CoursesManagementApp

Sprint Report

Λιόντος Αναστάσιος, 4409

Κίτσιος Κωνσταντίνος, 4388

Βαρδάρης Παναγιώτης, 4590

VERSIONS HISTORY

Date	Version	Description	Author
20/04/2022	0.1	Fill table of sprints	Κωνσταντίνος Κίτσιος, Αναστάσιος Λιόντος, Παναγιώτης Βαρδάρης
14/05/2022	0.2	Write use cases	Κωνσταντίνος Κίτσιος
15/05/2022	0.3	Fix use cases,write CRC cards, insert package UML diagram, use case UML and UML classes diagrams, metrics section	Αναστάσιος Λιόντος, Κωνσταντίνος Κίτσιος Παναγιώτης Βαρδάρης

1. Introduction

This document provides information concerning all sprints of the project.

1.1 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 the project (use cases , test cases). Section 4 specifies the design of the project (architecture, design patterns, UML class/packet diagrams etc.) Section 5 specifies different metrics of the project classes.

2. Scrum team and Sprint Backlog

Scrum team

Product Owner	Αναστάσιος Λιόντος
Scrum Master	Παναγιώτης Βαρδάρης
Development Team	• Αναστάσιος Λιόντος • Κωνσταντίνος Κίτσιος • Παναγιώτης Βαρδάρης

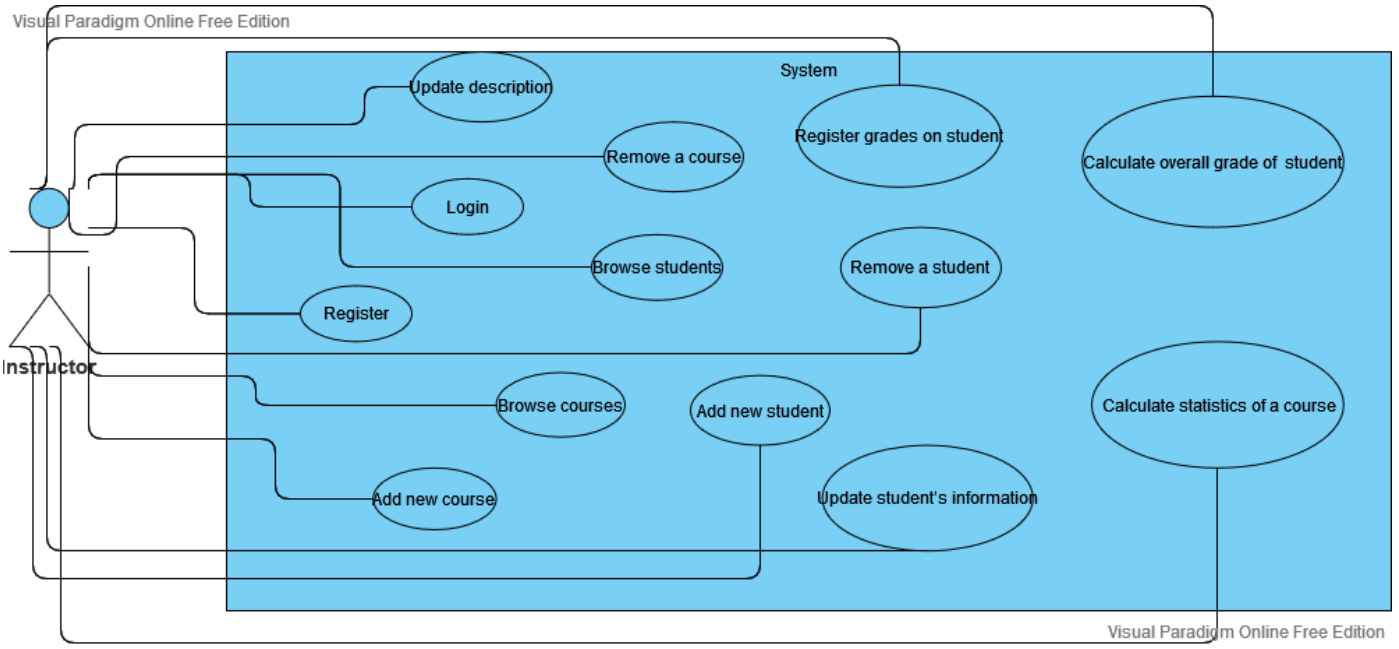
2.1 Sprints

Below is the list of sprints that we performed and the user stories that have been realized in each Sprint.

Sprint No	Begin Date	End Date	Number of weeks	User stories
1	**16/02/2022**	**16/03/2022**	**4**	**US0 - US3**
2	**17/03/2022**	**07/04/2022**	**3**	**US4 - US6**
3	**07/04/2022**	**28/04/20222**	**3**	**US7 - US9**
4	**03/05/2022**	**13/05/2022**	**1-2**	**US10 - US12**

3. Use Cases

At the beginning of this section there is a use case UML diagram and then are specified the concrete Use Cases that describe the interaction of the user with the applications, as delivered from the abstract user stories.



Use Case 0: Register

Use case ID	UC0
Actors	Instructor
Pre conditions	-
Main flow of events	1. The use case starts when the user selects "Register" option in home page of the application. 2. The user fills the properly form with his credentials. 3. The system registers the instructor.
Alternative flow 1	1. If the user leaves an empty field on the form. 2. The system displays an error message.
Post conditions	1. The system saves the instructor on the database.

Use Case 1 : Login

Use case ID	UC1
Actors	Instructor
Pre conditions	-
Main flow of events	1. The use case starts when the user opens webapp. 2. The user enters the credentials for login. 3. The user clicks login button. 4. The system displays home page.
Alternative flow	1. If credentials are incorrect a. The system displays an error message
Post conditions	1. The user is logged in system.

Use Case 2: Browse courses

Use case ID	UC2
Actors	Instructor
Pre conditions	-
Main flow of events	1. The use case starts when the user selects “Courses” option on the home page of application. 2. For each course: 2.1. The system displays title and description of the course. 2.2. If user clicks on “Show students” option. 2.3. The system displays grades of all students.
Post condition	-

Use Case 3: Add new course

Use case ID	UC3
Actors	Instructor
Pre conditions	1. The user has been entered in “Courses” menu
Main flow of events	1. The use case starts when user selects “Add Course” option in courses list. 2. The user gives title, description, year and semester of the new course. 3. The system saves the course.
Alternative flow	1. If the user leaves an empty field. 1.1. The system displays an error message.
Post condition	1. The new course saved in database.

Use Case 4: Remove a course

Use case ID	UC4
Actors	Instructor
Pre conditions	1. The user has been entered in “Courses” menu.
Main flow of events	1. The use case starts when the user selects “Delete” option in courses menu. 2. The user confirms that he want the course to be removed. 3. The system removes selected course.
Post condition	1. The selected course has deleted from database.

Use Case 5: Update description of a course

Use case ID	UC5
Actors	Instructor
Pre conditions	1. There must already exist a course.
Main flow of events	1. The use case starts when the user selects “Update” option in a particular course . 2. The user fills the properly form with a new description of the course. 3. The system updates the description of the course.
Post condition	1. The selected course has the updated description.

Use Case 6: Browse students

Use case ID	UC6
Actors	Instructor
Pre conditions	1. The course already exists and the students have enrolled a particular course.
Main flow of events	1. The use case starts when the user selects “Show students” option in the courses menu. 2. For each student: 2.1. The system displays id, name, semester, year and grades of the student. 2.2. If the user click “Show Students” option: 2.3. The system displays a list of students of a particular course.
Post condition	-

Use Case 7: Add new student

Use case ID	UC7
Actors	Instructor
Pre conditions	1. A course must exist.
Main flow of events	1. The use case starts when the user selects “Add student” in students menu. 2. The user fills the properly form with student’s name, year, semester. 3. The system saves the student.
Post condition	1. The new student saved in database.

Use Case 8: Remove a student

Use case ID	UC8
Actors	Instructor
Pre conditions	1. A student must be enrolled in a course.
Main flow of events	1. The use case starts when the user selects “Delete” option on a student in students menu. 2. The user confirms that he want the student to be removed. 3. The system deletes the student.
Post condition	1. The selected student has been removed from the database.

Use Case 9: Update student's information

Use case ID	UC9
Actors	Instructor
Pre conditions	1. There must be a student enrolled on a course.
Main flow of events	1. The use case starts when the user selects "Update" option on a particular student in student menu. 2. The user fills the properly form with the new information of the student. 3. The system updates student's information.
Post condition	1. The selected student has the updated information.

Use Case 10: Register grades on student

Use case ID	UC10
Actors	Instructor
Pre conditions	1. There must be a student enrolled on a course.
Main flow of events	1. The use case starts when the user selects "Add Grades" option on a student in students menu. 2. The user fills the properly form with student's grade (project grade, exam grade, project weight and exam weight). 3. The system updates student's grades.
Post condition	1. The grades of selected student have been updated.

Use Case 11: Calculate overall grade of student

Use case ID	UC11
Actors	Instructor
Pre conditions	1. There must be a student enrolled on a course and the student must have project grade and exam grade.
Main flow of events	1. The use case starts when the user selects “Save” option on a student’s grades form. 2. The system calculates overall grade of the student. 3. The system updates student’s overall grade.
Post condition	1. The overall grade of selected student has been updated.

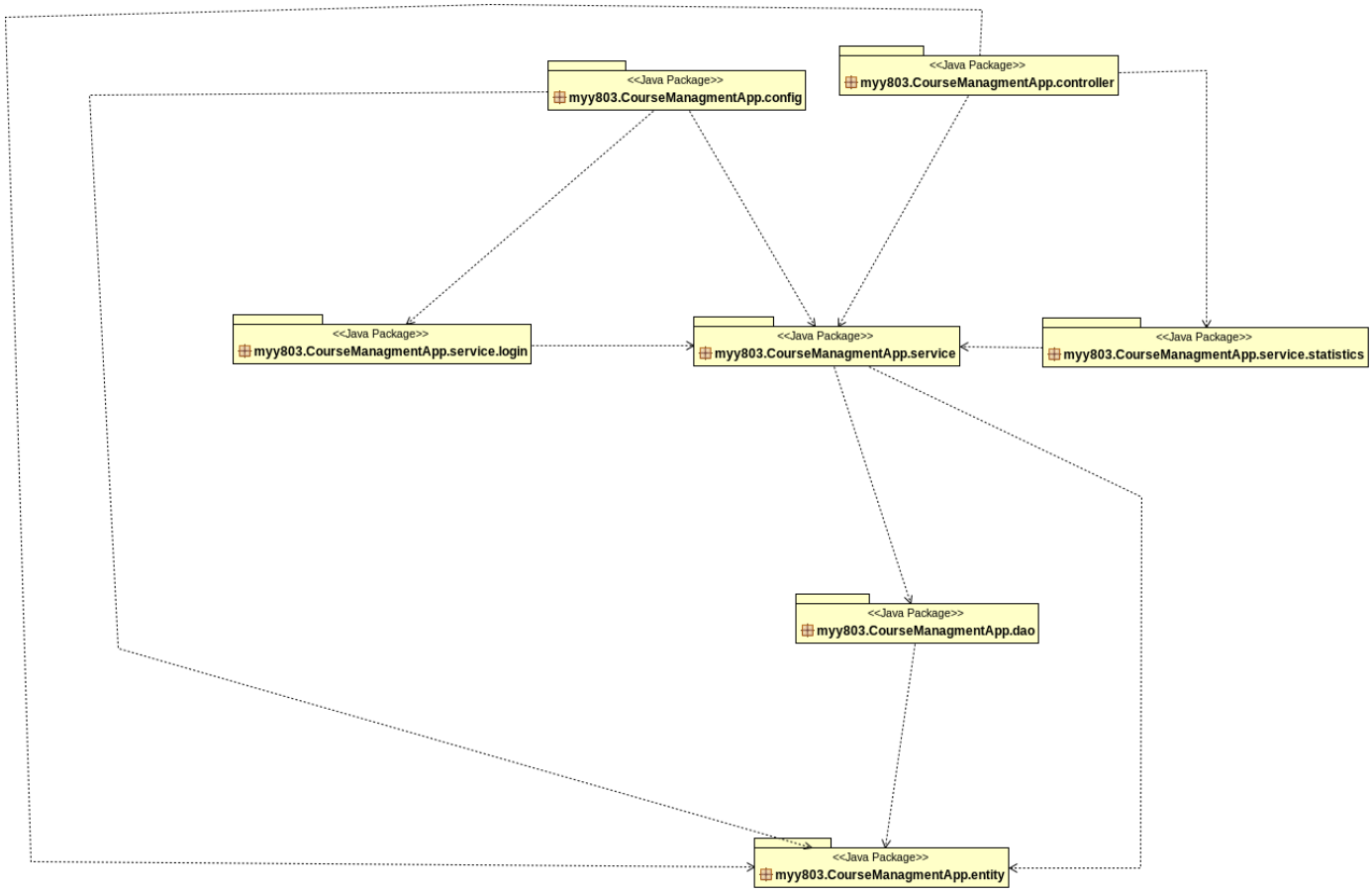
Use Case12: Calculate statistics of a course

Use case ID	UC12
Actors	Instructor
Pre conditions	1. There must be students with grades that enrolled on a course.
Main flow of events	1. The use case starts when the user selects “Show statistics” option on a course in course menu. 2. The user selects which statistic should be calculated. 3. The system prepares a dataset with students' grades. 4. The system calculates selected statistic based on the dataset. 5. The system displays the statistic’s calculated value.
Alternative flow 1	1. If the user selects skewness or percentile statistic and dataset has less than 5 values: 1.1. The system displays NaN.
Alternative flow 2	1. If the users selects percentile statistics without setup: 1.1. The system displays an error message.
Post condition	-

4. Design

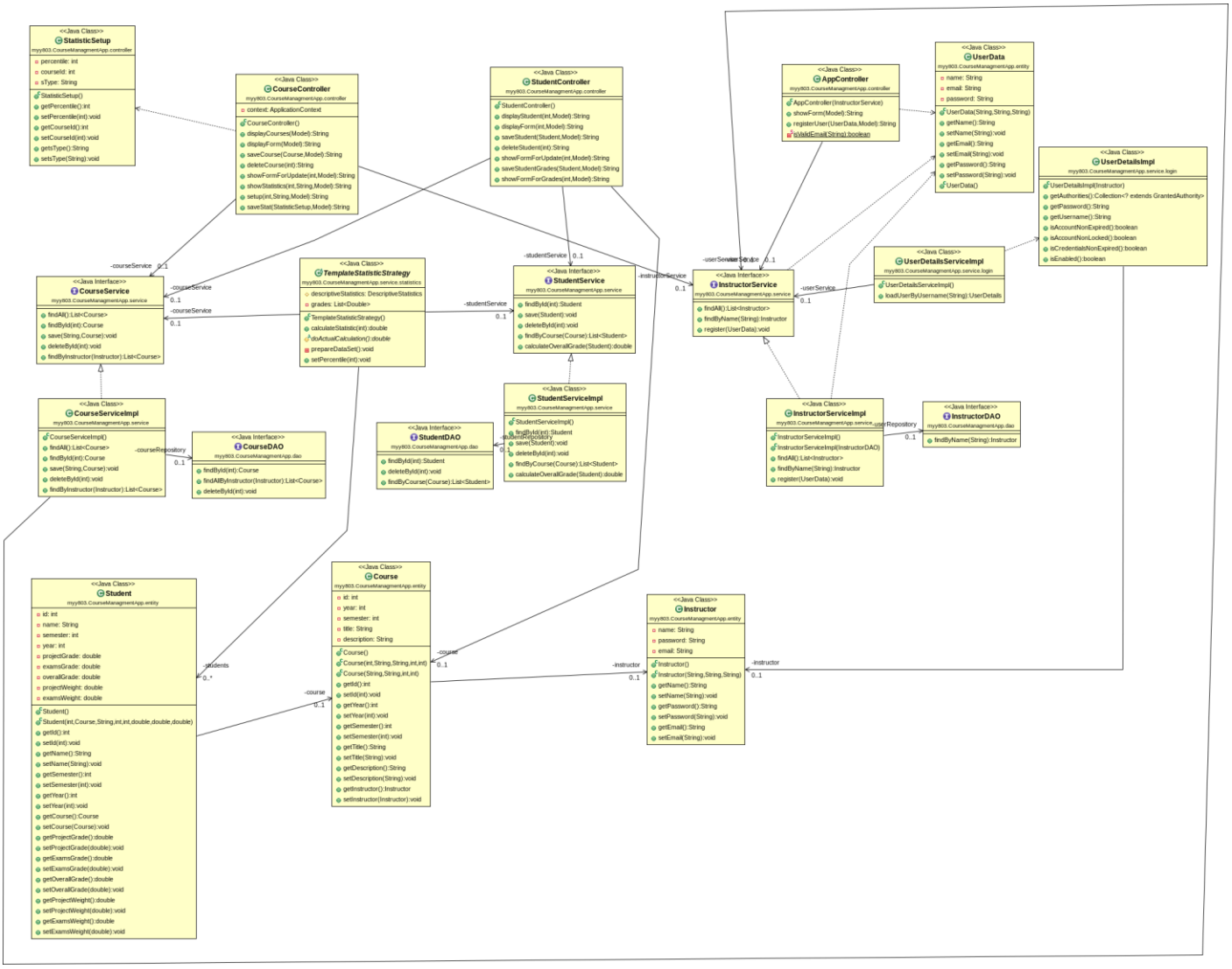
4.1 Architecture

The system's architecture is shown in the packet diagram below.



4.2 Design

The detailed design (except association relationships from all other classes to entity classes for simplicity reasons) of the system is shown in the UML class below.



CRC cards for package: myy803.CourseManagmentApp.entity

Class Name: Course	
Responsibilities: 1. id 2. year 3. semester 4. title 5. description 6. instructor	**Collaborations:** 1. myy803.CourseManagmentApp.entity.Instructor

Class Name: Student	
Responsibilities: 1. id 2. course 3. name 4. semester 5. year 6. projectGrade 7. examGrade 8. overallGrade	**Collaborations:** 1. myy803.CourseManagmentApp.entity.Course

Class Name: Instructor	
Responsibilities: 1. name 2. password 3. email	**Collaborations:**

CRC cards for package : myy803.CourseManagmentApp.service

Class Name: CourseServiceImpl	
Responsibilities: 1. courseRepository 2. courseService 3. findAll() 4. findById()	**Collaborations:** 1. myy803.CourseManagmentApp.dao.CourseDAO 2. myy803.CourseManagmentApp.service.CourseService 3. myy803.CourseManagmentApp.service.InstructorService

Class Name: StudentServiceImpl	
Responsibilities: 1. studentRepository 2. findById() 3. deleteById() 4. findByCourse()	**Collaborations:** 1. myy803.CourseManagmentApp.dao.StudentDAO 2. myy803.CourseManagmentApp.service.StudentService

Class Name: InstructorServiceImpl	
Responsibilities: 1. userRepository 2. findAll() 3. findByName() 4. register()	**Collaborations:** 1. myy803.CourseManagmentApp.dao.InstructorDAO 2. myy803.CourseManagmentApp.service.InstructorService 3. myy803.CourseManagmentApp.entity.UserData

CRC cards for package : myy803.CourseManagmentApp.service.login

Class Name: UserDetailsServiceImpl	
Responsibilities: 1. instructorService 2. loadUserByUsername() 	**Collaborations:** 1.myy803.CourseManagmentApp.service.login.UserDetailsImpl 2.myy803.CourseManagmentApp.service.InstructorService

Class Name: UserDetailsImpl	
Responsibilities: 1. instructor	**Collaborations:** 1. myy803.courseManagmentApp.entity.Instructor

CRC cards for package : myy803.CourseManagmentApp.service.statistics

Class Name: TemplateStatisticStrategy	
Responsibilities: 1. courseService 2. studentService 3. descriptiveStatistics 4. list of students 5. list of grades 6. calculateStatistic() 7. doActualCalculation() 8. prepareDataSet() 9. setPercentile()	**Collaborations:** 1. myy803.CourseManagmentApp.service.CourseService 2. myy803.CourseManagmentApp.service.StudentService 3. myy803.CourseManagmentApp.entity.Student

Class Name: KurtosisStatisticStrategy	
Responsibilities: 1. doActualCalculation()	**Collaborations:** 1. TemplateStatisticStrategy

Class Name: MaxStatisticStrategy	
Responsibilities: 1. doActualCalculation()	**Collaborations:** 1. TemplateStatisticStrategy

Class Name: MinStatisticStrategy	
Responsibilities: 1. doActualCalculation()	**Collaborations:** 1. TemplateStatisticStrategy

Class Name: MeanStatisticStrategy	
Responsibilities: 1. doActualCalculation()	**Collaborations:** 1. TemplateStatisticStrategy

Class Name: StandardDeviationStatisticStrategy	
Responsibilities: 1. doActualCalculation()	**Collaborations:** 1. TemplateStatisticStrategy

Class Name: VariancesStatisticStrategy	
Responsibilities: 1. doActualCalculation()	**Collaborations:** 1. TemplateStatisticStrategy

Class Name: PercentileStatisticStrategy	
Responsibilities: 1. doActualCalculation()	**Collaborations:** 1. TemplateStatisticStrategy

Class Name: MedianStatisticStrategy	
Responsibilities: 1. doActualCalculation()	**Collaborations:** 1. TemplateStatisticStrategy

Class Name: SkewnessStatisticStrategy	
Responsibilities: 1. doActualCalculation()	**Collaborations:** 1. TemplateStatisticStrategy

CRC cards for package : myy803.CourseManagmentApp.controller

Class Name: AppController	
Responsibilities: 1. instructorService 2. showForm() 3. registerUser() 4. isValidEmail()	**Collaborations:** 1. myy803.CourseManagmentApp.service.InstructorService 2. myy803.CourseManagmentApp.entity.UserData

Class Name: CourseController	
Responsibilities: 1. courseService 2. InstructorService 3. displayCourse() 4. displayForm() 5. saveCourse() 6. deleteCourse() 7. showFormForUpdate() 8. showStatistics() 9. setup() 10. saveStat()	**Collaborations:** 1. myy803.CourseManagmentApp.service.CourseService 2. myy803.CourseManagmentApp.controller.StatisticSetup 3. myy803.CourseManagmentApp.service.InstructorService

Class Name: StudentController	
Responsibilities: 1. studentService 2. courseService 3. course 4. displayStudent() 5. displayForm() 6. saveStudent() 7. deleteStudent() 8. showFormForUpdate() 9. saveStudentGrades() 10. showFormForGrades()	**Collaborations:** 1. myy803.CourseManagmentApp.service.StudentService 2. myy803.CourseManagmentApp.service.CourseService 3. myy803.CourseManagmentApp.entity.Course

5. Metrics

Class Name	CBO	LCOM3	WMC	RFC
AppControler	3	1	9	14
CourseControler	6	1	16	34
StudentControler	4	1	18	25
StudentServiceImpl	3	1	5	14
CourseServiceImpl	4	1	5	12
InstructorServiceImpl	3	1	5	15
UserDataServiceImpl	3	1	2	2
UserDataImpl	1	1	8	10
CourseDAO	2	1	3	3
StudentDAO	2	1	3	3
InstructorDAO	1	1	1	1
KurtosisStatisticStrategy	0	1	2	3
SkewnessStatisticStrategy	0	1	2	3
PercentileStatisticStrategy	0	1	3	4
MaxStatisticStrategy	0	1	2	3
MinStatisticStrategy	0	1	2	3
MeanStatisticStrategy	0	1	2	3
MedianStatisticStrategy	0	1	2	3
StandardDeviationStatisticStrategy	0	1	2	3
VarianceStatistitcStrategy	0	1	2	3
StatisticStrategy	0	1	2	2

TemplateStatisticStrategy	4	1	7	9
Course	2	1	3	3
Student	2	1	3	3
Instructor	1	1	1	1

6. Citation

- Application background image: Vecteezy