***Software Design***

***Document***

**for**

**DEE TechSpace Project**

**Version 1.0 approved**

**Prepared by NamNV\_1813168**

**Personal Project**

**2025-02-23**

**Table of Contents**

[**1. Introduction 1**](#_heading=h.1fob9te)

[1.1 Purpose 1](#_heading=h.3znysh7)

[1.2 Document Conventions 1](#_heading=h.2et92p0)

[1.3 Intended Audience and Reading Suggestions 2](#_heading=h.tyjcwt)

[1.4 Project Scope 3](#_heading=)

[1.5 References 4](#_heading=h.1t3h5sf)

[**2. Overall Description 5**](#_heading=h.4d34og8)

[2.1 Description of the Operating Environment 5](#_heading=)

[2.2 Assumptions 6](#_heading=)

[2.3 Constraints 6](#_heading=)

[**3. Functional Requirements 6**](#_heading=)

[3.1 Project and Task Management 6](#_heading=)

[3.2 Resource Management 7](#_heading=)

[3.3 Collaboration and Communication 7](#_heading=)

[3.4 Notifications and Alerts 7](#_heading=)

[3.5 Progress Monitoring and Reporting 7](#_heading=)

[3.6 Integration and Scalability 8](#_heading=)

[3.7 User Roles and Access Control 8](#_heading=)

[3.8 Security 8](#_heading=)

[**4. Non-Functional Requirements 8**](#_heading=)

[4.1 Performance Requirements 8](#_heading=)

[4.2 Security Requirements 9](#_heading=)

[4.3 Availability Requirements 9](#_heading=)

[4.4 Scalability Requirements 9](#_heading=)

[4.5 Usability Requirements 10](#_heading=)

[4.6 Usability Requirements 10](#_heading=)

[**5. Appendix 10**](#_heading=h.2s8eyo1)

[5.1 Terminology 10](#_heading=)

[5.2 References 11](#_heading=)

[5.3 Additional Information 11](#_heading=h.17dp8vu)

**Revision History**

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Date** | **Reason For Changes** | **Version** |
|  |  |  |  |
|  |  |  |  |

# Giới thiệu

## Mục đích

Tài liệu SDD này nhằm mô tả chi tiết về kiến trúc, các thành phần và cách thức hoạt động của hệ thống website dành cho sinh viên khoa Viễn Thông tại trường Bách Khoa. Tài liệu sẽ làm cầu nối giữa yêu cầu nghiệp vụ (SRS) và quá trình lập trình, triển khai hệ thống.

## Phạm vi

SDD bao gồm:

* Kiến trúc hệ thống (frontend, backend, cơ sở dữ liệu, hosting)
* Thiết kế mô-đun chi tiết cho từng chức năng:
  + Đặt lịch phòng thí nghiệm.
  + Đặt trước linh kiện điện tử.
  + Chat nhóm và video call.
  + Đăng bài trong group.
  + Mua bán linh kiện chip.
  + Tải lên/tải xuống tài liệu.
  + Tìm kiếm thông tin giảng viên/sinh viên.
  + Chat thông minh với trợ lý ảo (AI Chatbot).
* Thiết kế giao diện người dùng, bảo mật, hiệu năng và kiểm thử..

## Definitions, Acronyms, and Abbreviations

**SRS**: Software Requirements Specification

**SDD**: Software Design Document

**User**: Sinh viên, giảng viên, quản trị viên

**Admin**: Người quản trị hệ thống

**AI Chatbot**: Trợ lý ảo hỗ trợ tìm kiếm thông tin.

# Kiến Trúc Hệ Thống

## Góc nhìn sản phẩm

Hệ thống được thiết kế theo mô hình Client-Server, sử dụng kiến trúc phân tầng:

* **Frontend (Client):**
  + Sử dụng Next.js cho việc xây dựng giao diện người dùng.
  + Tailwind CSS hỗ trợ thiết kế giao diện đáp ứng (responsive) cho các thiết bị mobile và desktop.
* **Backend (Server):**
  + Dựa trên .NET Core để xây dựng các dịch vụ RESTful API.
  + Các tác vụ tìm kiếm nhanh được xử lý bởi module C++ tích hợp vào backend.
* **Cơ Sở Dữ Liệu:**
  + Sử dụng SQL Server cho việc lưu trữ dữ liệu.
  + Thiết kế CSDL theo mô hình quan hệ với các bảng được liên kết theo chức năng (ví dụ: lịch phòng thí nghiệm, đơn đặt linh kiện, bài đăng, thông tin người dùng…).
* **Hosting:**
  + Hệ thống được triển khai trên nền tảng Cloud (AWS/Azure/GCP), đảm bảo khả năng mở rộng và tính sẵn sàng cao.

## Mô Hình Triển Khai

**Client Layer:** Giao diện web chạy trên trình duyệt.

**Application Layer:** Các dịch vụ API được triển khai trên .NET Core, chịu trách nhiệm xử lý nghiệp vụ, giao tiếp với CSDL và tích hợp module C++.

**Data Layer:** SQL Server lưu trữ dữ liệu, có các cơ chế backup và replication đảm bảo an toàn dữ liệu.

**External Services:**

* Dịch vụ video call, chat nhóm tích hợp qua các giao thức WebRTC.
* Hệ thống thanh toán tích hợp (COD hoặc chuyển khoản).

## Công nghệ sử dụng

**Backend**: .NET Core 7, kiến trúc Microservices

**Frontend**: Next.js 14, TypeScript, Tailwind CSS

**Database**: SQL Server

**API**: RESTful API kết hợp WebSocket

**Xác thực**: OAuth2, JWT

**Lưu trữ dữ liệu**: AWS S3 cho tài liệu và hình ảnh

# Thiết Kế Chi Tiết Hệ Thống

### 3.1 Kiến trúc ứng dụng

Hệ thống sử dụng mô hình **Client-Server**, trong đó:

* **Client**: Ứng dụng Next.js (Frontend) chạy trên trình duyệt hoặc thiết bị di động, gửi yêu cầu đến server qua API.
* **Server**: Được thiết kế theo mô hình **Microservices**, mỗi dịch vụ đảm nhiệm một chức năng cụ thể.

Dưới đây là sơ đồ tổng quan về kiến trúc ứng dụng:

+-------------------+ +---------------------------+

| Client (SPA) | <---> | API Gateway |

+-------------------+ +---------------------------+

|

+--------------------------------+--------------------------------+

| Microservices Architecture |

+--------------------------------+--------------------------------+

| Auth Service | Booking Service | Chat Service | AI Service |

|--------------- |----------------- |-------------- |------------|

| User Accounts | Lab Reservations | WebRTC, Chat | AI Chatbot |

+-----------------------------------------------------------------+

| SQL Server Database |

+-----------------------------------------------------------------+

#### 3.1.1 Kiến trúc giao diện (Frontend)

Hệ thống được xây dựng theo kiến trúc **Single Page Application (SPA)** sử dụng **Next.js**. Điều này giúp giao diện phản hồi nhanh chóng, tải dữ liệu động mà không cần làm mới trang, và phù hợp với hệ thống có nhiều tương tác như chat, đặt lịch và mua bán linh kiện.

Các đặc điểm chính của kiến trúc SPA với Next.js:

* **Tải động dữ liệu**: Hỗ trợ fetch dữ liệu từ API một cách tối ưu.
* **Cải thiện hiệu suất**: Giảm số lần tải lại trang, mang lại trải nghiệm mượt mà cho người dùng.
* **Hỗ trợ SSR (Server-Side Rendering) và ISR (Incremental Static Regeneration)**: Tăng tốc độ tải trang và tối ưu SEO.
* **Tích hợp tốt với WebSocket**: Cung cấp khả năng cập nhật thời gian thực cho tính năng chat nhóm và video call.
* **Dễ dàng mở rộng và bảo trì**: Hỗ trợ modularization giúp phát triển nhanh chóng và dễ bảo trì.

Giao diện Next.js sẽ giao tiếp với backend thông qua API Gateway, đảm bảo tính bảo mật và tối ưu hiệu suất.

#### 3.1.2 Kiến trúc Microservices của Server

Server được xây dựng với các microservices sau:

* **Dịch vụ Xác thực (Authentication Service)**: Quản lý đăng nhập, đăng ký và phân quyền người dùng.
* **Dịch vụ Đặt lịch (Booking Service)**: Xử lý yêu cầu đặt lịch phòng thí nghiệm.
* **Dịch vụ Quản lý Linh kiện (Component Service)**: Quản lý thông tin linh kiện và đặt trước linh kiện.
* **Dịch vụ Chat & Video Call (Chat Service)**: Hỗ trợ nhắn tin thời gian thực và gọi video bằng WebRTC.
* **Dịch vụ Nội dung (Content Service)**: Quản lý bài đăng, bình luận và tương tác.
* **Dịch vụ Mua bán linh kiện (Marketplace Service)**: Hỗ trợ giao dịch mua bán linh kiện.
* **Dịch vụ Tìm kiếm (Search Service)**: Hỗ trợ tìm kiếm giảng viên, sinh viên và linh kiện.
* **Dịch vụ Trợ lý AI (AI Assistant Service)**: Tích hợp chatbot AI hỗ trợ tư vấn và tự động hóa một số quy trình.

Các microservices này giao tiếp với nhau thông qua **API Gateway**, giúp kiểm soát truy cập và tối ưu hóa luồng dữ liệu.

Hệ thống sử dụng mô hình **Client-Server**, trong đó:

* **Client**: Ứng dụng Next.js (Frontend) chạy trên trình duyệt hoặc thiết bị di động, gửi yêu cầu đến server qua API.
* **Server**: Được thiết kế theo mô hình **Microservices**, mỗi dịch vụ đảm nhiệm một chức năng cụ thể.

### 3.2 Mô hình cơ sở dữ liệu

Hệ thống sử dụng **SQL Server** để quản lý dữ liệu, đảm bảo tính nhất quán, an toàn và hiệu suất cao. Cơ sở dữ liệu được thiết kế theo mô hình **quan hệ (Relational Database)** với các bảng chính liên kết với nhau qua các khóa ngoại.

Dưới đây là sơ đồ tổng quan của kiến trúc cơ sở dữ liệu:

+------------------+ +----------------+ +----------------+

| Users |----| Bookings |----| Labs |

+------------------+ +----------------+ +----------------+

| ID (PK) | | ID (PK) | | ID (PK) |

| Name | | UserID (FK) | | Name |

| Email | | LabID (FK) | | Description |

| Role | | TimeSlot | | Status |

+------------------+ +----------------+ +----------------+

+------------------+ +----------------+ +----------------+

| Components |----| Orders |----| Messages |

+------------------+ +----------------+ +----------------+

| ID (PK) | | ID (PK) | | ID (PK) |

| Name | | UserID (FK) | | SenderID (FK) |

| Quantity | | ComponentID (FK)| | ReceiverID (FK)|

| Status | | Quantity | | Content |

+------------------+ +----------------+ +----------------+

#### 3.2.1 Tối ưu hóa hiệu suất

* **Sử dụng chỉ mục (Indexing)**: Tạo chỉ mục cho các cột tìm kiếm thường xuyên như email, labID, componentID để tăng tốc truy vấn.
* **Tối ưu truy vấn SQL**: Sử dụng **Stored Procedures** cho các thao tác phức tạp nhằm giảm tải truy vấn động.
* **Cơ chế caching**: Tích hợp Redis để lưu trữ dữ liệu tạm thời giúp giảm tải truy vấn đến SQL Server.
* **Bảo mật dữ liệu**: Sử dụng **Transparent Data Encryption (TDE)** để mã hóa dữ liệu nhạy cảm như mật khẩu và thông tin giao dịch. Hệ thống sử dụng **SQL Server** để quản lý dữ liệu, đảm bảo tính nhất quán, an toàn và hiệu suất cao. Cơ sở dữ liệu được thiết kế theo mô hình **quan hệ (Relational Database)** với các bảng chính liên kết với nhau qua các khóa ngoại.

#### 3.2.1 Cấu trúc cơ sở dữ liệu

Cấu trúc chính của cơ sở dữ liệu bao gồm:

* **Users**: Quản lý thông tin người dùng (ID, tên, email, mật khẩu, vai trò).
* **Labs**: Thông tin về các phòng thí nghiệm (ID, tên phòng, mô tả, trạng thái).
* **Bookings**: Quản lý lịch đặt phòng thí nghiệm (ID, userID, labID, thời gian, trạng thái).
* **Components**: Danh sách linh kiện điện tử (ID, tên linh kiện, số lượng, trạng thái tồn kho).
* **Orders**: Quản lý giao dịch mua bán linh kiện (ID, userID, componentID, số lượng, trạng thái đơn hàng).
* **Messages**: Lưu trữ tin nhắn trong hệ thống chat (ID, senderID, receiverID, nội dung, thời gian gửi).
* **Posts**: Quản lý bài đăng và tương tác (ID, userID, nội dung, thời gian đăng, loại bài viết).

#### 3.2.2 Quan hệ giữa các bảng

* **Users** có quan hệ 1-N với **Bookings** (mỗi người dùng có thể đặt nhiều phòng).
* **Users** có quan hệ 1-N với **Orders** (mỗi người dùng có thể thực hiện nhiều đơn hàng).
* **Labs** có quan hệ 1-N với **Bookings** (mỗi phòng có thể có nhiều lịch đặt).
* **Components** có quan hệ 1-N với **Orders** (mỗi linh kiện có thể xuất hiện trong nhiều đơn hàng).
* **Users** có quan hệ 1-N với **Messages** (mỗi người có thể gửi nhiều tin nhắn).
* **Users** có quan hệ 1-N với **Posts** (mỗi người dùng có thể đăng nhiều bài viết).

#### 3.2.3 Tối ưu hóa hiệu suất

* **Sử dụng chỉ mục (Indexing)**: Tạo chỉ mục cho các cột tìm kiếm thường xuyên như email, labID, componentID để tăng tốc truy vấn.
* **Tối ưu truy vấn SQL**: Sử dụng **Stored Procedures** cho các thao tác phức tạp nhằm giảm tải truy vấn động.
* **Cơ chế caching**: Tích hợp Redis để lưu trữ dữ liệu tạm thời giúp giảm tải truy vấn đến SQL Server.
* **Bảo mật dữ liệu**: Sử dụng **Transparent Data Encryption (TDE)** để mã hóa dữ liệu nhạy cảm như mật khẩu và thông tin giao dịch.

Các bảng dữ liệu chính:

* **Users**: Quản lý thông tin người dùng
* **Labs**: Danh sách phòng thí nghiệm
* **Bookings**: Quản lý lịch đặt phòng
* **Components**: Quản lý linh kiện điện tử
* **Orders**: Xử lý đơn hàng mua bán linh kiện
* **Messages**: Lưu trữ tin nhắn và video call
* **Posts**: Quản lý bài đăng và bình luận

### 3.3 Luồng xử lý chính

#### 3.3.1 Đặt lịch phòng thí nghiệm

# 4. Thiết Kế Về Bảo Mật và Hiệu Năng

## 4.1 Bảo Mật

**Xác Thực và Phân Quyền:**

Áp dụng xác thực 2 bước (2FA) cho người dùng.

Phân quyền chặt chẽ cho từng vai trò: Sinh viên, Giảng viên, Quản trị viên.

**Mã Hóa Dữ Liệu:**

Dữ liệu người dùng và giao tiếp giữa các thành phần đều được mã hóa (HTTPS, TLS).

**Kiểm Soát Truy Cập:**

API được bảo vệ bằng token và có giới hạn quyền truy cập phù hợp với từng chức năng.

**Kiểm Duyệt Nội Dung:**

Quản trị viên có quyền kiểm duyệt bài đăng, tài liệu, và các nội dung liên quan.

## 4.2 Hiệu Năng

**Khả Năng Xử Lý:**

Hệ thống hỗ trợ hơn 5000 người dùng đồng thời.

Sử dụng caching (Redis hoặc tương đương) cho các dữ liệu truy xuất thường xuyên.

**Tối Ưu Tốc Độ:**

Module tìm kiếm được tối ưu bằng C++ để đảm bảo thời gian phản hồi dưới 2 giây.

**Cân Bằng Tải:**

Sử dụng các giải pháp cân bằng tải (Load Balancer) của nhà cung cấp Cloud để đảm bảo hiệu năng và tính sẵn sàng.

# 5. Kế Hoạch Kiểm Thử

## 5.1 Các Loại Kiểm Thử

**Unit Test:** Kiểm tra đơn vị cho từng mô-đun backend và frontend.

**Integration Test:** Kiểm thử tích hợp giữa các module (ví dụ: giao tiếp API, WebSocket).

**System Test:** Kiểm thử hệ thống tổng thể, đảm bảo tất cả các yêu cầu từ SRS được đáp ứng.

**User Acceptance Test (UAT):** Kiểm thử với sự tham gia của người dùng cuối để đánh giá tính thân thiện và hiệu quả.

## 5.2 Công Cụ và Môi Trường Kiểm Thử

Sử dụng các framework như Jest (cho frontend) và xUnit/NUnit (cho backend).

Môi trường kiểm thử độc lập (staging) trước khi triển khai lên production.

# 6. Kế Hoạch Triển Khai và Bảo Trì

## 6.1 Môi Trường Triển Khai

**Cloud Hosting:** Triển khai trên AWS/Azure/GCP, đảm bảo tính sẵn sàng và mở rộng linh hoạt.

**CI/CD Pipeline:** Sử dụng các công cụ như GitLab CI, Jenkins hoặc GitHub Actions để tự động hóa quá trình build, test và deploy.

## 6.2 Bảo Trì và Cập Nhật

Lịch trình cập nhật định kỳ cho hệ thống, backup dữ liệu và kiểm tra bảo mật.

Hệ thống log và giám sát (monitoring) được triển khai để nhanh chóng phát hiện và xử lý sự cố.

# 7. Phụ Lục

## 7.1 Tài Liệu API và Cấu Trúc CSDL

**API Endpoints:** Tài liệu chi tiết về các endpoint, tham số đầu vào/ra và mã lỗi được cung cấp.

**Sơ Đồ CSDL:** Đính kèm sơ đồ ER và các mô tả bảng dữ liệu.

## 7.2 Danh Sách Quyền Hạn Người Dùng

Chi tiết phân quyền của từng loại người dùng: Sinh viên, Giảng viên, Quản trị viên (Admin) với các chức năng tương ứng.