UNIVERSITY MANAGEMENT SYSTEM

VISION

1. Introduction

1.1 Purpose

The purpose of this document is to collect, analyze, and define high-level needs and features of the The Management System of University. It focuses on the capabilities needed by the stakeholders and the target users, and why these needs exist. The details of how the The Management System of University fulfills these needs are detailed in the use-case and supplementary specifications.

1.2 Scope

The Management System of University will provide a way for the students to easily learn about their grades, courses and other personal information which university has about them, and for university personals to easily update students' information, set up or cancel courses, notify students about incoming events.

2. Positioning

2.1 Business Opportunity

As universities get crowded with more students and more instructors, the amount of information to be stored is increased gradually. Therefore, managing and accessing that information got difficult with traditional paper records. With systems like EBYS, students and university personal access and manage that information quickly with little effort.

2.2 Problem Statement

The problem of not being able to quickly access and manage information affects students and university personals. The impact of which is delays at every aspect of university management. A successful solution would be a quick and easy to use application for personals and students.

2.3 Product Position Statement

For students and university personals who have difficulty accessing and managing information. The Management System of University is a software application that provides quick and easy access and management of information unlike traditional records which causes spending lots of time in order to get things done.

USE CASES

1. System Boundary

Students, university personals and university's database.

2. Primary Actors

- Student
- Instructor
- Student Affairs Admin

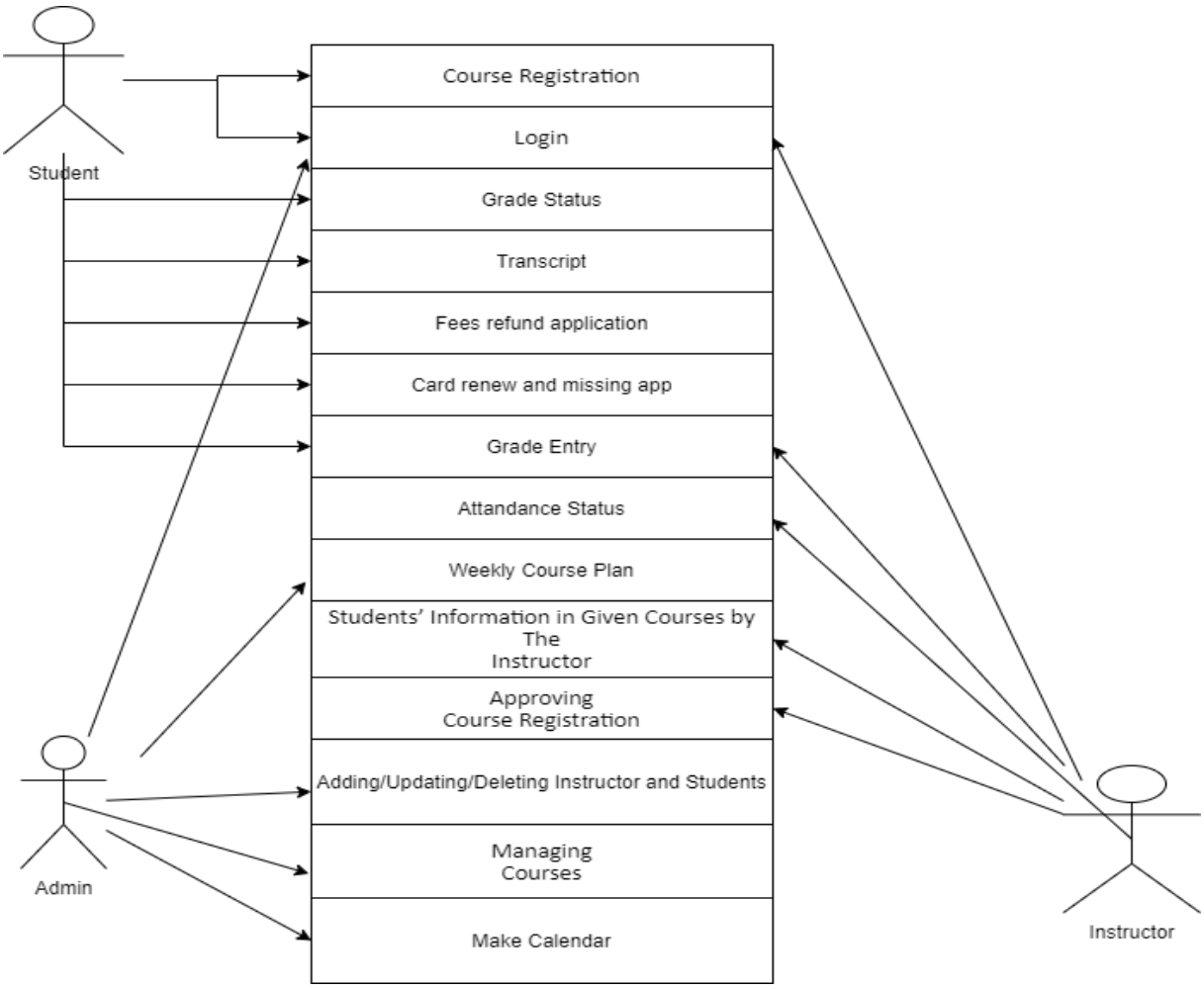
3. User Goals

- Student
  - Course registration
  - Grade, transcript, weekly course schedule and exam schedule view
  - Lost/Renew student cart application

- Instructor
    - Grade entry
    - Weekly course schedule view
    - View announcement
    - Course registration approval
    - Attendance status entry.

- Student Affairs Admin
    - Open course
    - Add/Delete student
    - Make calendar.

3.1 Use-Case Model



4. Use Cases

Use Case UC1: Registration

Scope: Management System

Level: User Goal

Primary Actor : Student

Stakeholders and Interests

- Student : Expects approval of their chosen courses.

- Advisor: Expects the students to submit their selected courses registration

Preconditions:

- The student affairs officer must add informations of the instructors and the students, assign the instructors to the each student, open courses that can taken by students.

- The student must be logged in.

Postconditions:

- The student has enrolled to the courses officially.

Main Scenario:

1. **Student** starts the **registration process.**
2. The system requests the Student ID.
3. Student enter the Student Id.
4. **System**, shows the selectable **courses** via using **Open Courses For Semester**. **Course**

selection proccess carries on according to following **GPA** situations:

- For GNO Above 3.0: The student selects courses from his/her **current term**. Also if the **maximum credit limit** is enough, the student can choose courses that are lower than AA **grade** from the previous **semesters**, or courses from the upper semester if **the overall average** is above 3.00

- For GPA Between 1.8-2.0: The student has to choose courses that he couldn't pass from previous semesters. Then if the credit limit is enough, the student can select courses from his/her current term.

- For GPA Under 1.8: The student can not select courses from his/her current term. He/she has to select course that he/she couldn't pass from the previous semesters.

5. Student chooses the course.

6. System stores the info of the registration.

7. System update **registered student list** and register student to the class.

8. 4. and 5. Steps repeat until the course selection is done.

9. Student sends selected courses to his/her **advisor**.

Extensions (Alternate flow):

*a. At any time, System error:

1.Student restarts System, logs in, and requests recovery of prior state.

2. System reconstructs prior state.

2a.System detects anomalies preventing recovery:

1. System signals error to the Student, records the error, and enters a clean state.

2. Student starts a new return.

1.When the registration's and add-drop week's deadlines are over and the student tries to register:

▪ The student clicks on the login button.

▪ A warning is displayed that he/she can not register on the screen.

2. Add-drop Week:

▪ The student drops the courses that he/she doesn't want to take and then sends the new version of selected courses to the advisor's approval.

▪ The student can select courses that he/she didn't select in the main scenario.

5. Total credit of the selected courses exceeds maximum credit of student.

System warns the student and forces him/her to re-choose courses. The contents of the selected course list are deleted.

Noun Analysis:

We eliminate the synonyms

Instructor is a synonyms for **advisor** .

Current term is a synonyms for **semester** .

The overall average is a synonyms for **GPA**.

We eliminate potential the user interface items

Registration process screen.

Attributes:

GPA, semester, advisor, student id,registered student list.

So we have the classes

1.Student

2. Course

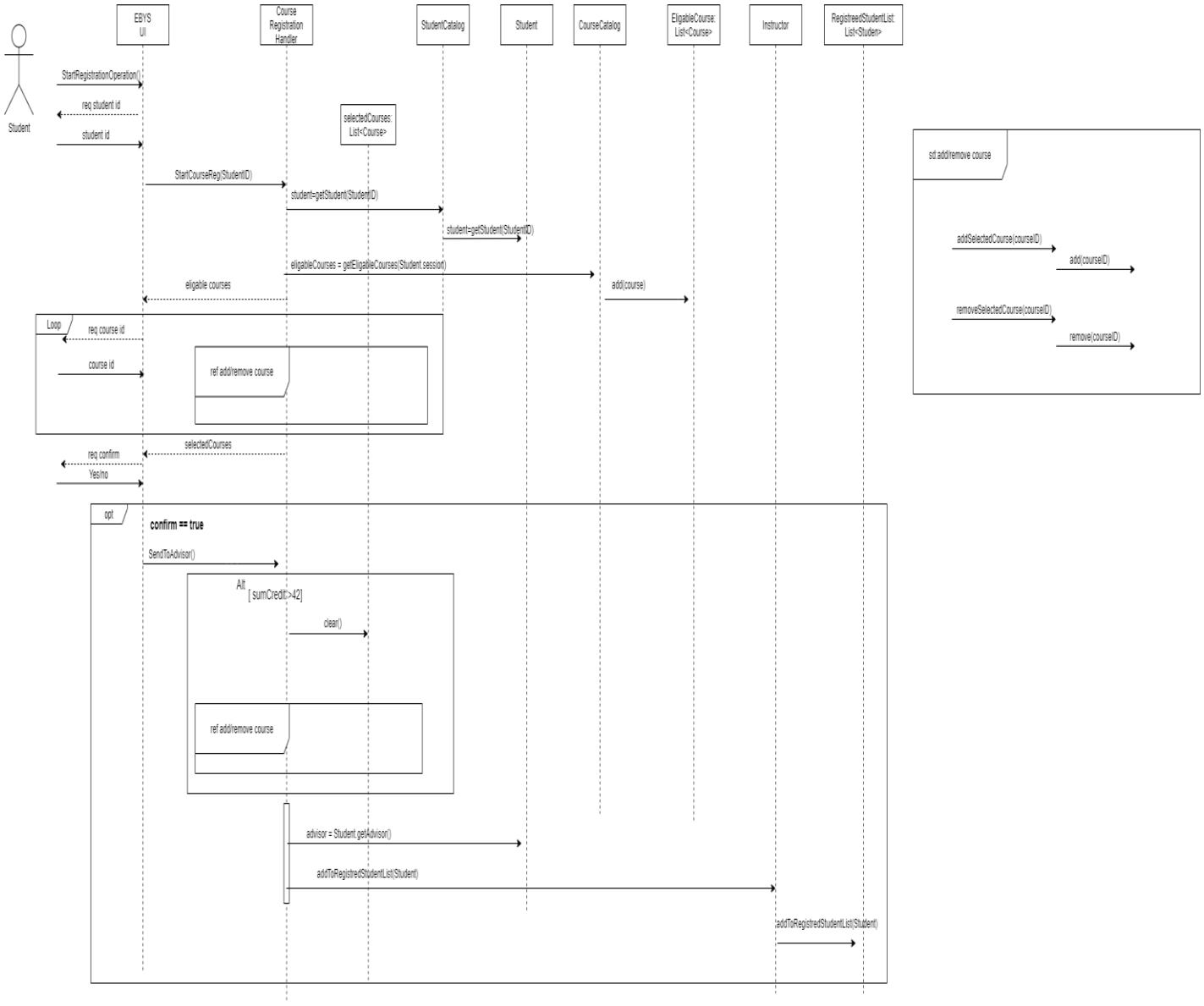
3. Open Courses For Semester

4. Instructor

Relationship between classes:

In this use case, the student makes a request to take courses through the system, the course selection must be approved by the consultant teacher in order to complete the final registration process for the courses he/she chooses. For this reason, student, course, instructor(advisor) classes should be connected.

Registration Sequence Diagram:



GRASP PATTERN FOR REGISTRATION:

Creator:

CourseRegistrationHandler is the creator pattern because CourseRegistrationHandler creates StudentCatalog, CourseCatalog classes .

Controller:

CourseRegistrationHandler is the controller because CourseRegistrationHandler controls the processes while a student registering a class. It seperates business and user interface layer. In the EBYS, CourseRegistrationHandler class represents the whole system and coordinates all of the system operations. It is used as controller for other use cases as well.

Information Expert:

StudentCatalog, CourseCatalog and RegisteredStudentList classes apply Information Expert pattern. These classes perform their operations on all fields related to them with their methods within their own classes.

Low Coupling and High Cohesion:

CourseRegistrationHandler class and Catalog classes(such as CourseCatalog, StudentCatalog) has connections. These connections provide a smaller amount of work. If we were to choose any class other than CourseRegistrationHandler, it would create unnecessary coupling between the CourseRegistrationHandler and other class. Since CourseRegistrationHandler is a class created only for course registration, there is no unnecessary workload. Thus, we provide low coupling. All jobs are distributed in collaboration between catalog classes, not through CourseRegistrationHandler class. By doing this, we were able to delegate operations to related classes and increase cohesion. Thus, we provide high cohesion.

Use Case2: Grade Entry

Scope: Management System

Level: User Goal

Primary Actor: Instructor

Stakeholder and Interests:

- Instructor: Wants to enter students grades quickly and easily.

- Student: Wants his/her grades to be entered quickly so he/she can see them sooner.

Preconditions: Instructor has registered to the EBYS and objection time hasn't elapsed or didn't started yet.

Postconditions: Objection time starts and students can view their grades.

Main Success Scenario (Basic Flow):

1. **Faculty member** logs in to **system** and goes to the **grade entry panel**.

2. System request Member Id.

3. Faculty member enters the member id.

4. System shows the **instructor's courses** and request **course id.**

3. Instructor choose the course with course id which he/she wants to enter the **students' grades**.

4. System shows the **student list of the selected course**.

5. Instructor **submit** the grades and starts the objection time if it is not already started.

6. The system asks if there will be a grade entry for another course. If the answer is positive, the system goes to step 4, if the answer is negative, the system exits.

Extensions (Alternative Flows):

*a. At any time, System error:

1. Instructor restarts System, logs in, and requests recovery of prior state.

2. Instructor reconstructs prior state.

2a. Instructor detects anomalies preventing recovery:

1. System signals error to the Instructor, records the error, and enters a clean state.

2. Instructor starts a new return.

5. Instructor submit a grade that isn't in the spesific interval.

System warns instructor and forces him/her to change the grade.

Noun Analysis:

We eliminate the synonyms

Faculty Member is a synonyms for **instructor**.

We eliminate potential the user interface items

Grade entry panel, submit.

Attributes:

Student list of the course,grades, course id, instructor courses.

So we have the classes

1.Course

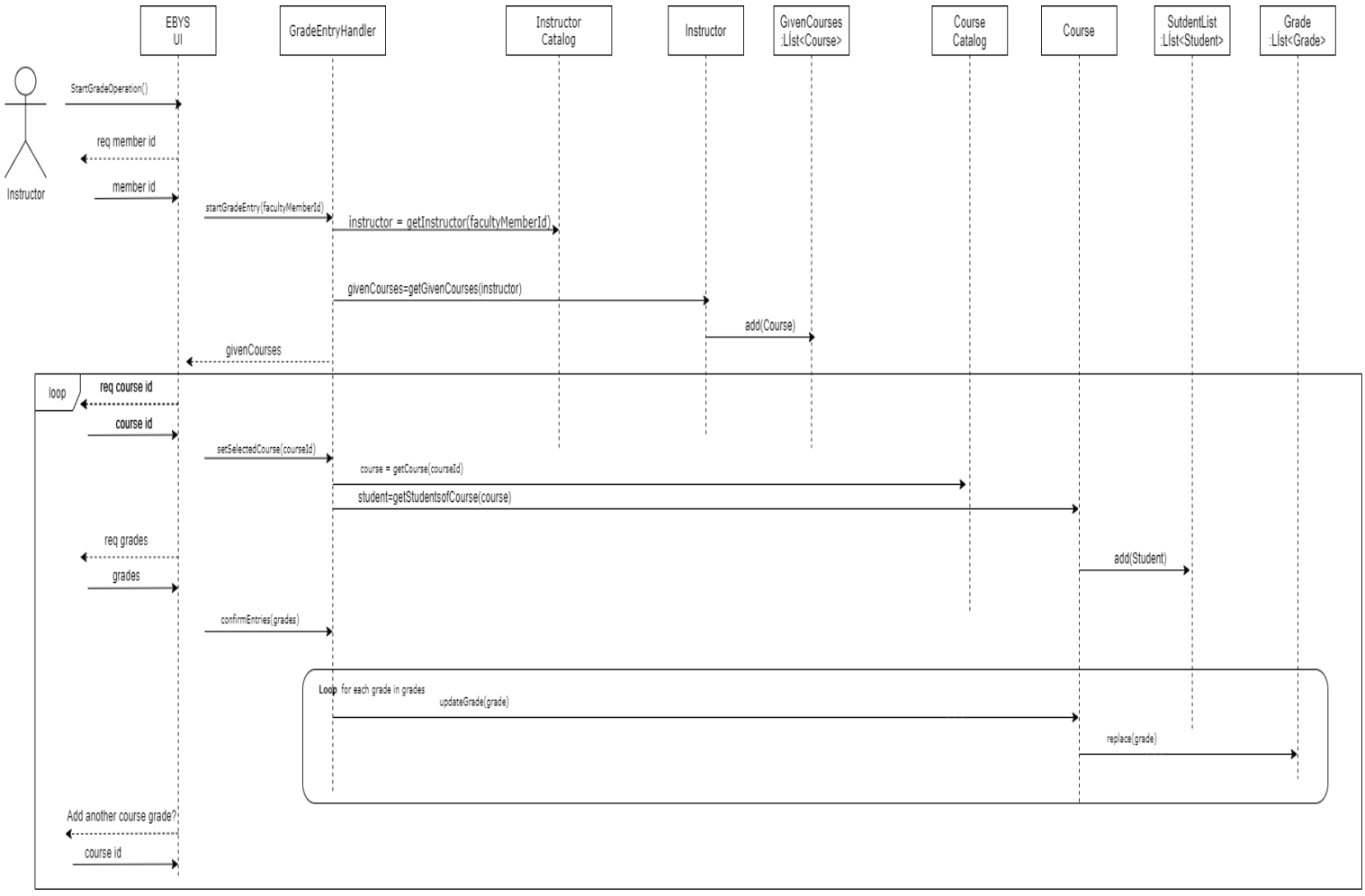
2.System

3.Instuctor

Relationship between classes:

In this case, the teacher shares grade information with students through the system. For this, students who take the course must enter their grade information into the system. To do this, the instructor class must be directly connected to the course class.

Grade Entry Sequence Diagram:



GRASP PATTERN FOR GRADE ENTRY

Creator:

Course is the creator pattern because Course creates Grade lists for the notes the instructor entered.

Controller:

GradeEntryHandler is the controller because GradeEntryHandler controls the processes while a instructor entering grades into the system. It seperates business and user interface layer. In the EBYS, GradeEntryHandler class represents the whole system and coordinates all of the system operations. It is used as controller for other use cases as well.

Information Expert:

StudentList, InstructorCatalog and CourseCatalog classes apply Information Expert

pattern. These classes perform their operations on all fields related to them with

their methods within their own classes.

Low Coupling and High Cohesion:

GradeEntryHandler class and Catalog classes(such as CourseCatalog, StudentCatalog) has connections. These connections provide a smaller amount of work. If we were to choose any class other than GradeEntryHandler, it would create unnecessary coupling between the GradeEntryHandler and other class. Since GradeEntryHandler is a class created only for entering grades by intructors, there is no unnecessary workload. Thus, we provide low coupling. All jobs are distributed in collaboration between catalog classes, not through GradeEntryHandler class. By doing this, we were able to delegate operations to related classes and increase cohesion. Thus, we provide high cohesion.

Use Case UC3: OPEN CLASSES FOR A TERM

Scope: Management System

Level: User Goal

Primary Actor: Administrator

Stakeholders and Interests:

University: Provides courses and wants to enter course informations for courses.

Administrator: Wants to open classes for a term. Wants to make changes on courses status.

Precondition: Admin has to log-in to the system.

Success Guarantee: Course informations and Instructor of cours(es) are successfully entered. Class status is set to active.

Main Success Scenario:

1. The Admin requests to **EBYS(University System)** open course(es) for a **Departmant** in **specific term.**

2. The system returns the courses from **Education Plan**.

3. Admin enters the **course ID** for opening course into the system.

4. The System asks the **location**, **instructor** and **schedule** informations from admin.

5. Admin enters the informations which are requested from system.

6. The system checks the information which admin entered into system. If entered informations are avaiable, system displays the information for the class. Update the **open courses for term.** Then asks for another process. If not, displays error.

7. If Admin responds affirmative, the system goes to step 3, if not admin ends process.

Extensions:

*a. At any time, System error:

1. Admin restarts System, logs in, and requests recovery of prior state.

2. Admin reconstructs prior state.

2a. Admin detects anomalies preventing recovery:

1. System signals error to the Admin, records the error, and enters a clean state.

2. Admin starts a new return.

3a. Invalid course ID.

1. The system displays errors and rejects entry.

2. Admin responds to the error.
3. System asks for the information again.

Noun Analysis For Entering Schedule

Attributes of classes:

Course id is attribute of course class

Associations

Course Catalog has Courses.

Instructor Catalog has Instructors.

Location Catalog has multiple Locations.

University has multiple departments.

Department has an education plan for a student.

Education Plan has all infromations about departmant course except Date,Instructor and location informations.

Classes

University

Department

Education Plan

Open course for term

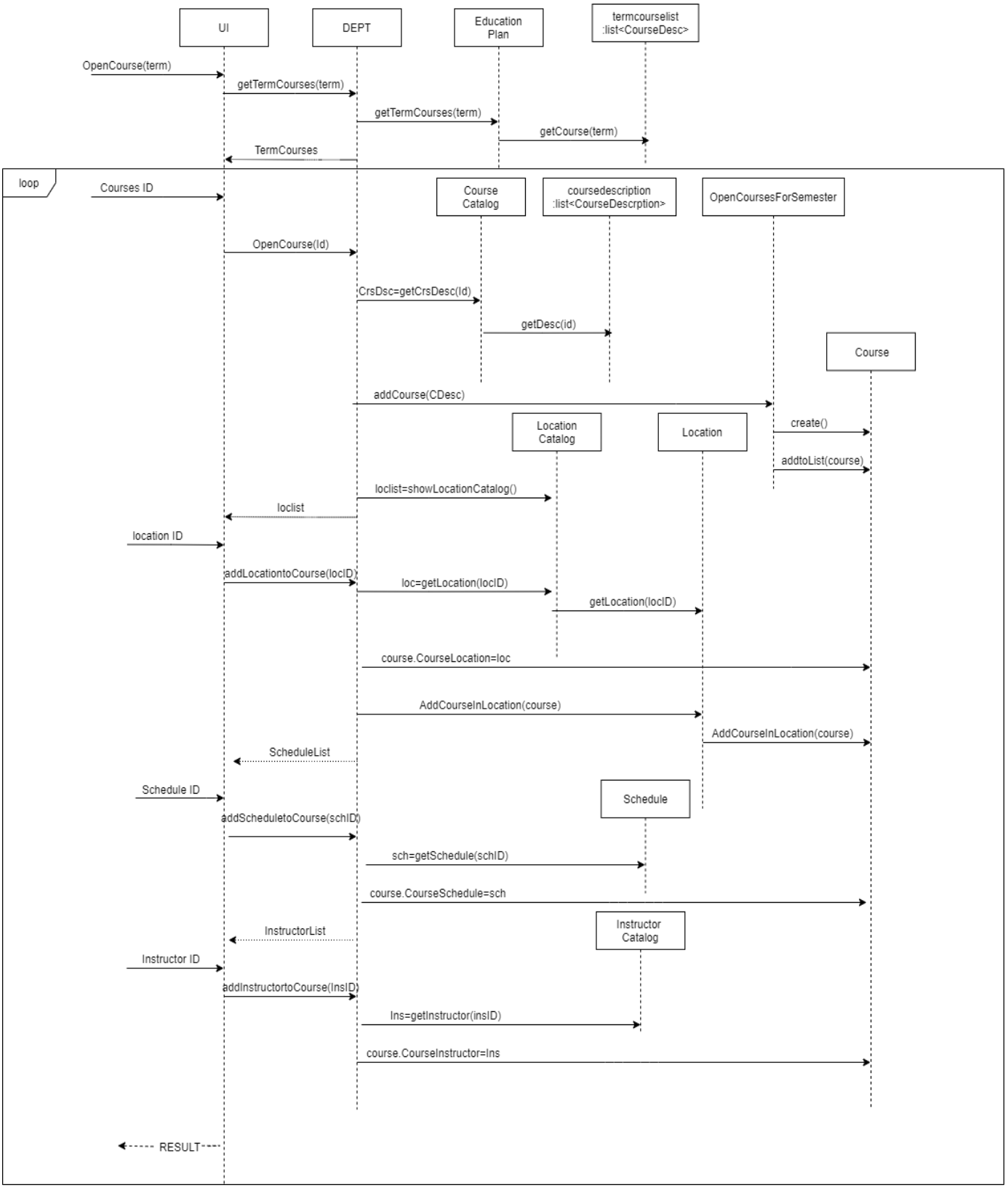
Location

Instructor

Schedule

Course

OPEN CLASSES FOR A TERM SEQUENCE DIAGRAM:



GRASP PATTERNS

Creator:

OpenCourseForSemester class applies Creator by creating a Course object. Since OpenCourseForSemester has a Course to add to open course list for each semester with course descriptions, it is appropriate for OpenCourseForSemester to be responsible for creation of a Course object.

Controller:

Department class applies Controller pattern. It separates business and user interface layer. Department class represents the whole system and coordinates all the system operations. It is used as controller for other use cases as well.

Information Expert:

InstructorCatalog, CourseCatalog and LocationCatalog classes apply Information Expert pattern. These classes perform their operations on all fields related to them with their methods within their own classes.

Low Coupling and High Cohesion:

Department class performs operations for whole system, add method is added to the Department. This method delegates operations to other classes which are OpenCoursesForTerm class. Hence, cohesion is increased. Department class and Catalog classes(such as CourseCatalog, InstructorCatalog) has connections. These connections provide a smaller amount of work. If we were to choose any class other than Department, it would create unnecessary coupling between the Department and other class. Since Department is a class created showTermCourses, openCourse methods etc. for processes. Thus, we provide low coupling.

DOMAIN MODEL