3. Thiết kế (Use-case design)

3.1. Xác định các thành phần thiết kế (Identify design elements)

3.1.1. Xác định các lớp (Identify classes)

STT	Tên lớp	Phương thức
1	StudentForm	+ selectManageStudents() + displayStudentUI(studentList) + displayAddForm() + updateAndDisplayStudentList() + displayStudentInfo(studentData)…
2	GradeForm	+ displayGradeUI(studentGrades) + displayAddGradeForm(studentID) + displayEditGradeForm(gradeData)
3	ScoreLookupForm	+ displayScoreLookupUI() + requestStudentGrades(studentID)
4	DepartmentForm	+ displayDepartmentUI(departmentList) + displayAddDepartmentForm()
5	AccountForm	+ displayLoginForm() + displayChangePasswordForm()
6	StudentController	+ requestStudentList() + createStudent(studentData) + requestDetails(studentID) + updateInfo(studentData) + requestDeletion(studentID) + requestSearch(keyword)
7	ScoreLookupController	+ requestStudentGrades(studentID)
8	GradeController	+ requestGradeList(studentID) + addGrade(gradeData) + updateGrade(gradeData)
9	CourseRegistrationController	+ requestAvailableCourses() + registerCourse(studentID, courseID)
10	DepartmentController	+ requestDepartmentList() + addDepartment(departmentData)
11	AccountController	+ validateLogin(username, password) + updatePassword(userID, newPassword)

STT	Tên lớp	Phương thức
12	Student	+ createStudentObject(studentData) + retrieveStudentData(studentID) + deleteStudent(studentID)
13	Grade	+ createGradeObject(gradeData) + retrieveGrades(studentID) + updateGrade(gradeData)
14	Course	+ createCourseObject(courseData) + retrieveCourseData(courseID)
15	Department	+ createDepartmentObject(departmentData) + retrieveDepartmentData(departmentID)
16	Account	+ createAccountObject() + createAccountObject(accountData) + getAccountInfo(accountID) + updateAccountInfo(accountID) + deleteAccountRecord(accountID)
17	RoleGroup	+ createRoleGroupList(RoleGroupList) + createRoleGroupObject(groupData) + retrieveData(groupID) + deleteRoleGroup(groupData)

Bảng 3.1 Xác định các lớp

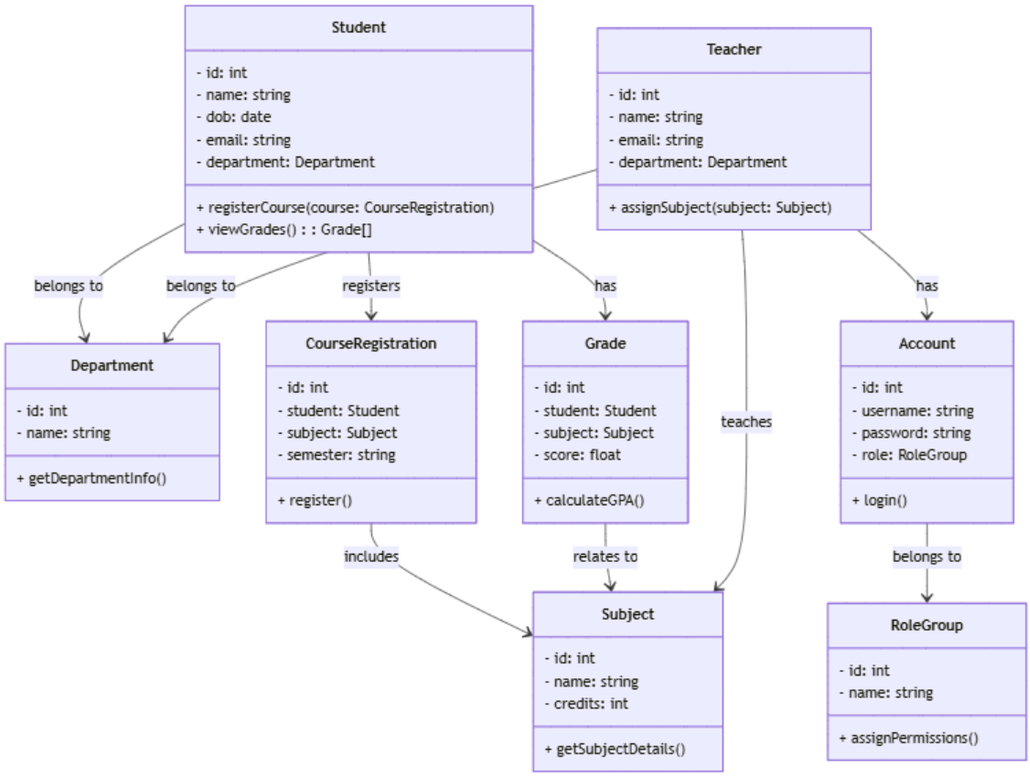
3.1.2. Xác định các gói (Identify packages)

STT	Tên gói	Thành phần
HCI layer		
1	UI	- StudentForm - SubjectForm - GradeForm - DepartmentForm - CourseRegistrationForm
PD layer		
2	StudentManagement	- StudentController - Student
3	SubjectManagement	- SubjectController - Subject

STT	Tên gói	Thành phần
4	GradeManagement	- GradeController - Grade
5	CourseRegistrationManagement	- CourseRegistrationManagementController - CourseRegistrationManagement
DM layer		
6	StudentDataManagement	- StudentAPI - Các bảng dữ liệu thuộc gói SinhVien trong CSDL
7	SubjectDataManagement	- SubjectAPI - Các bảng dữ liệu thuộc gói MonHoc trong CSDL
8	DepartmentDataManagement	- DepartmentAPI - Các bảng dữ liệu thuộc gói Khoa trong CSDL

Bảng 3.2 Xác định các gói

3.1.2. Thiết kế biểu đồ lớp (Class diagrams)



Hình 3.1 Biểu đồ lớp

Sơ đồ lớp này mô tả các thành phần chính trong hệ thống quản lý sinh viên, bao gồm quản lý sinh viên, môn học, điểm số, tài khoản, giảng viên và các dịch vụ liên quan.

- **Student (Sinh viên)** có thể đăng ký nhiều môn học (**Subject**) và có bảng điểm (**Grade**) để lưu trữ kết quả học tập.
- **Subject (Môn học)** có thể có nhiều giảng viên (**Teacher**) phụ trách giảng dạy, và mỗi môn học đều liên kết với điểm số để theo dõi quá trình học tập của sinh viên.
- **Department (Khoa)** đóng vai trò tổ chức các môn học và giảng viên trong hệ thống, giúp phân loại và quản lý các nhóm môn học chuyên ngành.

Bên cạnh đó, **CourseRegistration** là bảng đăng ký học phần của sinh viên, dùng để quản lý thông tin đăng ký các môn học theo từng kỳ. Điều này giúp theo dõi tiến trình học tập và lịch trình môn học của sinh viên trong hệ thống.

Phần quản lý tài khoản trong hệ thống được thể hiện qua các lớp **Account** và **Teacher**, trong đó mỗi giảng viên có một tài khoản (**Account**) để truy cập vào hệ thống. Tài khoản này được gán cho một **RoleGroup (Nhóm vai trò)**, giúp phân quyền và xác định các chức năng mà người dùng có thể thực hiện trong hệ thống.

3.2. Thiết kế cơ sở dữ liệu (Database design)

3.2.1. Lược đồ cơ sở dữ liệu

Hình 3.2 Lược đồ cơ sơ dữ liệu

1. Bảng users (Người dùng)

Quản lý thông tin đăng nhập của người dùng trong hệ thống.

- **Các cột quan trọng:**
  - _id: Khóa chính (ObjectId).
  - name, email, password: Thông tin cá nhân của người dùng.
  - role: Vai trò của người dùng (quản trị viên, giảng viên, v.v.).
  - department: Liên kết với phòng ban của người dùng (ObjectId tham chiếu bảng departments).
  - createdAt: Ngày tạo tài khoản.

2. Bảng departments (Phòng ban)

Lưu thông tin về các phòng ban trong hệ thống.

- **Các cột quan trọng:**
  - _id: Khóa chính (ObjectId).
  - name, code: Tên và mã phòng ban.
  - manager: Người quản lý phòng ban (ObjectId tham chiếu bảng users).
  - createdAt: Ngày tạo phòng ban.

3. Bảng students (Sinh viên)

Quản lý thông tin sinh viên trong hệ thống.

- **Các cột quan trọng:**
  - _id: Khóa chính (ObjectId).
  - name: Tên sinh viên.
  - studentId: Mã số sinh viên.
  - department: Phòng ban mà sinh viên thuộc về (ObjectId tham chiếu bảng departments).
  - status: Trạng thái học tập của sinh viên.
  - createdAt, updatedAt: Ngày tạo và cập nhật thông tin.

4. Bảng subjects (Môn học)

Quản lý danh sách các môn học trong hệ thống.

- **Các cột quan trọng:**
  - _id: Khóa chính (ObjectId).
  - name, code: Tên và mã môn học.
  - department: Môn học thuộc phòng ban nào (ObjectId tham chiếu bảng departments).
  - createdAt, updatedAt: Ngày tạo và cập nhật thông tin.

5. Bảng grades (Điểm số)

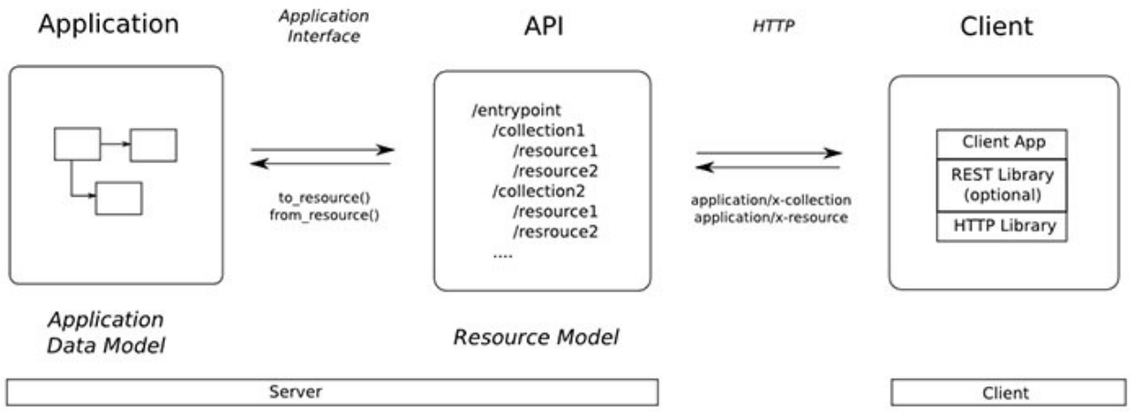
Lưu thông tin điểm số của sinh viên theo từng môn học.

- **Các cột quan trọng:**
  - _id: Khóa chính (ObjectId).
  - student: Sinh viên được chấm điểm (ObjectId tham chiếu bảng students).
  - subject: Môn học được chấm điểm (ObjectId tham chiếu bảng subjects).
  - grade: Điểm số của sinh viên.
  - createdAt, updatedAt: Ngày tạo và cập nhật thông tin.

Sơ đồ này thể hiện mối quan hệ giữa các thực thể trong hệ thống quản lý sinh viên, giúp dễ dàng truy xuất và quản lý thông tin.

3.4. Thiết kế API

3.4.1. Cấu trúc thư mục của API



Hình 3.3 Cấu trúc thư mục của API

REST hoạt động chủ yếu dựa vào giao thức HTTP. Các hoạt động cơ bản nêu trên sẽ sử dụng những phương thức HTTP riêng.

- **GET (SELECT):** Trả về một Resource hoặc một danh sách Resource.
- **POST (CREATE):** Tạo mới một Resource.
- **PUT (UPDATE):** Cập nhật thông tin cho Resource.
- **DELETE (DELETE):** Xoá một Resource.

Những phương thức hay hoạt động này thường được gọi là CRUD tương ứng với Create, Read, Update, Delete – Tạo, Đọc, Sửa, Xóa. [3]

3.4.2. Danh sách các API

STT	Chức năng	URL	Phương thức
1	Đăng nhập	/api/auth/login	POST
2	Đăng ký	/api/auth/register	POST
3	Đổi mật khẩu	/api/auth/change-password	PUT
4	Xác thực người dùng	/api/auth	POST,GET
5	Quản lý người dùng	/api/users	GET,POST,PUT,DELETE
6	Quản lý khoa/bộ môn	/api/departments	GET,POST,PUT,DELETE
7	Quản lý sinh viên	/api/students	GET,POST,PUT,DELETE
8	Quản lý môn học	/api/subjects	GET,POST,PUT,DELETE
9	Quản lý điểm số	/api/grades	GET,POST,PUT,DELETE
10	Đăng ký môn học	/api/courseRegistrations	GET,POST,PUT,DELETE

Bảng 3.3 Danh sách các API

3.4.3. Thiết kế chi tiết API

STT	URL	Mô tả chi tiết
1	/api/auth/login	Phương thức: POST Mục đích: Xác thực thông tin đăng nhập người dùng.

STT	URL	Mô tả chi tiết

		Tham số đầu vào: {
		mail: string,
		password: string
		}
		Kết quả trả về:
		- Thành công: {
		userid: int,
		result: boolean,
		success: true
		}
		- Thất bại: {
		result: false,
		success: true
		}

2	/api/auth/register	Phương thức: POST
		Mục đích: Tạo tài khoản quản lý cho người dùng.
		Tham số đầu vào: {
		mail: string,
		password: string
		}
		Kết quả trả về:
		- Thành công: {
		message: "Account added",
		success: true,
		Data: [{
		hoten: string,
		sodienthoai: string,
		mail: string,
		tenhienthi: string
		}]}
		- Thất bại: {
		message: "Registration failed",
		success: false}

3	/api/auth/change-password	Phương thức: PUT
		Mục đích: Đổi mật khẩu tài khoản người dùng.
		Tham số đầu vào: {
		mataikhoan: int,
		matkhaucu: string,
		maukhaumoi: string
		}
		Kết quả trả về:
		- Thành công: {
		message: "Password updated",
		success: true
		}

STT	URL	Mô tả chi tiết

		- Thất bại: {
		message: "Update failed",
		success: false
		}

4	/api/users	Phương thức: GET
		Mục đích: Lấy danh sách nhóm quyền.
		Tham số đầu vào: không có.
		Kết quả trả về:
		- Thành công: {
		"users": [
		{
		"id": "123",
		"name": "Nguyễn Văn A",
		"email": "userA@example.com",
		"role": "admin",
		"department": "IT"
		},
		{
		"id": "124",
		"name": "Trần Thị B",
		"email": "userB@example.com",
		"role": "teacher",
		"department": "Math"
		}
		],
		"success": true
		}
		- Thất bại: {
		roles: [],
		success: false
		}

		Phương thức: POST
		Mục đích: Thêm nhóm quyền vào cơ sở dữ liệu.
		Tham số đầu vào: {
		"name": "Nguyễn Văn C",
		"email": "userC@example.com",
		"password": "123456",
		"role": "student",
		"department": "Physics"
		}
		Kết quả trả về:
		- Thành công: {
		"message": "User added successfully",
		"success": true,
		"body": {
		"id": "125",

STT	URL	Mô tả chi tiết

		"name": "Nguyễn Văn C",
		"email": "userC@example.com",
		"role": "student",
		"department": "Physics"
		}
		}
		- Thất bại: {
		"message": "User creation failed",
		"success": false,
		"body": {}
		}
		
		Phương thức PUT:
		Mục đích: Cập nhật thông tin môn học
		Tham số đầu vào:{
		"id": "2001",
		"name": "Lập trình Python nâng cao",
		"code": "PY201",
		"department": "CNTT"
		}
		Kết quả trả về:{
		"message": "Subject updated",
		"success": true
		}
		
		Phương thức DELETE:
		Tham số đầu vào: {
		"id": "2001"
		}
		Kết quả trả về: {
		"message": "Subject deleted",
		"success": true
		}

5	/api/auth	Phương thức: POST
		Mục đích: Đăng nhập người dùng.
		Tham số đầu vào: {
		"email": "user@example.com",
		"password": "123456"
		}
		Kết quả trả về:
		- Thành công: {
		"message": "Login successful",
		"success": true,
		"token": "eyJhbGciOiJIUzI1..."
		}
		- Thất bại: {

STT	URL	Mô tả chi tiết

		"message": "Invalid email or password",
		"success": false
		}
		
		Phương thức: GET
		Mục đích: Lấy thông tin người dùng đã đăng nhập
		Kết quả trả về:
		Thành công: {
		"user": {
		"id": "123",
		"name": "Nguyễn Văn A",
		"email": "user@example.com",
		"role": "admin"
		},
		"success": true
		}
		Thất bại: {
		"message": "Unauthorized",
		"success": false
		}

6	/api/departments	Phương thức: GET
		Mục đích: Lấy danh sách quyền hạn của một nhóm quyền.
		Tham số đầu vào: mã nhóm quyền trên url.
		Kết quả trả về:
		{
		"departments": [
		{
		"id": "1",
		"name": "Công nghệ thông tin",
		"code": "CNTT",
		"manager": "Nguyễn Văn A"
		}
		],
		"success": true
		}
		
		- Thất bại: {success: false}
		
		Phương thức:POST
		Mục đích: Thêm một khoa/môn học
		Tham số đầu vào:
		{
		"name": "Toán ứng dụng",

STT	URL	Mô tả chi tiết

		"code": "MATH",
		"manager": "Trần Thị B"
		}
		Kết quả trả về:
		{
		"message": "Department added",
		"success": true
		}
		Phương thức: POST
		Mục đích: Thêm khoa/bộ môn mới
		Tham số đầu vào: {
		"name": "Khoa Toán",
		"code": "MATH",
		"manager": "Trần Văn B"
		}
		Kết quả trả về: {
		"message": "Department added",
		"success": true
		}
		
		Phương thức PUT:
		Mục đích: Cập nhật thông tin khoa/bộ môn
		Tham số đầu vào: {
		"id": "1",
		"name": "Công nghệ thông tin & truyền thông"
		}
		Kết quả trả về: {
		"message": "Department updated",
		"success": true
		}
		
		Phương thức DELETE:
		Tham số đầu vào: {
		"id": "1"
		}
		Kết quả: {
		"message": "Department deleted",
		"success": true
		}

7	/api/students	Phương thức: GET Mục đích: Lấy danh sách sinh viên Tham số đầu vào: Không có Kết quả trả về: { "students": [

STT	URL	Mô tả chi tiết

		{
		"id": "123",
		"name": "Nguyễn Văn A",
		"studentId": "SV001",
		"department": "CNTT",
		"status": "Đang học"
		}
		],
		"success": true
		}
		Phương thức: POST
		Mục đích: Thêm sinh viên mới
		Tham số đầu vào: {
		"name": "Lê Thị B",
		"studentId": "SV002",
		"department": "MATH",
		"status": "Đang học"
		}
		Kết quả trả về: {
		"message": "Student added",
		"success": true
		}
		
		Phương thức:PUT
		Mục đích: Cập nhật thông tin sinh viên
		Tham số đầu vào: {
		"id": "123",
		"status": "Bảo lưu"
		}
		Kết quả trả về: {
		"message": "Student updated",
		"success": true
		}
		
		Phương thức: DELETE
		Mục đích: Xóa sinh viên
		Phương thức đầu vào: {
		"id": "123"
		}
		Kết quả trả về: {
		"message": "Student deleted",
		"success": true
		}

8	/api/subjects	{ "subjects": [{ "id": "101",

STT	URL	Mô tả chi tiết
		"name": "Toán rời rạc", "code": "MATH101", "department": "MATH" }], "success": true }

		Phương thức: POST
		Mục đích: Thêm môn học mới.
		Tham số đầu vào: {
		"name": "Lập trình Python",
		"code": "CS102",
		"department": "CNTT"
		}
		Kết quả trả về:{
		"message": "Subject added",
		"success": true
		}

		Phương thức: PUT
		Mục đích: Cập nhật thông tin môn học.
		Tham số đầu vào: {
		"id": "101",
		"name": "Toán cao cấp"
		}
		Kết quả trả về:
		{
		"message": "Subject updated",
		"success": true
		}

		Phương thức: DELETE
		Mục đích: Xóa môn học.
		Tham số đầu vào: {
		"id": "101"
		}
		Kết quả trả về:
		{
		"message": "Subject deleted",
		"success": true
		}

9	/api/grades	Phương thức: GET Mục đích: Lấy danh sách điểm số của học sinh. Tham số đầu vào: không có. Kết quả trả về: { "grades": [{

STT	URL	Mô tả chi tiết
		"student": "SV001", "subject": "MATH101", "grade": 8.5 }], "success": true }

		Phương thức: POST
		Mục đích: Nhập điểm số cho sinh viên.
		Tham số đầu vào: {
		"student": "SV001",
		"subject": "MATH101",
		"grade": 9.0
		}
		Kết quả trả về:
		- Thành công: {
		"message": "Grade added",
		"success": true
		}
		- Thất bại: {
		message: "Added fail",
		success: false,
		Data: []
		}

		Phương thức: PUT
		Mục đích: Cập nhật điểm số của sinh viên.
		Tham số đầu vào: {
		"student": "SV001",
		"subject": "MATH101",
		"grade": 9.5
		}
		Kết quả trả về:{
		"message": "Grade updated",
		"success": true
		}

		Phương thức: DELETE
		Mục đích: Xóa điểm của sinh viên.
		Tham số đầu vào: {
		"student": "SV001",
		"subject": "MATH101"
		}
		Kết quả trả về:{
		"message": "Grade deleted",
		"success": true
		}

STT	URL	Mô tả chi tiết

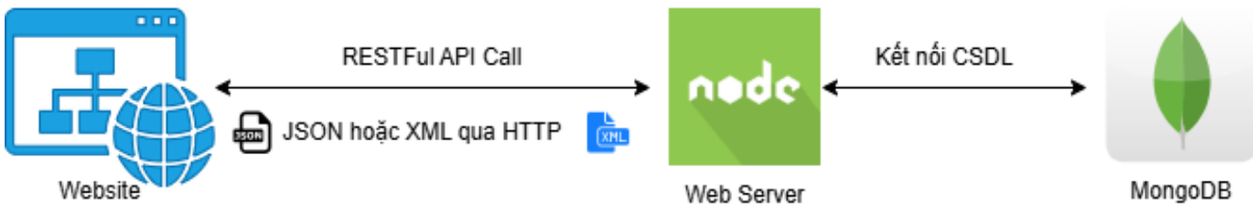
10	/api/products/CourseRegistrations	Phương thức: GET
		Mục đích: Lấy danh sách môn học sinh viên đã đăng ký.
		Tham số đầu vào: {
		"studentId": "SV001"
		}
		Kết quả trả về:
		{
		"registrations": [
		{
		"id": "5001",
		"studentId": "SV001",
		"subjectId": "MATH101",
		"semester": "2024-1"
		},
		{
		"id": "5002",
		"studentId": "SV001",
		"subjectId": "CS102",
		"semester": "2024-1"
		}
		],
		"success": true
		}
		
		Phương thức:POST
		Mục đích:Sinh viên đăng ký môn học
		Tham số đầu vào: {
		"studentId": "SV001",
		"subjectId": "MATH101",
		"semester": "2024-1"
		}
		Kết quả trả về: {
		"message": "Course registered",
		"success": true
		}
		
		Phương thức:PUT
		Mục đích:Cập nhật đăng ký môn học
		Tham số đầu vào: {
		"id": "5001",
		"subjectId": "CS102"
		}
		Kết quả trả về: {
		"message": "Course registration updated",
		"success": true

STT	URL	Mô tả chi tiết

		}
		Phương thức: DELETE
		Mục đích:Hủy đăng ký môn học
		Tham số đầu vào: {
		"id": "5001"
		}
		Kết quả trả về: {
		"message": "Course registration deleted",
		"success": true
		}

4. Cài đặt

4.1. Lựa chọn công nghệ



Hình 4.1 Sơ đồ triển khai hệ thống

***Mô tả sơ đồ:**

Sơ đồ kiến trúc tổng thể của hệ thống trong Hình 4.1 mô tả quy trình thực hiện một ứng dụng Web. Người dùng sẽ sử dụng các chức năng của ứng dụng, các yêu cầu đó sẽ được gửi đến API trên Web Server. Web Server sẽ xử lý các yêu cầu đó và kết nối đến cơ sở dữ liệu, lấy dữ liệu và trả lại về cho Web Server. Sau khi Web Server nhận được dữ liệu sẽ trả về kết quả theo yêu cầu cho ứng dụng Web. Cuối cùng sẽ hiển thị kết quả đó cho người dùng.

***Giới thiệu các công nghệ sử dụng:**

- Công nghệ sử dụng tạo Server: NodeJs tạo các RESTFul API cho bên client gọi để lấy dữ liệu từ cơ sở dữ liệu.
- Cơ sở dữ liệu sử dụng: MongoDB lưu trữ dữ liệu của phần mềm.
- Công nghệ sử dụng xây dựng Web Client: ReactJs thiết kế giao diện bên phía người dùng trên ứng dụng Web.

4.1.1 Database

***Giới thiệu:**

MongoDB là một hệ quản trị cơ sở dữ liệu NoSQL mã nguồn mở phổ biến, được phát triển bởi MongoDB Inc. MongoDB sử dụng cấu trúc dữ liệu dạng document thay vì bảng và dòng như trong hệ cơ sở dữ liệu quan hệ. Dữ liệu được lưu trữ dưới dạng BSON (Binary JSON), giúp linh hoạt trong việc truy vấn và xử lý dữ liệu. MongoDB thường được sử dụng cho các ứng dụng web, phân tán và xử lý dữ liệu lớn nhờ tính linh hoạt và khả năng mở rộng cao.

*Các thành phần chính trong kiến trúc của MongoDB:

- **Query Language (Ngôn ngữ truy vấn):** MongoDB sử dụng ngôn ngữ truy vấn dựa trên JSON, cho phép thao tác dữ liệu linh hoạt hơn so với SQL truyền thống.

- **Storage Engine (Bộ máy lưu trữ):** MongoDB hỗ trợ nhiều bộ máy lưu trữ, bao gồm WiredTiger (bộ máy lưu trữ mặc định) và In-Memory Engine (lưu trữ dữ liệu trong RAM để tăng tốc truy vấn).
- **Replication (Sao chép dữ liệu):** MongoDB hỗ trợ Replication thông qua **Replica Set**, giúp đảm bảo dữ liệu đồng bộ giữa các máy chủ khác nhau và đảm bảo khả năng khôi phục khi hệ thống gặp sự cố.
- **Sharding (Phân mảnh dữ liệu):** MongoDB cho phép phân chia dữ liệu thành các mảnh nhỏ ("shards") để tăng khả năng mở rộng ngang.
- **Aggregation Framework (Khung xử lý dữ liệu):** Cung cấp một cách linh hoạt để xử lý dữ liệu tổng hợp thay thế cho truy vấn SQL truyền thống.
- **Security (Bảo mật):** MongoDB cung cấp các tính năng bảo mật như xác thực người dùng, kiểm soát truy cập và mã hóa dữ liệu.
- **Indexing (Chỉ mục hóa):** MongoDB hỗ trợ nhiều loại chỉ mục như Single Field, Compound Indexes, Geospatial Indexes giúp tăng tốc truy vấn.

*Ưu điểm và nhược điểm của MongoDB:

- **Ưu điểm:**
  - **Linh hoạt:** Do sử dụng cấu trúc document thay vì bảng quan hệ, MongoDB cho phép lưu trữ dữ liệu phi cấu trúc một cách linh hoạt.
  - **Mở rộng ngang:** Hỗ trợ sharding giúp dễ dàng tăng quy mô hệ thống khi dữ liệu tăng.
  - **Hiệu suất cao:** Hỗ trợ truy vấn nhanh chóng nhờ chỉ mục hóa và caching.
  - **Dễ sử dụng:** Các thao tác truy vấn dựa trên JSON rất gần gũi và trực quan.
  - **Bảo mật cao:** Hỗ trợ nhiều tính năng bảo mật và quyền truy cập chi tiết.
  - **Hỗ trợ từ cộng đồng:** Có cộng đồng người dùng lớn, dễ dàng học hỏi và khắc phục sự cố.
- **Nhược điểm:**
  - **Không hỗ trợ giao dịch ACID hoàn chỉnh:** MongoDB chỉ hỗ trợ giao dịch trên phạm vi document thay vì hệ thống.
  - **Tiêu tốn tài nguyên:** Do không hỗ trợ quan hệ, dữ liệu lặp lại nên chiếm dung lượng bộ nhớ lớn hơn so với RDB

4.1.2 Server

***Giới thiệu:**

Node.js là một JavaScript runtime được build dựa trên Chrome's V8 JavaScript engine. Node.js sử dụng mô hình event-driven, non-blocking I/O khiến nó trở nên nhẹ và hiệu quả. NodeJS được phát triển bởi Ryan Dahl vào năm 2009 và có thể chạy trên nhiều hệ điều hành khác nhau: OS X, Microsoft Windows, Linux. [5]

***Lý do sử dụng:**

- NodeJS được viết bằng JavaScript với cộng đồng người dùng lớn mạnh. Nếu bạn cần hỗ trợ gì về NodeJS thì sẽ nhanh chóng có người hỗ trợ bạn.
- Tốc độ xử lý nhanh do có cơ chế xử lý bất đồng độ (non-blocking) nên NodeJS có thể xử lý hàng ngàn kết nối cùng lúc mà không gặp bất cứ khó khăn nào.
- Dễ dàng mở rộng: Nếu bạn có nhu cầu phát triển website thì tính năng dễ dàng mở rộng của NodeJS là một lợi thế cực kỳ quan trọng.

4.1.3 Web Client

*Giới thiệu:

React là một thư viện JavaScript declarative, hiệu quả và linh hoạt cho việc xây dựng giao diện người dùng. React cho phép bạn tạo những giao diện (UI) phức tạp từ những đoạn code nhỏ và độc lập. Những đoạn code này được gọi là "components". [6]

*Các thành phần cơ bản trong React:

- Virtual DOM (DOM ảo): Giúp tăng hiệu năng cho ứng dụng. Thay vì phải vẽ lại giao diện (rerender) mỗi lần thay đổi dữ liệu trong các component. Virtual DOM sẽ được tạo ra khi dữ liệu bị thay đổi, và được tính toán sự thay đổi giữa DOM ảo và DOM thật. Khi dữ liệu này thay đổi, trình duyệt sẽ rerender lại DOM đó.
- JSX (Javascript Syntax Extension): Là một dạng ngôn ngữ cho phép viết mã HTML trong Javascript nhanh hơn, an toàn hơn và dễ dàng hơn.
- Components: Ứng dụng web sử dụng ReactJs được xây dựng dựa trên các component. Và các component này có thể tái sử dụng ở nhiều nơi với các trạng thái và thuộc tính khác nhau. Mỗi component trong ReactJs có một trạng thái riêng, có thể thay đổi và ReactJs sẽ thực hiện cập nhật component dự trên sự thay đổi về trạng thái.
- Props và State:
  - Props: Giúp các component tương tác với nhauu, component nhận dữ liệu sẽ gọi là props và những dữ liệu đó sẽ được component con render ra.
  - State: là một tính năng riêng tư của một component duy nhất. State có thể thay đổi output trong quá trình chạy để đáp ứng với các hành động nhất định.

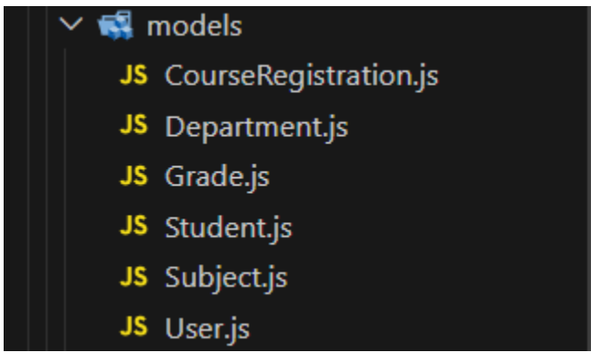
*Lý do sử dụng:

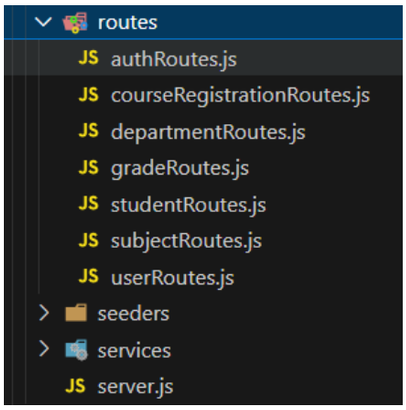
- Giúp viết mã Javascript dễ dàng hơn với JSX: ReactJS sử dụng JSX (Javascript Syntax Extention) hay còn gọi là phần bổ xung cú pháp Javascript. Là một sự hòa trộn giữa Javascript và XML.
- ReactJS cho phép tạo ra các component (thành phần) tương ứng với các giao diện. Các component này có thể tái sử dụng, hoặc kết hợp với các component khác để tạo giao diện hoàn chỉnh. Đây là chìa khóa giải quyết vấn đề khó khăn khi dự án ngày càng mở rộng.
- Thân thiện với SEO: Vì ReactJs có thể chạy tại phía Server, vì vậy dữ liệu trả về cho trình duyệt là văn bản HTML và nó không gây khó khăn cho Search Engine (công cụ tìm kiếm).
- Dễ dàng viết testcase để kiểm thử ứng dụng.
- Được nhiều thư viện hỗ trợ.
- Các chủ đề liên quan đến các vấn đề khó khăn khi phát triển ứng dụng web với ReactJS nhiều, hỗ trợ các nhà phát triển giải quyết vấn đề nhanh chóng

4.2. Cấu trúc mã nguồn

Link mã nguồn: https://github.com/vjintageboy/N06-PTTKPM-NHOM3/tree/caochien

4.2.1 Server



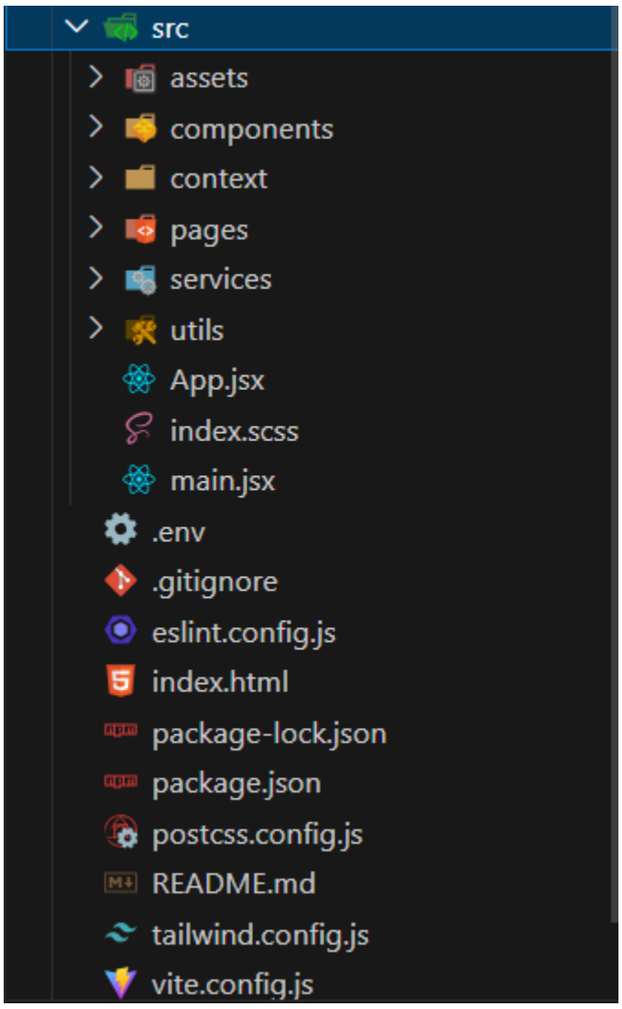


Hình 4.2 Các thư mục mã nguồn Server

STT	Thành phần	Mô tả
1	src	Thư mục chứa mã nguồn chương trình
2	src/models	Tệp chứa mã nguồn API truy suất với cơ sở dữ liệu của chương trình
4	src/models/CourseRegistration	Tệp chứa mã nguồn của API quản lý đăng ký môn học
5	src/models/Department	Tệp chứa mã nguồn của API quản lý khoa
6	src/models/Grade	Tệp chứa mã nguồn của API quản lý điểm
7	src/models/Student	Tệp chứa mã nguồn của API quản lý sinh viên
8	src/models/Subject	Tệp chứa mã nguồn của API quản lý môn học
9	src/models/User	Tệp chứa mã nguồn của API quản lý người dùng
10	src/routes/authRouters	Xử lý xác thực người dùng, đăng nhập, đăng ký và phân quyền.
11	src/routes/courRegistrationRoutes	Xử lý các API liên quan đến đăng ký môn học.
12	src/routes/departmentRoutes	Xử lý các API liên quan đến khoa.
13	src/routes/gradeRoutes	Xử lý các API liên quan đến điểm số của sinh viên.
14	src/routes/subjectRoutes	Xử lý các API liên quan đến môn học.
15	src/routes/userRoutes	Xử lý các API liên quan đến tài khoản người dùng.
16	src/routes/studentRoutes	Xử lý các API liên quan đến quản lý sinh viên.

Bảng 4.1 Chi tiết các thư mục mã nguồn Server

4.2.2 Web Client



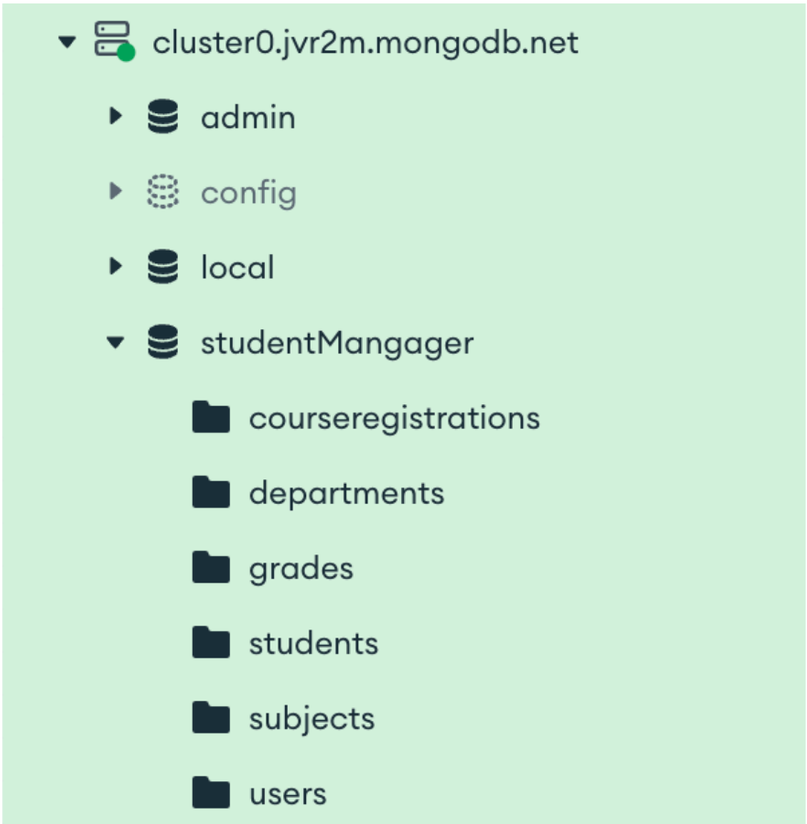
Hình 4.3 Các thư mục mã nguồn của Web Client

STT	Thành phần	Mô tả
1	src	Thư mục chứa mã nguồn chương trình
2	assets	Chứa các tài nguyên tĩnh như hình ảnh, biểu tượng, hoặc tệp CSS chung.
3	src/components	Chứa các thành phần giao diện dùng chung như button, modal, navbar, sidebar, v.v.
4	src/components/departments	Chứa các component liên quan đến quản lý khoa.
5	src/components/footer	Chứa phần footer (chân trang) của ứng dụng.
6	src/components/grades	Chứa các component liên quan đến quản lý điểm số.
7	src/components/header	Chứa thành phần header (tiêu đề) của trang.
8	src/components/ protectedRoutes	Xử lý logic bảo vệ các route yêu cầu quyền truy cập (ví dụ: chỉ quản trị viên mới có thể truy cập).
9	src/components/sidebar	Chứa giao diện thanh điều hướng bên trái của ứng dụng.

10	src/components/student	Chứa các component liên quan đến quản lý sinh viên.
11	src/components/subjects	Chứa các component liên quan đến quản lý môn học.
12	src/components/user	Chứa các component liên quan đến quản lý người dùng.

Bảng 4.2 Chi tiết các thư mục mã nguồn Web Client

4.2.3 Database



Hình 4.4 Các thư mục mã nguồn của Database

STT	Thành phần	Mô tả
1	admin	Database hệ thống mặc định của MongoDB, dùng để quản lý người dùng và quyền truy cập.
2	config	Database dùng để lưu trữ thông tin cấu hình của MongoDB.
3	local	Database nội bộ của MongoDB, chứa dữ liệu tạm thời như thông tin replication.
4	courseregistrations	Lưu thông tin đăng ký môn học của sinh viên.

5	department	Lưu thông tin về các khoa trong trường.
6	Grades	Lưu điểm số của sinh viên.
7	Students	Lưu thông tin của sinh viên.
8	subjects	Lưu thông tin về các môn học.
9	users	Lưu thông tin tài khoản người dùng trong hệ thống (có thể là sinh viên, giáo viên hoặc quản trị viên).

Bảng 4.3 Chi tiết các thư mục mã nguồn Database

5. Kết luận

5.1. Những việc đã làm được và kết quả đạt được

Cơ bản đã hoàn thiện các nhóm chức năng sau:

- Hoàn thiện giao diện quản lý sinh viên với các chức năng cơ bản, hệ thống có thể lưu trữ và hiển thị thông tin sinh viên một cách chính xác, chức năng đăng ký môn học hoạt động ổn định.
- Hỗ trợ phân quyền tài khoản cơ bản cho sinh viên và Quản lý Khoa.
- Quản lí sinh viên: có thể thực hiện các thao tác thêm, sửa, xoá, tìm kiếm thông tin nhân viên. Cung cấp thông tin đầu vào cho chức năng Quản lí tài khoản người dùng để quản lí tài khoản của sinh viên và kết nối với quản lí tài khoản người dùng để quản lí thông tin về sinh viên.
- Quản lí nhóm quyền: có thể thực hiện thêm, sửa, xoá, xem chi tiết, tìm kiếm các nhóm quyền. Cung cấp thông tin đầu vào cho chức năng Quản lí tài khoản người dùng để thực hiện phân quyền cho tài khoản của các khoa .
- Quản lí dịch vụ: Cơ bản hoàn thiện xong giao diện theo thiết kế.
- Thống kê: có thể thực hiện lấy dữ liệu thống kê tổng quan, danh sách học sinh, danh sách khoa, danh sách lớp.
- Đăng kí tài khoản: đã hoàn thiện giao diện
- Đăng nhập: đã hoàn thiện giao diện, cho phép người dùng đăng nhập
- Đăng xuất: cho phép người dùng đăng xuất khỏi hệ thống

5.2. Những điều chưa làm được

- Hệ thống hiện tại mới chỉ dừng lại ở việc quản lý tài khoản cho sinh viên và cho phép họ đăng nhập vào hệ thống với các vai trò khác nhau, chưa hoàn thiện giao diện của sinh viên được phân quyền.
- Chưa tích hợp tính năng xuất báo cáo, bảng điểm dưới dạng file PDF hoặc Excel.
- Chưa có cơ chế thông báo (notification) khi có thay đổi về điểm số hoặc lịch học của sinh viên.
- Chưa có chức năng xem và thanh toán học phí.

5.3. Đề xuất trong tương lai

- Cải thiện bảo mật: Tích hợp xác thực hai yếu tố (2FA) và mã hóa dữ liệu.
- Mở rộng chức năng: Thêm tra cứu lịch học, lịch thi, thêm xuất báo cáo học tập
- Tối ưu hóa giao diện: Đảm bảo hệ thống hoạt động tốt trên cả desktop và thiết bị di động.

Tài liệu tham khảo

[1] K. CNTT, Slide bài giảng môn học Phân tích thiết kế phần mềm, 2024.

[2] Sách giáo trình: Systems Analysis and Design_ An Object-Oriented Approach with UML-Wiley (2020)

[3] https://topdev.vn/blog/restful-api-la-gi/

[4] https://tenten.vn/tin-tuc/mysql-la-gi/

[5] https://viblo.asia/p/gioi-thieu-ve-nodejs-3P0lPyRg5ox

[6] https://monamedia.co/reactjs-la-gi/