Marmara University
Faculty of Engineering

CSE3063
Object-Oriented Software Design

Requirement Analysis Document

Instructor: Murat Can GANİZ Date:12.11.2023

	Department	Student Id Number	Name & Surname
1	CSE	150120012	Kadir BAT
2	CSE	150120055	Muhammed Talha KARAGÜL
3	CSE	150120020	Mustafa Said ÇANAK
4	CSE	150121520	Ensar Muhammet YOZGAT
5	CSE	150121076	Abdullah KAN
6	CSE	150120997	MOHAMAD NAEL AYOUBİ
7	CSE	150121021	Feyzullah ASILLIOĞLU

Abstract

In this project, our aim is to develop a functional course registration system tailored for use within our university department. The system is designed to support an interactive course registration process between students and advisors. Students will have the ability to enroll in courses, view their transcript information, consult their weekly class schedules, and monitor their performance in registered courses during the course registration process.

Furthermore, advisors will have the capability to guide and assist the students they advise, facilitate the course selection process, and view detailed student information through the system. The system aims to create an efficient and collaborative environment for both students and advisors, streamlining the course registration experience and enhancing the overall academic support system within the department.

Glossary

- **Course:** A syllabus module offered by the university.
- **Elective Course:** A course that may be chosen from any other department. When added to mandatory courses, form the total credits needed to graduate.
- **Non-Elective Course:** An essential, mandatory course in the department that students are required to take.
- **Prerequisite Course:** A course that must be completed by a student to be allowed to take a higher-level course.
- **Prerequisite Tree:** A hierarchical illustration that represents the required order of prerequisite courses, providing the relationships between courses.
- **User Interface:** Interactive labels used by students to interact with the system.
- **Course:** Request: A request sent by the student to his/her advisor to take a course at the beginning of the semester.
- **Weekly Schedule:** A timetable that provides the distribution of the courses over a week.
- **Credit Limit:** The maximum number of credits set by the university to take by a student for a semester.
- **Enrolment Capacity:** The maximum number of students to register for a course.
- **Check Permissibility:** Checking if a student is eligible or allowed to register for a course.
- **Actor:** A user that interacts with the system, which is a student and advisor in our case.
- **Advisor:** A lecturer who mentors many students, traces their performances, and approves the course requests sent by them for the upcoming semester.
- **Domain Model:** An illustration that shows conceptual real-world classes in the problem domain.
- **Use Cases:** A collection of success and failure scenarios that describes an actor using the system for a purpose.
- **JSON File:** A language-independent text file that contains readable and interchangeable data. These files include information for students and advisors.
- **Transcript:** A detailed record of the student that shows his/her completed courses.
- **Non-Functional Requirements:** Requirements that describe the attributes and properties of the system.
- **Functional Requirements:** Requirements that describe features and behaviors wanted in the system.
- **System Sequence Diagram:** An illustration that shows the events that external actors generate for a specific use case.

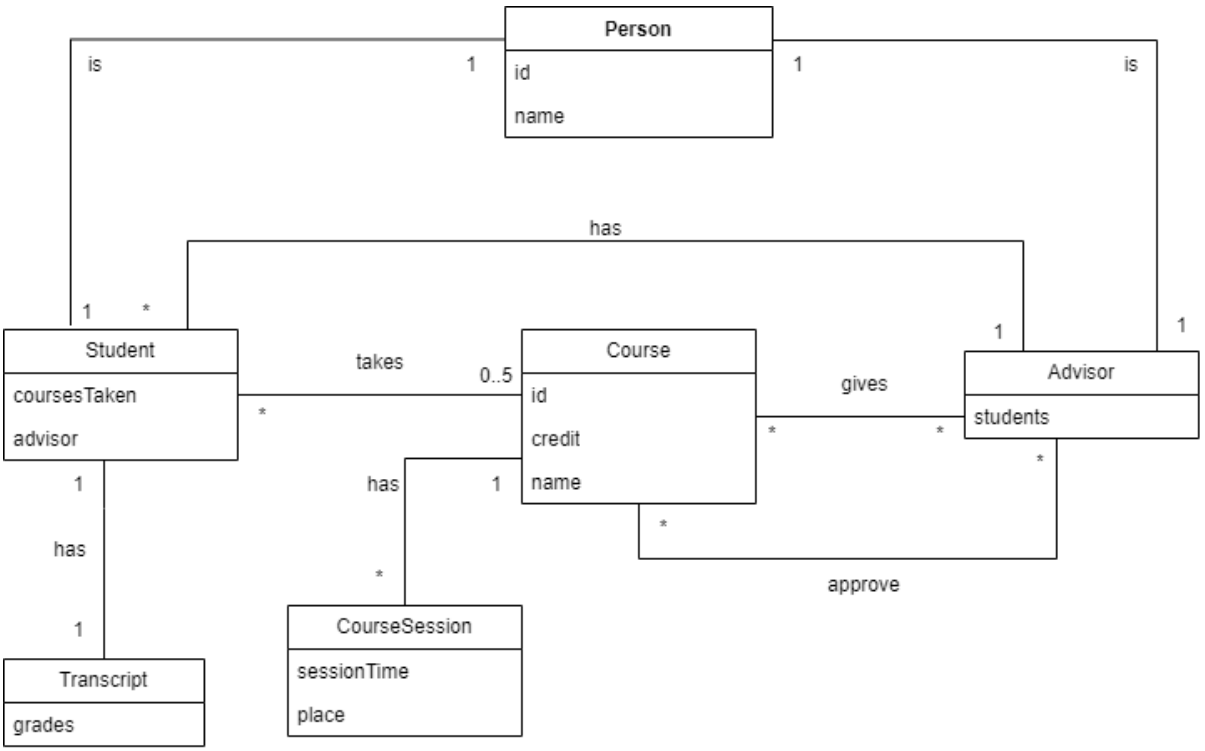
Functional Requirements

- Users who are Students and Advisor's personal data should be kept in .json files.
- The user should log in with a pre-created unique user ID and password. Login information will be checked from the contents of the pre-created .json files.
- For each student, the course registration process should be done based on availability and prerequisites and problems with handling registration should be checked.
- The system will provide the information on the prerequisite tree of courses to students based on their request to see the courses they can register for.
- Advisors can approve or deny courses and should decide the approval of their students' courses due to predefined university rules.
- Each student should have a transcript kept as separate .json files for each student. Successfully registered courses will be recorded on the student’s transcript which will have both failed and passed courses.
- Students can view their transcripts by sending requests to the system. Also, they can see their syllabus which shows the course section information.

Non-functional Requirements

- **Performance**: The login and course registration process should be quick and maintain efficiency with minimum delays.
- **Expandability**: As input data increases in the system, the program should allow a huge number of users without causing system failure.
- **Maintainability**: The program should be open for modifications and maintain any changes done to our code, which might help us as we move forward in the next iterations.
- **Security**: Users’ passwords should be encrypted due to security measures.
- **Accessibility**: The program should have a user-friendly interface to avoid complications.
- **Consistency**: Reliability of the program is a must as data or case conditions change, the program should give consistent results.

Domain Model



Use Cases

Use Case Name: Registration for the Course

Summary:

Designed to enable students to successfully register for courses in the upcoming semester. This scenario involves interactions where students and advisors engage in the course selection, validation, and confirmation processes.

Actor:

Primary Actor:

- Student

Supporting Actor:

- Course Registration System

Stakeholders and Interests:

- **Student:** Wants to successfully register for courses for the upcoming semester.
- **Advisor:** Provides academic counseling to his/her students who is a lecturer. And he/she is the person responsible for his/her students' course selections.
- **Course Registration System:** Allows students to see the courses they will take and their syllabus based on the courses they will take.

Preconditions:

- The student must be logged into the system.
- A student has only one advisor.

Success Guarantee (Postconditions):

- The courses selected by the Students
- Students’ requests are saved into the JSON file
- The requests sent to the Advisor

Flow of Events:

Main Success Scenario (or Basic Flow):

Actor Action (or Intention)	System Responsibility
1. The Students enter the Course Registration System.	
	2. Retrieves student information from the database (JSON file)
3. The Students can see a list of all the courses he/she can take.	
4. The Students select the course they want. The Student repeats steps 3-4 until he/she wants to save the selection.	
5. The Students save their course selection.	
	6. Saves information to the database (.json file) **7.** Place lessons in the syllabus
8. The Students can view their selected courses and total credits.	
9. The Students can see their syllabus.	
10. The Students send their selection requests to the advisor.	
	11. The student registration process updated

Extensions (or Alternative Flows):

4a. If there is an overflow in the credit system, give an error:

1. Go to step 2
2. Choose the courses that don't exceed the credit limit that can be taken

4b. If there is an overflow, the maximum number of courses is:

3. Go to step 2
4. Choose the courses that don't exceed the maximum number of courses that can be taken

4c. If the course you choose does not meet the prerequisites:

1. Go to step 2
2. Choose the courses that meet the prerequisites.

4d. If there is a conflict in the course section:

1. Give an error message to the Student
2. Go to step 2
3. Choose the courses that don’t conflict

7a. If the Advisor rejects a course:

1. Go to step 1
2. Do the operations again

8a. If the student wants to delete their selection:

1. Go to the deletion tab in the Course Registration Interface.
2. Prompt to user to which courses to be deleted

Special Requirements:

- The system must have up-to-date information on course availability, prerequisites, and enrollment capacity.

Technology and Data Variations List:

1-7a. Operations are done by keyboard.

Frequency of Occurrence:

- This use case occurs at the beginning of each semester when students need to register for courses.

Open Issues:

- How will the system handle problems where a course section reaches maximum capacity during registration?

Use Case Name: Course Registration Approve/Denied

Summary:

Crafted to facilitate advisors in efficiently guiding students through the course registration process for the upcoming semester. This scenario entails interactions where advisors and students collaboratively participate in course selection, validation, and confirmation procedures.

Actor:

Primary Actor:

- Advisor

Supporting Actor:

- Course Registration System

Stakeholders and Interests:

- **Student:** Wants to successfully register for courses for the upcoming semester.
- **Advisor:** Provides academic counseling to his/her students who is a lecturer. And he/she is the person responsible for his/her students' course selections.
- **Course Registration System:** Allows students to see the courses they will take and their syllabus based on the courses they will take.

Preconditions:

- The advisor must be logged into the system.
- An advisor must have at least one student.

Success Guarantee (Postconditions):

- The selected course information is sent to the Student
- Approve/denied saved into the JSON file
- The student's transcript is updated

Flow of Events:

Main Success Scenario (or Basic Flow):

Actor Action (or Intention)	System Responsibility
1. The Advisor enters the Course Registration System.	
	2. Retrieves students and requests information from the database (JSON file).
3. The Advisor sees all students under his/her responsibility.	
4. The Advisor chooses a student.	
	5. Retrieve student's courses from the database (JSON file).
6. The Advisor sees the courses requested by the student.	
7. The Advisor selects which courses to approve/deny.	
	8. Saves approved/denied courses to the database (JSON file).
	9. The student's course registration status is updated in the database (JSON file).

Special Requirements:

- The system must have up-to-date information on course availability, prerequisites, and enrollment capacity.

Technology and Data Variations List:

1-7a. Operations are done by keyboard.

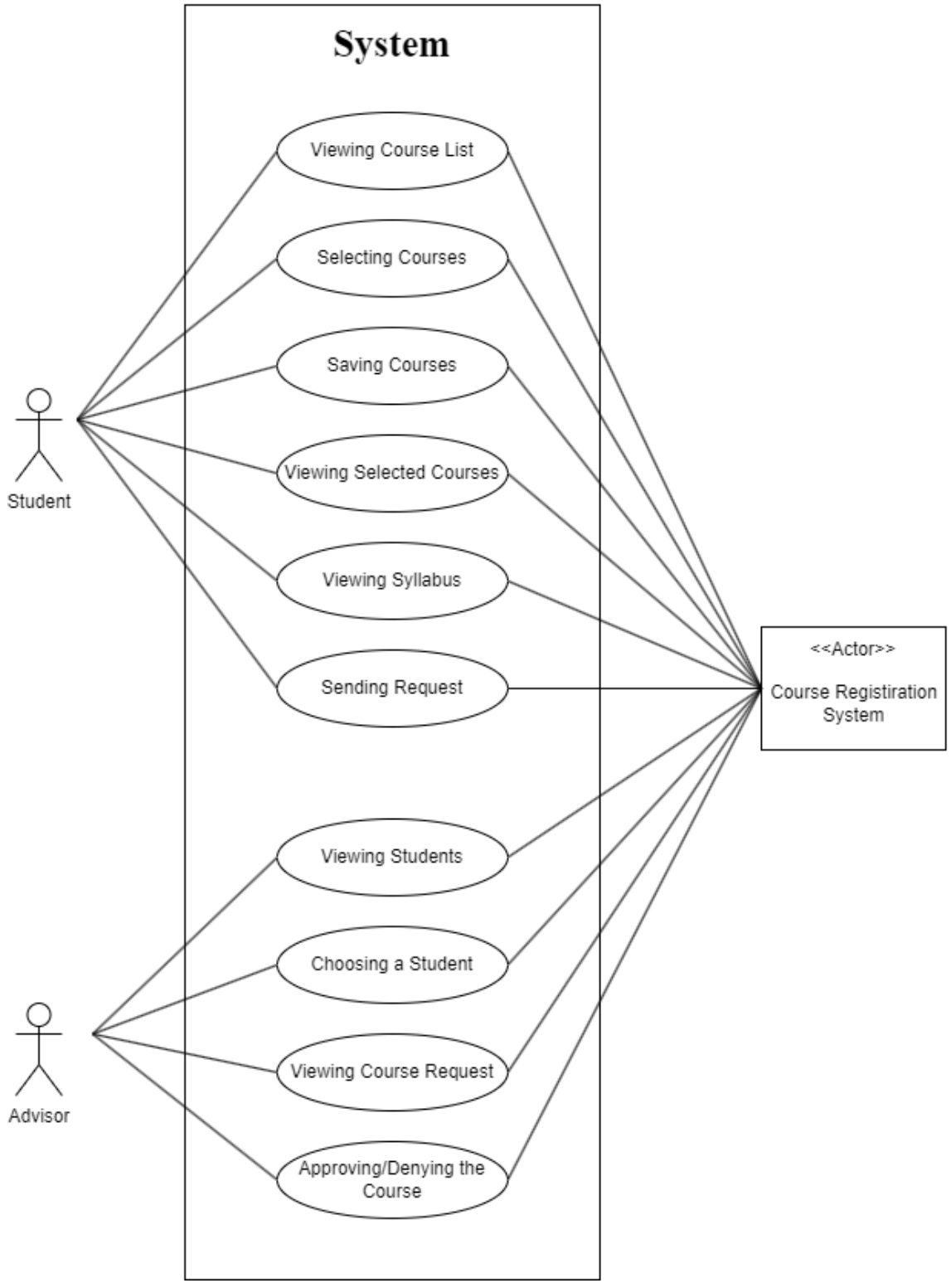
Frequency of Occurrence:

- This use case occurs at the beginning of each semester when students need to register for courses.

Open Issues:

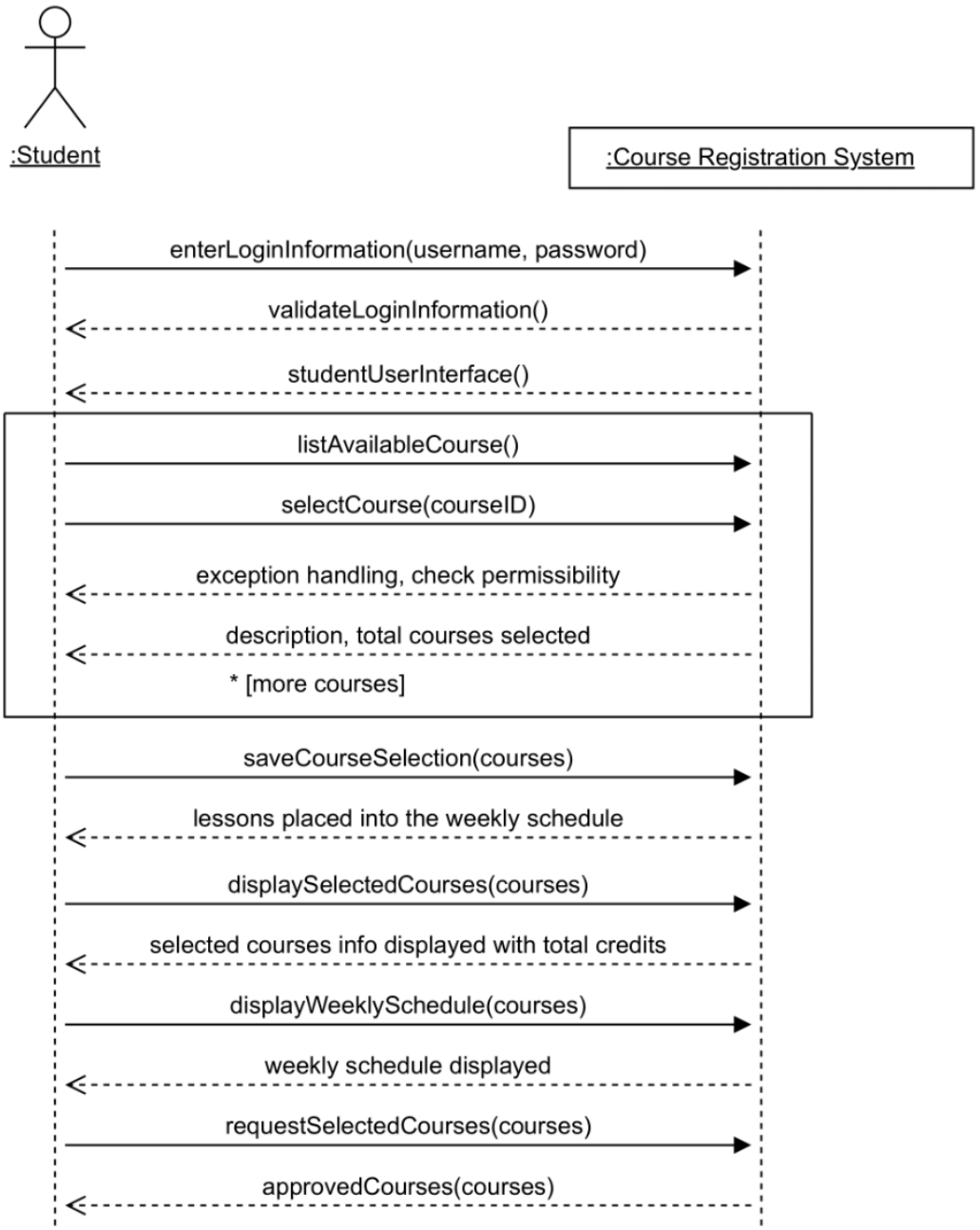
- How will the system handle problems where a course section reaches maximum capacity during registration?

Use Case Diagram



System Sequence Diagrams

Student - System (SSD_1)



Advisor - System (SSD_2)