Student Registration System

Vision

In this project, we will build a registration system for the university. Mainly it serves students and of course Instructors (advisors). With the help of this system, we are aiming for automation and preventing this whole registration process from manual human errors so, before public usage, we need to simulate first.

Description

In this project, we will design a registration system for our school. Student Registration System (SRS) as a kind of management information system can not only record the information for student registration each term quickly and efficiently but also do statistics on the student's basic information, and registration information and give results of the analysis. The subject's background is analyzed in the project first, and the technologies used in the development are introduced. Then the implementation of the system is given in detail. The system's upgrade and expansion have been prospected at the end of the project.

Scope

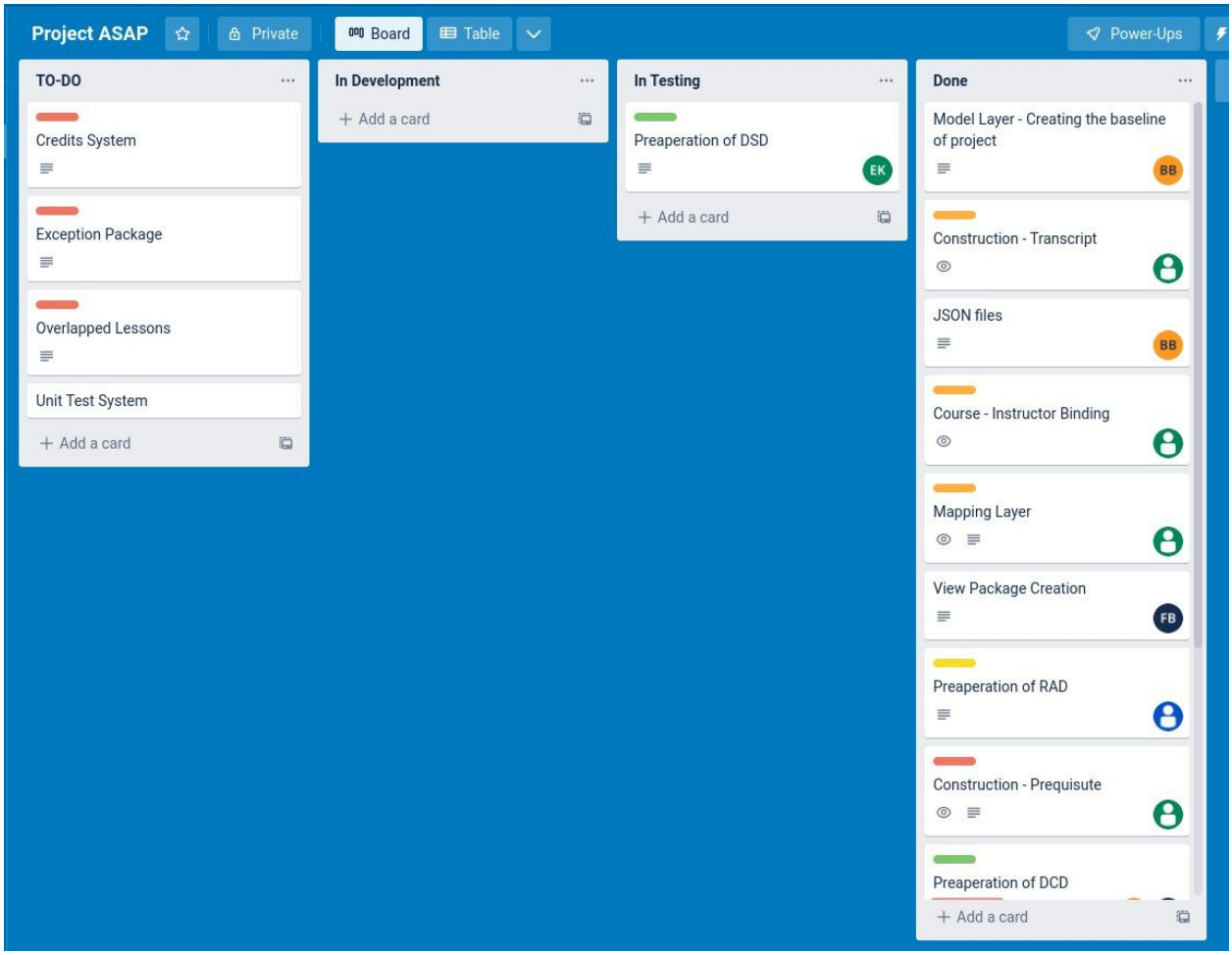
The school registration system will be a terminal-based simulation accessed through an IDE. The system will work as a simulation, in this simulation Students will be created through a JSON file. The advisor will work as same as the Student and we will implement it in Python.

How We Work

When we got the project first of all we assembled a team of course. Mainly we found and hire people by their references (friends of our friends etc.). We selected one team leader and one product owner for this project. Our team lead is Umut Emre Önder and our product owner is Ali Eren Şen.

We have chosen to work async and for communication, we used Discord channels and Whatsapp. We only huddled up when needed; frankly, we managed the finished iteration without too many huddles.

For issue tracking we used Trello, actually, we want to try a different and new product such as Linear and Clickup but because of the other projects and quizzes, we couldn’t find enough time to learn a new tool, and that's all.



Requirements

Functional Requirements

1. Knowing student information
    - Students will have semester information.
    - Students will have an advisor.
    - Students will have passed classes.
    - Students will have failed classes.
    - Students will have classes from next semester.
    - Students will have total completed credits.
    - Students will have transcripts.
2. Knowing Instructor information
    - Instructors will have classes which they will give.
    - Instructors will have students.
    - Instructors can be advisors.
3. Knowing course information
    - Courses will have credits.

- Courses will have pre-requisite courses.
- Courses will have an available semester
- Courses will have available hours.
- Courses will have required completed credits.

4. Course enrollment
   - Checking if students can take the selected courses.
     - If students can take the course, the advisor will approve it, and the course will be added to the student's schedule.
     - If students cannot take the course, inform the student with error messages.

Non-Functional Requirements

1. Each process of registration should be logged.
2. Each process of registration should be prompted to CLI.
3. The system should take input as separate JSON files.
4. The program should be implemented with Python.
5. The output will be mapped.

Business Rules

1. To register for a new course, students must pass it prerequisite courses.
2. Register to Technical Elective courses must have more than blocker credits.
3. Register for a course, and that course lecture hour can't collide with another course's lecture hour.
4. Register a course, that course's quota must not be exceeded.
5. If the student passed the course can't take it again

Use Case

Enrolling in a course

- A student selects their courses according to the semester's curriculum but also can take failed classes again and can take courses from the following semesters.
- Student sends selected courses to their advisor.
- Advisor will check the student's selected course list.
- The advisor can approve students' courses.
  - If the selected course's quota is full, the student cannot take that course.
  - Selected courses' lecture hours collide with each other. Advisor will approve the course that repeating or from the lower semester.
  - Students haven't passed the pre-requisite of the selected course.

    - Students haven't completed enough credits for the selected course.
    - Student course list contains technical elective courses, although he/she already took 2 TEs in the fall semester.
    - Student course list contains technical elective courses, although he/she already took 3 TEs in the Spring semester.
    - Student course list contains faculty technical elective courses in the FALL semester, and he/she is not graduating this semester.

!!! If any of the problems occur which listed above student can't take the class and of course, the advisor won't approve that course, informs the student, and drops that course from the student's course list.

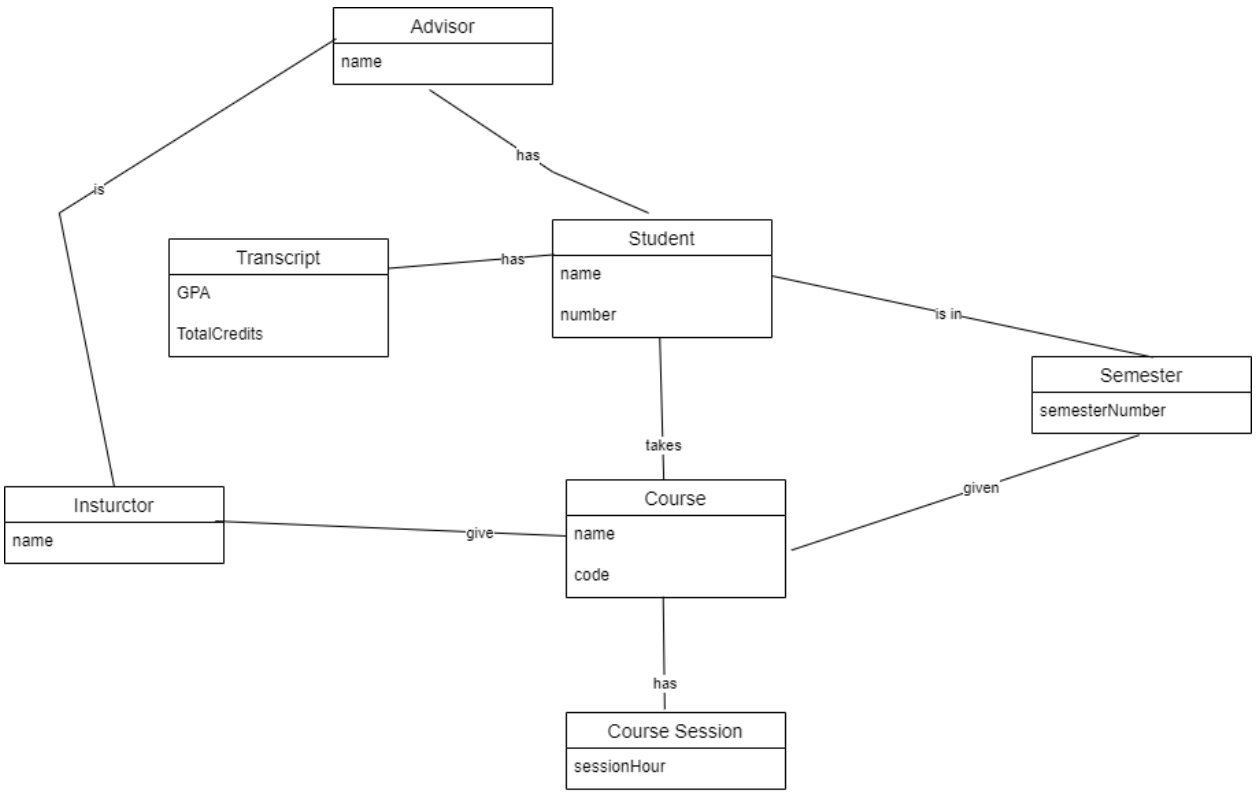
- Approved courses added to students' time schedules.

Iteration Plan

Third Iteration: On the third iteration, we will convert the program from Java to Python.

1. Check the requirement analysis and do the changes we need.
2. Creating the base structure of the project in Python.
3. Mapping the classes with the same objectives.
4. Implementing a log tracking system.
5. Writing unit tests according to that.

Domain Model

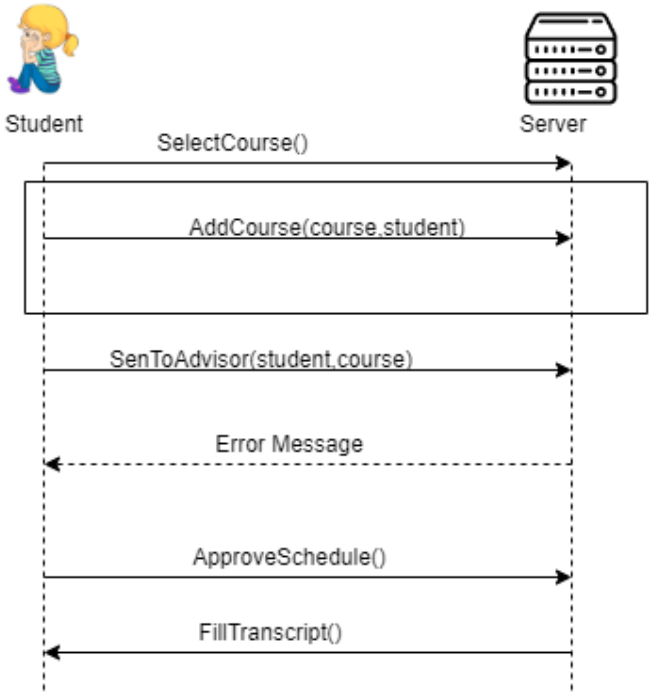


System Sequence Diagram

Enrolling to a course

Actors: Student, Lecturer(Advisor), Computer, System

1. Students select courses according to the curriculum of the semester and their list. (Failed classes etc.)
2. Students send the list to their own advisors.
3. The advisor checks the list, according to the rulebook.
4. The advisor approves or declines the list.
5. Courses approved are added to the student's time schedule and their transcripts.



Glossary

- Advisor: Actor who approves or disapproves student's course selection list.
- Course: Lessons students must have passed to graduate.
- Credits: Weight impression of the lecture.
- Curriculum: Overall content of the course.
- DTO: Data Transfer Object, it is used for implementing much clear data transfer.
- FTE Courses: Faculty Technical Elective courses.
- Instructor: A person who gives courses at universities, can be an advisor too.
- JSON File: Standard file format and data interchange format that uses human-readable text to store and transmit data objects consisting of attribute-value pairs and arrays.
- NTE Course: Non-Technical Elective courses.
- Pre-requisite courses: Courses required to be passed in order to get the connected course.
- Python: High-level general-purpose programming language.
- Schedule: Students' courses in a weekly plan.
- Session: Lecture hours available for the same lesson.
- Semester: A half-year term in a school.
- Student: A person formally engaged in learning, especially one enrolled in a school or college.
- TE Course: Technical Elective courses.
- Transcript: Lecture records of the student.
- UE Course: University Elective courses.

- Unit Test: Basic system tests for our simulation.