

2009

King Saud University

[REGISTRARS' SUBSYSTEM]

Subsystem that helps managing the work of registration employees in King Saud University.

Abstract

This text is directed to give a detailed description about the various steps that took place during the building of this subsystem. Realizing the importance of their work, we tried to facilitate their work through introducing a group of functionalities, including the following:

- Registering new students.
- Maintaining courses.
- Maintaining sections.
- Sending and receiving messages with managerial and educational board.
- Assigning courses to students and teachers.

The following sections will give both technical and non-technical readers a comprehensive idea about how this system built, and what functions can it introduce.

Contents

<i>Abstract</i>	2
Table of Figures	5
1. <i>Introduction</i>	6
2. <i>System Analysis</i>	7
2.1 Project Services and Scope	7
2.2 Use Case Diagram	8
2.3 Use Case Estimation	9
2.4 Release Planning	10
2.5 Task Assignment	10
2.6 Use Case Description	12
2.6.1 Register Student	12
2.6.2 Delete Student	13
2.6.3 Update Student Info	14
2.6.4 View Messages	15
2.6.5 Add Course	16
2.6.6 Edit Course	17
2.6.7 Delete Course	18
2.6.8 Add Section	19
2.6.9 Edit Section	20
2.6.10 Delete Section	21
2.6.11 Register Student in Course	22
2.6.12 Register Teacher to Course	23
2.6.13 Send Messages	24
3. <i>System Design</i>	25
3.1 Sequence Diagrams	25
3.1.1 Register Student	25
3.1.2 Delete Student	26
3.1.3 Update Student Info	27

3.1.4	View Messages	28
3.1.5	Add Course	29
3.1.6	Edit Course.....	30
3.1.7	Delete Course	31
3.1.8	Add Section.....	32
3.1.9	Edit Section.....	33
3.1.10	Delete Section	34
3.1.11	Register Student in Course	35
3.1.12	Register Teacher in Course	36
3.1.13	Send Messages	37
3.2	ER Diagram	38
3.3	Database Table Format	39
3.4	Class Description	40
3.5	Class Diagram	42
4.	<i>System Implementation</i>	43
5.	Test Cases.....	44
6.	Conclusion and Future Work	57

Table of Figures

Figure 1: Use Case Diagram	8
Figure 2: Sequence, Register Student	25
Figure 3: Sequence, Delete Student	26
Figure 4: Sequence, Update Student Info	27
Figure 5: Sequence, View Messages	28
Figure 6: Sequence, Add Course	29
Figure 7: Sequence, Edit Course	30
Figure 8: Sequence, Delete Course	31
Figure 9: Sequence, Add Section	32
Figure 10: Sequence, Edit Section	33
Figure 11: Sequence, Delete Section	34
Figure 12: Sequence, Register Student in Course	35
Figure 13: Sequence, Register Teacher in Course	36
Figure 14: Sequence, Send Messages	37
Figure 15: ER Diagram	38
Figure 16: Database Table Format	39
Figure 17: Class Diagram	42

1. Introduction

One of the most important tasks in university system is the tasks of managing students' enrollment and management of courses and sections. In this, we are talking about the tasks that registrars perform. Noticing the great job that those employees do, we decided; after the approval of our supervisors, to build a system that helps them complete their work in a quick and precise way. Of course, this system is a part of a bigger system that will include the various aspects of work in the university.

Briefly, this document will focus on the following points:

- System Analysis.
- System Design.
- System Implementation.
- System Testing.

2. System Analysis

2.1 Project Services and Scope

Actor	UC ID	UC Name	Use Case Brief Description
Registrar	1	Register Student	Register new Student
Registrar	2	Delete Student	Delete current student
Registrar	3	Update Student Info	Updating current student Information
Registrar	4	View Messages	Read received messages
Registrar	5	Add Course	Add New Course
Registrar	6	Edit Course	Updating course information
Registrar	7	Delete Course	Deleting current course
Registrar	8	Add Section	Adding new section
Registrar	9	Edit Section	Updating current section
Registrar	10	Delete Section	Deleting current section
Registrar	11	Register Student in Course	Assigning student to course
Registrar	12	Register Teacher in Course	Assigning teacher to course
Registrar	13	Send Message	Send notifications to students and teachers

2.2 Use Case Diagram

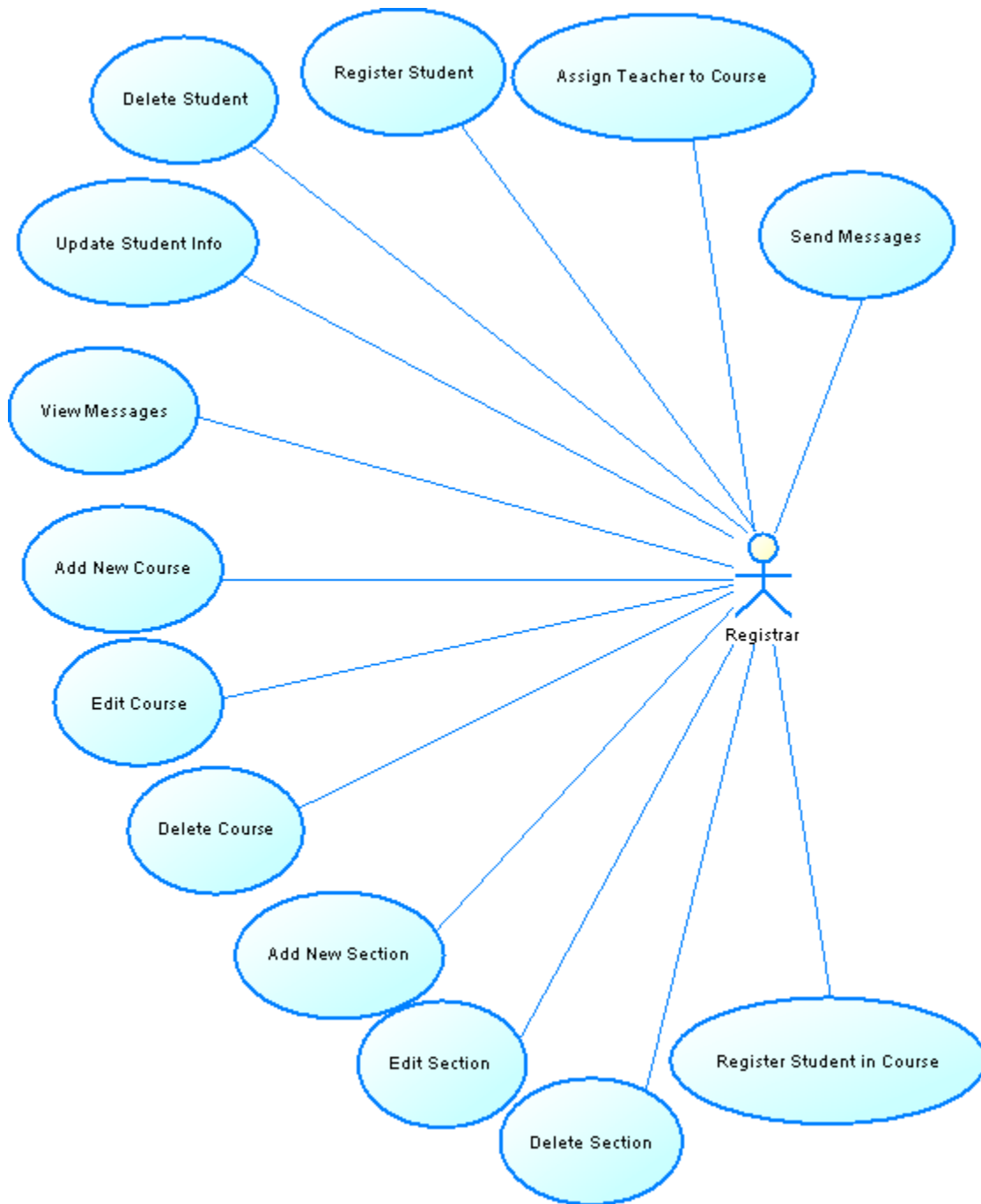


Figure 1: Use Case Diagram

2.3 Use Case Estimation

UC ID	UC Name	Development	Measurements: Development(Hours)						
			A	D	C	T	Doc	L	Total
1	Register Student	7	1	1	2	1	1	1	7
2	Delete Student	6	1	1	1	1	1	1	6
3	Update Student Info	6	1	1	1	1	1	1	6
4	View Messages	6	1	1	1	1	1	1	6
5	Add Course	7	1	1	2	1	1	1	7
6	Edit Course	7	1	1	2	1	1	1	7
7	Delete Course	7	1	1	2	1	1	1	7
8	Add Section	7	1	1	2	1	1	1	7
9	Edit Section	6	1	1	1	1	1	1	6
10	Delete Section	6	1	1	1	1	1	1	6
	Register Student in Course	6	1	1	1	1	1	1	6
	Register Teacher in Course	6	1	1	1	1	1	1	6
	Send Messages	6	1	1	1	1	1	1	6
Project total			13	13	18	13	13	13	83

2.4 Release Planning

Release#	Actor	UCID	UC Name	Priority
1	Registrar	1	Register Student	H
1	Registrar	2	Delete Student	H
1	Registrar	3	Update Student Info	H
1	Registrar	4	View Messages	H
2	Registrar	5	Add Course	M
2	Registrar	6	Edit Course	M
2	Registrar	7	Delete Course	M
2	Registrar	8	Add Section	M
2	Registrar	9	Edit Section	M
2	Registrar	10	Delete Section	M
3	Registrar	11	Register Student in Course	L
3	Registrar	12	Register Teacher in Course	L
3	Registrar	13	Send Messages	L

2.5 Task Assignment

Student ID	Student Name	Task
		Abstract
		Introduction
		Project Services and Scope
		Use Case Diagram
		Use Case Estimation
		Release Planning
		Task Assignment

		System Architecture
		Use Case Description
		Sequence Diagrams
		ER Diagram
		Database Table Format
		Class Description
		Class Diagram
		System Implementation
		System Testing

2.6 Use Case Description

2.6.1 Register Student

System: Registrars' Subsystem	UC ID: 1.
Use Case Name: Register Student.	Priority: H.
Primary Actor: Registrar.	Stakeholders: None.
Brief Description: Register new student in university.	
Trigger: Button Click. Trigger Type: External.	
Relationships: <ul style="list-style-type: none">- Includes: None.- Association: None.	
Input: None.	
Pre-conditions: System Running.	
Normal flow events: <ul style="list-style-type: none">• Registrar enters the new student info.• System validates the registrar entries.• Registrar requests the registration operation.• The system performs the addition operation and returns a confirmation.	
Actor	System
<ol style="list-style-type: none">1- Registrar enters the new student info.2- The registrar requests the registration operation.	<ol style="list-style-type: none">1- The System validates the registrar entries.2- System submits the information to the database and returns a confirmation.
Post-conditions: New student registered in university.	
Output: Confirmation of data entry.	
Test Case: 1.	

2.6.2 Delete Student

System: Registrars' Subsystem	UC ID: 2.
Use Case Name: Delete Student.	Priority: H.
Primary Actor: Registrar.	Stakeholders: None.
Brief Description: Deletes a current student from the system.	
Trigger: Button Click. Trigger Type: External.	
Relationships: <ul style="list-style-type: none">- Includes: None.- Association: None.	
Input: None.	
Pre-conditions: System Running, Student Existence.	
Normal flow events: <ul style="list-style-type: none">• Registrar selects a student and requests the deletion.• System deletes the student from database and returns a confirmation.	
Actor	System
1- Registrar selects a student and requests the deletion.	1- System deletes the student from the database and returns a confirmation.
Post-conditions: Student has no longer access to the system.	
Output: System displays a confirmation of the deletion.	
Test Case: 2.	

2.6.3 Update Student Info

System: Registrars' Subsystem	UC ID: 3.
Use Case Name: Update Student Info.	Priority: H.
Primary Actor: Registrar.	Stakeholders: None.
Brief Description: Updates the information of a current student.	
Trigger: Button Click, Input. Trigger Type: External.	
Relationships: - Includes: None. - Association: None.	
Input: student updated information.	
Pre-conditions: System Running, Student Existence.	
Normal flow events: <ul style="list-style-type: none"> • Registrar selects the student and system puts the data in change mode. • Registrar enters the new data. • System validates the registrar entries and submits changes to the database. 	
Actor	System
<ol style="list-style-type: none"> 1- Registrar selects a student. 2- Registrar enters the new data. 	<ol style="list-style-type: none"> 1- System puts the data in update mode. 2- System validates the entries and submits the data to database.
Post-conditions: New student information is available.	
Output: Confirmation of data updating.	
Test Case: 3.	

2.6.4 View Messages

System: Registrars' Subsystem	UC ID: 4.
Use Case Name: View Messages.	Priority: H.
Primary Actor: Registrar.	Stakeholders: None.
Brief Description: View received information.	
Trigger: Button Click. Trigger Type: External.	
Relationships: <ul style="list-style-type: none">- Includes: None.- Association: None.	
Input: None.	
Pre-conditions: System Running.	
Normal flow events: <ul style="list-style-type: none">• Registrar selects a message.• System displays the information of the selected message.	
Actor	System
1- Registrar selects a message.	1- System displays the message information.
Post-conditions: None.	
Output: None.	
Test Case: 4.	

2.6.5 Add Course

System: Registrars' Subsystem	UC ID: 5.
Use Case Name: Add Course.	Priority: M.
Primary Actor: Registrar.	Stakeholders: None.
Brief Description: Add new course to system.	
Trigger: New Input, Button Click. Trigger Type: External.	
Relationships: - Includes: None. - Association: None.	
Input: None.	
Pre-conditions: Selecting the right college to add the course to.	
Normal flow events: <ul style="list-style-type: none"> • Registrar inserts the information of the new course. • System validates the entries of registrar. • Registrar requests the addition of the course. • System adds the information to the database and confirms it. 	
Actor	System
<ol style="list-style-type: none"> 1- Registrar enters the information of the new course. 2- Registrar requests the addition of the course. 	<ol style="list-style-type: none"> 1- System displays the message information. 2- System adds the information and confirms the addition.
Post-conditions: New course available for teachers and students.	
Output: Addition confirmed by the system.	
Test Case: 5.	

2.6.6 Edit Course

System: Registrars' Subsystem	UC ID: 6.
Use Case Name: Edit Course.	Priority: M.
Primary Actor: Registrar.	Stakeholders: None.
Brief Description: Edit current course information.	
Trigger: New Input, Button Click. Trigger Type: External.	
Relationships: <ul style="list-style-type: none"> - Includes: None. - Association: None. 	
Input: Updated information of the course.	
Pre-conditions: None.	
Normal flow events: <ul style="list-style-type: none"> • Registrar selects a course, and system displays the current course information. • Registrar enters the new information, and system submits the entries to the database and submits it. 	
Actor	System
1- Registrar selects a course. 2- Registrar enters the new information and request updating.	1- System displays the message information. 2- System validates the entries and submits changes to database.
Post-conditions: None.	
Output: Confirmation of data updating.	
Test Case: 6.	

2.6.7 Delete Course

System: Registrars' Subsystem	UC ID: 7.
Use Case Name: Delete Course.	Priority: M.
Primary Actor: Registrar.	Stakeholders: None.
Brief Description: Delete current course from the system.	
Trigger: Course Selection, Button Click. Trigger Type: External.	
Relationships: <ul style="list-style-type: none">- Includes: None.- Association: None.	
Input: None.	
Pre-conditions: System Running, Course Existence.	
Normal flow events: <ul style="list-style-type: none">• Registrar selects a course and requests deletion.• System performs the deletion and confirms the deletion.	
Actor	System
1- Registrar selects a course and requests the deletion.	1- System performs the deletion and confirms the deletion.
Post-conditions: Course is no longer available for teachers and students.	
Output: System confirms the deletion.	
Test Case: 7.	

2.6.8 Add Section

System: Registrars' Subsystem	UC ID: 8.
Use Case Name: Add Section.	Priority: M.
Primary Actor: Registrar.	Stakeholders: None.
Brief Description: Add new section to system.	
Trigger: New Input, Button Click. Trigger Type: External.	
Relationships: - Includes: None. - Association: None.	
Input: None.	
Pre-conditions: System Running.	
Normal flow events: <ul style="list-style-type: none"> • Registrar inserts the information of the new section. • System validates the entries of registrar. • Registrar requests the addition of the section. • System adds the information to the database and confirms it. 	
Actor	System
1- Registrar enters the information of the new section. 2- Registrar requests the addition of the section.	1- System displays the message information. 2- System adds the information and confirms the addition.
Post-conditions: New section available for teachers and students.	
Output: Addition confirmed by the system.	
Test Case: 8.	

2.6.9 Edit Section

System: Registrars' Subsystem	UC ID: 9.
Use Case Name: Edit Section.	Priority: M.
Primary Actor: Registrar.	Stakeholders: None.
Brief Description: Edit current section information.	
Trigger: New Input, Button Click. Trigger Type: External.	
Relationships: <ul style="list-style-type: none">- Includes: None.- Association: None.	
Input: Updated information of the section.	
Pre-conditions: System Running.	
Normal flow events: <ul style="list-style-type: none">• Registrar selects a section, and system displays the current section information.• Registrar enters the new information, and system submits the entries to the database and submits it.	
Actor	System
1- Registrar selects a section. 2- Registrar enters the new information and request updating.	1- System displays the message information. 2- System validates the entries and submits changes to database.
Post-conditions: None.	
Output: Confirmation of data updating.	
Test Case: 9.	

2.6.10 Delete Section

System: Registrars' Subsystem	UC ID: 10.
Use Case Name: Delete Section.	Priority: M.
Primary Actor: Registrar.	Stakeholders: None.
Brief Description: Delete current section from the system.	
Trigger: Button Click. Trigger Type: External.	
Relationships: <ul style="list-style-type: none">- Includes: None.- Association: None.	
Input: None.	
Pre-conditions: System Running.	
Normal flow events: <ul style="list-style-type: none">• Registrar selects a section and requests deletion.• System performs the deletion and confirms the deletion.	
Actor	System
1- Registrar selects a section and requests the deletion.	1- System performs the deletion and confirms the deletion.
Post-conditions: Section is no longer available for teachers and students.	
Output: System confirms the deletion.	
Test Case: 10.	

2.6.11 Register Student in Course

System: Registrars' Subsystem	UC ID: 11.
Use Case Name: Register Student in Course.	Priority: L.
Primary Actor: Registrar.	Stakeholders: None.
Brief Description: Assignment student in new course.	
Trigger: Data Selection, Button Click. Trigger Type: External.	
Relationships: - Includes: None. - Association: None.	
Input: None.	
Pre-conditions: System Running.	
Normal flow events: <ul style="list-style-type: none"> • Registrar selects the student and selects the course and requests registration. • System performs the registration and returns a confirmation. 	
Actor	System
1- Registrar selects a student and selects a course and requests registration.	1- System performs the registration and returns a confirmation.
Post-conditions: Student can attend to his new course.	
Output: System confirms the registration.	
Test Case: 11.	

2.6.12 Register Teacher to Course

System: Registrars' Subsystem	UC ID: 12.
Use Case Name: Register teacher in Course.	Priority: L.
Primary Actor: Registrar.	Stakeholders: None.
Brief Description: Assignment teacher in new course.	
Trigger: Data Selection, Button Click. Trigger Type: External.	
Relationships: - Includes: None. - Association: None.	
Input: None.	
Pre-conditions: System Running.	
Normal flow events: <ul style="list-style-type: none"> • Registrar selects the teacher and selects the course and requests registration. • System performs the registration and returns a confirmation. 	
Actor	System
2- Registrar selects a teacher and selects a course and requests registration.	2- System performs the registration and returns a confirmation.
Post-conditions: Teacher can attend to his new course.	
Output: System confirms the registration.	
Test Case: 12.	

2.6.13 Send Messages

System: Registrars' Subsystem	UC ID: 13.
Use Case Name: Send Messages.	Priority: L.
Primary Actor: Registrar.	Stakeholders: None.
Brief Description: Send notifications to students and teachers.	
Trigger: New Input, Button Click. Trigger Type: External.	
Relationships: - Includes: None. - Association: None.	
Input: None.	
Pre-conditions: System Running.	
Normal flow events: <ul style="list-style-type: none"> Registrar selects the receiver and enters the information of the message and requests sending. System performs the operation and returns a confirmation. 	
Actor	System
<ol style="list-style-type: none"> Registrar selects a receiver and enters the message information. Registrar requests the sending. 	<ol style="list-style-type: none"> System validates the entries of the registrar. System performs the operation and returns a confirmation.
Post-conditions: None.	
Output: System confirms the sending of the message.	
Test Case: 13.	

3. System Design

3.1 Sequence Diagrams

3.1.1 Register Student

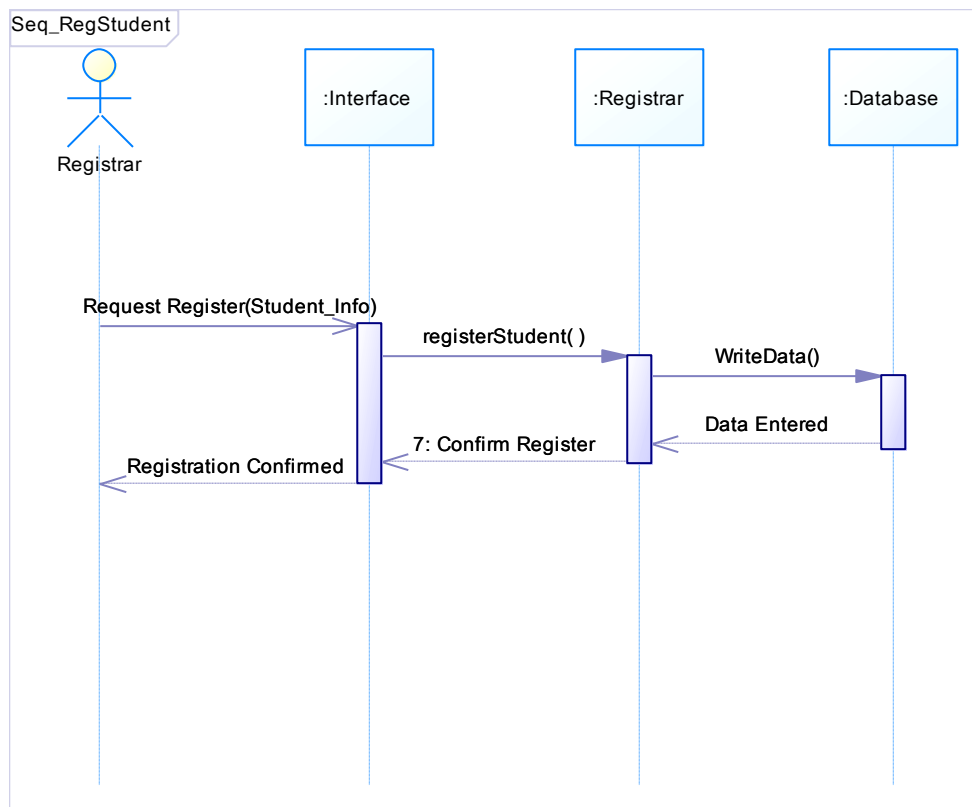


Figure 2: Sequence, Register Student

3.1.2 Delete Student

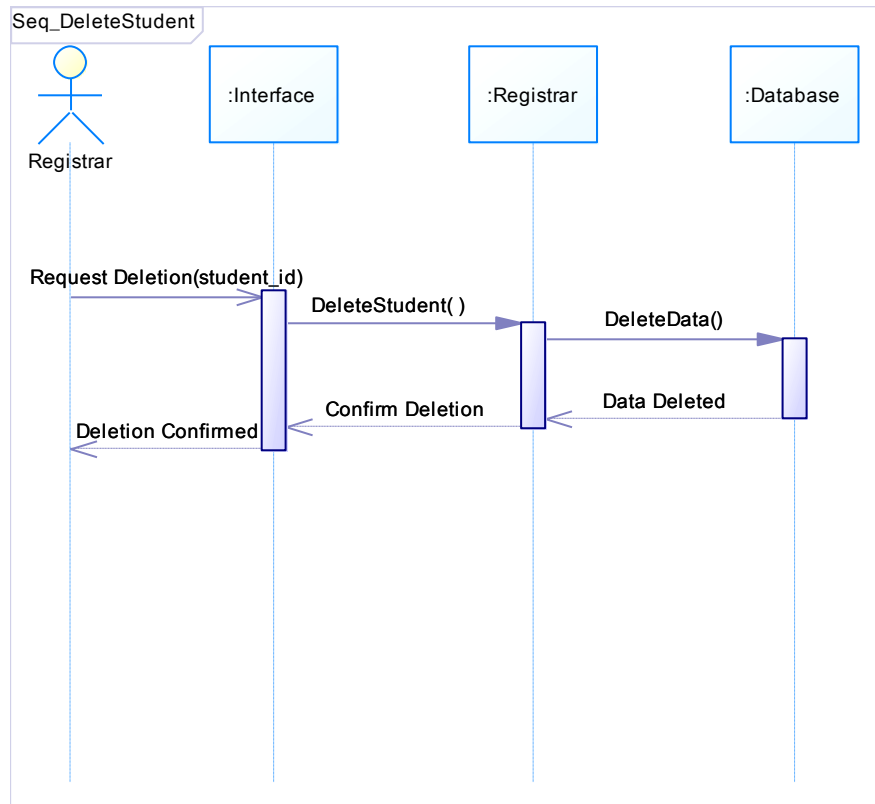


Figure 3: Sequence, Delete Student

3.1.3 Update Student Info

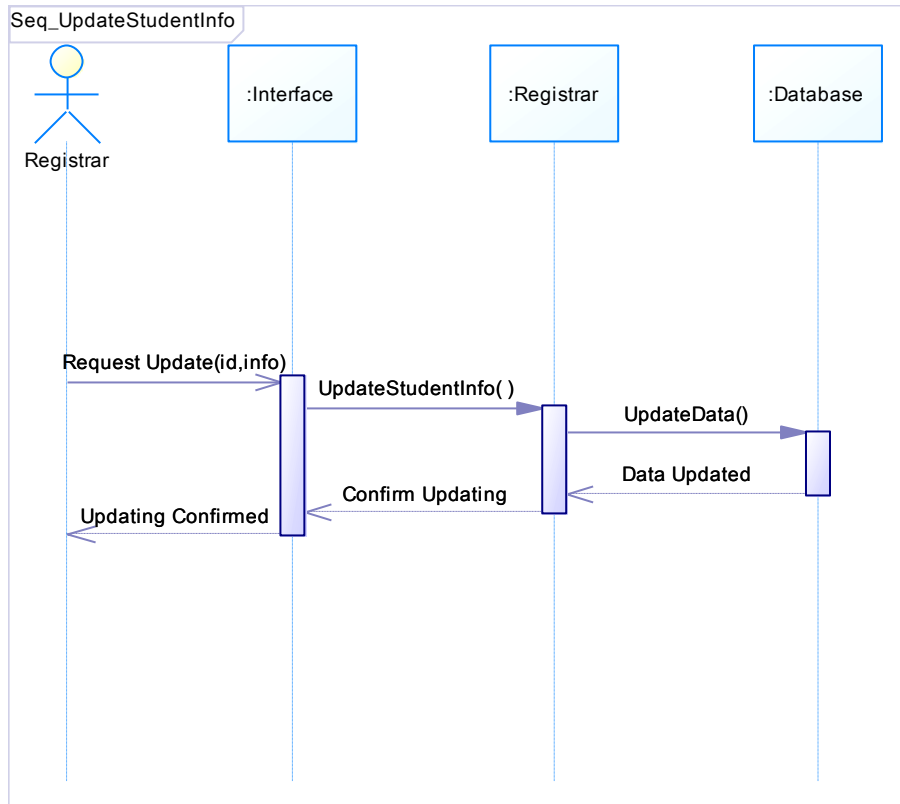


Figure 4: Sequence, Update Student Info

3.1.4 View Messages

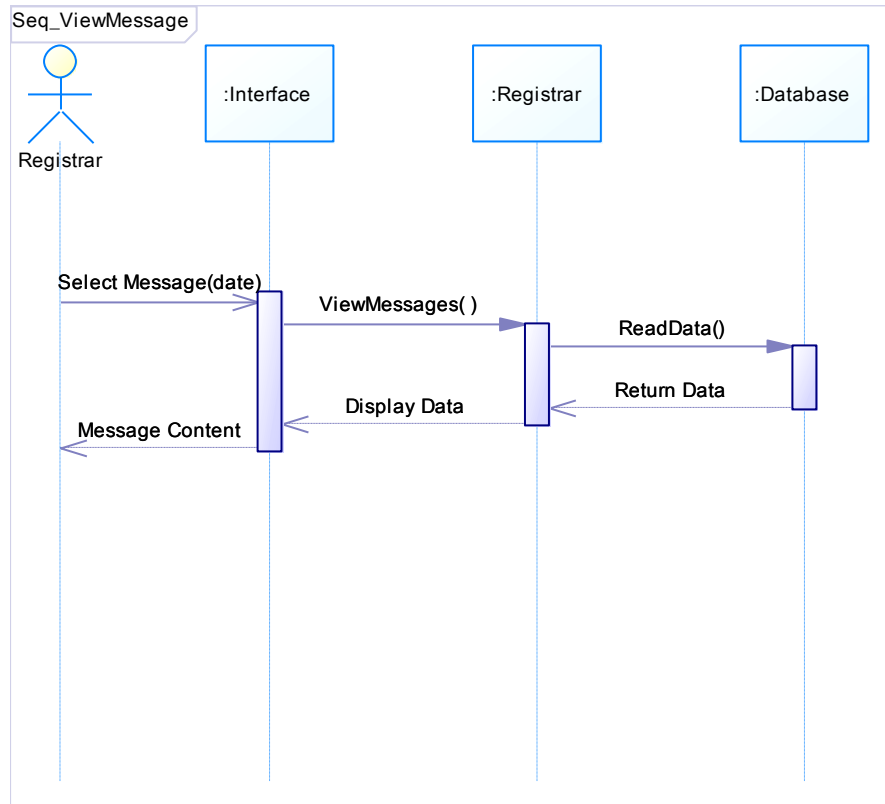


Figure 5: Sequence, View Messages

3.1.5 Add Course

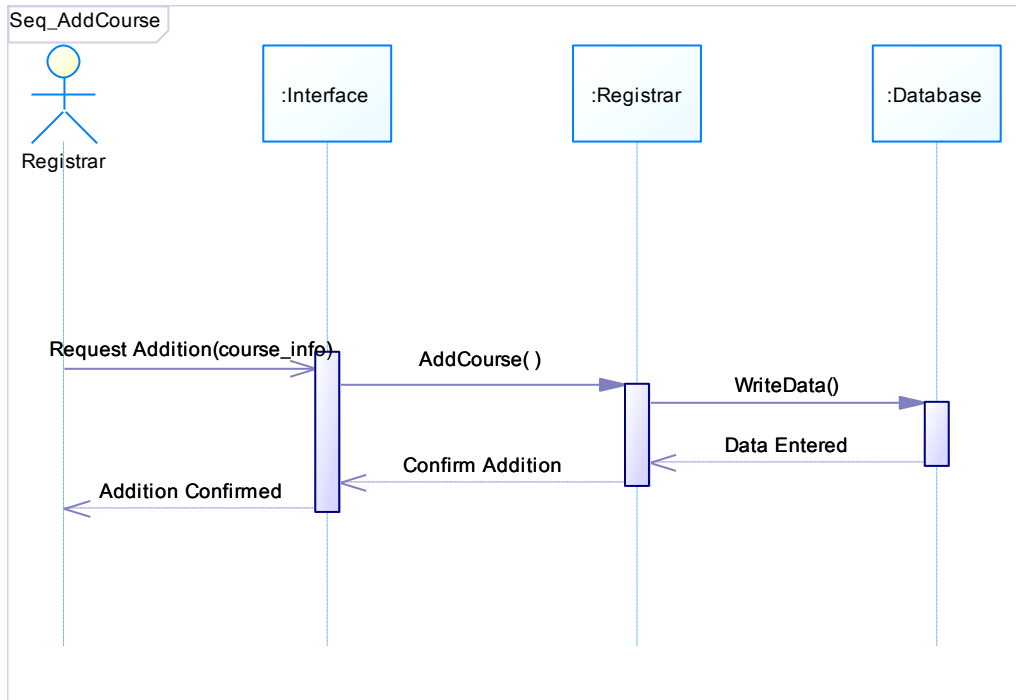


Figure 6: Sequence, Add Course

3.1.6 Edit Course

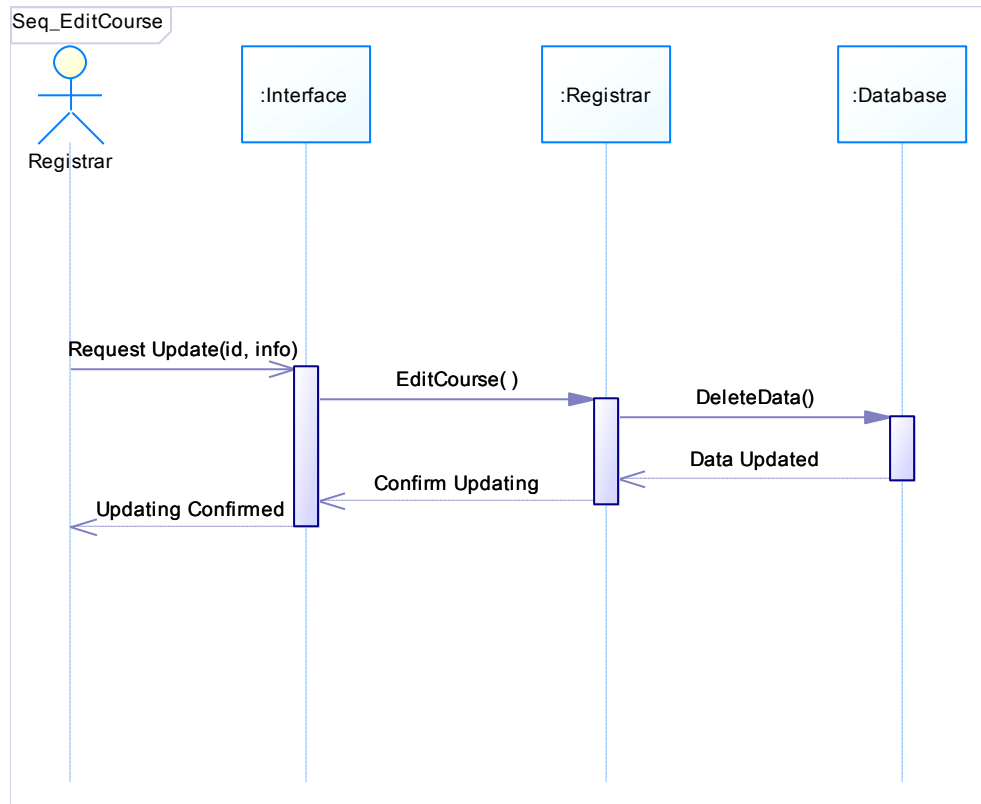


Figure 7: Sequence, Edit Course

3.1.7 Delete Course

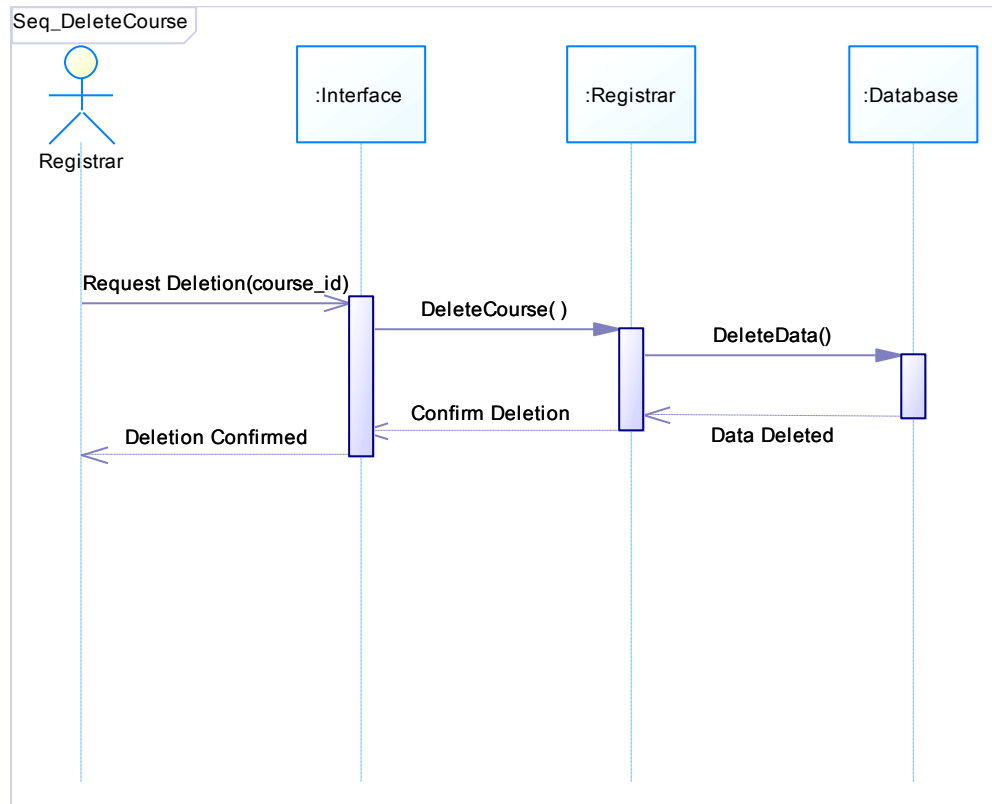


Figure 8: Sequence, Delete Course

3.1.8 Add Section

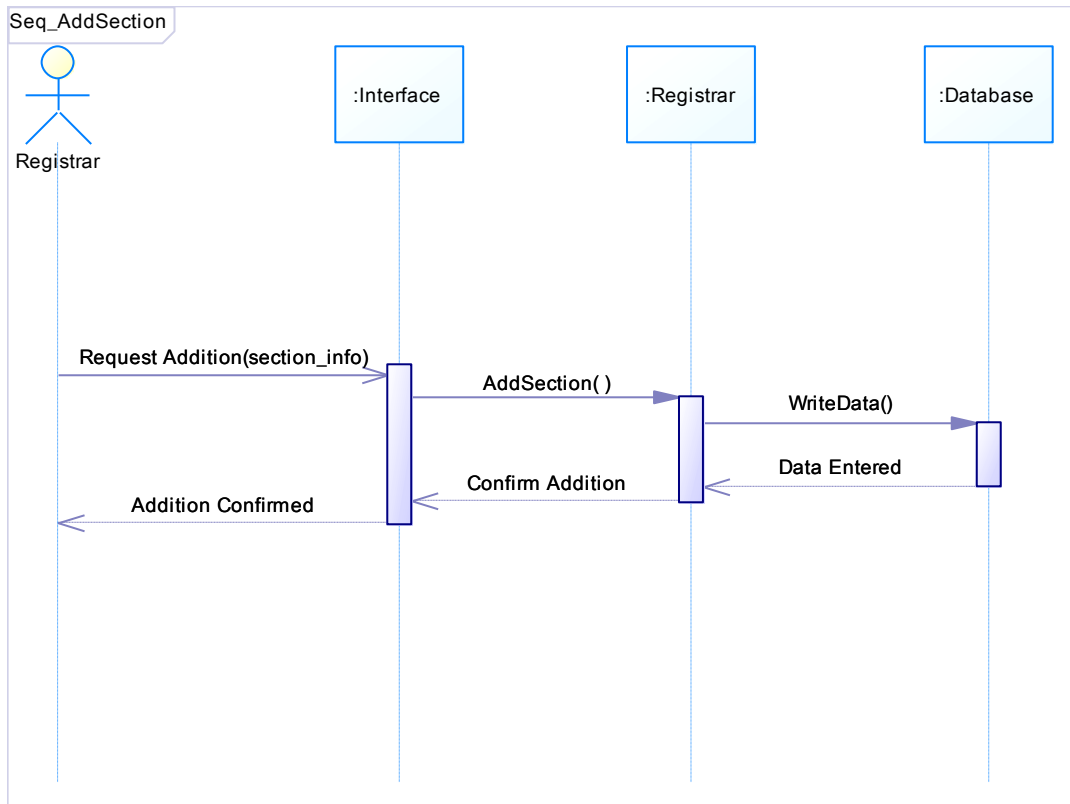


Figure 9: Sequence, Add Section

3.1.9 Edit Section

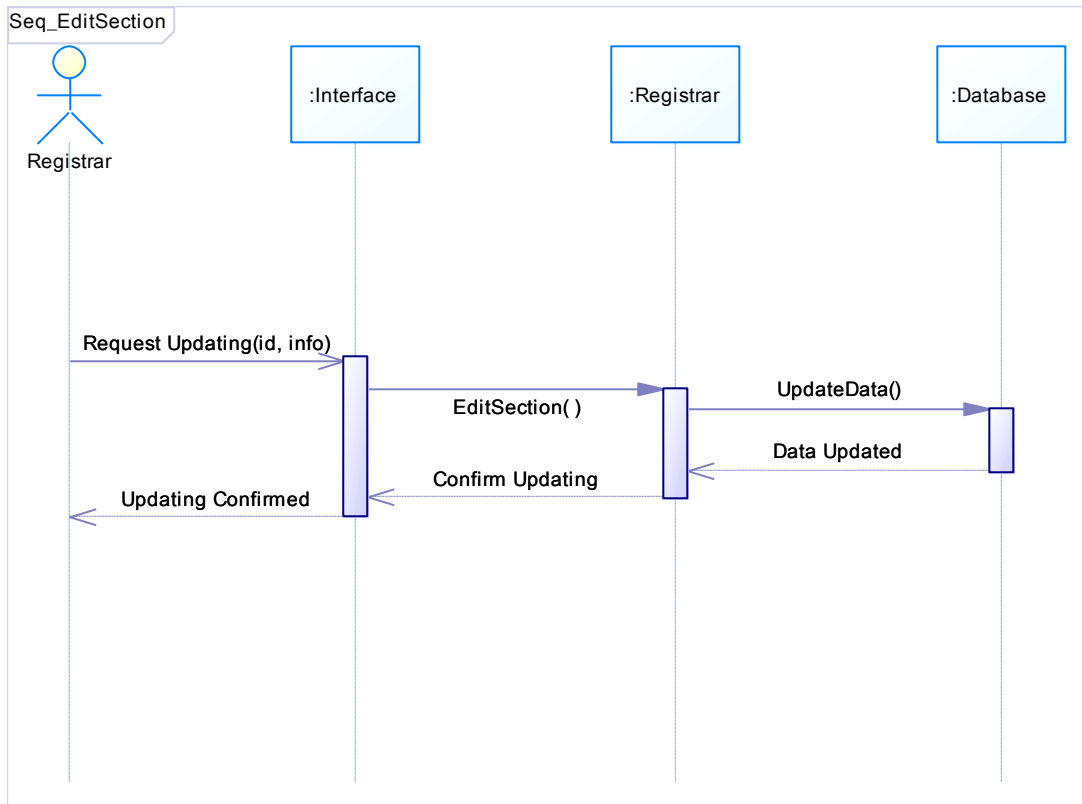


Figure 10: Sequence, Edit Section

3.1.10 Delete Section

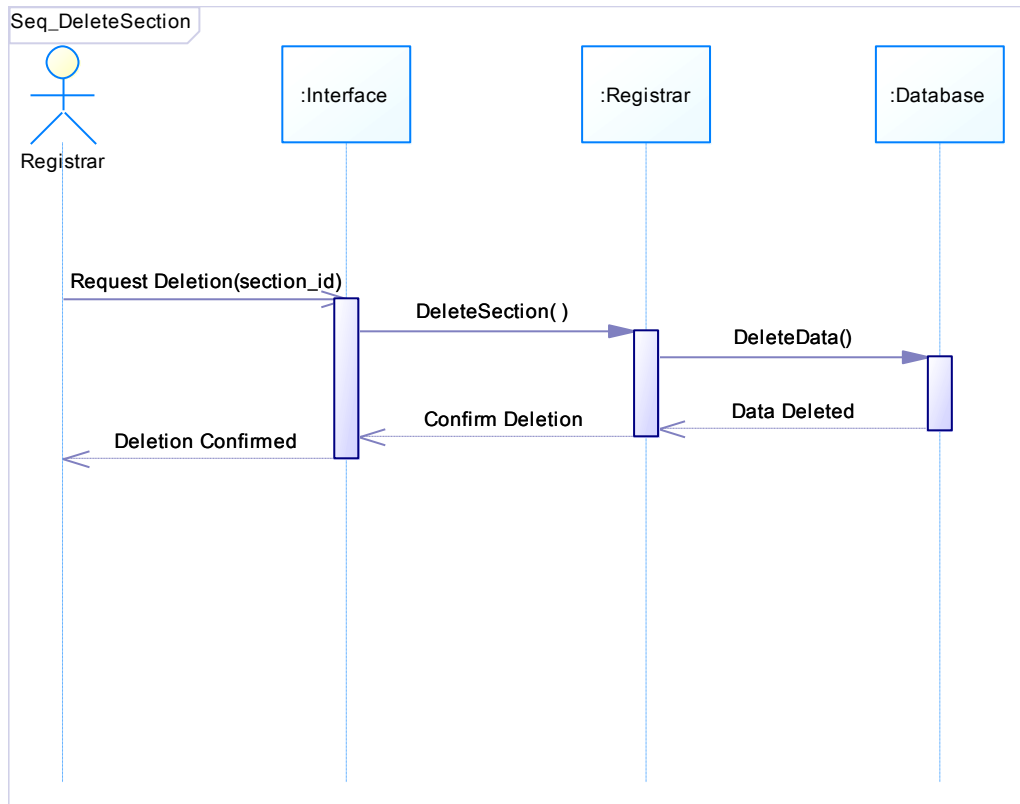


Figure 11: Sequence, Delete Section

3.1.11 Register Student in Course

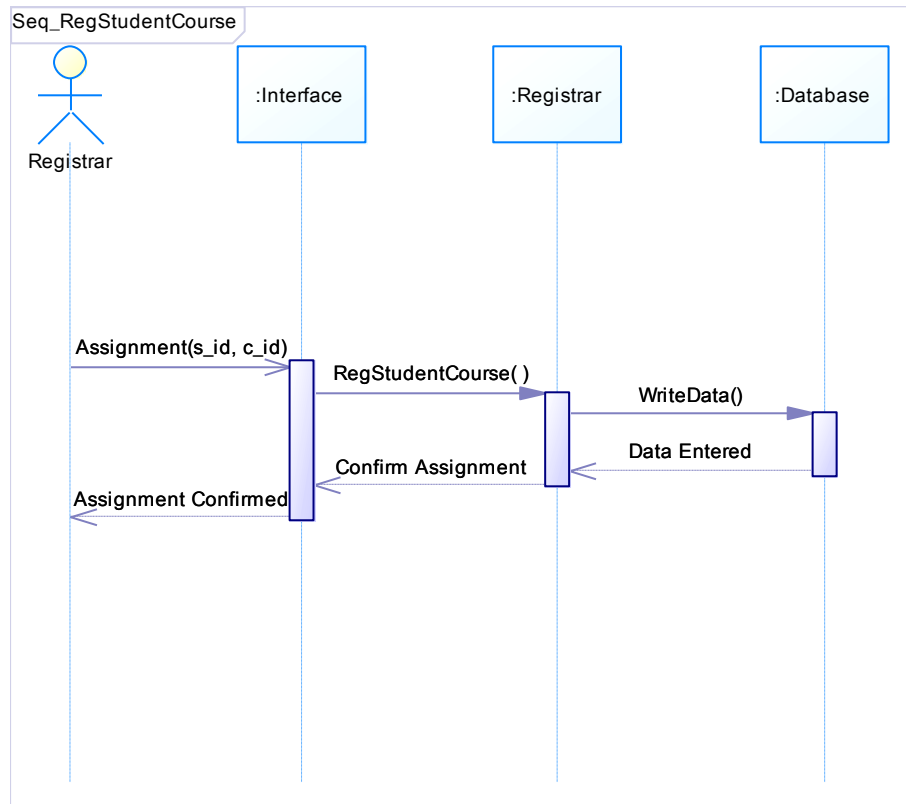


Figure 12: Sequence, Register Student in Course

3.1.12 Register Teacher in Course

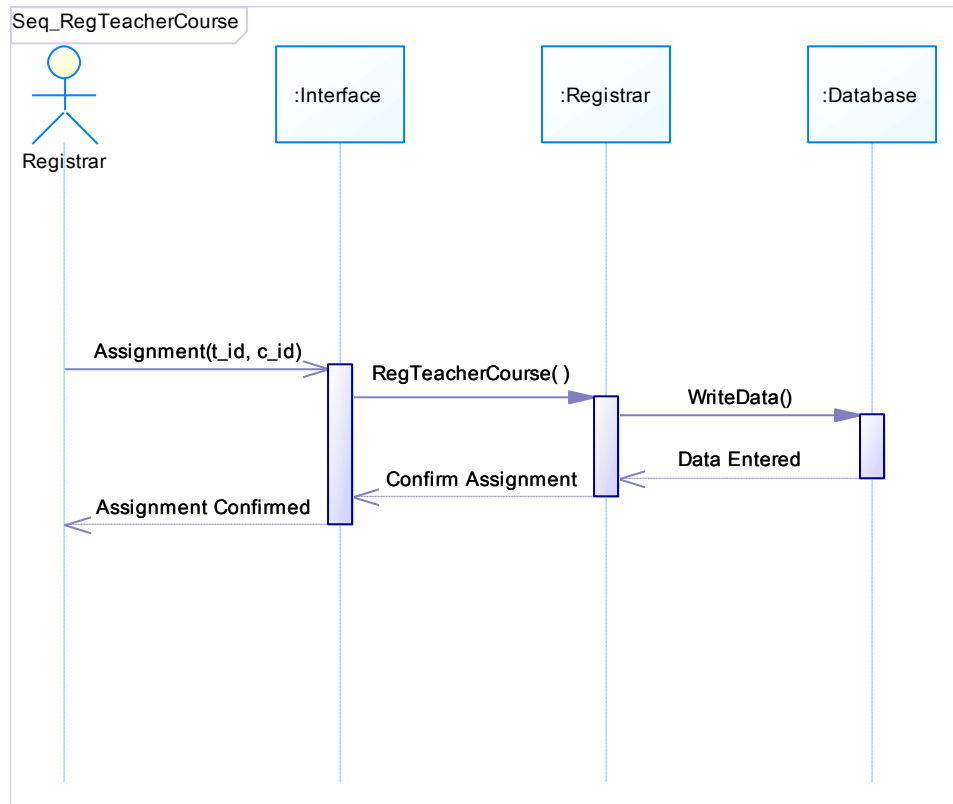


Figure 13: Sequence, Register Teacher in Course

3.1.13 Send Messages

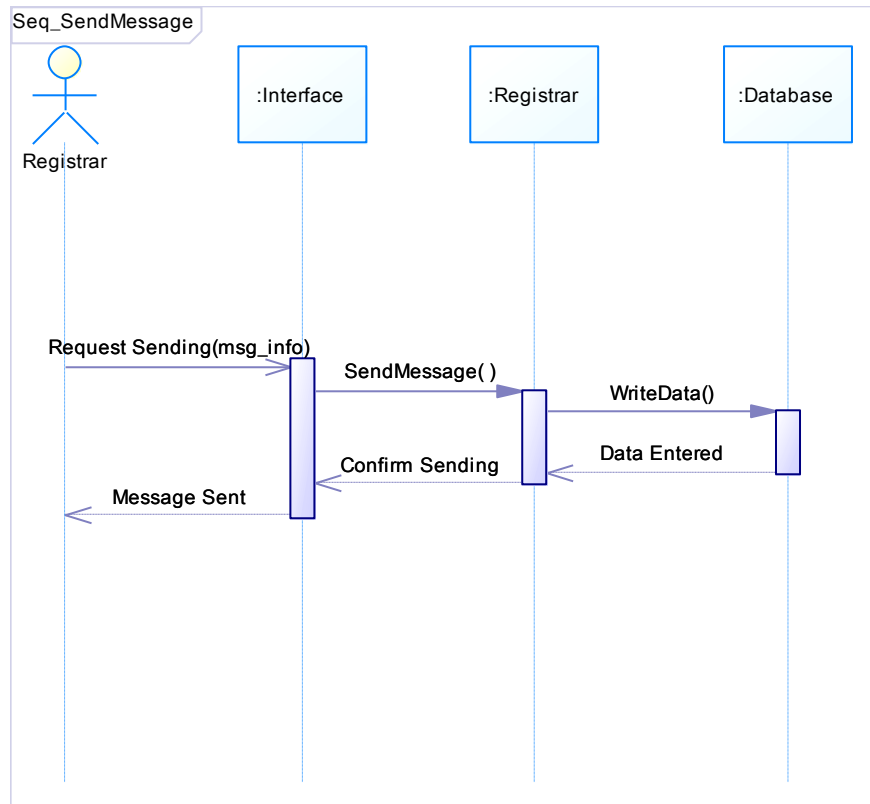


Figure 14: Sequence, Send Messages

3.2 ER Diagram

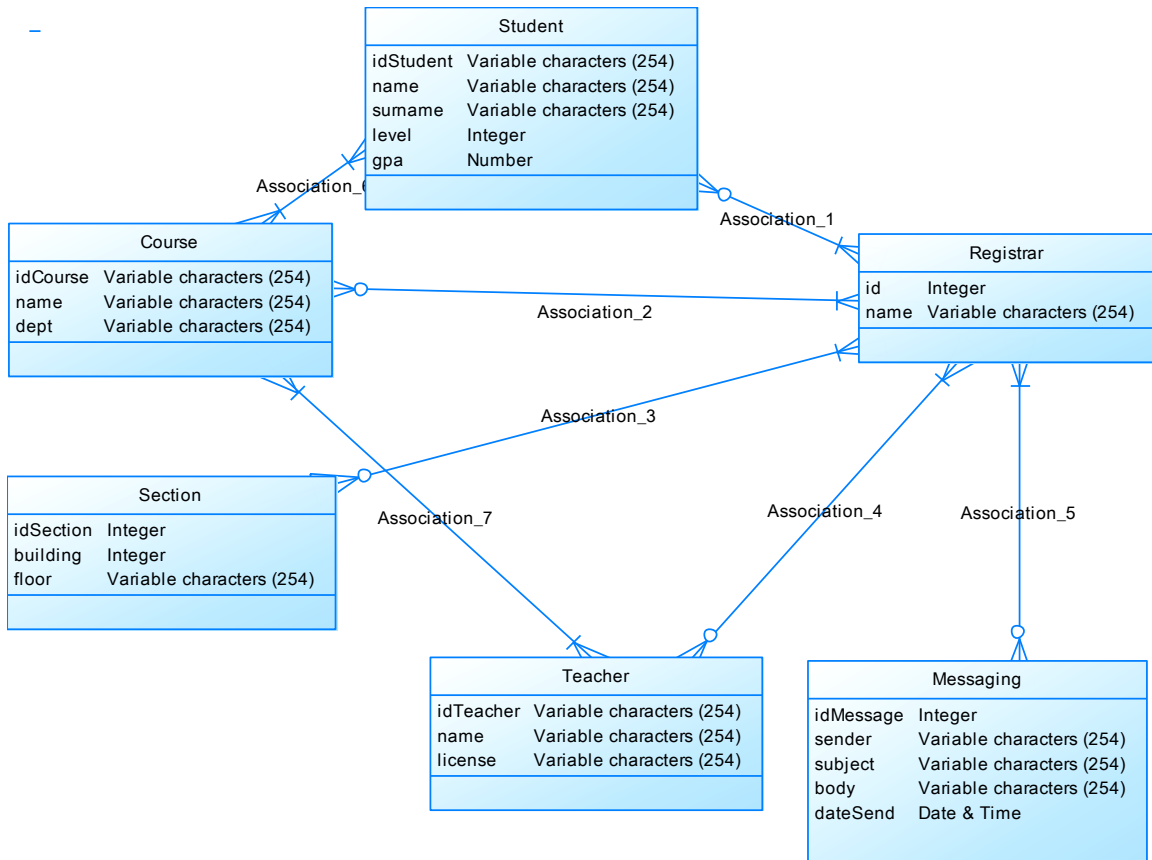


Figure 15: ER Diagram

3.3 Database Table Format

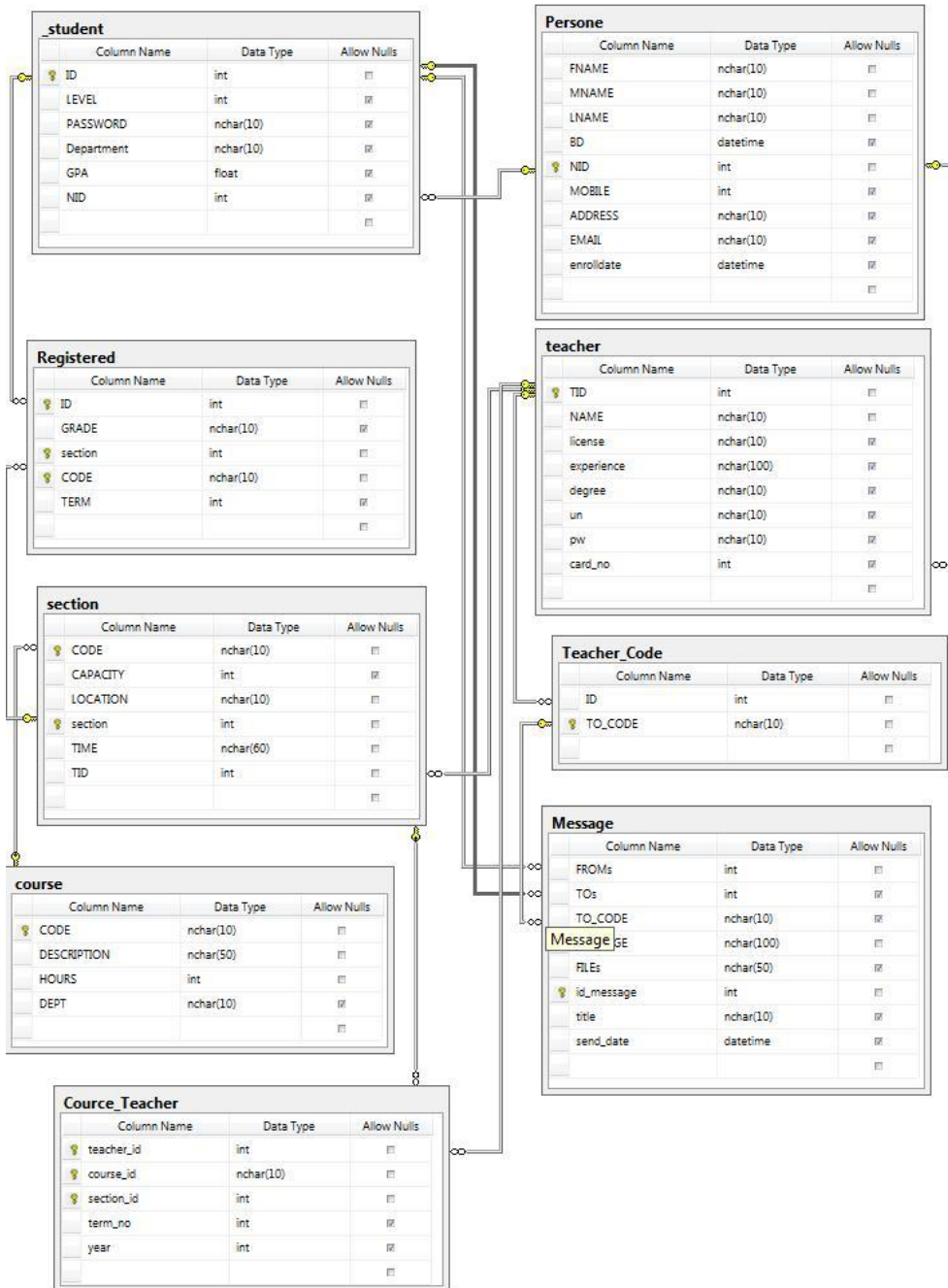


Figure 16: Database Table Format

3.4 Class Description

Class ID: 1	Class Name: Registrar
Inherits: None	
Attributes	Functions
id: int	RegisterStudent(newStudent):Boolean
name: string	DeleteStudent(idStudent): Boolean
	UpdateStudentInfo(idStudent):Boolean
	ViewMessages(idMessage):void
	AddCourse(newCourse):Boolean
	EditCourse(idCourse):Boolean
	DeleteCourse(idCourse):Boolean
	AddSection(newSection):Boolean
	EditSection(idSection):Boolean
	DeleteSection(idSection):Boolean
	RegisterStudentinCourse(idStudent, idCourse): Boolean
	RegisterTeacherinCourse(idTeacher, idCourse):Boolean
	SendMessage(newMessage):void

Class ID: 2	Class Name: Student
Inherits: None	
Attributes	Functions
idStudent: string	
name: string	
surName: string	
level: string	
gpa: double	

Class ID: 3	Class Name: Teacher
Inherits: None	
Attributes	Functions
Id: string	
Name: string	
License: string	

Class ID: 4	Class Name: Course
Inherits: None	
Attributes	Functions
idCourse: string	
Name: string	
Dept: string	

Class ID: 5	Class Name: Section
Inherits: None	
Attributes	Functions
idSection: int	
Building: string	
Floor: string	

Class ID: 6	Class Name: Messaging
Inherits: None	
Attributes	Functions
idMessage: int	
Sender: string	
Receiver: string	
Subject: string	
Body: string	
dateSend: DateTime	

3.5 Class Diagram

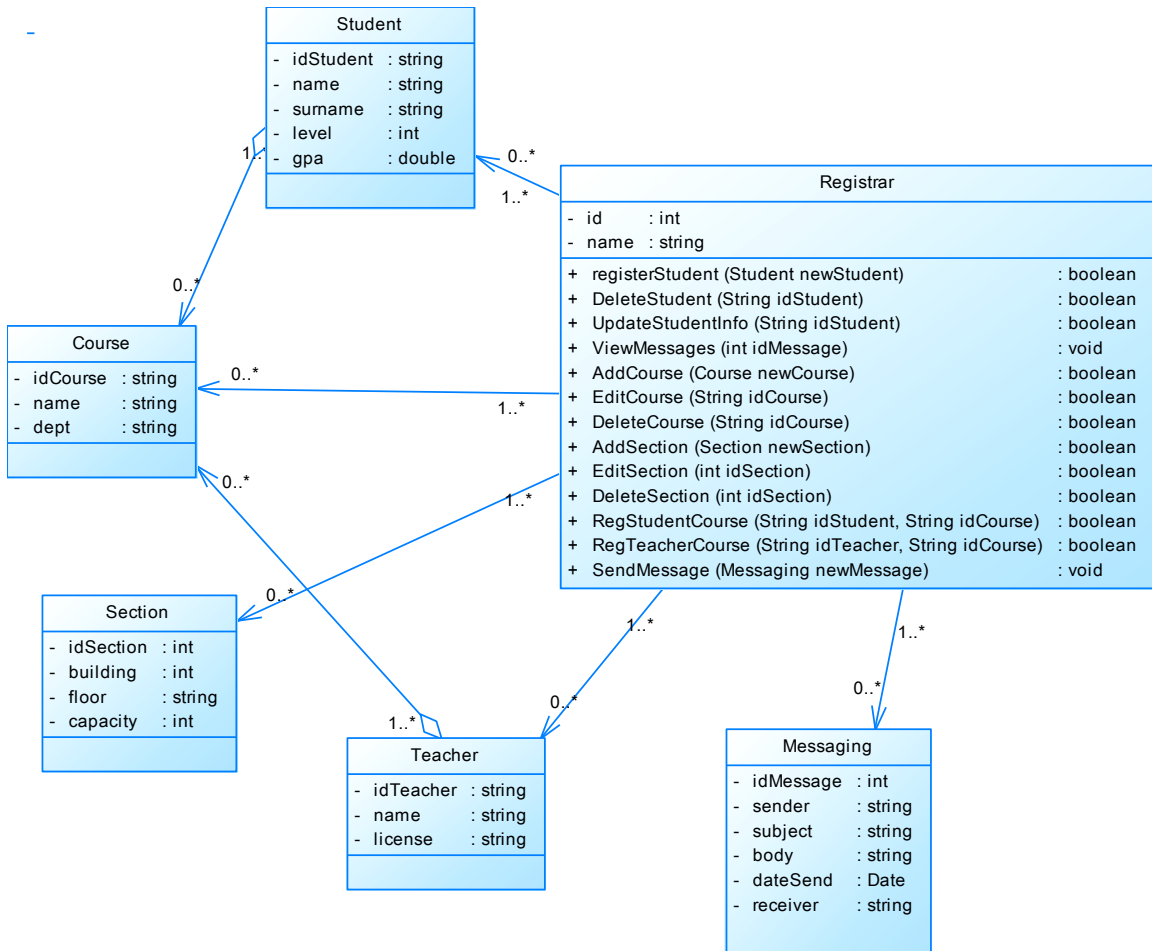


Figure 17: Class Diagram

4. *System Implementation*

Here we will list the tools we intend to use to build the software:

- Environment: Microsoft Visual Studio.NET 2008.
- Framework: .NET Framework 3.5.
- Programming Language: Visual Basic.NET 2008 or Visual C#.
- Database Engine: Microsoft SQL Server 2005 Express Edition.
- Application Type: Web-based application.

5. Test Cases

Test Case #: 1.		Test Case Name: Register Student.	
System: KSU Electronic System.		Subsystem: Registrars Subsystem.	
Designed by:		Design Date:	
Executed by:		Execution Date:	
Short Description: Registering new student in university.			
Per Conditions: None.			
Step	Action	Expected System Response	Pass/Fail
1	Registrar enters information of the new student	System validates the entries, enters new data, returns confirm msg.	Pass.
2	Registrar enters information of a new student but misses some info	System validates the entries and returns an error msg of the missing info.	Fail.
3	Registrar enters information of the new student.	System tries to save new data but returns msg of duplicated data.	Fail.
Post Conditions: <ol style="list-style-type: none"> 1- New student added to the system. 2- Registrar must complete the missing data. 3- Registrar revokes student addition. 			

Test Case #: 2.		Test Case Name: Delete Student.	
System: KSU Electronic System.		Subsystem: Registrars Subsystem.	
Designed by:		Design Date:	
Executed by:		Execution Date:	
Short Description: Deleting current student from the system.			
Per Conditions: Student must exist in the system.			
Step	Action	Expected System Response	Pass/Fail
1	Registrar selects a student and requests deleting	System deletes the student and return confirmation	Pass.
2	Registrar selects a student and requests deleting	System finds related objects to the selected student and refuses deletion	Fail.
3	Registrar requests deleting without selecting student	System returns error msg asking to select a student	Fail.
Post Conditions:			
<ol style="list-style-type: none"> 1- Student has been deleted from the system. 2- Student can't be deleted until related objects are released. 3- Registrar must select a student to complete the operation. 			

Test Case #: 3.		Test Case Name: Update Student Info.	
System: KSU Electronic System.		Subsystem: Registrars Subsystem.	
Designed by:		Design Date:	
Executed by:		Execution Date:	
Short Description: Updating existing student information.			
Per Conditions: Student must exist in the system.			
Step	Action	Expected System Response	Pass/Fail
1	Registrar selects a student and requests update	System puts the student info in update mode	Pass.
2	Registrar enters the updated info	System confirms entries and enters new data.	Pass.
3	Registrar enters some data in wrong format.	System returns error msg showing wrong data place.	Fail.
4	Registrar corrects errors and requests update	System validates data and completes update operation	Pass.
Post Conditions:			
<ol style="list-style-type: none"> 1- Student info has been updated. 2- Student info can't be updated because of wrong data format. 3- Updating confirmed after data format correction. 			

Test Case #: 4.		Test Case Name: View Messages.	
System: KSU Electronic System.		Subsystem: Registrars Subsystem.	
Designed by:		Design Date:	
Executed by:		Execution Date:	
Short Description: Reading received messages.			
Per Conditions:			
Msg box must contain new/old messages.			
Step	Action	Expected System Response	Pass/Fail
1	Registrar selects a msg to read	System displays the content of selected msg	Pass.
2	Registrar selects a msg to read.	System returns error msg: "can't find requested msg, maybe deleted"	Fail.
3	Registrar selects a msg to read.	System returns error due to poor connection with server.	Fail.
Post Conditions:			
<ol style="list-style-type: none"> 1- Registrar can read wanted msg. 2- Registrar must connect system admin to check on wanted msg. 3- Registrar must try again later. 			

Test Case #: 5.		Test Case Name: Add Course.	
System: KSU Electronic System.		Subsystem: Registrars Subsystem.	
Designed by:		Design Date:	
Executed by:		Execution Date:	
Short Description: Registering new course in university.			
Per Conditions: None.			
Step	Action	Expected System Response	Pass/Fail
1	Registrar enters information of the new course	System validates the entries, enters new data, returns confirm msg.	Pass.
2	Registrar enters information of a new course but misses some info	System validates the entries and returns an error msg of the missing info.	Fail.
3	Registrar enters information of the new course.	System tries to save new data but returns msg of duplicated data.	Fail.
Post Conditions:			
<ul style="list-style-type: none"> 1- New course added to the system. 2- Registrar must complete the missing data. 3- Registrar revokes course addition. 			

Test Case #: 6.		Test Case Name: Edit Course.	
System: KSU Electronic System.		Subsystem: Registrars Subsystem.	
Designed by:		Design Date:	
Executed by:		Execution Date:	
Short Description: Updating existing course information.			
Per Conditions: Course must exist in the system.			
Step	Action	Expected System Response	Pass/Fail
1	Registrar selects a course and requests update	System puts the course info in update mode	Pass.
2	Registrar enters the updated info	System confirms entries and enters new data.	Pass.
3	Registrar enters some data in wrong format.	System returns error msg showing wrong data place.	Fail.
4	Registrar corrects errors and requests update	System validates data and completes update operation	Pass.
Post Conditions:			
<ul style="list-style-type: none"> 1- Course info has been updated. 2- Course info can't be updated because of wrong data format. 3- Updating confirmed after data format correction. 			

Test Case #: 7.		Test Case Name: Delete Course.	
System: KSU Electronic System.		Subsystem: Registrars Subsystem.	
Designed by:		Design Date:	
Executed by:		Execution Date:	
Short Description: Deleting current course from the system.			
Per Conditions: Course must exist in the system.			
Step	Action	Expected System Response	Pass/Fail
1	Registrar selects a course and requests deleting	System deletes the course and return confirmation	Pass.
2	Registrar selects a course and requests deleting	System finds related objects to the selected course and refuses deletion	Fail.
3	Registrar requests deleting without selecting course	System returns error msg asking to select a course	Fail.
Post Conditions:			
<ul style="list-style-type: none"> 1- Course has been deleted from the system. 2- Course can't be deleted until related objects are released. 3- Registrar must select a course to complete the operation. 			

Test Case #: 8.		Test Case Name: Add Section.	
System: KSU Electronic System.		Subsystem: Registrars Subsystem.	
Designed by:		Design Date:	
Executed by:		Execution Date:	
Short Description: Registering new section in university.			
Per Conditions: None.			
Step	Action	Expected System Response	Pass/Fail
1	Registrar enters information of the new section	System validates the entries, enters new data, returns confirm msg.	Pass.
2	Registrar enters information of a new section but misses some info	System validates the entries and returns an error msg of the missing info.	Fail.
3	Registrar enters information of the new section.	System tries to save new data but returns msg of duplicated data.	Fail.
Post Conditions:			
<ul style="list-style-type: none"> 1- New section added to the system. 2- Registrar must complete the missing data. 3- Registrar revokes section addition. 			

Test Case #: 9.		Test Case Name: Edit Section.	
System: KSU Electronic System.		Subsystem: Registrars Subsystem.	
Designed by:		Design Date:	
Executed by:		Execution Date:	
Short Description: Updating existing section information.			
Per Conditions: Section must exist in the system.			
Step	Action	Expected System Response	Pass/Fail
1	Registrar selects a section and requests update	System puts the section info in update mode	Pass.
2	Registrar enters the updated info	System confirms entries and enters new data.	Pass.
3	Registrar enters some data in wrong format.	System returns error msg showing wrong data place.	Fail.
4	Registrar corrects errors and requests update	System validates data and completes update operation	Pass.
Post Conditions:			
<ul style="list-style-type: none"> 1- Section info has been updated. 2- Section info can't be updated because of wrong data format. 3- Updating confirmed after data format correction. 			

Test Case #: 10.		Test Case Name: Delete Section.	
System: KSU Electronic System.		Subsystem: Registrars Subsystem.	
Designed by:		Design Date:	
Executed by:		Execution Date:	
Short Description: Deleting current Section from the system.			
Per Conditions: Section must exist in the system.			
Step	Action	Expected System Response	Pass/Fail
1	Registrar selects a section and requests deleting	System deletes the section and return confirmation	Pass.
2	Registrar selects a section and requests deleting	System finds related objects to the selected section and refuses deletion	Fail.
3	Registrar requests deleting without selecting section	System returns error msg asking to select a section	Fail.
Post Conditions:			
<ul style="list-style-type: none"> 1- Section has been deleted from the system. 2- Section can't be deleted until related objects are released. 3- Registrar must select a section to complete the operation. 			

Test Case #: 11.		Test Case Name: Register Student in Course.	
System: KSU Electronic System.		Subsystem: Registrars Subsystem.	
Designed by:		Design Date:	
Executed by:		Execution Date:	
Short Description: Registering a student in a course.			
Per Conditions: Student and course must exist in the system.			
Step	Action	Expected System Response	Pass/Fail
1	Registrar selects a student and a course and requests assignment	System performs the assignment and returns a confirmation.	Pass.
2	Registrar requests assignment without selecting student/course	System returns error msg requesting the selection of student/course.	Fail.
3	Registrar selects a student and a course and requests assignment	System returns error msg showing that the selected student already taking the selected course.	Fail.
Post Conditions:			
<ol style="list-style-type: none"> 1- The student should attend to the new course. 2- Registrar should complete the required information before assignment completion. 3- Registrar revokes the assignment operation. 			

Test Case #: 12.		Test Case Name: Register Teacher in Course.	
System: KSU Electronic System.		Subsystem: Registrars Subsystem.	
Designed by:		Design Date:	
Executed by:		Execution Date:	
Short Description: Registering a teacher in a course.			
Per Conditions: Teacher and course must exist in the system.			
Step	Action	Expected System Response	Pass/Fail
1	Registrar selects a teacher and a course and requests assignment	System performs the assignment and returns a confirmation.	Pass.
2	Registrar requests assignment without selecting teacher/course	System returns error msg requesting the selection of teacher/course.	Fail.
3	Registrar selects a teacher and a course and requests assignment	System returns error msg showing that the selected teacher already teaches the selected course.	Fail.
Post Conditions:			
<ul style="list-style-type: none"> 1- The student should attend to teach the new course. 2- Registrar should complete the required information before assignment completion. 3- Registrar revokes the assignment operation. 			

Test Case #: 13.		Test Case Name: Register Teacher in Course.	
System: KSU Electronic System.		Subsystem: Registrars Subsystem.	
Designed by:		Design Date:	
Executed by:		Execution Date:	
Short Description: Sending messages to students/teachers/others in the system.			
Per Conditions: Receiver must exist in the system.			
Step	Action	Expected System Response	Pass/Fail
1	Registrar enters the receiver, subject, msg body, and requests sending	System performs sending and returns confirmation	Pass.
2	Registrar misses some required information while sending a msg	System returns error msg requesting entering the missing information.	Fail.
Post Conditions:			
1- Msg sent to the wanted receiver. 2- Registrar should complete the required information before sending completion.			

6. Conclusion and Future Work

At the end of this document, we found that a lot of work is yet to be done, and the most important work to do is the integration with other subsystems. We hope that our humble work fulfills the requirements of devoted registrars and that our system will help them to accomplish their work perfectly.