FALL 2022

CSE 3063

OBJECT ORIENTED SOFTWARE DESIGN

TERM PROJECT

ITERATION - 3

Requirement Analysis Document (RAD)

Prepared by

Group 6

MARMARA UNIVERSITY
FACULTY OF ENGINEERING
COMPUTER ENGINEERING DEPARTMENT

TABLE OF CONTENTS

Before Begin:

This document was prepared using the Requirement Analysis Document that we prepared for iteration 2. It covers all changes, including additions, deletions, and rephrasing, in the latest document.

The changes in this document are represented by using highlight colors yellow, blue and red.

- Yellow Highlight ( ) stands for additions.
- Blue Highlight ( ) stands for modifying/ revising. (Especially for paragraphs)
- Red Highlight ( ) stands for deletions.

1. STAKEHOLDERS

- **Customer**
  - Murat Can GANİZ
- **Developers**
  - İsmail ÖKSÜZ
  - Emre SAĞIROĞLU
  - Serkan KORKUT
  - Onur ALKURT
  - Mustafa YANAR
  - Erdem PEHLİVANLAR
  - Ertan KARAOĞLU
  - Yasin ÇÖREKÇİ
  - Mertkan TURAN
  - Mehmet Akif GÜLMÜŞ
  - Yunus KAYA

2. VISION

Our simulation aims to educate students on the process of registering for courses at Marmara University's Computer Engineering Department by simulating the procedures and policies that they need to follow. The goal is to help students understand and successfully navigate the course registration process.

3. SCOPE

The scope of this project is to create a simulation of the process for registering for courses at Marmara University's Computer Engineering Department. The simulation will cover the procedures and guidelines that students need to follow in order to successfully register for courses, as well as the policies and guidelines of the department. The simulation is intended to be used as a learning tool for students and will not involve the development of an actual course registration system or its integration with existing university systems.

4. PROBLEM STATEMENTS

The goal of this project is to address issues that may arise in the student course registration process at Marmara University's Computer Engineering Department. One such issue is ensuring that students can only enroll in courses that meet certain requirements, such as having available spots, having completed prerequisites, and having a schedule that does not conflict with other courses. The simulation will help to ensure that students can properly register for courses by verifying that these conditions are met.

5. FUNCTIONAL REQUIREMENTS

- Students are created by using a json file for each semester.
- Advisors are added by using a json file.
- When students are created, an advisor is defined for each student.
- The software will use a json file to read the list of courses, types of courses, prerequisites, quota, and semester (1 to 8).
- When a student wishes to add a course to their calendar, the registration system makes sure there are no conflicts.
- The system must make sure that all prerequisites for a course are taken before adding it to a student.
- A student must be able to enroll in a course as long as there is availability in the course's quota.
- A student can take a maximum of 10 courses in a semester.
- To enroll in technical elective courses or graduation project, a student must have completed the required number of credits. The system must ensure that this requirement is met.

6. NON-FUNCTIONAL REQUIREMENTS

- Java programming language was used for this project.
- Python programming language was used for this project.
- Junit 5 was used to prepare Unit Tests.
- Unittest module for Python was used to prepare Unit Tests.
- Log4j was used to log the actions in a log file.
- Logging module for Python was used to log the actions in a log file.
- Draw.io was used to draw diagrams.
- We distributed tasks by a Kanban Board via Trello.
- Discord was used for group meetings that we made online.
- We write our documents using Google Docs.
- No Graphical User Interface was used in this project.

7. USE CASES

7.1. Main Scenario

Actor: Student

1- Student enters the course registration system.

2- Student adds the courses in the course registration system.

3- Student sends the chosen courses to the advisor for approval.

4- Course registration is completed.

5- The system shows log records.

6- Student logs out.

Extensions:

2.a) If the student's credit is less than the minimum required credits to take the course, the student cannot take the course.

7.2. Advisor Approbation Scenario

Actors: Advisor

1- Advisor checks collision.

2- Advisor checks course quota.

3- Advisor checks prerequisites courses.

4- Advisor checks if the student has chosen more than 10 lessons.

5- Advisor approves the selected courses and sends them to the student.

Extensions:

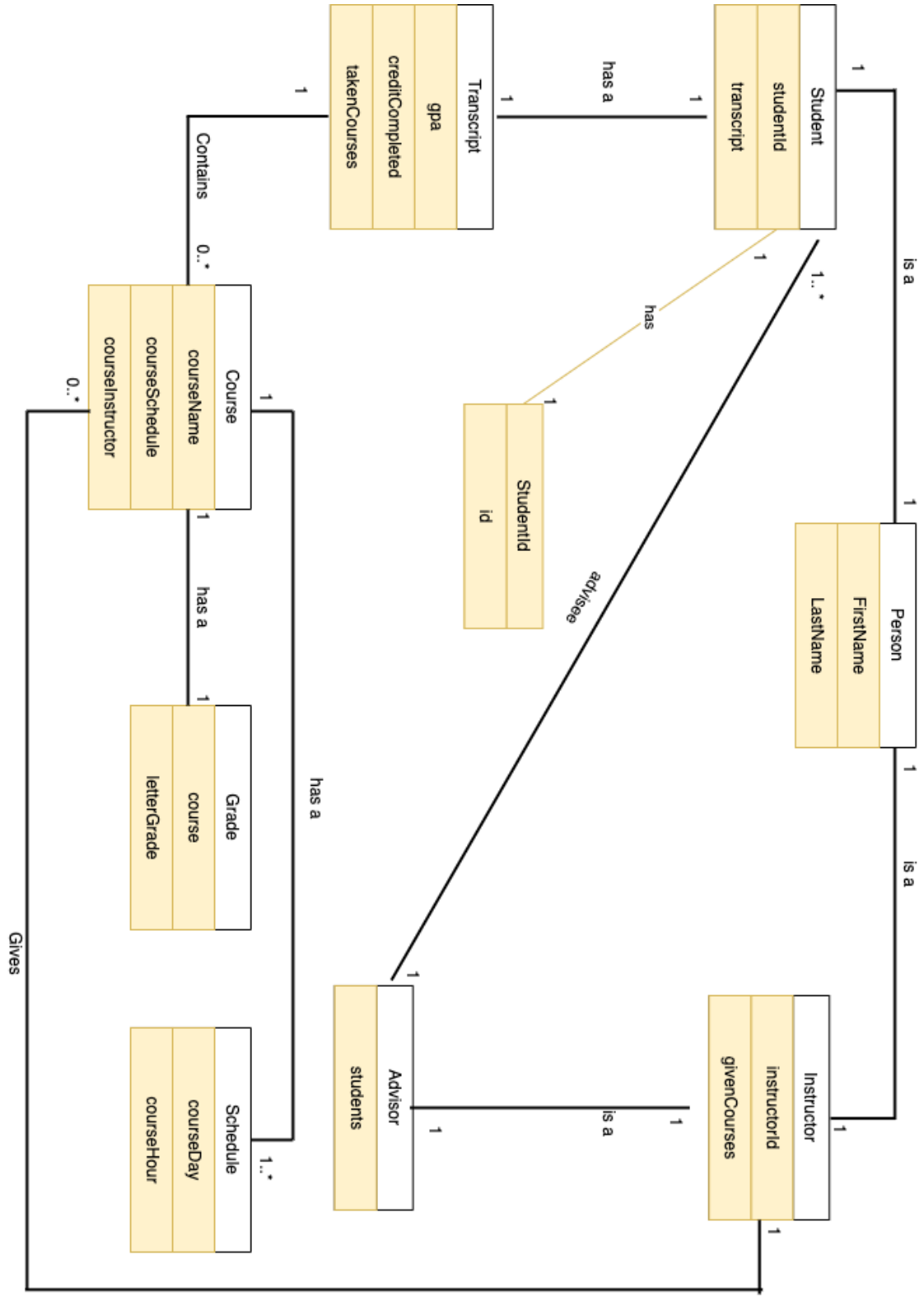
1.a) If the course that the student wants to take overlaps with another course, the advisor will not approve the course enrollment.

2.a) If the course quota is full, the advisor will not approve the course enrollment.

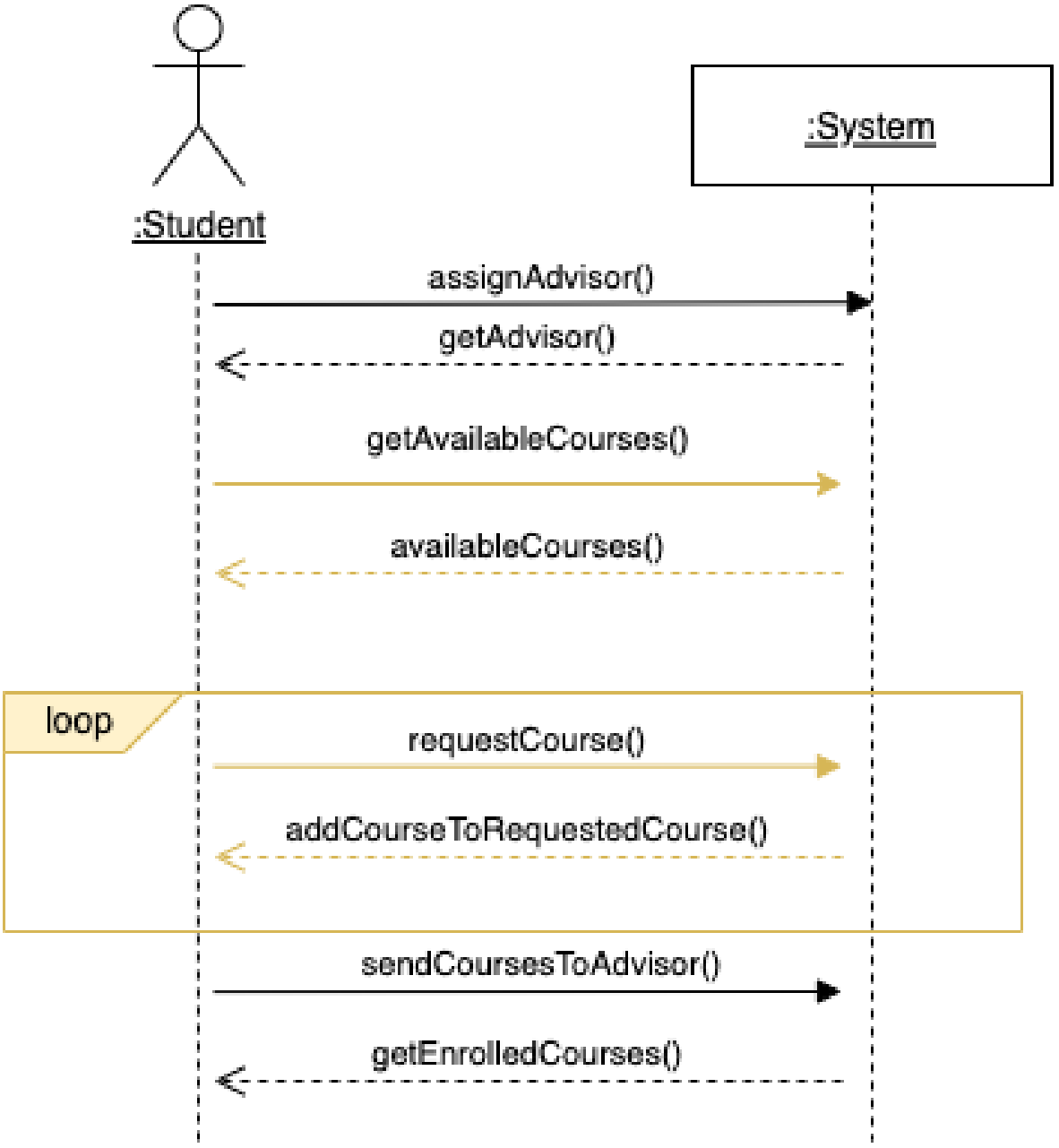
3.a) If the student has not successfully completed the prerequisite of the course he/she wants to register, he/she cannot take the course.

4.a) If the student has chosen more than 10 courses, the advisor will not approve the course selection.

8. DOMAIN MODEL



9. SYSTEM SEQUENCE DIAGRAM:



10. GLOSSARY

- **Advisor:** A person whose duty it is to offer guidance to students on enrolling in courses.
- **Course:** Lessons that students need to pass in order to graduate.
- **Course Grade:** Average of the students grades in the assignments for a specific course.
- **Credit:** The recognition for having taken a course at school or university.
- **Cumulative GPA:** The average of all semester GPAs.
- **Elective Course:** Courses that are optional to take. Students can choose the lessons if they are interested in the topics.
- **Faculty Elective Course:** Elective courses that have related topics about the department.
- **GPA:** Abbreviation for “Grade Point Average”. GPA is a number representing the average value of the grades earned in courses over time.
- **Grade:** Used to represent a student's level of achievement in a course or subject.
- **Graduation Project:** Courses that a student can enroll in after collecting at least 165 credits. Subclass of Mandatory Course.
- **ICreditRequirement:** An interface checks the required credits.
- **Instructor:** Lecturer of the courses.
- **Java:** Java is a class-based, object-oriented programming language that we use to develop that system.
- **Json File:** Json is a structure used to store data and move it between different platforms. JSON is text written in object notation format.
- **JUnit 5:** A unit testing framework.
- **Letter Grade:** A system used in schools to evaluate the performance of a student on a specific assignment or test.
- **Logging:** Is a method of recording events or actions that take place while a software program is running.
- **Log4j:** A Java-based logging utility.
- **Mandatory Course:** Courses that every student must take.
- **Non-Technical Elective Course:** Those courses' topics are not much related to the department according to the Technical Elective Courses.

- **Person:** Superclass of student and advisor with name, surname, email, and phone number.
- **Prerequisite:** Course or other requirement that a student must have taken prior to enrolling in a specific course or program.
- **Python:** A class-based, object-oriented programming language known for its simplicity, readability, and flexibility. It supports multiple programming paradigms, including object-oriented, imperative, and functional programming styles.
- **Quota:** The number or percentage of persons of a specified kind permitted to enroll in a course.
- **Registration:** The action or process of registering or of being registered.
- **Schedule:** A plan that lists the courses and other academic activities that a student is expected to attend.
- **Semester:** A division constituting half of the regular academic year.
- **Student:** The main character of the course registration system under the Person class.
- **Student ID:** Unique ID that students have.
- **Transcript:** A record of a student's academic courses, grades, and credits issued by their school.
- **Unit Test:** A test that evaluates the individual components of a larger system.