# CSE3063 – Requirement Analysis Document

## Project#2 – Iteration#1

**Group 1**

| **Name** | **-** | **ID** |
|---|---|---|
| Bahadır Alacan | - | 150118042 |
| Bekir Nazmi Görkem | - | 150118017 |
| Berk Kırtay | - | 150118043 |
| Burak Çağlayan | - | 150118027 |
| Erkam Karaca | - | 150118021 |
| Hasan Fatih Başar | - | 150118015 |
| Mehmet Mücahit Özen | - | 150118031 |
| Rasim Sadıkoğlu | - | 150118009 |

## Requirement Analysis Specifications

**Courses input:** We need a JSON file which is composed of all the courses throughout eight semesters. These courses may also have prerequisite courses. System will read this JSON file to get course information.

**Student transcripts:** Each student will have their own records file. Those files will have student information, course information (failed, passed, and currently taken classes), GPA and credits information of the student.

**Professors:** Each course is taught by a professor. professors will have their basic information. such as the courses they teach etc. Some professors can be advisors so that they can control course registrations.

**Simulation:** We will generate random students to simulate system. Students can have different numbers of failed and passed courses. System will control course registrations that made by the students. It will reject or approve them based on transcript information of the students. For example, let's say there is a course which has a prerequisite course, and a random student failed this prerequisite course. If this student attempts to register to this course. then system will not allow student to take the course.

## Use Cases:

Our customer will see a random simulation cycle in system. According to customer's desires, we will build and change our software. In an example run system will do:

Actors: Student, Management System, Advisor

1. Student opens the management system.
2. Student sees available courses in his/her screen.
3. Student adds courses.
4. Student reviews chosen courses in his/her screen.
5. Student sends courses to the advisor approval.
6. Advisor approves.
7. Course registration is completed by system.
8. Student reviews his/her grades.

## Alternatives

**5a.** Some courses may be dropped due to some restrictions. student can return the 3rd step.

**6a.** Advisor may drop some courses or change the dates of courses. then student can return to the 3rd step.

**Functional Requirements**

- System must display available courses.
- Student can add courses to his/her course list
- Student can send course list to the advisor approval.
- System must keep track of limits (quota, prerequisite, etc.) of courses.
- Advisor can deny or approve student's course(s).
- Lecturer can assign notes to students. System must display notes (numeric and letter) of the students.

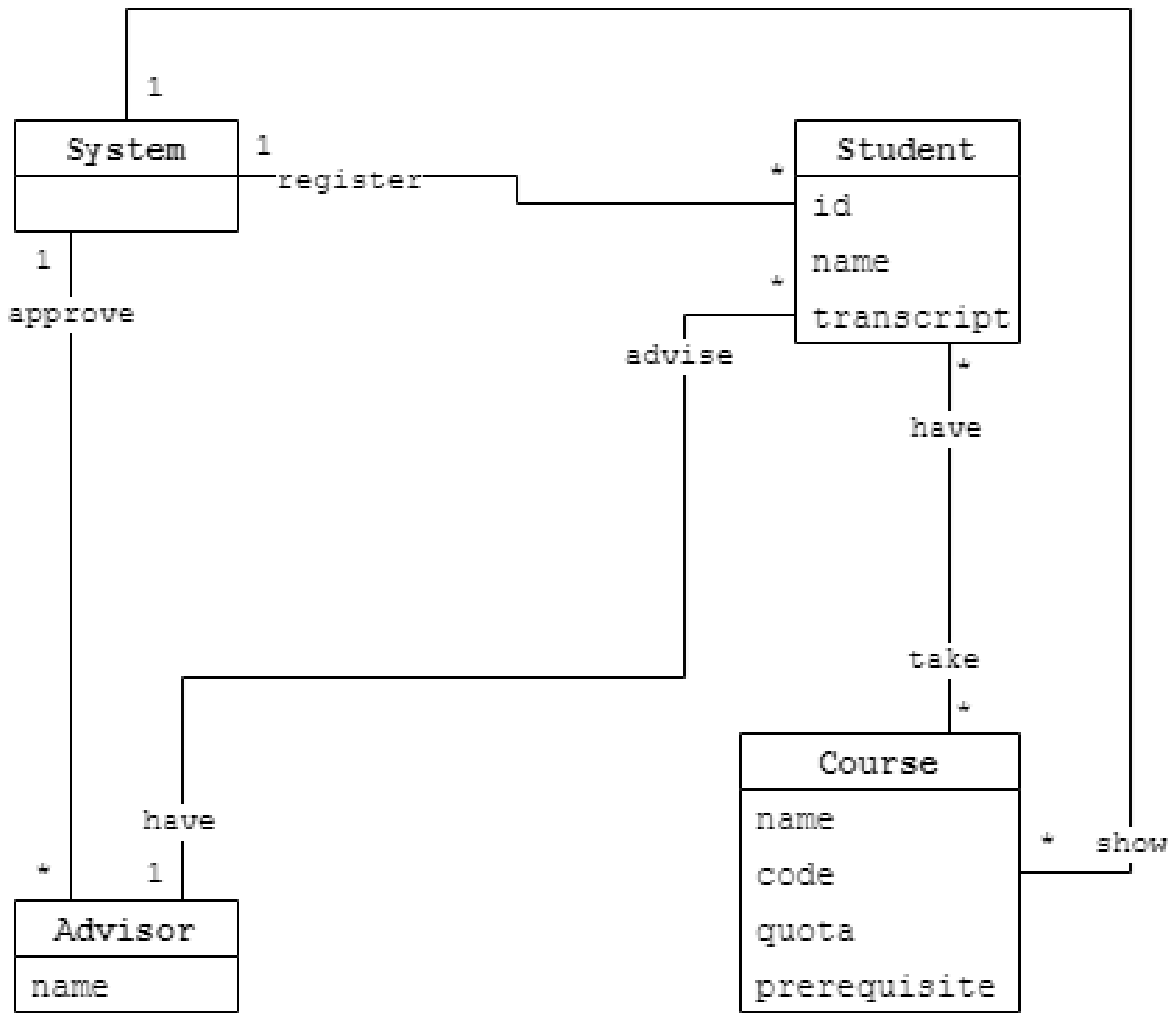
**Non-Functional Requirements**

- Parameters and data should be read from JSON.
- System must be secure.
- System must be consistent for all students and all courses.
- System must be always responsive,
- System must run on different platforms.
- System must be up almost always.
- System must be open for extensions.
- System must be easy to maintain.

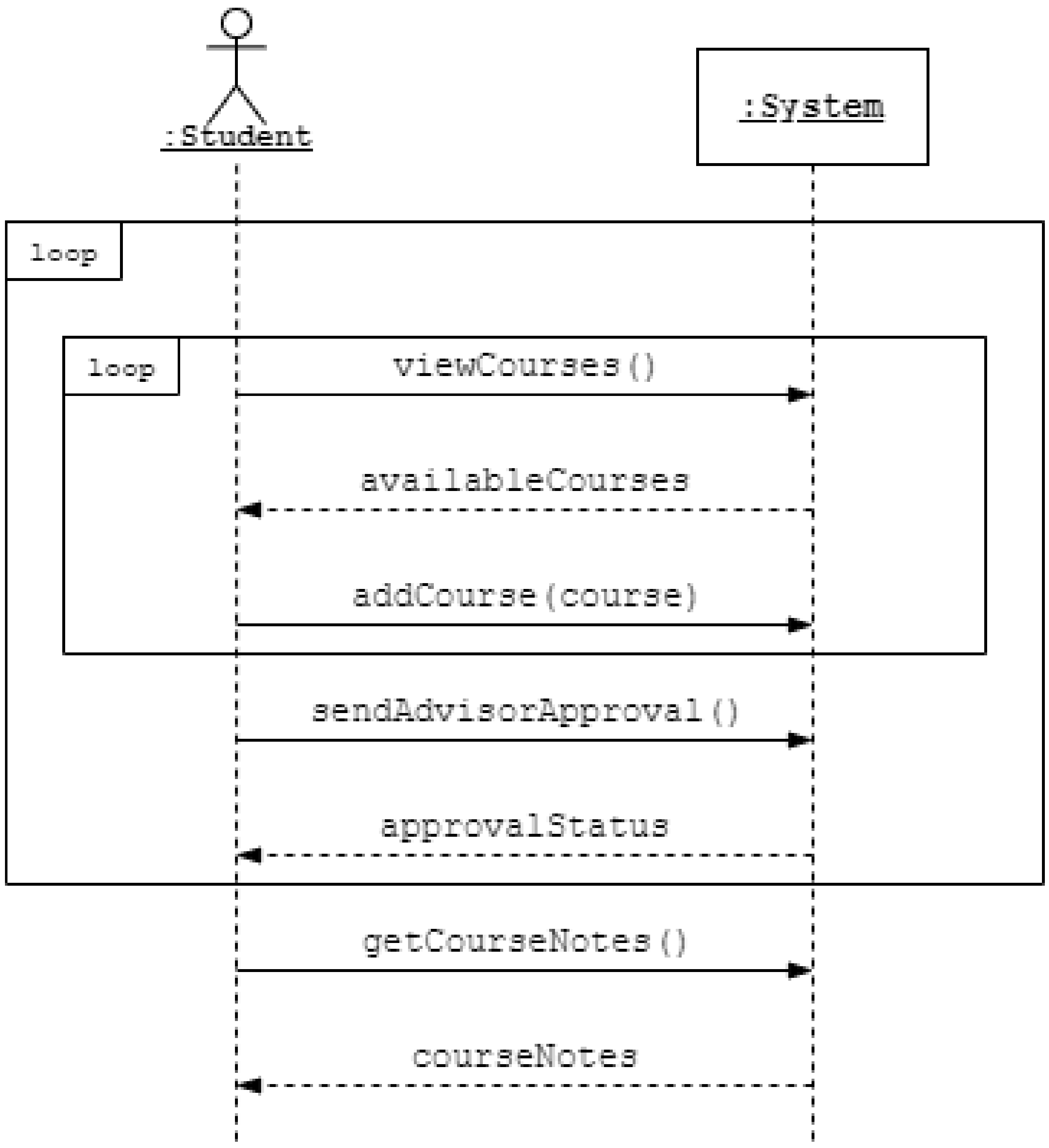
## Glossary

- **Student:** The person who attempts to register to the available courses.
- **Management System:** Checks for the availability status of the courses that student attempts to register.
- **Advisor:** Checks if student is eligible to register to a course, such as students completed credits, collision, limit etc.
- **Simulation:** The area our program starts and process.
- **Course:** University courses for the students.
- **Transcript:** Detailed student course info.
- **Person Generator:** Random student and advisor generator for simulation.
- **Data IO Handler:** Responsible for IO operations for student and course data.
- **Logger:** To log and print necessary info and warnings.
- **Unit Test:** Responsible for testing every atomic functionality of the program.

## Domain Class Diagram



## System Sequence Diagram


:Student
:System
loop
loop
viewCourses()
availableCourses
addCourse(course)
sendAdvisorApproval()
approvalStatus
getCourseNotes()
courseNotes