Student Registration System

Problem Statement

In this project, we need to write software for simulating the registration of university courses. It will be similar to the system that our department has. Proper business logic should be implemented according to the registration rules and regulations for taking courses.

Vision

The ultimate goal of this project is to build a student registration system that will be simulated like the real university student registration system with the given constraints. We aim to improve the efficiency and accuracy and enhance the overall student experience.

Scope

There should be many courses with a prerequisite tree. The course registration process will start with choosing the student. In order to test our software we will generate students randomly. For storing students and their transcripts, we will use JSON files. According to the students' information, we will simulate a course registration process. We will either choose randomly or by the JSON file name. After that, we will list the courses that the student can take by business logic. Each step must be shown one by one on the user interface. We should give feedback prompts to the student for some types of situations. For example, for some types of courses, there is a quote for registration and this should be handled in the process.

Functional Requirements

1- Human class information

- **a.** Humans will have a first name.
- **b.** Humans will optionally have a middle name.
- **c.** Humans will have a last name.
- **d.** Humans will have a method for getting their full names.

2- Faculty Member class information

- **a.** Faculty members will have a department

3- Student class information

- **a.** Students will have a student ID.
- **b.** Students will have a grade.
- **c.** Students will have an advisor.
- **d.** Students will have a transcript.
- **e.** Students will have an enrolled section list.

4- Department class information

- **a.** Departments will have an instance.
- **b.** Departments will have a code.
- **c.** Departments will have courses.
- **d.** Departments will have students.
- **e.** Departments will have lecturers.

f. Departments will have a current season value

g. Departments will have assistants

5- CourseRecord class information

a. CourseRecord will have a course.

b. CourseRecord will have a letter grade.

c. CourseRecord will have a score.

d. CourseRecord will have a grade.

e. CourseRecord will have pass or fail information.

6- Transcript class information

a. Transcripts will have records of the taken courses.

7- Course class information

a. Courses will have a code.

b. Courses will have credits.

c. Courses will have ECTS.

d. Courses will have a quota.

e. Courses will have theoretical hours.

f. Courses will have applied hours.

g. Courses will have lecturers.

h. Courses will have assistants.

i. Courses will have classes.

j. Courses will have the first season to be taken in information.

k. Courses will have the first year to be taken in information.

l. Courses will have a name.

m. Courses will have a section list

8- TechnicalElective class information

a. Technical elective courses will have required credits.

9- MandatoryCourse class information

a. Mandatory courses will have a number to be taken each season information.

b. Mandatory courses will have prerequisites.

10- Section class information

a. Sections will have a number of weekly class hours.

b. Sections will have a course.

c. Sections will have a class schedule.

d. Sections will have an instructor.

e. Sections will have a student list.

f. Sections will have class days.

g. Sections will have class hours.

11- EngineeringProject class information

a. Engineering projects will have required credits.

12- Registration information

a. Check all the requirements to enroll in a course. If all the conditions are met add the courses to the curriculum of the student. For the other cases, prompt the student about the requirements they failed to meet.

b. The system should check whether there is a conflict in the courses chosen by the student. If more than two course hours overlap, the system should not allow students to register. If the overlapping class time is less than 2 hours, the advisor has to decide.

Non-Functional Requirements

1- The application will be written in Java.

2- Inputs will be taken from a file in JSON format.

3- Outputs will be written to files in JSON format.

4- Registration phases will be logged to the screen and a log file.

Business Rules

1- To register for a course, that course shouldn't collide with the previous courses that the student added.

2- To register for a course, the student must pass the prerequisites of that course.

3- To register for a TE course and engineering project, the student must complete the required credits.

4- To register for a course, the student must take that type of course according to the maximum choosable number for that season.

Use Case: Student Registration

Register for a full season at the university

Actors: Student, System, Advisor

1- The student will select the courses that he/she wants.

2- The system will check for some of the requirements.

3- The student will send the selected courses to his/her advisor.

4- The advisor will check the courses

5- The advisor will approve the courses.

6- The courses will be added to the student's curriculum.

Alternative 2: Failing at system control

2a-) At step 2, there is at least 2 hours collision between some of the courses that the student wants to take. The system will not let the student to take that course.

2b-) At step 2, the student did not pass the prerequisite of that course. The system will not let the student to take that course.

2c-) At step 2, the quota for the elective that the student wants to take is already full. The system will not let the student to take that course.

2d-) At step 2, the number of credits that the student completed doesn't meet the requirement for the TE and the engineering project. The system will not let the student to take that course.

2e-) At step 2, the student added more than 2 TE in the fall season. The system will not let the student to take that course.

Alternative 4: Failing at advisor check

4a-) At step4, the advisor does not approve the student's syllabus when they notice that two important courses overlap. The importance of the courses is indicated according to the priority evaluation.

4b-) At step 4, the student did not pass the prerequisite of that course. The system will prompt an error regarding the prerequisite condition is not met and the student should change the courses he/she wants to take.

4c-) At step 4, the student tries to take a course that is overlapping with another course with higher priority.

Use Case: Advisor Support

Check the courses that the student sends to take

Actors: Advisor, System, Student,

1- The advisor will receive the courses that the student wants to take from the system.

2- The advisor will check for some of the requirements.

3- The advisor will approve the selected courses.

Alternative 2: Failing at checking the courses

2a-) At step 2, the student tries to take a course that is overlapping at least with another course with higher priority.

2b-) At step 2, the student did not pass the prerequisite of that course. The system will prompt an error regarding the prerequisite condition is not met and the student should change the courses he/she wants to take.

Use Case: System Check

Check the courses that the student tries to take

Actors: System, Student, Advisor,

1- The system will check the courses that the student wants to take.

2- The system will give permission to send the selected courses to the advisor.

Alternative 1: Failing at checking the courses

1a-) At step 1, the student tries to take courses that has at least 2 hours collision between each other. The system will prompt an error regarding the collision and the student must change the courses he/she wants to take.

1b-) At step 1, the student did not pass the prerequisite of the course he/she wants to take. The system will not let the student to take that course.

1c-) At step 1, the quota for the elective that the student wants to take is already full. The system will not let the student take the course.

1d-) At step 1, the number of credits that the student completed doesn't meet the requirement for the TE and the engineering project. The system will not let the student take the course.

1e-) At step 1, the student added more than the max choosable number of elective course in the fall season. The system will not let the student take the course.

Iteration Plan

First Iteration: Analyze the problem, draw the UML and diagrams and complete the fundamental tasks.

1- Complete the requirement analysis of the course registration system.

2- Draw the UML according to the requirement analysis.

3- Write the fundamental classes that are drawn in the UML.

4- Implement the course-taking condition functionalities.

5- Implement the registration system.

6- Implement the registration approval system

7- Implement the logging system

Second Iteration: did some refactoring to the first iteration. Update UML and DSD according to new version.

1- Implement registration system.

2- Fix errors in the syllabus. Identify the courses that should be selected in case of conflict.

3- Junit should be used for unit testing

4- The advisor should not only be authorized to approve the syllabus. Specify scenarios that it won't approve of.

Third Iteration: Port the project to Python. Fix some errors and mistakes that we had in the previous iterations. Update DCD, UML and DSD according to new version.

1- Port the project to Python and translate the syntax to make the project work.

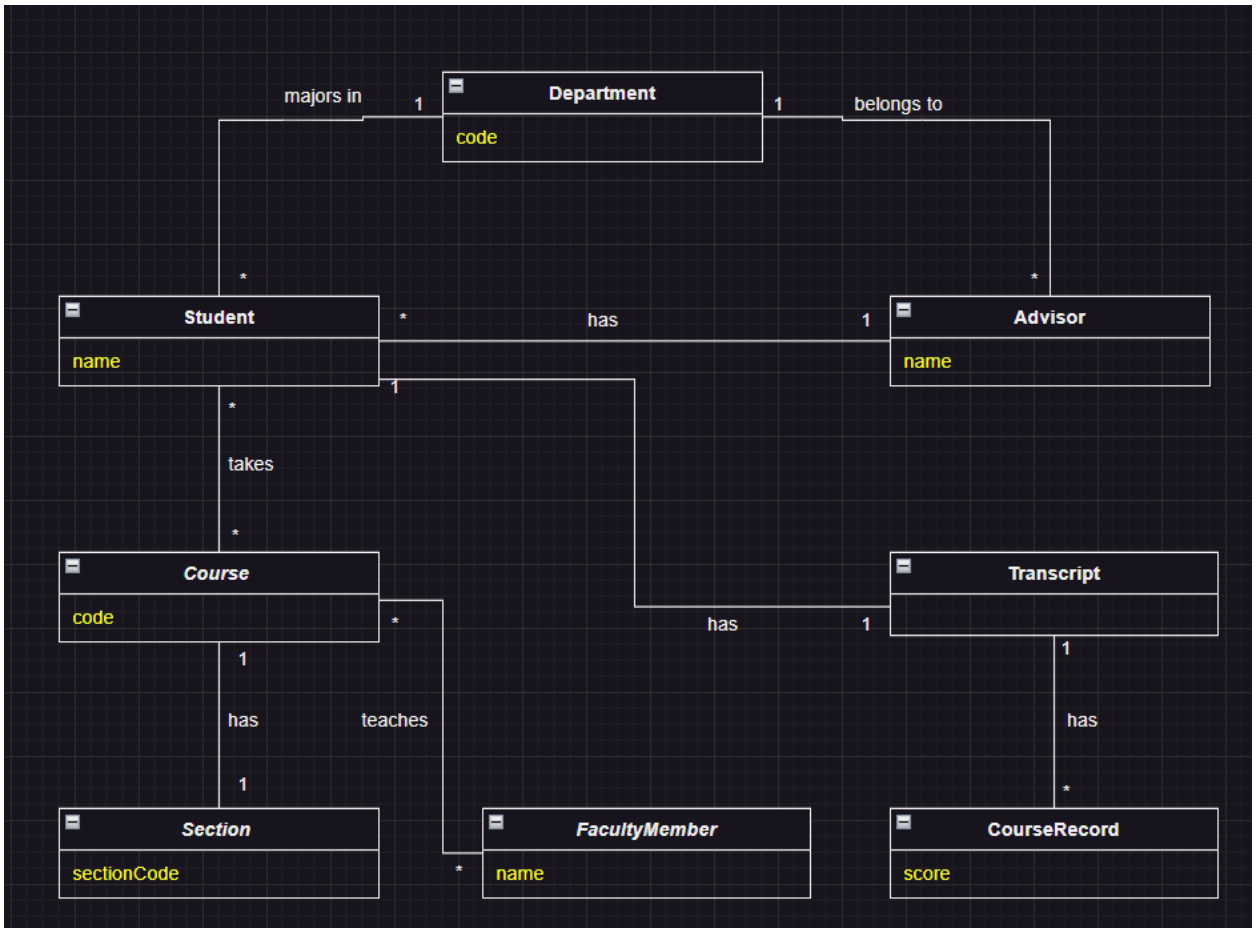
2- Remove unnecessary static and final values and handle them in a proper way.

3- Add necessary try-catch blocks

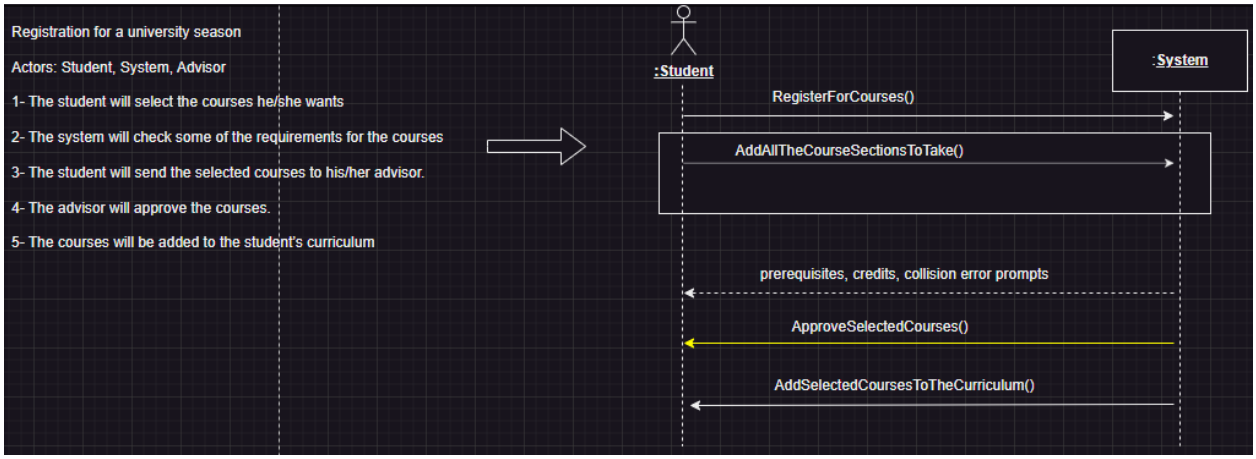
4- Format the problem statement, vision and scope in RAD.

5- Update and fix the UML, domain model, DCD and DSD

Domain Model



System Sequence Diagram



Glossary

- Simulation: The bootstrap class where the main method is located.
- Department: The university program that the student is studying.
- Student: The main actor in the university registration system.
- Lecturer: The person who teaches the courses.
- Assistant: The person who teaches the labs.
- Advisor: The person who is a lecturer and reviews the registration request of the student.
- Season: Academic terms like Fall, Spring and Summer
- Elective Course: Electives are courses a student takes by choice.
- TE: Technical Elective
- FTE: Faculty Technical Elective
- NTE: Non-Technical Elective
- Mandatory Course: Compulsory courses for the student
- Tuple: Key-value pair
- Section: Course sections means simultaneous instances of a course offering with a unique student enrollment
- Course Record: represents the courses the student has taken so far
- Lab Section: Lab classes are usually associated with science classes. Students will take the material they've learned in a lecture and attempt to apply what they've learned in various lab experiments
- Course Section: When the capacity of compulsory courses is full, the newly opened section is called for students.
- Season: term(fall, spring, summer)
- Grade: Indicates the student's year in university
- Registration System: The system that checks whether there is a conflict between the courses chosen by the students during the course selection.