CSE3036 Object Oriented Software Design Project

Iteration 2 RAD

Group 1

150121043 Bora Duman
150121051 Arda Öztürk
150120076 Furkan Gökgöz
150121032 Mehmet Toprak Balıkçı
150120022 Tolga Fehmioğlu
150121538 Muhammed Enes Gökdeniz
150121002 Enes Torluoğlu

1. Editor's Notes

This document has been edited for the second iteration of the project. Newly added sections of text will be shown by a yellow highlight, except for this very note, and changes in images will be shown in yellow or red.

1.1 Executive Summary

This project aims to create a Course Registration System using Java, designed to facilitate the registration process for students and advisors in the department. The system will require users to authenticate using usernames and passwords and will adhere to the department's existing rules for course registration.

The architecture consists of multiple classes, including Course, CourseSection, Staff, Lecturer, Advisor, Student, Grade, Registration, and Transcript. These classes will include methods that go beyond basic data storage, implementing the various functionalities required for a complete registration system. Object-oriented principles such as inheritance, polymorphism, and aggregation are incorporated into the system's design.

Data will be stored and retrieved from JSON files. The first iteration will support at least 10 third-year students, 2 advisors, a number of lecturers, and a minimum of 10 courses, some of which will have prerequisites. Interaction with the system will be via a command-line interface.

For the second iteration of development, a messaging system will be added to be used between the advisor and student which a message can be about whatever, whenever. Also, each advisor will have an assistant now, which the assistants will be teaching the laboratory sessions of their corresponding advisor's course.

2.1 Glossary of Important Concepts

- ☐ **Authentication**: The process by which the system validates the identity of a user through a username and password.
- ☐ **Advisor**: A staff member designated to guide and approve students' course selections within the academic department.
- ☐ **Student**: An individual enrolled in the academic department who can register for courses through the system.
- ☐ **Course**: A subject of study offered within the department that students can enroll in.
- ☐ **Course Section**: A specific class offering of a course, which may vary by time, day, or instructor.
- ☐ **Transcript**: A formal record of a student's academic performance, detailing the courses taken, grades received, and other academic activities.
- ☐ **Command-Line Interface (CLI)**: The text-based user interface through which users will interact with the system.
- ☐ **Role-Specific Access**: A system feature that displays options and functionalities based on the authenticated user's role (e.g., Student, Advisor).
- ☐ **Message:** A message that is sent between the advisor and student. It can be sent anytime about anything.

- ☐ **Assistant:** As the name suggests, an assistant of an advisor. They will be teaching the lab sessions of courses.

3.1 List of Functional and Non-Functional Requirements

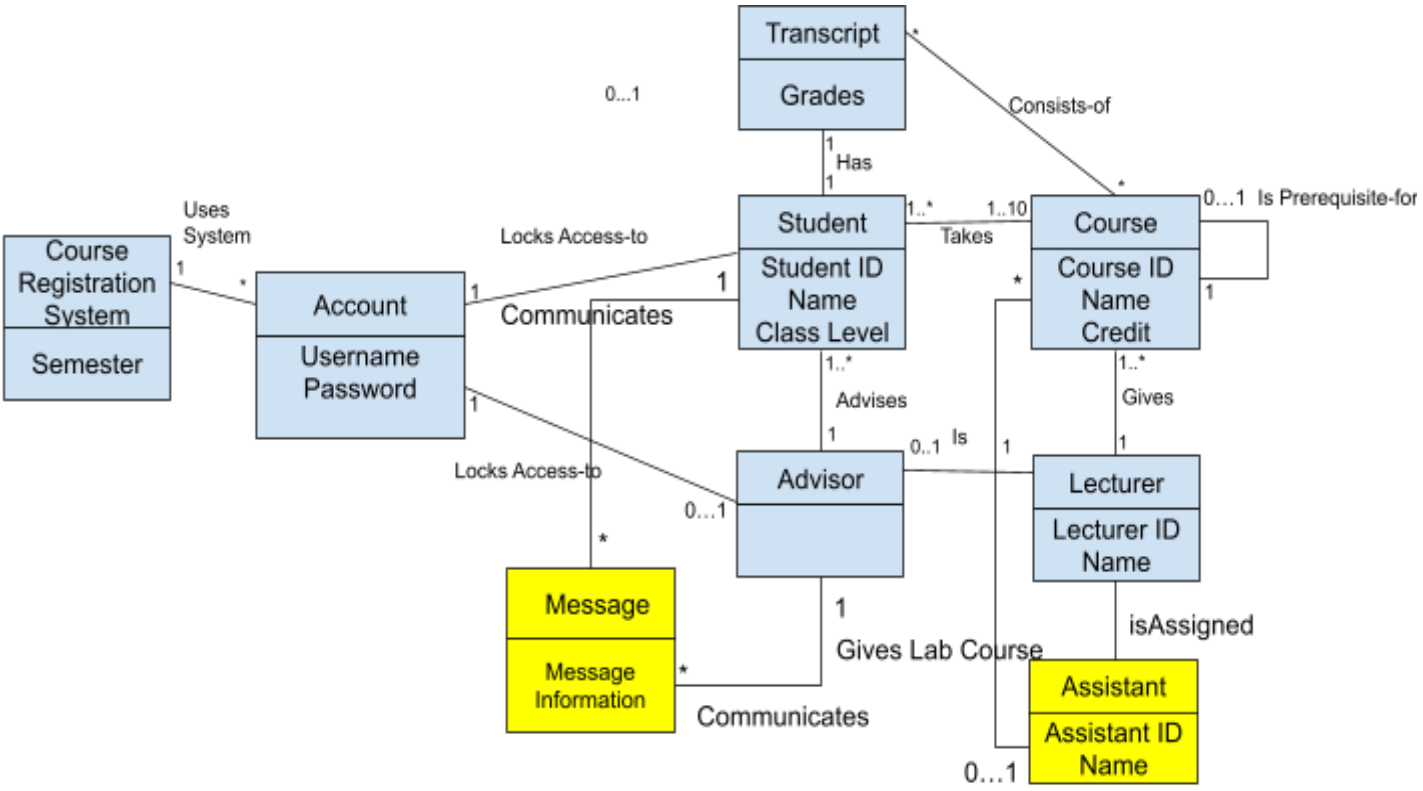
Functional Requirements:

1. **User Authentication:** Verify the identity of the user based on the username and password.
2. **Course Registration:** Allow students to register for available courses.
3. **Advisor Approval:** Enable advisors to approve or reject course selections made by students.
4. **Transcript Update:** Update a student's transcript after each semester.
5. **Data Persistence:** Store and retrieve user and course data from JSON files.
6. **Role-Specific Access:** Display options based on the role of the authenticated user.
7. **Message Feature:** A message is sent between the advisor and student.

Non-Functional Requirements:

1. **Performance:** The system should respond to user input within 2 seconds.
2. **Scalability:** The architecture should allow for the addition of more user roles in the future.
3. **Usability:** The command-line interface should be intuitive.
4. **Security:** Passwords should be invisible while being typed logging in.
5. **Portability:** The system should be operable on both Windows and Linux operating systems.

4. Domain Model



5. Use Cases

Use Case UC1: Registration to Course

Scope: Course Registration System

Level: user goal

Primary Actor: Student

Stakeholders And Interests:

Student: Wants to log-in to the system and enroll in courses.

Advisor: Want to see students' course enrollment request and validate them

Course Registration System: Wants to check information of the people trying to log in. Wants to save enrollments to file system correctly. Wants to accurately pull required information from file system to provide them to user.

Pre-Conditions: Student is registered in the system.

Success Guarantee: Student enrolled in the course. System saved enrollment.

Main Success Scenario:

1. Student opens the courses registration system log in page.
2. Student enters credentials.
3. Course registration system checks credential data and confirms log-in.
4. Student checks received messages.
5. Student sends messages to advisor.
6. Student enters course registration panel.
7. Student chooses courses from relevant courses.
8. System saves wanted courses to be checked by advisor later.
9. Student logs out the system.

Extensions:

3a. Invalid Credentials:

1. The system shows error message and require the student to re-enter their credentials.

This step is repeated until the correct credentials are entered.

5a. Course hour overlaps with another course:

1. Student can’t add the course.

2. Student can drop the other course at that section.

Special Requirements:

-A computer to enter to the course registration system.

Primary Actor: Advisor

Main Success Scenario:

1. Advisor opens the courses registration system login page.
2. Advisor enters credentials.
3. Course registration system checks credential data and confirms log-in.
4. Advisor enters course registration panel.
5. Advisor checks courses by chosen students.
6. Advisor approves courses by chosen students in a certain period of time .
7. Advisor logs out the system.

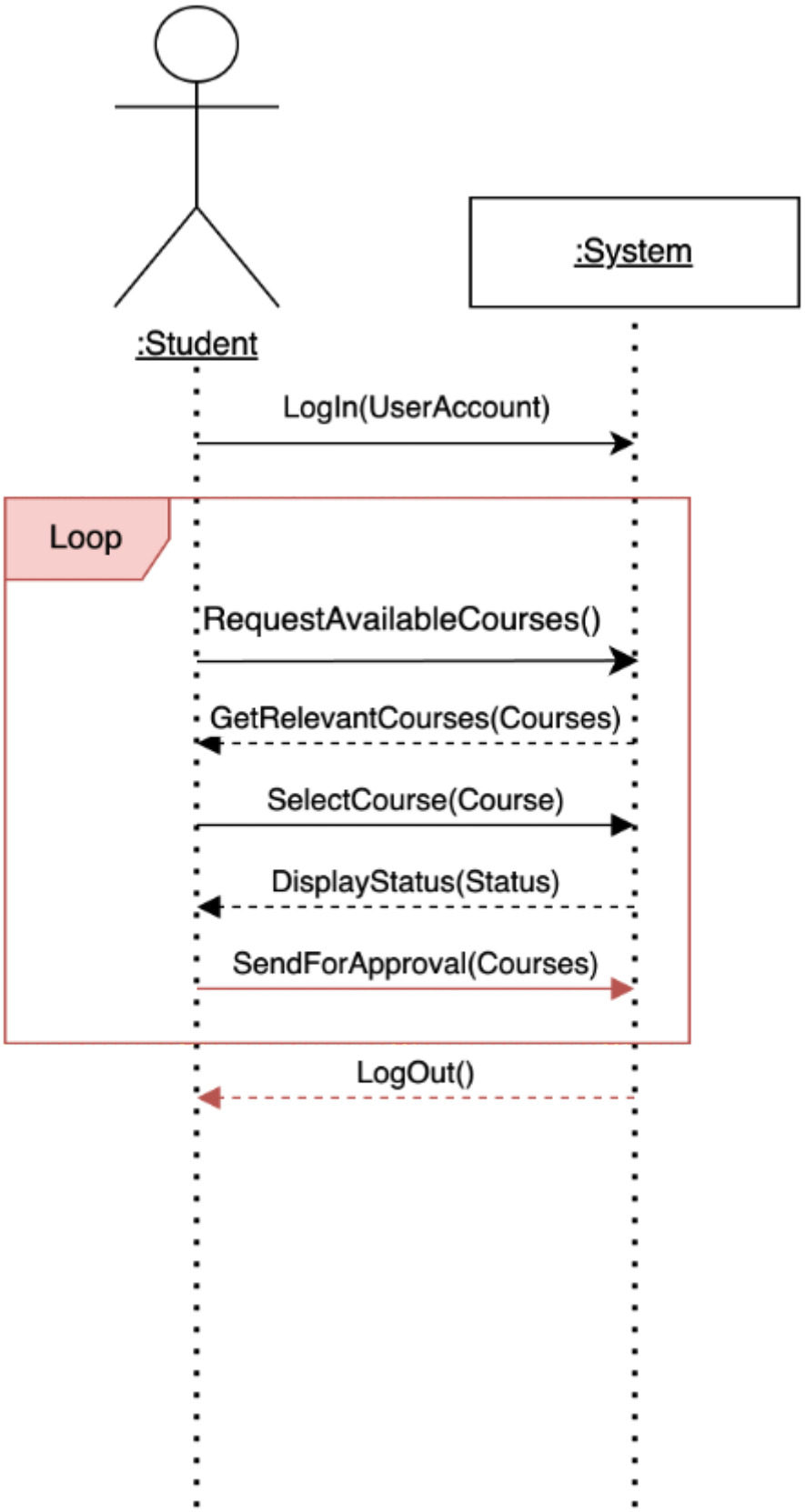
Extensions:

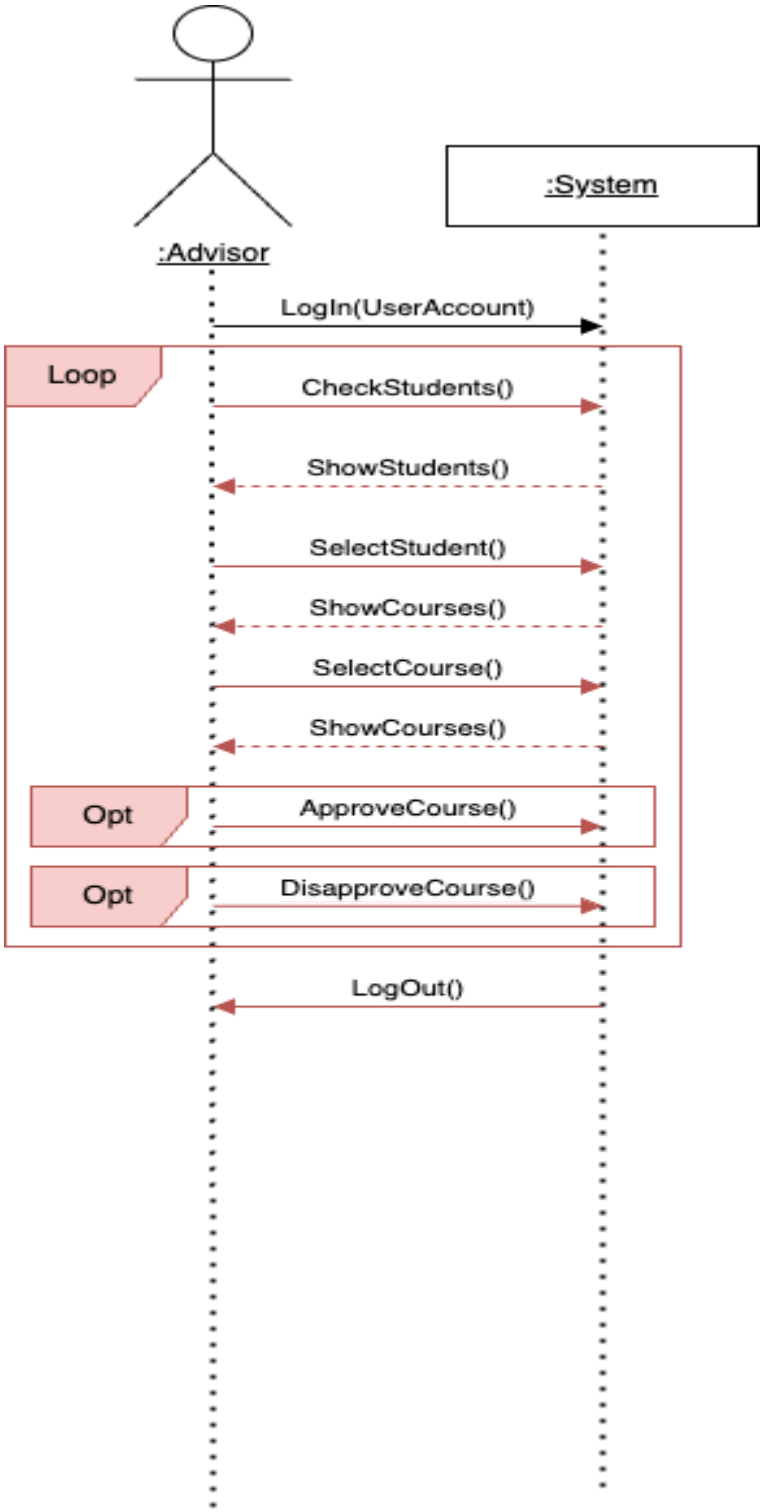
3a. Invalid Credentials:

1. The system shows error message
2. Advisors need to re-enter their credentials.

This step is repeated until the correct credentials are entered.

6. System Sequence Diagram




:System
:Advisor
LogIn(UserAccount)
Loop
CheckStudents()
ShowStudents()
SelectStudent()
ShowCourses()
SelectCourse()
ShowCourses()
Opt
ApproveCourse()
Opt
DisapproveCourse()
LogOut()

7. Credits: iteration 1.

150121043 Bora Duman - In the beginning of the project's code, wrote all the classes in their most basic form. Later, worked on the Student class. Wrote the first four parts of the Requirement Analysis Document (RAD), worked on the Design Class Diagram (DCD).

150121051 Arda Öztürk - Worked on the Advisor class, including its unit tests. Worked on the DCD and the domain model.

150120076 Furkan Gökgöz - Worked on the Student class, including its unit tests. Worked on the Design Sequence Diagram (DSD) and the domain model.

150121032 Mehmet Toprak Balıkçı - Worked on the Transcript class and the System Sequence Diagram (SSD).

150120022 Tolga Fehmioğlu - Worked on the Person and User classes. Worked on the DSD.

150121538 Muhammed Enes Gökdeniz - Worked on the University File System class.

150121002 Enes Torluoğlu - Worked on the Grade, CourseSection and Course classes, and their unit tests.

7.1 Credits: iteration 2.

150121043 Bora Duman - Updated the first four parts of RAD. Worked on the newly implemented Assistant and Message classes. Also worked on updating and debugging older classes.

150121051 Arda Öztürk - Worked on updating the DCD and domain model. Also worked on updating and debugging older classes.

150120076 Furkan Gökgöz - Worked on updating the DSD and domain model.

150121032 Mehmet Toprak Balıkçı - Worked on updating the SSD and DSD. Also worked on updating and debugging the older classes

150120022 Tolga Fehmioğlu - Worked on updating the DSD. Also worked on the Assistant and Message classes.

150121538 Muhammed Enes Gökdeniz - Worked on debugging the older classes.

150121002 Enes Torluoğlu - Worked on testing and debugging the older classes, for a healthier transition to the second iteration.