CoursesManagementApp

Sprint Report

VERSIONS HISTORY

Date	Version	Description	Author
18/5/2022	1.0	Final Report	

1 Introduction

This document provides information concerning all sprints of the project.

1.1 Purpose

The purpose of this project is to develop a Web application to help an instructor 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 this Sprint's backlog. Section 3 specifies the use cases that describe the interaction of the user with the applications. Section 4 specifies the main design concepts for this release of the project.

2 Scrum team and Sprint Backlog

2.1 Scrum team

Product Owner	
Scrum Master	
Development Team	

2.2 Sprints

Sprint No	Begin Date	End Date	Number of weeks	User stories
1	**25/3/2022**	**27/3/2022**	**1**	**US1**
2	**28/3/2022**	**13/4/2022**	**3**	**US2, US3, US4, US5**
3	**14/4/2022**	**1/5/2022**	**3**	**US6, US7, US8, US9**
4	**3/5/2022**	**8/5/2022**	**1**	**US10, US11**
5	**9/5/2022**	**15/5/2022**	**1**	**US12**

3 Use Cases

3.1 UC1: LoginUser

Use case ID	UC1
Actors	Instructor
Preconditions	None
Main flow of events	1. The use case starts when the instructor opens the application at the browser. 2. The system asks from the user for username and password to login. 3. The system login the instructor.
Alternative flow 1	If the username or/and password is wrong, then the system prints an error and asks again from the instructor for the correct credentials.
Post conditions	Instructor can see and edit the courses that he teaches.

3.2 UC2: BrowseCourses

Use case ID	UC2
Actors	Instructor
Preconditions	Instructor successfully logged in.
Main flow of events	1. The use case starts when the instructor opens the page /courses in browser. 2. He can see the courses that he teaches, edit these courses, delete them, and add a new one.
Alternative flow 1	If the instructor hasn't registered any course yet, then he sees an empty form.
Post conditions	None

3.3 UC3: AddNewCourse

Use case ID	UC3
Actors	Instructor
Preconditions	Instructor successfully logged in.
Main flow of events	1. The use case starts when the instructor presses the Add Course button in the courses page. 2. The system shows the form for course creation. 3. The instructor adds the proper information for the course and press the Submit button. 4. The system creates the course and redirects the instructor to the courses page.
Alternative flow 1	If the instructor forgets to add any information the system informs him to add the missing info.
Alternative flow 2	If the course id, that the instructor adds, exists already, then the system returns an error page and the system doesn't save the course.
Alternative flow 3	If instructor presses the Cancel button in the form, then the system redirects the instructor to courses page without create a new course.
Post conditions	The courses list has a new entry.

3.4 UC4: DeleteCourse

Use case ID	UC4
Actors	Instructor
Preconditions	Instructor successfully logged in.
Main flow of events	1. The use case starts when the instructor presses the Delete button for the course that he wants to delete. 2. The system shows a confirm dialog to delete the course. 2.1. If the instructor presses OK, then the system deletes the course. 2.2. If the instructor presses Cancel, then nothing happened.
Post conditions	The system removes a course from the courses list.

3.5 UC5: UpdateCourse

Use case ID	UC5
Actors	Instructor
Preconditions	Instructor successfully logged in.
Main flow of events	1. The use case starts when the instructor presses the Update button for the course that he wants to update. 2. The system shows the form for course update. 3. The instructor makes the changes that he wants and presses the Submit button. 4. The system updates the description of the course and redirects the instructor to courses page.
Alternative flow 1	If the instructor updates the course id into an existing id, then the system returns an error page and the system doesn't update the course.
Alternative flow 2	If the instructor makes empty one of the form's fields, then the system informs him that he must fulfill it in order to update the course.
Alternative flow 3	If the instructor presses the Cancel button, then the system does nothing and redirects the instructor to courses page.
Post conditions	The system updates the course.

3.6 UC6: BrowseStudents

Use case ID	UC6
Actors	Instructor
Preconditions	Instructor successfully logged in.
Main flow of events	1. The use case starts when the instructor presses the course id of a specific course. 2. The system redirects the instructor to students' page of the chosen course. 3. The instructor can see, update, delete the student registrations and create a new one.
Alternative flow 1	If the student registrations list is empty, then the instructor sees an empty table.
Post conditions	None

3.7 UC7: AddNewStudent

Use case ID	UC7
Actors	Instructor
Preconditions	Instructor successfully logged in.
Main flow of events	1. The use case starts when the instructor presses the Add Student button in the students' page. 2. The system shows the form for student registration. 3. The instructor adds the proper information (including the grades) for the student and press the Submit button. 4. The system registers the student and redirects the instructor to the students' page.
Alternative flow 1	If the instructor forgets to add any information the system informs him to add the missing info.
Alternative flow 2	If the A.M. , that the instructor adds, exists already, then the system returns an error page and the system doesn't register the student.
Alternative flow 3	If instructor presses the Cancel button in the form, then the system redirects the instructor to students' page without create a new student.
Post conditions	The students' list has a new entry.

3.8 UC8: DeleteStudent

Use case ID	UC8
Actors	Instructor
Preconditions	Instructor successfully logged in.
Main flow of events	1. The use case starts when the instructor presses the Delete button for the student that he wants to delete. 2. The system shows a confirm dialog to delete the student. 2.1. If the instructor presses OK, then the system deletes the student. 2.2. If the instructor presses Cancel, then nothing happened.
Post conditions	The system removes a student from the students list.

3.9 UC9: UpdateStudent

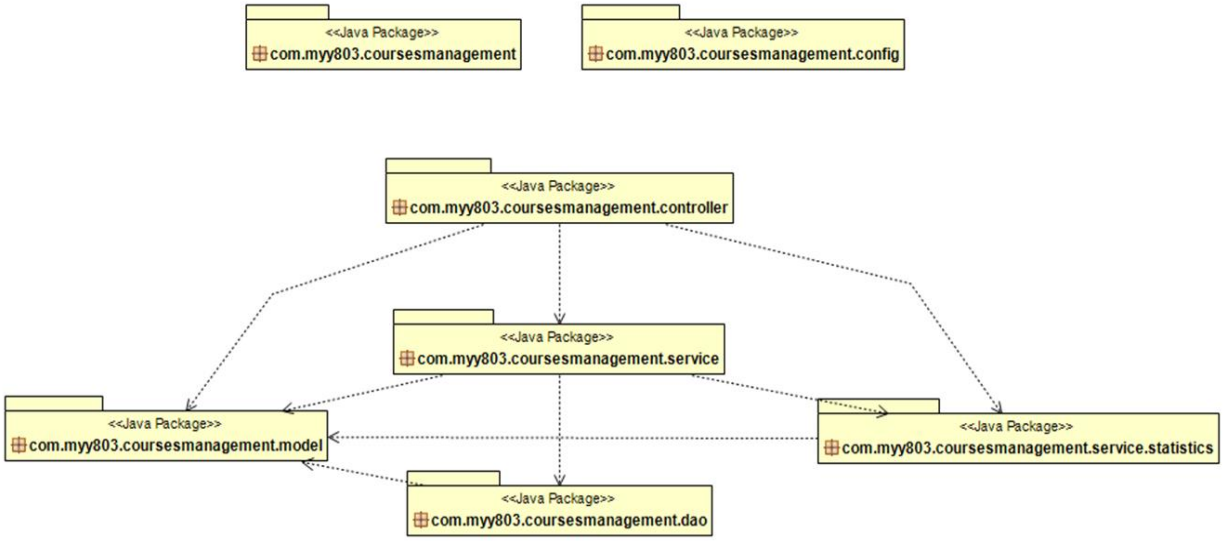
Use case ID	UC9
Actors	Instructor
Preconditions	Instructor successfully logged in.
Main flow of events	1. The use case starts when the instructor presses the Update button for the student that he wants to update. 2. The system shows the form for student update. 3. The instructor makes the changes that he wants and presses the Submit button. 4. The system updates the information of the student (including the grades) and redirects the instructor to students' page.
Alternative flow 1	If the instructor updates the A.M. into an existing A.M. , then the system returns an error page and the system doesn't update the student.
Alternative flow 2	If the instructor makes empty one of the form's fields, then the system informs him that he must fulfill it in order to update the student.
Alternative flow 3	If the instructor presses the Cancel button, then the system does nothing and redirects the instructor to students' page.
Post conditions	The system updates the student.

3.10 UC10: ShowStatistics

Use case ID	UC10
Actors	Instructor
Preconditions	Instructor successfully logged in.
Main flow of events	1. The use case starts when the instructor presses the Statistics button in the students' page. 2. The system redirects the instructor to the statistics page. 3. The instructor can see the statistics for the grades of the current course.
Alternative flow 1	If the instructor hasn't registered yet any students, then the statistics have the NaN price.
Alternative flow 2	If the instructor hasn't registered enough students, then some statistics could be NaN.
Post conditions	None

4 Design

4.1 Architecture



4.2 Design

Class Name: CoursesManagementApplication	
Responsibilities: ▪ This class is the main class of the project.	**Collaborations:** ▪ -

Class Name: SecurityConfig	
Responsibilities: ▪ Configures the usernames and the passwords for the users of the app.	**Collaborations:** ▪ -

Class Name: CoursesMgtAppController	
Responsibilities: ▪ This class is the controller of the web app, so it is the middle class between view classes and model classes. ▪ This class is responsible for mapping the pages (/courses, /students etc.) with the proper html templates.	**Collaborations:** ▪ Course, StudentRegistration ▪ CourseService, StudentRegistrationService ▪ KurtosisStatisticStrategy, MaxStatisticStrategy, MeanStatisticStrategy, MedianStatisticStrategy, MinStatisticStrategy, PercentilesStatisticStrategy, SkewnessStatisticStrategy, StandardDeviationStatisticStrategy, StatisticStrategy, VarianceStatisticStrategy

Class Name: Course	
Responsibilities: ▪ It is the model class for the course, it contains the fields that a course has.	**Collaborations:** ▪ -

Class Name: StudentRegistration	
Responsibilities: ▪ It is the model class for the student registration, it contains the fields that a student registration has.	**Collaborations:** ▪ -

Class Name: CourseService	
Responsibilities: ▪ This class is the interface for the course service that the controller uses to save, delete, etc.	**Collaborations:** ▪ Course ▪ StatisticStrategy

Class Name: CourseServiceImpl	
Responsibilities: ▪ Implements the CourseService interface.	**Collaborations:** ▪ Course ▪ CourseDAO ▪ KurtosisStatisticStrategy, MaxStatisticStrategy, MeanStatisticStrategy, MedianStatisticStrategy, MinStatisticStrategy, PercentilesStatisticStrategy, SkewnessStatisticStrategy, StandardDeviationStatisticStrategy, StatisticStrategy, VarianceStatisticStrategy

Class Name: StudentRegistrationService	
Responsibilities: ▪ This class is the interface for the student registration service that the controller uses to save, delete, etc.	**Collaborations:** ▪ StudentRegistration

Class Name: StudentRegistrationServiceImpl	
Responsibilities: ▪ Implements the StudentRegistrationService interface.	**Collaborations:** ▪ StudentRegistration ▪ StudentRegistrationDAO

Class Name: CourseDAO	
Responsibilities: ▪ It is the interface of the Data Access Object for the course. Interacts with the database for save, delete, etc.	**Collaborations:** ▪ Course

Class Name: CourseDAOImpl	
Responsibilities: ▪ Implements the CourseDAO interface.	**Collaborations:** ▪ Course

Class Name: StudentRegistrationDAO	
Responsibilities: ▪ It is the interface of the Data Access Object for the student registration. Interacts with the database for save, delete, etc.	**Collaborations:** ▪ StudentRegistration

Class Name: StudentRegistrationDAOImpl	
Responsibilities: ▪ Implements the StudentRegistrationDAO interface.	**Collaborations:** ▪ StudentRegistration

Class Name: StatisticStrategy	
Responsibilities: ▪ It is the interface that each statistic class will implement to calculate the statistic price.	**Collaborations:** ▪ Course

Class Name: TemplateStatisticStrategy	
Responsibilities: ▪ It is an abstract class that prepares the grades for the other classes that extend this one to calculate each statistic.	**Collaborations:** ▪ Course ▪ StudentRegistration

Class Name: MinStatisticStrategy	
Responsibilities: ▪ Extends the TemplateStatisticStrategy class and calculates the Min value for the grades of the given course.	**Collaborations:** ▪ -

Class Name: MaxStatisticStrategy	
Responsibilities: ▪ Extends the TemplateStatisticStrategy class and calculates the Max value for the grades of the given course.	**Collaborations:** ▪ -

Class Name: MeanStatisticStrategy	
Responsibilities: ▪ Extends the TemplateStatisticStrategy class and calculates the Mean value for the grades of the given course.	**Collaborations:** ▪ -

Class Name: StandardDeviationStatisticStrategy	
Responsibilities: ▪ Extends the TemplateStatisticStrategy class and calculates the Standard Deviation value for the grades of the given course.	**Collaborations:** ▪ -

Class Name: VarianceStatisticStrategy	
Responsibilities: ▪ Extends the TemplateStatisticStrategy class and calculates the Variance value for the grades of the given course.	**Collaborations:** ▪ -

Class Name: PercentilesStatisticStrategy	
Responsibilities: ▪ Extends the TemplateStatisticStrategy class and calculates the Percentiles value for the grades of the given course.	**Collaborations:** ▪ -

Class Name: KurtosisStatisticStrategy	
Responsibilities: ▪ Extends the TemplateStatisticStrategy class and calculates the Kurtosis value for the grades of the given course.	**Collaborations:** ▪ -

Class Name: SkewnessStatisticStrategy	
Responsibilities: ▪ Extends the TemplateStatisticStrategy class and calculates the Skewness value for the grades of the given course.	**Collaborations:** ▪ -

Class Name: MedianStatisticStrategy	
Responsibilities: ▪ Extends the TemplateStatisticStrategy class and calculates the Median value for the grades of the given course.	**Collaborations:** ▪ -

4.3 UML Class Diagrams

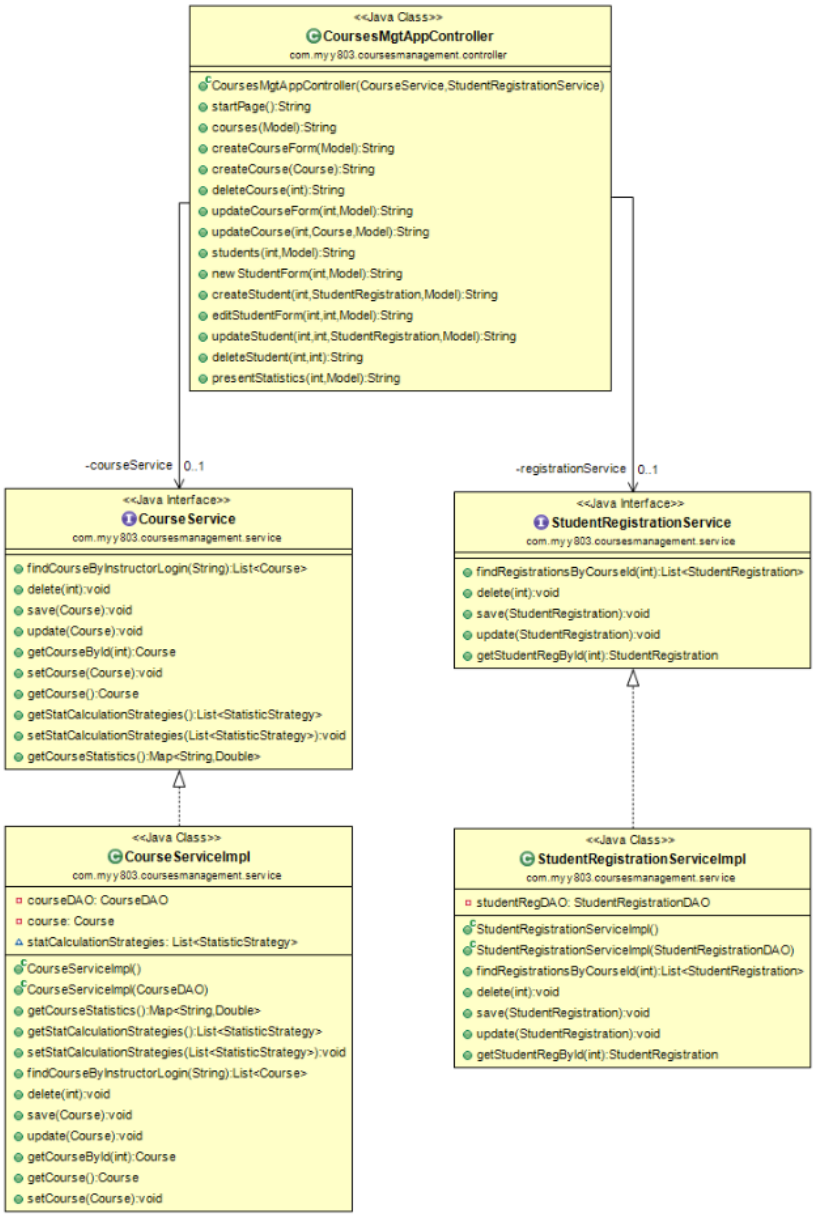


FIG. 1 – CONTROLLER AND SERVICE PACKAGES

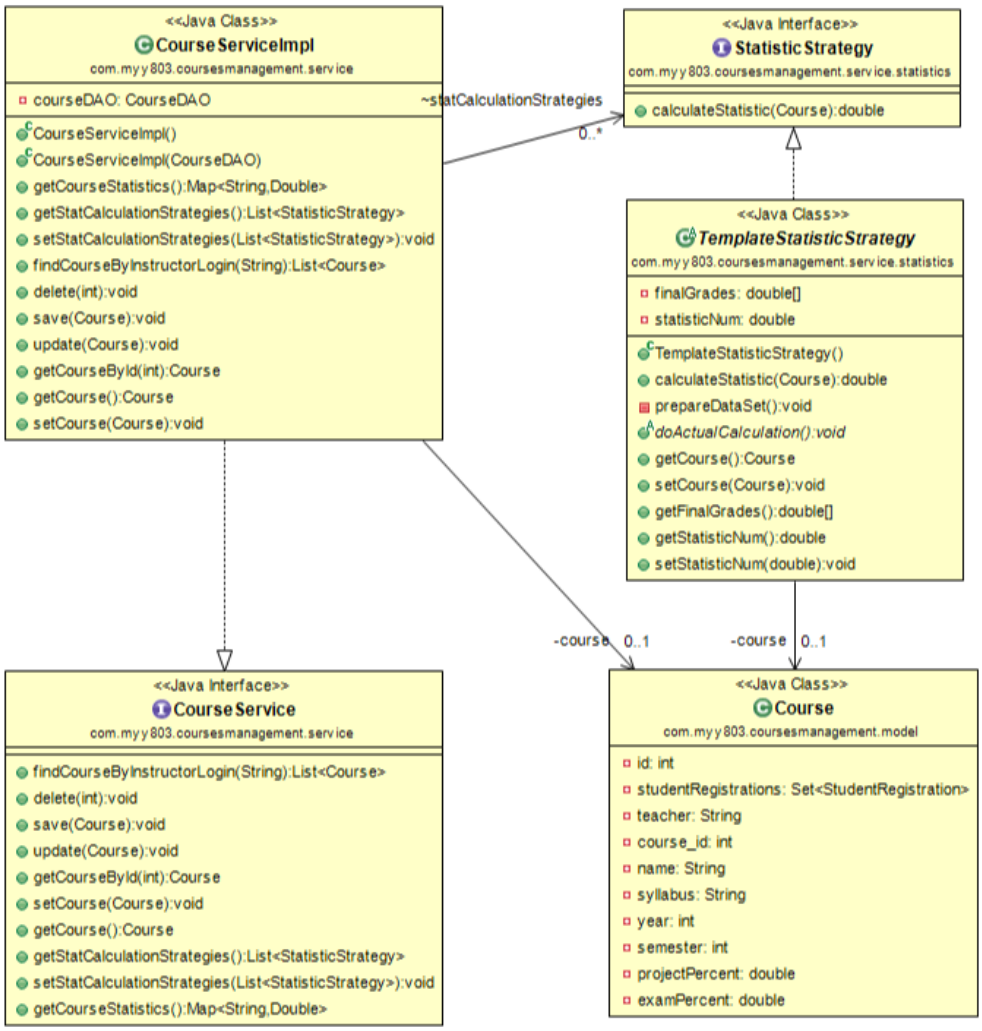


FIGURE 2 - COURSE SERVICE AND STATISTICS

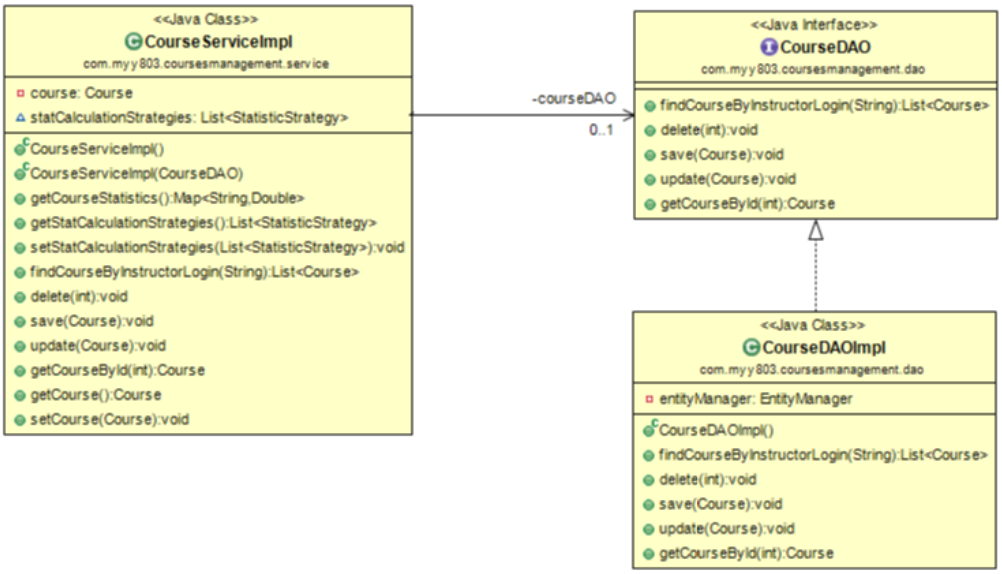


FIGURE 3 - COURSE DAO

FIGURE 4 - STUDENT REGISTRATION DAO