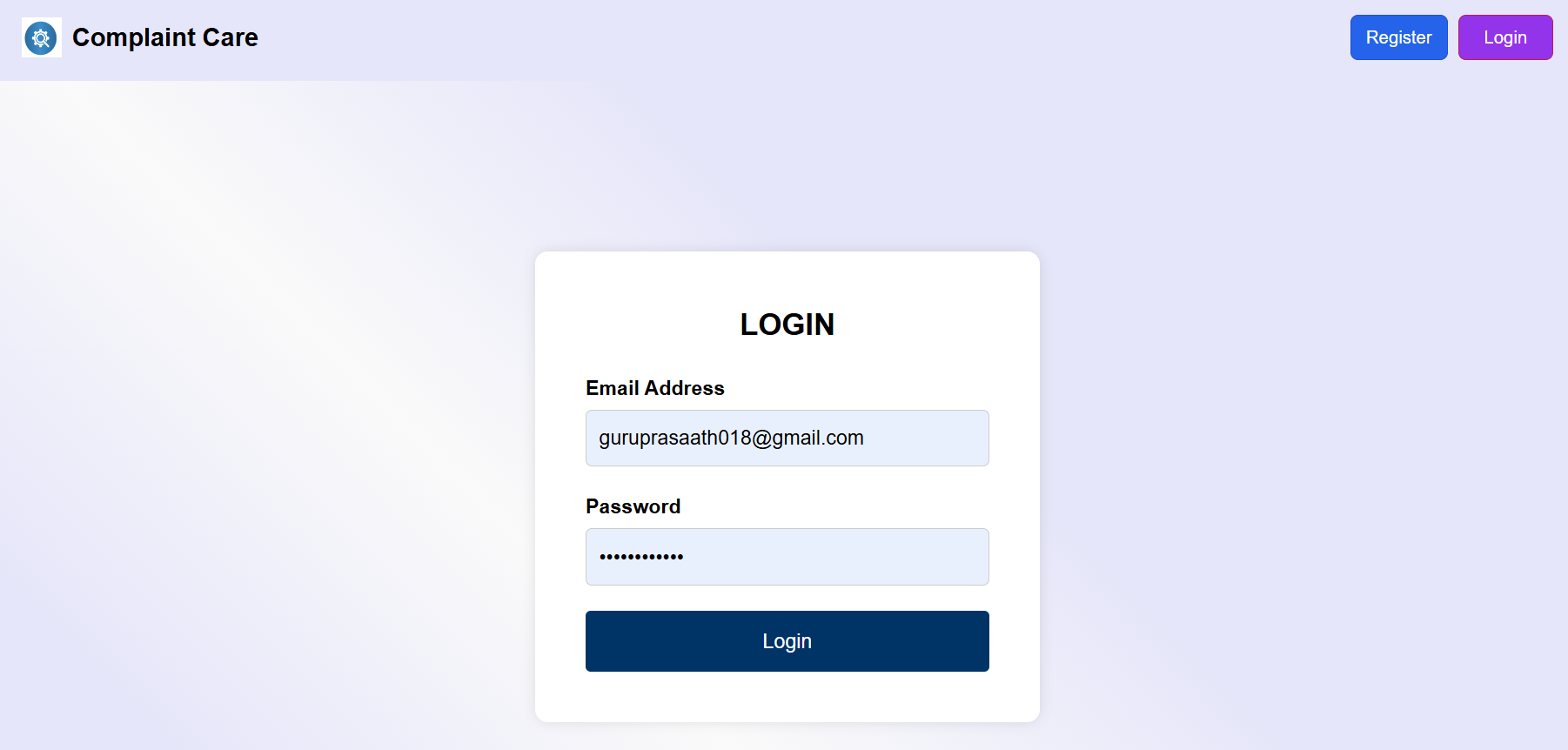
**HOME PAGE:**



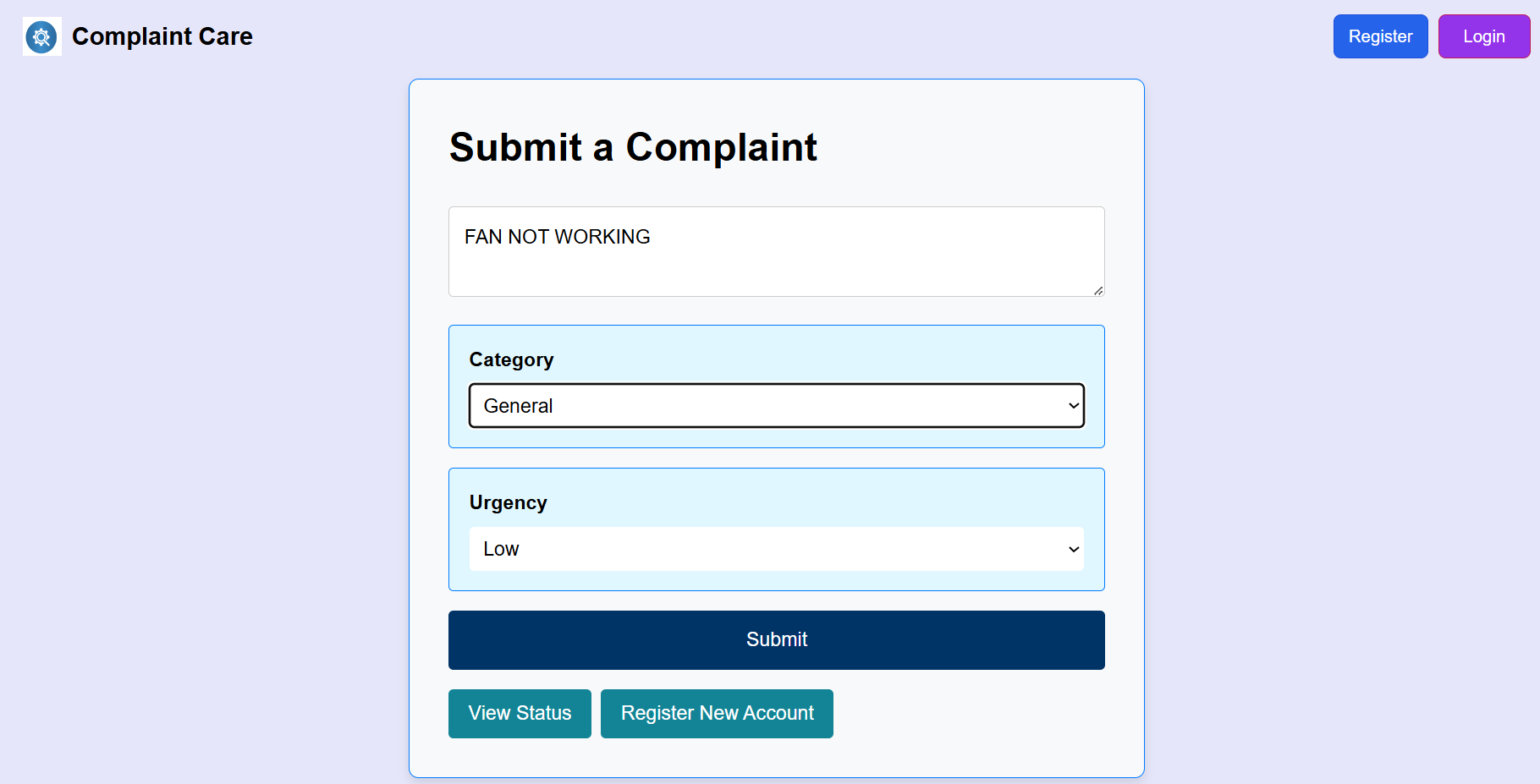
**REGISTRATION PAGE:**



**LOGIN PAGE:**



**COMPLAINT REGISTRATION PAGE:**



**COMPLAINT STATUS PAGE:**



**10. Testing**

* Unit tests for individual functions and components using Jest
* Integration tests for API endpoints using Supertest
* End-to-end tests using Cypress

**11. Screenshots or Demo**

https://github.com/guruprasaathdgp/FINAL.git

**12. Known Issues**

* Occasionally, the app may experience delays in reflecting status updates due to network latency.
* User registration email validation needs improvement to prevent duplicate entries.

**13. Future Enhancements**

* Implement real-time notifications using WebSockets.
* Add a search and filter functionality for complaints.
* Enhance security measures for sensitive data.
* Develop a mobile application version of the system.
* Integrate with third-party services for better complaint tracking and analytics.