MARMARA UNIVERSITY

FACULTY OF ENGINEERING

COMPUTER ENGINEERING DEPARTMENT

CSE3063

Object Oriented Software Design

Java Project – Fall23 – Iteration 2

1.	Aksanur KONUK	150120049
2.	Duru BAŞTUNALI	150120075
3.	Elife KOCABEY	150120054
4.	Eren ÇETİN	150120008
5.	Nidanur DEMİRHAN	150120042
6.	Semih BAĞ	150120070
7.	Zeynep YILMAZ	150120066

1. Purpose And Description

In this project, we will try to develop a course registration system. In the first phase of the project, only two different user types will be considered and developed. These user types will be divided into two: student and consultant.

When students log in to the system, they will be able to choose from the courses offered to them. After making a selection, they will be able to send it to the consultant for approval.

When the advisor logs into the system, he will be able to see the students he advises and their information. They will also be able to see course selection requests submitted by their students that are awaiting approval. Advisors will be able to approve or disapprove course selection requests from students.

In short, we will develop a course registration system similar to the real world.

2. Glossary Of Important Concepts

(In ascending order)

1. Adding course: Adding new course to the courses have taken.
2. Advisor: A person who is responsible for adding/dropping courses for students, as well as teaching courses.
3. ApproveRequest: Advisor accepts the changes which the student wants to make.
4. Cancel button: Allows changes not to be saved.
5. Command-line: It is a text-based interface that provide user to make operation on Course Registration System.
6. Course: It is the integrity of information created by the combination of interconnected information.
7. Course Registration System: It is a system that students can request adding or removing courses, advisors can accept or reject those requests.
8. CourseSchedule: It is a indicator that what days of the week and time a course is on.
9. Course Section: Gives information about course.
10. CourseType: The courses are grouped within themselves. They are four course types. Mandator, Faculty, Technical and Non-Technical.
11. Credit: It is a quantity that indicates the weight of the courses.
12. DisapproveRequest: Advisor doesn’t accept the changes which the student wants to make.
13. Dropping course: Dropping course from courses have taken.
14. Exit: The user in the system shuts down the system and changes are saved to the JSON files permanently.
15. Faculty: It is a course that is related to the other departments.
16. GPA: It is a grading system used for academic data notes of information.

17. Grade: Indicator showing how successful a lesson was passed
18. Id: Everything in the system has a unique name.
19. JSON: It is a file format that provide storing datas.
20. Lecturer: A person who is responsible for teaching courses.
21. Login: Accessing Course Registration System with correct combination of username and password.
22. Logout: The changes are saved temporarily on the system.
23. Mandatory: The type of the courses the student must take according to the term he/she in school.
24. Menu: It is a list of what the user can do in Course Registration System.
25. Non-Technical: It is a course type that is open for more self-improvement.
26. Notification: When an advisor approves or disapproves a request, he/she provides feedback to the student.
27. Password: It is a combination of characters that enable the use course registration system.
28. Prerequisite Course: Before taking that course, other courses related to it must be taken.
29. Profile: Each user in the system has a profile and the profile consists of the name and surname of the person.
30. Quota: It is a limit that indicates how many students can take any course.
31. Request: It is a type of informative signal that a student sends to the advisor what courses he/she wants to add.
32. Save button: Allows changes to be saved.
33. Syllabus: A table that shows the days and time of courses.
34. Student: A person who is responsible for completing courses.
35. Technical: It is a more detailed course on specific areas related to the department.
36. Term: It is a type of time according to university.
37. Transcript: Students' grade of lecture records
38. User: It is a person that try to use Course Registration System. It can be student, lecturer or advisor.
39. UserName: It is a name. The one of the factors that enable the use course registration system.

3. List Of Requirements

3.1 FUNCTIONAL REQUIREMENTS

LOGIN INTERFACE

- A login page must be loaded first when the system is started.
- Users must be able to log in to the system by entering their username and password on the login page.
- When trying to log in with a user profile information that does not exist in the system, an error message should be displayed on the screen and then the reset login interface should be displayed on the screen.
- If the password in the database does not match the password entered by the user, an error message should be printed on the screen and the login interface should be printed on the screen again.
- The program should be able to terminate with the cancel option in the login interface.

MENU

- Each of the headings in the menu content should be considered as a component and should be designed in a way that new menu options can be easily integrated or removed when necessary.
- Users can access the tab they want to select by clicking on the relevant tab number from the keyboard.
- Each user should be able to get the necessary information from the menu tabs created according to the user type.

COURSES LIST

- Students should be able to see in a list which courses are included in the department.
- Students should be able to see which instructor gave the relevant course or other information about the course.

COURSES OFFERED

- Students should be able to learn the list of opened courses from the relevant menu tab.

SELECTED COURSES

- The student type user should be able to choose the courses he/she wants from the courses offered to him/her and create a course registration list.
- Students should be able to send the list of selected courses they have created to their advisor for approval.
- Students should be able to remove any course from the list of selected courses they have already created.
- Students should be able to cancel the list of selected courses that has already been created.
- Students should be able to see the status and content of the approval request they have previously submitted.

ADVISOR INTERFACE

- Advisors should be able to see the students they are interested in in a list.
- Advisors should be able to view course selection approval requests they receive.
- Advisors can approve any approval request
- Advisors should be able to reject any approval request.

PROFILE INTERFACE

- The profile tab must be selected from the menu.
- When the Profile tab is selected, the user's name and surname must be printed on the screen.
- At the same time, this tab should offer the student the option to change their password.

NOTIFICATION INTERFACE

- The notification tab must be selected from the student menu.
- If there is an unread notification sent to the student, it should be displayed in this tab.
- At the same time, if there is a read notification, these notifications should also be shown.

TRANSCRIPT INTERFACE

- This tab should enter to view transcripts from the student menu.

- In the transcript tab, student information, department information, and the student's course history should be printed on the screen.

SYLLABUS INTERFACE

- It is necessary to enter the syllabus tab from the student menu.
- A course schedule specific to the courses chosen by the student should be printed in the syllabus tab.

LECTURER INTERFACE

- Lecturer should be able to see the list of courses he/she teaches.
- Lecturer must have access to a list of all students in a course he/she teaches.
- Lecturer should be able to see his own syllabus specific to the lessons he/she teaches.

STUDENT INTERFACE

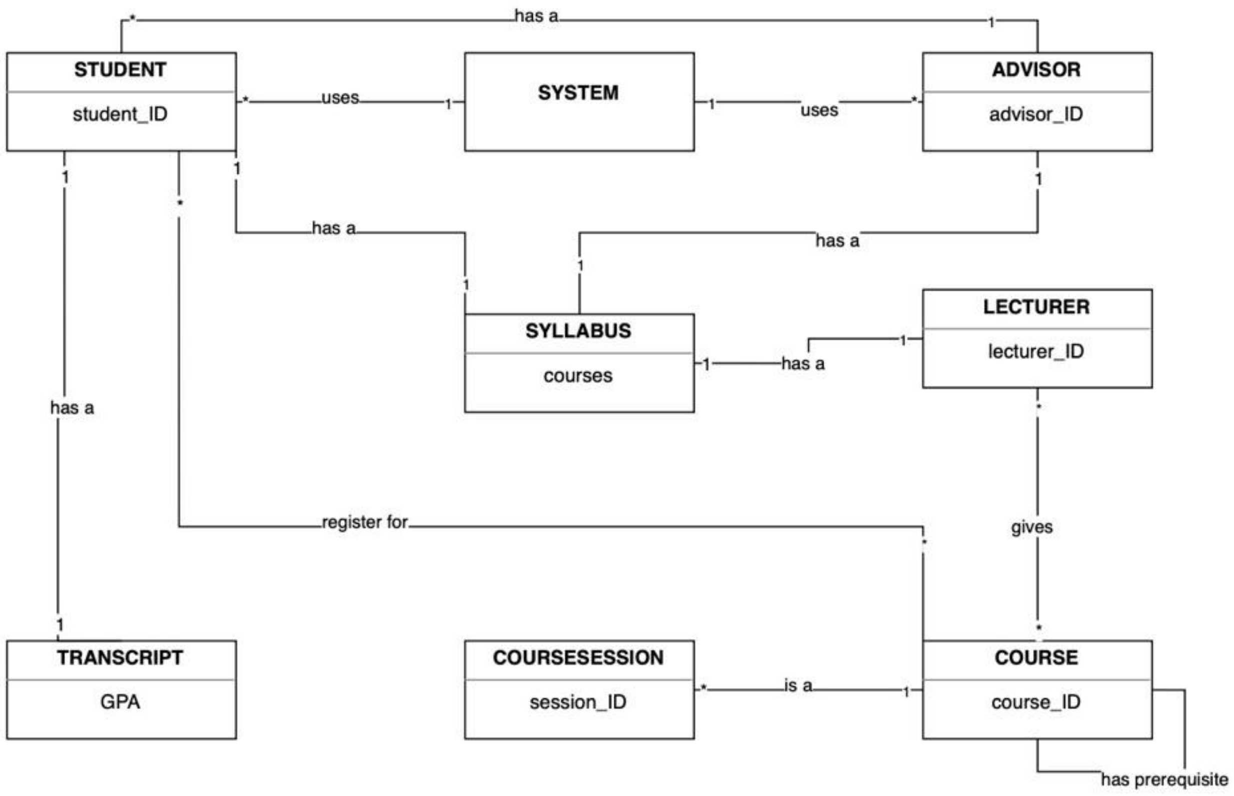
- The student should be able to see the profile information.
- The student should be able to see notifications from the advisor.
- The student should be able to see transcripts.
- The student should be able to see the weekly syllabus according to the selected/approved courses.
- The student should be able to see all the courses he/she will take during his/her educational life.
- The student should be able to see approved courses.
- The student should be able to see offered courses.
- The student should be able to see selected courses and send selected courses for approval.

3.2 NON-FUNCTIONAL REQUIREMENTS

- JSON filing system should be used to store data.
- When the system is run, the relevant json files should be read and processed into memory as objects.
- When the program is closed, all changes made to the objects so far should be processed into json files.
- When the user logs in successfully, the user-specific menu interface should be printed on the screen.
- The system should automatically adjust the courses a student can take.

- The courses a student can take should be created with that student's information. While doing this, conditions such as average score and affiliated course status should be taken into consideration.
- Since a graphical user interface will not be developed yet, the menu to be created in the terminal should be simple and understandable.

4. Domain Model



5. Use Cases

Use Case 1: Login System for User
Primary Actor: User
Stakeholders and Interests:

- Student: wants to log in to the system for class registration
- Advisor: wants to log in to the system for student's class registration approval or rejection
- Lecturer: wants to log in to the system to view the classroom content for each of their class

Preconditions: The user should have their username and password saved in files.

Postcondition: Login is confirmed. The system is reached by the user and the main menu is formed based on their user type such as student, advisor, or lecturer.

Main Success Scenario:

1. The user is welcomed by a login screen.
2. The login screen displays an area for the user to enter their username and password.
3. The user will enter their username and password information together.
   1a. No such user exists; System gives a warning as "Incorrect Username/Password".
   2a. The password doesn't match for the given username; System gives a warning as "Incorrect Username/Password".
   2. The login screen displays an are for the user to enter their username and password.
4. If username and password information are both correct and valid for the associated account, the login process occurs and the user will successfully reach their account for further actions.

Use Case 2: Basic Course Registration System (Student Perspective)
Actor: Student
PreConditions: The student is logged into the course registration system.
Postconditions: The student successfully sent the course registration request and the system saved it in the Json file

Main Flow:

1. Student-specific menu content is printed on the screen
2. The student enters an input (selected courses tab order) from the keyboard.

2a- The student entered a value that is not in the menu
2b- Appropriate error message is printed on the screen
   1- Student-specific menu content is printed on the screen
3. The list of courses opened on the screen is printed.
4. The student enters the numbers of the courses he or she wants to take from the keyboard.
   4a-If a number that should not be entered is entered, the courses indicated by the valid numbers are added to the selection list.
      For others, the error message is printed on the screen.
   4- The student enters the numbers of the courses he or she wants to take from the keyboard.
5a- The student presses the save button
5b- The student presses the cancel button
   5ba- Cancellation confirmation text is printed on the screen
      4- The student enters the numbers of the courses he or she wants to take from the keyboard.
6. The student presses the send button for the advisor's approval.
7. The registration request process ends successfully

Use Case 3: Basic Course Registration System (Advisor Perspective)
Actor: Advisor
Preconditions: The advisor is logged into the course registration system.
Postconditions: The advisor has reviewed and approved course selections for their assigned students, and the students are successfully registered for the approved courses.

Main Flow:

1. The advisor initiates the course registration process for their assigned students.
2. The system presents a list of students for whom the advisor is responsible.
3. The system presents a list of students who have submitted course selections for advisor approval.
4. The advisor selects a student from the list.
5. The system displays the available courses for the selected student.
6. The advisor reviews the courses and selections made by the student.
   1a. If the advisor decides to reject a student's course selections, they provide feedback, and the student can make revisions to their selections.
   2a. If the selections are approved, the system registers the student for the chosen course.

4. The advisor selects a student from the list.

7. Once all students' course selections have been reviewed, the advisor confirms the registration process for approved students.

   1a. If the course registration system encounters technical issues or errors at any point in the main flow, the use case may terminate with an error message.

8. The use case ends and the advisor has successfully completed the course registration for their assigned students.

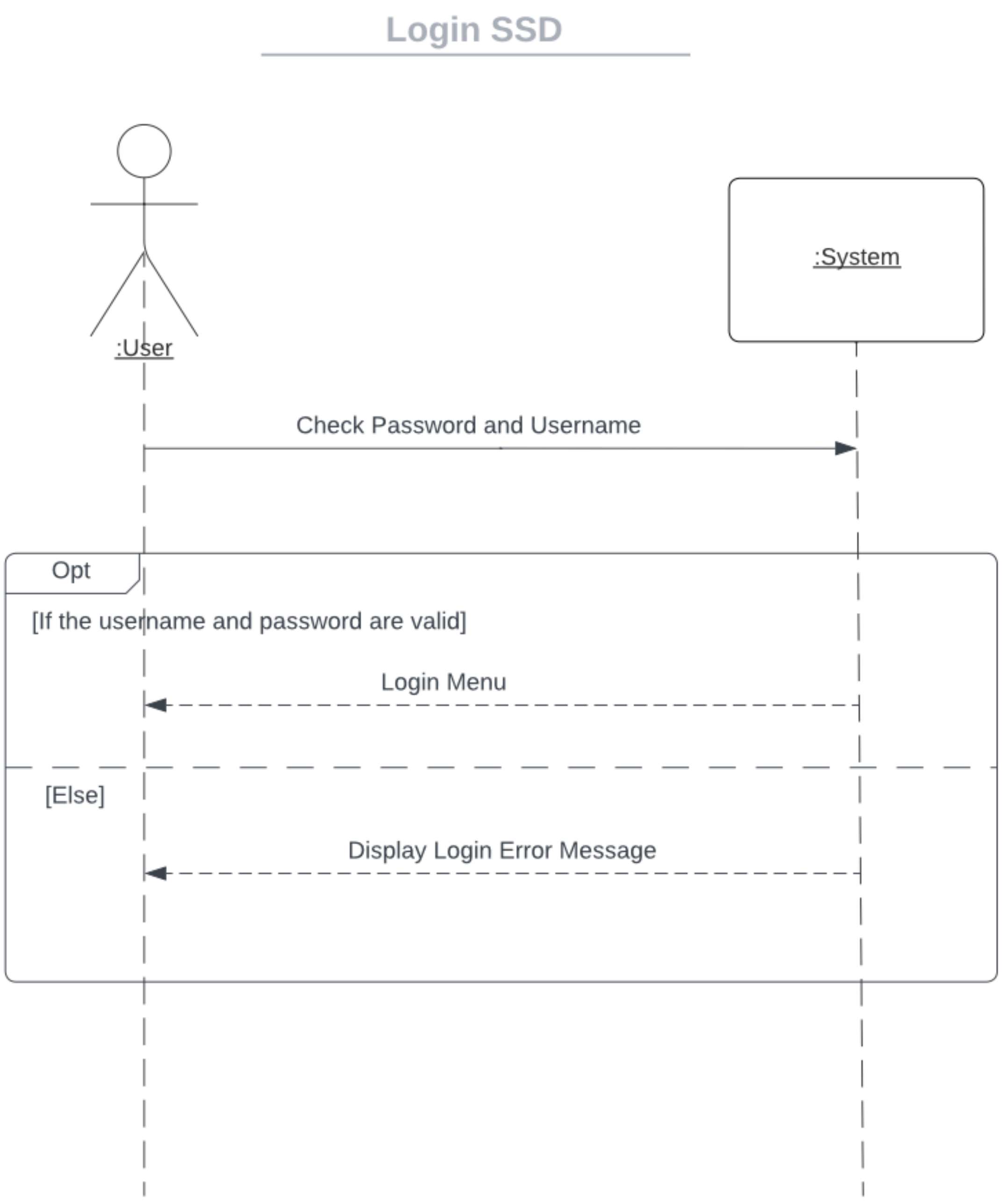
Use Case 4: See students enrolled in her/his courses (Perspective of Lecturer)
Actor: Lecturer
Preconditions: The lecturer is logged into the course registration system.
Postconditions: The lecturers can see student information ,who take his/her courses, successfully.

Main Flow:

1. The lecturer menu is displayed.
2. The lecturer enters an input in the menu page to go to the Courses interface.
2a. The lecturer entered invalid value,
2b. The appropriate error message is printed
  1- The lecturer menu is displayed.
3. The teacher presents a comprehensive list of the courses he/she offers.
4. The lecturer enters an input to select a course..
4a. The lecturer entered invalid value.
4b. The appropriate error message is printed
  3- The teacher presents a comprehensive list of the courses he/she offers.
5. The lecturer sees the list of students who are registered in the course.

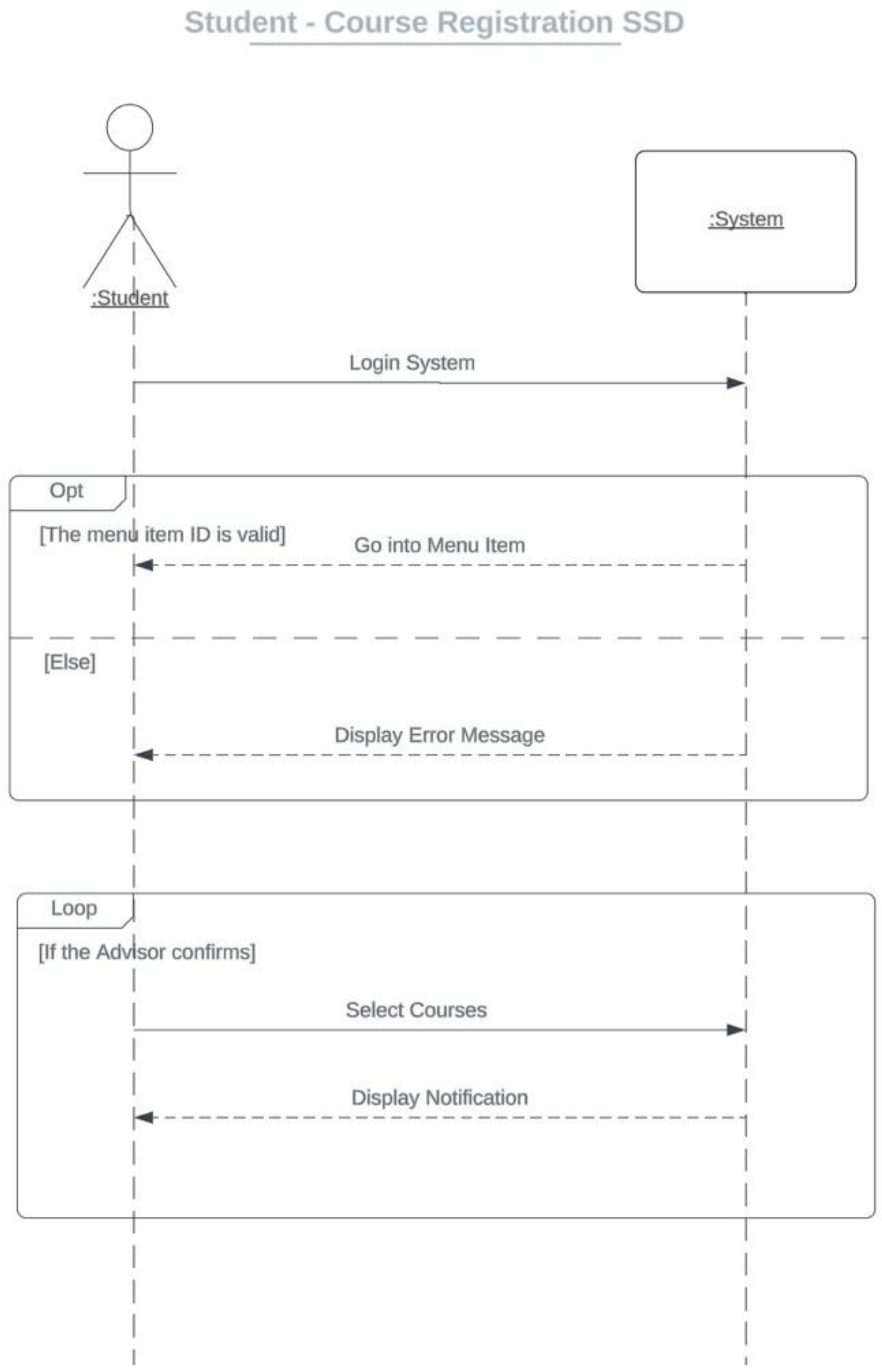
6. System Sequence Diagrams

Login SSD:

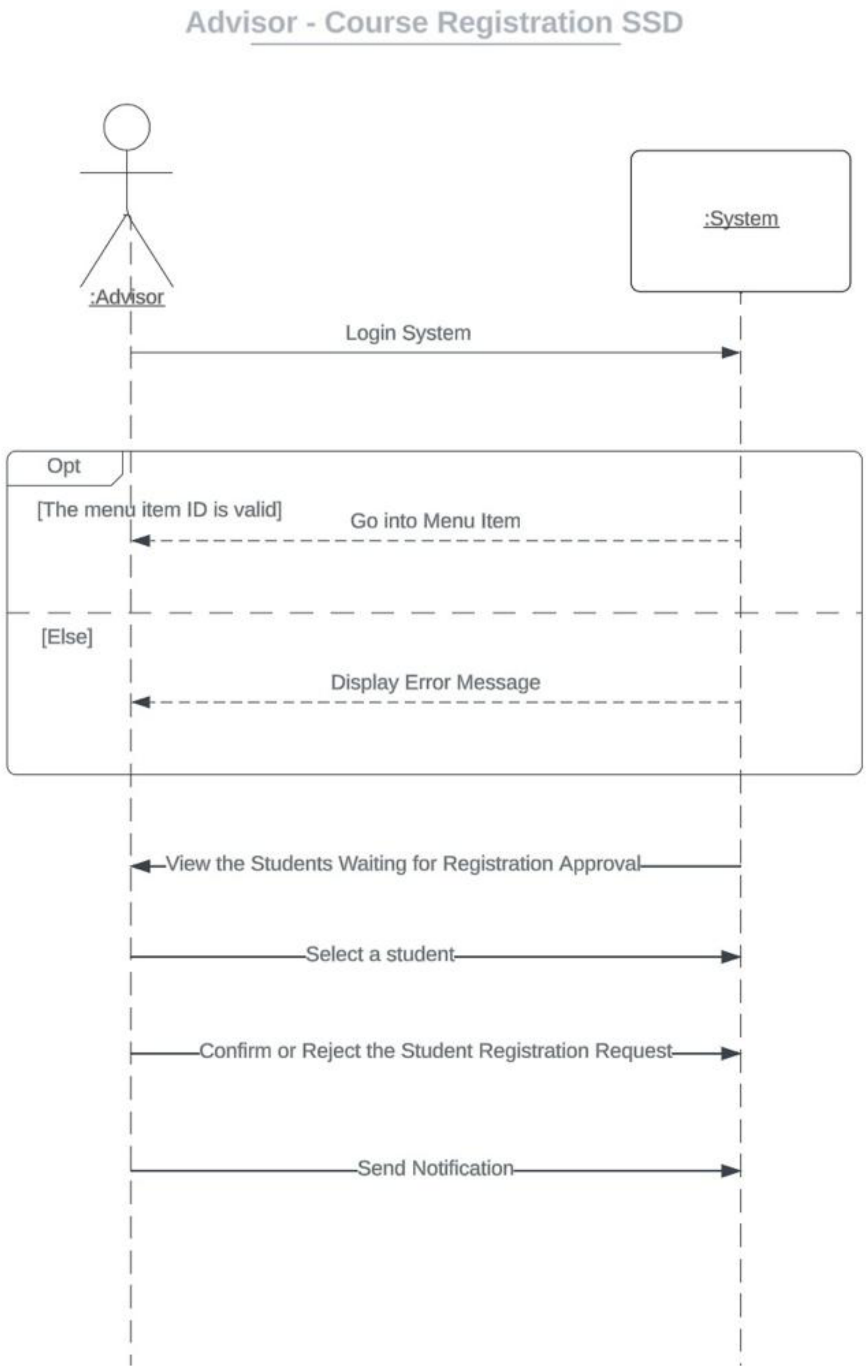


Login SSD
:User
:System
Check Password and Username
Opt
[If the username and password are valid]
Login Menu
[Else]
Display Login Error Message

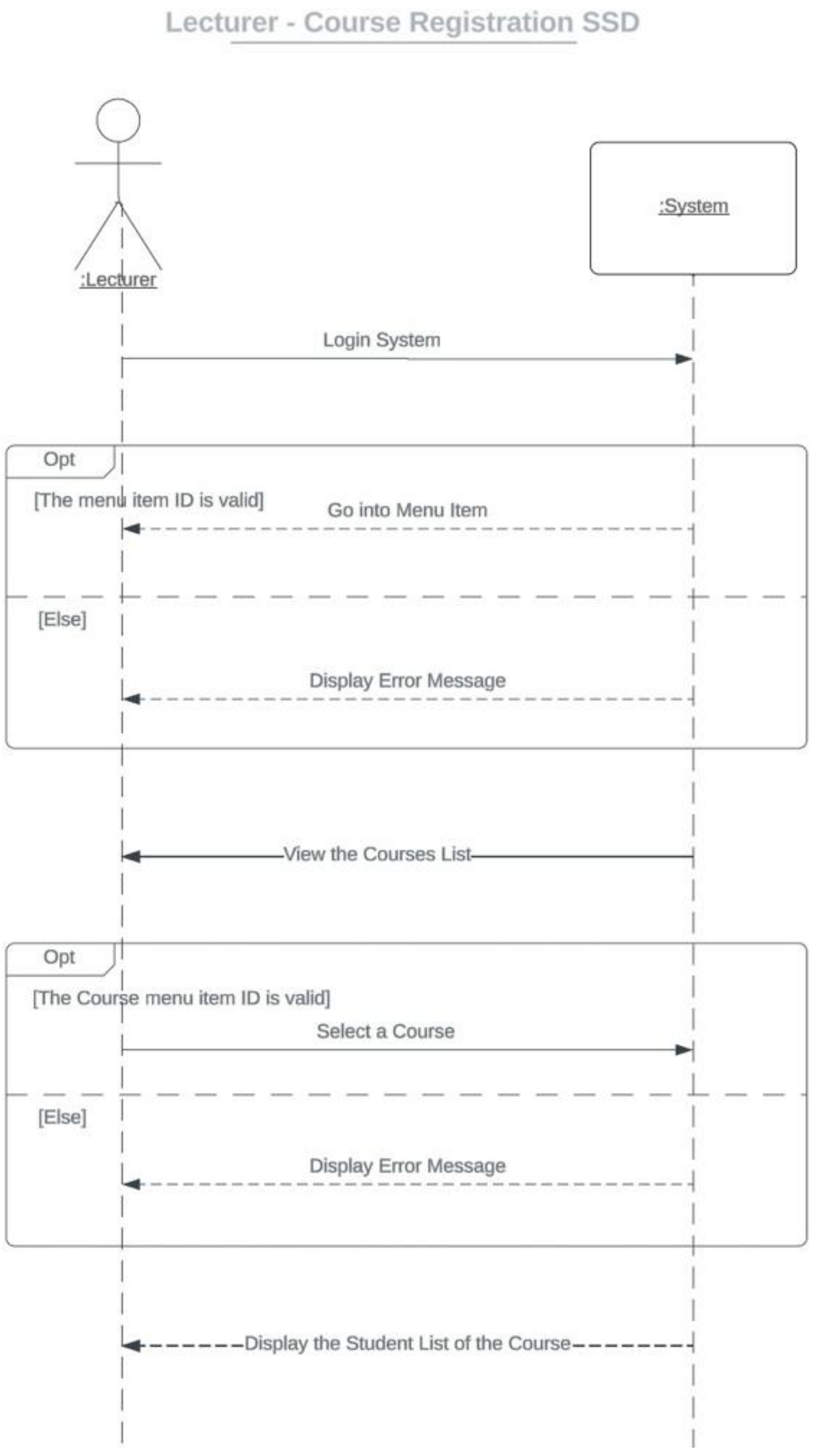
Student - Course Registration SSD:



Advisor - Course Registration SSD


Advisor - Course Registration SSD
:Advisor
:System
Login System
Opt
[The menu item ID is valid]
Go into Menu Item
[Else]
Display Error Message
View the Students Waiting for Registration Approval
Select a student
Confirm or Reject the Student Registration Request
Send Notification

Lecturer - Course Registration SSD



7. Team Work

7.1 Iteration 1

Semih Bağ (Group Representive):

- Preparing a use case for the rad file.
- Help with the dsd diagram prepared for the rad file.
- Contribution to uml diagram.
- Design and development of the interface that will communicate with the user.
- Development of the SystemMessage class, which is necessary to convey the inputs from the user interface to the system class.
- Development of the login page.
- Writing main menu pages that trigger the action the user wants to perform.
- Development of pages containing all courses, offered courses, selected courses, and approved courses for the student user.
- Developing pages for advisor type users to see and manage their students and course selection requests.
- Writing the function that allows the pages that the system should show to the user when the login process is successful to be produced and made ready.
- Writing a container function that analyzes the SystemMessage signals generated by the input information received by the user on each page and triggers the appropriate SystemClass functions to run with the necessary parameters. (listenUserInterface).
- Optimization and bug solutions.

Aksanur Konuk:

- Adding methods for determining the courses that a student can take in the current semester and managing the course selection process. (in Student Class)
- The Person and Password classes are created.
- Preparing Login System Sequence Diagram
- Preparing Student Course Registration System Sequence Diagram.
- Preparing DCD diagram.

Duru Baştunalı:

- Login confirmation by current user’s username and password for student and advisor.
- Reading all the students’ information from the JSON file and creating all student objects.

- Writing the changes of student information back to the associated JSON file when exiting the system.
- Preparing student JSON files for only 3rd year students.
- Preparing advisor and lecturer JSON files for CSE department instructors.
- Preparing courses JSON files for mandatory CSE courses.
- Preparing “Use Case UC1 : Login”.

Elife Kocabey:

- Creating a System Sequence Diagram (SSD) for the Student actor, based on the student use case.
- Creating the Transcript class.
- Creating the Student class, which extends from the Person class.
- Adding methods to Student class for creating a list of selectable courses by filtering the courses the student can take from the curriculum.

Eren Çetin:

- For Course Registration System, The advisor class is created. The advisor has an ability to approve or disapprove request coming from the student and set appropriate properties according to approve or disapprove.
- For Course Registration System, The lecturer class is created.
- For the Requirement Analysis Document, The glossary part is created.

Nidanur Demirhan:

- Reading the information required for the creation of Course objects from the JSON file.
- Reading the necessary information for the creation of Advisor objects from the JSON file.
- Reading the information required for the formation of Lecturer objects from the JSON file.
- Filling in missing information with the necessary functions due to the formation order of the objects.
- Writing the changing properties of course objects during execution to the required JSON file.
- Preparation of the "Use Case UC3: Advisor" file.

Zeynep Yılmaz:

- Creating a domain model for the rad file.
- Create a course class and create a method that keeps students enrolled in the course.
- Creating the courseSession class inherited from the Course class.
- Writing methods that create pages in the pageContentCreator class, these pages: selected courses, approved courses, all courses, selectable courses, menu for users.

7.2 Iteration 2

Semih Bağ (Group Representive):

- Contribution to uml diagram.
- Development of profile, password change, my lessons, notifications, transcript, course schedule, teacher main menu pages.
- Optimizing the pages prepared in the previous iter and getting rid of some unnecessary page types that cause repetition.
- Optimizing the inclusive listenUserInterface function developed in the previous iteration and adding the functions added in this iteration and making them ready for use.
- Suppressing the general error message, especially for errors that may occur during course selection.
- Developing the functions required to highlight the selected courses and making other courses affected by the selection appear in the appropriate color, and solving the bugs that arise in these functions.
- Optimization and bug solutions.

Aksanur Konuk:

- Adding new methods to incorporate additional functionalities into the student registration process. (Student Class)
- Adding Day enum.
- Preparing a use case for lecturer
- Preparing Student DSD
- Adjusting DCD diagram

Duru Baştunalı:

- Optimizing the code where it's found necessary.
- Login confirmation by current user's username and password for lecturer.
- Adjusting reading from JSON file due to additions to student JSON files.
- Adjusting writing to JSON file due to additions to student JSON files.
- Content of the pages are updated: selected courses, approved courses, all courses, selectable courses.
- Content of the pages are created: syllabus, transcript, notifications, lecturer's courses, lecturer main menu, profile.
- Adding more student JSON files for 2nd, and 4th year students.
- Adding more lecturer JSON files for various department instructors.
- Adding more courses JSON files for all courses including non-technical electives, technical electives and faculty electives.
- Adjusting UML Diagram, adding the new classes and transferring the updates.

Elife Kocabey:

- Creating a System Sequence Diagram (SSD) for the Lecturer actor based on the lecturer use case.
- Updating the Student and Advisor SSDs.
- Updating the Domain Model.
- Adding a method to the Transcript class that calculates the total credits completed to date and changing some variables.
- Adding several control methods into the Student class to refine the course selection process. These controls include checking course conflicts, checking GPA for taking upper-level courses, and limitations on the number of courses based on their type (mandatory, non-technical, technical, faculty).
- Adding Hour enum.

Eren Çetin:

- The Syllabus class is created to show a weekly schedule for each person in the system (Student, Advisor, Lecturer). Person, Lecturer, Advisor and Student classes are updated for Syllabus class.
- The CourseSchedule class is created to keep track of courses to days and time of the week it occurs.
- The advisor class is updated for sending notification to the student for informing.

- For the Requirement Analysis Document, The glossary part is updated and Lecturer DSD is created.
- The unit tests of advisor, syllabus are added.

Nidanur Demirhan:

- Updating the course object formation due to new features added to the course object.
- Assigning days and hours to all courses.
- Calculating GPA for Students.
- Updating changes in users' password information in JSON file.
- Creating each of the Creation methods as new classes and optimizing the SystemDomain class.
- Preparing unit tests for CreateCourse and CreateAdvisor classes.

Zeynep Yılmaz:

- Writing methods for password checks in the Password class.
- Adding more student JSON files for 1st and 2nd grade students.
- Updating the DSD part in the rad file.
- Update functional requirements.
- Updating the classes I am responsible for in the DCD section.
- Writing unit tests of password checking methods in the Password class.
- Writing unit tests for student class.

7.3 Iteration 3

Semih Bağ (Group Representive):

- Converting the source codes of all pages from java to python
- Converting the listenUserInterface function in SystemClass from Java to Python
- Converting PageType Enum to Python.
- Converting FunctionType Enum to Python.
- Converting SystemMessage class to Python.
- Converting UserInterface class to Python.
- Help with bug and error solutions

Aksanur Konuk:

- Converting Mark Enum to Python.
- Converting CourseType Enum to Python.
- Converting Id class to Python.
- Converting Student class methods to Python.
- Converting test_Student unit test to Python.

Duru Baştunalı:

- Varying student and advisor JSONs.
- Re-designing login methods as Polymorphism: Person has an abstract login method. All Person classes (Student, Advisor, Lecturer) are implementing this abstract method in different ways.
- Added logging for CreateStudent's Exception handling: if an error occurs in a student JSON file, log the file and continue execution without that student.
- Converting CreateStudent into Python class.
- Converting CreateAdvisor into Python class.
- Converting CreateLecturer test into Python unit test.
- Converting CreatePage into Python code.

Elife Kocabey:

- Converting Hour Enum to Python.
- Converting Grade Enum to Python.
- Converting Person class to Python.
- Converting Syllabus class to Python.
- Converting Transcript class to Python.
- Converting GradeClass class to Python.
- Converting Student class methods to Python.

Eren Çetin:

- Converting Day Enum to Python.
- Converting Advisor class to Python.
- Converting Lecturer class to Python.
- Converting test_Advisor unit test to Python.
- Converting test_Syllabus unit test to Python.
- Adding test_Lecturer unit test to Python.

Nidanur Demirhan:

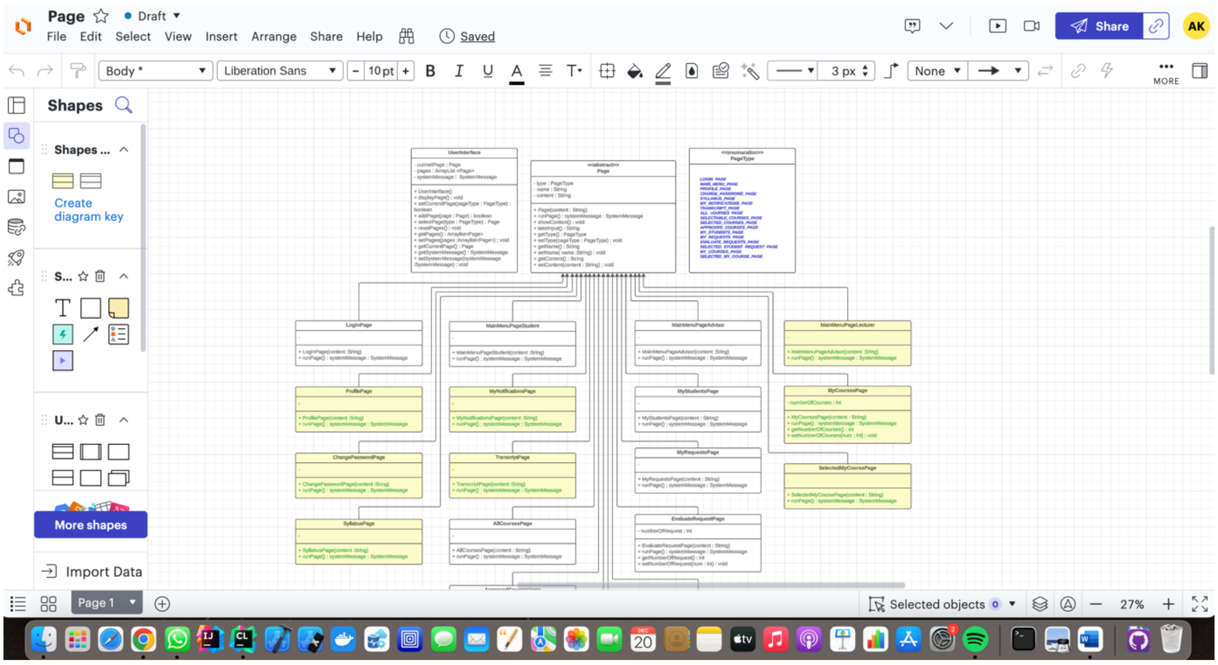
- Converting CreateCourse class into Python.
- Converting CreateLecturer class into Python.
- Converting JSON file updating methods in SystemClass into Python.
- Converting CreateCourse unit test into python unit test.
- Converting CreateCourse unit test into python unit test.
- Create equals methods for person objects.Create polymorphic equals method for course objects.
- Add logging for CreateCourse and updating in JSON file errors.

Zeynep Yılmaz:

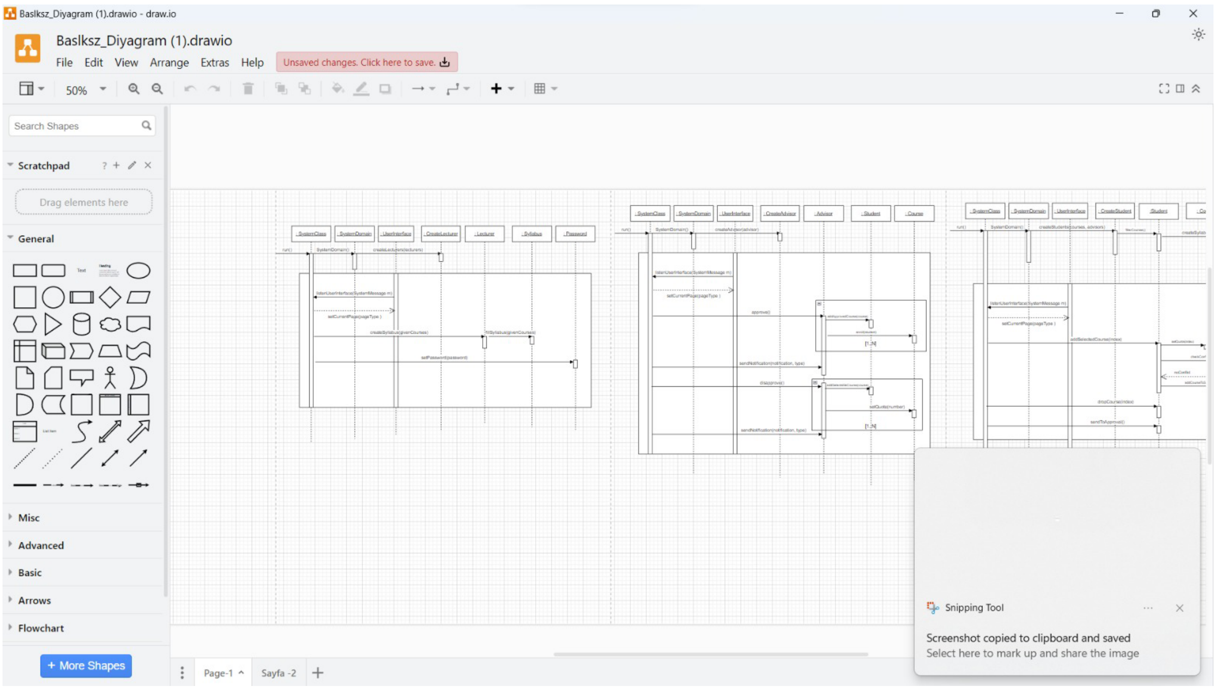
- Converting Course class to python
- Converting CourseSession class to python
- Converting Password class to python
- Writing a Password test unit
- Logging in Student class

7.4 Tools

LucidChart:



Draw.io:



Unsaved changes. Click here to save.

Screenshot copied to clipboard and saved
Select here to mark up and share the image