**Project Overview**

* **Created a secure login system** for users to access vaccination records.
* **Scalable Architecture** with separate Frontend, Backend, and Database layers.
* **Secure Data Flow** between layers using HTTPS and JWT authentication.
* **Proper Error Handling and Logging** implemented.
* **NoSQL Cloud Database** (MongoDB Atlas) for flexible data storage.
* **Environment Variables** are used to protect sensitive information.

Flow Diagram of System

[User Browser] Like Chrome

↓

(1) React Frontend (Login Page, Dashboard)

↓ (HTTPS Secure API calls)

(2) Node.js Backend (Express APIs)

↓

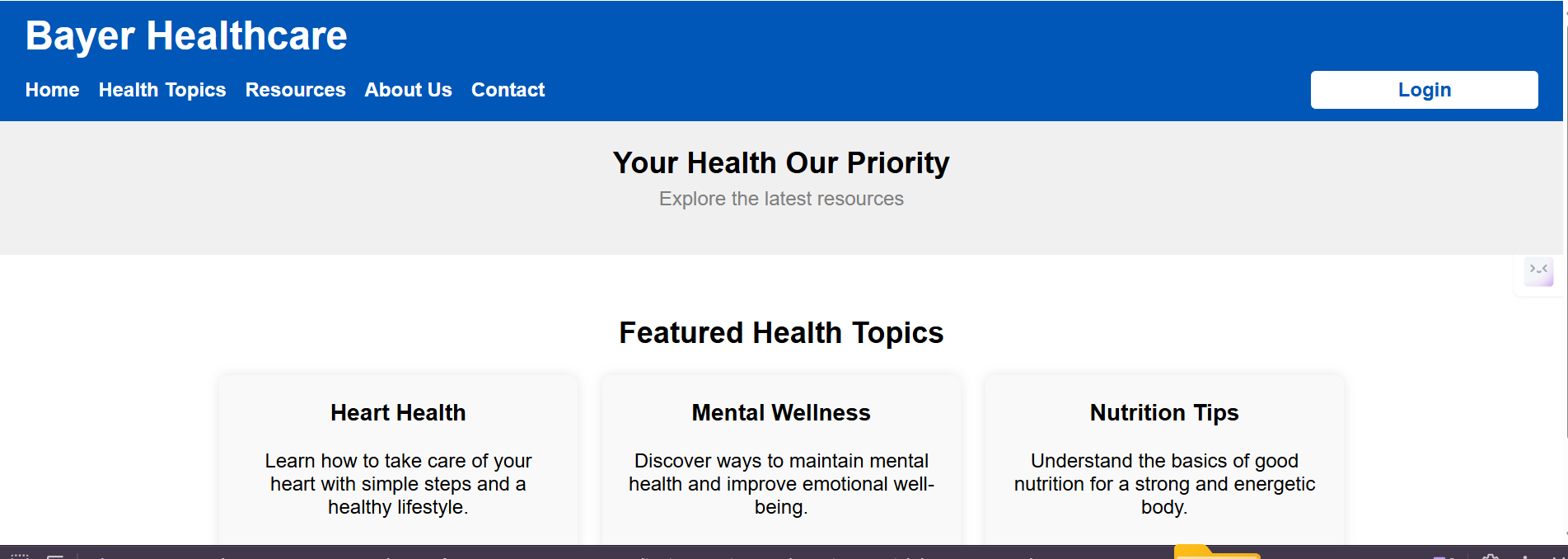
(3) MongoDB Atlas (Cloud Database)

While Developing we tried to take care of these points :

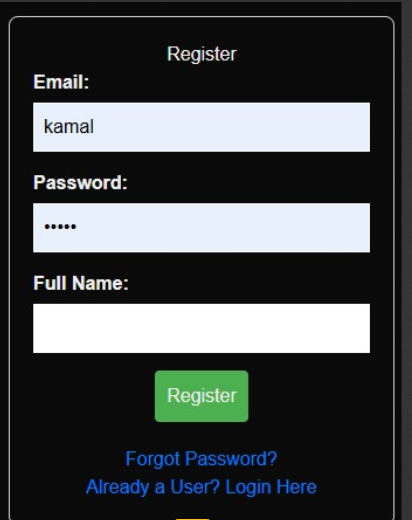
**Development Details**

**Step 1: Frontend Layer (React App)**

* **Home Page:** Simple landing page.



* **Login Page:**
  + Secure email/password form.



* **Dashboard (After Login):**
  + Display Patient Details
  + Display Provider Details
  + Show Vaccination Records
* **Data Fetching:**
  + Secure API communication with Backend.

✅ **Frontend Security:**

* Always use **HTTPS URLs**.
* **Never expose secret keys** or sensitive data in frontend code.
* **JWT Token storage** securely in localStorage with expiration checks.

**Step 2: Backend Layer (Node.js Server)**

* **Express.js APIs:**
  + User Authentication (Login & Token generation).
  + Secure fetching of patient/provider/vaccination data.
* **Request Validation:**
  + All incoming data is validated.
* **Error Handling & Logging:**
  + Wrapped all API logic with try...catch.
  + Consistent error responses for frontend.

**Step 3: Database Layer (MongoDB Atlas)**

* **NoSQL Database (Cloud-based MongoDB Atlas)**.
* **Collections:**
  + patients
  + providers
  + vaccinationRecords
* **Each Document:**
  + Stores user, patient, provider, and vaccination data securely.

✅ **Database Security:**

* Enabled **IP Whitelisting** (only backend server can access DB).
* **Database Encryption** enabled at rest.
* **Strong Passwords** used for database users.

**Step 4: Authentication and Authorization**

* **Login API:**
  + Validate user credentials.
  + Generate **JWT Token** on successful login.
* **Protected Routes:**
  + Only **logged-in users** (valid JWT) can access patient/provider/vaccination data.

**Step 5: HIPAA Compliance (Basic Measures Implemented)**

* All communication over **HTTPS**.
* **Sensitive data encryption** (e.g., hashed passwords).
* **No patient data** printed in server logs.
* (Optional Advanced) **Role-based Access Control (RBAC)** support for future.

**✅ While Developing, We Ensured:**

* **Scalable and maintainable architecture**.
* **Separation of Concerns** between frontend, backend, and database.
* **Security-first approach** at every layer.
* **Clean Code Practices** with modular structure.
* **Future Readiness** for features like audit logs, RBAC, notifications, etc.