Requirements Analysis Diagram

Vision:

The main goal of this project is to simulate a course registration system. Students can send enrollment request to a course, system checks for quota and whether there are any prerequisite courses. On the other hand, advisor checks for other issues that may occur such as (time conflict (clash), the number of courses etc.) Nowadays, course registration systems are a huge easiness in students' lives. Having well organized course registration systems with decent functionalities is big advantage since there are many students. Taking all details, even the tiniest ones, into consideration and trying to level up performances of our system will make every student happier. Not facing problems, and going through stress while choosing and registering courses is a huge relief for students.

Problem Description:

In today's age, university student information, course information, teacher information and transcripts are very complex. Complexities can be experienced as these structures take up large areas. A course registration systems should be organized very well and pay attention to the smallest details in order to make it an easier and better experience for the students. Especially universities that have big number of student attending, and also new enrolled students, should have their systems managed and organized in best possible way

Functional Requirements:

- The system must create random students.
- The system must be able to read from a json file and write through to a json file.
- The system must check for course prerequisites, course quotas , etc.

- The system must allow advisor to check requirements and advisors can deny or allow a student to take certain course

Non-Functional Requirements:

Usability:

Outputs and system logs should have proper names and must be clearly understandable, easy for users' use.

Flexibility:

Whenever new courses, advisors, students are added, system should integrate them without any bugs.

Performance:

The system must do its tasks in a proper amount of time. No delays.

Reliability:

The source code should be tested and should not include any bugs.

Data Integrity:

All data should be stored in json files

Maintainability:

The source code must be easy to understand so if any bugs occur developer can detect with no effort. Also all possible errors should be logged in a file

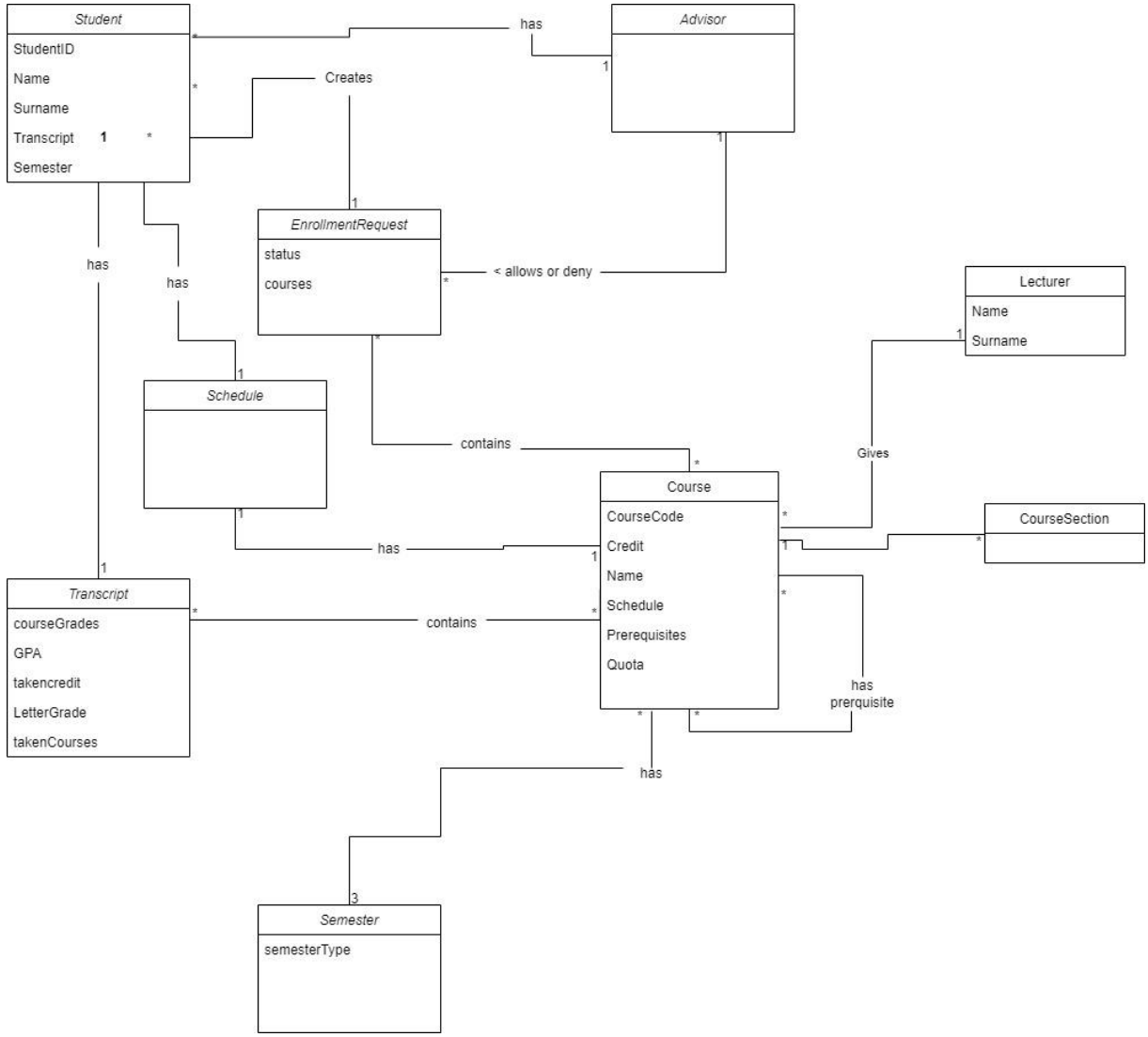
Use Cases:

Actors: Customer

- System will be launched in command line (terminal).

- Randomly created students will select randomly created courses.
- System checks for course quota or other requirements
- All students have randomly generated advisor
- Randomly created advisors check every course enrollment requests and generate an output(allow/deny)
- All operations are logged and count of every operation is produced as an output

Domain Model:



Glossary:

• Course: The course is what student registers through BYS, Information Management System.

• Prerequisite : something that is necessary, that needs to be accomplished in order to continue, e.g. you need to pass programming 1 before you can take programming 2, that is from where prerequisite course term comes.

• Technical electives : are courses (generally in computer science, mathematics, engineering, or business) oriented toward the design or use of computers.

• Non-technical electives (NTEs): are courses that complement your Engineering courses, usually taken and approved in upper years, generally you can choose between categories of Social Science and Arts and Humanities courses

• Credit: Every course has a certain number of credits, a student needs to finish designated numb er of credits to pass the semester or to be able to take some other courses in different academic year

• Student: Student is a user who is interacting with and within the system.

• Advisor: Advisor is a person who is assigned to students in order to check whether they can take specific courses, regarding on quota, credits, prerequisites etc. Furthermore he/she helps students with other problems regarding university as well.

• RegistrationManager: This is the manager which helps the advisor to control the courses and students' informa1on.

• CourseManager: This is the manager where the courses are being managed.

• FileManager: This manager writes transcripts and problems to files, as its names suggest, it manages the files.

• StudentManager: In this manager students are being created randomly.

• CourseSection: This helps to assign dates to courses.

• Transcript: The place where students' course information is kept.

• Python : A programming language(with which the third iteration of the project is done).

__str__ - method in Python that is equivalent to toString() method in Java

__init__ - The __init__ method is the Python equivalent of the C++ constructor in an object-oriented approach. The __init__ function is called every time an object is created from a class

Self - self represents the instance of the class. By using the “self” we can access the attributes and methods of the class in python. It binds the attributes with the given arguments. The reason you need to use self. is because Python does not use the @ syntax to refer to instance attributes

Private(__) - Private methods are those methods that should neither be accessed outside the class nor by any base class. In Python, there is no existence of Private methods that cannot be accessed except inside a class. However, to define a private method prefix the member name with the double underscore “__”

• Main: This represents the main class of design.

• Functional Requirement: A requirement that the system must be able to fulfill and to do.

• Non-Functional Requirement: A requirement that specifies how the system should do it giving it required instruc1ons.

• BYS : Marmara University Information Management System.

• JSON : JSON is a text-based data format that is the lightweight alternative to XML widely used on the Web for data interchange.

Project Scope

Without the Course Registration System, managing and maintaining the details of the students and courses is a tedious job for any organization. So, this registration system will store all the details of the students including the background information, and course related information. The main goal of this project is to simulate a course registration system. Students can send enrollment request to a course, system checks for quota and whether there are any prerequisite courses. On the other hand, advisor checks for other issues that may occur such as (4me conflict (clash), the number of courses etc.)
For the first iteration of your Java project, we should create a small but functional course registration simulation. We must have several classes such as Course, CourseSection, Staff, Lecturer, Advisor, Student, Grade, Registration, Transcript and CourseRegistrationSystem, CourseRegistrationSimulation (Controller pattern) etc .

Majority of our classes must have responsibilities, to be more exact meaningful operations (methods) for implementing business logic. An important reminder is to avoid seeing classes as data holders only. There should be at least 10 students, 5 advisors, lecturers and at least 10 different courses at least 3 of them having prerequisities. For the second iteration there should be at least 400 students. About 50

students for each of the 8 semesters. All the presequities should be implemented.

What contribution would the project make?

This is an era of information technology where automation of each and every activity is gaining importance. The project would bring many easinesses to both students and advisors. It would be easier for everybody who is using or kind of participating in the system. There are many advantages of this system, some of them are: time saving, increased efficiency, better overview of general situation regarding students and courses, more accuracy.