SWE209 Object Oriented Analysis and Design Homework 2

Ozan Gül B211202036

STUDENT INFORMATION SYSTEM

REQUIREMENTS ANALYSIS DOCUMENT (RAD)

1. Introduction

1.1 Purpose

It is a Requirement Analysis Document (RAD) for new web-based student information system for a private university. It is a new document created to make the design of an innovative, simpler, uncomplicated system easier for students, academicians and university administrators. In this way, a system has been designed where students can easily follow their course schedules, grades and make course selections. It is aimed for academicians and administrators to be able to manage, grade ,view students', to send information and give assignments to students, to send course material as online in an easy interface. It is an easily accessible and easily understandable system that provides convenience to both students and academicians, especially when online exams need to be held and online courses need to be given. With this RAD described purpose,scope,objectives, current system, proposed system, functional and non-functional requirements, with modeling use case diagrams, and end of this RAD, prepared glossary for the new student information system.

1.2 Scope

The scope of this project is to manage students by academicians using a web browser (web-based), to provide effective online education (when necessary), to keep student data, to send course materials, to conduct online exams, to track courses and grades by students, or to update grades by teachers,to provide online access assignments given to students (for both academicans and students). to view absances students online in a system.

1.3 Objectives

Objectives are the student information system to:

- Those who use the system (academicians, students, admin (rectorate)) will be able to access this system over the internet (This system will be web-based.)
- The system provides an interface where academicians can view their own data, their students' data, send course materials to the system and change their students' data.
- The system provides an interface where students can view their assignments,their grades, submit their assignments online,to apply any course or exam like english exam proficiency application and take midterms and quizzes online.
- The system allows academicians to deliver courses online and present lecture recordings in a way that students can watch and repeat the course later.

2. Current System

In the system this private university currently uses, students can only view their own grades. Students hand-deliver their assignments. Students experience a lot of problems during assignment submission and hand delivery. The university does not have any online education system. For this reason, it is not among the universities that support online education during state of emergency events. Likewise, all exams are conducted face to face. This university does not have any alternatives regarding these issues. Students can only view their transcripts and course schedules. The design of this university's system is not suitable for user interface design testing. Non-technological structures were used for the user interface. In interface design, the advanced structure of programming languages or design patterns are used incompletely and inadequately. The interface and database are inadequate in terms of HTML, CSS, Javascript and SQL.

3. Proposed System

3.1 Overview

The new developed system will use software design pattern. It will use HTML, CSS and Javascript structures at an advanced level for design. In order to use the data in the database more efficiently, Microsoft SQL server can be used to process SQL commands more smoothly and save them in the system not only to view them, but also to view course resources, view and send assignments.

3.2 Functional Requirements

- The system should allow *students* to view their course schedules on a weekly, daily and monthly basis.
- The system should provide an interface where *students* can view their scores (quiz, midterm, final) and transcripts.
- The system should provide *academicians* with an interface where *students* can enter their grades.
- The system should create an attended list that allows *academicians* to check whether *students* are absent or not. And *academicians* enter their attandance.
- The system provides an interface where *academicians* can upload course materials.
- The system should create an interface for *academicians* to teach online courses when necessary.
- The system should store *students'* data in the database. (Midterm Exam, Quiz Grades, Attendance information)
- The system should enable *students* to log into online courses.
- The system should keep course records in the database.
- The system should create an interface where *students* can watch old course recordings and upload course materials at any time.
- The system creates an interface for *students* to apply for subjects other than their compulsory courses (for example English exemption exam application).
- The system allows accessibility for the *administrator*. It provides a platform for the *administrator* to supervise and control *students* and *academicians*.
- The system should create an interface where *academicians* create a adding exams to study for exams other than regular exams in the application screen(interface which created by the system)

3.3 Nonfunctional Requirements

Security

- The system must ensure the security of the data in the database.
- The system must support highly secure passwords for academicians,students,system administrator to log in.

Supportability

- Management of the system should be conducted easily.
- Operations such as system control and maintenance must be completed effortlessly and professionally.

Usability

- The user interface should be simple, clear and understandable. Academics and students should not waste time searching for their information.

Performance

- When the academician enters the student's grades, the system should transfer them to the interface where the students will see them within 3 seconds.
- The system should directly transmit live and online lessons to students without any delay.

Availability

- The system must remain open at 99% activity within a day. And the system must be accessible every second.

Implementation

- The system must be implemented in MSSQL Database. Data must be transferred corrently and consistent.
- The system must be implemented using HTML,CSS,Javascript and design patterns, CSS should be implemented in the front-end.

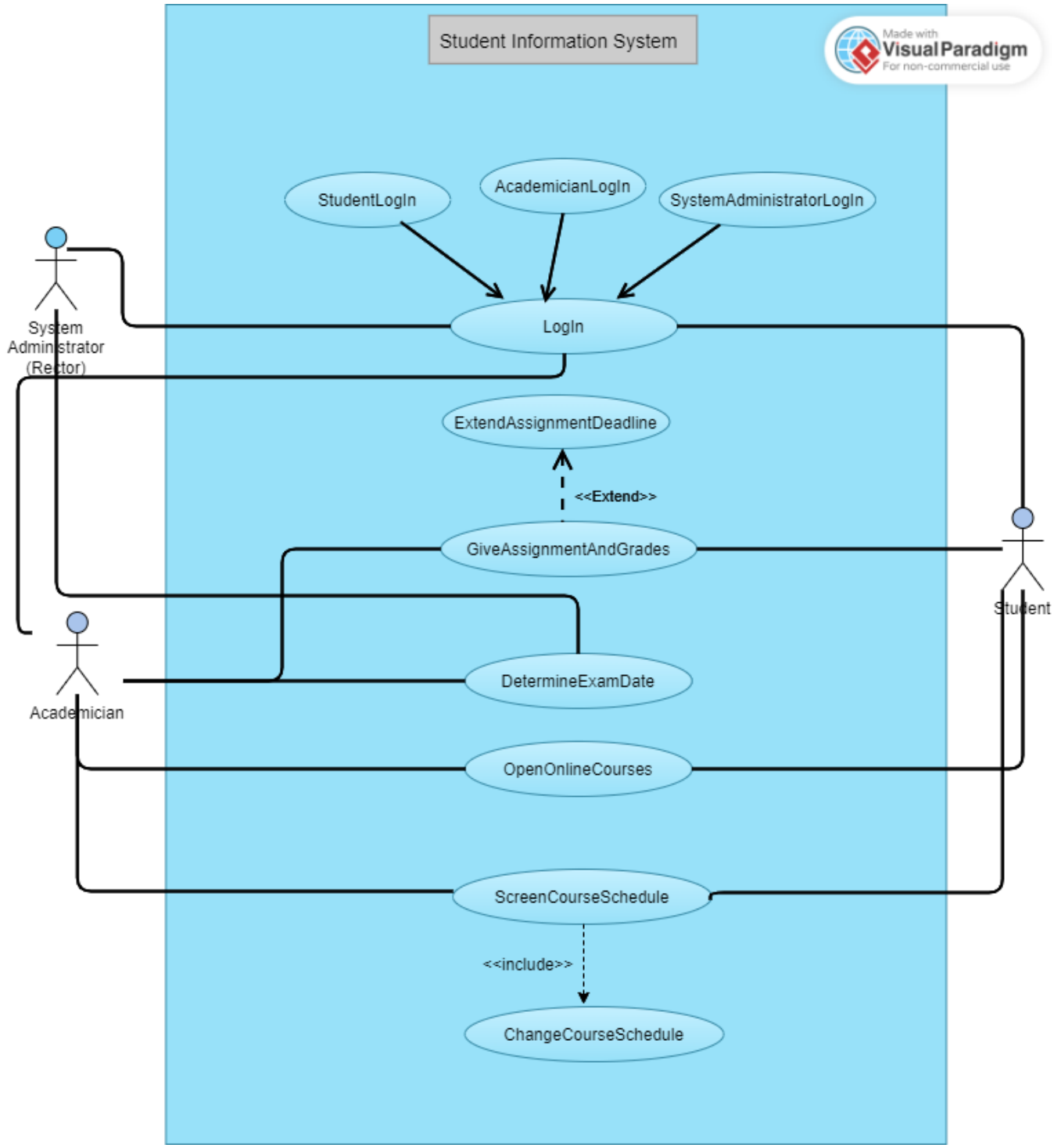
3.4 System Models

3.4.1 Scenarios

This is a scenario example about the system.

Scenario name	Applying for an Exam
Participating actor instances	Ozan:Student Mehmet:Academician Hamza:Administrator
Flow of events	1. Ozan wants to apply for exemption from the English exam in order not to take English courses in the future. 2. Mehmet adds the English exemption exam applications to the application screen and organizes the exam. 3. Ozan logs in as a student to the application screen and applies using the English Proficiency Exam application button. 4. Hamza checks whether Ozan is qualified to apply for the exemption exam. 5. Hamza contacts Mehmet regarding Ozan's exam entry. 6. Mehmet accepts Ozan to take the exam.

3.4.1 Use Case Models



Use case name	OpenOnlineCourses
Participating actors	Initiated by *Academician* Communicated by *Student*
Flow of events	1. The *Academician* logged into the *System* and request the open the interface to open online courses. 2. The *Academician* open the new online course from OpenOnlineCourse button. And the *System* send notification to the *Student.* 3 . The *Student* read the notification and request the *Academician* to join the online courses. 4. The *Academician* approve the request of the *Student.*
Entry conditions	The *Academician* logged into the System to open new online course.
Exit conditions	The Student watched the online courses and logged out from the *System.*
Quality requirements	The notification about the new online courses opening must be sent instantly from the *System*. The actions taken by the *Academician* in the online course should be conveyed to the *Student* with a delay of no more than 1 second. (If the *Student* or *Academician* has no internet problems).

Use case name	DetermineExamDate
Participating actors	Initiated by *Academician* Communicated by *System Administrator*
Flow of events	1. The *Academician* wants to do quiz or exam for *Students* whether they are enough the course information. 2. The *Academician* logged into the *System* and click the “Exam Date“ button (as operation). After that, the *Academician* click “Create Exam Date” button selecting the type of the exam from the interface which the *System provide.* When the *Academician* completed the operation, the *System* send notification to the *System Administrator.* 3. The *System Administrator* get the notification and enter the *System.* The *System Administrator* checks the date, confirms or disapproves. The *System* send notification the *Academician* about the result. 4 . The *Academician* checks the notification. Logs into the system. If there is no problem, the *Academician* exits the system. If the exam is not approved, the use case may repeat.
Entry conditions	The *Academician* logged into the system to create new exam date.
Exit conditions	The *Academician* logged out the system if there is no problem about the exam date.
Quality requirements	When the *Academician* creates a new exam date, the *System* should be up-to-date, checking whether the *Students* have any classes on that date.

Use case name	GiveAssignmentAndGrades
Participating actors	Initiated by *Academician* Communicated by *Student*
Flow of events	1. The *Academician* prepare the assignment for the *Student* and request *System* to open Assignment Interface. 2. The *Academician* upload the assignment to the *System*. And *System* send notification to the *Student.* 3. The *Student* take the notification and logged into the *System.* The *Student* download the assignment. The *Student* complete the assignment. 4. The *Student* logged into the System. The *Student* request The *System* to open Assignment Interface for student. The *Student* upload the assignment. 5. The *Academician* take the assignment and control it. 6. The *Academician* logged into the *System* again. And send request the *System* to open the grades page. *The Academician* give grades to the *Student* for assignments.
Entry conditions	The *Academician* logged into the *System* for giving assignment.
Exit conditions	The *Academician* give grades for assignments and logged out from the *System.*
Quality requirements	Notification that the assignment has been uploaded to the *System* should be sent to the *Student* within 10 seconds. There should be no problems when the *Student* upload the assignment.

Use case name	ChangeCourseSchedule
Participating actors	Initiated by *Academician* Communicated by *Student*
Flow of events	1. The *Academician* must change the course schedule for any problem. The *Academician* logged into the *System.* 2. The *Academician* send request firstly the *System* to screen the course schedule using " ScreenCourseSchedule" use case (as a button). After that, The *Academician click to the "ChangeCourseSchedule"* button. The *System get the* request and response it successfully. 3.The *Academician* changed the course schedule. When this operation completed, the *System* send notification all the *Student* who take this course. 4. The *Student* take the notification and enter the system to control course schedule. The *Student* click the "Course Schedule" button. The *System* get the request and response it successfully. *5.* The *Student* can see the interface of new course schedule.
Entry conditions	The *Academician* entered the *System* to change the course schedule.
Exit conditions	The *Student* checked the new course schedule in detailed and logged out from the *System.*
Quality requirements	Response of the *System* must be within 2 second. The System must remain up to date in case of changes to the course program. This authorization must be given only Academician.

Use case name	LogIn
Participating actors	*Student* *Academician* *System Administrator*
Flow of events	1. A user (*Student, Academician, System Administrator) request* the *System* to log in. The *System* return different login page for every user. 2. User enter the username and password. *The system* verifies it. 3. The user successfully logs in to the system.
Entry conditions	The user (*Student, Academician, System Administrator) request* the *System* to log in.
Exit conditions	The user successfully logs in to the system.
Quality requirements	During verification, the system should ask users 2FA verification.

Use case name	StudentLogIn
Participating actors	**Inherited** from Authenticate use case. (Student)
Flow of events	1. The *Student request* the *System* to log in. The *System* return different login page for every user. 2. The *Student* enter the username and password. *The system* verifies it. 3. The *Student* successfully logs in to the system.
Entry conditions	**Inherited** from Authenticate use case
Exit conditions	**Inherited** from Authenticate use case
Quality requirements	**Inherited** from Authenticate use case

Use case name	AcademicianLogIn
Participating actors	**Inherited** from Authenticate use case. (Academician)
Flow of events	1. The *Academician request* the *System* to log in. The *System* return different login page for every user. 2. The *Academician* enter the username and password. *The system* verifies it. 3. The *Academician* successfully logs in to the system.
Entry conditions	**Inherited** from Authenticate use case
Exit conditions	**Inherited** from Authenticate use case
Quality requirements	**Inherited** from Authenticate use case

Use case name	SystemAdministratorLogIn
Participating actors	**Inherited** from Authenticate use case. (System Administrator)
Flow of events	1. The *System Administrator request* the *System* to log in. The *System* return different login page for every user. 2. The *System Administrator* enter the username and password. *The system* verifies it. 3. The *System Administrator* successfully logs in to the system.
Entry conditions	**Inherited** from Authenticate use case
Exit conditions	**Inherited** from Authenticate use case
Quality requirements	**Inherited** from Authenticate use case

Use case name	ScreenCourseSchedule
Participating actors	Initiated by a user (Student, Academician, System Administrator)
Flow of events	1. The user wants to see the course schedule of belonging a course. The user logged into the *System.* 2. The user send request the *System* to screen the course schedule using " Screen Course Schedule" button. The *System get the* request and response it successfully. 3. The *user* can see the interface of a course schedule.
Entry conditions	The *user* (*Student, Academician, System Administrator*)entered the *System* to screen the course schedule.
Exit conditions	The *user* (*Student, Academician, System Administrator*)checked the course schedule in detailed and logged out from the *System.*
Quality requirements	Response of the *System* must be within 2 second. The System must remain up to date in case of changes to the course program.

Use case name	ExtendAssignmentDeadline
Participating actors	Initiated by *Student* Communicated by *Academician*
Flow of events	1. The *Student* logs into the System. On the assignment details page, the *Student* clicks a button labeled "Request Extension." Upon clicking, the *System* receives confirmation from the *Student* indicating a request for an extension. The *System* processes the extension request and notifies the Academician. 2. The *Academician* reviews the extension request. 3. The *Academician* approves or rejects the request. The *System* notifies the *Student* about the status of their request. If the request is approved, the *Student* receives information about the new deadline. If the request is rejected, the *Student* receives information about the reasons for rejection.
Entry conditions	The *Student* entered the *System* to request extend assignment deadline.
Exit conditions	The *Student* logs out from the System after receiving the notification.
Quality requirements	If there is an issue processing the extension request or sending notifications, the System should provide a clear error message to the student.

4. GLOSSARY

Academician: An academician is a person who grades students and conduct face-to-face and online education. Communicate with admin and helps manage students.

Student: A Student is the person who takes exams and takes notes. The student can log into the system, review course notes, and apply for the exam. Student can review the course schedule.He is the person who can send and receive assignments by communicating with the academician.

System Administrator: Administrator is the rector of the school. Administrator is the person who supervises students and academicians.

GiveAssignmentAndGrades: It is the operation where the academician gives homework to the student using the homework assignment interface, evaluates and grades the homework delivered on time.

ChangeCourseSchedule: It refers to the ability of the academician to change the course schedule one time in case of any problem or situation that requires postponement.

OpenOnlineCourses: It is a part of the online education system that allows the Academician to communicate with students online and give lectures via the system interface in any situation that requires online education.

DetermineExamDate: It is the process of determining the distribution of a required exam by the academician through the system and approving the communication with the admin using the system.

Log In: It refers to logging into the system for each user. (System Administrator, Academician, Student)

ScreenCourseSchedule: It is button that actively displays the course schedule for any user. It is a use case.

ExtendAssignmentDeadline: is a use case button given to the academician to extend the duration of the assignment for students who cannot complete the assignment.
