**# Hướng dẫn về Dự án HIV Treatment and Medical Services System**

Chào các thành viên nhóm! Dưới đây là giải thích chi tiết về dự án HIV Treatment and Medical Services mà chúng ta sẽ thực hiện. Tôi sẽ giải thích những gì đã được cấu hình và làm thế nào để tiếp tục phát triển dự án.

**## Cấu trúc dự án**

Dự án của chúng ta được chia thành 2 phần chính:

**### 1. Backend (Phía máy chủ)**

Backend được xây dựng bằng .NET 8 với kiến trúc Clean Architecture, gồm các dự án con:

- \*\*HIVSystem.API\*\*: Cung cấp các API endpoints cho frontend gọi đến

- \*\*HIVSystem.Core\*\*: Chứa logic nghiệp vụ cốt lõi và các interfaces

- \*\*HIVSystem.Infrastructure\*\*: Chứa code làm việc với cơ sở dữ liệu và các dịch vụ bên ngoài

- \*\*HIVSystem.Tests\*\*: Dành cho kiểm thử (unit tests và integration tests)

**### 2. Frontend (Giao diện người dùng)**

Frontend được xây dựng bằng React, với cấu trúc cơ bản:

- \*\*public\*\*: Chứa các file tĩnh như index.html

- \*\*src\*\*: Chứa mã nguồn React

## Những gì đã được cấu hình

**### Backend**

1. \*\*Cấu trúc dự án .NET\*\*: Solution và các project đã được tạo theo mô hình Clean Architecture

2. \*\*Database Context\*\*: Đã thiết lập `AppDbContext` kết nối với SQL Server

3. \*\*Dependency Injection\*\*: Hệ thống DI để quản lý các services

4. \*\*API Controllers\*\*: Controller cơ bản và health check để kiểm tra hệ thống

5. \*\*Repository Pattern\*\*: Interface cơ bản để làm việc với dữ liệu

**### Frontend**

1. \*\*React App cơ bản\*\*: Setup React với một trang đơn giản

2. \*\*Package.json\*\*: Cấu hình với các dependency cần thiết

3. \*\*Public folder\*\*: Cấu hình HTML và manifest

**### Khác**

1. \*\*.gitignore\*\*: Đã cấu hình để loại trừ các file không cần thiết khi đưa lên git

2. \*\*README.md\*\*: Hướng dẫn cài đặt và sử dụng dự án

**## Các công nghệ sử dụng và cần học**

**### Backend**

1. \*\*C#\*\*: Ngôn ngữ chính để phát triển backend

2. \*\*.NET 8\*\*: Framework mới nhất của Microsoft

3. \*\*Entity Framework Core\*\*: ORM để làm việc với cơ sở dữ liệu

4. \*\*SQL Server\*\*: Hệ quản trị cơ sở dữ liệu

5. \*\*RESTful API\*\*: Kiến thức về cách thiết kế API

6. \*\*Dependency Injection\*\*: Hiểu cách DI hoạt động trong .NET

7. \*\*Clean Architecture\*\*: Kiến trúc phân lớp giúp code dễ bảo trì

**### Frontend**

1. \*\*JavaScript/TypeScript\*\*: Ngôn ngữ để viết React

2. \*\*React\*\*: Thư viện UI

3. \*\*React Router\*\*: Quản lý định tuyến trang

4. \*\*Axios\*\*: Thư viện để gọi API

5. \*\*HTML/CSS\*\*: Kiến thức nền tảng để xây dựng UI

## Hướng dẫn phát triển tiếp theo

**### Các bước tiếp theo cho Backend**

**1. \*\*Tạo Domain Models\*\*:**

- Tạo các lớp entity trong HIVSystem.Core/Entities như: Patient, Doctor, Treatment, Appointment, Medicine...

- Ví dụ:

```csharp

public class Patient

{

public int Id { get; set; }

public string FullName { get; set; }

public DateTime DateOfBirth { get; set; }

public string Gender { get; set; }

public string ContactNumber { get; set; }

public string Email { get; set; }

// Các thuộc tính khác

}

```

**2. \*\*Cấu hình DbContext\*\*:**

- Thêm DbSet cho mỗi entity vào AppDbContext

- Ví dụ:

```csharp

public DbSet<Patient> Patients { get; set; }

public DbSet<Doctor> Doctors { get; set; }

```

**3. \*\*Tạo Repository\*\*:**

- Triển khai interface IRepository cho mỗi entity

- Ví dụ:

```csharp

public class PatientRepository : IRepository<Patient>

{

// Code triển khai các phương thức

}

```

**4. \*\*Tạo Services\*\*:**

- Tạo các service xử lý logic nghiệp vụ

- Ví dụ:

```csharp

public interface IPatientService

{

Task<Patient> GetPatientByIdAsync(int id);

// Các phương thức khác

}

```

**5. \*\*Tạo Controllers\*\*:**

- Tạo API endpoints để frontend có thể gọi

- Ví dụ:

```csharp

[Route("api/patients")]

public class PatientsController : BaseApiController

{

private readonly IPatientService \_patientService;

// Các action methods: Get, Post, Put, Delete

}

```

**6. \*\*Tạo Authentication\*\*:**

- Cài đặt JWT để xác thực người dùng

### Các bước tiếp theo cho Frontend

**1. \*\*Cấu trúc thư mục\*\*:**

- components: Các component tái sử dụng

- pages: Các trang chính

- services: Các service gọi API

- utils: Tiện ích

**2. \*\*Tạo Router\*\*:**

- Cấu hình React Router để điều hướng giữa các trang

**3. \*\*Tạo Layout\*\*:**

- Tạo layout chung cho toàn bộ ứng dụng (header, sidebar, footer)

**4. \*\*Các trang cơ bản\*\*:**

- Đăng nhập/Đăng ký

- Trang chủ/Dashboard

- Quản lý bệnh nhân

- Quản lý lịch hẹn

- Quản lý thuốc

**5. \*\*State Management\*\*:**

- Sử dụng React Context hoặc Redux để quản lý trạng thái

## Lời khuyên cho người mới

### Nếu bạn làm Backend:

1. \*\*Học C# cơ bản\*\*: Cú pháp, kiểu dữ liệu, hàm, lớp, interface

2. \*\*Học Entity Framework Core\*\*: Cách làm việc với database

3. \*\*Học ASP.NET Core\*\*: Tạo API endpoints, routing, controller

4. \*\*Học về xác thực JWT\*\*: Đăng nhập, đăng ký, quản lý token

### Nếu bạn làm Frontend:

1. \*\*Học JavaScript cơ bản\*\*: Biến, hàm, loop, array, object

2. \*\*Học React cơ bản\*\*: Components, props, state, hooks

3. \*\*Học React Router\*\*: Điều hướng trang

4. \*\*Học cách gọi API\*\*: Sử dụng fetch hoặc axios

## Tài nguyên học tập

### Backend:

- Microsoft Learn: https://learn.microsoft.com/vi-vn/dotnet/

- W3Schools C#: https://www.w3schools.com/cs/

- Entity Framework Core: https://learn.microsoft.com/vi-vn/ef/core/

### Frontend:

- W3Schools React: https://www.w3schools.com/react/

- React Official Docs: https://reactjs.org/docs/getting-started.html

**## Quy trình làm việc**

1. Clone dự án từ GitHub

2. Tạo nhánh mới cho tính năng bạn làm

3. Viết code và test

4. Push code và tạo pull request

5. Review và merge vào nhánh chính

\*\*Lưu ý\*\*: Nhớ luôn pull code mới nhất từ nhánh chính trước khi bắt đầu làm việc để tránh xung đột.

## Kết luận

Đây là khởi đầu tốt cho dự án của chúng ta. Dù có vẻ phức tạp ban đầu, nhưng các ae sẽ nhanh chóng làm quen với cấu trúc dự án. Nếu có bất kỳ câu hỏi nào, hãy liên hệ với tôi để được hỗ trợ.

Chúc các ae thành công và cùng nhau xây dựng một dự án tốt!