MARMARA UNIVERSITY

FACULTY OF ENGINEERING

GROUP 2

COURSE REGISTRATION SYSTEM - ITERATION 3

Student Names:

1- Yusuf Demir – 150120032
2- Sena Ektiricioğlu – 150120047
3- Beyza Nur Kaya – 150120077
4- Muhammed Hayta – 150121068
5- Selin Aydın – 150120061
6- Eren Duyuk – 150120509
7- Mustafa Emir Uyar - 150120007

Table of Contents

1. INTRODUCTION

1.1 Vision

The vision of the project is to create a course registration system that will simplify the process of enrolling in courses and submitting them for approval to advisors. The proposed system will provide students with a command line interface that allows them to easily browse through available courses, select the ones they wish to enroll in and submit their selections and send the selections to approval. This will significantly reduce the time and effort required for students to manage their course schedules while ensuring the accuracy of their course records.

1.2 Scope

The student module offers various features, including course search and selection, course updates, and the ability to send course selections for approval to an advisor. It also provides a streamlined registration process with clear instructions. On the other hand, the advisor module offers course management functionality for students and includes the ability to approve or deny course enrollment requests. Along with that, we offer the admin module as well. This module will help the system to be up to date and can make changes directly without waiting for an action if you have the admin privilege.

1.3 System Constraints

- The system must be compatible with the existing university JSON file.
- The system should have scalability to cater to medium-sized student populations.
- Basic security measures must be implemented to safeguard user information.
- The system must comply with university guidelines.

1.4 Stakeholders

- **Students:** End-users responsible for course selection and registration.
- **Advisor:** Users who provide management and functionality to course enrollment approval for students.

- **Admin**: Users with actions that are connected to the database management systems, they can add/remove courses to the system, they can also modify ECTS credit constraint. As well. It brings functionality to the system without changing the JSON files.

1.5 Problem Statements

- Issue 1: Selecting courses can be a challenging and time-consuming task for students.
  Impact: It often leads to delays and errors.

- Issue 2: Advisors encounter manual processes when approving courses.
  Impact: It can result in reduced efficiency, higher workloads, and an increased likelihood of errors.
- Issue 3: Managing the data from outer files is hard for the customers.
  Impact: It can result in corrupt data; it is harder to add regular jobs like adding course.

2.REQUIREMENTS

2.1 Functional Requirements

- Both students and advisors must have secure login credentials to access the system.
- Students can view a list of available courses
- Students can select and enroll in available courses.
- Students can remove themselves from selected courses while in draft status.
- Students can see their timetable.
- Students should be notified when add-drop week is enabled/disabled.
- Advisors can see the courses that students want to enroll in.
- Advisors can approve or reject the course enrollment requests from students.
- Advisor should see the conflicts of course selection of a student.
- Advisor can see the timetable of students.
- The system should display detailed information about each course, including the course name, lecturer name etc.
- The system should ensure seamless integration with existing university JSON files.

- The system should consider the conflicts during course selection.
- The system should have constraints which are stored and can be changed by Admin.
- A student can take the final project if he/she has completed 165 credits.
- Admin should modify courses and constraints in the system.
- Student should see denied courses as notification.
- Advisor should see student's approval request as notification.
- If the student's condition is suitable, the advisor can fine register.
- Advisor should determine final registration of a student.

2.2 Non-Functional Requirements:

- The system should ensure the security of user data, and secure authentication methods.
- The command line interface should be intuitive and easy to use for both students and advisors.
- The system should be stable and available for use.
- The system should improve the user experience by providing feedback and sending some notifications after each transaction.
- Rewrite the program in python.

3.GLOSSARY

TERM	DEFINITION
Transcript	It contains calculations such as completed credits and GPA.
GPA	The GPA, or Grade Point Average, is a number that indicates how high you scored in your courses on average. Using a scale from 1.0 to 4.0, your GPA tracks your progress during your studies.

Prerequisite	It is generally a course that you must complete before enrolling in a second course.
Advisor	Academic advisor is someone in a professional position at an institution who may guide prospective and current students through the admission process, course registration and selection, program planning, degree completion, scholarships and more.
Credit	It is a unit of measurement that represents the amount of academic work a student has completed.
Letter Grade	Symbolic representation of student's academic performance in a course.
ECTS	It is a standard for comparing the study attainment and performance of students across the European Union and other collaborating European countries.
Authorization	Checking and validating users credential.
Admin	Admis is the person who can add courses to the system and delete courses from the system.
Constraint	Limits of features in the system.
Registration Week	Time period during which students can choose the courses they want to take from the system during the semester
Finalize Registration	After finalize registration, students and advisors cannot add or delete courses.

Add-Drop	The time period during the semester when students can drop the course they are taking and take the course they want

4.USE CASES

4.1 Use Case UC1: Course Selection

Scope: University Course Registration System

Level: User-goal

Primary Actor: Student

Stakeholders and Interests:

-Student: wants to view the courses that they can select and choose according to that information, send the courses they had selected to their advisor.

Preconditions: Student is identified before the registration and successfully login to the system.

Success Guarantee or Postconditions: Student can view the courses they can take. They can correctly select the courses. They also need to send them to the advisor with no issues.

Main Success Scenario or Basic Flow:

1. Student opens the system.
2. Student sees the courses available for himself/herself.
3. Student select courses s/he needs to get with their section if there are any provided.
4. Student saves the selected courses with their section information.
5. Student sees the timetable to be informed about any course that is overlapping.
6. Student sends the selected courses to his/her advisor.
7. Student closes the system.

Extensions or Alternative Cases:

***a.** At any time, student may want to exist the system.

1. Student select the exit command.

2. Student exits the system.

1a. Student is informed if the add and drop week has begun.

1. Student is able to modify the courses s/he takes when the add & drop week has come.
2. Student can select and drop any course that s/he taken if it is add & drop week.
3. Student returns to step 2 of the main scenario if it is add & drop week.

3a. Student may select courses s/he not needed.

1. Student removes these courses from the selected ones.

2. Return back to step 2.

4a. Student chooses the course that s/he cannot get, therefore while saving there is an error.

4a1. Student may enter non existing course code.

1. Save operation cannot be done.

2. System gives the error message.

3. Return back to step 2.

4a2. If the section is full, it cannot be selected.

1. System does not save the course.
2. System gives the error message.
3.Return back to step 2.

4.2 Use Case UC2: Course Approvals

Scope: University Course Registration System

Level: User-goal

Primary Actor: Advisor

Stakeholders and Interests:

-Advisor: wants to view their students they advise, see what courses they selected, review elected courses, and approve or reject their application status.

Preconditions: Advisor is identified before the approval and successfully login to the system.

Success Guarantee or Postconditions: Advisor views the students s/he advises. Advisor views the courses selected by the student s/he advises. Advisor reviews the courses selected and approves or rejects accordingly.

Main Success Scenario or Basic Flow:

1. Advisor opens the system.
2. Advisor views the students that s/he is responsible for.
3. Advisor selects the student and sees the courses that the student selected.
4. Advisor checks the timetable of the student if there are any course overlaps and notified how many overlapping courses the student has.
5. Advisor reviews the courses that student selected before.
6. Advisor approves the courses that student selected.
7. Advisor makes finalize registration.
8. Advisor closes the system.

Extensions or Alternative Cases:

***a.** At any time, advisor may want to exist the system.

1. Advisor select the exit command.

2. Advisor exits the system.

1a. Advisor is informed if the add drop week has begun.

1. Advisor returns back to step 2 if the student's registration status has changed.

3a. Advisor may select non-existing student id.

1. System cannot select the student.

2. System gives the error message.

3. Return back to step 2.

3b. In the case of all selected courses being approved or rejected by the advisor.

1. Advisor can select 'final registration'.
2. System saves selection.
3. Return back to step 2.

5a. Advisor can reject the courses that the student selected partially or all of them.

1.The system changes the status of student's course registration by rejecting.

2.System gives the error message to student.

6a. Advisor can choose another student instead of closing the system.

1.Return back to the step 2.

4.3 Use Case UC3: System Management

Scope: University Course Registration System

Level: User-goal

Primary Actor: Admin

Stakeholders and Interests:

-Admin: wants to organize (edit, delete, or add new) courses, advisors, and students, enable add drop week, identify other constraints of the system.

Preconditions: Admin is identified before the registration and successfully login to the system.

Success Guarantee or Postconditions: Admin can view the courses offered by the school, students and lecturers identified in the system. They can organize (edit, delete, or add) them. They also can start add-drop week, identify the system constraints.

Main Success Scenario or Basic Flow:

1. Admin opens the system.
2. System shows operations that s/he can do: organize students, advisors, courses, and change the system constraints.
3. Admin chooses one of the operations s/he can do.
4. Admin takes the necessary actions.
   Admin repeats steps 2-4 until s/he wants to exit the system.
5. Admin exits the system.

Extensions or Alternative Cases:

*a. At any time, admin may want to exist the system.

1. Admin select the exit command.

2. Admin exits the system.

*b. At any time, admin may give incorrect input.

1. System gives an error message to prompt admin to enter the correct input.

2. Admin re-enters the input.

3. The system continues to work from where it left off.

3a. Admin may choose to organize courses.

1. System gives a list of courses.

1a. Admin may want to add course.

1. Admin selects ‘add course option.
2. Admin enters the necessary information for the new course to be added into the system.
3. System saves the new course information.

1b. Admin may delete a course.

1. Admin selects ‘delete course option.

2. Admin selects course to delete.
3. System saves the deletion.

2. Return to step 4.

3b. Admin may select option to change system constraints.

1. System shows all constraints to change.

1a. Admin may want to activate/deactivate add-drop week.

1. Admin selects add-drop option to change its activation.
2. Admin changes the activation status for add-drop.
3. System saves the changes.

1b. Admin may want to activate/deactivate registration week.

1. Admin selects add-drop option to change its activation.
2. Admin changes the activation status for add-drop.
3. System saves the changes.

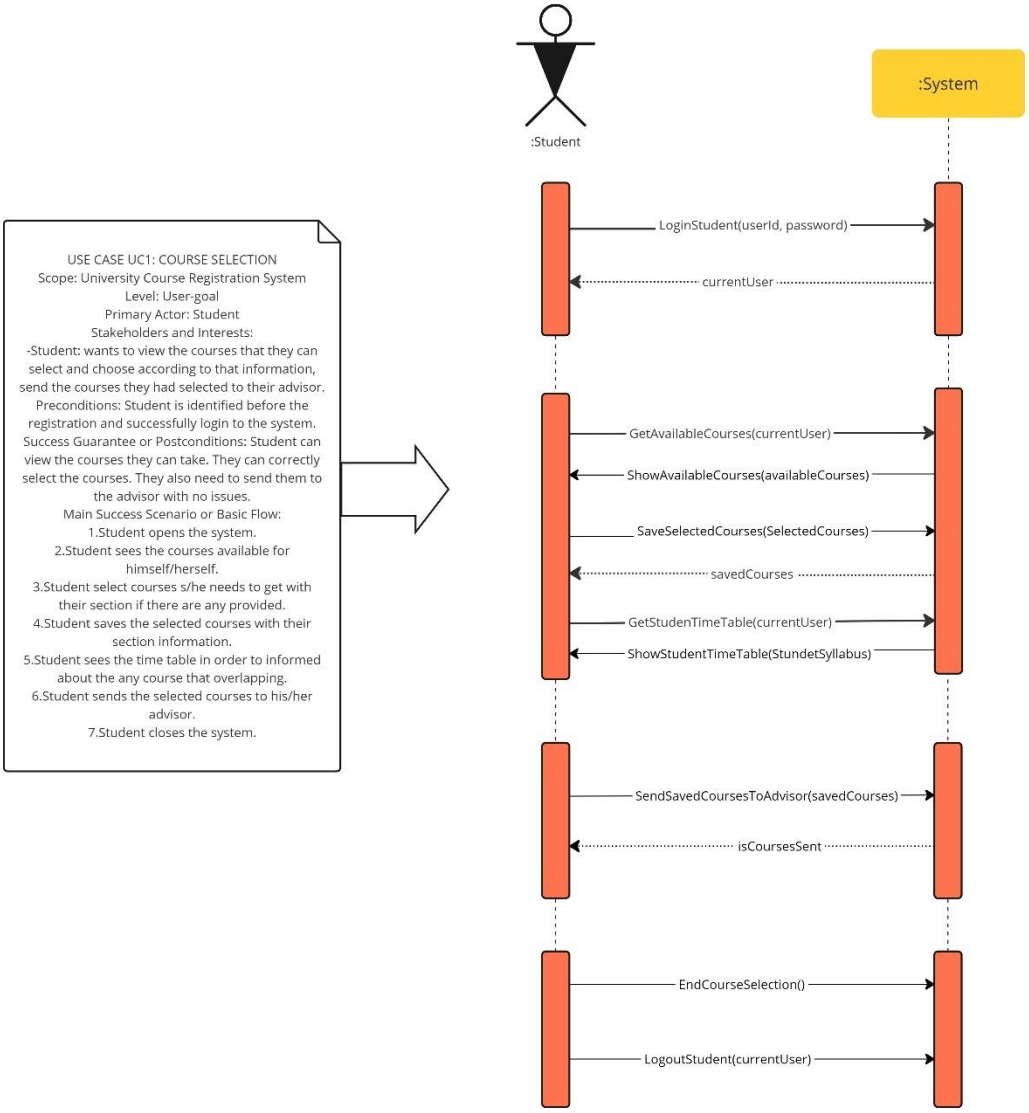
1c. Admin may want to change maximum number of courses that can be taken by student for a semester.

1. Admin selects 'max number of courses that can be taken' option to change it.
2. Admin changes max number of courses that can be taken by student.
3. System saves the changes.

2. Return to step 4.

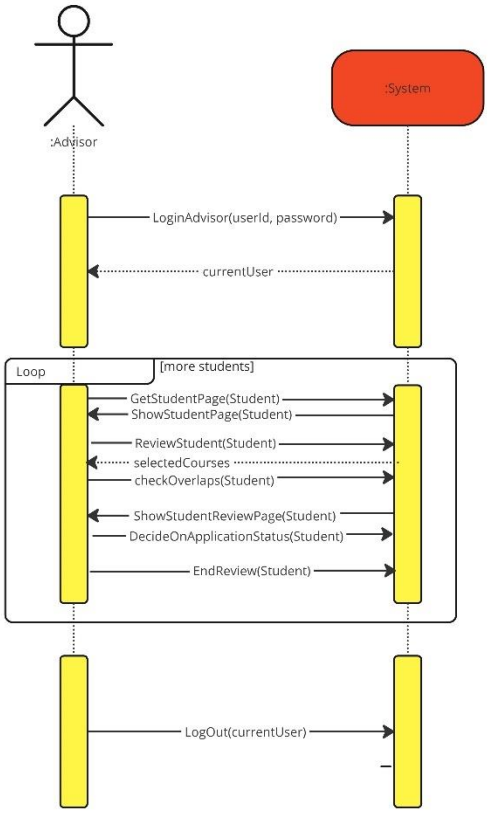
5.SSD DIAGRAMS

5.1 Student SSD



5.2 Advisor SSD

USE CASE UC2: COURSE APPROVALS
Scope: University Course Registration System
Level: User-goal
Primary Actor: Advisor
Stakeholders and Interests:
-Advisor: wants to view their students they advise, see what courses they selected, review elected courses, and approve or reject their application status.
Preconditions: Advisor is identified before the approval and successfully login to the system.
Success Guarantee or Postconditions: Advisor views the students s/he advises. Advisor views the courses selected by the student s/he advises. Advisor reviews the courses selected and approves or rejects accordingly.
Main Success Scenario or Basic Flow:
1.Advisor opens the system.
2.Advisor views the students that s/he is responsible for.
3.Advisor selects the student and sees the courses that the student selected.
4. Advisor checks the syllabus of the student if there is any course overlaps and notified how many overlapping courses the student has.
5.Advisor reviews the courses that student selected before.
6.Advisor approves the courses that student selected.
7.Advisor makes finalize registration
8.Advisor closes the system.



5.3 Admin SSD

Main Success Scenario or Basic Flow:
1. Admin opens the system.
2. System shows operations that s/he can do: organize students, advisors, courses, and change the system constraints.
3. Admin chooses one of the operations s/he can do.
4. Admin takes the necessary actions.
Admin repeats steps 2-4 until s/he wants to exit the system.
5. Admin exits the system.

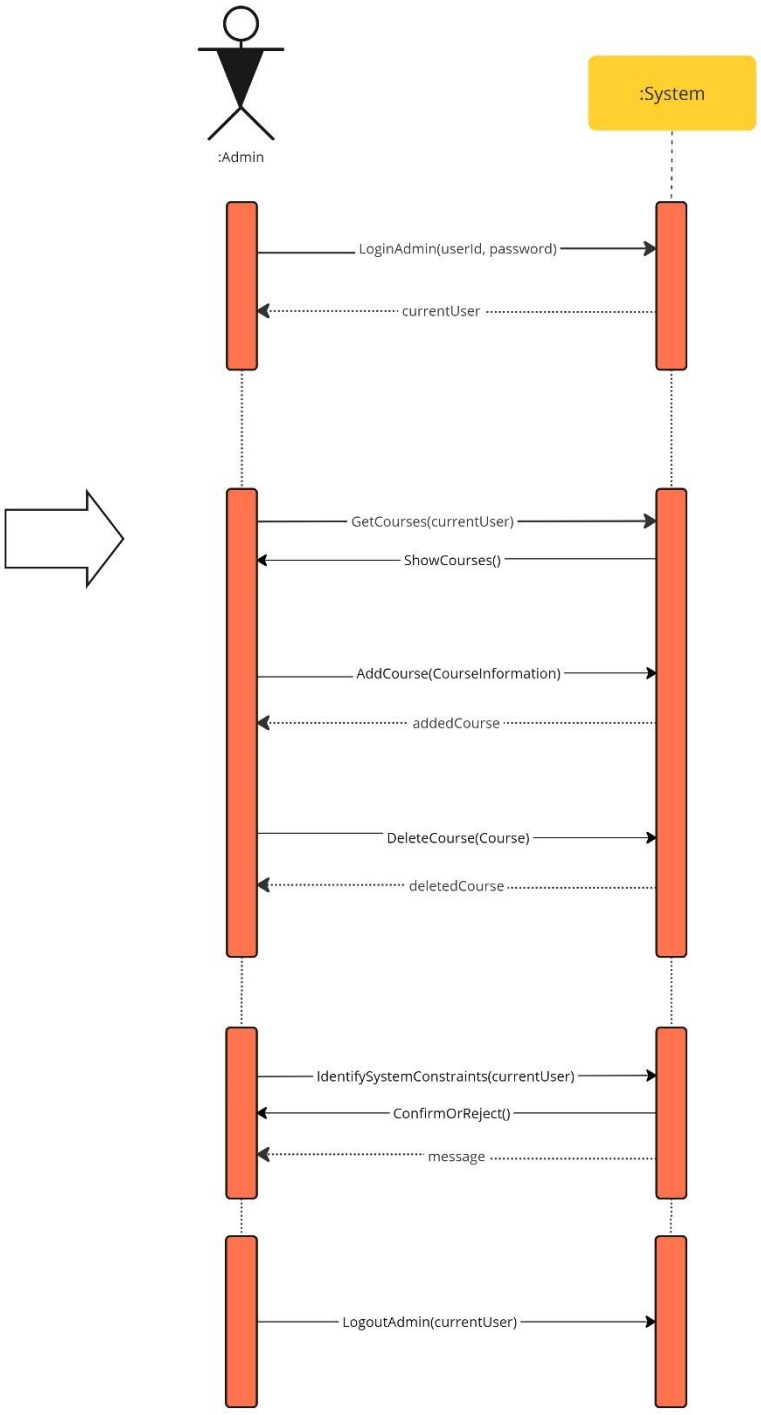
Extensions or Alternative Cases:
*a. At any time, admin may want to exist the system.
1. Admin select the exit command.
2. Admin exits the system.
*b. At any time, admin may give incorrect input.
1. System gives an error message to prompt admin to enter the correct input.
2. Admin re-enters the input.
3. The system continues to work from where it left off.

3a. Admin may choose to organize courses.
1.System gives a list of courses.
1a. Admin may want to add course.
1.Admin selects 'add course option.
2.Admin enters the necessary information for the new course to be added into the system.
3.System saves the new course information.
1b. Admin may delete a course.
1.Admin selects 'delete course option.
2.Admin selects course to delete.
3.System saves the deletion.
2.Return to step 4.

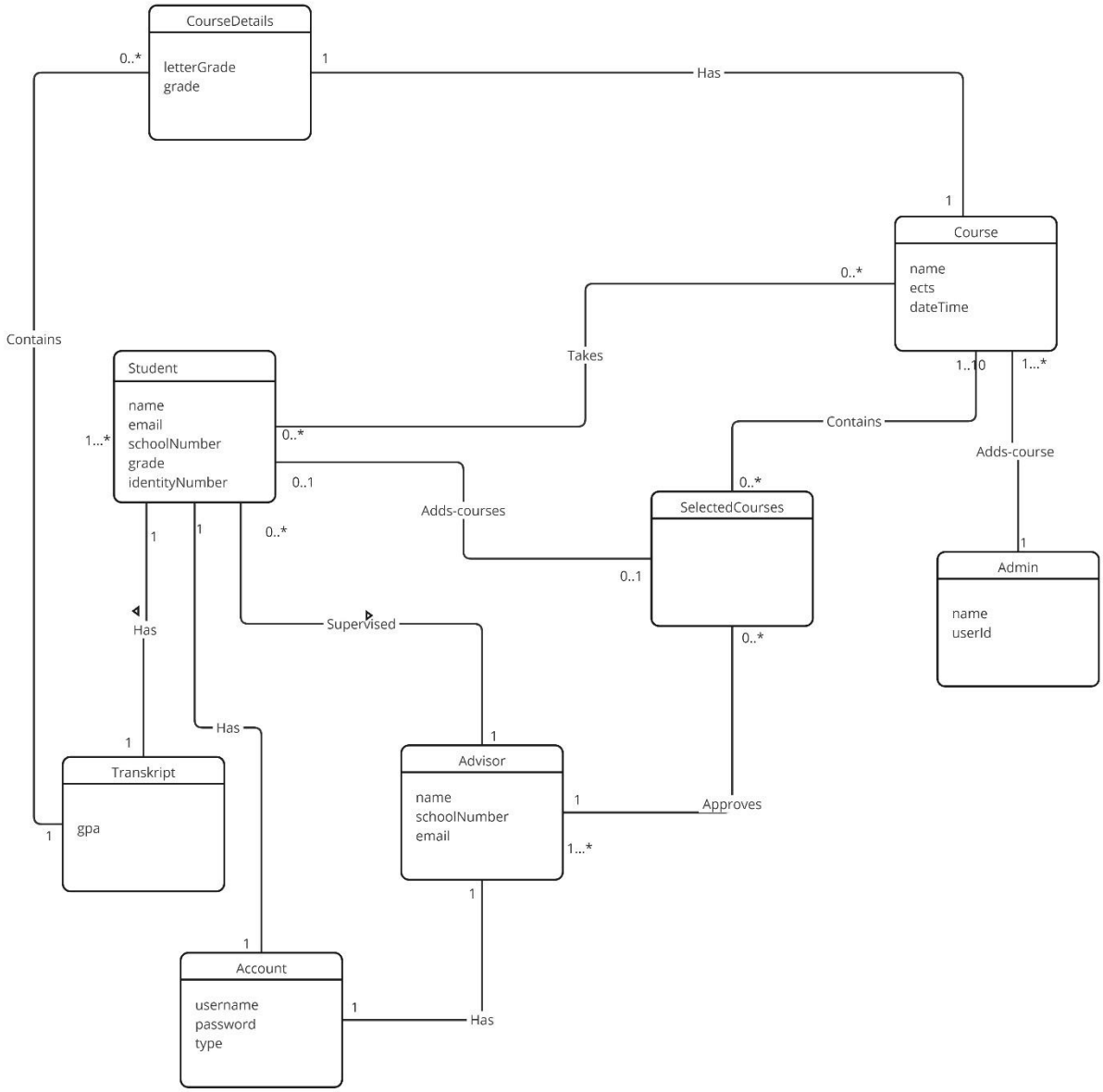
3b. Admin may select option to change system constraints.
1.System shows all constraints to change.
1a. Admin may want to activate/deactivate add-drop week.
1.Admin selects add-drop option to change its activation.
2.Admin changes the activation status for add-drop.
3.System saves the changes.

1b. Admin may want to activate/deactivate registration week.
1.Admin selects add-drop option to change its activation.
2.Admin changes the activation status for add-drop.
3.System saves the changes.

1c. Admin may want to change maximum number of courses that can be taken by student for a semester.
1.Admin selects 'max number of courses that can be taken' option to change it.
2.Admin changes max number of courses that can be taken by student.
3.System saves the changes.
2.Return to step 4.



6. DOMAIN MODAL



7.TEAMWORK

7.1 Group Members

1- Yusuf Demir – 150120032 (Group Representative)
2- Sena Ektiricioğlu – 150120047
3- Beyza Nur Kaya – 150120077
4- Muhammed Hayta – 150121068
5- Selin Aydın – 150120061
6- Eren Duyuk – 150120509
7- Mustafa Emir Uyar - 150120007

7.2 Project Management

We are working with 7 people in this project by following iterative development. We follow four phases in each iteration. We hold many meetings during the iteration to discuss, plan and work together which can be either online or face-to-face. We use GitHub exclusively. Issues are created and assigned to group members to manage the project easily. We also use GitHub projects for each iteration. Issues are connected to corresponding GitHub projects. Thus, we are able to follow the related issues of iterations and see and manage them in a kanban view.

During the development, we always try to consider KISS and YAGNI principles. We try to stick with the customer requirements.

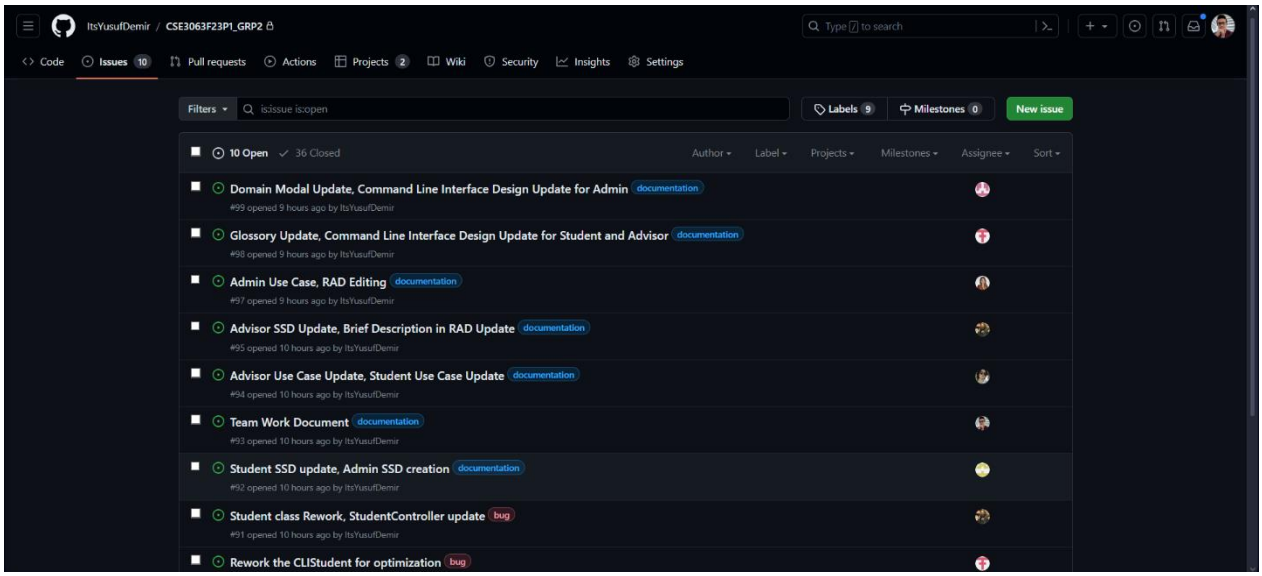


Figure 1: Issues

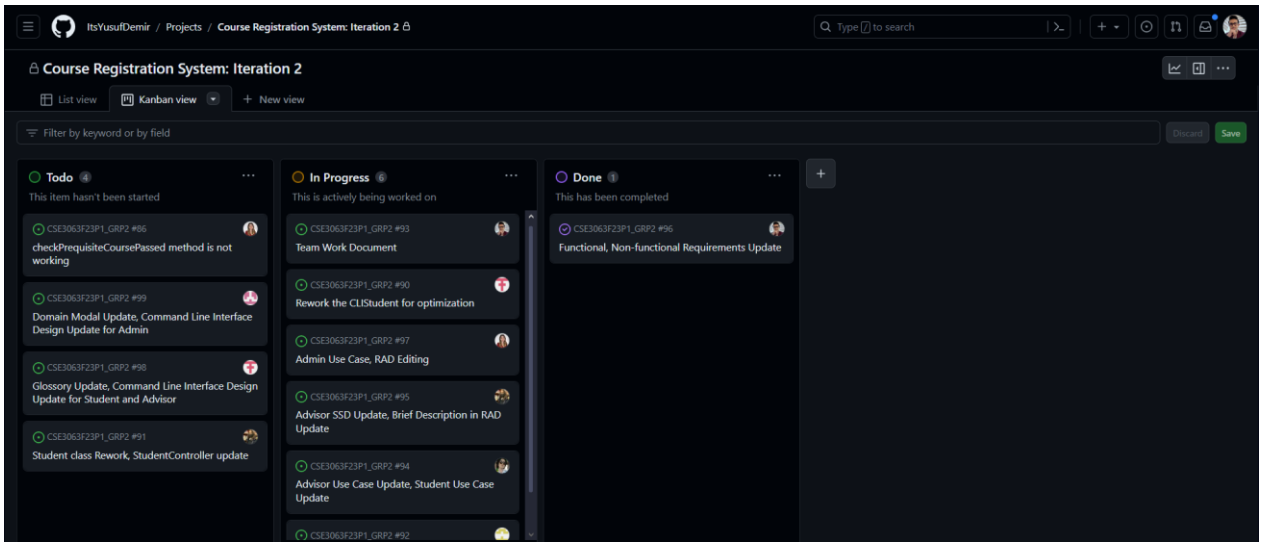


Figure 2: Kanban View

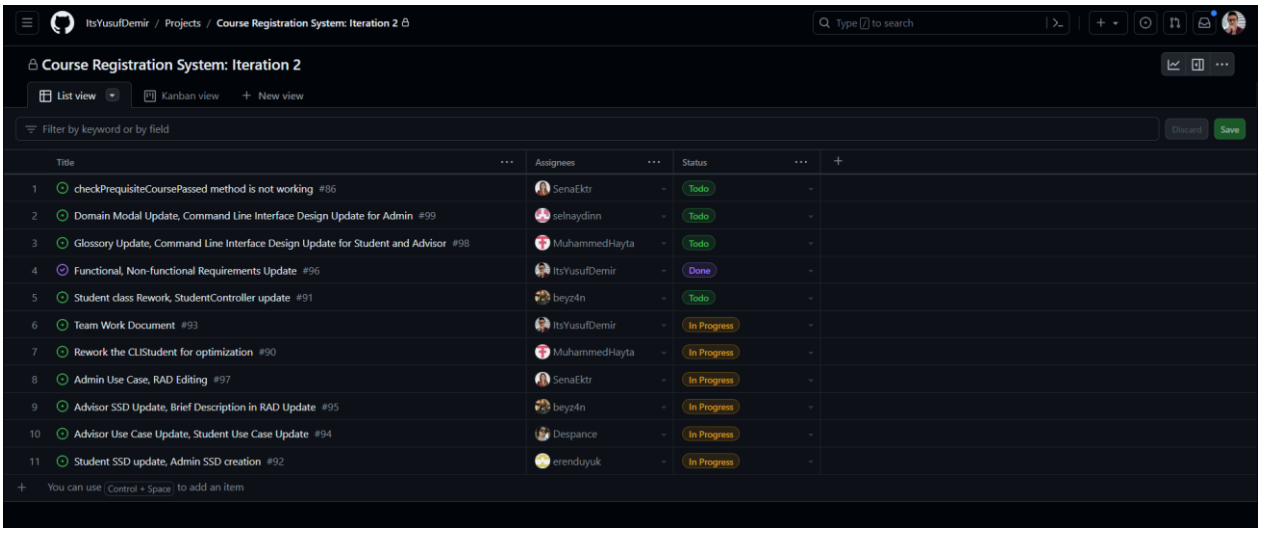


Figure 3: List View

For the designing part, we decided to use Miro which is a drawing tool which allows drawing together online. It made it very easy to design sequence and class diagrams.

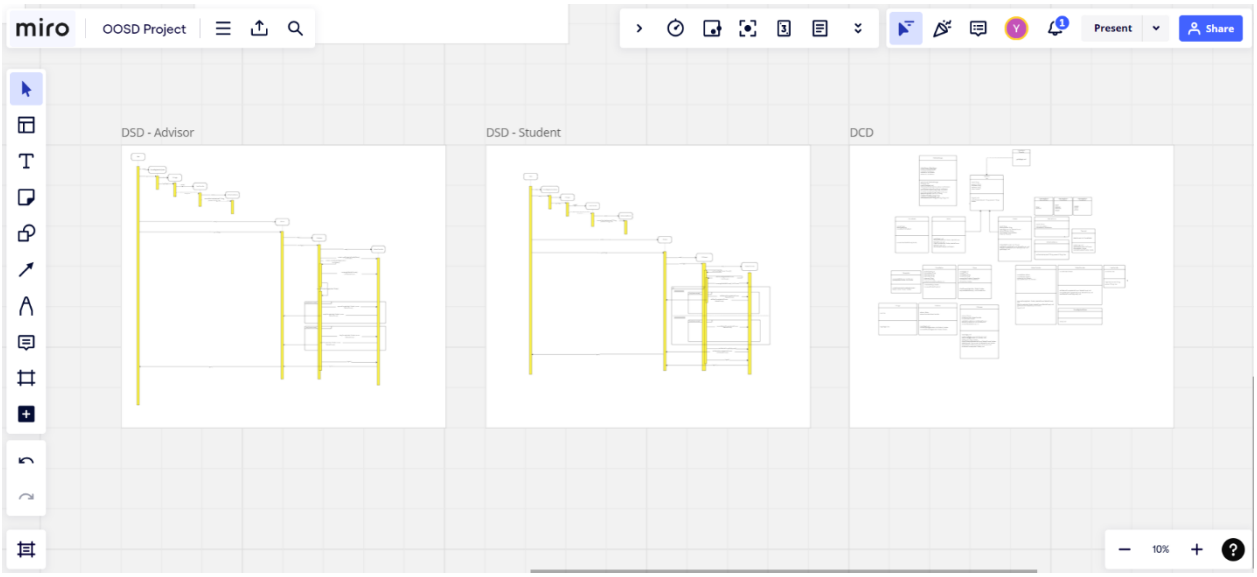


Figure 4: Miro

7.3 Iteration I Roles and Responsibilities

Yusuf Demir – 150120032 (Group Representative)

Yusuf Demir is the team leader of the project. Meeting arrangements, GitHub management work and coordination between group members are done by him.

Requirement Analysis: Domain modal design.

Design Phase: UML class design for classes DatabaseManager, AuthenticationService. Sequence diagram of Advisor (for the parts CLIAdvisor and AdvisorController).

Implementation: CourseRegistrationSystem and DatabaseManager classes.

Testing: Testing the completed credits method.

Sena Ektiricioğlu – 150120047

Requirement Analysis: Student use case design.

Design Phase: UML class design for classes Course and CourseSection. Sequence diagram of Student (for the parts CLIStudent and StudentController).

Implementation: Course, CourseSection, and Prerequisite classes.

Testing: Testing the perquisite check method.

Beyza Nur Kaya – 150120077

Requirement Analysis: Advisor use case design.

Design Phase: UML class design for classes Advisor and PrerequisiteCourse. Sequence diagram of Student (for the parts Student and CLIStudent).

Implementation: Student, ApprovalStatus, and StudentController classes.

Testing: Testing the calculate gpa method.

Muhammed Hayta – 150121068

Requirement Analysis: Advisor SSD design.

Design Phase: UML class design for classes User, and AdvisorController. Sequence diagram of Advisor (for the parts AuthenticateService and Advisor).

Implementation: Transcript, CLIStudent, and AuthService classes.

Testing: Testing the get passed courses method.

Selin Aydın – 150120061

Requirement Analysis: Glossory of important concepts, RAD edit.

Design Phase: UML class design for class Student. Sequence diagram of Student (for the parts CourseRegistrationSystem, CLILogin, UserController, and AuthenticationSystem).

Implementation: CLILogin, CLIAdviser, and SelectedCourse classes.

Testing: Testing the user credentials method.

Eren Duyuk – 150120509

Requirement Analysis: Command line interface design.

Design Phase: UML class design for classes CLILogin, CLIStudent and CLIAdvisor. Sequence diagram of Advisor (for the parts CourseRegistrationSystem, CLILogin, and UserController).

Implementation: Advisor, and AdvisorController classes.

Testing: Testing the order student list method.

Mustafa Emir Uyar – 150120007

Requirement Analysis: Student SSD design.

Design Phase: UML class design for classes Student, StudentController, UserController, Showable and SelectedCourse. Sequence diagram of Advisor (for the parts Advisor, and CLIAdvisor).

Implementation: User, and UserController, Saveable and Showable classes.

Testing: Testing the user credentials in detail.

7.4 Iteration II Roles and Responsibilities

Yusuf Demir – 150120032 (Group Representative)

Yusuf Demir is the team leader of the project. Meeting arrangements, GitHub management work and coordination between group members are done by him.

Requirement Analysis: Functional and non-functional requirements update.

Design Phase: UML class design for class DatabaseManager. Sequence diagram of Admin (for the parts Main and Admin).

Implementation: Constraint and User (update) classes.

Testing: Testing the completed credits method (Rework).

Sena Ektiricioğlu – 150120047

Requirement Analysis: Admin use case design, RAD editing.

Design Phase: UML class design for classes Constraint and User (update). Sequence diagram of Admin (for the parts AdminController and CLIAdmin).

Implementation: CLIAdmin (First half of the methods) class.

Testing: Testing the perquisite check method (Rework).

Beyza Nur Kaya – 150120077

Requirement Analysis: Advisor SSD update, RAD descriptions update.

Design Phase: UML class design for classes Admin and Student (update). Sequence diagram of Student (for the parts StudentController and CLIStudent).

Implementation: AdminController class.

Testing: Testing the calculate GPA method (Rework).

Muhammed Hayta – 150121068

Requirement Analysis: Glossory update, commond line interface design update for student and advisor.

Design Phase: UML class design for classes CLIStudent, and StudentController. Sequence diagram of Advisor (for the parts AdvisorController and CLIAdvisor).

Implementation: CLIStudent (update) class.

Testing: Testing the get passed courses method (Rework).

Selin Aydın – 150120061

Requirement Analysis: Domain model update, command line interface design for admin.

Design Phase: UML class design for class AdminCLI. Sequence diagram of Student (for the parts Student and CLIStudent).

Implementation: Student (Update), Advisor (Update), and DatabaseManager (Update)classes.

Testing: Testing the user credentials method (Rework).

Eren Duyuk – 150120509

Requirement Analysis: Student SSD update, Admin SSD.

Design Phase: UML class design for classes AdminController, and Advisor (update). Sequence diagram of Advisor (for the parts Advisor, and AdvisorController).

Implementation: CLIAdmin (second half of the methods), class.

Testing: Testing the letter grade conversion method.

Mustafa Emir Uyar – 150120007

Requirement Analysis: Advisor use case update, Student use case update.

Design Phase: UML class design for classes Utils, and Color. Sequence diagram of Admin (for the parts Admin, and AdminController).

Implementation: Admin, and AdminController classes.

Testing: Testing the edit constraint method.

7.5 Iteration III Roles and Responsibilities

Yusuf Demir – 150120032 (Group Representative)

Yusuf Demir is the team leader of the project. Meeting arrangements, GitHub management work and coordination between group members are done by him.

Requirement Analysis: Functional and non-functional requirements update.

Design Phase: DCD Advisor.

Implementation: Python Conversion (DatabaseManager, CourseRegistrationSystem, Main)

Testing: Testing the completed credits method (Rework).

Sena Ektiricioğlu – 150120047

Requirement Analysis: Functional and non-functional requirements update.

Design Phase: Admin DCD update.

Implementation: Python Conversion (Course, CourseSection, Prerequisite, Constraint, AdminController)

Testing: Testing the perquisite check method (Rework).

Beyza Nur Kaya – 150120077

Requirement Analysis: Advisor use case update.

Design Phase: DCD CourseStatus, ApprovalStatus update.

Implementation: Python Conversion (Student, StudentController, CourseGrade, Enums)

Testing: Testing the calculate GPA method (Rework).

Muhammed Hayta – 150121068

Requirement Analysis: Glossory update, commond line interface design update for student and advisor.

Design Phase: DCD constraint update.

Implementation: Python Conversion (Transcript, CLIStudent, AuthenticationService)

Testing: Testing the get passed courses method (Rework).

Selin Aydın – 150120061

Requirement Analysis: RAD update.

Design Phase: DCD StudentController

Implementation: Python Conversion (CLIAdvisor, CLILogin, SelectedCourse)

Testing: Testing the user credentials method (Rework).

Eren Duyuk – 150120509

Requirement Analysis: Glossory update.

Design Phase: DCD AdvisorController update.

Implementation: Python Conversion (Advisor, AdvisorController, CLIAdmin)

Testing: Testing the letter grade conversion method.

Mustafa Emir Uyar – 150120007

Requirement Analysis: Advisor use case update, Student use case update.

Design Phase: DCD AdminController update.

Implementation: Python Conversion (User, Util, Color, Showablle, UserController, Admin)

Testing: Testing the edit constraint method.