CSE3063 OBJECT-ORIENTED SOFTWARE DESIGN

TERM PROJECT

REQUIREMENT ANALYSIS FOR ITERATION 1

Name Surname	Student ID
Yunus Emre Gezici (Project Manager)	150121066
Ali Arda Fıstıkçı	150121006
Muhammed Hasan Erzincanlı	150121031
Alperen Burak Koçyiğit	150121035
Cihat Emre Vardiş	150121037
Enzel Ebrar Albayrak	150123841

Group 1

Members

1. Project Description

This program is an enhanced course registration system for Computer Science Engineering (CSE). The system is designed to make the course registration process easier while following department rules and policies. With a user login systme, users will be able gain access to use system functionalities based on their role: students register for courses, advisors approve registrations. This program is built with basic functionalities. It is a core system of future features. User will interact with program in command-line interface.

This core system has basic features like login feature, basic course registration with prerequisite, limit students to a maximum number of courses and advisor approval. Users will be access to system with their username and password with login interface. Students will register for maximum number of 5 courses. Students will register for courses only if they have completed the prerequisite courses. Advisors is a key role in this program. They will gain access to course registration system like students. They will approve the registration. If they don't approve, a student cannot be registered for courses. Overall, this program is an efficient, organized, and secure course registration system.

2. Glossary

Definitions:

In Iteration 1, the core system architecture is being built, with the main objective of implementing basic functionalities such as login, course registration with prerequisite checking, and the creation of foundational classes. The system will be tested with sample data, including at least 10 students, 2 advisors, and 10 courses. Data will be stored and managed through JSON files, which will serve as the foundation for later iterations, where more advanced features will be added.

Key Terms:

1. Authentication:
The process by which users (Students and Advisors) log into the system using their assigned usernames and passwords.

2. Course Registration:
A feature that allows students to register for courses according to the department's rules and regulations. In Iteration 1, students will be able to register for courses, and the system will check for prerequisites.

3. Course:
A specific academic course offered by the department. Each course has a set of prerequisites and may be taken in a particular academic year.

4. Student:
A person enrolled in the department who can register for courses, view their schedules, and track their academic progress. In Iteration 1, each student will be limited to registering for a maximum of 5 courses.

5. Advisor:
A faculty member who provides academic guidance to students. Advisors can view student progress and assist in course registration, though they do not register for courses themselves.

6. CourseRegistrationSystem:
The main controller class for managing the course registration process. This class manages the registration process, including course enrollment, prerequisite checking, and registration status notifications.

7. Pre-requisite Checking:
A system functionality that ensures a student meets the necessary requirements (such as completing prerequisite courses) before registering for a particular course.

8. JSON Files:
The method of data storage used in Iteration 1. Student data, course registrations, and other system configurations are stored in JSON files.

3. **Functional Requirements** and **Non-Functional Requirements**

Functional Requirements

User Login (Authentication):

The system should allow users (students, advisors, and department schedulers) to log in with their username and password.

Course Registration:

Students should be able to register for courses in accordance with department rules.

Prerequisite courses should be checked and rules should be applied during registration.

Conflict and Capacity Management:

The system should check time conflicts and capacity limits of courses.

When capacity is full, students should be added to the waiting list and notified if there is a vacancy.

Data Persistence:

Data should be stored in JSON files; optionally, it should be supported by a database such as SQLite in the next stage.

Non-Functional Requirements

Usability:

It should clearly notify the user of errors and provide correction opportunities.

Performance:

It should be ensured that users log in and course registration processes are carried out quickly and without interruption.

Condition and capacity checks should be made in real time during course registration.

Security:

User login information should be encrypted and stored securely.

Variables should not be randomly changed.

Protection should be provided against operations that the user does not have authorization in the system.

Scalability:

The system should be expandable for new user roles (Department Head, Admin, Student Affairs) that may be added in the future.

It should be able to respond to the increase in the number of users or the need to store more data.

Ease of Maintenance:

The code should be highly modular and the responsibilities of each class should be clearly defined.

Portability:

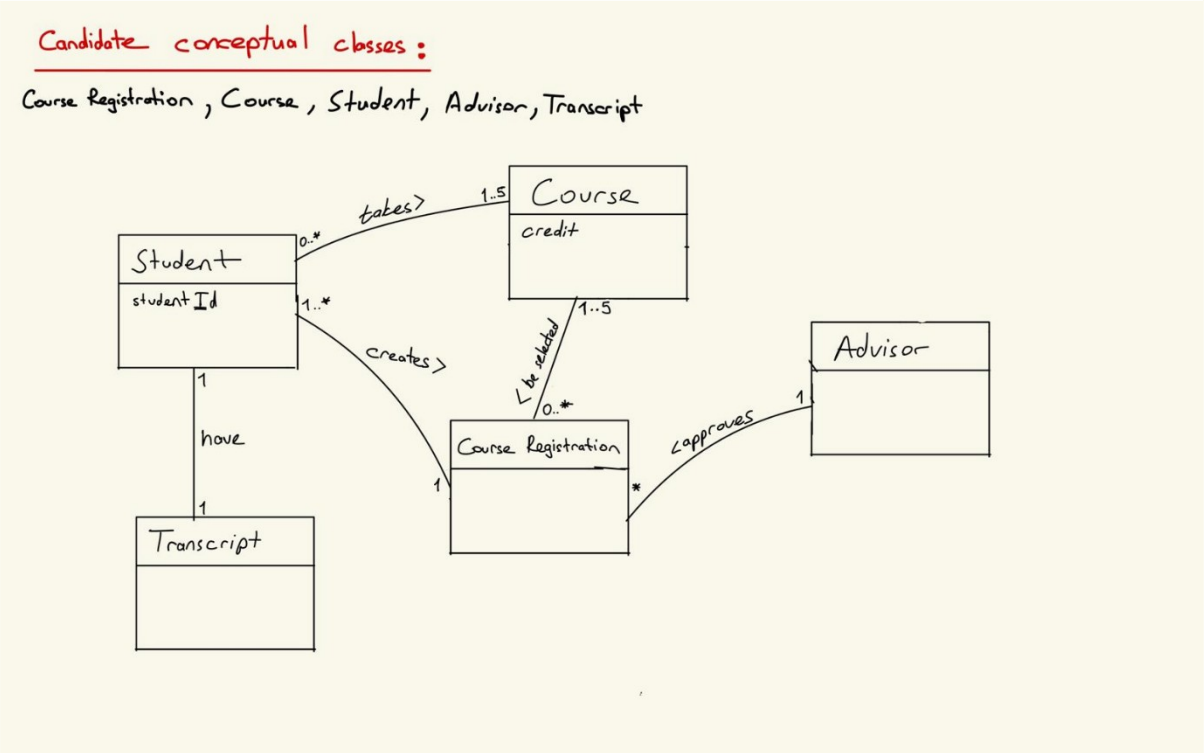
The system should be able to run in a platform-independent environment such as a command-line interface or Python.

It should be possible to transfer Java code to Python without any problems (Iteration 3).

Reliability and Accuracy:

The system is expected to maintain accuracy in student registration data and reliably perform conflict and capacity checking during course registration.

4.**Domain Model**



5.Use Cases

Use Case: Student assigns to a course

1. Student writes his identification number and password correctly.

1a. If student enters wrong information, program returns log in screen and prints an error message.

2. Program shows student's courses that are available to register.

3. Student chooses a course and send to advisor for approval.

4. Student exists from the system.

Use Case: Advisor approves courses of a student

1. Advisor writes his identification number and password correctly.

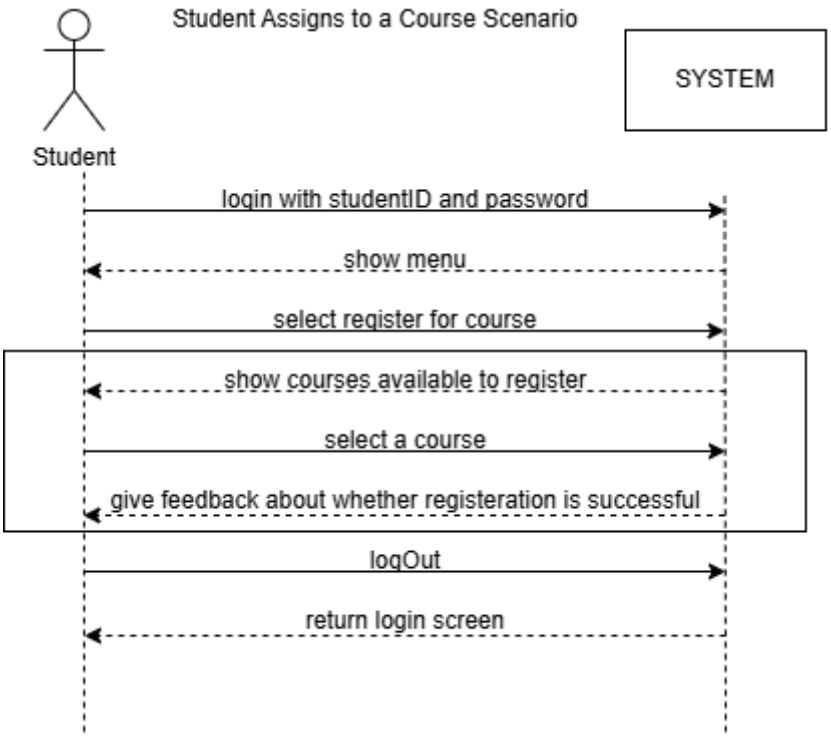
1a. If advisor enters wrong information, program returns log in screen and prints an error message

2. Program shows the list of students that are advised from this advisor.

3. Advisor selects a student.
4. Program shows the list of courses that are waited for approval of that student.
5. Advisor decides if he/she approves or not.
6. Program registers that student for those courses.
7. Advisor turns back and can choose another student.
8. If the advisor does not choose any student exists from the system.

6.System Sequence Diagram

Student Assigns to a Course Scenario



Advisor Approves Course Registerations Scenario