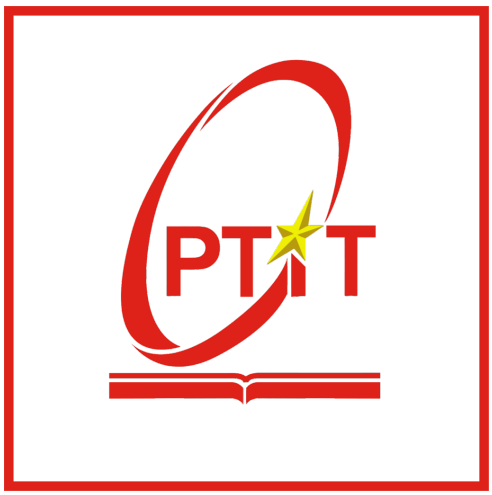
**BỘ THÔNG TIN VÀ TRUYỀN THÔNG**
**HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG**




# BÁO CÁO MÔN HỌC
# LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG
# ĐỀ TÀI: HỆ THỐNG QUẢN LÝ ĐIỂM SINH VIÊN

**GIẢNG VIÊN HƯỚNG DẪN: NGÔ TIẾN ĐỨC**

**NHÓM 07**

**B23DCCN345:** Trịnh Đặng Huy Hoàng
**B23DCCN275:** Nguyễn Đức Hải
**B23DCCN960:** Nguyễn Chí Khanh

**HÀ NỘI, 2025**

# Mục lục

# Phân công công việc

| STT | Họ tên | Công việc |
|---|---|---|
| 1 | Trịnh Đặng Huy Hoàng | - Thiết kế và triển khai module User (đăng ký, đăng nhập)<br>- Xây dựng chức năng Email và Reset Password<br>- Viết báo cáo |
| 2 | Nguyễn Đức Hải | - Thiết kế và triển khai module Document (quản lý tài liệu)<br>- Xây dựng module Semester (quản lý học kỳ)<br>- Thiết kế module Subject (quản lý môn học) |
| 3 | Nguyễn Chí Khanh | - Thiết kế và triển khai module Grade (quản lý điểm)<br>- Xây dựng chức năng Thống kê và Analytics<br>- Tích hợp AI Chatbot với OpenRouter API |

# Lời mở đầu

## Bối cảnh và động lực của dự án

Trong thời đại công nghệ số, các trường đại học ngày càng chú trọng việc ứng dụng công nghệ thông tin trong quản lý đào tạo. Một trong những công tác quan trọng là quản lý điểm sinh viên, nơi khối lượng dữ liệu lớn (sinh viên, lớp học phần, môn học, kết quả học tập...) cần được lưu trữ và truy xuất thường xuyên.

Tại Học viện Công nghệ Bưu chính Viễn thông (PTIT), việc tổng hợp và tra cứu điểm vẫn chủ yếu thực hiện thủ công hoặc thông qua các file bảng tính, gây tốn thời gian và dễ sai sót. Chính vì vậy, việc xây dựng một hệ thống quản lý điểm sinh viên toàn diện là cần thiết, nhằm hỗ trợ việc lưu trữ, xử lý và truy xuất dữ liệu học tập một cách khoa học, nhanh chóng và chính xác.

Từ nhu cầu đó, nhóm thực hiện đề tài ”Hệ thống quản lý điểm sinh viên”, tập trung vào việc áp dụng các nguyên lý Lập trình Hướng đối tượng (OOP) để xây dựng một ứng dụng web hoàn chỉnh với backend Spring Boot và cơ sở dữ liệu MySQL, phục vụ cho việc quản lý điểm sinh viên hiệu quả.

## Mục tiêu của dự án

Mục tiêu của đề tài là xây dựng hệ thống quản lý điểm sinh viên áp dụng các nguyên lý OOP. Cụ thể:

- Thiết kế kiến trúc hệ thống theo mô hình MVC (Model-View-Controller) với Spring Boot.
- Áp dụng các nguyên lý OOP: Đóng gói (Encapsulation), Kế thừa (Inheritance), Đa hình (Polymorphism), Trừu tượng (Abstraction).
- Xây dựng cơ sở dữ liệu MySQL với các bảng, ràng buộc khóa chính – khóa ngoại, đảm bảo tính toàn vẹn dữ liệu.
- Phát triển các chức năng nghiệp vụ:
  - Đăng ký, đăng nhập, quên mật khẩu với email xác thực.
  - Quản lý học kỳ, môn học và điểm số.
  - Tính toán điểm trung bình, GPA tự động.
  - Quản lý tài liệu học tập (upload, download, bookmark).
  - Thống kê và phân tích kết quả học tập.
  - Tích hợp AI Chatbot hỗ trợ sinh viên tra cứu thông tin.
- Xây dựng RESTful API để tích hợp với frontend.
- Đảm bảo tính bảo mật với mã hóa mật khẩu (BCrypt) và token-based authentication.

# Phạm vi và công nghệ sử dụng

Đề tài tập trung vào phát triển backend với kiến trúc RESTful API.

**Công nghệ sử dụng:**

- Ngôn ngữ lập trình: Java 17+
- Framework: Spring Boot 3.x (Spring MVC, Spring JDBC)
- Hệ quản trị cơ sở dữ liệu: MySQL 8.0
- Công cụ thiết kế: MySQL Workbench
- Thư viện hỗ trợ: BCrypt (mã hóa mật khẩu), JavaMailSender, Apache HttpClient
- API tích hợp: OpenRouter AI API (GPT-4o-mini)
- Build tool: Maven

**Phạm vi dữ liệu:**

- Sinh viên (MSSV, họ tên, email, mật khẩu,...)
- Học kỳ (tên, thời gian, GPA)
- Môn học (tên, mã môn, số tín chỉ)
- Bảng điểm (điểm thành phần, điểm TB, điểm chữ, GPA)
- Tài liệu (file PDF, DOC, DOCX,..)
- Lịch sử chat AI
- Token đặt lại mật khẩu

# Tầm quan trọng của dự án

Hệ thống quản lý điểm sinh viên đóng vai trò nền tảng cho các ứng dụng học tập sau này. Việc áp dụng OOP và Spring Boot giúp:

- Dễ dàng bảo trì và mở rộng hệ thống nhờ kiến trúc phân tầng rõ ràng.
- Tăng tính tái sử dụng code thông qua các Service, Repository.
- Tích hợp dễ dàng với frontend (React, Vue.js) thông qua RESTful API.
- Góp phần số hóa công tác quản lý đào tạo, nâng cao hiệu quả làm việc.

# Chương 1

# Tổng quan và phân tích hệ thống

## 1.1 Giới thiệu chung

Hệ thống quản lý điểm sinh viên được thiết kế theo kiến trúc MVC (Model-View-Controller) sử dụng Spring Boot framework. Hệ thống áp dụng các nguyên lý OOP để đảm bảo tính module hóa, dễ bảo trì và mở rộng. Mục tiêu chính của hệ thống là:

- Quản lý thông tin người dùng (sinh viên) với authentication và authorization.
- Quản lý học kỳ, môn học và điểm số với tính toán tự động.
- Lưu trữ và quản lý tài liệu học tập.
- Thống kê và phân tích kết quả học tập.
- Hỗ trợ sinh viên thông qua AI Chatbot.
- Đảm bảo bảo mật và khôi phục mật khẩu.

## 1.2 Kiến trúc hệ thống

### 1.2.1 Mô hình MVC

Hệ thống được xây dựng theo mô hình MVC:

- **Model:** Các entity class đại diện cho dữ liệu nghiệp vụ.
- **Controller:** Các REST Controller xử lý HTTP request và trả về response
- **Service:** Tầng nghiệp vụ chứa logic xử lý
- **Repository:** Tầng truy xuất dữ liệu sử dụng Spring JDBC Template

### 1.2.2 Các nguyên lý OOP áp dụng

**Đóng gói (Encapsulation)**

Tất cả các entity class đều áp dụng đóng gói với private fields và public getter/setter:

```
1 package com.studentmgmt.backend.model;
2
3 import java.time.LocalDateTime;
4
5 public class User {
6     private Long id;
7     private String studentId;
8     private String email;
9     private String password;
```

```
10      private String fullName;
11      private LocalDateTime createdAt;
12
13      // Getters and Setters
14      public Long getId() { return id; }
15      public void setId(Long id) { this.id = id; }
16
17      public String getStudentId() { return studentId; }
18      public void setStudentId(String studentId) { this.studentId
            = studentId; }
19
20      public String getEmail() { return email; }
21      public void setEmail(String email) { this.email = email; }
22
23      // ... other getters/setters
24  }
```

Listing 1.1: Ví dụ Encapsulation trong User.java

### Trừu tượng hóa (Abstraction)

Hệ thống sử dụng các interface và abstract class để định nghĩa hành vi chung:

- Service layer định nghĩa các business logic trừu tượng (ví dụ: ‘GradeCalculationService‘, ‘AnalyticsService‘).
- Repository pattern che giấu chi tiết triển khai truy xuất dữ liệu (sử dụng ‘JdbcTemplate‘).
- RowMapper pattern trừu tượng hóa việc mapping ResultSet sang Object.

### Kế thừa (Inheritance)

Các controller đều kế thừa từ các base class của Spring Framework (thông qua annotations). Ví dụ, ‘CustomUserDetailsService‘ implement interface ‘UserDetailsService‘ của Spring Security.

```
1   package com.studentmgmt.backend.security;
2
3   import java.util.Collections;
4   import org.springframework.beans.factory.annotation.Autowired;
5   import org.springframework.security.core.userdetails.UserDetails
        ;
6   import org.springframework.security.core.userdetails.
        UserDetailsService;
7   import org.springframework.security.core.userdetails.
        UsernameNotFoundException;
8   import org.springframework.stereotype.Service;
9   import com.studentmgmt.backend.model.User;
10  import com.studentmgmt.backend.repository.UserRepository;
11
12  @Service
13  public class CustomUserDetailsService
14      implements UserDetailsService {
15
16      @Autowired
```

```
17      private UserRepository userRepository;
18
19      @Override
20      public UserDetails loadUserByUsername(String studentId)
21          throws UsernameNotFoundException {
22          User user = userRepository.findByStudentId(studentId);
23          if (user == null) {
24              throw new UsernameNotFoundException("User not found"
                    );
25          }
26          return org.springframework.security.core.userdetails.
                User
27                  .withUsername(user.getStudentId())
28                  .password(user.getPassword())
29                  .authorities(Collections.emptyList())
30                  .build();
31      }
32  }
```

Listing 1.2: Kế thừa trong CustomUserDetailsService.java

**Đa hình (Polymorphism)**

Đa hình được thể hiện qua việc override các method và sử dụng dependency injection. Các ‘RowMapper‘ là ví dụ điển hình của việc override phương thức ‘mapRow‘.

```
1   private static class UserRowMapper implements RowMapper<User> {
2       @Override
3       public User mapRow(ResultSet rs, int rowNum)
4           throws SQLException {
5           User user = new User();
6           user.setId(rs.getLong("id"));
7           user.setStudentId(rs.getString("student_id"));
8           user.setEmail(rs.getString("email"));
9           user.setPassword(rs.getString("password"));
10          user.setFullName(rs.getString("full_name"));
11          var ts = rs.getTimestamp("created_at");
12          user.setCreatedAt(ts != null ? ts.toLocalDateTime() :
                null);
13          return user;
14      }
15  }
```

Listing 1.3: Ví dụ Polymorphism trong UserRepository.java

## 1.3 Công nghệ và công cụ

### 1.3.1 Spring Boot Framework

- **Spring MVC:** Xây dựng RESTful API
- **Spring JDBC:** Truy xuất cơ sở dữ liệu

- **Spring Security:** Bảo mật và authentication
- **Spring Mail:** Gửi email reset password
- **Dependency Injection:** Quản lý dependencies

### 1.3.2 MySQL Database

- **MySQL 8.0:** Hệ quản trị cơ sở dữ liệu quan hệ
- **InnoDB Engine:** Hỗ trợ transaction và foreign key
- **UTF8MB4:** Encoding hỗ trợ tiếng Việt

### 1.3.3 Thư viện và API

- **BCrypt:** Mã hóa mật khẩu
- **JavaMail:** Gửi email
- **Apache HttpClient:** HTTP client cho API calls
- **Jackson:** JSON parsing
- **OpenRouter API:** Tích hợp AI chatbot (GPT-4o-mini)

# Chương 2

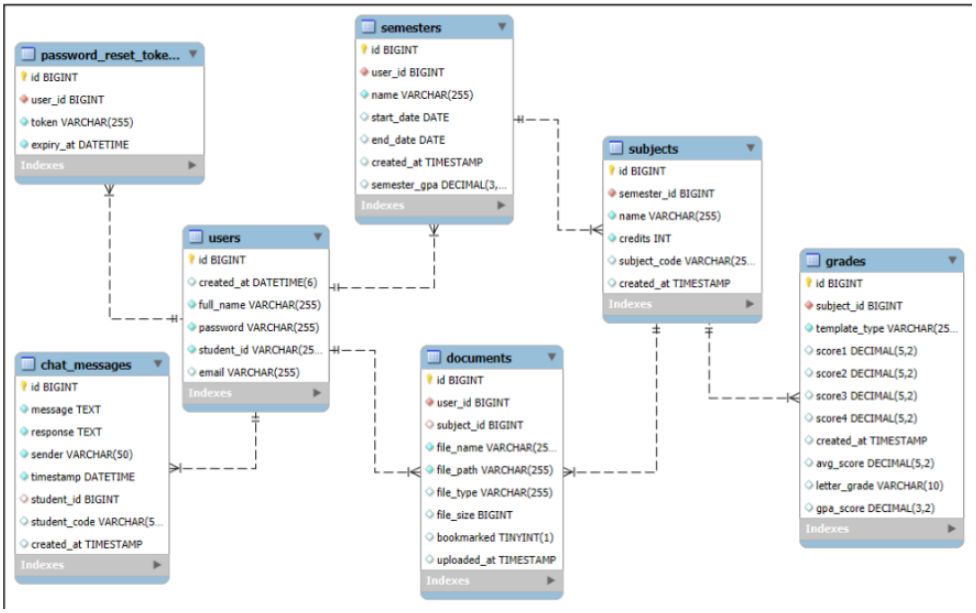
# Thiết kế cơ sở dữ liệu

## 2.1 Mô hình cơ sở dữ liệu

Cơ sở dữ liệu bao gồm **7 bảng chính**:

| STT | Tên bảng | Chức năng chính |
|---|---|---|
| 1 | users | Lưu thông tin tài khoản người dùng (sinh viên). |
| 2 | semesters | Quản lý các học kỳ của sinh viên. |
| 3 | subjects | Lưu thông tin môn học thuộc từng học kỳ. |
| 4 | grades | Quản lý điểm số và xếp loại học phần. |
| 5 | documents | Lưu trữ tài liệu học tập và tệp tin đính kèm. |
| 6 | chat_messages | Ghi lại lịch sử hội thoại giữa sinh viên và AI. |
| 7 | password_reset_tokens | Lưu thông tin token phục vụ đặt lại mật khẩu. |

Bảng 2.1. Danh sách các bảng trong cơ sở dữ liệu




Hình 2.1. Mô hình quan hệ hệ thống quản lý điểm sinh viên

## 2.2 Mô tả chi tiết các thực thể

### 2.2.1 USERS (Người dùng)

**Cấu trúc:** Users (id, student_id, email, password, full_name, created_at)

**Mô tả các thuộc tính:**

- **id:** Khóa chính (PRIMARY KEY), định danh duy nhất.

- **student_id:** Khóa ngoại tham chiếu đến users.
- **student_code:** Mã sinh viên.
- **created_at:** Thời điểm lưu bản ghi.

### 2.2.2 PASSWORD_RESET_TOKENS (Token đặt lại mật khẩu)

**Cấu trúc:** PasswordResetTokens (id, user_id, token, expiry_at)
**Mô tả các thuộc tính:**

- **id:** Khóa chính.
- **user_id:** Khóa ngoại tham chiếu đến users.
- **token:** Chuỗi token UUID duy nhất.
- **expiry_at:** Thời gian hết hạn (15 phút).

### 2.2.3 SEMESTERS (Học kỳ)

**Cấu trúc:** Semesters (id, uuse_id, name, start_date, end_date, created_at, semester_gpa)
**Mô tả các thuộc tính:**

- **id:** Khóa chính.
- **user_id:** Khóa ngoại liên kết đến người dùng tương ứng.
- **name:** Tên học kỳ (ví dụ: ”Học kỳ 1 năm 2025”).
- **start_date, end_date:** Ngày bắt đầu và kết thúc học kỳ.
- **semester_gpa:** Điểm trung bình học kỳ.
- **created_at:** Thời điểm lưu bản ghi.

### 2.2.4 SUBJECTS (Môn học)

**Cấu trúc:** Subjects (id, semester_id, name, credits, subject_code, created_at)
**Mô tả các thuộc tính:**

- **id:** Khóa chính.
- **semester_id**: Khóa ngoại chỉ định học kỳ mà môn học thuộc về.
- **name:** Tên môn học.
- **credits:** Số tín chỉ của môn.
- **subject_code:** Mã môn học.
- **created_at:** Thời điểm lưu bản ghi.

### 2.2.5 GRADES (Điểm số)

**Cấu trúc:** Grades (id, subject_id, template_type, score1, score2, score3, score4, avg_score, letter_grade, gpa_score, created_at)

**Mô tả các thuộc tính:**

- **id:** Khóa chính.
- **subject_id**: Khóa ngoại liên kết đến môn học tương ứng.
- **template_type:** Kiểu mẫu chấm điểm.
- **score1 – score4:** Điểm thành phần.
- **avg_score:** Điểm trung bình của môn học.
- **letter_grade:** Quy đổi điểm chữ.
- **gpa_score:** Điểm hệ số 4 tương ứng.
- **created_at:** Thời điểm lưu bản ghi.

### 2.2.6 DOCUMENTS (Tài liệu)

**Cấu trúc:** Documents (id, user_id, subject_id, file_name, file_path, file_type, file_size, bookmarked, uploaded_at)

**Mô tả các thuộc tính:**

- **id:** Khóa chính.
- **user_id:** Khóa ngoại trỏ đến sinh viên sở hữu tài liệu.
- **subject_id**: Khóa ngoại trỏ đến môn học có tài liệu đó (có thể NULL).
- **file_name:** Tên tệp tài liệu.
- **file_path:** Đường dẫn lưu tệp
- **file_type:** Loại tệp.
- **file_size:** Dung lượng của file.
- **bookmarked:** Đánh dấu tài liệu yêu thích.
- **uploaded_at:** Thời điểm tải tài liệu lên hệ thống.

### 2.2.7 CHAT MESSAGES (Tin nhắn Chat)

**Cấu trúc:** ChatMessages (id, message, response, sender, timestamp, student_id, student_code, created_at)

**Mô tả các thuộc tính:**

- **id:** Khóa chính.
- **subject_id**: Khóa ngoại liên kết với người dùng.
- **subject_code**: Mã định danh sinh viên.

- **message:** Nội dung tin nhắn người dùng gửi cho AI.
- **response:** Phản hồi từ AI Assistant.
- **sender:** Xác định ai là người gửi (user hoặc AI).
- **timestamp:** Thời gian gửi tin nhắn.
- **created_at:** Thời điểm lưu bản ghi.

## 2.3 Script SQL tạo bảng

### 2.3.1 Bảng users

```sql
CREATE TABLE `users` (
  `id` bigint NOT NULL AUTO_INCREMENT,
  `created_at` datetime(6) DEFAULT NULL,
  `full_name` varchar(255) NOT NULL,
  `password` varchar(255) NOT NULL,
  `student_id` varchar(255) NOT NULL,
  `email` varchar(255) DEFAULT NULL,
  PRIMARY KEY (`id`),
  UNIQUE KEY `student_id` (`student_id`),
  UNIQUE KEY `email` (`email`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4
  COLLATE=utf8mb4_0900_ai_ci;
```

Listing 2.1: Tạo bảng users

### 2.3.2 Bảng semesters

```sql
CREATE TABLE `semesters` (
  `id` bigint NOT NULL AUTO_INCREMENT,
  `user_id` bigint NOT NULL,
  `name` varchar(255) NOT NULL,
  `start_date` date DEFAULT NULL,
  `end_date` date DEFAULT NULL,
  `created_at` timestamp NULL DEFAULT CURRENT_TIMESTAMP,
  `semester_gpa` decimal(3,2) DEFAULT NULL,
  PRIMARY KEY (`id`),
  KEY `user_id` (`user_id`),
  CONSTRAINT `semesters_ibfk_1` FOREIGN KEY (`user_id`)
    REFERENCES `users` (`id`) ON DELETE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4
  COLLATE=utf8mb4_0900_ai_ci;
```

Listing 2.2: Tạo bảng semesters

### 2.3.3 Bảng subjects

```sql
CREATE TABLE `subjects` (
  `id` bigint NOT NULL AUTO_INCREMENT,
  `semester_id` bigint NOT NULL,
  `name` varchar(255) NOT NULL,
  `credits` int NOT NULL,
  `subject_code` varchar(255) DEFAULT NULL,
  `created_at` timestamp NULL DEFAULT CURRENT_TIMESTAMP,
  PRIMARY KEY (`id`),
  KEY `semester_id` (`semester_id`),
  CONSTRAINT `subjects_ibfk_1` FOREIGN KEY (`semester_id`)
    REFERENCES `semesters` (`id`) ON DELETE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4
  COLLATE=utf8mb4_0900_ai_ci;
```

Listing 2.3: Tạo bảng subjects

### 2.3.4 Bảng grades

```sql
CREATE TABLE `grades` (
  `id` bigint NOT NULL AUTO_INCREMENT,
  `subject_id` bigint NOT NULL,
  `template_type` varchar(255) NOT NULL,
  `score1` decimal(5,2) DEFAULT NULL,
  `score2` decimal(5,2) DEFAULT NULL,
  `score3` decimal(5,2) DEFAULT NULL,
  `score4` decimal(5,2) DEFAULT NULL,
  `created_at` timestamp NULL DEFAULT CURRENT_TIMESTAMP,
  `avg_score` decimal(5,2) DEFAULT NULL,
  `letter_grade` varchar(10) DEFAULT NULL,
  `gpa_score` decimal(3,2) DEFAULT NULL,
  PRIMARY KEY (`id`),
  KEY `subject_id` (`subject_id`),
  CONSTRAINT `grades_ibfk_1` FOREIGN KEY (`subject_id`)
    REFERENCES `subjects` (`id`) ON DELETE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4
  COLLATE=utf8mb4_0900_ai_ci;
```

Listing 2.4: Tạo bảng grades

### 2.3.5 Bảng documents

```sql
CREATE TABLE `documents` (
  `id` bigint NOT NULL AUTO_INCREMENT,
  `user_id` bigint NOT NULL,
  `subject_id` bigint DEFAULT NULL,
  `file_name` varchar(255) NOT NULL,
  `file_path` varchar(255) NOT NULL,
  `file_type` varchar(255) DEFAULT NULL,
  `file_size` bigint DEFAULT NULL,
  `bookmarked` tinyint(1) DEFAULT '0',
```

```
  `uploaded_at` timestamp NULL DEFAULT CURRENT_TIMESTAMP,
  PRIMARY KEY (`id`),
  KEY `user_id` (`user_id`),
  KEY `subject_id` (`subject_id`),
  CONSTRAINT `documents_ibfk_1` FOREIGN KEY (`user_id`)
    REFERENCES `users` (`id`) ON DELETE CASCADE,
  CONSTRAINT `documents_ibfk_2` FOREIGN KEY (`subject_id`)
    REFERENCES `subjects` (`id`) ON DELETE SET NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4
  COLLATE=utf8mb4_0900_ai_ci;
```

Listing 2.5: Tạo bảng documents

### 2.3.6 Bảng chat_messages

```
CREATE TABLE `chat_messages` (
  `id` bigint NOT NULL AUTO_INCREMENT,
  `message` text NOT NULL,
  `response` text NOT NULL,
  `sender` varchar(50) NOT NULL,
  `timestamp` datetime NOT NULL,
  `student_id` bigint DEFAULT NULL,
  `student_code` varchar(50) DEFAULT NULL,
  `created_at` timestamp NULL DEFAULT CURRENT_TIMESTAMP,
  PRIMARY KEY (`id`),
  KEY `idx_chat_messages_student_id` (`student_id`),
  KEY `idx_chat_messages_student_code` (`student_code`),
  KEY `idx_chat_messages_timestamp` (`timestamp`),
  CONSTRAINT `fk_chat_messages_student`
    FOREIGN KEY (`student_id`)
    REFERENCES `users` (`id`) ON DELETE SET NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4
  COLLATE=utf8mb4_0900_ai_ci;
```

Listing 2.6: Tạo bảng chat_messages

### 2.3.7 Bảng password_reset_tokens

```
CREATE TABLE `password_reset_tokens` (
  `id` bigint NOT NULL AUTO_INCREMENT,
  `user_id` bigint NOT NULL,
  `token` varchar(255) NOT NULL,
  `expiry_at` datetime NOT NULL,
  PRIMARY KEY (`id`),
  KEY `user_id` (`user_id`),
  KEY `token` (`token`),
  CONSTRAINT `password_reset_tokens_ibfk_1`
    FOREIGN KEY (`user_id`)
    REFERENCES `users` (`id`) ON DELETE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4
  COLLATE=utf8mb4_0900_ai_ci;
```

Listing 2.7: Tạo bảng password_reset_tokens

# Chương 3

# Triển khai hệ thống

## 3.1 Kiến trúc phân tầng

Hệ thống được tổ chức theo kiến trúc phân tầng:

1. **Controller Layer:** Xử lý HTTP request/response
2. **Service Layer:** Chứa business logic
3. **Repository Layer:** Truy xuất dữ liệu
4. **Model Layer:** Định nghĩa data model

## 3.2 Module Authentication (User Management)

Module này quản lý toàn bộ quá trình xác thực và thông tin người dùng.

### 3.2.1 Đăng ký và Đăng nhập

Luồng đăng ký và đăng nhập sử dụng ‘BCryptPasswordEncoder‘ để đảm bảo an toàn mật khẩu.

- **Đăng ký:** ‘AuthController‘ nhận thông tin user. ‘UserRepository‘ kiểm tra ‘studentId‘ đã tồn tại chưa. Nếu chưa, ‘passwordEncoder‘ mã hóa mật khẩu và ‘UserRepository‘ lưu user mới vào DB.
- **Đăng nhập:** ‘AuthController‘ nhận ‘studentId‘ và ‘password‘. ‘UserRepository‘ tìm user theo ‘studentId‘. Nếu tìm thấy, ‘passwordEncoder.matches‘ được dùng để so sánh mật khẩu người dùng nhập với mật khẩu đã mã hóa trong DB.

```
@RestController
@RequestMapping("/api/auth")
@CrossOrigin(origins = "http://localhost:3000")
public class AuthController {

    @Autowired
    private UserRepository userRepository;

    private final BCryptPasswordEncoder passwordEncoder
        = new BCryptPasswordEncoder();

    @PostMapping("/register")
    public ResponseEntity<?> register(@RequestBody User user) {
        if (userRepository.existsByStudentId(user.getStudentId()
           )) {
            Map<String, String> response = new HashMap<>();
```

```
            response.put("message", "Student ID already exists");
            return ResponseEntity.badRequest().body(response);
        }
        user.setPassword(passwordEncoder.encode(user.getPassword()));
        Long newUserId = userRepository.save(user);

        return ResponseEntity.ok(...);
    }

    @PostMapping("/login")
    public ResponseEntity<?> login(
        @RequestBody LoginRequest loginRequest) {
        User user = userRepository.findByStudentId(
            loginRequest.getStudentId());

        if (user == null || !passwordEncoder.matches(
            loginRequest.getPassword(), user.getPassword())) {
            Map<String, String> response = new HashMap<>();
            response.put("message", "Invalid student ID or password");
            return ResponseEntity.badRequest().body(response);
        }

        return ResponseEntity.ok(...);
    }

    public static class LoginRequest { ... }
}
```

Listing 3.1: AuthController.java - Đăng ký và Đăng nhập

### 3.2.2 Chức năng Quên mật khẩu

Đây là một luồng nghiệp vụ phức tạp, phối hợp giữa nhiều service:

1. **Yêu cầu:** Người dùng nhập email tại ‘POST /api/auth/forgot-password‘.
2. **Tìm User:** ‘AuthController‘ gọi ‘userRepository.findByEmail‘ để tìm tài khoản.
3. **Tạo Token:** Nếu email tồn tại, ‘PasswordResetTokenService‘ được gọi. Service này tạo một token UUID ngẫu nhiên và thời gian hết hạn (15 phút).
4. **Lưu Token:** ‘PasswordResetTokenRepository‘ lưu token này vào bảng ‘password_reset_tokens‘ cùng với ‘user_id‘ tương ứng.
5. **Gửi Email:** ‘EmailService‘ được kích hoạt, gửi một email đến người dùng chứa đường link đến trang frontend (‘http://localhost:3000/reset-password?token=...‘) kèm theo token vừa tạo.
6. **Đặt lại:** Người dùng nhấn vào link, nhập mật khẩu mới tại ‘POST /api/auth/reset-password‘.

7. **Xác thực Token:** ‘AuthController‘ gọi ‘tokenRepository.findByToken‘ để kiểm tra token. Nếu token không tồn tại hoặc đã hết hạn (‘expiryAt.isBefore(LocalDateTime.now())‘), hệ thống báo lỗi.

8. **Cập nhật Mật khẩu:** Nếu token hợp lệ, ‘userRepository.updatePasswordById‘ được gọi để cập nhật mật khẩu mới (đã được mã hóa bằng BCrypt) cho ‘user_id‘ tương ứng.

9. **Xóa Token:** ‘tokenRepository.deleteByToken‘ được gọi để vô hiệu hóa token ngay sau khi sử dụng thành công.

```
@Service
public class EmailService {
    @Autowired
    private JavaMailSender mailSender;
    // ...
    public void sendResetPasswordEmail(String toEmail, String
        token) {
        String subject = "Yeu cau dat lai mat khau";

        String resetLink = "http://localhost:3000/reset-password
            ?token="
            + token;

        String text = """
                Xin chao,
                Ban da yeu cau dat lai mat khau...
                Hay nhap vao lien ket ben duoi...
                %s
                Lien ket nay se het han sau 15 phut.
                """.formatted(resetLink);

        SimpleMailMessage message = new SimpleMailMessage();
        message.setFrom(...);
        message.setTo(toEmail);
        message.setSubject(subject);
        message.setText(text);

        mailSender.send(message);
    }
}
```

Listing 3.2: EmailService.java - Gửi email Reset

## 3.3 Module Quản lý Học kỳ (Semester)

### 3.3.1 CRUD Học kỳ

‘SemesterController‘ quản lý các thao tác cơ bản như tạo, xem, xóa học kỳ. Dữ liệu được lưu thông qua ‘SemesterRepository‘.

```
@RestController
@RequestMapping("/api/semesters")
public class SemesterController {
```

```
    @Autowired
    private SemesterRepository semesterRepository;
    @Autowired
    private UserRepository userRepository;
    // ...
    @PostMapping
    public ResponseEntity<?> createSemester(
        @RequestBody Map<String, Object> request) {
        try {
            Long userId = parseUserId(request.get("userId"));
            if (!userRepository.existsById(userId)) {
                return ResponseEntity.badRequest()
                    .body(Map.of("message", "User not found"));
            }
            Semester semester = new Semester();
            semester.setUserId(userId);
            semester.setName(((String) request.get("name")).trim
                ());

            Long newId = semesterRepository.save(semester);
            semester.setId(newId);

            return ResponseEntity.ok(semester);
        } catch (Exception e) {
            return ResponseEntity.badRequest().body(...);
        }
    }
}
```

Listing 3.3: SemesterController.java - Tạo học kỳ

### 3.3.2 Tự động hóa tính GPA học kỳ

Đây là một nghiệp vụ quan trọng. Thay vì tính toán GPA ở phía client hoặc mỗi khi tải trang, hệ thống thực hiện tính toán ở backend và lưu kết quả vào cột ‘semester_gpa‘ của bảng ‘semesters‘.

Luồng tính toán được kích hoạt ở nhiều nơi:

- **Khi điểm thay đổi (CRUD Grade):** ‘GradeController‘ (khi ‘createGrade‘, ‘updateGrade‘, ‘deleteGrade‘) sẽ gọi ‘semesterGpaService.recalculateSemesterGpaOnGradeChange(subjectId)‘. Service này tự động tìm học kỳ chứa ‘subjectId‘ đó và tính lại GPA.
- **Khi xóa môn học:** ‘SubjectController‘ (khi ‘deleteSubject‘) cũng gọi ‘semesterGpaService.calculateS để cập nhật lại GPA sau khi môn học bị xóa.
- **API thủ công:** ‘SemesterController‘ cung cấp endpoint ‘POST /id/calculate-gpa‘ để kích hoạt tính toán thủ công nếu cần.

## 3.4 Module Quản lý Môn học (Subject)

### 3.4.1 CRUD Môn học

‘SubjectController‘ quản lý môn học, đảm bảo mỗi môn học phải thuộc về một học kỳ (‘semesterId‘).

### 3.4.2 Xử lý xóa (Cascade Deleting)

Khi một môn học bị xóa, hệ thống cần đảm bảo tính toàn vẹn dữ liệu. ‘SubjectController‘ thực hiện logic này:

1. Tìm ‘semesterId‘ của môn học sắp bị xóa.
2. Gọi ‘gradeRepository.deleteBySubjectId(id)‘ để xóa tất cả các bản ghi điểm (‘grades‘) liên quan đến môn học này trước.
3. Gọi ‘subjectRepository.deleteById(id)‘ để xóa môn học.
4. Gọi ‘semesterGpaService.calculateSemesterGpa(semesterId)‘ để tính lại GPA cho học kỳ (vì số tín chỉ và điểm đã thay đổi).

```
@DeleteMapping("/{id}")
public ResponseEntity<?> deleteSubject(@PathVariable Long id,
    @RequestParam Long userId) {
    try {
        Subject subject = subjectRepository.findById(id);
        Long semesterId = null;
        if (subject != null && subject.getSemester() != null) {
            semesterId = subject.getSemester().getId();
        }

        gradeRepository.deleteBySubjectId(id);

        subjectRepository.deleteById(id);

        if (semesterId != null) {
        semesterGpaService.calculateSemesterGpa(semesterId);
        }
        return ResponseEntity.ok(Map.of(
            "message", "Da xoa mon hoc thanh cong"));
    } catch (Exception e) {
        return ResponseEntity.badRequest().body(...);
    }
}
```

Listing 3.4: SubjectController.java - Xử lý xóa môn học

## 3.5 Module Quản lý Điểm (Grade)

Đây là module cốt lõi, chứa logic nghiệp vụ tính toán điểm.

### 3.5.1 Entity Grade với Business Logic

Lớp ‘Grade.java‘ không chỉ là một Model đơn thuần mà còn chứa các phương thức nghiệp vụ quan trọng, thể hiện rõ tính đóng gói của OOP:

- ‘calculateLetterGrade()‘: Quy đổi ‘avgScore‘ (điểm trung bình hệ 10) sang điểm chữ (A+, A, B+,...).

- ‘calculateGpaScore()‘: Quy đổi ‘avgScore‘ sang điểm hệ 4.0.

```java
public class Grade {

    private BigDecimal avgScore;
    private String letterGrade;
    private BigDecimal gpaScore;

    public String calculateLetterGrade() {
        if (this.avgScore == null) return null;
        double score = this.avgScore.doubleValue();

        if (score >= 8.95) return "A+";
        if (score >= 8.45) return "A";
        if (score >= 7.95) return "B+";
        if (score >= 6.95) return "B";
        if (score >= 6.45) return "C+";
        if (score >= 5.45) return "C";
        if (score >= 4.95) return "D+";
        if (score >= 3.95) return "D";
        return "F";
    }

    public BigDecimal calculateGpaScore() {
        if (this.avgScore == null) return null;
        double score = this.avgScore.doubleValue();

        if (score >= 8.95) return new BigDecimal("4.0");
        if (score >= 8.45) return new BigDecimal("3.7");
        if (score >= 7.95) return new BigDecimal("3.5");
        if (score >= 6.95) return new BigDecimal("3.0");

        return new BigDecimal("0.0");
    }
}
```

Listing 3.5: Grade.java - Tính toán điểm chữ và GPA

### 3.5.2 Service Tính toán điểm (GradeCalculationService)

‘GradeCalculationService‘ chịu trách nhiệm điều phối việc tính toán.

- Hàm ‘calculateAndUpdateGradeAvg(gradeId)‘ được gọi bởi ‘GradeController‘ mỗi khi tạo hoặc cập nhật điểm.
- Nó lấy ‘templateType‘ (ví dụ: ”10-20-70”), gọi ‘analyticsService.calculateGradeAverage‘ để tính điểm trung bình dựa trên trọng số.
- Sau đó, nó gọi các hàm ‘calculateLetterGrade()‘ và ‘calculateGpaScore()‘ (từ ‘Grade.java‘) để quy đổi.
- Cuối cùng, nó cập nhật cả 3 giá trị (‘avgScore‘, ‘letterGrade‘, ‘gpaScore‘) vào CSDL.

```java
public class GradeCalculationService {
    @Autowired
    private GradeRepository gradeRepository;
    @Autowired
    private AnalyticsService analyticsService;
    @EventListener(ApplicationReadyEvent.class)
    public void autoUpdateAllGradesOnStartup() {
        try {
            List<Grade> allGrades = gradeRepository.findAll();
            if (allGrades.isEmpty()) {
                return;
            }
            for (Grade grade : allGrades) {
                try {
                    Double avg = analyticsService.
                       calculateGradeAverage(grade);
                    BigDecimal avgScore = avg != null ?
                       BigDecimal.valueOf(avg) : BigDecimal.ZERO
                       ;
                    grade.setAvgScore(avgScore);
                    String letterGrade = grade.
                       calculateLetterGrade();
                    BigDecimal gpaScore = grade.
                       calculateGpaScore();
                    grade.setLetterGrade(letterGrade);
                    grade.setGpaScore(gpaScore);
                    gradeRepository.save(grade);
                } catch (Exception e) {
                }
            }
        } catch (Exception e) {
        }
    }
    public void calculateAndUpdateGradeAvg(Long gradeId) {
        try {
            Grade grade = gradeRepository.findById(gradeId)
                .orElseThrow(() -> new RuntimeException("Khong
                   tim thay grade: " + gradeId));
            Double avg = AnalyticsService.calculateGradeAverage(
               grade);
            grade.setAvgScore(avg != null ? BigDecimal.valueOf(
               avg) : null);
            String letterGrade = grade.calculateLetterGrade();
            BigDecimal gpaScore = grade.calculateGpaScore();
            grade.setLetterGrade(letterGrade);
            grade.setGpaScore(gpaScore);
            gradeRepository.save(grade);
        } catch (Exception e) {
            throw new RuntimeException("Loi tinh diem TB", e);
        }
    }
    @Async
    public void updateAllExistingGradesAvg() {
```

```
        try {
            List<Grade> allGrades = gradeRepository.findAll();
            if (allGrades.isEmpty()) {
                return;
            }

            for (Grade grade : allGrades) {
                if (grade.getAvgScore() == null) {
                    Double avg = analyticsService.
                        calculateGradeAverage(grade);

                    grade.setAvgScore(avg != null ? BigDecimal.
                        valueOf(avg) : BigDecimal.ZERO);

                    String letterGrade = grade.
                        calculateLetterGrade();
                    BigDecimal gpaScore = grade.
                        calculateGpaScore();
                    grade.setLetterGrade(letterGrade);
                    grade.setGpaScore(gpaScore);
                    gradeRepository.save(grade);
                }
            }
        } catch (Exception e) {
        }
    }
    @Async
    public void updateAllLetterAndGpaGrades() {
        try {
            List<Grade> allGrades = gradeRepository.findAll();
            if (allGrades.isEmpty()) {
                return;
            }
            for (Grade grade : allGrades) {
                try {
                    if (grade.getAvgScore() != null) {
                        String oldLetter = grade.getLetterGrade
                            ();
                        BigDecimal oldGpa = grade.getGpaScore();
                        String newLetter = grade.
                            calculateLetterGrade();
                        BigDecimal newGpa = grade.
                            calculateGpaScore();
                        if (!newLetter.equals(oldLetter) ||
                            (newGpa != null && !newGpa.equals(
                                oldGpa))) {
                            grade.setLetterGrade(newLetter);
                            grade.setGpaScore(newGpa);
                            gradeRepository.save(grade);
                        }
                    }
                } catch (Exception e) {
                }
```

```
            }
        } catch (Exception e) {
        }
    }
}
```

Listing 3.6: GradeCalculationService.java - Tính toán điểm chữ và GPA

### 3.5.3 Tự động cập nhật dữ liệu cũ (Startup Task)

Để đảm bảo dữ liệu cũ (được nhập trước khi có logic tính toán) cũng được cập nhật, ‘GradeCalculationService sử dụng ‘@EventListener(ApplicationReadyEvent.class)‘.

Khi ứng dụng Spring Boot khởi động xong, sự kiện ‘ApplicationReadyEvent‘ được bắn ra, kích hoạt hàm ‘autoUpdateAllGradesOnStartup‘. Hàm này duyệt qua *tất cả* các bản ghi ‘grades‘ trong CSDL, tính toán lại và lưu ‘avgScore‘, ‘letterGrade‘, ‘gpaScore‘ cho từng bản ghi.

```
@Service
@Transactional
public class GradeCalculationService {
    // ... (Autowired repositories)
    @EventListener(ApplicationReadyEvent.class)
    public void autoUpdateAllGradesOnStartup() {
        try {
            List<Grade> allGrades = gradeRepository.findAll();
            if (allGrades.isEmpty()) return;

            for (Grade grade : allGrades) {
                // LUÔN TÍNH ẠLI ĐỂIM TRUNG BÌNH
                Double avg = analyticsService.
                   calculateGradeAverage(grade);
                grade.setAvgScore(avg != null ?
                    BigDecimal.valueOf(avg) : BigDecimal.ZERO);
                grade.setLetterGrade(grade.calculateLetterGrade
                   ());
                grade.setGpaScore(grade.calculateGpaScore());

                gradeRepository.save(grade);
            }
        } catch (Exception e) {
        }
    }
    public void calculateAndUpdateGradeAvg(Long gradeId) {
        Grade grade = gradeRepository.findById(gradeId)
            .orElseThrow(() -> new RuntimeException("..."));

        Double avg = analyticsService.calculateGradeAverage(
           grade);
        grade.setAvgScore(avg != null ? BigDecimal.valueOf(avg)
           : null);

        grade.setLetterGrade(grade.calculateLetterGrade());
        grade.setGpaScore(grade.calculateGpaScore());
```

```
33
34              gradeRepository.save(grade);
35          }
36      }
```

Listing 3.7: GradeCalculationService.java - Tự động cập nhật khi khởi động

## 3.6 Module Quản lý Tài liệu (Document)

Module này xử lý việc tải lên và tải xuống file.

### 3.6.1 Upload File

‘DocumentController‘ định nghĩa logic upload:

1. Nhận ‘MultipartFile‘, ‘userId‘, và ‘subjectId‘ (tùy chọn).
2. Kiểm tra định dạng file (MIME type và extension) có nằm trong danh sách ‘ALLOWED_MIME_TYPE (pdf, doc, docx) hay không.
3. Tạo tên file lưu trữ duy nhất bằng ‘UUID.randomUUID()‘ để tránh trùng lặp.
4. Lưu file vật lý vào thư mục ‘/uploads‘ trên server.
5. Lưu metadata (tên file gốc, đường dẫn file đã lưu, ‘userId‘, ‘subjectId‘,...) vào bảng ‘documents‘ trong CSDL thông qua ‘DocumentRepository‘.

### 3.6.2 Download File

Khi download, ‘DocumentController‘ dùng ‘documentId‘ để tìm file trong CSDL, lấy ‘filePath‘ và ‘fileType‘. Sau đó, nó đọc file từ thư mục ‘/uploads‘ bằng ‘UrlResource‘, set ‘ContentType‘ và ‘CONTENT_DISPOSITION‘ (với tên file gốc) để trình duyệt tự động tải về.

```
1   @RestController
2   @RequestMapping("/api/documents")
3   public class DocumentController {
4
5       private final Path fileStorageLocation;
6       private static final List<String> ALLOWED_MIME_TYPES =
            Arrays.asList(
7               "application/pdf",
8               "application/msword",
9               "application/vnd.openxmlformats-officedocument.
                    wordprocessingml.document");
10
11      public DocumentController() throws IOException {
12          this.fileStorageLocation = Paths.get("uploads")
13              .toAbsolutePath().normalize();
14          if (!Files.exists(this.fileStorageLocation)) {
15              Files.createDirectories(this.fileStorageLocation);
16          }
17      }
18
```

```
    @PostMapping("/upload")
    public ResponseEntity<?> uploadDocument(
            @RequestParam("file") MultipartFile file,
            @RequestParam("userId") Long userId,
            @RequestParam(value = "subjectId", required = false)
                Long subjectId,
            ...) {

        String fileType = file.getContentType();
        if (!ALLOWED_MIME_TYPES.contains(fileType)) {
            return ResponseEntity.badRequest().body(...);
        }

        String storedFileName = UUID.randomUUID() + "." +
           fileExtension;
        Path targetLocation = this.fileStorageLocation
            .resolve(storedFileName);

        Files.copy(file.getInputStream(), targetLocation,
            StandardCopyOption.REPLACE_EXISTING);

        Document doc = new Document();
        doc.setFileName(...);
        doc.setFilePath(targetLocation.toString());
        doc.setFileType(fileType);
        doc.setUserId(userId);
        doc.setSubjectId(subjectId);
        documentRepository.save(doc);

        return ResponseEntity.ok(Map.of("message", "Upload thanh
            cong"));
    }
}
```

Listing 3.8: DocumentController.java - Upload File

## 3.7 Module Analytics và Thống kê

Module này cung cấp các API cho frontend để vẽ biểu đồ và hiển thị thống kê tổng quan.

### 3.7.1 AnalyticsService

‘AnalyticsService‘ thực hiện các tính toán thống kê. Điểm mấu chốt là service này không tính toán lại GPA. Thay vào đó, nó tận dụng dữ liệu đã được tính toán và lưu sẵn:

- ‘calculateOverallGPA(userId)‘: Tính GPA tích lũy. Hàm này lấy tất cả học kỳ của user, duyệt qua từng học kỳ, lấy ‘semesterGpa‘ (đã được ‘SemesterGpaService‘ tính và lưu sẵn) và tổng số tín chỉ của học kỳ đó để tính trung bình có trọng số.
- ‘getSemesterChartData(userId)‘: Chuẩn bị dữ liệu cho biểu đồ. Hàm này cũng lấy danh sách học kỳ và ‘semesterGpa‘ đã được lưu sẵn để trả về (labels, gpaData).

Việc này giúp các API thống kê phản hồi rất nhanh, vì không phải truy vấn và tính toán lại toàn bộ bảng điểm mỗi lần gọi.

```
@Service
public class AnalyticsService {
    @Autowired
    private SemesterRepository semesterRepository;
    @Autowired
    private SubjectRepository subjectRepository;

    public Map<String, Object> calculateOverallGPA(Long userId)
       {
        List<Semester> semesters = semesterRepository.
           findByUserId(userId);

        double totalWeightedScore = 0;
        int totalCredits = 0;

        for (Semester semester : semesters) {

            BigDecimal semesterGpa = semester.getSemesterGpa();

            if (semesterGpa != null &&
                semesterGpa.compareTo(BigDecimal.ZERO) > 0) {

                List<Subject> subjects = subjectRepository
                    .findBySemesterId(semester.getId());
                int semesterCredits = subjects.stream()
                    .mapToInt(Subject::getCredits).sum();

                totalWeightedScore += semesterGpa.doubleValue()
                    * semesterCredits;
                totalCredits += semesterCredits;
            }
        }
        double overallGpa = totalCredits > 0 ?
            round(totalWeightedScore / totalCredits) : 0.0;
        return Map.of("overallGpa", overallGpa, ...);
    }
}
```

Listing 3.9: AnalyticsService.java - Tính GPA tích lũy

## 3.8 Module AI Chatbot

Đây là một tính năng nâng cao, áp dụng kỹ thuật RAG (Retrieval-Augmented Generation) đơn giản.

### 3.8.1 Luồng xử lý Chat

1. **Request:** Frontend gửi ‘ChatRequest‘ (chứa ‘message‘ và ‘studentId‘) đến ‘AIChatController‘.

2. **Context Building:** ‘AdvancedAIChatService‘ được gọi. Hàm ‘buildSmartContext‘ thực hiện:
    - Truy vấn bảng ‘users‘ để lấy thông tin sinh viên (Tên, MSSV).
    - Truy vấn (JOIN) các bảng ‘semesters‘, ‘subjects‘, ‘grades‘ để lấy toàn bộ dữ liệu học tập (thống kê tổng quan, các môn học, điểm số chi tiết, điểm TB, điểm chữ).
    - Xây dựng một chuỗi (string) context lớn chứa toàn bộ thông tin này.
3. **Call API:** ‘OpenRouterAIService‘ được gọi, nhận ‘userMessage‘ và ‘databaseContext‘.
4. **System Prompt:** Dịch vụ này tạo một ”system prompt” (lời nhắc hệ thống) đặc biệt. Lời nhắc này bao gồm vai trò của AI (”Bạn là trợ lý AI...”) và toàn bộ chuỗi context dữ liệu học tập.
5. **AI Response:** Request (gồm system prompt và user message) được gửi đến API OpenRouter, sử dụng model ‘openai/gpt-4o-mini‘. AI sẽ đọc context và trả lời câu hỏi của người dùng dựa trên dữ liệu đó (ví dụ: ”Tôi qua môn Lập trình hướng đối tượng chưa?”, ”Điểm TB của tôi là bao nhiêu?”).
6. **Save History:** Câu hỏi và câu trả lời của AI được lưu lại vào bảng ‘chat_messages‘.
7. **Response:** ‘AIChatController‘ trả ‘ChatResponse‘ về cho frontend.

```
@Service
public class AdvancedAIChatService {
    public ChatResponse processAdvancedMessage(ChatRequest
       request) {
        try {

            String databaseContext = buildSmartContext(request);

            String aiResponse = openRouterAIService.
               getAIResponse(
                request.getMessage(), databaseContext);

            saveChatHistory(request, aiResponse);

            ChatResponse response = new ChatResponse();
            response.setResponse(aiResponse);

            return response;
        } catch (Exception e) {
        }
    }

    private String buildSmartContext(ChatRequest request) {
        StringBuilder context = new StringBuilder();

        try {
            User user = findUser(request);
            if (user != null) {
                context.append("Thong tin sinh vien");
                context.append("• Ho ten: ").append(user.
                   getFullName()).append("\n");
```

```
29
30              }
31
32              context.append("\nThong ke hoc tap:");
33              context.append(getLearningStatistics(user));
34              context.append("\n");
35
36              context.append("Du lieu hoc tap chi tiet:");
37              context.append(getAllAcademicData(user));
38
39          } catch (Exception e) {
40              context.append("\nLoi: ")
41                  .append(e.getMessage());
42          }
43          return context.toString();
44      }
45  }
```

Listing 3.10: AdvancedAIChatService.java - Xây dựng Context

```
1   @Service
2   public class OpenRouterAIService {
3       @Value("${openrouter.api.key}")
4       private String openrouterApiKey;
5       private final String OPENROUTER_API_URL =
6           "https://openrouter.ai/api/v1/chat/completions";
7
8       public String getAIResponse(String userMessage, String
            context) {
9           // ...
10          String response = callOpenRouterAPI(userMessage, context
                );
11          return response;
12      }
13
14      private String callOpenRouterAPI(String userMessage, String
            context) {
15          CloseableHttpClient httpClient = HttpClients.
                createDefault();
16          try {
17              HttpPost httpPost = new HttpPost(OPENROUTER_API_URL)
                    ;
18
19              String requestBody = createRequestBody(userMessage,
                    context);
20
21              httpPost.setHeader("Authorization", "Bearer " +
                    openrouterApiKey);
22
23              httpPost.setEntity(new StringEntity(requestBody, "
                    UTF-8"));
24
25              CloseableHttpResponse response = httpClient.execute(
                    httpPost);
```

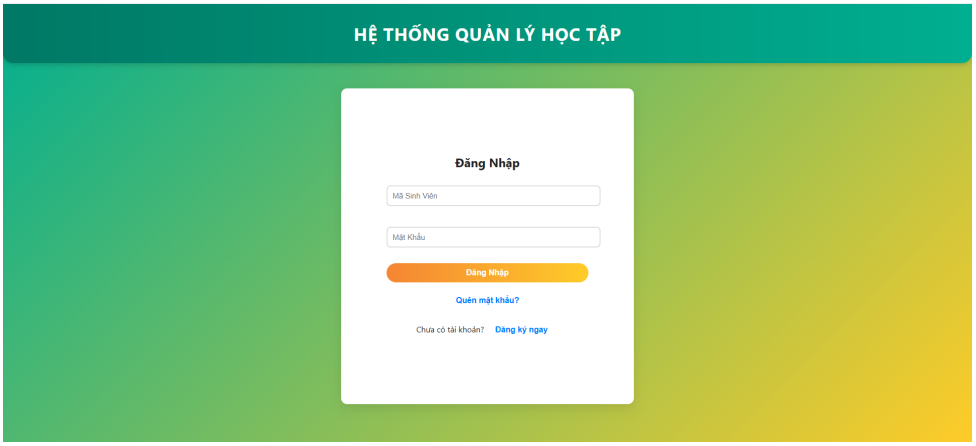
```
26
27              String aiResponse = parseOpenRouterResponse(
                    responseBody);
28              return aiResponse;
29          } catch (Exception e) {
30              // ...
31          }
32      }
33
34      private String createRequestBody(String userMessage, String
            context) {
35
36          String systemPrompt = createSystemPrompt(context);
37
38          requestMap.put("model", "openai/gpt-4o-mini");
39          java.util.Map<String, String> systemMessage = new java.
                util.HashMap<>();
40          systemMessage.put("role", "system");
41          systemMessage.put("content", systemPrompt);
42          messages.add(systemMessage);
43
44          java.util.Map<String, String> userMsg = new java.util.
                HashMap<>();
45          userMsg.put("role", "user");
46          userMsg.put("content", userMessage);
47          messages.add(userMsg);
48          // ...
49      }
50
51      private String createSystemPrompt(String context) {
52          return String.format("""
53              PROMT
54               """, context != null ? context : "Chua co du lieu.")
                     ;
55      }
56  }
```

Listing 3.11: OpenRouterAIService.java - Gọi API AI

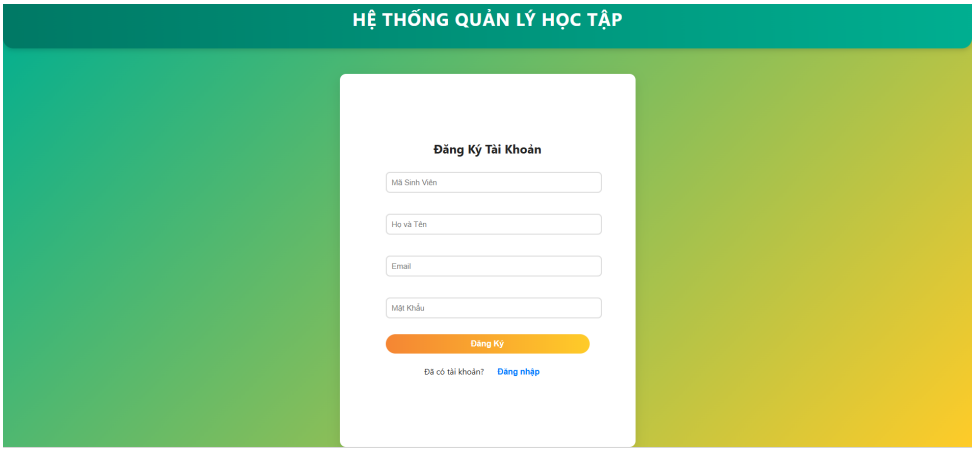
# Chương 4

# Demo website

## 4.1 Giao diện đăng nhập/đăng ký

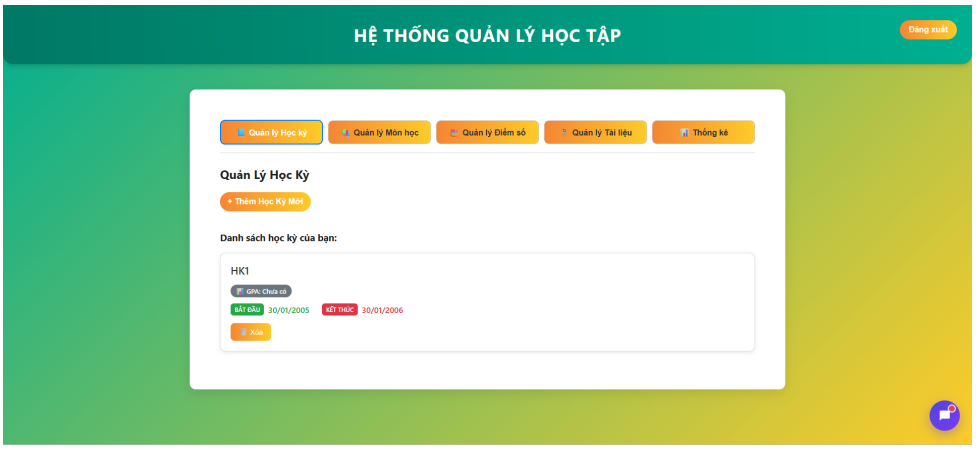



Hình 4.1. Giao diện đăng nhập



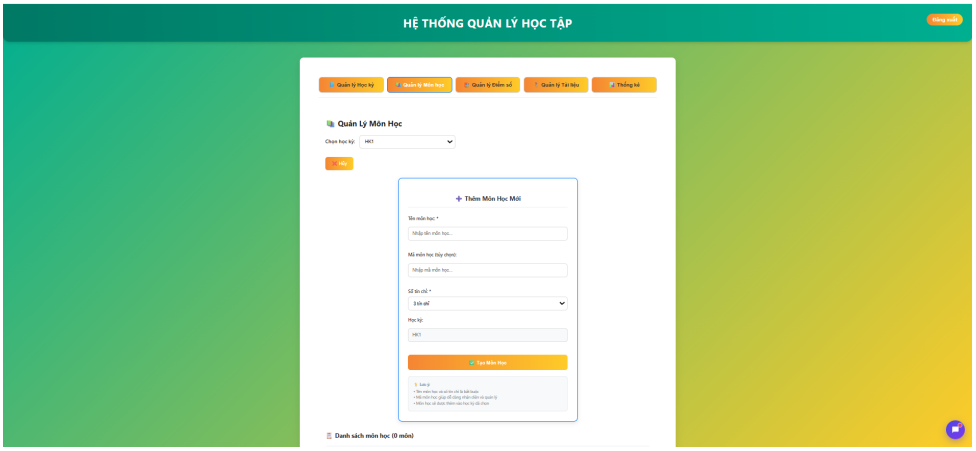

Hình 4.2. Giao diện đăng ký

## 4.2 Giao diện quản lý học kỳ



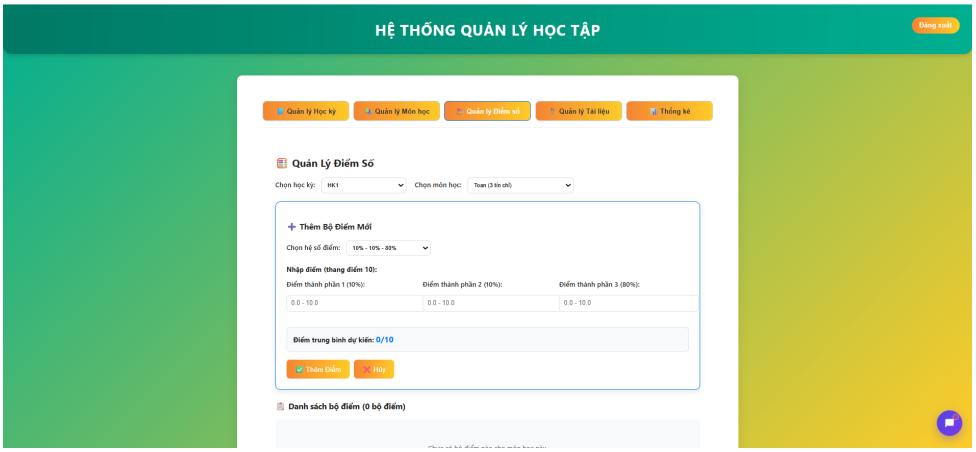

Hình 4.3. Giao diện tạo học kỳ mới

## 4.3 Giao diện quản lý môn học



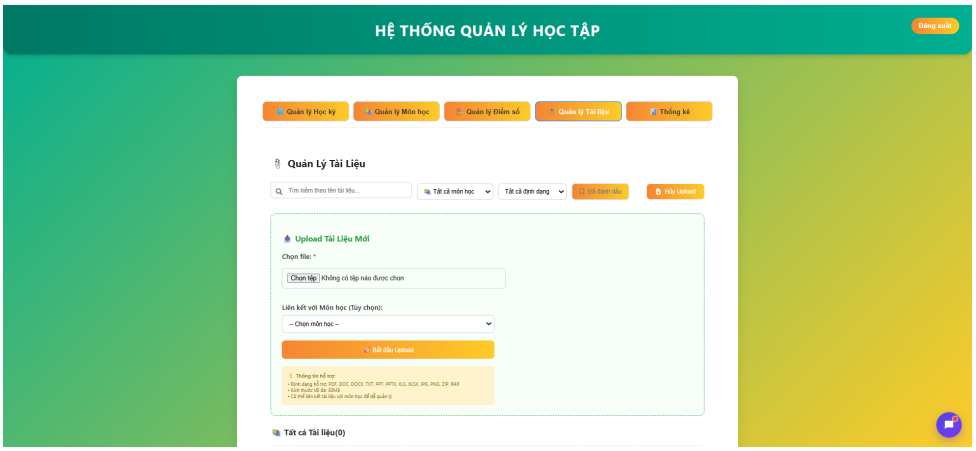

Hình 4.4. Giao diện tạo môn học mới

## 4.4 Giao diện quản lý điểm



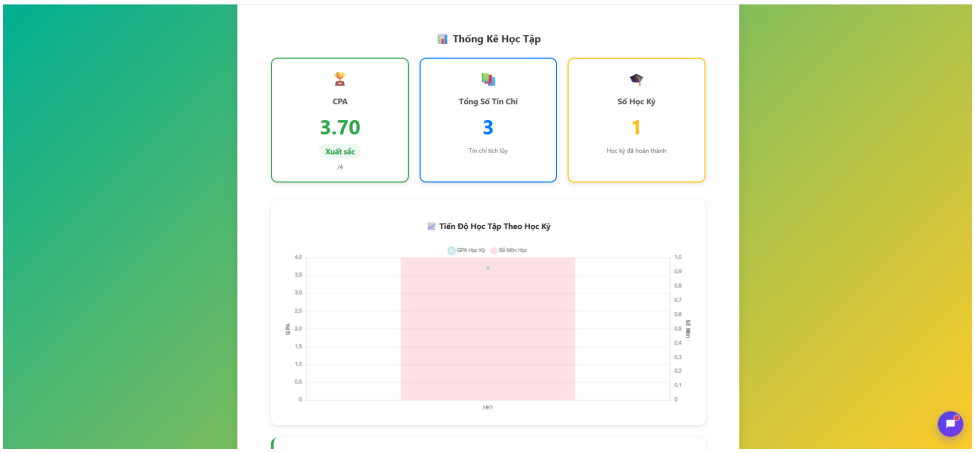

Hình 4.5. Giao diện nhập điểm

## 4.5 Giao diện quản lý tài liệu




Hình 4.6. Giao diện quản lý tài liệu

## 4.6 Giao diện thống kê




Hình 4.7. Giao diện thống kê điểm và GPA

## 4.7 Giao diện chatbot


Hình 4.8. Giao diện chatbot hỗ trợ sinh viên

# Chương 5

# Kết quả và đánh giá

## 5.1 Kết quả đạt được

### 5.1.1 Chức năng đã triển khai

Hệ thống đã hoàn thiện các chức năng sau:

1. **Authentication và Authorization:**
    - Đăng ký tài khoản với mã hóa BCrypt
    - Đăng nhập với xác thực password
    - Quên mật khẩu với email token (15 phút)
    - Đặt lại mật khẩu an toàn
2. **Quản lý học kỳ:**
    - Tạo, xem, xóa học kỳ
    - Tính GPA học kỳ tự động
    - Hiển thị danh sách học kỳ theo thời gian
3. **Quản lý môn học:**
    - Thêm môn học vào học kỳ
    - Quản lý tín chỉ và mã môn
    - Xóa môn học (cascade xóa điểm)
4. **Quản lý điểm số:**
    - Nhập điểm theo template linh hoạt
    - Tính điểm trung bình tự động
    - Quy đổi điểm chữ (A+, A, B+,...)
    - Tính GPA thang 4.0
    - Cập nhật điểm và tính lại tự động
5. **Quản lý tài liệu:**
    - Upload file
    - Download tài liệu đã lưu
    - Bookmark tài liệu yêu thích
    - Tìm kiếm tài liệu theo tên
    - Gắn tài liệu với môn học
6. **Thống kê và Analytics:**

- Tính GPA tổng hợp toàn khóa
- Thống kê số môn, số tín chỉ
- Biểu đồ GPA theo học kỳ
- Phân tích điểm chữ phổ biến

7. **AI Chatbot:**
    - Tích hợp OpenRouter API (GPT-4o-mini)
    - Trả lời câu hỏi về điểm số
    - Phân tích kết quả học tập
    - Đưa ra lời khuyên học tập
    - Lưu lịch sử chat

### 5.1.2 Áp dụng OOP

Hệ thống đã thành công trong việc áp dụng các nguyên lý OOP:

| Nguyên lý | Cách áp dụng |
|---|---|
| Encapsulation | Tất cả entity có private fields với getter/setter. Repository che giấu chi tiết truy xuất DB. |
| Abstraction | Service layer trừu tượng hóa business logic. Interface RowMapper định nghĩa mapping behavior. |
| Inheritance | Controller kế thừa annotations của Spring. Entity có thể kế thừa base class. |
| Polymorphism | Dependency Injection cho phép thay thế implementation. Method overriding trong RowMapper. |

Bảng 5.1. Áp dụng OOP trong hệ thống

### 5.1.3 Design Patterns sử dụng

1. **MVC Pattern:** Phân tách Model-View-Controller
2. **DAO Pattern:** Repository layer truy xuất dữ liệu
3. **Service Pattern:** Business logic trong Service layer
4. **Singleton Pattern:** Spring Bean mặc định là singleton
5. **Factory Pattern:** KeyHolder factory tạo generated key
6. **Template Method:** JdbcTemplate cho database operations

## 5.2 Testing và Validation

### 5.2.1 Unit Testing

Hệ thống được test với các trường hợp:

- Test đăng ký với student_id đã tồn tại
- Test đăng nhập với sai password
- Test tính điểm trung bình với các template khác nhau
- Test quy đổi điểm chữ ở các ngưỡng
- Test upload file với định dạng không hợp lệ
- Test token reset password hết hạn

### 5.2.2 Integration Testing

- Test cascade delete khi xóa semester
- Test tính GPA tự động khi thêm/sửa/xóa điểm

## 5.3 Bảo mật

### 5.3.1 Các biện pháp bảo mật đã áp dụng

1. **Password Security:**
    - Mã hóa BCrypt với salt tự động
    - Không lưu plaintext password
    - Validation độ dài tối thiểu
2. **Token Security:**
    - UUID random cho reset token
    - Thời gian hết hạn 15 phút
    - Xóa token sau khi sử dụng
3. **Input Validation:**
    - Validate file type khi upload
    - Sanitize SQL parameters
    - CORS configuration
4. **SQL Injection Prevention:**
    - Sử dụng PreparedStatement
    - Parameterized queries
    - JdbcTemplate escaping

## 5.4 Hạn chế và vấn đề gặp phải

### 5.4.1 Hạn chế kỹ thuật

- Chưa implement JWT authentication đầy đủ
- AI response phụ thuộc vào external API
- File storage trên local disk (chưa dùng cloud)

# Chương 6

# Hướng phát triển và kết luận

## 6.1 Hướng phát triển tương lai

### 6.1.1 Tính năng mới

1. **Authentication nâng cao:**
    - Triển khai JWT authentication đầy đủ
    - Two-factor authentication (2FA)
    - OAuth2 integration (Google, Facebook)
    - Session management
2. **Role-based Authorization:**
    - Role STUDENT: xem điểm, upload tài liệu
    - Role ADMIN: quản lý toàn bộ hệ thống
3. **Notification System:**
    - Email thông báo điểm mới
    - Push notification
    - Nhắc nhở deadline
    - Thông báo từ giảng viên
4. **Advanced Analytics:**
    - Machine learning dự đoán kết quả
    - Phân tích xu hướng học tập
    - So sánh với trung bình lớp
    - Recommendation system
5. **Mobile Application:**
    - Phát triển app iOS/Android
    - Sync dữ liệu real-time
    - Offline mode
    - Push notification native

## 6.2 Bài học kinh nghiệm

### 6.2.1 Kiến thức kỹ thuật

1. **Spring Boot Framework:**
    - Hiểu sâu về Dependency Injection
    - Làm việc với Spring JDBC Template
    - Configuration và Annotations
    - RESTful API best practices
2. **Database Design:**
    - Thiết kế schema chuẩn hóa
    - Foreign key constraints
    - Indexing strategy
    - Transaction management
3. **OOP Principles:**
    - Áp dụng SOLID principles
    - Design patterns thực tế
    - Code organization
    - Separation of concerns
4. **API Integration:**
    - HTTP client implementation
    - JSON parsing
    - Error handling
    - Rate limiting awareness

### 6.2.2 Quản lý dự án

1. **Teamwork:**
    - Phân chia công việc rõ ràng
    - Code review practices
    - Git workflow
    - Communication skills
2. **Development Process:**
    - Incremental development
    - Testing-driven approach
    - Documentation importance
    - Version control

### 6.2.3 Kỹ năng giải quyết vấn đề

- Debug hiệu quả với logging
- Research và đọc documentation
- Stack Overflow và community support
- Trial and error methodology
- Root cause analysis

## 6.3 Kết luận

### 6.3.1 Đánh giá tổng quan

Dự án "Hệ thống Quản lý Điểm Sinh Viên" đã đạt được các mục tiêu đề ra:

- **Áp dụng thành công OOP:** Hệ thống được thiết kế theo các nguyên lý OOP với kiến trúc phân tầng rõ ràng, dễ bảo trì và mở rộng.
- **Chức năng hoàn chỉnh:** Triển khai đầy đủ các module quản lý user, học kỳ, môn học, điểm số, tài liệu và AI chatbot.
- **Công nghệ hiện đại:** Sử dụng Spring Boot, MySQL, RESTful API và tích hợp AI, đáp ứng xu hướng phát triển phần mềm hiện nay.
- **Bảo mật tốt:** Mã hóa password, token authentication, input validation đảm bảo an toàn dữ liệu.
- **User Experience:** Tính toán điểm tự động, AI hỗ trợ thông minh mang lại trải nghiệm tốt cho người dùng.

### 6.3.2 Ý nghĩa của dự án

**Ý nghĩa học thuật**

- Củng cố kiến thức về Lập trình Hướng đối tượng
- Thực hành với framework Spring Boot chuyên nghiệp
- Hiểu sâu về database design và SQL
- Làm việc với RESTful API và integration

**Ý nghĩa thực tiễn**

- Giải quyết bài toán thực tế trong quản lý giáo dục
- Có thể triển khai cho trường học thực tế
- Nền tảng cho các dự án lớn hơn
- Portfolio project cho sinh viên

### 6.3.3 Lời cảm ơn

Nhóm em xin chân thành cảm ơn thầy giáo Ngô Tiến Đức đã tận tình hướng dẫn và truyền đạt kiến thức về Lập trình Hướng đối tượng. Những kiến thức này là nền tảng quan trọng giúp nhóm hoàn thành dự án.

Nhóm cũng xin cảm ơn các thành viên đã nhiệt tình tham gia, hỗ trợ lẫn nhau và cùng nhau vượt qua những khó khăn trong quá trình thực hiện.

Mặc dù đã cố gắng hết sức, báo cáo không tránh khỏi những thiếu sót. Nhóm rất mong nhận được sự góp ý từ thầy để hoàn thiện hơn nữa.

*Hà Nội, 18 tháng 11 năm 2025*
*Nhóm 07*

# Phụ lục

## Phụ lục A: Cấu trúc thư mục dự án

```
student-management-system/
   backend/
       src/
           main/
               java/
                   com/studentmgmt/backend/
                       config/
                           SecurityConfig.java
                       controller/
                           AuthController.java
                           SemesterController.java
                           SubjectController.java
                           GradeController.java
                           DocumentController.java
                           AIChatController.java
                           AnalyticsController.java
                       model/
                           User.java
                           Semester.java
                           Subject.java
                           Grade.java
                           Document.java
                           ChatMessage.java
                           PasswordResetToken.java
                       repository/
                           UserRepository.java
                           SemesterRepository.java
                           SubjectRepository.java
                           GradeRepository.java
                           DocumentRepository.java
                           ChatMessageRepository.java
                           PasswordResetTokenRepository.java
                       service/
                           GradeCalculationService.java
                           SemesterGpaService.java
                           AnalyticsService.java
                           AdvancedAIChatService.java
                           OpenRouterAIService.java
                           EmailService.java
                           PasswordResetTokenService.java
                       dto/
                           ChatRequest.java
                           ChatResponse.java
                       security/
                           JwtTokenUtil.java
```

```
                    CustomUserDetailsService.java
                BackendApplication.java
            resources/
                application.properties
    pom.xml
database/
    schema.sql
    Dump20251028.sql
README.md
```

# Phụ lục B: Configuration Files

## application.properties

```
# Database Configuration
spring.datasource.url=jdbc:mysql://localhost:3306/
    student_management?useSSL=false
spring.datasource.username=root
spring.datasource.password=your_password
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver

# JPA Configuration
spring.jpa.hibernate.ddl-auto=none
spring.jpa.show-sql=true

# Server Configuration
server.port=8080

# File Upload
spring.servlet.multipart.max-file-size=10MB
spring.servlet.multipart.max-request-size=10MB

# Mail Configuration
spring.mail.host=smtp.gmail.com
spring.mail.port=587
spring.mail.username=your-email@gmail.com
spring.mail.password=your-app-password
spring.mail.properties.mail.smtp.auth=true
spring.mail.properties.mail.smtp.starttls.enable=true

# OpenRouter API
openrouter.api.key=your-openrouter-api-key

# JWT Configuration
jwt.secret=your-jwt-secret-key-at-least-256-bits
jwt.expiration=86400000
```

## pom.xml

```xml
<dependencies>
    <!-- Spring Boot Starter Web -->
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-web</artifactId>
    </dependency>

    <!-- Spring Boot Starter JDBC -->
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-jdbc</artifactId>
    </dependency>

    <!-- MySQL Connector -->
    <dependency>
        <groupId>mysql</groupId>
        <artifactId>mysql-connector-java</artifactId>
        <version>8.0.33</version>
    </dependency>

    <!-- Spring Security -->
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-security</artifactId>
    </dependency>

    <!-- Spring Mail -->
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-mail</artifactId>
    </dependency>

    <!-- JWT -->
    <dependency>
        <groupId>io.jsonwebtoken</groupId>
        <artifactId>jjwt-api</artifactId>
        <version>0.11.5</version>
    </dependency>

    <!-- Apache HttpClient -->
    <dependency>
        <groupId>org.apache.httpcomponents</groupId>
        <artifactId>httpclient</artifactId>
        <version>4.5.14</version>
    </dependency>

    <!-- Lombok -->
    <dependency>
        <groupId>org.projectlombok</groupId>
        <artifactId>lombok</artifactId>
        <optional>true</optional>
    </dependency>
</dependencies>
```

Listing 6.1: Maven Dependencies

# Phụ lục C: API Endpoints Documentation

## Authentication APIs

| Method | Endpoint | Description |
|---|---|---|
| POST | /api/auth/register | Đăng ký tài khoản mới |
| POST | /api/auth/login | Đăng nhập hệ thống |
| POST | /api/auth/forgot-password | Gửi email reset password |
| POST | /api/auth/reset-password | Đặt lại mật khẩu |

Bảng 6.1. Authentication APIs

## Semester APIs

| Method | Endpoint | Description |
|---|---|---|
| GET | /api/semesters | Lấy danh sách học kỳ |
| GET | /api/semesters/{id} | Lấy chi tiết học kỳ |
| POST | /api/semesters | Tạo học kỳ mới |
| DELETE | /api/semesters/{id} | Xóa học kỳ |
| POST | /api/semesters/{id}/calculate-gpa | Tính GPA học kỳ |

Bảng 6.2. Semester APIs

## Grade APIs

| Method | Endpoint | Description |
|---|---|---|
| GET | /api/grades/subject/{id} | Lấy điểm theo môn học |
| GET | /api/grades/user/{id} | Lấy tất cả điểm của user |
| POST | /api/grades | Tạo điểm mới |
| PUT | /api/grades/{id} | Cập nhật điểm |
| DELETE | /api/grades/{id} | Xóa điểm |
| POST | /api/grades/{id}/calculate-avg | Tính điểm TB |

Bảng 6.3. Grade APIs

# Tài liệu tham khảo


1. Spring Framework Documentation. (2024). *Spring Boot Reference Documentation*. `https://docs.spring.io/spring-boot/docs/current/reference/html/`
2. MySQL Documentation. (2024). *MySQL 8.0 Reference Manual*. `https://dev.mysql.com/doc/refman/8.0/en/`
3. Bloch, J. (2018). *Effective Java* (3rd ed.). Addison-Wesley Professional.
4. Walls, C. (2022). *Spring in Action* (6th ed.). Manning Publications.
5. OpenRouter API Documentation. (2024). `https://openrouter.ai/docs`
6. Martin, R. C. (2008). *Clean Code: A Handbook of Agile Software Craftsmanship*. Prentice Hall.