Marmara University Engineering Faculty
Department of Computer Engineering

CSE3063 – Object Oriented Software Design

Course Registeration System
Requirement Analysis Document (RAD)
Iteration #2

Instructor: Murat Can Ganiz

December 20, 2023

STUDENTS – GROUP 16

BURAK KARAYAĞLI - 150121824
BARIŞ GİRAY AKMAN - 150121822
EFE ÖZGEN - 150121077
HAKKI KOKUR - 150120033
MUSA ÖZKAN - 150121058
TAMER ÜNAL - 150119795
ATİLLA GEL - 150119564

1 Introduction

The name of the project is Course Registration System. Main purpose in the project is providing users to get assigned to the necessary lectures through their advisors and simplifying the registration process of the course registration.

Users will enter the system through the user name and password given during the registration to the system. To enable the course at the system, there must be at least one lecturer to lecture. Otherwise, course won't be enabled for that semester. Each student who have taken any of the courses will be graded through the system at the end of the semester. And these grades will be processed through transcript.

2 Scope

The project's scope is to develop a robust university course registration system, accommodating various user roles: students, advisors, lecturers, and staff members.

The system will offer a range of essential functionalities, including user authentication, course enrollment, grade management, and administrative tools. The core entities within the system, such as Person, Student, Advisor, Lecturer, and Course, will be intricately connected to facilitate operations like course registration, student management, course approval, and transcript viewing. Data management will rely on JSON files and Command-line menus will serve as the user interface, enabling users to interact seamlessly with the system.

3 Constraints of the System

Certain functionalities are explicitly out of scope for this project. For instance, the project does not involve the development of a graphical user interface (GUI) for the system, meaning users will interact solely through command-line interfaces. Additionally, the project uses JSON files for data storage, but it does not implement a full-fledged relational database system. Consequently, advanced database management features, such as complex queries and optimizations, are not part of the scope.

While the project defines various roles, like students, advisors, and lecturers, it does not address the creation or management of user accounts and authentication, assuming that these aspects are managed separately.

Finally, the system does not incorporate real-time or web-based features, meaning that all interactions occur within the confines of the command-line interface.

4 Goals of This Iteration

In our initial iteration, we constructed a basic course selection and validation system. In this iteration, our primary goal was to enhance the added features to be more user-friendly and realistic. To achieve this, we first refactored various methods within the SystemController.java file to leverage object-oriented principles by distributing them to relevant classes. We diversified the existing menu and controller classes to strengthen the MVC architecture and organized them according to their respective responsibilities.

SystemController became AdvisorController, LecturerController, and StudentController, while the menu classes were expanded to Menu, AdvisorMenu, LecturerMenu, and StudentMenu. Additionally, we modularized the code, making it more adaptable for future feature additions. In the Student class, we converted some of the operations that were partially implemented in the controller into methods. These methods now handle tasks such as prerequisite checks, fetching courses based on the student's semester and transcript, and verifying credit and quota limits. Similar enhancements were made in the Advisor and Lecturer classes.

New/Updated Features:
- Tracking the student's status (waiting, approved, finished, etc.).
- Returning suitable courses for the student based on their semester and transcript.
- Checking course semester compatibility with the student's semester.
- Verifying course quotas.
- Ensuring prerequisite course completion.
- Validating time intervals.
- Enabling individual course selection and withdrawal.
- Allowing advisors to approve or reject individual course requests.
- Permitting lecturers to add mandatory, technical, and non-technical courses.
- Providing lecturers with a list of students registered for their courses.
- Checking the number of credits to select technical electives, ensuring that students meet the credit requirements before enrolling in these specialized courses.
- Diversifying the transcripts and calculating both cumulative and semester-specific GPAs, allowing for a more comprehensive evaluation of the student's academic performance.
- Scaling up the system to accommodate a larger user base, increasing the number of students from 10 to 48, advisors from 3 to 13, and lecturers from 3 to 13. This expansion ensures that our system can efficiently serve a growing educational community.

With these improvements, our code has evolved into a more modular structure, making it easier to enhance and add new features in the future.

5 Requirement Analysis

5.1 Vision

Course registration system for computer engineering department. It is a system for student's choosing courses and waiting for approval from the advisors.

5.2 Glossary

5.2.1 Person

Definition: Represents an individual within the system.
Purpose: Serves as the base entity for various user roles like students, staff, advisors, and lecturers.
Example: Contains attributes such as name, surname, username, and password.

5.2.2 Staff

Definition: Represents university staff members.
Purpose: Includes employees like advisors and lecturers who have specific responsibilities related to managing students and courses.
Example: Staff members can set office hours, manage student registrations, and have roles like advisors or lecturers.

5.2.3 Advisor

Definition: A staff member responsible for advising students.
Purpose: Guides and supports students in academic matters, assists with course selection, and manages student course organization.
Example: Advisors maintain lists of assigned students and facilitate course approvals.

5.2.4 Lecturer

Definition: A staff member responsible for teaching courses.
Purpose: Delivers course content, facilitates discussions, and assesses students' performance through assignments and exams.
Example: Lecturers conduct lectures, manage course materials, and evaluate student progress.

5.2.5 Student

Definition: An individual enrolled in the university.
Purpose: Pursues academic programs, selects courses, and maintains academic records within the system.
Example: Students have personal details, academic statuses, and can enroll in or drop courses.

5.2.6 Course

Definition: Represents a university course.
Purpose: Provides information about course content, prerequisites, and scheduling for student enrollment.
Example: Courses have attributes like name, description, prerequisites, and scheduling details.

5.2.7 Mandatory Course

Definition: A type of course that students must take.
Purpose: Essential components of academic programs, ensuring students fulfill degree requirements.
Example: Mandatory courses have specific lecture dates, quotas, and, in some cases, lab hours.

5.2.8 Non-Technical Elective Course

Definition: An elective course unrelated to technical subjects.
Purpose: Offers students flexibility to choose courses outside their core academic disciplines.
Example: Non-technical elective courses have lecture dates, quotas, and may be offered remotely.

5.2.9 Technical Elective Course

Definition: An elective course related to technical subjects.
Purpose: Allows students to choose specialized courses within their academic discipline.
Example: Technical elective courses may have prerequisites, lecture dates, quotas, and required credit values.

5.2.10 Grade

Definition: Represents a student's performance grade in a specific course.
Purpose: Indicates a student's performance and progress within a course.
Example: A grade includes the course and the associated grade (e.g., "A," "B," "C," etc.).

5.2.11 Time Interval

Definition: Represents a time interval, such as lecture times.
Purpose: Used to define course schedules, office hours, and other time-related data.
Example: A time interval includes start and end times and the day of the week.

5.2.12 Controllers (AdvisorController, LecturerController, StudentController)

Definition: Manage interactions and operations related to advisors, lecturers, and students, respectively.
Purpose: Act as intermediaries between user interfaces and system functionality.
Example: AdvisorController manages student advising and course approvals.

5.2.13 Menus (AdvisorMenu, LecturerMenu, StudentMenu)

Definition: Provide user interfaces for advisors, lecturers, and students, respectively.
Purpose: Allow users to interact with the system and perform various tasks.
Example: StudentMenu provides options for course enrollment and viewing transcripts.

5.2.14 Course Section

Definition: Details of the class, time, and lecturers for the courses.
Purpose: Course sections provide specific information about when and where a course is held, as well as who is responsible for its instruction.
Example: A student checks the course section to find out the schedule and location of a particular class.

5.2.15 JSON

Definition: Notation used for data storage.
Purpose: JSON (JavaScript Object Notation) is a lightweight data-interchange format used to store and exchange information between systems.
Example: A web application uses JSON to transmit data between the server and client in a structured format.

5.2.16 Course Registration

Definition: The process of courses being selected by students and approved by advisors.
Purpose: Course registration enables students to choose the classes they wish to attend, while also ensuring that advisors can review and approve their selections.
Example: During course registration, a student selects their desired classes through the university's online portal, and the advisor approves the choices.

5.3 Requirements

5.3.1 Functional Requirements

- The system must support different roles, including Students, Advisors, Lecturers, and Staff.
- Students must be able to enroll in various types of courses, including Mandatory Courses, Non-Technical Elective Courses, and Technical Elective Courses.
- Students must be able to view the details of the courses they have enrolled in.
- Advisors must be able to manage a list of students.
- Advisors must be able to organize the course selections of their assigned students.
- Advisors must be able to approve or reject course selections made by students.
- Advisors must be able to add and delete students.
- Lecturers must be able to manage and view the courses they are responsible for.
- Lecturers must be able to add and delete courses they are lecturing.
- Staff members must have access to certain administrative functionalities, including managing user accounts.
- The system must allow for course prerequisites to be defined.
- The system must keep track of grades for individual courses.
- Students must be able to add and drop courses within specific timeframes.
- Students must be able to view their transcripts.

- The system must support the concept of "Sections" for courses, including setting dates and quotas for sections.
- The system must provide user authentication and login functionality.
- The system must ensure data consistency and maintain relationships between various entities, such as students, courses, and advisors.

5.3.2 Non-Functional Requirements

- The system must accept at max 5 courses for each student
- The system must inform the user at the end of operations
- The system should maintain detailed logs os system activities and errors for debugging and auditing purposes
- The system should comply with relevant data protection and privacy regulations
- The system should have at least 40 students from all classes, 5 advisors, lecturers and at least 10 different courses including NTE, FTE, TE at least 5 of them having prerequisites.

6 Use Cases

6.1 Use Case: Create Person

Steps:

Actor Actions	System Responses
1. User provides a name, surname, username, and password.	1. System creates a new person with the provided information.
	2. System responds with a success message: "Person created successfully."

6.2 Use Case: Retrieve Person Information

Steps:

Actor Actions	System Responses
1. User requests information about a person.	1. System retrieves the person's information (name and username).
	2. System displays the person's information.
	3. System responds by displaying the person's name and username.

6.3 Use Case: Update Person Information

Steps:

Actor Actions	System Responses
1. User selects to update a person's information.	2. User provides new name, surname, username, or password.
3. System updates the person's information with the new data.	4. System responds with a success message: "Person information updated successfully."

6.4 Use Case: Authenticate

Steps:

Actor Actions	System Responses
1. User enters their username and password.	2. System checks the provided credentials.
3. If credentials are valid, the system retrieves the user's information and displays it along with the main menu.	4. System responds with a success message: "Authentication successful."
5. System displays the user's information and the main menu.	

6.5 Use Case: Enroll in Course

Steps:

Actor Actions	System Responses
1. Student selects a course to enroll in.	2. System adds the selected course to the student's list of selected courses.
	3. System responds with a success message: "Course enrolled successfully."

6.6 Use Case: Drop Course

Steps:

Actor Actions	System Responses
1. Student selects a course to drop.	2. System removes the selected course from the student's list of selected courses.
	3. System responds with a success message: "Course dropped successfully."

6.7 Use Case: View Selected Courses

Steps:

Actor Actions	System Responses
1. Student requests to view their selected courses.	2. System retrieves and displays the list of courses selected by the student.
	3. System responds by displaying the list of selected courses.

6.8 Use Case: View Available Courses

Steps:

Actor Actions	System Responses
1. Student requests to view available courses.	2. System retrieves and displays the list of courses available for enrollment.
	3. System responds by displaying the list of available courses.

6.9 Use Case: View Transcript

Steps:

Actor Actions	System Responses
1. Student requests to view their transcript.	2. System retrieves and displays the student's transcript, including grades for completed courses.
	3. System responds by displaying the student's transcript.

6.10 Use Case: Send Approval Request

Steps:

Actor Actions	System Responses
1. Student sends an approval request for a specific action (e.g., dropping a course).	2. System processes the request and notifies the relevant staff (advisor or lecturer).
3. Staff reviews and approves/rejects the request.	4. System responds with an appropriate message based on the staff's decision.

6.11 Use Case: View Courses

Steps:

Actor Actions	System Responses
1. Staff member (advisor or lecturer) requests to view the courses they are responsible for.	2. System retrieves and displays the list of courses associated with the staff member.
	3. System responds by displaying the list of courses.

6.12 Use Case: View Students

Steps:

Actor Actions	System Responses
1. Staff member (advisor or lecturer) requests to view the list of students they are advising or teaching.	2. System retrieves and displays the list of students associated with the staff member.
	3. System responds by displaying the list of students.

6.13 Use Case: Add Office Hours

Steps:

Actor Actions	System Responses
1. Advisor selects to add office hours to their schedule.	2. Advisor specifies the date and time for the office hours.
3. System adds the office hours to the advisor's schedule.	4. System responds with a success message: "Office hours added successfully."

6.14 Use Case: Delete Office Hours

Steps:

Actor Actions	System Responses
1. Advisor selects to delete office hours from their schedule.	2. Advisor specifies the office hours to be deleted.
3. System removes the specified office hours from the advisor's schedule.	4. System responds with a success message: "Office hours deleted successfully."

6.15 Use Case: Add Course (Lecturer)

Steps:

Actor Actions	System Responses
1. Lecturer selects to add a course they will be lecturing.	2. Lecturer provides the details of the course (course information).
3. System adds the course to the lecturer's list of lectured courses.	4. System responds with a success message: "Course added successfully."

6.16 Use Case: Delete Course (Lecturer)

Steps:

Actor Actions	System Responses
1. Lecturer selects to delete a course they were lecturing.	2. Lecturer specifies the course to be removed.
3. System removes the specified course from the lecturer's list of lectured courses.	4. System responds with a success message: "Course deleted successfully."

6.17 Use Case: Create Course

Steps:

Actor Actions	System Responses
1. Administrator selects to create a new course.	2. Administrator provides course details, including short name, full name, description, prerequisites, semester, credit, and class hours.
3. System creates the new course with the provided information.	4. System responds with a success message: "Course created successfully."

6.18 Use Case: Update Course Information

Steps:

Actor Actions	System Responses
1. Administrator selects to update course information.	2. Administrator provides updated course information, including prerequisites and class hours.
3. System updates the course information with the new data.	4. System responds with a success message: "Course information updated successfully."

6.19 Use Case: Remove Prerequisite

Steps:

Actor Actions	System Responses
1. Administrator selects to remove a prerequisite from a course.	2. Administrator specifies the prerequisite to be removed.
3. System removes the specified prerequisite from the course.	4. System responds with a success message: "Prerequisite removed successfully."

6.20 Use Case: Add Prerequisite

Steps:

Actor Actions	System Responses
1. Administrator selects to add a prerequisite to a course.	2. Administrator specifies the prerequisite to be added.
3. System adds the specified prerequisite to the course.	4. System responds with a success message: "Prerequisite added successfully."

6.21 Use Case: Login

Steps:

User Actions	System Responses
1. User enters their username and user password.	2. System checks the username and user password.
3. If credentials are valid, the system retrieves the user's information and displays it along with the main menu.	4. System responds with a success message: "Authentication successful."
5. System displays the user's information and the main menu.	

6.22 Use Case: Logout

Steps:

User Actions	System Responses
1. User selects the logout option from the menu.	2. System logs the user out of the system.
	3. System responds with a success message: "Logout successful."

6.23 Use Case: Student Course Organization

Steps:

User Actions	System Responses
1. Student selects to organize their course selections.	2. Student arranges the order and preferences of their selected courses.
3. System updates the student's course organization based on their preferences.	4. System responds with a success message: "Course organization updated successfully."

6.24 Use Case: Approve Student Selection

Steps:

User Actions	System Responses
1. Advisor receives a request from a student to approve their course selection.	2. Advisor reviews the student's course selection.
3. Advisor approves the student's course selection.	4. System responds with a success message: "Course selection approved."

6.25 Use Case: Reject Student Selection

Steps:

User Actions	System Responses
1. Advisor receives a request from a student to approve their course selection.	2. Advisor reviews the student's course selection.
3. Advisor rejects the student's course selection.	4. System responds with a message indicating the rejection and reason.

6.26 Use Case: View Enrolled Students (Lecturer)

Steps:

User Actions	System Responses
1. Lecturer selects to view the list of students enrolled in their courses.	2. System retrieves and displays the list of students enrolled in the lecturer's courses.
	3. System responds by displaying the list of enrolled students.

6.27 Use Case: Give Course

Steps:

Actor Actions	System Responses
1. Select given course from menu	2. List the course that can given by him/her
3. Write a course that wanna give	4. Add the written course to selectable list
5. Write the quota	6. Go to step 2

6.28 Use Case: Add Course

Steps:

Actor Actions	System Responses
1. Select add course from menu	2. Print the what need to adding course
3. Write the course full name	
4. Write the course short name	
5. Write the description of course	6. Save the new course informations

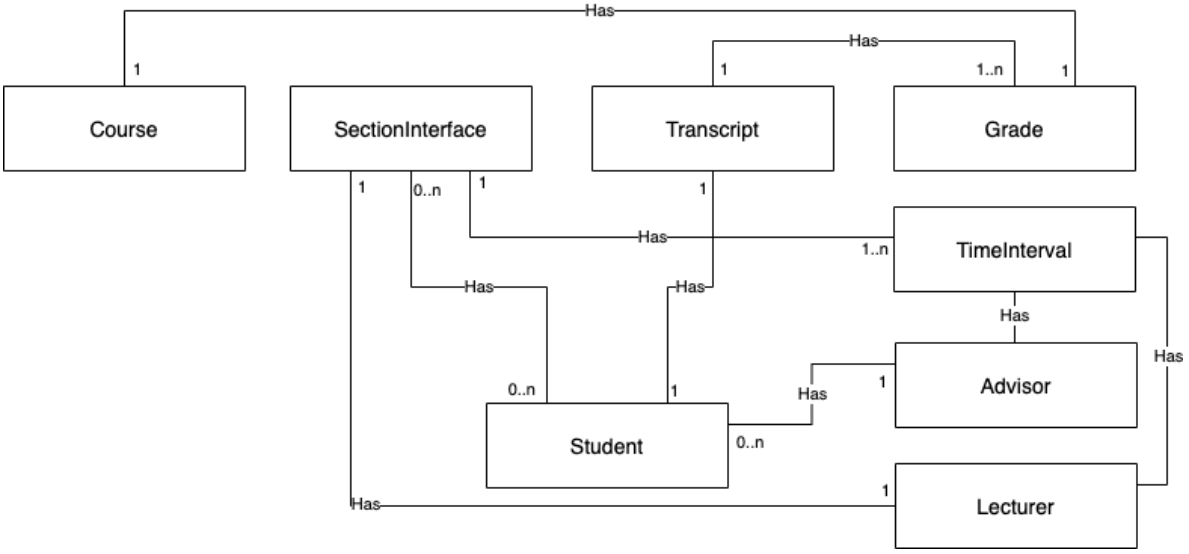
6.29 Use Case: Logout

Steps:

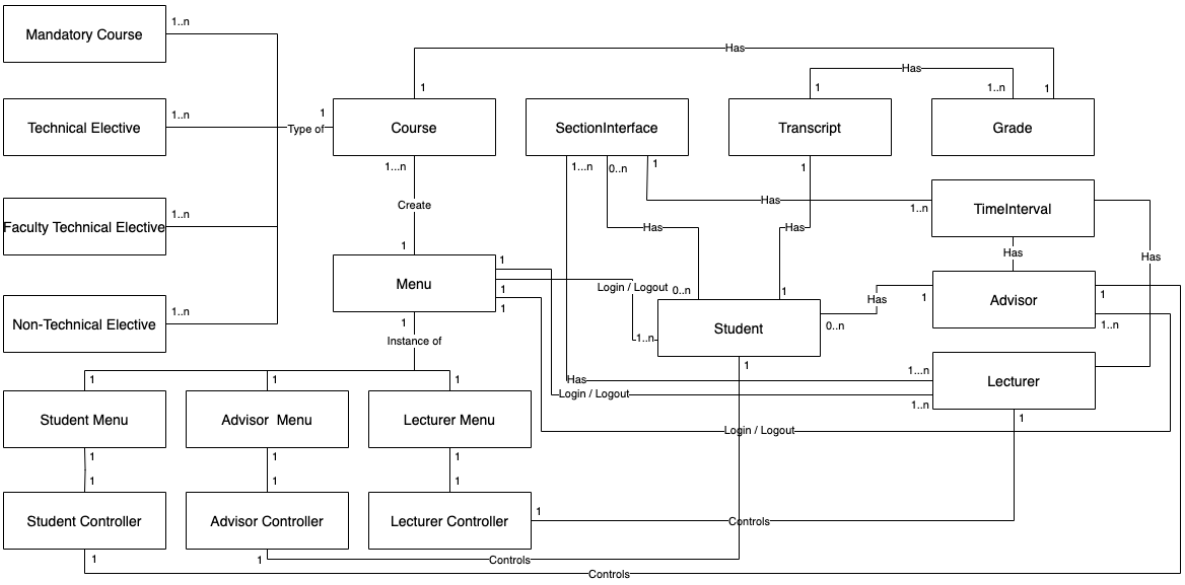
Actor Actions	System Responses
1. Select the logout from menu	2. Select the logout from menu
	3. Print the login screen

7 Diagrams and Models

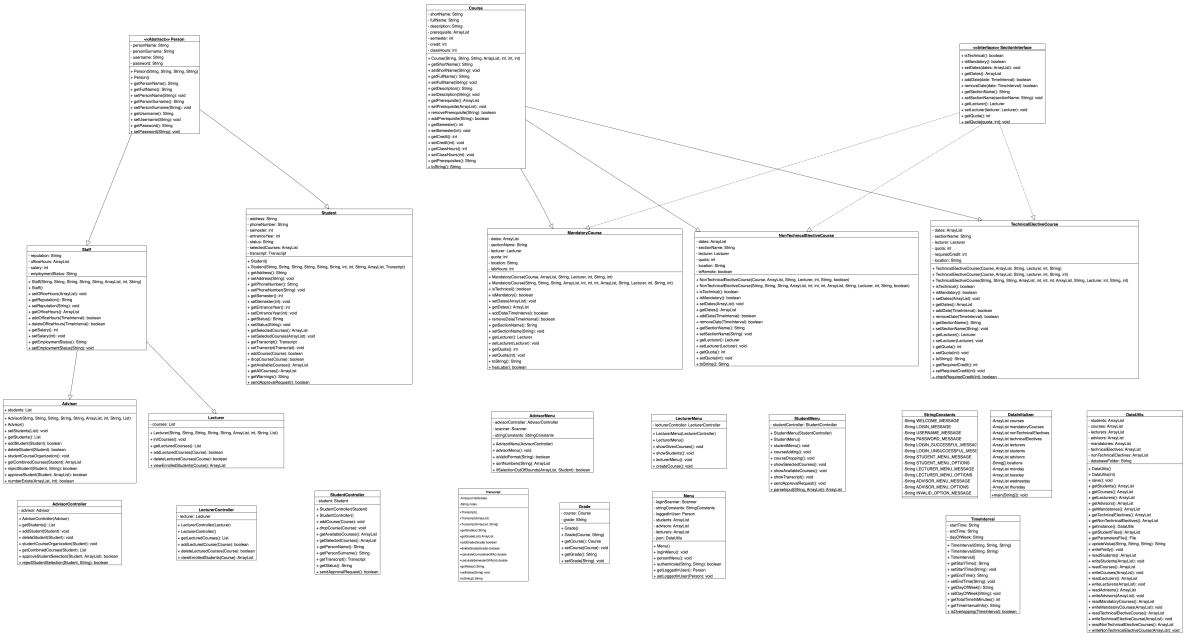
7.1 Domain Model - Without Concepts



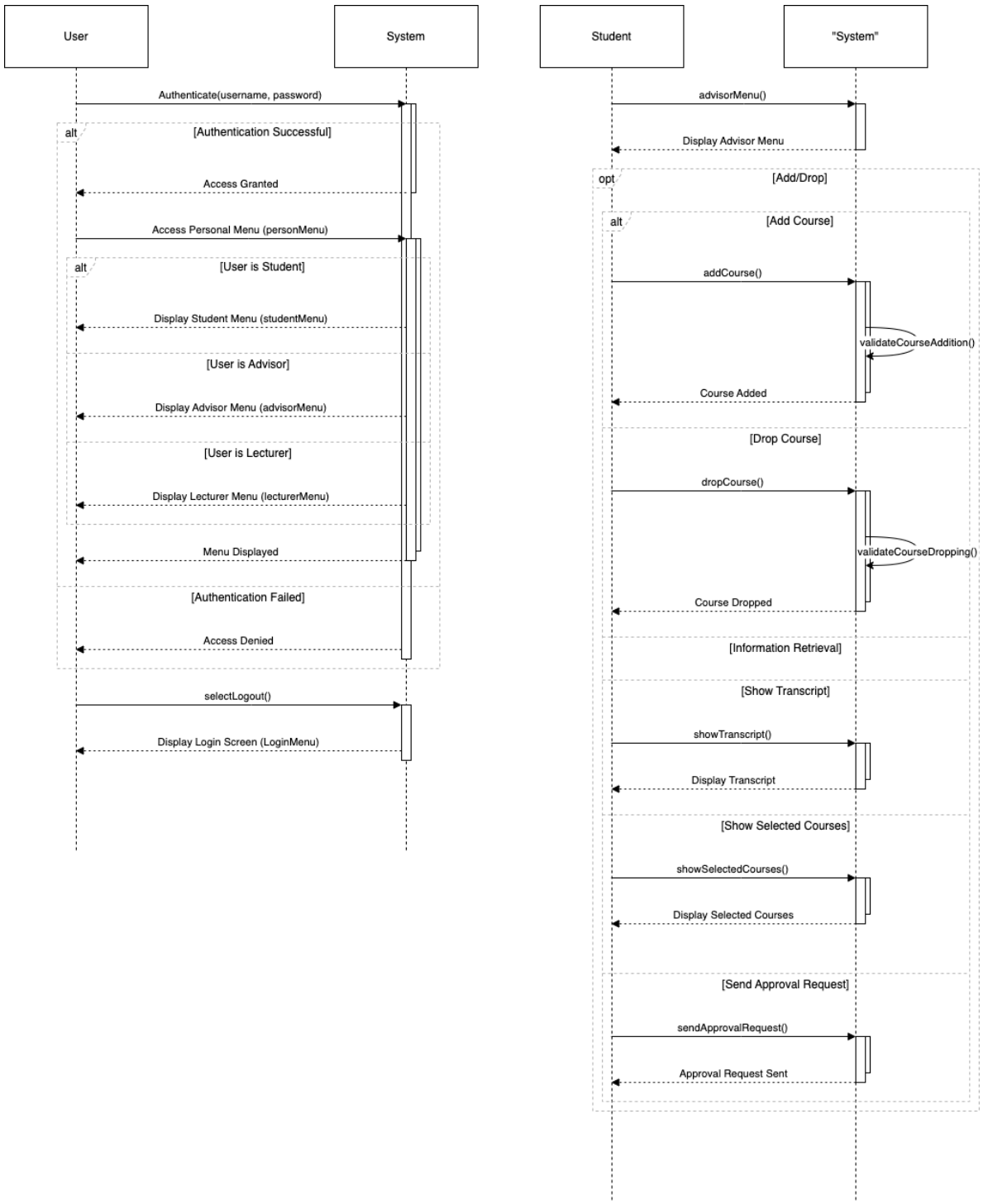
7.2 Domain Model - With Concepts

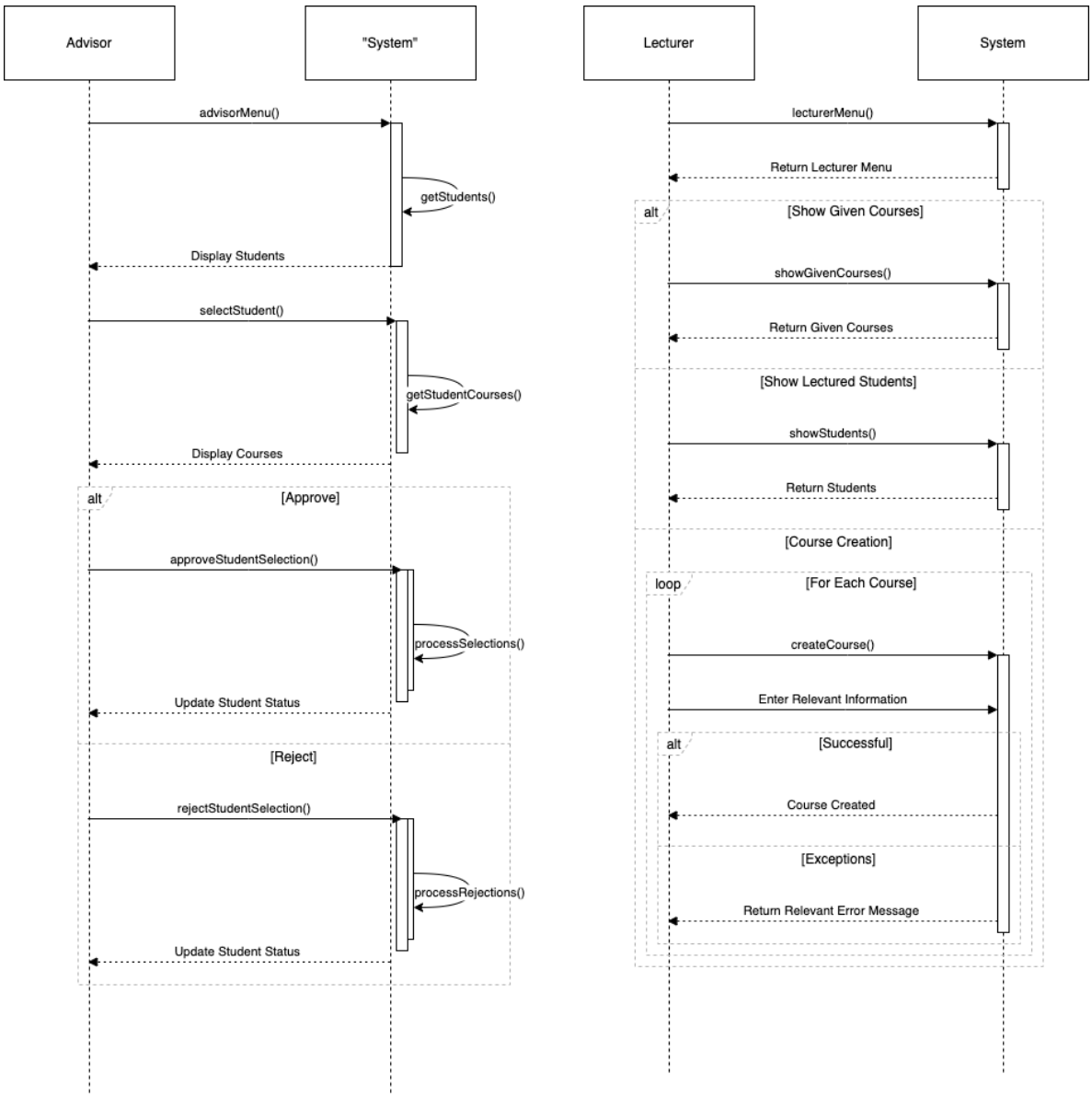


7.3 Design Class Diagram (DCD)



7.4 System Sequence Diagrams (SSD)




Advisor
"System"
advisorMenu()
getStudents()
Display Students
selectStudent()
getStudentCourses()
Display Courses
alt
[Approve]
approveStudentSelection()
processSelections()
Update Student Status
[Reject]
rejectStudentSelection()
processRejections()
Update Student Status
Lecturer
System
lecturerMenu()
Return Lecturer Menu
alt
[Show Given Courses]
showGivenCourses()
Return Given Courses
[Show Lectured Students]
showStudents()
Return Students
[Course Creation]
loop
[For Each Course]
createCourse()
Enter Relevant Information
alt
[Successful]
Course Created
[Exceptions]
Return Relevant Error Message

7.5 Design Sequence Diagrams (DSD)