Marmara University

Engineering Faculty - Computer Engineering

CSE3063 – Object-Oriented Software Design

Project Iteration-3

Requirements Analysis Document

Group-5 Members

Havva Nur Özen 150119771

İlker Keklik 150120074

Yasin Küçük 150120062

Kerim Hasan Yıldırım 150120040

Sarp Sıraç Müftüler 150119717

Yusuf Duman 150120023

Table of Contents

Vision

The project aims to simulate a course registration system for Computer Engineering students at Marmara University. The registration system will contain the students, lecturers, advisors, course, and the Department. To simulate the system, students, lecturers, advisors, and courses will be created manually considering the necessary number of instances to access all the functionalities of the project.

Scope

The scope of the project involves the creation of a comprehensive simulation for the Computer Engineering course registration system at Marmara University. The primary objectives are to develop user-specific modules for students, lecturers, and advisors, enabling functionalities such as course registration, grade submission, academic planning, and resource management. Manual creation of instances for students, lecturers, advisors, and courses will be undertaken to ensure robust testing and accessibility of all system features. The system will include a secure authentication mechanism, data validation, and authorization controls. Documentation, including user manuals and technical guides, will be provided to facilitate user understanding. The project also considers potential future enhancements and scalability, presenting a well-structured timeline for development, testing, and deployment. Overall, the simulated course registration system aims to streamline administrative processes and enhance the overall academic experience at Marmara University.

Description of The Project

Every person in the system has a randomly generated person ID assigned to them. Besides person IDs, all staff and students have randomly generated student IDs and staff IDs. To differentiate the ID types, the prefixes of the IDs are assigned differently for person ID, staff ID, and student ID.

All students have an advisor assigned to them to approve the requested courses by the student. Every course request needs approval from the advisor for the student to take the course.

All courses have semester information, and some courses have prerequisite courses.

There are conditions students must consider before requesting a course:

1) The student must have successfully completed prerequisite courses of the course.
2) The student must have enough completed credits to enroll for the course depending on the course's semester information.

3) The course the student wants to choose must be a valid course in the department.
4) The total of active courses and sent course requests of the student can't be more than the maximum allowed number of courses for a semester.

If all the conditions are not met by the student, the system will not let the student send the request to his/her advisor. If the student meets all the conditions, he/she can send a request to the advisor to take the course for its approval.

Advisors possess the ability to review and approve/reject course requests from students and access a list of students under their supervision. It has also more functional abilities for the courses he teaches and the students he supervises.

In addition to course operations, students can utilize the system to view their transcripts, encompassing information on total completed credits, both successful and unsuccessful course attempts, and corresponding letter grades.

Use Case for Course Registration System

Actors: Student, System

1. The student accesses the Course Registration System and enters their student ID and password.
2. The student lists the courses he/she has taken.
3. The student displays the Transcript.
4. The student lists courses that he/she has not completed and is not enrolled in.
5. The student selects the option to register for a new course.
    a. The system prompts the student to enter the course code for the desired course.
6. The student enters the course code.
    a. The system checks if the student meets the prerequisites for the selected course.
    b. If the prerequisites are not met, the system provides an error message and returns to the menu.
7. If the prerequisites are met, the system displays the available sections for the selected course, showing section numbers, quotas, number of students, and lecture times.
8. The student selects a section by entering the section number.
    a. The system sends the registration request to the advisor.
9. The advisor receives the registration request and can either confirm or reject it.
    a. If confirmed, the system updates the student's registration status to "Confirmed."

b. If rejected, the system updates the student's registration status to "Rejected."

Actors: Advisor, System

1. Advisor accesses the Course Registration System and enters their staff ID and password.
2. Advisor lists active registrations under his/her responsibility.
3. Advisor lists the students he/she advises.
4. Advisor evaluates a registration request.
5. Advisor lists the courses he/she teaches.
6. Advisor assigns grades to the students of the courses he/she teaches on the next screen after listing their courses.

Actors: Lecturer, System

1. Lecturer accesses the Course Registration System and enters their staff ID and password.
2. Lecturer displays a list of courses taught by himself/herself.
3. The lecturer selects a specific course (e.g., "Digital Logic Design" with the course code "CSE3215") to perform actions.
4. The system presents options for the selected course:
    a. Option 1: List students enrolled in the course.
    b. Option 2: Grade a student in the course.
    c. Option 0: Go back to the previous menu.
5. The lecturer chooses option 1 to list students enrolled in the course.
6. If there are no enrolled students, an empty list is displayed.
7. The lecturer chooses option 2 to grade a student.
8. The system prompts the lecturer to enter the student number for grading.
9. If the entered student number is not found, an appropriate message is displayed.
10. The lecturer enters a grade for the selected student (e.g., "86").
11. The system updates the student's grade for the course.
12. The system confirms the successful grading to the lecturer.

Functional Requirements

1. The system must ensure that each student and advisor is assigned an identical student number and staff number, respectively.
   - **Priority**: High
   - **Criticality**: It is crucial for both students and advisors to log in to the system and perform essential operations.
   - **Risks**: Failure to provide identical student or advisor numbers will result in the absence of a private account for users, preventing them from carrying out necessary operations.

2. The system must provide the available courses for students to send course registration requests.
   - **Priority**: High
   - **Criticality**: This functionality is crucial for students to make informed decisions about their course registration, impacting the core user experience.
   - **Risks**: If the available courses are not accurately presented, students might request courses that are not being offered, leading to confusion and potential enrollment issues.

3. The system must provide a transcript for each student.
   - **Priority**: High
   - **Criticality**: While important for students to track their academic progress, it is not as time-sensitive as some other functionalities.
   - **Risks**: Not providing transcripts can cause students not to complete any course.

4. The system must control the prerequisites of courses when students send registration requests.
   - **Priority**: High
   - **Criticality**: Essential for ensuring that students meet necessary requirements before attempting a course, preventing enrollment in classes for which they are not adequately prepared.
   - **Risks**: Without proper prerequisite control, students may register for courses they are not qualified to take, leading to difficulties and disruptions in the learning process.

5. The system must provide advisors with the sent course registrations.
   - **Priority**: High
   - **Criticality**: Critical for advisors to review and manage student course registrations, ensuring proper academic guidance and resource allocation.

- **Risks**: If advisors cannot access sent course registrations, there is a risk of delayed or inefficient approval processes, impacting the overall efficiency of the academic advising system.

6. The system must assign Lecturers to courses.
   - **Priority**: High
   - **Criticality**: Important for providing students with grades to complete the courses.
   - **Risks**: Without proper advisor assignments, students cannot receive grades which means they cannot complete the courses.

7. The system must allow advisors to give grades to students in the selected course.
   - **Priority**: High
   - **Criticality**: Essential for students to complete the courses.
   - **Risks**: If advisors cannot assign grades, students cannot receive proper feedback, and academic records may be incomplete or inaccurate.

8. The system must provide advisors with the courses and students of the courses.
   - **Priority**: High
   - **Criticality**: Crucial for advisors to have access to the necessary information for effective academic supervision and support.
   - **Risks**: Without proper access to course and student information, advisors cannot reach the students of the courses and provide them with grades, potentially impacting students' transcripts.

9. The system must save all data before terminating the program.
   - **Priority**: High
   - **Criticality**: Critical for data integrity and system reliability, ensuring that no information is lost during or after system shutdown.
   - **Risks**: Failure to save data before termination may result in the loss of critical information, impacting the overall functionality and reliability of the system.

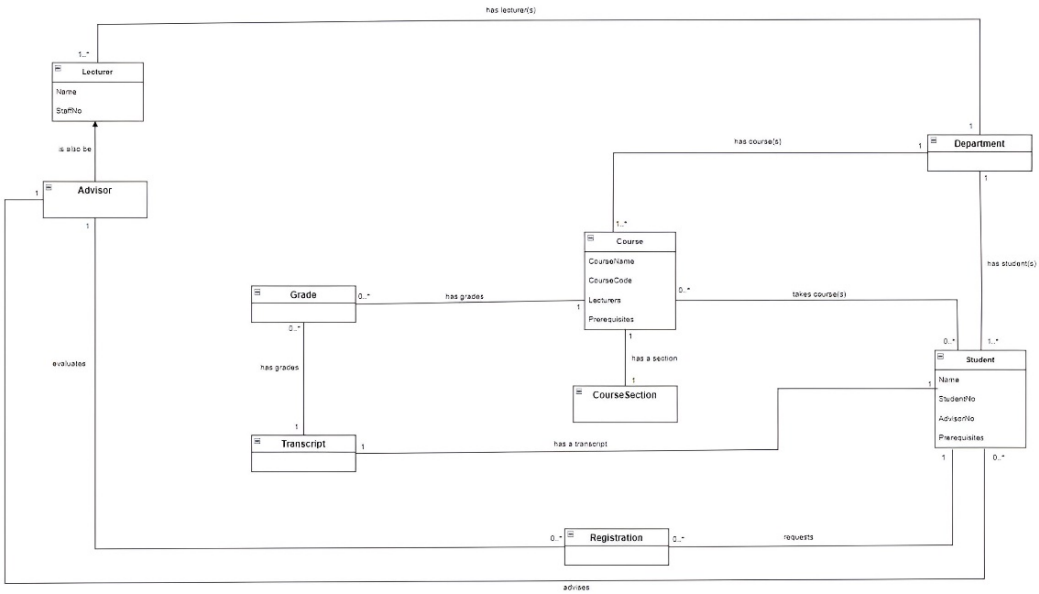
10. The system must check whether the student has taken what he wants to take before, whether he has taken it now, whether he can take more than 5 courses, more than 36 credits, whether he can take courses from an upper class, and whether he complies with the course prerequisites.
    - **Priority**: High
    - **Criticality**: Preventing course repetition and semester conflicts is crucial for accurate academic records and progression.
    - **Risks**: Failing to accurately track course history may lead to misinformation in academic records. Academic Progression Issues: Overlooking course repetition limits can hinder students' academic progress.

11. The system must implement logging functionality to record significant events during its operation.
    - **Priority:** Medium
    - **Criticality:** Essential for system monitoring, issue diagnosis, and maintenance.
    - **Risks:** Absence of proper event logging may impede the system's ability to identify and address issues promptly, affecting overall reliability.

Non-Functional Requirements

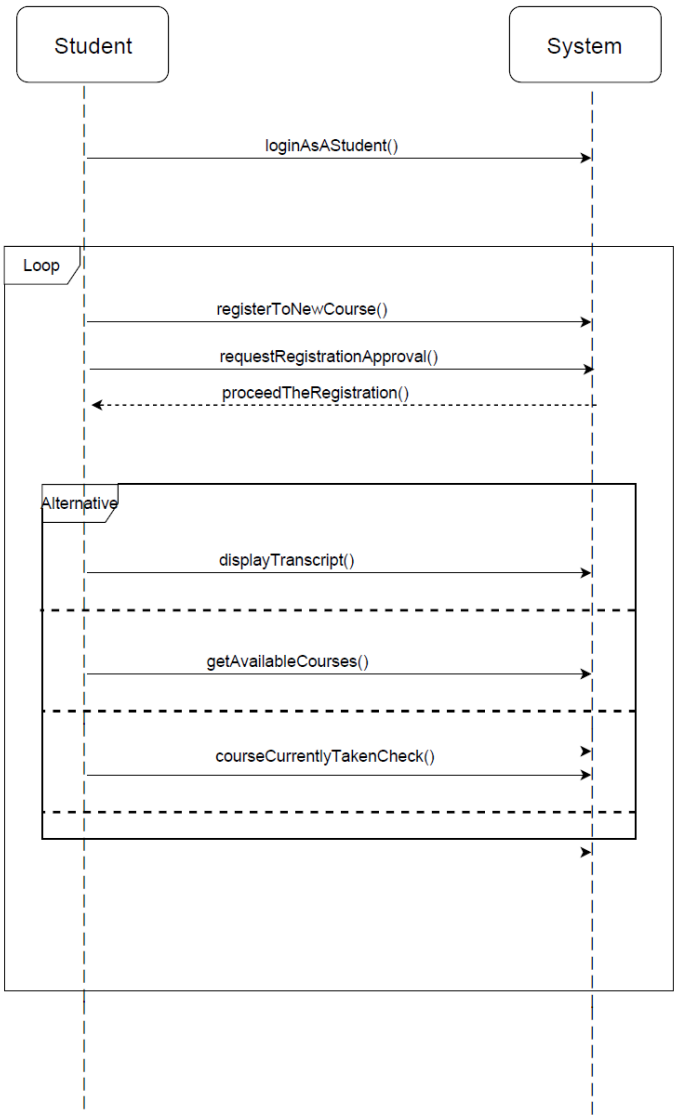
1. **Security:** The system must implement robust security measures to protect user data and prevent unauthorized access.
2. **Performance:** The system should provide efficient response times, ensuring a smooth user experience.
3. **Availability:** The system must have minimal downtime, ensuring continuous availability for users.
4. **Maintainability:** The system should be designed for easy maintenance, with considerations for extensibility and reusability of code components.
5. **Portability:** The system should be compatible with standard web browsers, ensuring accessibility across different platforms.
6. **Product Cost:** The system development and maintenance costs should be kept within a defined budget.

Domain Model

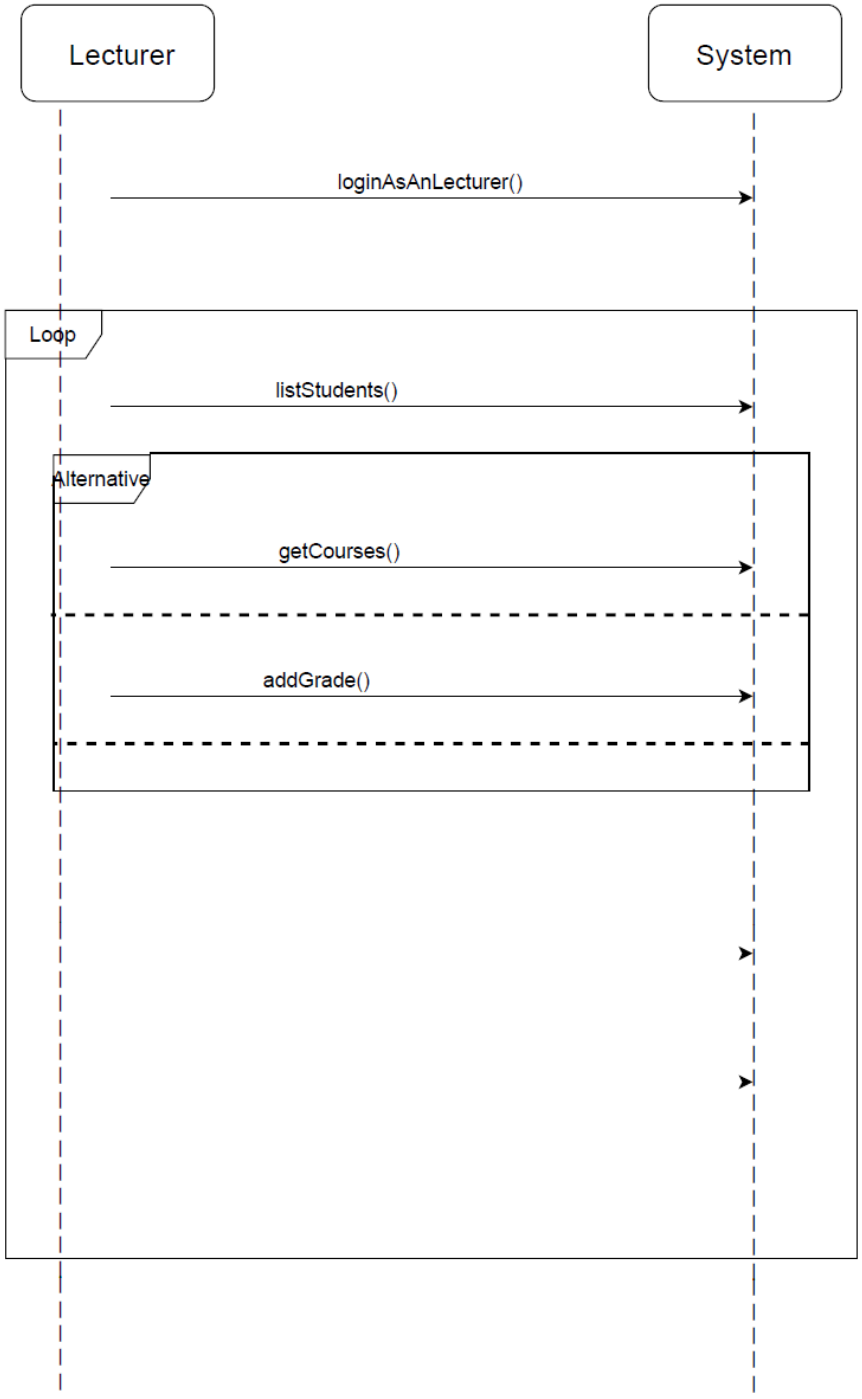


System Sequence Diagram (SSD)

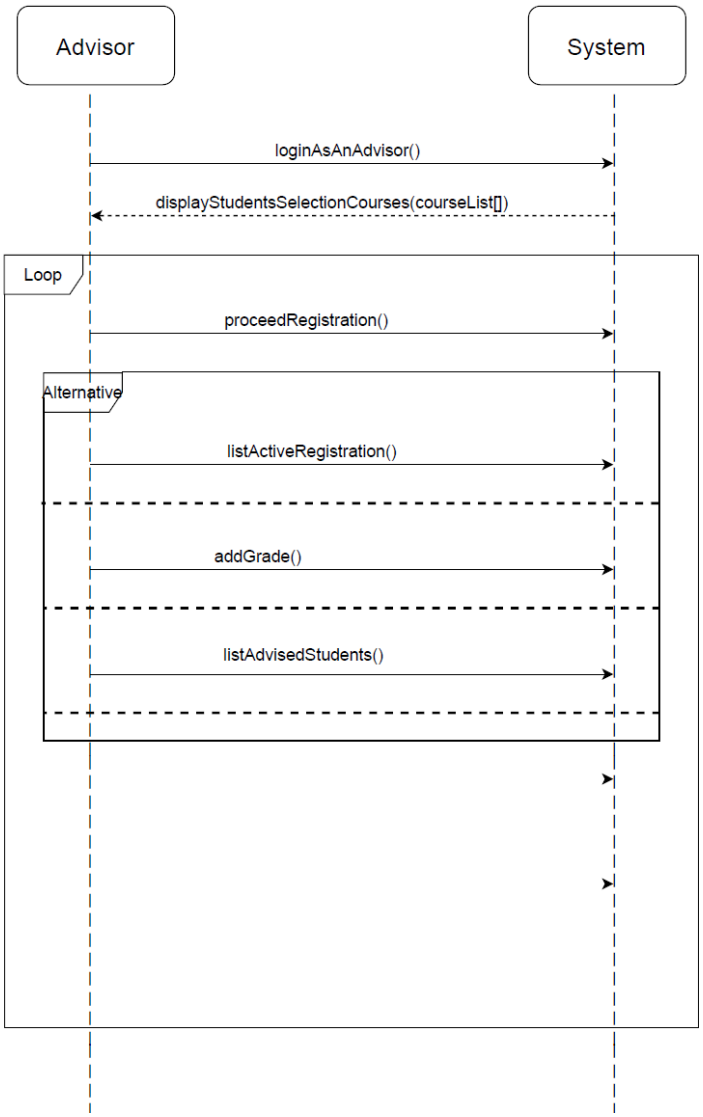
1) Actor: Student



2) Actor: Lecturer



3) Actor: Advisor



Glossary

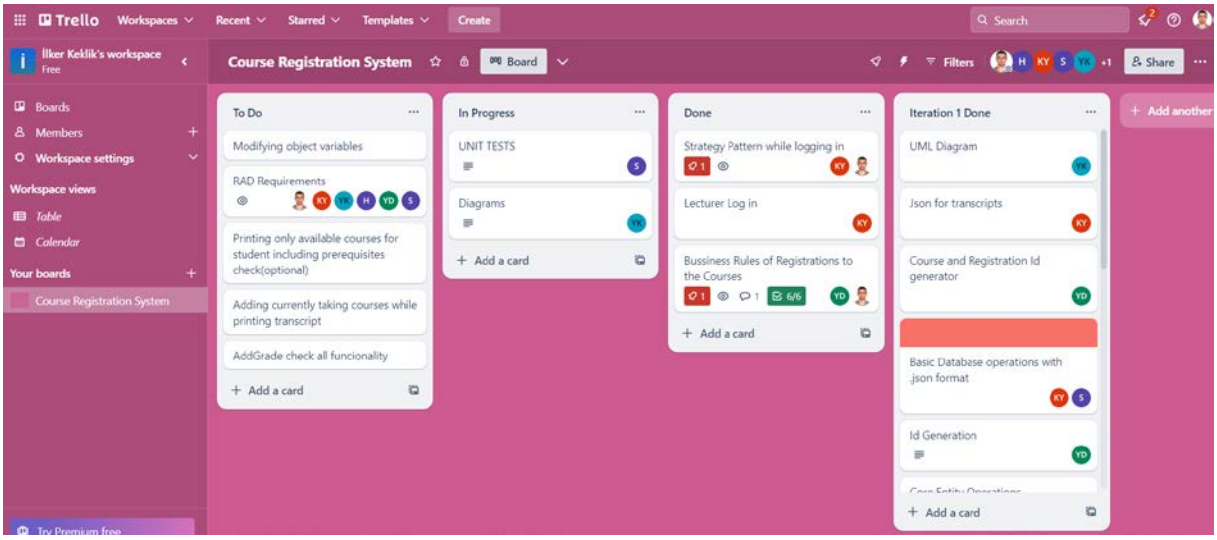
1. **Advisor:** An individual assigned to guide and approve course requests for students, assisting in academic planning.
2. **Course Registration System:** A simulated platform designed to mimic the process of registering for courses, involving students, lecturers, advisors, courses, and the Department.
3. **Prerequisite Courses:** Courses that must be successfully completed before a student can enroll in a specific course.
4. **Transcript:** A record of a student's academic history, including completed and failed courses, total credits, and letter grades.
5. **Criticality:** The importance of a requirement to the overall system functionality and user experience.
6. **Priority:** The ranking order of features or functionalities based on their importance.
7. **Risk:** Potential issues or challenges that may arise in meeting a specific requirement, along with strategies to mitigate them.
8. **Semester:** One half of an academic year.
9. **Credit:** A numerical representation of the value assigned to a course.
10. **Course**: It is a set of classes that focus on a certain topic given throughout the semester.
11. **JSON File**: A JSON file is a file that stores simple data structures and objects in JavaScript Object Notation (JSON) format, which is a standard data interchange format.
12. **Quota**: The maximum number of students who can enroll in a course.

Report

Iteration 1

First we all met and talked about the project. We considered about project requirements , entities that we need , relationships between them, business rules we have to consider etc. Also we created a board in Trello which is a project management tool allows us to share and store our issues. Then Kerim and İlker prepared a template of our UML class diagram. Then we talked and make decisions for the our domain to be able to recognize possible issues that we would be facing. Then, We create SSD, DCD, DSD, Domain Model and RAD documents. After that, we shared the workload. Decided which part of the project requirement will be implemented by which person.

Yusuf and İlker started to prepare the template form of the domain. Then Kerim starts to implement main ui. Yasin and Sarp implemented the course, course section classes and also its relationships and business rules like course prerequisities. İlker implemented the registration logic and Havva added the transcript and grade operations of the students. İlker implemented the core functions that will be used for some of the main operations. Kerim and Yusuf implemented the json serialization and deserialization functionality of our program. Also Yusuf created the IDGenerator that is used to create unique id's for different purposes. Kerim and İlker improved the main UI. Also Sarp created the unit test for functionalities of our domain classes. We made a little updates in the documents along the process.



Iteration 2

We had a meeting as group to talk about the changes we want to apply for the iteration 2 after the demo session of iteration 1. We decided to add more bussiness rules for the process of registration to courses. Also , we planned to use new design patterns. We decided to connect course sections to the system that includes quota of the courses, day and time information to use in overlap controls. Then, we updated the documents according to the decisions we made.

First, İlker and Kerim used strategy pattern to handle our login operations of students, advisor and lecturers. Havva and Yasin crated course section and adding new course sections to all existing courses. Yusuf added overlap check , registration check from upper class and gradiation project registration check. Also İlker added quota check for course sections when student tries to register to new course section and check for maximum credit for one semester for students. Sarp implemented new unit tests and also modified the old ones. We did a little updates in the documents when we faced with unexpected situations during the implementation.