TRƯỜNG ĐẠI HỌC BÁCH KHOA
ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
KHOA KHOA HỌC VÀ KỸ THUẬT MÁY TÍNH

CÔNG NGHỆ PHẦN MỀM (CO3001)

BÀI TẬP LỚN

Hệ thống hỗ trợ Tutor tại Trường Đại học Bách khoa Đại học Quốc gia - TP.HCM

Giáo viên hướng dẫn: Phan Trung Hiếu, CSE-HCMUT

Sinh viên - Nhóm 3:
Nguyễn Tấn Phát - 2352888 *(CN01)*
Vũ Hà Như Ngọc - 2352818 *(CN01)*
Lê Diệu Quỳnh - 2353036 *(CN01)*
Mã Nhật Tiến - 2353178 *(CN01)*
Bùi Phan Khánh Duy - 2352170 *(CN01)*
Lương Đức Huy - 2352384 *(CN01)*
Nguyễn Ngọc Phát - 2352887 *(CN01)*
Văn Bá Trọng Khiêm - 2352546 *(CN01)*

THÀNH PHỐ HỒ CHÍ MINH, THÁNG 9 NĂM 2025

Mục lục

Danh sách Ký hiệu

Danh sách Từ viết tắt

Bảng 1: Danh sách Từ viết tắt

STT	Chữ viết tắt	Chữ viết đầy đủ
1	AI	Trí tuệ nhân tạo (Artificial Intelligence)
2	APP	Application (Ứng dụng)
3	BM	Bộ môn
4	DB	Database (Cơ sở dữ liệu)
5	DOCX	Microsoft Word Document (Tài liệu Microsoft Word)
6	DRP	Disaster Recovery Plan (Kế hoạch khôi phục sau thảm họa)
7	Excel	Microsoft Excel (Phần mềm bảng tính)
8	GPA	Điểm trung bình học tập (Grade Point Average)
9	HCMUT_DATACORE	Lõi dữ liệu Đại học Bách khoa
10	HCMUT_LIBRARY	Thư viện Đại học Bách khoa
11	HTTPS	Hypertext Transfer Protocol Secure (Giao thức truyền tải siêu văn bản an toàn)
12	ID	Identifier (Mã định danh)
13	iOS	iPhone Operating System (Hệ điều hành của iPhone)
14	MB	Megabyte
15	MSSV	Mã số sinh viên
16	MVC	Model-View-Controller (Một mẫu kiến trúc phần mềm)
17	NCS	Nghiên cứu sinh
18	OTP	One-Time Password (Mật khẩu dùng một lần)
19	PCTSV	Phòng Công tác Sinh viên
20	PDF	Portable Document Format (Định dạng tài liệu di động)
21	PĐT	Phòng Đào tạo
22	PPT	PowerPoint Presentation (Bài trình chiếu PowerPoint)
23	SĐT	Số điện thoại
24	SMS	Short Message Service (Dịch vụ tin nhắn ngắn)
25	SV	Sinh viên
26	TLS	Transport Layer Security (Bảo mật tầng truyền tải)
27	UC	Use Case (Ca sử dụng)
28	US	User Story (Câu chuyện người dùng)
29	2FA	Two-Factor Authentication (Xác thực hai yếu tố)

Danh sách Hình ảnh

Danh sách Bảng

Danh sách thành viên & khối lượng công việc

STT	Họ Tên	MSSV	Vai trò	% Hoàn thành
1	Nguyễn Tấn Phát	2352888	Scrum Master	100%
2	Vũ Hà Như Ngọc	2352818	Business Analyst	100%
3	Lê Diệu Quỳnh	2353036	Product Owner	100%
4	Mã Nhật Tiến	2353178	Quality Controller	100%
5	Bùi Phan Khánh Duy	2352170	Techlead + SysAdmin	100%
6	Lương Đức Huy	2352384	Developer	100%
7	Nguyễn Ngọc Phát	2352887	Developer	100%
8	Văn Bá Trọng Khiêm	2352546	Developer	100%

Bảng 2: Danh sách thành viên & khối lượng công việc

4. Triển khai hệ thống

4.1. Sơ đồ triển khai (Deployment View)

4.1.1 Giới thiệu

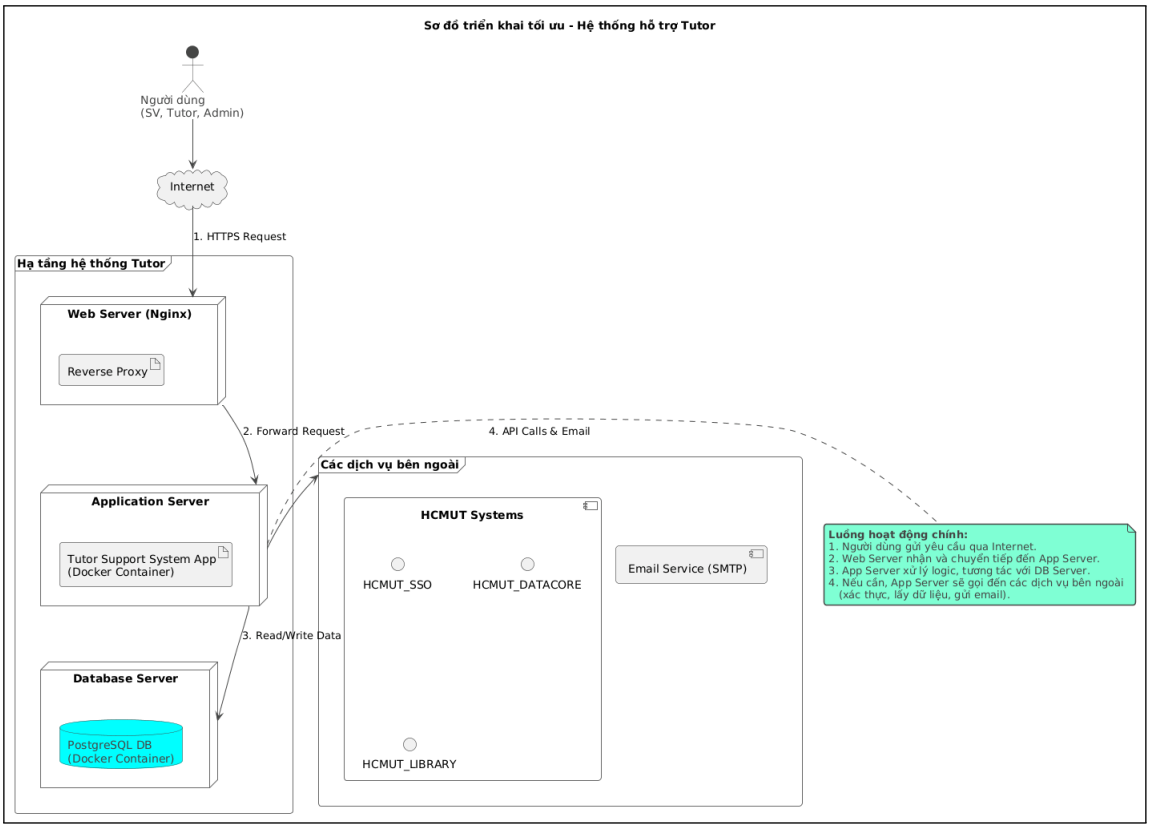
Sơ đồ triển khai (Deployment Diagram) mô tả kiến trúc vật lý của "Hệ thống hỗ trợ Tutor", thể hiện cách các thành phần phần mềm được phân bổ và vận hành trên các nút (node) phần cứng. Sơ đồ này cung cấp một cái nhìn tổng quan về môi trường thực thi của hệ thống, bao gồm các máy chủ, cơ sở dữ liệu, và sự tương tác giữa chúng cũng như với các hệ thống bên ngoài. Kiến trúc được lựa chọn là mô hình **client-server ba lớp (3-tier)** hiện đại, bao gồm:

- **Presentation Tier (Client)**: Giao diện người dùng trên trình duyệt web.
- **Application Tier (Server)**: Máy chủ ứng dụng xử lý logic nghiệp vụ.
- **Data Tier (Database)**: Máy chủ cơ sở dữ liệu để lưu trữ và quản lý dữ liệu.

Mô hình này đảm bảo tính linh hoạt, khả năng mở rộng và bảo mật cho hệ thống.

4.1.2. Sơ đồ triển khai hệ thống

Sơ đồ dưới đây được tạo bằng PlantUML, mô tả chi tiết các thành phần và luồng tương tác.



Hình 1: Sơ đồ triển khai hệ thống hỗ trợ Tutor

4.1.3. Mô tả các thành phần

STT	Tên thành phần	Mô tả	Công nghệ/Phần mềm
1	Người dùng	Các tác nhân (Sinh viên, Tutor, Admin) tương tác với hệ thống thông qua trình duyệt web	Trình duyệt Web (Chrome, Firefox, Safari)
2	Web Server (Nginx)	Đóng vai trò là Reverse Proxy, tiếp nhận yêu cầu từ Internet và chuyển tiếp đến Application Server. Tăng cường bảo mật và cân bằng tải.	Nginx
3	Application Server	Chứa toàn bộ logic nghiệp vụ của hệ thống, được đóng gói trong Docker container để dễ dàng triển khai và quản lý.	Java Spring Boot / Node.js, Docker
4	Database Server	Chịu trách nhiệm lưu trữ và quản lý toàn bộ dữ liệu của hệ thống, được triển khai trong Docker container.	PostgreSQL / MySQL, Docker
5	Hệ thống của HCMUT	Các dịch vụ công nghệ thông tin tập trung của trường mà hệ thống cần tích hợp, bao gồm: HCMUT_SSO , HCMUT_DATACORE, HCMUT_LIBRARY .	API (REST/SOAP)
6	Email Service	Dịch vụ bên ngoài chịu trách nhiệm gửi các thông báo và nhắc nhở qua email cho người dùng.	SMTP Server (ví dụ: SendGrid, AWS SES)

Bảng 3: Các thành phần trong sơ đồ triển khai hệ thống hỗ trợ Tutor

4.1.4. Luồng hoạt động

Để làm rõ hơn sự tương tác giữa các thành phần, dưới đây là mô tả luồng hoạt động của hai kịch bản tiêu biểu:

A. **Kịch bản 1: Người dùng đăng nhập vào hệ thống**

A.1. Người dùng mở trình duyệt, truy cập vào địa chỉ của hệ thống và nhấn nút đăng nhập thông qua **HCMUT_SSO**.

A.2. Yêu cầu (*HTTPS Request*) được gửi qua Internet đến **Web Server (Nginx)**.

A.3. **Nginx** giải mã SSL và chuyển tiếp yêu cầu đến **Application Server**.

A.4. **Application Server** nhận yêu cầu, chuyển hướng người dùng đến trang đăng nhập của **HCMUT_SSO**.

A.5. Sau khi người dùng đăng nhập thành công trên **HCMUT_SSO**, dịch vụ này sẽ trả về một *token* xác thực cho **Application Server**.

A.6. **Application Server** sử dụng *token* này để gọi API đến **HCMUT_DATACORE**, lấy và đồng bộ thông tin cơ bản của người dùng (họ tên, MSSV, khoa, vai trò) vào **Database Server**.

A.7. Cuối cùng, **Application Server** trả về một phiên làm việc (*session*) và giao diện trang chủ cho người dùng.

B. **Kịch bản 2: Tutor tạo lịch rảnh mới**

B.1. **Tutor** sau khi đăng nhập, truy cập chức năng "Tạo lịch rảnh" và điền thông tin (ngày, giờ).

B.2. Yêu cầu tạo lịch được gửi qua Internet, đi qua **Web Server** và đến **Application Server**.

B.3. **Application Server** xác thực quyền của Tutor, kiểm tra tính hợp lệ của dữ liệu (ví dụ: không trùng lịch đã có).

B.4. Nếu hợp lệ, **Application Server** sẽ thực hiện một câu lệnh ghi (`INSERT`) để lưu thông tin lịch rảnh mới vào bảng `Schedules` trong **Database Server**.

B.5. **Database Server** xác nhận ghi thành công.

B.6. **Application Server** gửi lại một thông báo "Tạo lịch thành công" cho giao diện của Tutor.

C. **Kịch bản 3: Sinh viên đặt lịch học cố định với Tutor**

C.1. Sinh viên, sau khi đăng nhập, chọn môn học và xem các khung giờ rảnh (*availability*) mà Tutor đã thiết lập. Sinh viên chọn một khung giờ phù hợp và nhấn "Đặt lịch".

C.2. Yêu cầu đặt lịch (*HTTPS Request*) được gửi qua Internet tới **Web Server (Nginx)**.

C.3. **Nginx** chuyển tiếp yêu cầu đến **Application Server**.

C.4. **Application Server** nhận yêu cầu và thực hiện các bước xác thực nghiệp vụ:

- **Bước 4a**: Truy vấn **Database Server** để kiểm tra lại xem khung giờ mà sinh viên chọn có còn trống không và có nằm trong lịch rảnh hợp lệ của Tutor không.
- **Bước 4b**: Truy vấn **Database Server** để kiểm tra lịch trình của chính sinh viên đó, đảm bảo không bị trùng với các lịch học/lịch thi khác đã đăng ký.

C.5. Nếu tất cả các điều kiện đều hợp lệ, **Application Server** sẽ tạo một bản ghi lịch học cố định mới trong **Database Server**, liên kết sinh viên, Tutor, và môn học với khung giờ đã chọn. Trạng thái của lịch được đặt là "Đã xác nhận" (*Confirmed*).

C.6. Sau khi **Database Server** xác nhận lưu thành công, **Application Server** thực hiện hai hành động song song:

- **Bước 6a**: Gửi một phản hồi thành công về cho trình duyệt của sinh viên, hiển thị thông báo "Bạn đã đặt lịch thành công".
- **Bước 6b**: Kết nối đến **Email Service (SMTP Server)** để gửi email thông báo xác nhận lịch học cho cả sinh viên và Tutor.

4.2. Sơ đồ phát triển (Development View)

4.2.1 Giới thiệu

Sơ đồ phát triển, hay **Sơ đồ gói (Package Diagram)**, mô tả cấu trúc tĩnh và cách tổ chức mã nguồn của hệ thống từ góc nhìn của đội ngũ phát triển. Mục tiêu của sơ đồ này là trình bày một kiến trúc phần mềm có tính module hóa cao, rõ ràng và dễ bảo trì, tuân thủ nguyên tắc *tách biệt các mối quan tâm (Separation of Concerns)*.
Hệ thống được thiết kế theo kiến trúc **phân lớp (Layered Architecture)**, một mô hình phổ biến trong các ứng dụng web hiện đại. Các lớp chính bao gồm:

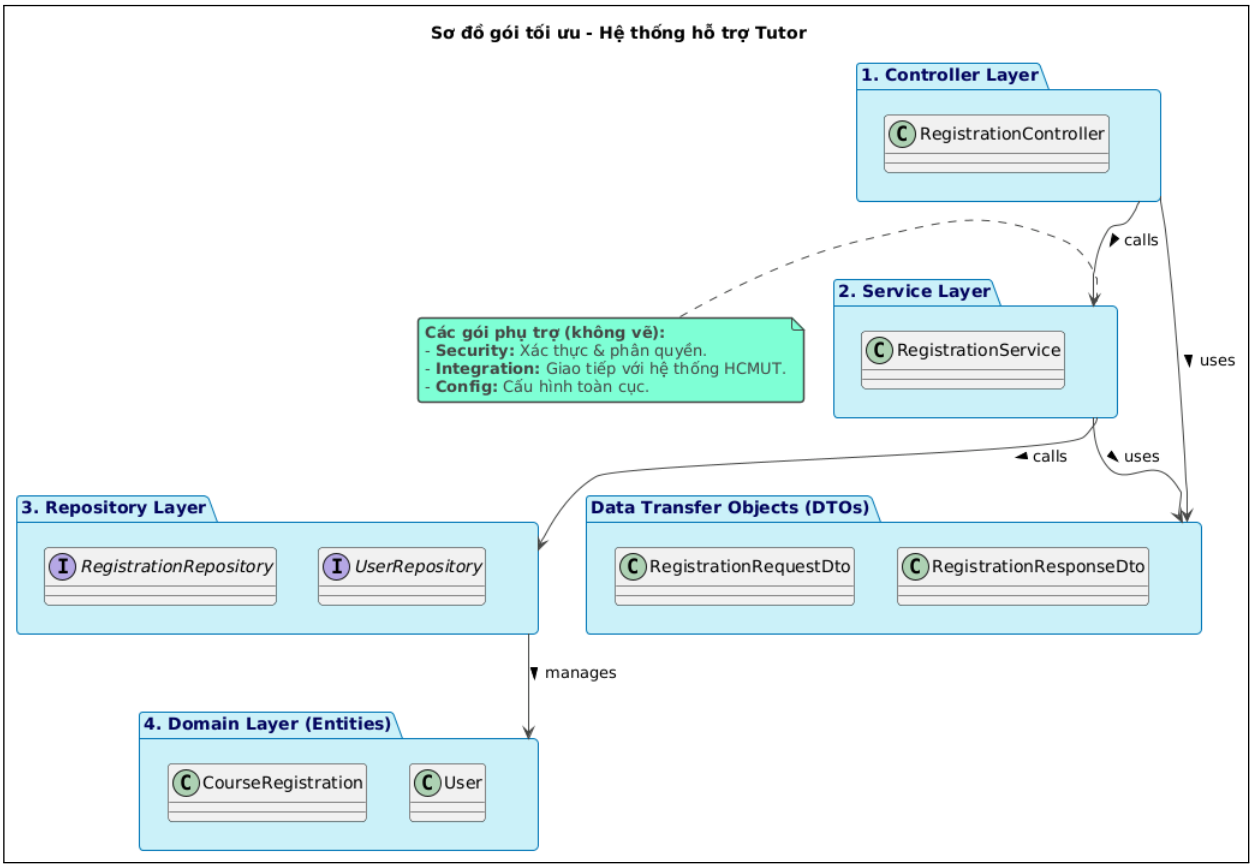
- **Controller Layer**: Tiếp nhận và xử lý các yêu cầu HTTP từ người dùng.
- **Service Layer**: Chứa đựng toàn bộ logic nghiệp vụ cốt lõi của hệ thống.
- **Repository Layer**: Chịu trách nhiệm truy cập và tương tác với cơ sở dữ liệu.
- **Domain/Model Layer**: Định nghĩa các đối tượng và thực thể dữ liệu.

Ngoài ra, hệ thống còn có các gói hỗ trợ cho các chức năng xuyên suốt như:

- **Security**: Quản lý xác thực, phân quyền và bảo mật truy cập.
- **Configuration**: Cấu hình hệ thống, môi trường và thông số hoạt động.
- **Integration**: Tích hợp với các hệ thống bên ngoài như HCMUT_SSO, HCMUT_DATACORE, Email Service,...

4.2.2. Sơ đồ gói (Package Diagram)

Sơ đồ dưới đây minh họa cách mã nguồn được tổ chức thành các gói logic và mối quan hệ phụ thuộc giữa chúng.



Hình 2: Sơ đồ tổ chức các gói của hệ thống

4.2.3. Mô tả các gói

Bảng 4: Chức năng của từng gói (package) trong kiến trúc phần mềm

STT	Tên gói	Chức năng chính
1	Controller Layer	Chứa các lớp chịu trách nhiệm tiếp nhận yêu cầu HTTP từ client, gọi đến các Service tương ứng để xử lý và trả về phản hồi (response).
2	Service Layer	Chứa toàn bộ logic nghiệp vụ của ứng dụng. Đây là nơi các quy trình như ghép cặp Tutor, xử lý đăng ký, tạo báo cáo được thực thi. Gói này điều phối hoạt động giữa Repositories và các thành phần khác.
3	Repository Layer	Chứa các interface/lớp định nghĩa các phương thức để truy cập dữ liệu. Gói này trừu tượng hóa lớp truy cập dữ liệu, giúp Service không cần biết chi tiết về cách dữ liệu được lưu trữ.
4	Domain (Entities)	Chứa các lớp thực thể (Entity) ánh xạ trực tiếp tới các bảng trong cơ sở dữ liệu, định nghĩa cấu trúc dữ liệu cốt lõi của hệ thống.
5	DTOs (Data Transfer Objects)	Chứa các lớp dùng để truyền dữ liệu giữa các lớp, đặc biệt là giữa Controller và Service. Việc sử dụng DTO giúp che giấu cấu trúc của Domain và chỉ truyền đi những dữ liệu cần thiết.
–	Các gói phụ trợ	Bao gồm Security, Integration, Config. Các gói này cung cấp các chức năng xuyên suốt và được sử dụng chủ yếu bởi Service Layer nhưng không được vẽ ra để giữ cho sơ đồ đơn giản.

4.2.4. Luồng xử lý dữ liệu qua các lớp

Để minh họa cách các gói tương tác với nhau, luồng xử lý cho chức năng "Sinh viên đăng ký môn học" (UC-04):

- **Request**: Sinh viên gửi yêu cầu đăng ký môn học từ giao diện người dùng. Yêu cầu này được gửi đến một endpoint trong Controller Layer.
- **Controller Layer**: Lớp `RegistrationController` nhận yêu cầu. Nó xác thực dữ liệu đầu vào (đóng gói trong một đối tượng `RegistrationRequestDto` từ gói DTOs) và gọi phương thức trong Service Layer.
- **Service Layer**: Lớp `RegistrationService` thực thi logic nghiệp vụ. Nó sử dụng các lớp trong Repository Layer để kiểm tra các quy tắc nghiệp vụ (ví dụ: giới hạn số môn đăng ký).
- **Repository Layer**: Các lớp như `UserRepository` và `RegistrationRepository` tương tác với cơ sở dữ liệu. Chúng làm việc với các đối tượng từ Domain Layer (ví dụ: `User`, `CourseRegistration`).
- **Response**: Sau khi xử lý xong, Service Layer trả về kết quả (thường dưới dạng một DTO khác) cho Controller Layer, và Controller sẽ tạo phản hồi HTTP gửi về cho client.