PHENIKAA UNIVERSITY

PHENIKAA SCHOOL OF COMPUTING

COURSE: SOFTWARE ARCHITECTURE

TOPIC:

DEVLOPMENT OF TRAINING PROGRAM MANAGEMENT SOFTWARE

REPORT: LAB 4 – MICROSERVICES DECOMPOSITION AND COMMUNICATION

Code Course : CSE703110-1-1-25

Class : N01

Instructor : THS. Vũ Quang Dũng

Group : 12

Members : 1. Đồng Đại Đạt : KTPM.EL2 – 23010877

2. Nguyễn Cao Hải : KTPM.EL2 – 23013124

3. Ngô Nhật Quang : KTPM.EL1 – 23010134

4. Nguyễn Nam Khánh : KTPM.EL2 – 23010771

Hanoi, November 2025

Table of Contents

1. Overview and Alignment with Lab 4

Lab 4 focuses on the transition from a traditional monolithic architecture to a microservices-oriented design by emphasizing business capability decomposition, service boundaries, communication strategies, and system context modeling. The primary objective of this lab is not implementation, but architectural reasoning and design clarity.

Although UniMIS is currently implemented as a modular monolith, its architecture is intentionally designed to follow microservices principles. Each domain is clearly separated by business capability, enforces strict data ownership, and communicates with other domains through well-defined contracts. As a result, the UniMIS system fully aligns with the conceptual goals of Lab 4, even though it is deployed as a single application.

2. System Overview – UniMIS (Current MVP)

UniMIS is an academic program management system designed to support the management of training programs, courses, students, and organizational workflows within a university context. The system provides CRUD functionality for academic data, supports approval workflows for curriculum changes, records audit logs for traceability, and enables report generation and file storage.

From an implementation perspective, UniMIS consists of a FastAPI backend using asynchronous SQLAlchemy, a PostgreSQL database, and a React-based frontend built with TypeScript and Vite. These components together form a complete web-based information system.

Logically, UniMIS adopts a modular monolith architecture. Each business domain is implemented as an independent module with its own API layer, service layer, schemas, and database models. While all modules run within the same application process, they are designed to behave as if they were independent services, closely mirroring a microservices architecture at the design level.

3. Decomposition by Business Capability

In accordance with Lab 4 requirements, UniMIS is decomposed based on business capabilities rather than technical layers. Each major business function of the academic management domain is mapped to a dedicated module that represents a potential microservice boundary.

The system includes separate modules for user management and authentication, organizational structure management, training program management, course management, student information management, approval workflows, audit logging, file storage, and reporting. Each module owns its corresponding data entities and is responsible for enforcing business rules within its domain.

A strict boundary rule is enforced across all modules. Even though all data resides in a single database, no module is allowed to directly query the tables of another module. When data from another domain is required, it must be obtained through a public service interface or a dedicated query service exposed by the owning module. This design simulates the "no shared database" principle of microservices and ensures loose coupling between domains.

4. External Dependencies

In addition to internal modules, UniMIS interacts with several external systems that lie outside the system boundary. These external dependencies are identified explicitly, as required by Lab 4.

The system integrates with an Email and SMS provider to handle user verification, password recovery, and workflow notifications. It also relies on an object storage service such as MinIO or Amazon S3 to store uploaded files, document templates, and exported reports. Optionally, UniMIS may integrate with an external education portal, such as a Ministry of Education reporting system, to synchronize academic data or submit official reports.

These external systems are treated as independent entities and are accessed only through well-defined interfaces.

5. Service and Module Contracts

Lab 4 emphasizes the importance of service contracts as the primary means of communication between services. In UniMIS, this concept is implemented through both public REST APIs and internal module interfaces.

Public APIs are exposed under a versioned namespace, such as /api/v1, and are consumed by the frontend application and, potentially, external clients. These APIs cover

authentication, program management, course management, workflow operations, audit log retrieval, file handling, report generation, and student information management.

In addition to public APIs, UniMIS defines explicit internal contracts between modules. A representative producer–consumer scenario occurs when the training program module requires course information while managing curriculum items. In this case, the program module acts as the consumer, while the course module acts as the producer. Instead of directly accessing course tables, the program module invokes a query service exposed by the course module to retrieve a summarized view of course data. This interaction strictly follows the same principle described in Lab 4, where one service must never access another service's database directly.

6. Communication Strategy (Synchronous vs. Asynchronous)

UniMIS applies both synchronous and asynchronous communication patterns based on business requirements and user experience considerations, as required by Lab 4.

Synchronous communication is used for operations that require immediate feedback, such as user authentication, CRUD operations on academic data, business rule validation before submitting approval requests, and file downloads initiated by users. These interactions typically use HTTP-based request–response communication.

Asynchronous communication is applied to operations that are time-consuming or do not need to block the user interface. Examples include exporting reports in DOCX or PDF format, sending notification emails when approval workflows change state, and performing background tasks such as file scanning or indexing. These operations are executed using background jobs or message queues, ensuring system responsiveness and scalability.

7. C4 Model – Level 1 (System Context)

To satisfy the C4 Model Level 1 requirement, UniMIS is represented as a single system boundary interacting with users and external systems. The primary actors include system administrators, academic staff, lecturers, and, where applicable, students.

Within this context, UniMIS serves as the central system through which users manage academic programs, courses, workflows, and reports. External systems such as the Email/SMS provider and object storage service interact with UniMIS to support

notifications and file management. Optional integration with external education portals is also represented at this level.

This system context view provides a high-level understanding of how UniMIS fits into its operational environment without exposing internal implementation details.

8. Conclusion

Although Lab 4 is framed around microservices architecture, its core principles are fully applicable to UniMIS through a modular monolith design. By decomposing the system according to business capabilities, enforcing strict data ownership, defining clear service contracts, distinguishing between synchronous and asynchronous communication, and modeling the system context using C4 Level 1, UniMIS satisfies all conceptual requirements of the lab.

This design not only meets the objectives of Lab 4 but also establishes a strong foundation for future evolution toward a fully distributed microservices architecture.