Marmara University Engineering Faculty
Department of Computer Engineering

CSE3063 – Object Oriented Software Design

Course Registeration System
Requirement Analysis Document (RAD)

Instructor: Murat Can Ganiz

November 17, 2023

STUDENTS – GROUP 16

BURAK KARAYAĞLI - 150121824
BARIŞ GİRAY AKMAN - 150121822
EFE ÖZGEN - 150121077
HAKKI KOKUR - 150120033
MUSA ÖZKAN - 150121058
TAMER ÜNAL - 150119795
ATİLLA GEL - 150119564

1 Introduction

The name of the project is Course Registration System. Main purpose in the project is providing users to get assigned to the necessary lectures through their advisors and simplifying the registration process of the course registration.

Users will enter the system through the user name and password given during the registration to the system. To enable the course at the system, there must be at least one lecturer to lecture. Otherwise, course won't be enabled for that semester. Each student who have taken any of the courses will be graded through the system at the end of the semester. And these grades will be processed through transcript.

2 Requirement Analysis

2.1 Vision

Course registration system for computer engineering department. It is a system for student's choosing courses and waiting for approval from the advisors.

2.2 Glossary

2.2.1 Advisor

Definition: The person authorized to approve students' course registrations. Advisor is also the person that students can consult with in case of any problems.
Purpose: Advisors play a crucial role in ensuring that students make informed decisions about their academic journey, providing guidance and support.
Example: A student consults their advisor to discuss their course selections and academic goals.

2.2.2 Lecturer

Definition: Person responsible for teaching courses.
Purpose: Lecturers deliver course content, guide students in their studies, and contribute to the overall learning experience.
Example: The lecturer conducts lectures, facilitates discussions, and assesses student understanding through assignments and exams.

2.2.3 Transcript

Definition: Document showing the student's course grades, credits, etc.
Purpose: Transcripts serve as official records of a student's academic performance, documenting courses taken, grades received, and overall progress.
Example: A student submits their transcript when applying for a job or pursuing further education.

2.2.4 Course Section

Definition: Details of the class, time, and lecturers for the courses.
Purpose: Course sections provide specific information about when and where a course is held, as well as who is responsible for its instruction.
Example: A student checks the course section to find out the schedule and location of a particular class.

2.2.5 JSON

Definition: Notation used for data storage.
Purpose: JSON (JavaScript Object Notation) is a lightweight data-interchange format used to store and exchange information between systems.
Example: A web application uses JSON to transmit data between the server and client in a structured format.

2.2.6 Course Registration

Definition: The process of courses being selected by students and approved by advisors.
Purpose: Course registration enables students to choose the classes they wish to attend, while also ensuring that advisors can review and approve their selections.
Example: During course registration, a student selects their desired classes through the university's online portal, and the advisor approves the choices.

2.3 Requirements

2.3.1 Functional Requirements

- The system must allow students to register for courses
- The system must allow advisors to approve or deny course registrations
- The system must allow students to view their transcripts
- The system must allow students to change their registration
- The system must allow students to drop courses
- The system must allow students to take courses that have prerequisites
- The system must allow students to view their grades
- The system must allow lecturer to add courses
- The system must allow users to log in to the system
- The system must allow users to log out from the system
- The system must allow users to view course information

2.3.2 Non-Functional Requirements

- The system must accept at max 5 courses for each student
- The system must inform the user at the end of operations
- The system should maintain detailed logs os system activities and errors for debugging and auditing purposes
- The system should comply with relevant data protection and privacy regulations

3 Use Cases

3.1 Use Case: Login

Steps:

Actor Actions	System Responses
1. Enter the username and user password	1. Check the username and user password
	2. Get the user information
	3. Print the user information page
	4. Print the main menu

3.2 Use Case: Attempt to login with invalid username and password

Steps:

Actor Actions	System Responses
1. Enter the username and user password	2. Check the username and user password
	3. Show warning “user don’t exist”
4. Enter the correct username and password	5. Check and redirect to information page

3.3 Use Case: Register in Course

Steps:

Actor Actions	System Responses
1. Select the courses	Display the selected course
2. Send the selected courses	3. Test the selection is valid or not
	4. Print the stage information

3.4 Use Case: Attempt to take more courses than allowed

Steps:

Actor Actions	System Responses
1. Select the courses	
2. Send the selected courses	3. Test the selection is valid or not
	4. Print the stage information

3.5 Use Case: Approve the Courses

Steps:

Actor Actions	System Responses
1. Select the waiting approval list from menu	2. List the waiting approval
	3. Give some selection (approve, deny, skip)
4. Make selection	5. Implement the selection and go step 2

3.6 Use Case: Find Course Info

Steps:

Actor Actions	System Responses
1. Select the find course info from menu	2. List courses abbreviation and give an input area
3. Write the course abbreviation	4. Search the course detail
	5. If course exists, then show detail or give empty text

3.7 Use Case: Give Course

Steps:

Actor Actions	System Responses
1. Select given course from menu	2. List the course that can given by him/her
3. Write a course that wanna give	4. Add the written course to selectable list
5. Write the quota	6. Go to step 2

3.8 Use Case: Add Course

Steps:

Actor Actions	System Responses
1. Select add course from menu	2. Print the what need to adding course
3. Write the course full name	
4. Write the course short name	
5. Write the description of course	6. Save the new course informations

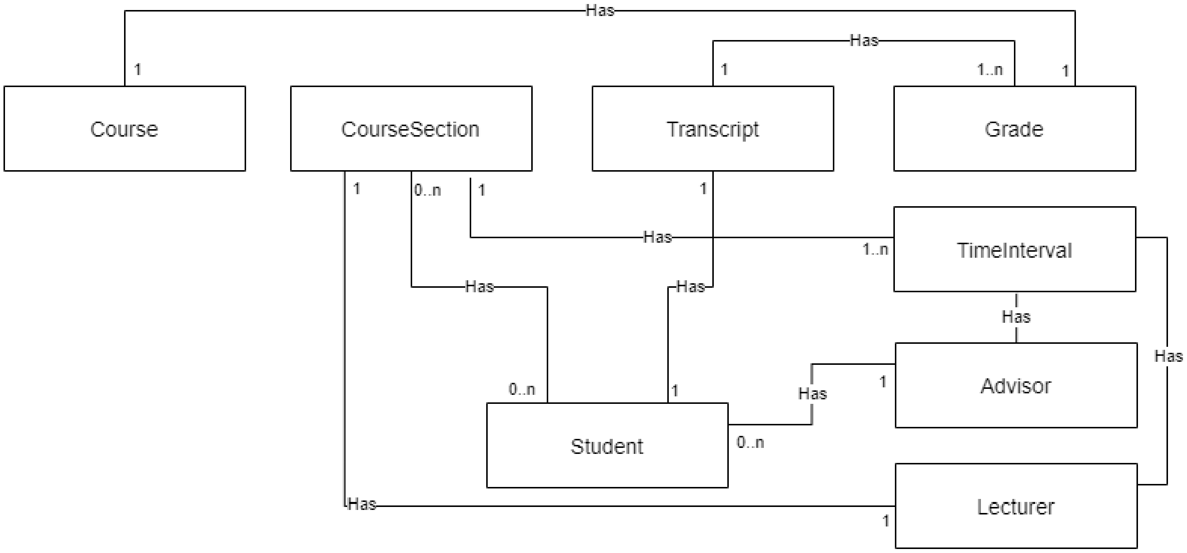
3.9 Use Case: Logout

Steps:

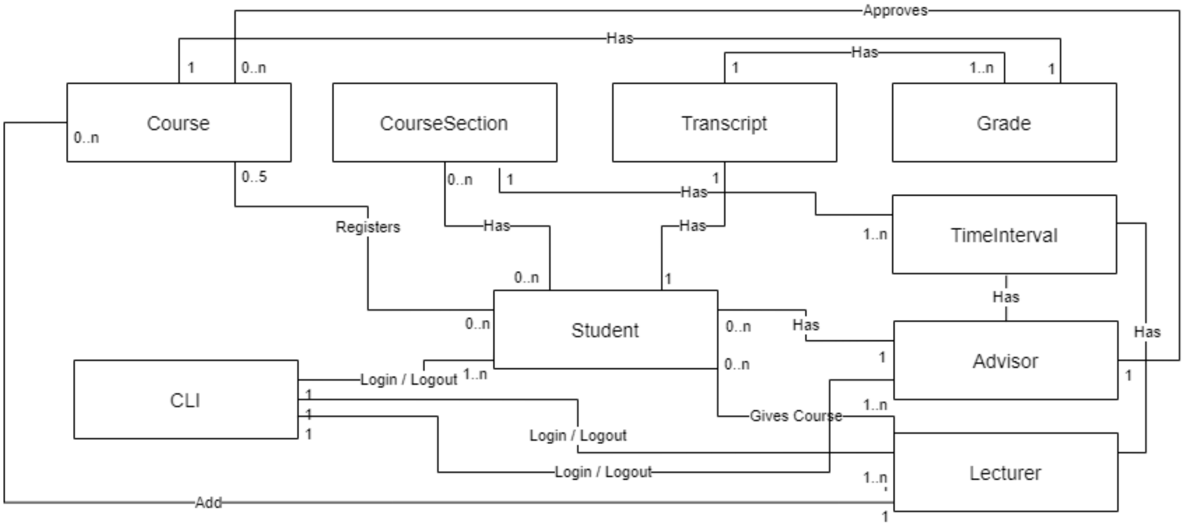
Actor Actions	System Responses
1. Select the logout from menu	2. Select the logout from menu
	3. Print the login screen

4 Diagrams and Models

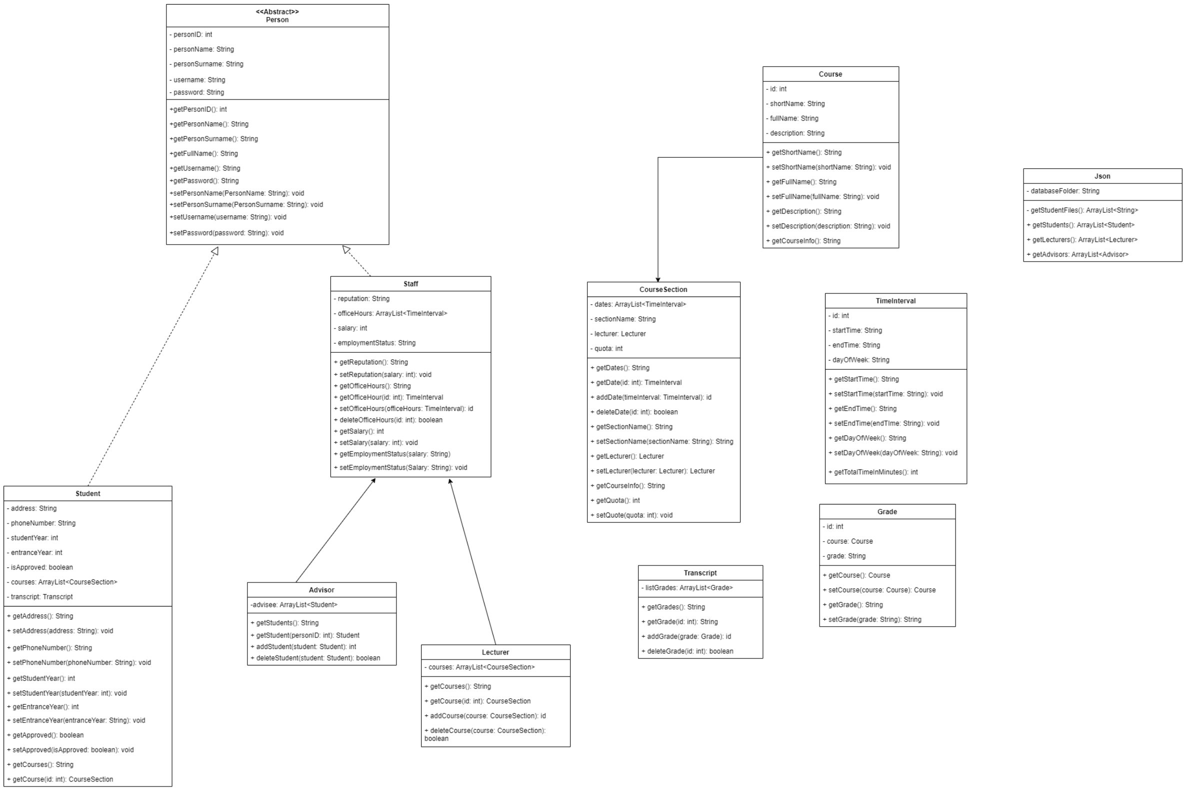
4.1 Domain Model - Without Concepts



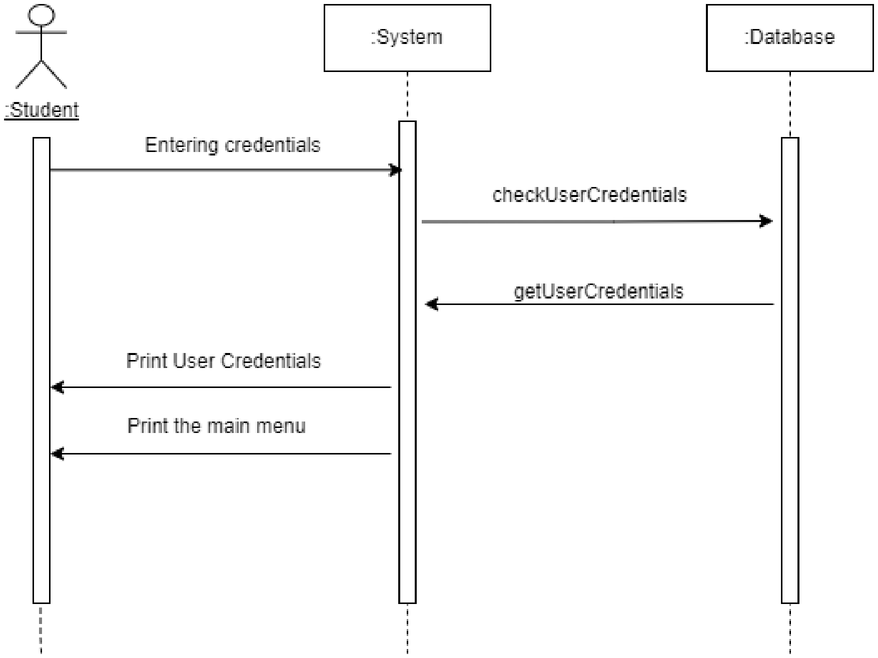
4.2 Domain Model - With Concepts

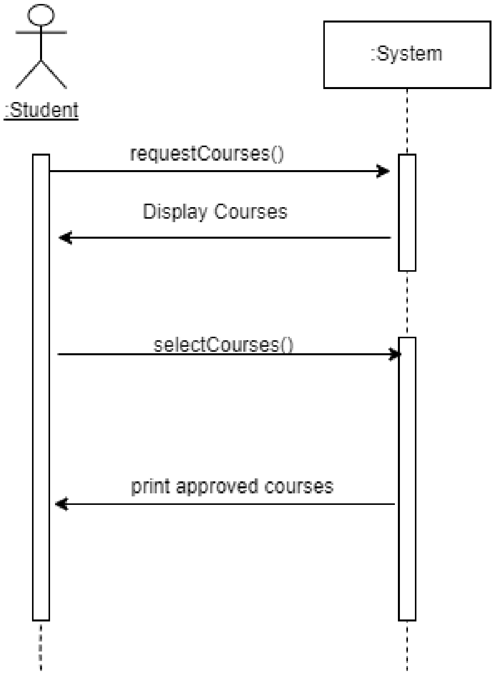


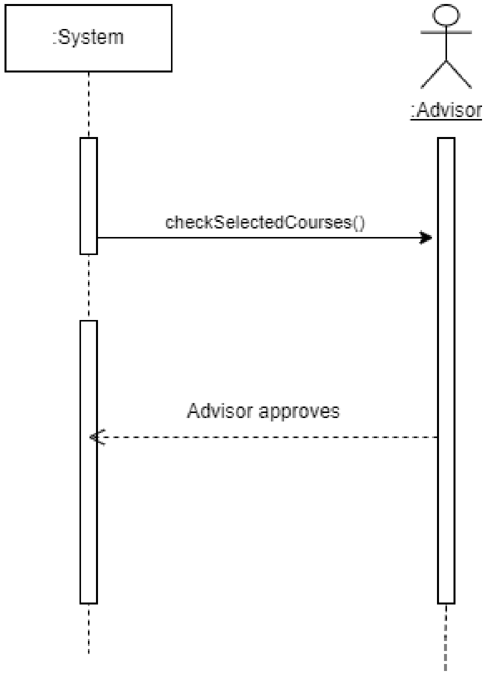
4.3 Design Class Diagram (DCD)


<<Abstract>>
Person
- personID: int
- personName: String
- personSurname: String
- username: String
- password: String
+getPersonID(): int
+getPersonName(): String
+getPersonSurname(): String
+getFullName(): String
+getUsername(): String
+getPassword(): String
+setPersonName(PersonName: String): void
+setPersonSurname(PersonSurname: String): void
+setUsername(username: String): void
+setPassword(password: String): void
Course
- id: int
- shortName: String
- fullName: String
- description: String
+ getShortName(): String
+ setShortName(shortName: String): void
+ getFullName(): String
+ setFullName(fullName: String): void
+ getDescription(): String
+ setDescription(description: String): void
+ getCourseInfo(): String
Json
- databaseFolder: String
- getStudentFiles(): ArrayList<String>
+ getStudents(): ArrayList<Student>
+ getLecturers(): ArrayList<Lecturer>
+ getAdvisors: ArrayList<Advisor>
Staff
- reputation: String
- officeHours: ArrayList<TimeInterval>
- salary: int
- employmentStatus: String
+ getReputation(): String
+ setReputation(salary: int): void
+ getOfficeHours(): String
+ getOfficeHour(id: int): TimeInterval
+ setOfficeHours(officeHours: TimeInterval): id
+ deleteOfficeHours(id: int): boolean
+ getSalary(): int
+ setSalary(salary: int): void
+ getEmploymentStatus(salary: String)
+ setEmploymentStatus(Salary: String): void
CourseSection
- dates: ArrayList<TimeInterval>
- sectionName: String
- lecturer: Lecturer
- quota: int
+ getDates(): String
+ getDate(id: int): TimeInterval
+ addDate(timeInterval: TimeInterval): id
+ deleteDate(id: int): boolean
+ getSectionName(): String
+ setSectionName(sectionName: String): String
+ getLecturer(): Lecturer
+ setLecturer(lecturer: Lecturer): Lecturer
+ getCourseInfo(): String
+ getQuota(): int
+ setQuote(quota: int): void
TimeInterval
- id: int
- startTime: String
- endTime: String
- dayOfWeek: String
+ getStartTime(): String
+ setStartTime(startTime: String): void
+ getEndTime(): String
+ setEndTime(endTIme: String): void
+ getDayOfWeek(): String
+ setDayOfWeek(dayOfWeek: String): void
+ getTotalTimeInMinutes(): int
Student
- address: String
- phoneNumber: String
- studentYear: int
- entranceYear: int
- isApproved: boolean
- courses: ArrayList<CourseSection>
- transcript: Transcript
+ getAddress(): String
+ setAddress(address: String): void
+ getPhoneNumber(): String
+ setPhoneNumber(phoneNumber: String): void
+ getStudentYear(): int
+ setStudentYear(studentYear: int): void
+ getEntranceYear(): int
+ setEntranceYear(entranceYear: String): void
+ getApproved(): boolean
+ setApproved(isApproved: boolean): void
+ getCourses(): String
+ getCourse(id: int): CourseSection
Grade
- id: int
- course: Course
- grade: String
+ getCourse(): Course
+ setCourse(course: Course): Course
+ getGrade(): String
+ setGrade(grade: String): String
Transcript
- listGrades: ArrayList<Grade>
+ getGrades(): String
+ getGrade(id: int): String
+ addGrade(grade: Grade): id
+ deleteGrade(id: int): boolean
Advisor
-advisee: ArrayList<Student>
+ getStudents(): String
+ getStudent(personID: int): Student
+ addStudent(student: Student): int
+ deleteStudent(student: Student): boolean
Lecturer
- courses: ArrayList<CourseSection>
+ getCourses(): String
+ getCourse(id: int): CourseSection
+ addCourse(course: CourseSection): id
+ deleteCourse(course: CourseSection): boolean

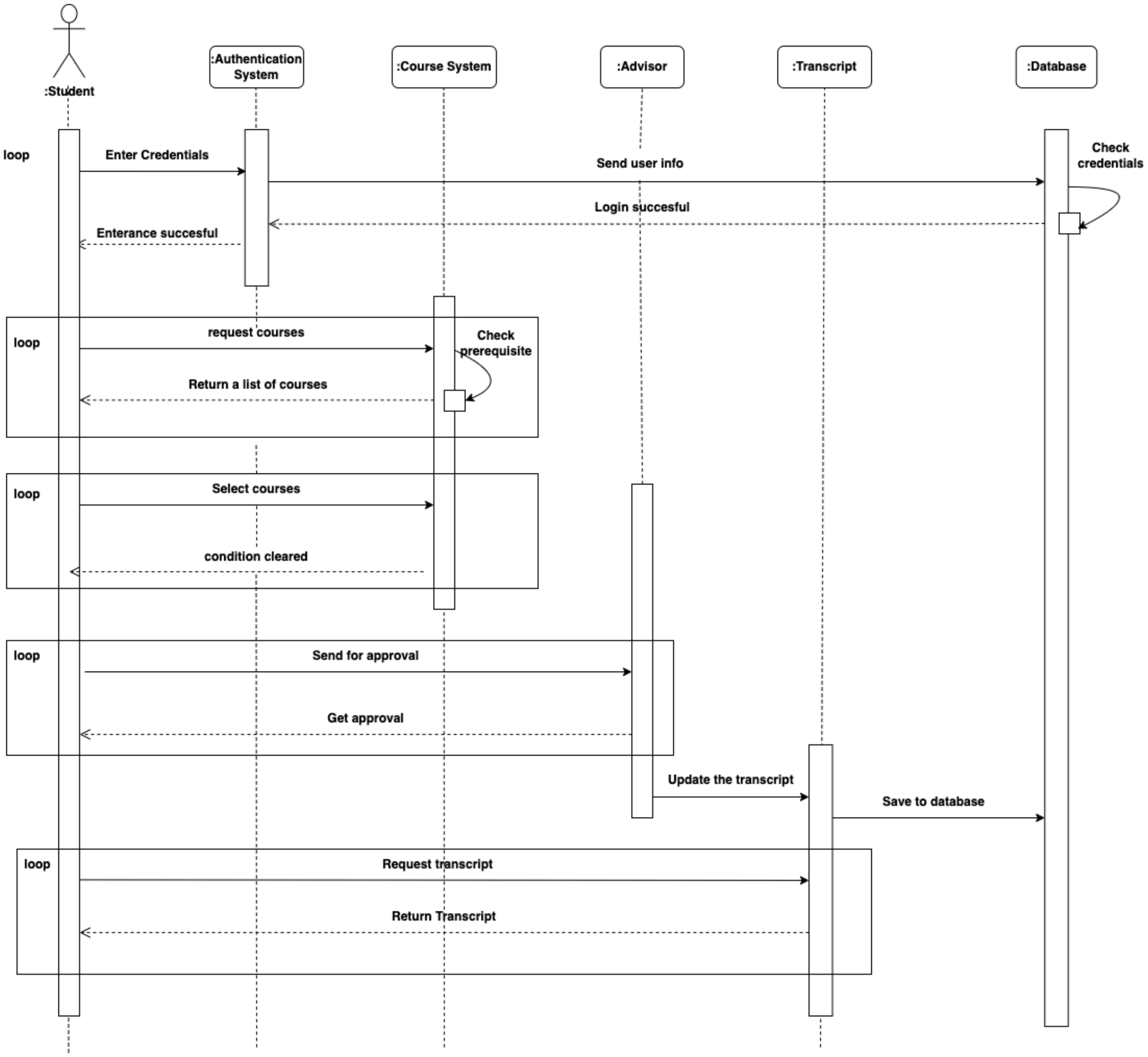
4.4 System Sequence Diagram (SSD)







4.5 Design Sequence Diagrams (DSD)


:Student
:Authentication System
:Course System
:Advisor
:Transcript
:Database
loop
Enter Credentials
Send user info
Check credentials
Login succesful
Enterance succesful
loop
request courses
Check prerequisite
Return a list of courses
loop
Select courses
condition cleared
loop
Send for approval
Get approval
Update the transcript
Save to database
loop
Request transcript
Return Transcript