CoursesManagementApp

Sprint Report

Σπυρίδων Σιλίρας, 3330

VERSIONS HISTORY

Date	Version	Description	Author
12/5/2022	1.0.2	Final Update	Σπυρίδων Σιλίρας

(**Σημείωση:** Είχα δηλώσει ομάδα με έναν συμφοιτητή μου με ΑΜ 3367, αλλά δεν ασχολήθηκε με την εργασία με αποτέλεσμα να μην τον συμπεριλάβω στο report.)

1 Introduction

This document provides information concerning the **second** sprint of the project.

1.1 Purpose

The objective of this project is to develop a Web application that allows an instructor to manage the grading of the courses that he teaches.

1.2 Document Structure

The rest of this document is structured as follows. Section 2 describes out Scrum team and specifies the this Sprint's backlog. Section 3 specifies the main design concepts for this release of the project.

2 Scrum team and Sprint Backlog

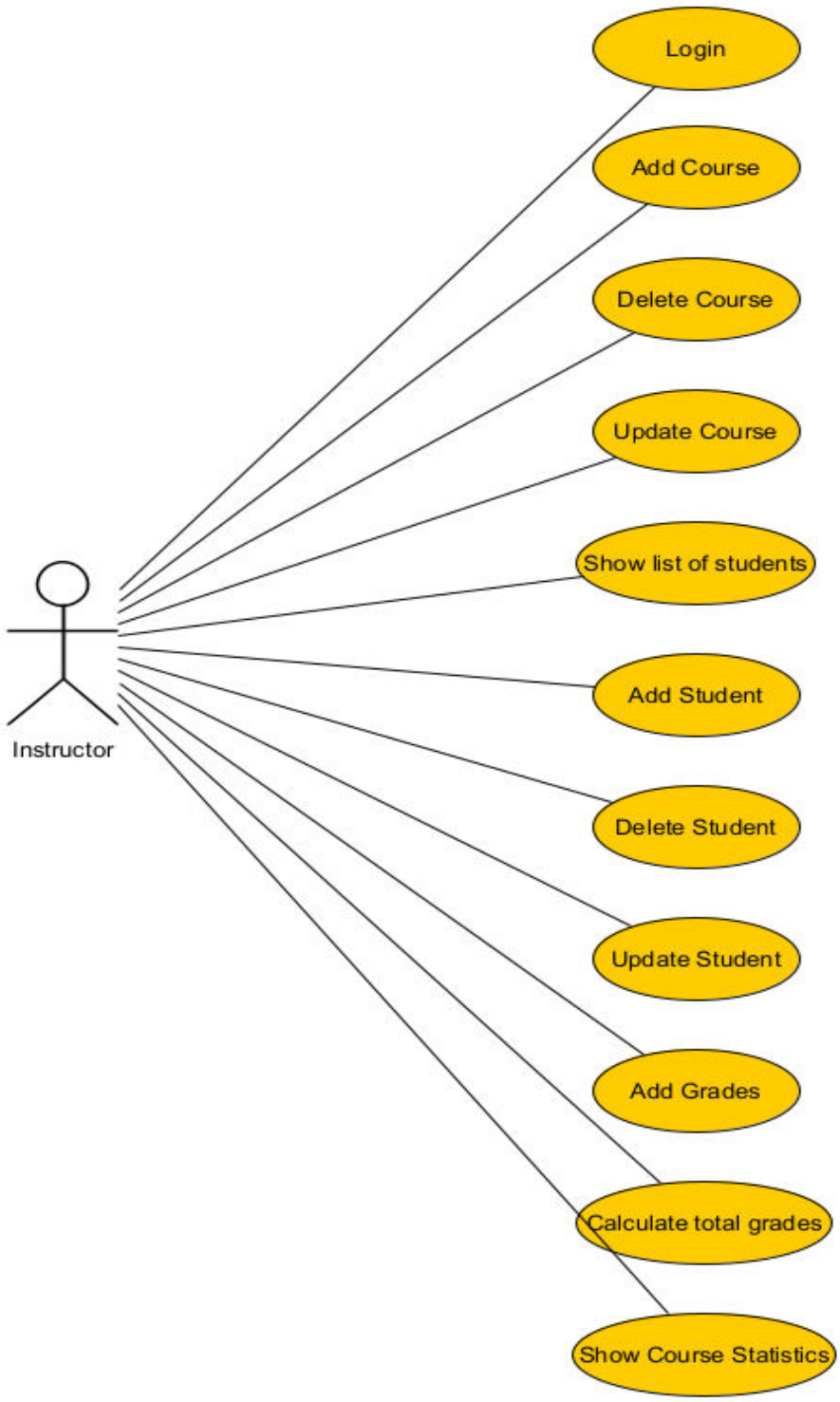
2.1 Scrum team

Product Owner	Απόστολος Ζάρας
Scrum Master	Σπυρίδων Σιλίρας
Development Team	Σπυρίδων Σιλίρας

2.2 Sprints

Sprint No	Begin Date	End Date	Number of weeks	User stories
1	**9/3/2022**	**19/3/2022**	**2**	**5**
2	**27/4/2022**	**12/5/2022**	**2**	**7**

3 Use Cases


Login
Add Course
Delete Course
Update Course
Show list of students
Add Student
Delete Student
Update Student
Add Grades
Calculate total grades
Show Course Statistics
Instructor

3.1 InstructorLogin

(Να σημειωθεί ότι δεν κατάφερα να φτιάξω το login όπως μας ζητείται, απλά έχω ένα πεδίο username σαν αρχική σελίδα που χρησιμοποιείται και ως instructor στις υπόλοιπες ενέργειες που θα ακολουθήσουν.)

Use case ID	UC1
Actors	Instructor
Pre conditions	1.The system has successfully start.
Main flow of events	1. The use case starts when the user starts the application. 2. The system asks for the username of the user. 3. The system shows the list of courses for this user.
Post conditions	1.The course's list appears.

3.2 AddCourse

Use case ID	UC2
Actors	Instructor
Pre conditions	1.The list of courses is visible.
Main flow of events	1.The use case starts when the user selects the 'Add Course' button. 2.The system shows a course form. 3.The user fills the form with specific new course details. 4.The system updates the list of courses.
Post conditions	1.The course's list have been updated.

3.3 DeleteCourse

Use case ID	UC3
Actors	Instructor
Pre conditions	1.The list of courses is visible.
Main flow of events	1.The use case starts when the user selects the 'Delete' button. 2.The system informs user that a specific course will be deleted. 3.The system updates the list of courses.
Post conditions	1.The course's list have been updated.

3.4 UpdateCourse

Use case ID	UC4
Actors	Instructor
Pre conditions	1.The list of courses is visible.
Main flow of events	1.The use case starts when the user selects the 'Update' button. 2.The system shows a course form. 3.The user fills the updated details for the course. 4. The system updates the list of courses with the new information for the course.
Post conditions	1.The course's list have been updated.

3.5 ShowListOfStudents

Use case ID	UC5
Actors	Instructor
Pre conditions	1.The list of courses is visible.
Main flow of events	1.The use case starts when the user selects the 'Students' button in a particular course. 2.The system shows the list of students for this course.
Post conditions	1.The students list for this course have been appeared.

3.6 AddStudent

Use case ID	UC6
Actors	Instructor
Pre conditions	1.The list of students for a specific course is visible.
Main flow of events	1.The use case starts when the user selects the 'Add Student' button. 2.The system shows a student form. 3.The user fill the form with student information. 4.The system updates the list of students.
Post conditions	1.The students list have been updated.

3.7 DeleteStudent

Use case ID	UC7
Actors	Instructor
Pre conditions	1.The list of students for a specific course is visible.
Main flow of events	1.The use case starts when the user selects the 'Delete' button. 2.The system informs user that the student of this course will be deleted. 3.The system updates the list of courses.
Post conditions	1.The students list have been updated.

3.8 UpdateStudent

Use case ID	UC8
Actors	Instructor
Pre conditions	1.The list of students for a specific course is visible.
Main flow of events	1.The use case starts when the user selects the 'Update / Add grades' button. 2.The system shows a course form. 3.The user fills the updated details for the student. 4. The system updates the list of students with the new information for the student.
Post conditions	1.The students list have been updated.

3.9 AddGradesAndCalculateTotalGrade

Use case ID	UC9
Actors	Instructor
Pre conditions	1.The list of students for a specific course is visible.
Main flow of events	1.The use case starts when the user selects the 'Update / Add grades' button. 2.The system shows a course form. 3.The user fills the project and exam grades for the student. 4. The system calculates the total grade with respected to a weighted average of exam and project grade. 5.The system updates the list of students.
Post Conditions	1.The students list have been updated.

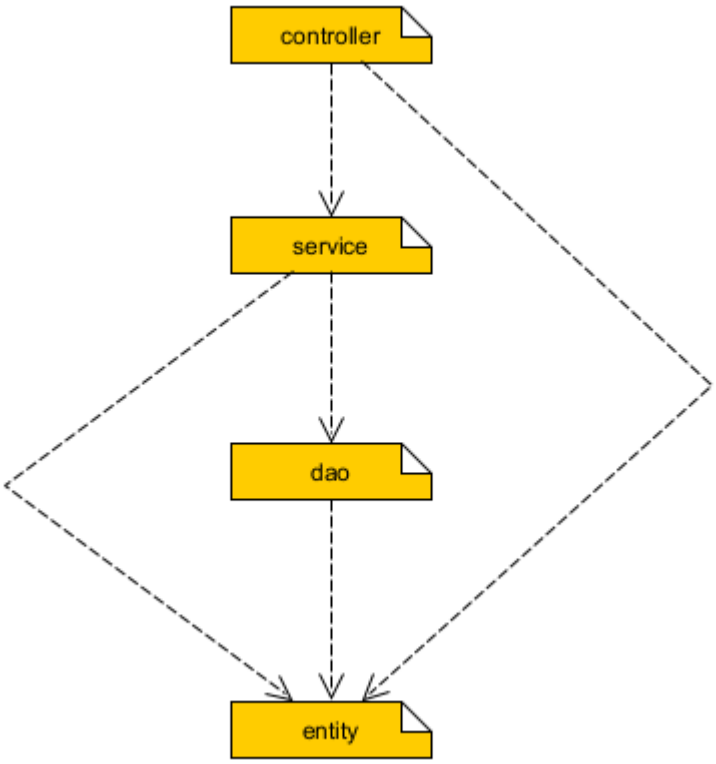
3.10 ShowStatistics

Use case ID	UC10
Actors	Instructor
Pre conditions	1.The list of students for a specific course is visible.
Main flow of events	1.The use case starts when the user selects the 'Show Statistics' button. 2.The system calculates descriptive statistics for student's grades of a particular course. 3.The system shows the list with the statistics for the students of the course.
Post conditions	1.The statistics have been appeared.

4 Design

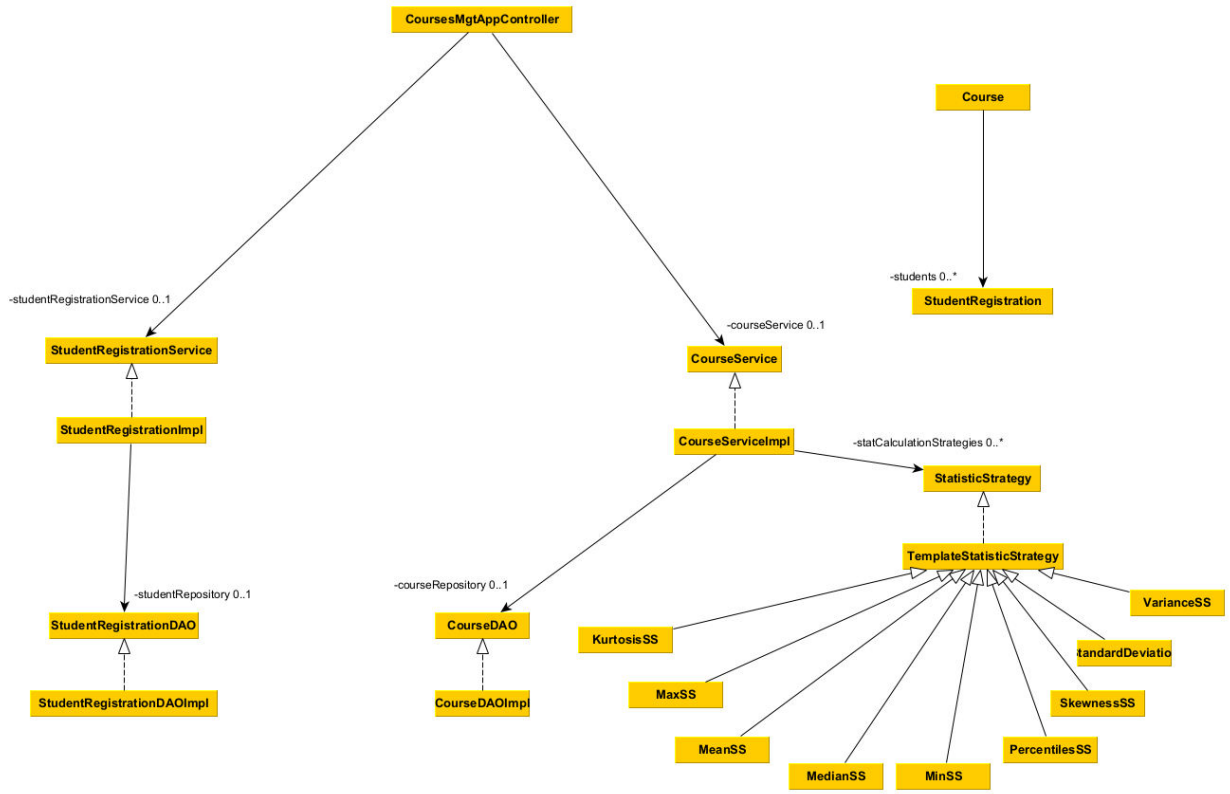
4.1 Architecture

Packages Diagram



4.2 Design

UML Diagram



CRC Cards

Class Name: Course	
Responsibilities: ▪ Model class ▪ Has a list of StudentRegistration objects ▪ Save and retrieve data from db	**Collaborations:** ▪ StudentRegistration

Class Name: StudentRegistration	
Responsibilities: ▪ Model class ▪ Save and retrieve data from db	**Collaborations:**

Class Name: CourseDAO	
Responsibilities: ▪ Interface with data access methods for Course ▪ Keeps the Service layer independent from the specific implementation	**Collaborations:** ▪ Course

Class Name: CourseDAOImpl	
Responsibilities: ▪ Implements the CourseDAO interface ▪ Implements db methods using the EntityManager interface ▪ Save and retrieve data from db	**Collaborations:** ▪ Course

Class Name: StudentRegistrationDAO	
Responsibilities: ▪ Interface with data access methods for StudentRegistration ▪ Keeps the Service layer independent from the specific implementation	**Collaborations:** ▪ StudentRegistration

Class Name: StudentRegistrationDAOImpl	
Responsibilities: ▪ Implements StudentRegistrationDAO interface ▪ Implements db methods using the EntityManager interface ▪ Save and retrieve data from db	**Collaborations:** ▪ StudentRegistration ▪ Course

Class Name: CourseService	
Responsibilities: ▪ Interface with the definition of operations of the service provided to the users ▪ Separates controller from DAO layer ▪ Maintainability and easy extension of the application	**Collaborations:** ▪ Course

Class Name: CourseServiceImpl	
Responsibilities: ▪ Implements methods from CourseService interface ▪ Uses CourseDAO to save and retrieve data ▪ Calculates and provides the statistics of a course to the users.	**Collaborations:** ▪ Course ▪ CourseDAO

Class Name: CourseServiceImpl	
Responsibilities: ▪ Implements methods from CourseService interface ▪ Uses CourseDAO to save and retrieve data ▪ Calculates and provides the statistics of a course to the users.	**Collaborations:** ▪ Course ▪ CourseDAO

Class Name: StatisticStrategy	
Responsibilities: ▪ Satisfies the maintainability and easy extension of the application ▪ Employs the Strategy pattern	**Collaborations:** ▪ Course

Class Name: TemplateStatisticStrategy	
Responsibilities: ▪ Used to avoid duplicate code in the classes that implement the StatisticStrategy interface ▪ Prepares the dataset for the calculation of the descriptive statistics	**Collaborations:** ▪ Course ▪ StatisticStrategy

Class Name: MaxStatisticStrategy, MinStatisticStrategy, …	
Responsibilities: ▪ Implements the StatisticStrategy interface ▪ Calculates the descriptive statistic given the dataset and using tha Apache math library	**Collaborations:** ▪ StatisticStrategy ▪ TemplateStatisticStrategy

Class Name: StudentRegistrationService	
Responsibilities: ▪ Interface with the definition of operations of the service provided to the users ▪ Separates controller from DAO layer ▪ Maintainability and easy extension of the application	**Collaborations:** ▪ StudentRegistration

Class Name: StudentRegistrationImpl	
Responsibilities: ▪ Implements methods from StudentRegistrationService interface ▪ Uses StudentRegistrationDAO to save and retrieve data	**Collaborations:** ▪ StudentRegistration ▪ StudentRegistrationDAO

Class Name: CoursesMgtAppController	
Responsibilities: ▪ Uses service layer objects to manipulate domain objects and update the view (gui)	**Collaborations:** ▪ Course ▪ StudentRegistration ▪ CourseService ▪ StudentRegistrationService