PHENIKAA UNIVERSITY

PHENIKAA SCHOOL OF COMPUTING

COURSE: SOFTWARE ARCHITECTURE

TOPIC:

DEVLOPMENT OF TRAINING PROGRAM MANAGEMENT SOFTWARE

REPORT: LAB 1 – REQUIREMENTS ELICIATION & MODELING

DOCUMENT: SOFTWARE ARCHITECTURE

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

Mục Lục

1. Abstract

a. Problem Description

The selected problem focuses on developing a Training Program Management System that supports universities in organizing and maintaining academic program structures across multiple cohorts and faculties. Managing training programs traditionally involves scattered documents, inconsistent updates, and manual coordination between departments, which often leads to missing information, duplicated work, and difficulties in tracking curriculum changes. The proposed system aims to centralize curriculum data, streamline course management, and provide a clear, structured view of training programs for administrators, faculty members, and students. This digitalization helps ensure accuracy, consistency, and long-term maintainability in academic curriculum management.

b. Description of Lab 01

This lab focuses on the first phase of the software architecture process: requirements elicitation and system modeling. The main objective is to identify key functional and non-functional requirements of the selected system, determine the primary actors, and extract architecturally significant requirements (ASRs) that will influence architectural decision-making in later stages. A UML Use Case Diagram is constructed to model the system boundary, actors, and high-level interactions, providing a visual foundation for understanding the external behavior of the system. The outcomes of this lab establish the groundwork for architectural design activities in subsequent labs.

This foundation establishes the scope and the critical architectural drivers that will be addressed in subsequent labs, particularly the design of the Layered Architecture in next Lab 2.

2. Requirements Elicitation

a. Actors

Actor	Role / Responsibilities	Interaction With System
Admin	Manage the entire system, authorize, and manage users	Login → Account Management → Assign Roles
	Lecturer management, organizational structure	Login → Account Management→ Update lecturer

Training Management Department	Create/edit/delete training programs by course, training program structure, knowledge block, tuition fees,..	Training Program Management → Assign subjects to semesters → Export PDF/Excel
Student / Viewer	Look up training programs, tuition fees, and knowledge blocks,…	CRUD subject → Update lecturer → Assign subject to knowledge block

b. Functional Requirements (FRs)

ID	Functional Requirement	Priority
FR-01	The system must allow Academic to create, update, and delete training programs for each intake year.	High
FR-02	The system must allow to manage course information, including credits, prerequisites, lecturers, and knowledge blocks	High
FR-03	The system must allow the assignment of courses to semesters within a training program.	High
FR-04	The system must allow the management of departments, faculties, and lecturers.	Medium
FR-05	The system must generate downloadable PDF files of tuition fee structures.	Medium
FR-06	The system must allow Admin to manage user accounts and assign roles.	High
FR-07	The system must allow Students and Viewers to view training programs and tuition information by academic year.	Medium
FR-08	The system allows copying training programs to facilitate adding new and editing training programs.	Medium

c. Non-Functional Requirements (NFRs)

ID	Attribute	Description	Impact
NFR-01	**Performance (Latency)**	The system must load a complete training program (including courses, knowledge blocks, and tuition details) within **2.0 seconds** under normal load.	High
NFR-02	**Security (Integrity)**	All sensitive academic data (program structures, tuition fee configurations, user roles) must be securely stored . Access to modification functions must enforce strict role-based control.	Critical
NFR-03	**Reliability (Availability)**	The system must maintain at least 99.0% uptime during academic administrative hours	Critical

NFR-04	**Usability**	The user interface must be intuitive and optimized for desktop, allowing users to easily navigate through training programs and course structures.	Medium

d. Architecturally Significant Requirements (ASRs)

ID	Quality Attribute	Description	Impact
ASR-01	**Modifiability**	Updating or adding a new training program for a new intake must not affect the Course or Lecturer components.	Requires Layered Architecture, separating Program Logic from Course Repository; ensures maintainability.
ASR-02	**Security**	Only Admin may change training program structures, tuition data, or knowledge blocks.	Requires centralized Authorization Service in the Business Logic Layer; enforces role-based access.
ASR-03	**Scalability**	System Support application extension, easy to add new functions	Encourages stateless backend, Docker deployment, and potential horizontal scaling.

3. Use Case Modeling

a. Objective of the Use Case Model

This section presents the Use Case Model for the Training Management System/Curriculum Management System. Its primary goal is to delineate the system boundary, identify the main actors, and describe the high-level functional interactions. This model is constructed based on the functional requirements identified by the team, including curriculum management by enrollment year, course module management, semester course assignment, faculty/major/lecturer management, user authorization, and the display/export of tuition fee information.

The Use Case Model establishes the foundation for understanding the external system behavior and serves as a critical input for architectural decisions in subsequent Labs, particularly in defining the layered architecture and clearly segmenting business responsibilities

b. System Actors

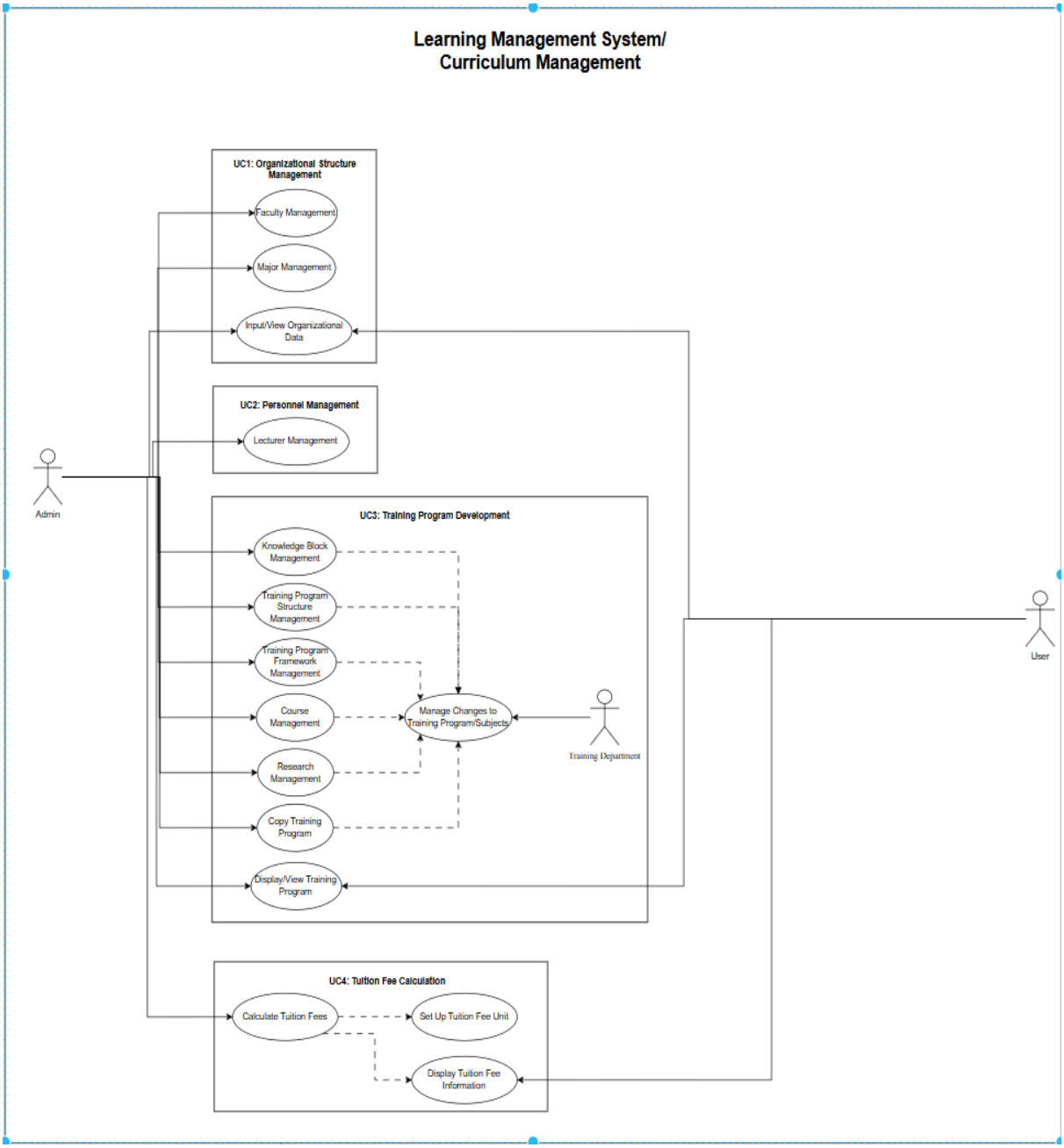
The system interacts with three primary actors:

- Admin: Manages the entire system, encompassing user accounts and roles, core system configuration, and high-level data management.

- Training Management Department: Manages all core academic operational processes, including managing curricula, courses, knowledge blocks, lecturer assignments, and tuition fee policies.

- Student/Viewer: Performs inquiry and viewing tasks, specifically accessing the curriculum , course lists, and tuition fee information relevant to a given academic year.

c. Overview Use Case Diagram



d. Use Case List by Functional Group

The following table summarizes the Use Cases categorized by the system's main functional groups.

UC1: Organizational Structure Management

UC-1.1 Manage Faculty	
Description	Admin performs Add/Edit/Delete (or Deactivate)/Assign/Search/Display operations for Faculty information.
Trigger	Admin opens Organizational Structure -> Faculty.
Preconditions	Admin is authenticated and authorized; system and database are available.
Postconditions	Faculty data is updated according to the selected action; the list is refreshed; audit information is recorded.
Normal Flows	1) Admin accesses Faculty management. 2) System displays list and available actions. 3) Admin performs a management action. 4) System validates and applies changes.
Alternative Flows	A1) Admin uses filters/keyword search to narrow results. A2) Admin cancels an in-progress form.
Exceptions	E1) Duplicate code/name. E2) Deletion conflicts with dependent data -> suggest deactivation. E3) System/DB error.
Priority	High
Frequency of Use	Medium
Business Rules	Faculty code is unique; required fields must be provided; status lifecycle is enforced.
Assumptions	Faculty includes at least code, name, description, status, created/updated metadata.

UC-1.2 Manage Major	
Description	Admin performs Add/Edit/Delete (or Deactivate)/Assign/Search/Display operations for Major information and manages the Major–Faculty linkage.
Trigger	Admin opens Organizational Structure->Major.
Preconditions	Admin is authenticated and authorized; at least one Faculty exists.
Postconditions	Major data and Major–Faculty linkage are updated; list is refreshed; audit is recorded.
Normal Flows	1) Admin opens Major management. 2) System shows list/actions. 3) Admin performs a management action. 4) When

	required, Admin selects a valid Faculty. 5) System validates and saves.
Alternative Flows	A1) Admin reassigns a Major to another Faculty. A2) Filter by Faculty/status.
Exceptions	E1) Duplicate code/name. E2) Selected Faculty invalid/inactive. E3) Deletion blocked by curriculum references.
Priority	High
Frequency of Use	Medium
Business Rules	Each Major must belong to a valid Faculty; codes are unique; only active units can be assigned.
Assumptions	Major includes code, name, faculty_id, status, metadata.

UC-1.3 Query Faculty/Major	
Description	Student/Viewer searches and views the list/details of Faculties and Majors.
Trigger	Student/Viewer opens Faculty/Major Directory.
Preconditions	View access is allowed; system is available.
Postconditions	User sees accurate lists/details of Faculties and Majors.
Normal Flows	1) User accesses the directory. 2) System displays data. 3) User searches/filters. 4) System returns the matching results.
Alternative Flows	A1) View details of a selected Faculty/Major.
Exceptions	E1) System unavailable.
Priority	Medium
Frequency of Use	High
Business Rules	Student/Viewer sees only active/approved records.
Assumptions	Search supports basic keyword and filter conditions.

UC2: Personnel Management

UC-2.1 Query Faculty/Major	
Description	Admin performs Add/Edit/Delete (or Deactivate)/Assign/Search/Display operations for lecturer profiles.
Trigger	Admin opens Personnel->Lecturers.
Preconditions	Admin is authenticated and authorized; organizational units exist if assignment is required.
Postconditions	Lecturer data/assignments are updated; list refreshed; audit recorded.
Normal Flows	1) Admin opens Lecturer management. 2) System shows list/actions. 3) Admin performs a management action. 4) System validates and applies changes.

Alternative Flows	A1) Assign lecturers to Faculty/Major/Department if supported.
Exceptions	E1) Duplicate lecturer code/email. E2) Hard delete blocked by teaching/curriculum links.
Priority	Medium
Frequency of Use	Medium
Business Rules	Lecturer identifiers are unique; assignment only to valid active units.
Assumptions	Lecturer includes code, name, degree/title, contact, unit linkage, status.

UC-2.2 Query Faculty/Major	
Description	Student/Viewer searches and views the list/details of lecturers.
Trigger	Student/Viewer opens Lecturer Directory.
Preconditions	View access is allowed.
Postconditions	Lecturer lists/details are shown correctly.
Normal Flows	1) User opens directory. 2) System displays list. 3) User searches/filters.
Alternative Flows	A1) Filter lecturers by Faculty/Major.
Exceptions	E1) System unavailable.
Priority	Medium
Frequency of Use	Medium
Business Rules	Only active lecturers are visible to Student/Viewer.
Assumptions	Basic search by name/code is supported.

UC3: Curriculum Development

UC-3.1 Curriculum Development	
Description	Admin performs Add/Edit/Delete/Assign/Search/Display operations for course/module records.
Trigger	Admin opens Curriculum -> Courses/Modules.
Preconditions	Admin is authenticated and authorized.
Postconditions	Course/module data is updated; list refreshed; audit recorded.
Normal Flows	1) Open management screen. 2) Perform a management action. 3) System validates credits/prerequisites. 4) Save changes.
Alternative Flows	A1) Search/filter by code, name, status.
Exceptions	E1) Duplicate course code. E2) Invalid credit/prerequisite.

Priority	High
Frequency of Use	High
Business Rules	Course code unique; credit value must be valid; prerequisite must exist when specified.
Assumptions	Course includes code, name, credits, optional prerequisites, status.

UC-3.2 Manage Enrollment Cohorts	
Description	Admin performs Add/Edit/Delete/Assign/Search/Display operations for enrollment cohorts/years.
Trigger	Admin opens Curriculum -> Enrollment Cohorts/Years.
Preconditions	Admin is authenticated and authorized.
Postconditions	Cohort/year data is updated and ready for CTĐT versioning.
Normal Flows	1) Open cohort management. 2) Create/update/deactivate cohort. 3) Save.
Alternative Flows	A1) Create a new cohort based on a previous template (if supported).
Exceptions	E1) Duplicate cohort identifier/year.
Priority	Medium
Frequency of Use	Medium
Business Rules	Cohort identifier must be unique; cohort should map to valid majors/programs.
Assumptions	Cohort includes year, code, status, metadata.

UC-3.3 Manage Knowledge Blocks	
Description	Admin performs Add/Edit/Delete/Assign/Search/Display operations for knowledge blocks within the curriculum.
Trigger	Admin opens Curriculum -> Knowledge Blocks.
Preconditions	Admin is authenticated and authorized.
Postconditions	Knowledge block data is updated; list refreshed; audit recorded.
Normal Flows	1) Open blocks screen. 2) Perform a management action. 3) Save.
Alternative Flows	A1) Configure ordering or optional credit constraints.
Exceptions	E1) Duplicate block code/name.
Priority	High
Frequency of Use	Medium
Business Rules	Block code unique; only active blocks can be assigned into CTĐT.
Assumptions	Block includes code, name, description, optional rules, status.

UC-3.4 Manage CTĐT Structure	
Description	Admin performs Add/Edit/Delete/Assign/Search/Display operations for the CTĐT structure, organized by semester/course groups.
Trigger	Admin opens Curriculum -> CTĐT Structure.
Preconditions	Admin is authenticated; courses/blocks exist.
Postconditions	CTĐT structure by semester/groups is updated and stored.
Normal Flows	1) Select major/cohort/version. 2) Open structure editor. 3) Arrange semesters/groups. 4) Assign courses/blocks. 5) Save.
Alternative Flows	A1) Reorder semesters/groups.
Exceptions	E1) Constraint violations based on framework rules.
Priority	High
Frequency of Use	High during curriculum revisions
Business Rules	Structure must comply with CTĐT framework constraints; only active entities can be assigned.
Assumptions	The system supports semester-based organization.

UC-3.5 Manage CTĐT Framework	
Description	Admin performs Add/Edit/Delete/Assign/Search/Display operations for the CTĐT framework, based on program standards.
Trigger	Admin opens Curriculum ->CTĐT Framework.
Preconditions	Admin is authenticated and authorized.
Postconditions	Framework standards/credit rules are updated and versioned.
Normal Flows	1) Open framework management. 2) Configure standards and constraints. 3) Save.
Alternative Flows	A1) Compare framework versions (if supported).
Exceptions	E1) Conflicts with existing approved structures.
Priority	High
Frequency of Use	Medium
Business Rules	Framework changes should not invalidate historical cohorts; versioning is recommended.
Assumptions	Framework includes total credits, rules, applicability range.

UC-3.6 Copy CTĐT

Description	Admin copies an existing CTĐT to create a new version for a different enrollment cohort or year.
Trigger	Admin selects Copy CTĐT on a source program.
Preconditions	Source CTĐT exists; Admin has copy permission.
Postconditions	A new CTĐT version is created for the target cohort/year.
Normal Flows	1) Select source CTĐT. 2) Choose target cohort/year/major. 3) Confirm scope. 4) System duplicates data. 5) Admin reviews.
Alternative Flows	A1) Copy structure without tuition settings if policy allows.
Exceptions	E1) Target already has a CTĐT version -> system prompts resolution.
Priority	Medium
Frequency of Use	Medium
Business Rules	Copy must preserve data integrity; edits on the new version must not affect the source.
Assumptions	CTĐT supports versioning by cohort/year.

UC-3.7 Display/Query CTĐT	
Description	Admin/Student/Viewer views the CTĐT based on major, cohort, or version.
Trigger	Admin/Student/Viewer opens View CTĐT.
Preconditions	CTĐT data exists; view access is allowed.
Postconditions	The selected CTĐT is displayed correctly.
Normal Flows	1) User selects major/cohort/version. 2) System loads structure, blocks, courses. 3) Display details.
Alternative Flows	A1) Export CTĐT (if enabled).
Exceptions	E1) Missing or invalid selected version.
Priority	Medium
Frequency of Use	High
Business Rules	Student/Viewer sees only approved/active versions.
Assumptions	Filters by major/cohort/version are available.

UC4: Tuition Fee Calculation

UC-4.1 Set Up Tuition Fee Unit	
Description	Admin configures the unit price and calculation basis for tuition fees according to regulations.
Trigger	Admin opens Tuition -> Fee Unit Setup.
Preconditions	Admin is authenticated and authorized.

Postconditions	Tuition unit price and calculation basis are stored; audit recorded.
Normal Flows	1) Open settings. 2) Input unit price/rules. 3) Validate. 4) Save.
Alternative Flows	A1) Configure by cohort/major if supported.
Exceptions	E1) Invalid input ranges or rule conflicts.
Priority	Medium
Frequency of Use	Medium
Business Rules	Unit settings must follow institutional regulations; effective dating/versioning is recommended.
Assumptions	Tuition configuration is linked to cohorts/CTĐT versions.

UC-4.2 Display Tuition Fee Information	
Description	Admin/Student/Viewer views tuition fee information specific to a CTĐT, cohort, or academic year.
Trigger	Admin/Student/Viewer opens View Tuition for a selected CTĐT/cohort/year.
Preconditions	Tuition units/rules are configured; view access is allowed.
Postconditions	Tuition information is displayed accurately.
Normal Flows	1) Select major/cohort/year/CTĐT. 2) System retrieves/calculates tuition. 3) Display breakdown.
Alternative Flows	A1) Export tuition information (if enabled).
Exceptions	E1) Missing tuition configuration for selection.
Priority	Medium
Frequency of Use	Medium
Business Rules	Displayed tuition must align with the active, approved configuration.
Assumptions	Calculation uses credits and configured unit prices.

4. Conclusion & Reflection

The requirements elicitation phase for the Training Program Management System successfully established the project scope through well-defined Functional, Non-Functional, and Architecturally Significant Requirements. The UML Use Case Diagram offers a clear visualization of system interactions across key academic actors and highlights the essential workflows involved in creating and managing training programs.

The identified ASRs—particularly those related to Modifiability, Security, and Scalability—strongly guide the need for a structured architectural approach such as the Layered Architecture. These requirements ensure that the system's design in **Lab 2: Layered Architecture Design (Logical View)** will support maintainable program updates, secure role-based access, and efficient performance during peak academic periods. This foundation provides a sustainable architectural direction for subsequent development phases.