PHENIKAA UNIVERSITY

PHENIKAA SCHOOL OF COMPUTING

Requirements Elicitation and Modeling for the Student Grade Management System (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, December 5, 2025

Lab 1 – Requirements Elicitation & Use Case Modeling

1. Abstract/Summary

The Student Grade Management System (SGMS) is built to support a small department in managing student enrollment, course information, and recording/viewing grades. In Lab 1, requirements were elicited and classified into Functional Requirements, Non-Functional Requirements, and Architecturally Significant Requirements (ASRs).

A UML Use Case Diagram was created to model the core system behaviors with two main actors: the Admin and the Faculty. These deliverables establish the foundational scope for the SGMS and identify key architectural drivers—specifically simplicity, security, and data integrity. These factors will guide the design of the Layered Architecture in the next Lab.

2. Lab Specific Section: I. Requirements Elicitation & Modeling

2.1 Software Requirements Specifications (SRS)

2.1.1 Functional Requirements (FRs)

ID	Description	Priority
FR-01	The system must allow an Admin to create and manage student profiles (ID, name, date of birth, etc.).	High
FR-02	The system must allow an Admin to create and manage course information (course code, course name, credits).	High
FR-03	The system must allow a Faculty member to record and update grades for students in assigned courses.	Critical
FR-04	The system must allow users to view student–course enrollments and associated grades.	High
FR-05	The system must calculate and display a student's GPA based on completed courses.	Medium

2.1.2 Non-Functional Requirements (NFRs)

ID	Attribute	Description	Priority
NFR-01	Performance	Grade retrieval and GPA calculation must execute within 1.0 second for a single student query.	Medium
NFR-02	Security	Only authenticated Admins and Faculty members may modify student data or grades.	Critical
NFR-03	Reliability	Grade data must not be lost in the event of an unexpected system shutdown (ensured via persistent file storage or SQLite).	High
NFR-04	Usability	The user interface must be simple and easy to use for administrative staff without specialized IT backgrounds.	Medium

2.1.3 Architecturally Significant Requirements (ASRs)

ASR ID	Quality Attribute	Requirement Statement	Architectural Rationale
ASR-1	Data Integrity	All student, course, and grade data must ensure correctness through primary/foreign key relationships.	Requires a clear Persistence Layer (SQLite/JSON) and a normalized data model to enforce constraints.
ASR-2	Security	Only authorized roles (Admin/Faculty) may perform data write operations.	Necessitates an authorization mechanism within the Business Logic Layer of the 2-tier architecture.
ASR-3	Modifiability	Changing the storage mechanism (e.g., CSV → SQLite) must not affect the user interface or business logic.	Promotes a Layered Design: UI → Business Logic → Persistence, ensuring separation of concerns.

2.2 Modeling Artifact: UML Use Case Diagram

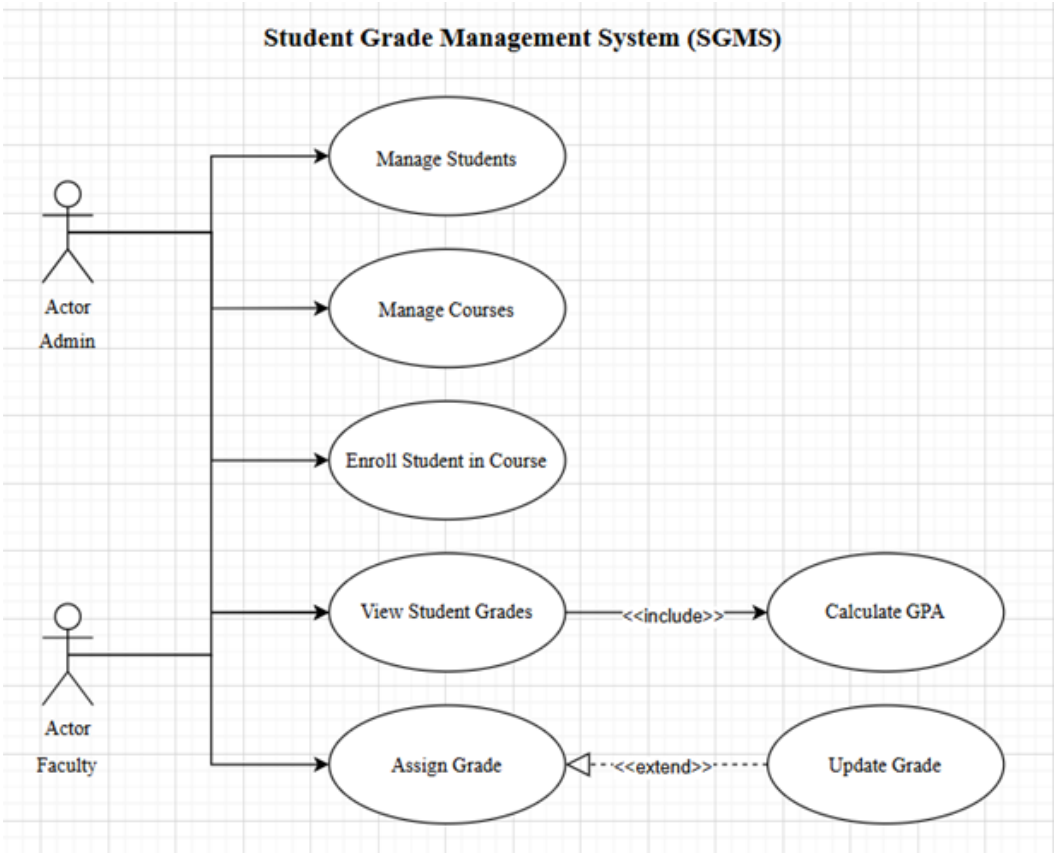

Diagram Explanation:

+ Actors

- Admin: Manages students and courses.
- Faculty: Assigns and updates grades.

+ Core Use Cases

- Manage Students (CRUD)
- Manage Courses (CRUD)
- Assign Grade
- Update Grade
- View Student Grades
- Calculate GPA

- Use Case Relationships

- Include: "Calculate GPA" is included when viewing detailed student academic records.
- Extend: "Update Grade" extends "Assign Grade" for adjustment/correction scenarios.

3. Architectural Design (note: Problem analysis of next lab)

The requirements elicited in Lab 1 have a direct impact on the design challenge of Lab 2: Layered Architecture Design.

3.1 The Problem Statement

The primary objective is to design a 2-Tier Layered Architecture for the Student Grade Management System (SGMS) that guarantees a clear separation of concerns between the user interface and system logic. This architecture must rigorously enforce secure access control to prevent unauthorized data modifications, ensure data consistency across all student and grade records, and support extensibility, allowing for seamless updates to storage mechanisms without impacting the core business logic.

3.2 Impact of ASRs on Layered Architecture

ASR-3 (Modifiability) → Layered Structure

ASR-3 dictates that modifications to the underlying storage mechanism, such as switching from a file-based system to a database, must remain isolated and not impact the User Interface or Business Logic. To achieve this, the architecture adopts a strict Layered Structure where the Persistence Layer exposes a defined data access interface (Repository). The Business Logic Layer is designed to interact solely through this abstraction, ensuring complete independence from the specific storage technology used.

ASR-2 (Security) → Business Logic Layer

ASR-2 requires that all access control and validation checks be centralized within the Business Logic Layer rather than relying solely on the client-side interface. This placement guarantees that critical operations, such as grade modification, cannot be bypassed or manipulated through the UI. Furthermore, it ensures that security policies are enforced consistently across all system access points, regardless of the request's origin.

The next step (Lab 2) will use these requirements to formally define the four architectural layers and model the logical components (Controller, Service, Repository).

4. Conclusion & Reflection

The requirements elicitation phase has clearly defined the scope of the SGMS, modeled system interactions via the UML Use Case Diagram, and identified critical factors to be addressed in the architecture.

The key ASRs—including security, data integrity, and modifiability—demonstrate the suitability of the Layered Architecture pattern in the next Lab. Lab 2 will focus on formally defining the architectural layers (UI, Business Logic, Persistence) and modeling modules such as Controller, Service, and Repository along with their relationships.