



FPT UNIVERSITY

ON-JOB TRAINING PROJECT

Sinh viên ngành Kỹ thuật phần mềm

Report 1 – Tuần 1 và 2

Sinh viên thực hiện: Trần Tấn Phát

Mã số sinh viên: SE183281

Giảng viên hướng dẫn: Ngô Đăng Hà An

– Hồ Chí Minh, Tháng 9 2025 –

Database

1. Mục tiêu

- Làm quen với PostgreSQL và công cụ quản trị **pgAdmin 4**.
- Thiết kế cơ sở dữ liệu phục vụ cho ứng dụng
- Thực hành tạo bảng `Users` với đầy đủ các trường thông tin.
- Thêm dữ liệu mẫu để kiểm thử.

2. Công cụ sử dụng

- **PostgreSQL 16** (Hệ quản trị CSDL quan hệ).
- **pgAdmin 4** (công cụ quản lý CSDL trực quan).
- **SQL Script** để định nghĩa bảng và insert dữ liệu mẫu.

3. Phân tích yêu cầu dữ liệu

Ứng dụng cần quản lý thông tin người dùng, bao gồm:

- Tài khoản đăng nhập (username, email, password).
- Thông tin cá nhân (họ tên, số điện thoại, ngày sinh, avatar).
- Phân quyền (role: user/admin).
- Trạng thái tài khoản (active/inactive).
- Quản lý thời gian tạo & cập nhật.

4. Thiết kế cơ sở dữ liệu

Tên cột	Kiểu dữ liệu	Ràng buộc	Mô tả
Id	SERIAL	PRIMARY KEY	Khóa chính, tự tăng
Username	VARCHAR(50)	UNIQUE, NOT NULL	Tên đăng nhập
Email	VARCHAR(100)	UNIQUE, NOT NULL	Email đăng nhập
Password	VARCHAR(255)	NOT NULL	Mật khẩu (hash)
FullName	VARCHAR(100)	NULL	Họ và tên đầy đủ
Phone	VARCHAR(20)	NULL	Số điện thoại
AvatarUrl	TEXT	NULL	Đường dẫn ảnh đại diện
DateOfBirth	DATE	NULL	Ngày sinh
Role	VARCHAR(20)	DEFAULT 'user'	Vai trò (user/admin)
IsActive	BOOLEAN	DEFAULT TRUE	Trạng thái hoạt động
CreatedAt	TIMESTAMP	DEFAULT now()	Ngày tạo
UpdatedAt	TIMESTAMP	DEFAULT now()	Ngày cập nhật

4.2. Script tạo bảng

Đầu tiên tạo một Database mới:

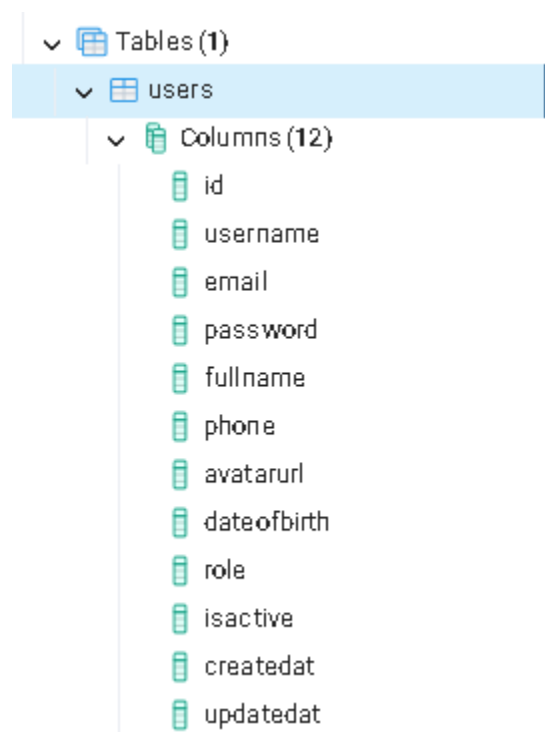
Tạo connection mới:

Sau đó sử dụng script như sau để tạo bảng:

```
DROP TABLE IF EXISTS Users;

CREATE TABLE Users (
    Id SERIAL PRIMARY KEY,
    Username VARCHAR(50) UNIQUE NOT NULL,
    Email VARCHAR(100) UNIQUE NOT NULL,
    Password VARCHAR(255) NOT NULL,
    FullName VARCHAR(100),
    Phone VARCHAR(20),
    AvatarUrl TEXT,
    DateOfBirth DATE,
    Role VARCHAR(20) DEFAULT 'user',
    IsActive BOOLEAN DEFAULT TRUE,
    CreatedAt TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    UpdatedAt TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
```

Sau khi thực thi truy vấn sẽ có bảng mới được tạo ra như hình:



4.3. Script thêm dữ liệu mẫu

```
INSERT INTO Users (  
    Username, Email, Password, FullName, Phone, AvatarUrl,  
    DateOfBirth, Role, IsActive, CreatedAt, UpdatedAt  
)  
  
VALUES  
  
('an01', 'an@example.com', '123456', 'Nguyen Van An',  
'0901234567', 'https://via.placeholder.com/150', '1999-01-15',  
'admin', TRUE, CURRENT_TIMESTAMP, CURRENT_TIMESTAMP),  
  
('khanh02', 'khanh@example.com', '123456', 'Le Khanh',  
'0956789012', 'https://via.placeholder.com/150', '1997-07-18',  
'admin', FALSE, CURRENT_TIMESTAMP, CURRENT_TIMESTAMP),  
  
('binh03', 'binh@example.com', '123456', 'Tran Van Binh',  
'0912345678', 'https://via.placeholder.com/150', '2000-05-20',  
'user', TRUE, CURRENT_TIMESTAMP, CURRENT_TIMESTAMP),  
  
('chi04', 'chi@example.com', '123456', 'Le Thi Chi',  
'0923456789', 'https://via.placeholder.com/150', '2001-09-10',  
'user', TRUE, CURRENT_TIMESTAMP, CURRENT_TIMESTAMP),  
  
('ha05', 'ha@example.com', '123456', 'Do Thi Ha', '0945678901',  
'https://via.placeholder.com/150', '2002-03-22', 'user', FALSE,  
CURRENT_TIMESTAMP, CURRENT_TIMESTAMP),  
  
('lam06', 'lam@example.com', '123456', 'Nguyen Hoang Lam', '0967890123',  
'https://via.placeholder.com/150', '2003-11-11', 'user', TRUE,  
CURRENT_TIMESTAMP, CURRENT_TIMESTAMP);
```

4.4. Script kiểm tra dữ liệu mẫu

Sau đây là một số câu truy vấn để kiểm tra

-Lấy toàn bộ người dùng: `SELECT * FROM Users;`

	id [PK] integer	username character varying (50)	email character varying (100)	password character varying (255)	fullname character varying (100)	phone character varying (20)	avatarurl text	dateofbirth date	role character varying	isactive boolean	createdat timestamp without time zone	updatedat timestamp without time zone
1	1	an01	an@example.com	123456	Nguyen Van An	0901234567	https://via.placeholder.com/150	1999-01-15	admin	true	2025-09-16 11:29:01.552927	2025-09-16 11:29:01.552927
2	2	khanh02	khanh@example.com	123456	Le Khanh	0956789012	https://via.placeholder.com/150	1997-07-18	admin	false	2025-09-16 11:29:01.552927	2025-09-16 11:29:01.552927
3	3	binh03	binh@example.com	123456	Tran Van Binh	0912345678	https://via.placeholder.com/150	2000-05-20	user	true	2025-09-16 11:29:01.552927	2025-09-16 11:29:01.552927
4	4	chi04	chi@example.com	123456	Le Thi Chi	0923456789	https://via.placeholder.com/150	2001-09-10	user	true	2025-09-16 11:29:01.552927	2025-09-16 11:29:01.552927
5	5	ha05	ha@example.com	123456	Do Thi Ha	0945678901	https://via.placeholder.com/150	2002-03-22	user	false	2025-09-16 11:29:01.552927	2025-09-16 11:29:01.552927
6	6	lam06	lam@example.com	123456	Nguyen Hoang Lam	0967890123	https://via.placeholder.com/150	2003-11-11	user	true	2025-09-16 11:29:01.552927	2025-09-16 11:29:01.552927

-Lọc theo role admin: `SELECT * FROM Users WHERE Role = 'admin';`

	id [PK] integer	username character varying (50)	email character varying (100)	password character varying	full name character varying (100)	phone character varying	avatarurl text	dateofbirth date	role character varying (20)	isactive boolean	createdat timestamp without time zone	updatedat timestamp without time zone
1	1	an01	an@example.com	123456	Nguyen Van An	0901234567	https://via.placeholder.com/150	1999-01-15	admin	true	2025-09-16 11:29:01.552927	2025-09-16 11:29:01.552927
2	2	khánh02	khanh@example.com	123456	Le Khanh	0956789012	https://via.placeholder.com/150	1997-07-18	admin	false	2025-09-16 11:29:01.552927	2025-09-16 11:29:01.552927

-Lọc theo user chưa active: `SELECT * FROM Users WHERE IsActive = FALSE;`

	id [PK] integer	username character varying (50)	email character varying (100)	password character varying	full name character varying (100)	phone character varying (20)	avatarurl text	dateofbirth date	role character varying (20)	isactive boolean	createdat timestamp without time zone	updatedat timestamp without time zone
1	2	khánh02	khanh@example.com	123456	Le Khanh	0956789012	https://via.placeholder.com/150	1997-07-18	admin	false	2025-09-16 11:29:01.552927	2025-09-16 11:29:01.552927
2	5	ha05	ha@example.com	123456	Do Thi Ha	0945678901	https://via.placeholder.com/150	2002-03-22	user	false	2025-09-16 11:29:01.552927	2025-09-16 11:29:01.552927

-Lọc theo user sinh sau năm 2000: `SELECT * FROM Users WHERE DateOfBirth >= '2000-01-01';`

	id [PK] integer	username character varying (50)	email character varying (100)	password character varying (255)	full name character varying (100)	phone character varying	avatarurl text	dateofbirth date	role character	isactive boolean	createdat timestamp without time zone	updatedat timestamp without time zone
1	3	binh03	binh@example.com	123456	Tran Van Binh	0912345678	https://via.placeholder.com/150	2000-05-20	user	true	2025-09-16 11:29:01.552927	2025-09-16 11:29:01.552927
2	4	chi04	chi@example.com	123456	Le Thi Chi	0923456789	https://via.placeholder.com/150	2001-09-10	user	true	2025-09-16 11:29:01.552927	2025-09-16 11:29:01.552927
3	5	ha05	ha@example.com	123456	Do Thi Ha	0945678901	https://via.placeholder.com/150	2002-03-22	user	false	2025-09-16 11:29:01.552927	2025-09-16 11:29:01.552927
4	6	lam06	lam@example.com	123456	Nguyen Hoang Lam	0967890123	https://via.placeholder.com/150	2003-11-11	user	true	2025-09-16 11:29:01.552927	2025-09-16 11:29:01.552927

5. Kết quả đạt được

- Tạo thành công database và bảng `Users`.
- Insert dữ liệu mẫu đầy đủ với cả **admin** và **user**.
- Kiểm thử thành công các truy vấn cơ bản.
- Sẵn sàng tích hợp với **backend API (.NET)** ở bước tiếp theo.

Back-end

1. Tạo mới dự án backend

- ✓ .NET 8.0 (LTS)
- ✓ Authentication = None
- ✓ HTTPS = ☒
- ✓ OpenAPI = ☒
- ✓ Use controllers = ☒
- ✗ Container support = OFF
- ✗ Do not use top-level statements = OFF
- ✗ .NET Aspire orchestration = OFF

Additional information

ASP.NET Core Web API C# Linux macOS Windows API Cloud Service Web Web API

Framework ⓘ
[.NET 8.0 (Long Term Support)]

Authentication type ⓘ
[None]

☒ Configure for HTTPS ⓘ
☐ Enable container support ⓘ

Container OS ⓘ
[Linux]

Container build type ⓘ
[Dockerfile]

☒ Enable OpenAPI support ⓘ
☐ Do not use top-level statements ⓘ
☒ Use controllers ⓘ
☐ Enlist in .NET Aspire orchestration ⓘ

Aspire version ⓘ
[9.2]

[Back] [Create]

2. Kết nối dự án Back-end với Database

Bước 1: +Cài package NuGet

Tools → NuGet Package Manager → Package Manager Console.

+Dán 2 dòng này vào đây là thư viện để EF Core làm việc với PostgreSQL.:

Install-Package Microsoft.EntityFrameworkCore.Tools

Install-Package Npgsql.EntityFrameworkCore.PostgreSQL

Install-Package Microsoft.EntityFrameworkCore.Design

Install-Package Npgsql.EntityFrameworkCore.PostgreSQL.Design

Bước 2: Cấu hình Connection String

+Mở file appsettings.json.

+Thêm connection string:

"ConnectionStrings": {


```
"DefaultConnection":  
"Host=localhost;Port=5432;Database=ojt_db;Username=postgres;Password=12345"  
}
```

Bước 3: Scaffold models từ DB

```
Scaffold-DbContext  
"Host=localhost;Port=5432;Database=ojt_db;Username=postgres;Password=12345"  
Npgsql.EntityFrameworkCore.PostgreSQL -OutputDir Models
```

Sơ lược về code

- **Program.cs**: cấu hình dịch vụ cho ứng dụng, bao gồm Swagger, CORS, và kết nối DbContext với PostgreSQL.
- **Models/**: chứa các class được scaffold từ database, ví dụ User.cs, OjtDbContext.cs. Đây là nơi định nghĩa bảng Users với các trường như Username, Email, Password, Role...
- **Controllers/**: chứa các controller xử lý API:
 - AuthController: có API /login dùng để xác thực người dùng.
 - UsersController: cung cấp các API CRUD (GET, POST, PUT, DELETE) cho bảng Users.
- **Kết quả chạy**: API có thể test bằng Swagger hoặc Postman, trả dữ liệu JSON từ PostgreSQL.

Kết quả đạt được

- Kết nối thành công dự án ASP.NET Core với PostgreSQL qua Entity Framework Core.
- Scaffold thành công DbContext và Models từ bảng Users.
- Xây dựng được API cơ bản:
 - **Login API** kiểm tra username/password.
 - **CRUD API** cho bảng Users (GET, POST, PUT, DELETE).
- Test API bằng Postman thành công, dữ liệu trả về đúng với Database.
- Cấu hình Swagger UI để kiểm thử API trực tiếp trong trình duyệt.
- Cấu hình CORS để ReactJS frontend gọi API không bị chặn.

Những thứ học được

- Biết cách tạo dự án Web API bằng ASP.NET Core.
- Hiểu cách dùng Entity Framework Core để kết nối và thao tác với PostgreSQL.
- Nắm được cách tổ chức Controllers, Models trong kiến trúc Web API.
- Hiểu về middleware (Swagger, CORS, Authorization).

Những thứ chưa làm được

- Chưa có xác thực nâng cao bằng **JWT Token** (mới dừng ở mức kiểm tra username/password).
- Chưa triển khai upload ảnh avatar (mới lưu đường dẫn URL).

Front-end

Bước 1: Tạo project React

Trong terminal chạy những câu lệnh sau:

```
npx create-react-app frontend-app
```

Bước 2: Cài thêm axios (nếu muốn gọi API dễ hơn)

```
npm install axios
```

Bước 3: Cài thêm router dom để quản lý đường dẫn

```
npm install react-router-dom
```

Sơ lược về code

- **App.js**: định nghĩa routing cho toàn bộ ứng dụng, gồm /login, /admin, /user, /unauthorized.
- **pages/**: chứa các trang chính như LoginPage.js, AdminPage.js, UserPage.js.
- **services/**: chứa authService.js để gọi API backend (login, CRUD user).
- **context/**: chứa AuthContext.js quản lý trạng thái đăng nhập và thông tin user (role, token).
- **components/**: chứa PrivateRoute.js để kiểm tra quyền truy cập vào route.
- **Luồng hoạt động**: người dùng nhập username/password → gọi API login → lưu thông tin vào Context + localStorage → điều hướng đến trang phù hợp với role.

Kết quả đạt được

- Khởi tạo thành công project ReactJS.
- Xây dựng form **Login** kết nối với API backend.
- Lưu thông tin người dùng vào localStorage và quản lý qua Context API.
- Thực hiện **phân quyền (role-based routing)**:
 - Admin → /admin, User → /user.
 - Nếu chưa login hoặc sai role → bị chặn và điều hướng.
- Gọi API bằng **axios** để lấy dữ liệu từ backend.
- Tạo cấu trúc thư mục rõ ràng (pages/, services/, components/, context/).

Những thứ học được

- Hiểu cách khởi tạo và tổ chức dự án ReactJS.
- Biết cách gọi API từ frontend bằng axios.
- Biết cách quản lý trạng thái đăng nhập bằng Context API.
- Nắm được cơ chế **routing** và **PrivateRoute** trong React.

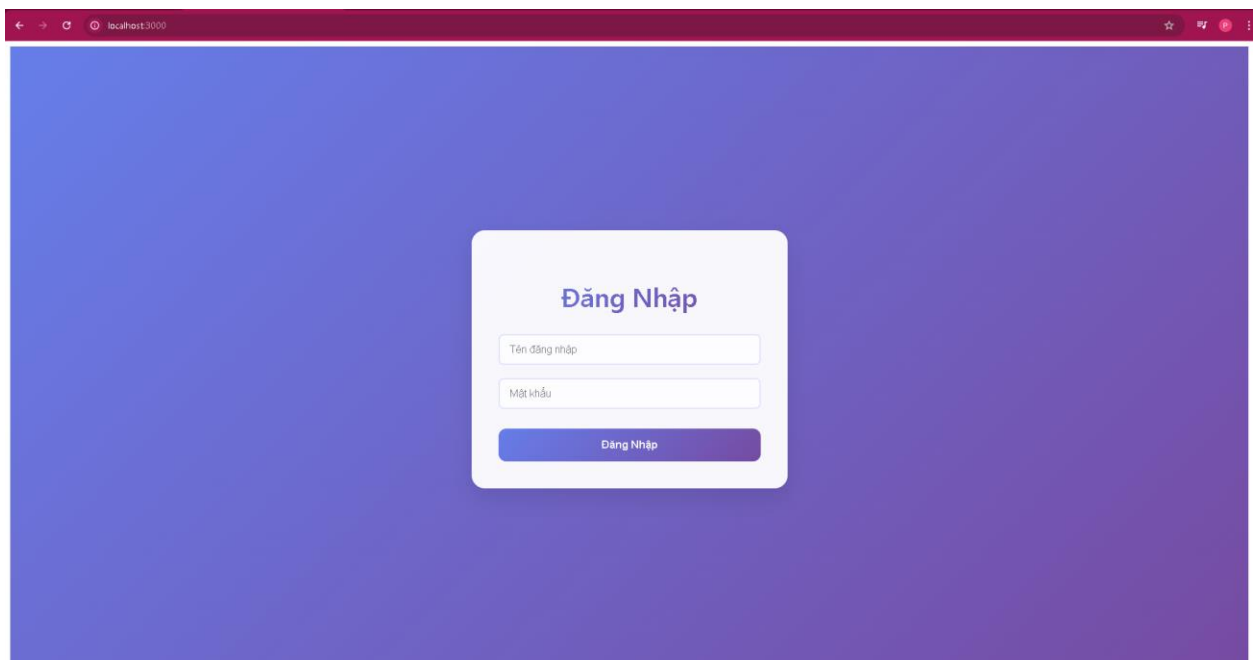
Những thứ chưa làm được

- Chưa có giao diện nâng cao (chỉ dùng form và trang cơ bản).
- Chưa có CRUD User ở frontend (mới login và phân quyền).
- Chưa có tính năng upload avatar từ frontend.

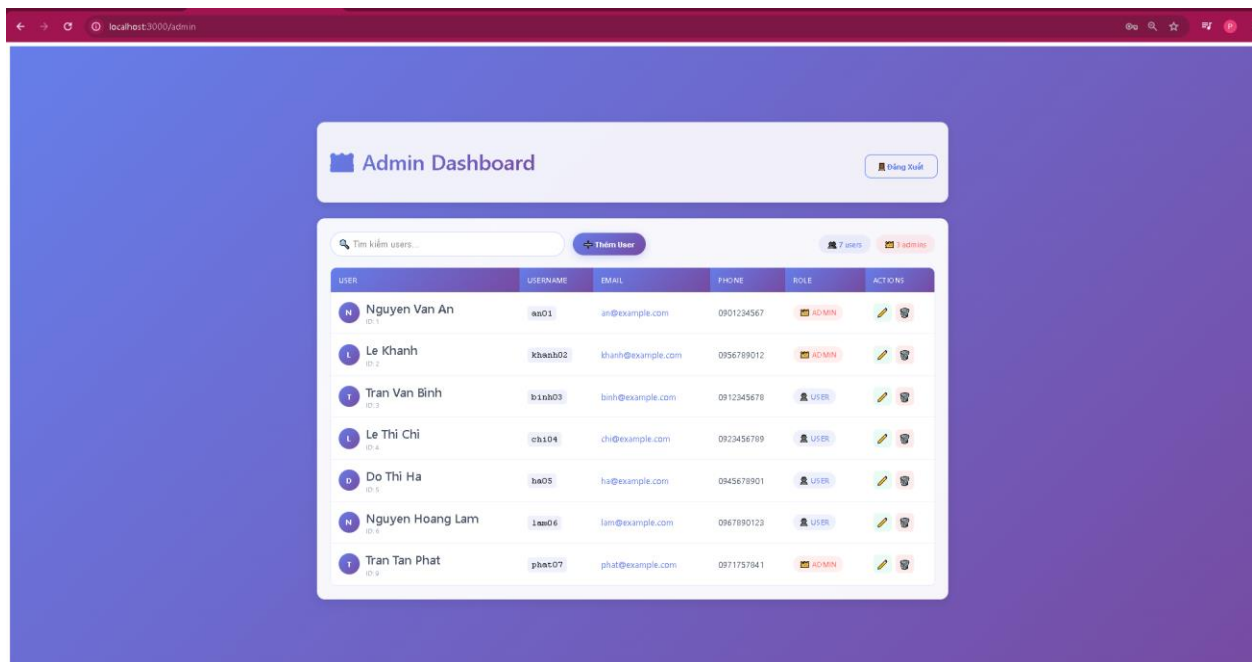
Tổng thể ứng dụng

UI

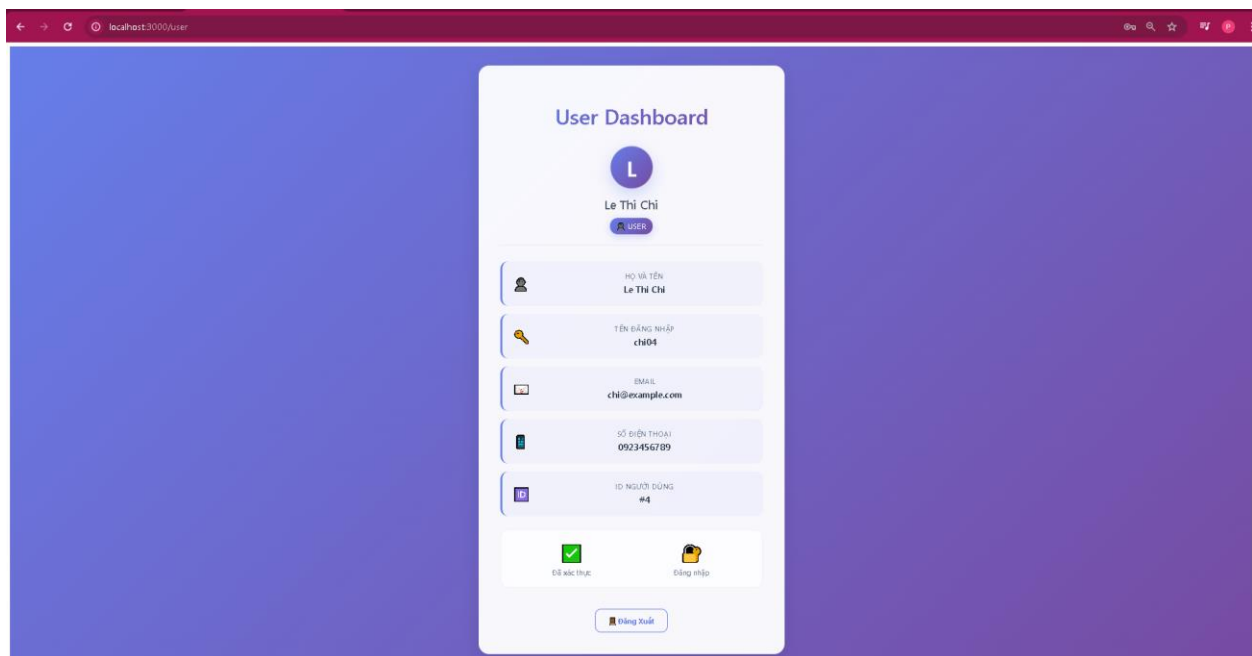
- Giao diện ban đầu: đơn giản, chỉ có trang **Login** để minh họa kết nối React ↔ API ↔ Database.



- Sau khi đăng nhập:
 - **Admin** được điều hướng đến trang quản lý user (Admin Page).



- User được điều hướng đến trang thông tin cá nhân (User Page).



API Backend

Trong quá trình phát triển, nhóm đã xây dựng và triển khai thành công hệ thống API với các chức năng chính sau:

Auth

POST /api/Auth/login

Users

GET /api/Users

POST /api/Users

GET /api/Users/{id}

PUT /api/Users/{id}

DELETE /api/Users/{id}

Schemas

LoginRequest >

User >

1. Auth (Xác thực)

- **POST /api/Auth/login**
 - Cho phép người dùng đăng nhập vào hệ thống bằng username và password.
 - Kiểm tra thông tin từ Database, trả về thông tin người dùng và role (user / admin).
 - Đây là bước quan trọng để phân quyền trên frontend.

2. Users (Quản lý người dùng)

- **GET /api/Users**
 - Lấy danh sách toàn bộ người dùng trong hệ thống.
- **POST /api/Users**
 - Thêm mới một người dùng.
- **GET /api/Users/{id}**
 - Lấy thông tin chi tiết của một người dùng theo id.
- **PUT /api/Users/{id}**
 - Cập nhật thông tin người dùng theo id.
- **DELETE /api/Users/{id}**
 - Xóa người dùng theo id.

Sử dụng Git & GitHub trong dự án

Để quản lý mã nguồn, **Git** và lưu trữ dự án trên **GitHub** được sử dụng tại repo cá nhân:

🔗 [fullstack-training-project-ojt-fa25](https://github.com/fullstack-training-project-ojt-fa25)

- **Mục đích sử dụng Git/GitHub:**
 - Lưu trữ toàn bộ mã nguồn backend, frontend, database và báo cáo.
 - Theo dõi lịch sử thay đổi code.
 - Dễ dàng khôi phục khi có lỗi phát sinh.
- **Một số lệnh Git cơ bản đã sử dụng:**
 - `git clone <repo>` # Tải dự án về máy Theo dõi lịch sử thay đổi code.
 - `git add .` # Thêm thay đổi.
 - `git commit -m "Nội dung commit"` # Ghi lại thay đổi
 - `git push origin main` # Đẩy code lên GitHub
 - `git pull origin main` # Lấy code mới nhất từ GitHub
- **Quản lý dự án:**
 - Thực hiện commit sau mỗi phần quan trọng (ví dụ: hoàn thành API login, kết nối database, hoàn thành giao diện login ReactJS).
 - Dự án được tổ chức theo từng **MOOC** để dễ dàng theo dõi tiến độ học tập.