MARMARA UNIVERSITY

Department of Computer Science Engineering

CSE3063 Object Oriented Software Design Course

Requirement Analysis Document (RAD)

150118014 - Can Gök
150118024 - Oruç Berat Turan
150118025 - Berkin Polat
150118044 - Özgür Taylan

About Project

University is the place for everyone to learn, experiment and discover. Every university around the world has more than one focus, department or research topics. In this great environment, Marmara University is one of the most crowded and known universities in Turkey. The Computer Science and Engineering department of Marmara University is in need of a system to find the problems and bottlenecks of the registration process of courses defined in the curriculum. A meaningful and well working course registration system is required for continuous education.

We aim to develop a new simulation for the course registration system to find problems. This system will include a new and well designed user-friendly interface for increasing efficiency, sub algorithms to detect problems and bottlenecks, efficient algorithms to apply department rules to simulation. After the development process, the course registration system will find the openings and bottlenecks of the current procedures and luckily the department will be able to fix them.

Requirement Specification Vision (Purpose)

This documentation describes both functions and requirements of the course registration simulation. The purpose in general is to find the problems of the course registration system with logging and printing useful outputs which eventually can be used for fixing the problems of the system.

The users will have an opportunity to register in a course or run a batch for multiple students for registering courses. Furthermore, this document is for both stakeholders and developers.

Problem Statement

Course registration simulation can be used iteratively in random states with numerous batches to find different problems and dead ends in the system. To explain it furthermore, the simulation is a problem finder and logger for the course registration system.

Scope

Simulation is a problem finder for the course registration process so that these found problems can be solved. The goal of the simulation is finding the problems, dead-ends and bottlenecks of the system and giving a proper output to make them solutible. In order to make it possible, simulation will work on different circumstances and will save different states. After many iterations, the problems will be recorded by the system and output will be ready for the solution process.

System Constraints

The simulation will run in every device that has a java runtime environment installed.

Stakeholders

Murat Can Ganiz: Customer
Can Gök
Oruç Berat Turan
Berkin Polat
Özgür Taylan

Glossary

User: Person who starts the simulation.

Student: Sub component of the simulation that tries to register for the courses in order to complete the eight semester education.

Advisor: The one who applies the department rules and procedures on the course registration process.

Course: Lecture that is given to a limited or unlimited group of students in accordance to the scheduled times and predefined semesters.

Transcript: The document that proves the success grade of the student in a specific course.

JSON File: Described as JavaScript Object Notation. An easy to access storage file which will be used as simulation input.

Functional Systems:

1. Randomly chosen students should try to register randomly but only for unfinished courses.
2. The list of the available courses that is in harmony with department rules can be listed for the student.
3. The system will get the information available as a json file.
4. The simulation must print logs to console and into a log file.
5. There has to be a batch process for a student to register for a course.
6. A course can be taken by many students if quota is not exceeded.
7. Department curriculum can be listed to the students including different lecture types.
8. Registration process of each student must be printed as a json output file containing a transcript before the registration, a transcript after the registration and a summary of the problems.
9. General statistics about these problems for the given semester have to be printed.
10. Students who have problems must be listed and their outputs have to be reachable.

Non-Functional Systems

1. Usability:
    a. Simulation needs to be user friendly.
2. Accessibility:
    a. Data should be easily accessible.
3. Supportability:
    a. Simulation should work on each platform that runs JRE.
    b. Simulation has to be platform independent.
4. Performance:
    a. Simulation has to be time efficient on input & output operations.
    b. Simulation should be able to support large amounts of data to be processed.
    c. Simulation needs to be memory efficient.
    d. Simulation should produce outputs & logs in a reasonable time amount.
5. İmplementation
    a. Project will be implemented using Java 8+.
    b. JSON format will be used for every input file.
    c. PHP will be used to get the department's curriculum from the website.

Use Cases:

Use Case 1: Apply a Course
Actors: Student, System

1. Student enters to the platform.
2. System lists the courses for the Student.
3. Student selects a course from courses listed. (Iterative Process)
4. System shows all of the courses selected.
5. Student starts registration process by sending courses selected to Advisor.
6. System shows a success/error message.

Alternative: Selecting Failure

3a. Student can't select a course that he/she didn't pass from course's prerequisite course(s).
Student can't add the selected course to submission list.

3b. Student can't select a lab or elective course that has no availabe quota.
Student will try to select a section with available quota for the course by returning to step 3 again.

3c. Student can't find an empty-quota section for the course.
Student will select another course by returning to step 3 again.

Use Case 2: Register Student to a Course
Actors: Advisor, System

1. Advisor asks for the list of the advisee student courses.
2. System lists the advisee student's courses.
3. Advisor checks the proper requirements via System.
4. *Advisor approves the Student to register.
5. System enrolls the Student to the courses.

Alternative: Approval Failure

4a. Advisor will not approve submissions with hour-collisioned courses.
Student will select the course which is repeating or lower semester courses.

4b. Advisor will not approve if the student didn't complete enough credits.
Student can't get approval of the selected course.
4c. Advisor will not approve if the student already exceeded the limit of TE courses.
Student can't get approval of the selected course.

4d. Advisor will not approve if the student selects an FTE course and he/she is not graduating this semester.
Student can't get approval of the selected course.