PHENIKAA UNIVERSITY

PHENIKAA SCHOOL OF COMPUTING

Microservices Architecture Decomposition & System Communication Design for SGMS

Course: Software Architecture

Course Class: CSE703110-1-2-25(N02)

Group 7:

1. Le Thi Kieu Trang **ID: 23010502**

2. Quach Huu Nam **ID: 23012358**

3. Trieu Tien Quynh **ID: 23010648**

Hanoi, January 7, 2026

Lab 4 – Microservices Architecture Decomposition & System Communication Design

1. Abstract

In previous practical sessions (Lab 1-3), the SGMS system was built following a Monolithic Layered Architecture. Although this architecture is initially easy to develop, scaling, maintenance, and independent deployment become difficult as the system grows.

The goal of Lab 4 is to redesign SGMS according to the Microservices Architecture, where the system is decomposed into independent services that communicate with each other via standard protocols (HTTP/REST).

2. Decomposition by Business Capability

Based on the core business functions of SGMS, the system is decomposed into microservices. Each service is responsible for managing its own logic and separate database, without directly accessing each other's databases.

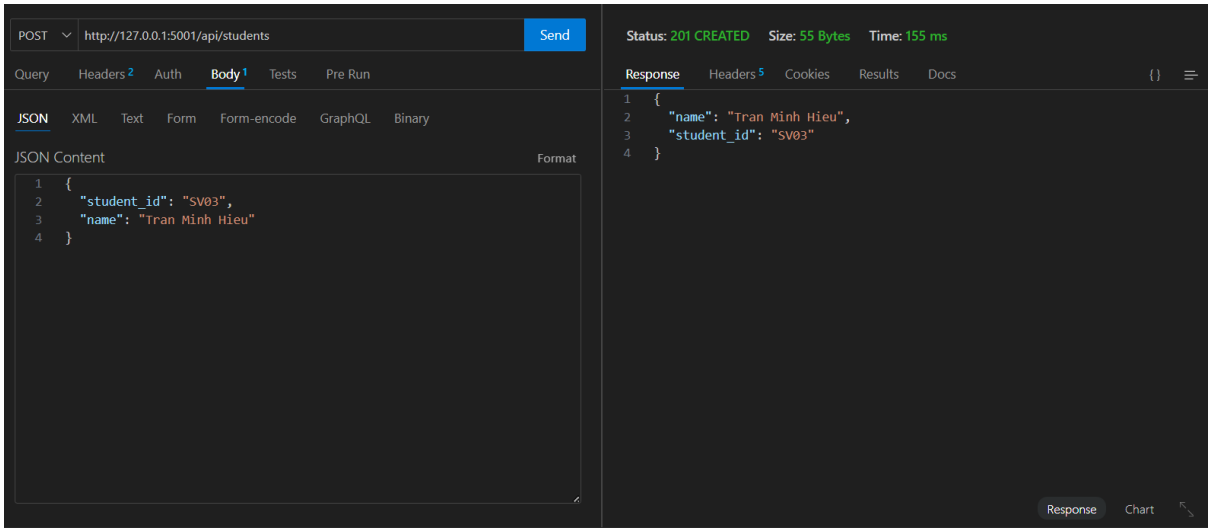
Business Capability	Microservice Name	Owned Data (Entities)	Key Functions
User Management & Authentication	Identity Service	Users, Roles, Tokens	Login, authentication, and authorization (Admin, Faculty, Student).
Student Information Management	Student Service	Students	Store student profiles/records (ID, Name).
Curriculum Management	Course Service	Courses	Manage the course catalog and credits.
Academic & Grade Management	Grade Service	Enrollments	Process course registration, grade entry, and GPA calculation.
Notification Management	Email Service	Email Logs, Templates	Receive requests and send grade notification emails (Asynchronous processing).

3. Defining Service Contracts

- ❖ Create New Student

- Purpose: Create sample student data (Setup Data) to test the Student Service.

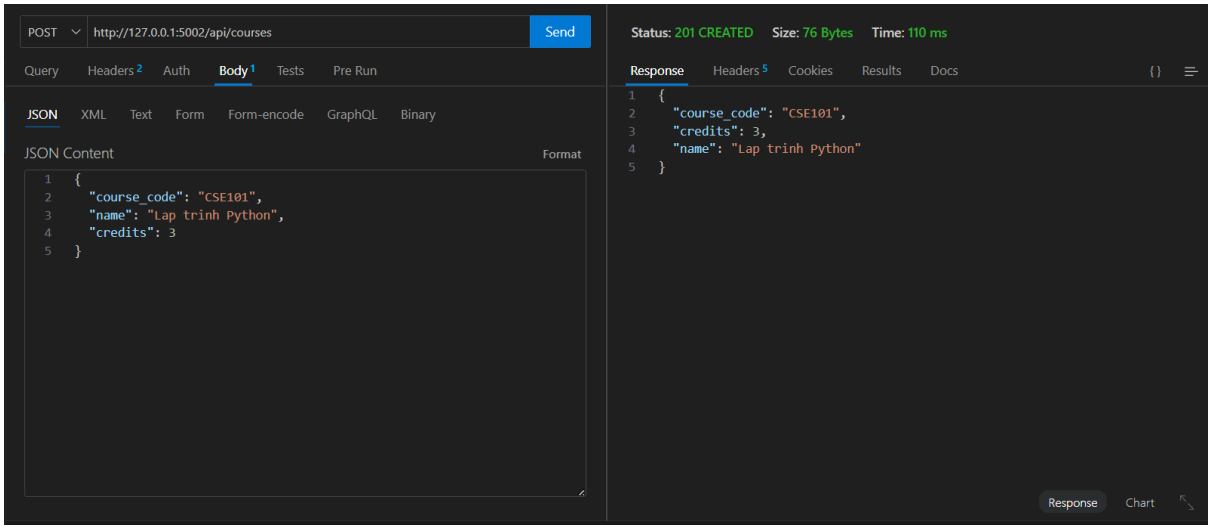
- Significance: Confirm that the Student Service has successfully received and stored student profile information.



❖ Create Course

- Purpose: Initialize courses and define credit values to support future calculations by the Grade Service.

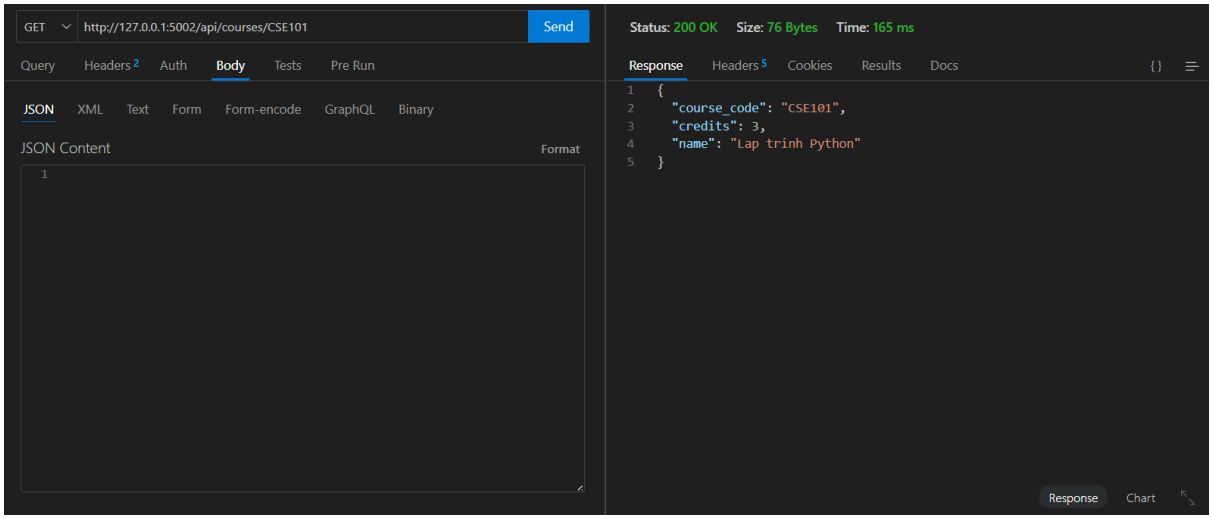
- Significance: Confirm that the Course Service is functioning correctly and has stored course and credit data. This serves as the data foundation for the Grade Service to retrieve credit information via API during the GPA calculation step.



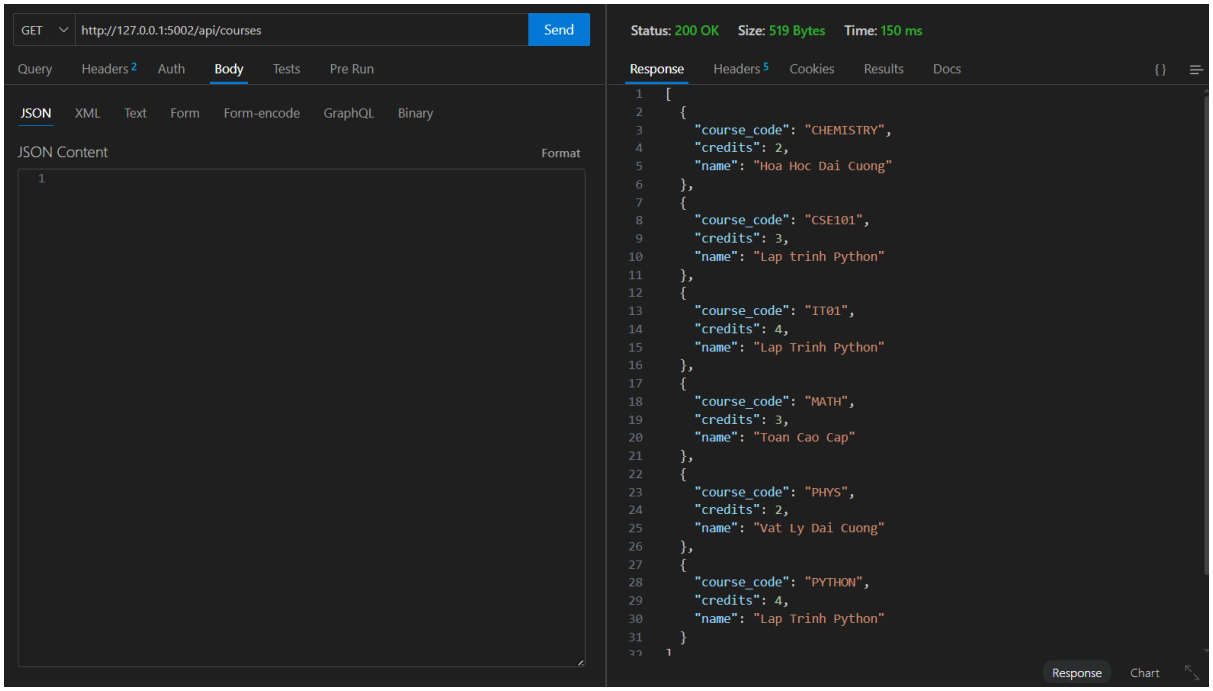
3.1. Course Service (Producer)

- Purpose: To verify that the Course Service correctly provides credit information via the API.

- Significance: Confirms the accuracy of the source data. It ensures the "Producer" role within the architecture: This is a critical verification step to guarantee that the Course Service is ready to supply data to other services.



Testing the API for retrieving course details

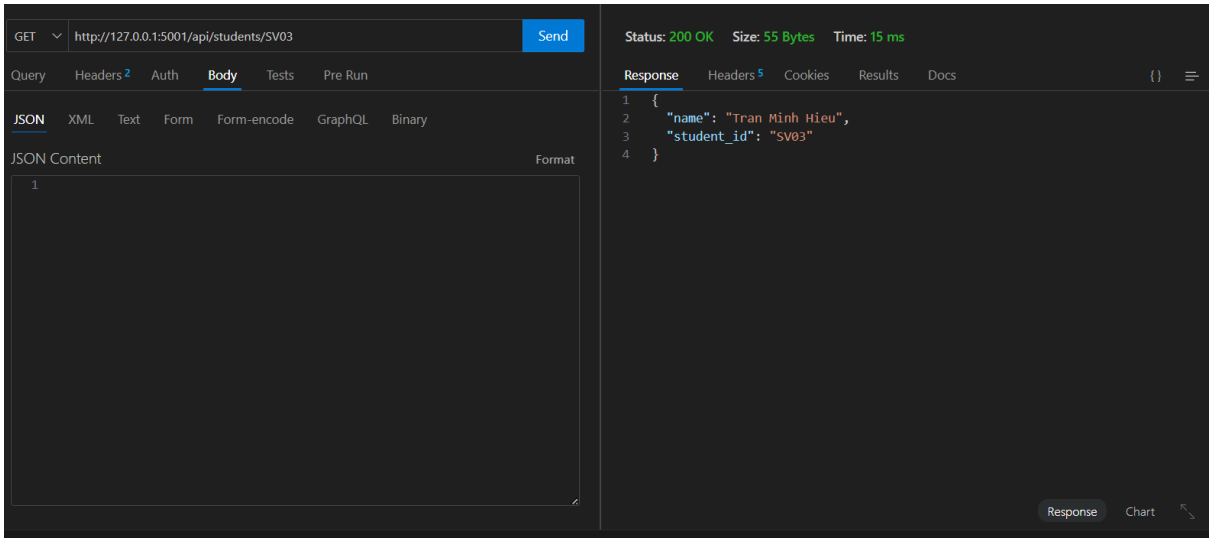


Testing the API for retrieving the list of all courses

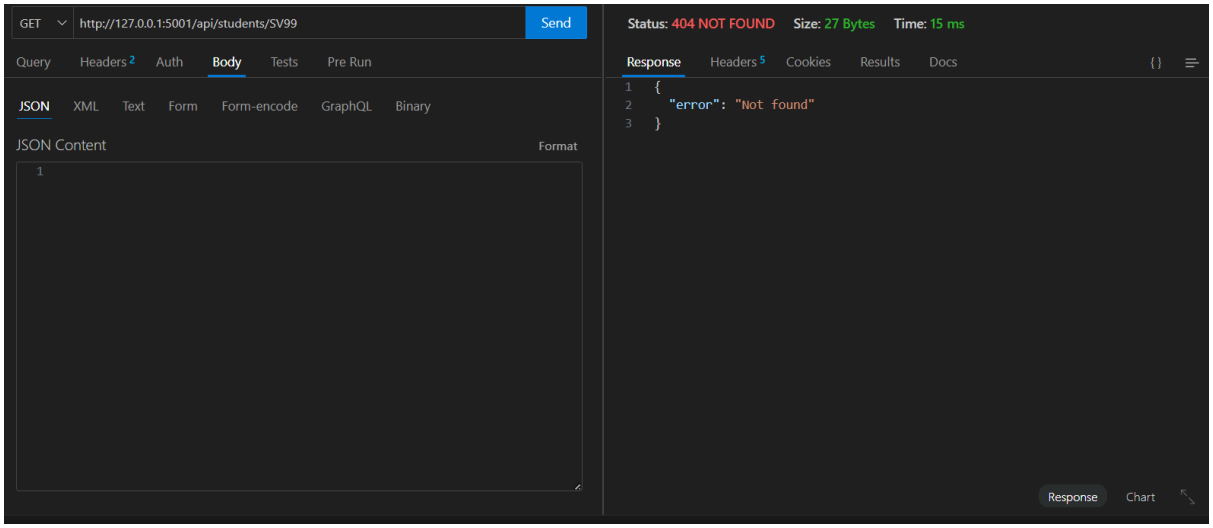
3.2. Student Service (Producer)

- Purpose: To verify that the Student Service correctly provides personal profiles via the API.

- Significance: Validates that the student exists, has been successfully stored, and can be retrieved from the database. It ensures inter-service data integrity within the Microservices architecture; this is a mandatory validation step.



Testing the API to query student information



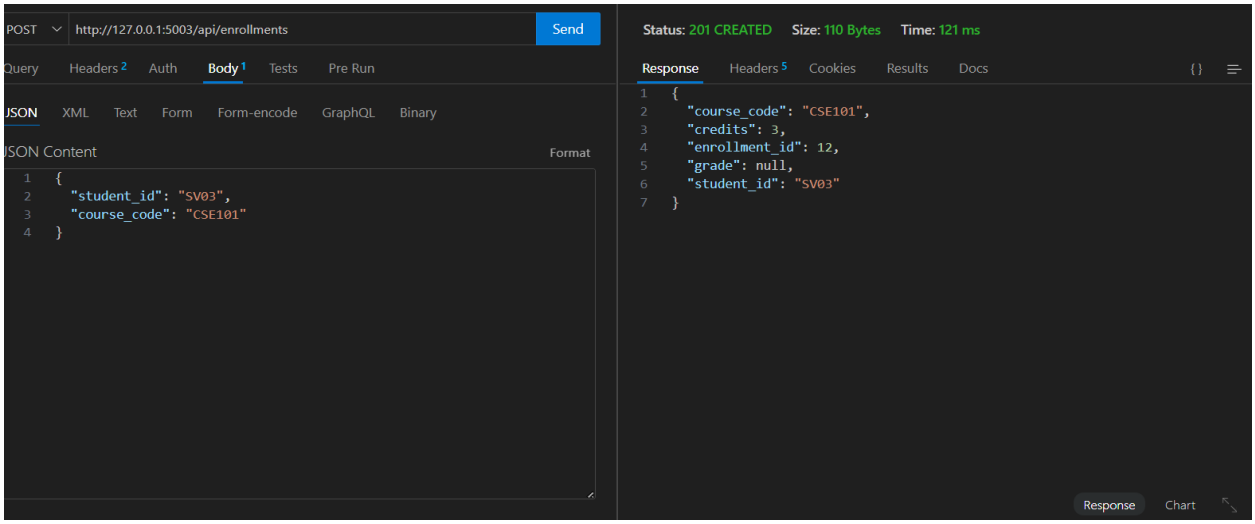
Testing the API query for non-existent student information

3.3. Grade Service (Consumer)

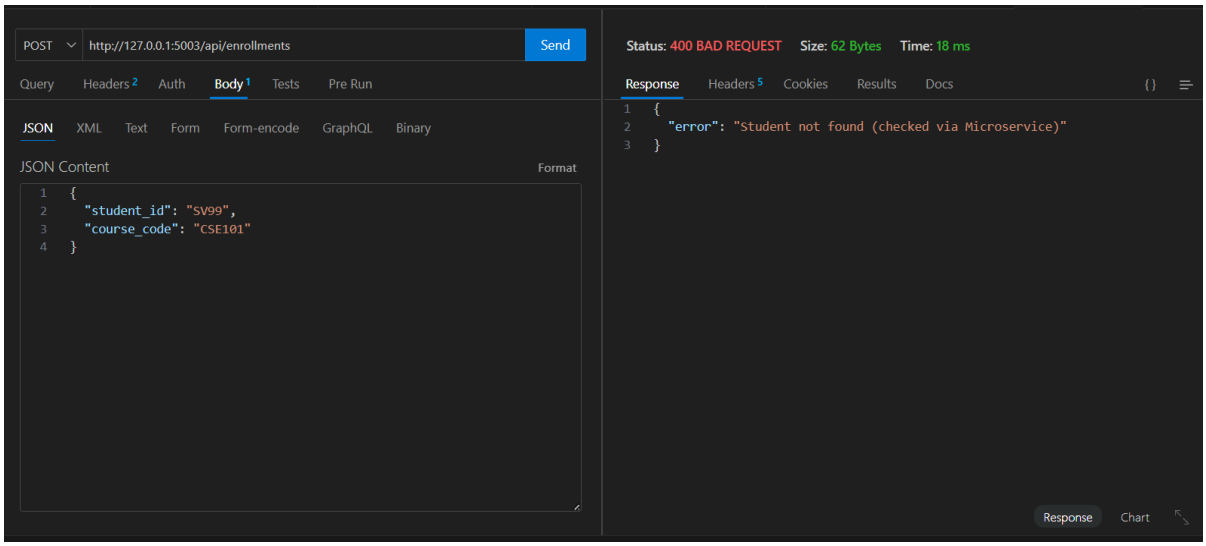
❖ Testing the Registration Function

- Purpose: To test the course enrollment function (Enroll Student). This step triggers the Grade Service to act as a Consumer (calling the Student/Course Services to validate data before saving).

- Significance: Confirms the successful execution of the registration business workflow. The Grade Service has recorded student SV01 enrolling in CHEMISTRY. The generated enrollment ID (11) will be used for the subsequent grade entry step



Testing the Course Registration function: Success Case

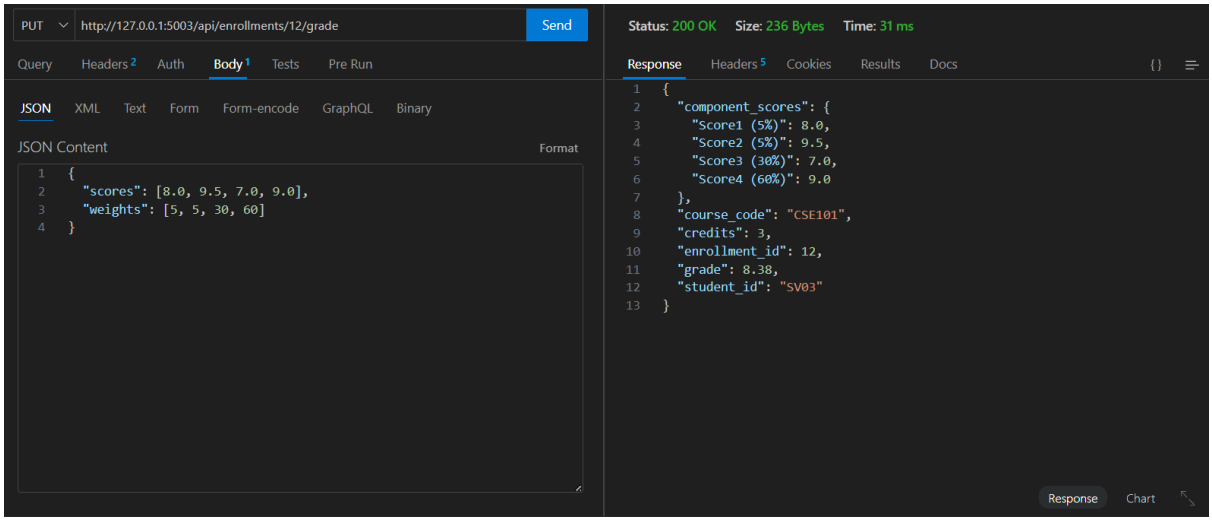


Testing the Course Registration function: Failure Case

❖ Testing Grade Entry, Weighting, and Subject Average Calculation

- Purpose: To test the complex business logic of the Grade Service. The system must accept component scores (Attendance, Assignments, Midterm, Final) and apply a weighting formula (e.g., 5% - 5% - 30% - 60%) to automatically calculate the Final Grade.

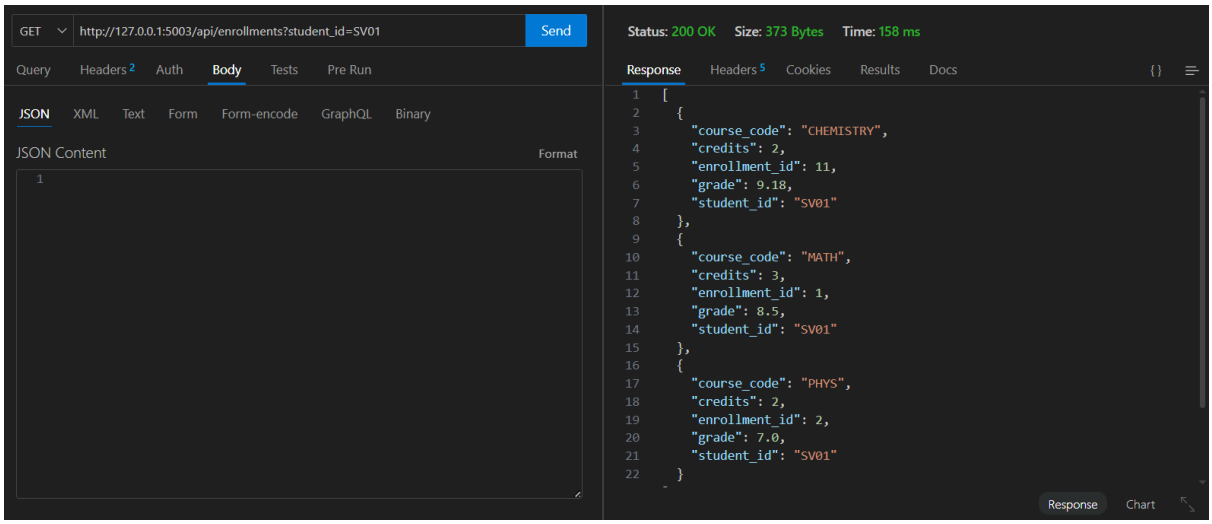
- Significance: Ensures data accuracy: Grades are calculated automatically rather than entered manually, minimizing human error.



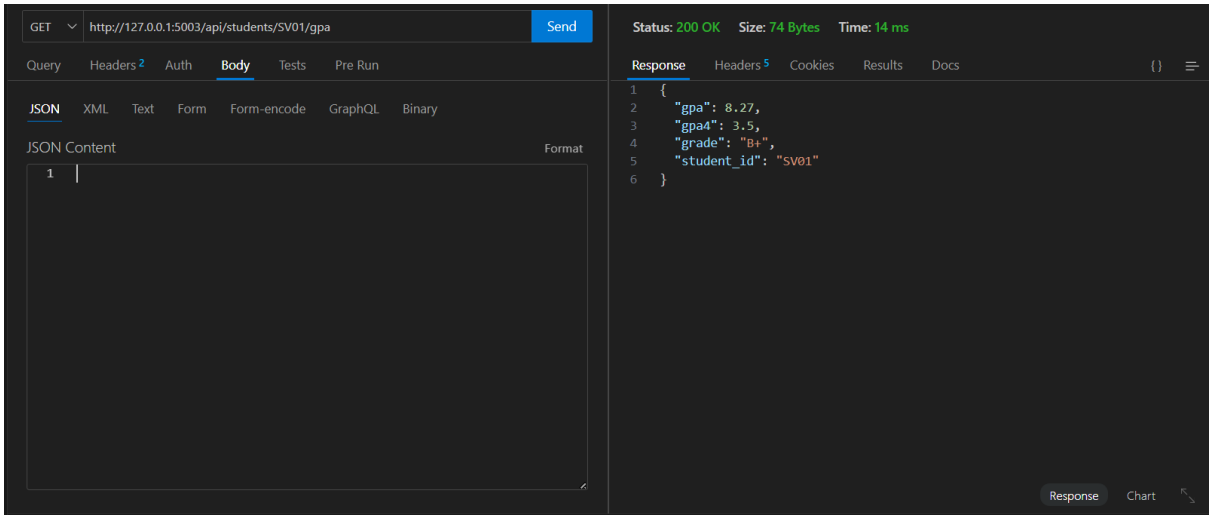
❖ Test Grade Service - Business Logic

- Purpose: To test the system's Data Aggregation capabilities. The Grade Service needs to retrieve the student's entire academic history, join it with data from the Course Service to obtain credit values, and then sort and display the transcript or calculate the cumulative GPA.

- Significance: Demonstrates that the system architecture correctly handles data linking: Even though Grade and Course data reside in different tables/services, the system can successfully aggregate them into a complete report (Transcript). It also ensures a positive User Experience: The returned list is organized logically (by Student → by Course), enabling Faculty/Admins to easily manage and track academic progress.



Grades for courses registered by the student



Calculating GPA for registered and passed courses

4. Communication Strategy

The system employs a combination of Synchronous and Asynchronous communication, depending on specific business requirements.

Interaction	Communication Type	Design Rationale
Client → Identity/ Student/Course	Synchronous (HTTP/REST)	Users require immediate visibility of data (Course lists, Students) or login status to proceed with subsequent operations.
Grade Service → Student Service	Synchronous (HTTP/REST)	Data Integrity: Before saving grades, the system is required to verify that the student actually EXISTS. If the student does not exist, the process must halt immediately and return an error (400 Bad Request).
Grade Service → Course Service	Synchronous (HTTP/REST)	Similarly, accurate credit information must be retrieved from the Course Service to calculate the GPA at the exact moment of registration.
Grade Service → Email Service	Asynchronous (Internal Threading / HTTP)	User Experience (UX): Sending emails can be time-consuming (1-2 seconds or longer). Faculty members should not be forced to wait for the email transmission to complete before receiving the "Grade Entry Successful" notification. Therefore, this task is executed in the background (Fire-and-forget).

5. C4 Model

5.1. C4 Model – Level 1 (System Context Diagram)

The diagram represents the C4 Level 1 System Context of the SGMS platform. It illustrates the interactions between different user roles (Admin, Faculty, and Student), the SGMS

system, and external systems such as the Email Service. The diagram focuses on high-level system boundaries and interactions without exposing internal implementation details.

Mode Description

- Central System: SGMS Platform – A student grade management system designed using Microservices Architecture.

- Các tác nhân (Actors):

+ Admin: Manages student records and the course catalog.

+ Faculty: Enters grades and updates academic results for students.

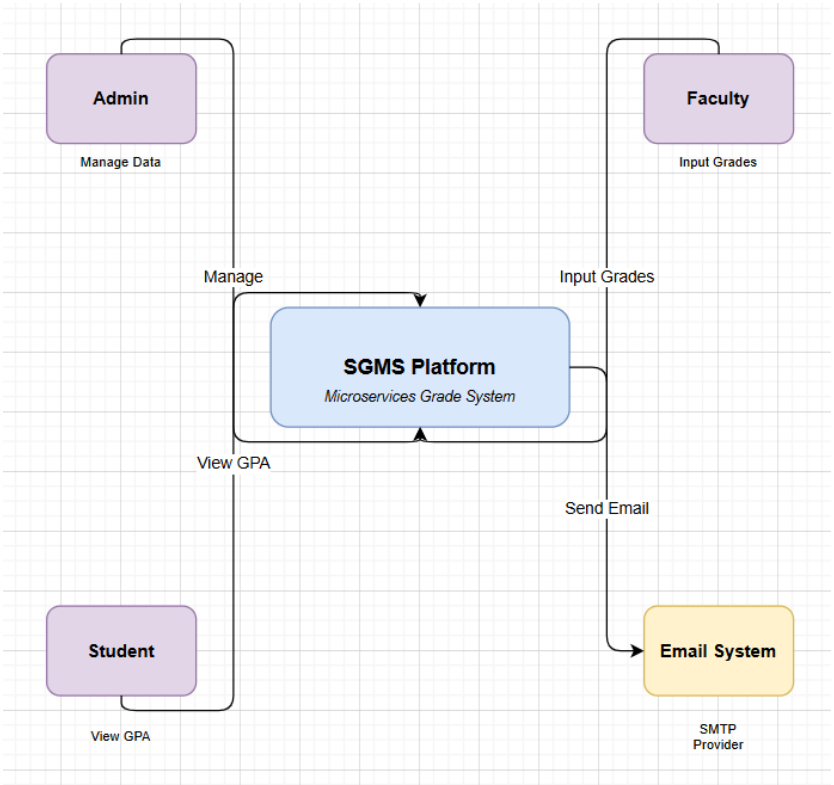
+ Student: Views personal grades and GPA.

- External Systems:

+ Identity Service: Provides user authentication and authorization functions.

+ Email Service: A system that sends email notifications regarding grade-related changes.

This model clearly defines the scope of the SGMS system, the user roles, and the external auxiliary systems that SGMS requires for integration.



5.2 High-Level Communication Diagram

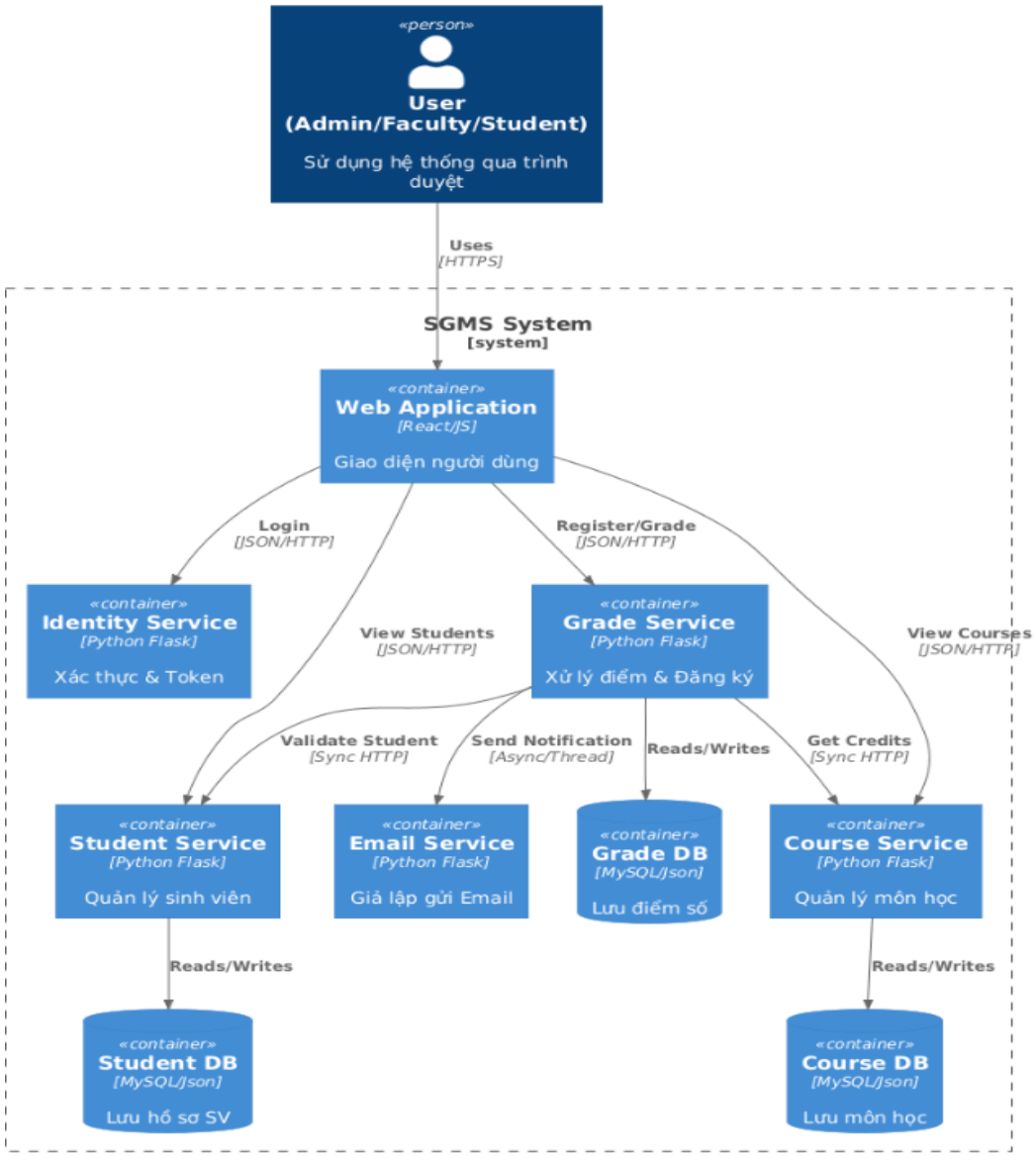
After defining the system context using the C4 Level 1 diagram, the following high-level communication diagram illustrates how the microservices within the SGMS platform interact with each other.

Design Objectives:

+ Define Service Boundaries: Ensure each microservice owns distinct and separate data.

+ Establish Communication Patterns: Clearly distinguish between tasks requiring synchronous and asynchronous processing.

+ Abstract Technical Details: The diagram focuses on Data Flow, omitting internal implementation details such as Controllers, Database Schemas, or infrastructure configurations.



Key Microservices:

+ Student Service: Manages student information.

+ Course Service: Manages the course catalog and credit values.

+ Grade Service: Handles course registration, grade entry, and GPA calculation.

+ Identity Service: Handles user authentication and authorization.

+ Email Service: Sends grade notifications and academic status updates.

This diagram clearly demonstrates the core mindset of Microservices Architecture, ensuring that services operate independently and maintain loose coupling. Instead of relying on direct data access, components interact exclusively via standardized APIs, thereby preserving data isolation and system scalability.

6. Conclusion

Through the implementation of Lab 4, the SGMS project has successfully executed the strategic transition from a Monolithic design mindset to a Microservices architecture.

The clear separation of core services—including Student, Course, and Grade—not only enables the system to achieve Loose Coupling by minimizing cross-dependencies between components, but also unlocks flexible Scalability. Specifically, this allows resources to be concentrated on the Grade Service to handle high traffic loads during peak exam periods.

Furthermore, this new architecture significantly enhances Maintainability, empowering the development team to easily control, debug, and upgrade individual modules independently without disrupting the entire system.