# CoursesManagementApp

# Sprint Report

<GangOfThree(4-1)>
<Κωνσταντίνος Ορφανούδης-3303,
Παρασκευάς Πέττης-3136,
Σταύρος Τσούκλας-4285>

# VERSIONS HISTORY

| Date | Version | Description | Author |
|---|---|---|---|
| 17-02-2022 | v.1 | Model classes and DAO layer,my sql connection and started testing for model classes and DAO.In addition maven dependencies included and writing use cases. | Κώστας,Παρης,Σταύρος |
| 01-03-2022 | v.2 | Service layer and adding extra model classes(instructor,student,secretariat). | Κώστας,Παρης,Σταύρος |
| 10-03-2022 | v.3 | MVC(controllers)for instructor,student and secretariat | Κώστας,Παρης,Σταύρος |
| 25-03-2022 | v.4 | Designing HTML pages and checking login and register user stories.Starting testing for service layer. | Κώστας,Παρης,Σταύρος |
| 6-04-2022 | v.5 | Continuing user stories (given and extra) | Κώστας,Παρης,Σταύρος |
| 10-05-2022 | v.6 | Finished user stories | Κώστας,Παρης,Σταύρος |
| 16-05-2022 | v.7 | Video presentation | Κώστας,Παρης,Σταύρος |

# 1 Introduction

This document provides information concerning the **<1>** sprint of the project.

## 1.1 Purpose

Στην πρώτη φάση θέλαμε να καταλάβουμε πως θα συνδέσουμε την my-sql βάση και να δημιουργήσουμε τις οντότητες(πίνακες στη βάση). Μετά με το JPA repository φτιάξαμε τα interfaces για την διασύνδεση με την βάση(save,update,delete,find by).Τεστάρουμε τα models και τα DAOS για να ελέγξουμε την σωστή λειτουργία τους και να προχωρήσουμε στα επόμενα επίπεδα.

## 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

<For the user stories included in this release specify below corresponding tests using a typical tabular form.>

## 2.1 Scrum team

| | |
|---|---|
| **Product Owner** | Mr Zarras |
| **Scrum Master** | Kostas,Paris,Stavros |
| **Development Team** | Kostas,Paris,Stavros |

## 2.2 Sprints 2

**<List below the sprints that you performed and the user stories that have been realized in each Sprint>**

| Sprint No | Begin Date | End Date | Number of weeks | User stories |
|---|---|---|---|---|
| 1 | 17-02-2022 | 01-03-2022 | 2 | ---------- |
| 2 | 01-03-2022 | 10-03-2022 | 1 | ----------- |
| 3 | 10-03-2022 | 25-03-2022 | 2 | ------------- |
| 4 | 25-03-2022 | 5-04-2022 | 2 | Login and register<br>Add course,add StudentRegistration,list of courses,list of StudentRegistrations, |
| 5 | 6-04-2022 | 10-05-2022 | 4 | List of instructor's courses,list of students in my course, edit the weights of exam and project of my courses,<br>delete students in the course register grades project (if my course has project) only once, register grades exam, calculate overall grades of students per course, calculate statistics, add delete update courses, add delete update instructors-students, list of courses passed<br>list of courses (current semester) register my courses this semester, calculate my average grade(of all my passed courses), history with all attempts |
| 6 | 16-05-2022 | 17-05-2022 | 1 | Video presentation |

# 3 Use Cases

<Specify the concrete Use Cases that describe the interaction of the user with the applications, as derived from the abstract user stories. Give a **UML Use Case diagram** and the **detailed use case descriptions**.>

## 3.1 <ShowMyCourses>

| | |
|---|---|
| **Use case ID** | UC1 |
| **Actors** | Instructor |
| **Pre conditions** | The Instructor has logged into the system with username and password. |
| **Main flow of events** | 1.The use case starts when the instructor selects from the menu <<ShowMyCourses>><br>2.The system outputs the list of instructor's courses according to instructor's login. |
| **Alternative flow 1** | 1.At any time the instructor may leave the system. |
| **Post conditions** | The instructor receives the list of his courses. |

## 3.2 <ShowRegistrationsInMyCourse>

| | |
|---|---|
| **Use case ID** | UC2 |
| **Actors** | Instructor |
| **Pre conditions** | The Instructor has logged into the system with username and password. |
| **Main flow of events** | 1.The use case starts when the instructor selects from the menu <<ShowRegistrationsInMyCourse>><br>2.The system performs the UC1<br>2.The system asks the instructor to enter the courseId of the preferred course.<br>3.The instructor enters the courseId.<br>4.While the instructor types wrong courseId:<br>4.1 The system asks the instructor to enter one courseId from the above list of courses.<br>4.2 The instructor types wrong coursed.<br>5.The Instructor types correct coursed from the above list.<br>6.The system outputs the registrations for the selected course. |
| **Alternative flow 1** | 1.At any time the instructor may leave the system. |

| | |
|---|---|
| **Post conditions** | The instructor receives the list of registrations for the selected course. |

## 3.3 <DeleteStudentsInMyCourse>

| | |
|---|---|
| **Use case ID** | UC3 |
| **Actors** | Instructor |
| **Pre conditions** | The Instructor has logged into the system with username and password. |
| **Main flow of events** | 1.The use case starts when the instructor selects from the menu <<DeleteStudentsInMyCourse>><br>2.The system goes to step 2 of UC2.<br>2.The system asks the instructor to enter the registrationId of the preferred course.<br>3.The instructor enters the registrationId.<br>4.While the instructor types wrong registrationId:<br>4.1 The system asks the instructor to enter one registrationId from the above list of registartions.<br>4.2 The instructor types registrationId.<br>5.The Instructor types correct registrationId from the above list.<br>6.The system deletes the registration and outputs the updated list of registrations of the course. |
| **Alternative flow 1** | 1.At any time the instructor may leave the system. |
| **Post conditions** | The instructor deleted the registration from the list of registrations of the course. |

## 3.4 <EditProjectweightInMyCourse>

| | |
|---|---|
| **Use case ID** | UC4 |
| **Actors** | Instructor |
| **Pre conditions** | The Instructor has logged into the system with username and password. |
| **Main flow of events** | 1.The use case starts when the instructor selects from the menu <<EditProjectWeightInMyCourse>><br>2.The system performs the UC1<br>2.The system asks the instructor to enter the courseId of the preferred course.<br>3.The instructor enters the courseId.<br>4.While the instructor types wrong courseId:<br>4.1 The system asks the instructor to enter one courseId from the above list of courses.<br>4.2 The instructor types wrong coursed.<br>5.The Instructor types correct coursed from the above list.<br>6. If the course has project:<br>6.1. The system asks the instructor to enter a positive float less than 1.<br>6.2 While the instructor enters a wrong project-weight:<br>6.2.1 The system asks the instructor to enter positive float less than 1.<br>6.2.2 The instructor enters a project-weight.<br>6.3 The system computes the exam-weight as 1 - project-weight.<br>7. else:<br>7.1 The system outputs error ”The selected course doesn’t have project” and returns to the main menu |
| **Alternative flow 1** | 1.At any time the instructor may leave the system. |
| **Post conditions** | The instructor has registered the project-weight for the selected course. |

## 3.5 <RegisterExamGradesInMyCourse>

| | |
|---|---|
| **Use case ID** | UC5 |
| **Actors** | Instructor |
| **Pre conditions** | The Instructor has logged into the system with username and password. |
| **Main flow of events** | 1.The use case starts when the instructor selects from the menu <<RegisterExamGradesInMyCourse>><br>2.The system goes to step2 of UC1.<br>3.The system asks the instructor to select the preferred course.<br>4. While the instructor types a wrong courseId:<br>4.1 The system asks the instructor to enter a courseId from the above list<br>4.2 The instructor types a courseId<br>5. The system goes to step 2 of UC2.<br>6. for each registration in the registartion’s list:<br>6.1 The system asks the instructor to enter the examGrade.<br>6.2 While the Instructor enters a negative grade or a grade bigger than 10:<br>6.2.1: The system asks the instructor to enter a positive grade smaller or equal to 10.<br>6.2.2: The instructor enters a grade.<br>6.3. The system updates the registration with the exam grade.<br>7. The system outputs the updated list with the registrations. |
| **Alternative flow 1** | 1.At any time the instructor may leave the system. |
| **Post conditions** | The instructor has registered the exam grades of the registrations from the list of registrations of the course. |

## 3.6 <RegisterProjectGradesInMyCourse>

| | |
|---|---|
| **Use case ID** | UC6 |
| **Actors** | Instructor |
| **Pre conditions** | The Instructor has logged into the system with username and password. |
| **Main flow of events** | 1.The use case starts when the instructor selects from the menu <<RegisterProjectGradesInMyCourse>><br>2.The system goes to step2 of UC1.<br>3.The system asks the instructor to select the preferred course.<br>4. While the instructor types a wrong courseId:<br>4.1 The system asks the instructor to enter a courseId from the above list<br>4.2 The instructor types a courseId<br>5.If the course has project:<br>5.1 The system goes to step 2 of UC2.<br>5.2.for each registration in the registartion's list:<br>5.2.1 The system asks the instructor to enter the projectGrade.<br>5.2.2 While the Instructor enters a negative grade or a grade bigger than 10:<br>5.2.2.1: The system asks the instructor to enter a positive grade smaller or equal to 10.<br>5.2.2.2: The instructor enters a grade.<br>5.2.3 The system updates the registration with the exam grade.<br>5.3 The system outputs the updated list with the registrations.<br>6.else:<br>6.1 The system goes to step 2 of the current use case. |
| **Alternative flow 1** | 1.At any time the instructor may leave the system. |

| | |
|---|---|
| **Post conditions** | The instructor has registered the project grades of the registrations from the list of registrations of the course. |

## 3.7 <CalculateOverallGradesInCourse>

| | |
|---|---|
| **Use case ID** | UC7 |
| **Actors** | Instructor,Secretariat |
| **Pre conditions** | The Instructor or the Secretariat has logged into the system with username and password and the instructor has registered project-weight with UC4. |
| **Main flow of events** | 1.The use case starts when the instructor selects from the menu <<CalculateOverallGradesInCourse>><br>2.The system goes to step2 of UC1.<br>3.The system asks the instructor or the secretariat to select the preferred course.<br>4. While the instructor or the secretariat types a wrong courseId:<br>4.1 The system asks the instructor or the secreatriat to enter a courseId from the above list.<br>4.2 The instructor or the secreatriat types a courseId.<br>5.If the course has project:<br>5.1 The system goes to step 2 of UC2.<br>5.2.for each registration in the registartion's list:<br>5.2.1 The system computes the overall grades according to the weights.<br>6.else:<br>6.1 The system calculates the average of exam-grade and project-grade as overall-grade.<br>7.The system outputs the list with the overall grades of the registrations in the course. |
| **Alternative flow 1** | 1.At any time the instructor or the secreatriat may leave the system. |

| | |
|---|---|
| **Post conditions** | The instructor or the secretariat has calculated the overall-grades of the registrations from the list of registrations of the course. |

## 3.8 <CalculateDescriptiveStatisticsGradesInCourse>

| | |
|---|---|
| **Use case ID** | UC8 |
| **Actors** | Instructor,Secretariat |
| **Pre conditions** | The Instructor or the Secretariat has logged into the system with username and password and has computed the overall grades of the registrations in the course. |
| **Main flow of events** | 1.The use case starts when the instructor selects from the menu <<CalculateDescriptiveStatisticsInCourse>><br>2.The system goes to step2 of UC1.<br>3.The system asks the instructor to select the preferred course.<br>4. While the instructor types a wrong courseId:<br>4.1 The system asks the instructor to enter a courseId from the above list.<br>4.2 The instructor types a courseId.<br>5 The system gets the overall grades of the students in the course and<br>calculates the descriptive statistics<br>6. The system outputs a Map with the statistic's name as a key and it's value as<br>Value |
| **Alternative flow 1** | 1.At any time the instructor or the secreatriat may leave the system. |
| **Post conditions** | The instructor or the secretariat has calculated the descriptive statistics of the registrations from the list of registrations of the course. |

## 3.9 <ShowAllCourses>

| | |
|---|---|
| **Use case ID** | UC9 |
| **Actors** | Secretariat,Student |
| **Pre conditions** | The Secreatariat or the student has logged into the system with username and password. |
| **Main flow of events** | 1.The use case starts when the secretariat or the student selects from the menu <<ShowAllCourses>><br>2.The system outputs a list with all the courses |
| **Alternative flow 1** | 1.At any time the secreatariat or the student may leave the system. |
| **Post conditions** | The secretariat or the student gets the list with all the courses. |

## 3.10 <AddCourse>

| | |
|---|---|
| **Use case ID** | UC10 |
| **Actors** | Secretariat |
| **Pre conditions** | The Secreatariat has logged into the system with username and password. |
| **Main flow of events** | 1.The use case starts when the secretariat selects from the menu <<AddCourse>><br>2.The system outputs a list with all the courses and asks the secretariat to enter a coursed which is not contained in the list of courses.<br>3. The secretariat enters a courseId<br>4.While the secretariat enters a contained courseId:<br>4.1 The system asks the secretariat to enter a non-contained courseId<br>4.2 The secretariat enters a courseId<br>5. The systems asks the secretariat to enter a non-contained name for the course<br>6. The secretariat enters a name for the course<br>7. While the secretariat enters a contained name<br>7.1 The system asks the secretariat to enter a non-contained name<br>7.2 The secretariat enters a name for the course. |

| | |
|---|---|
| | 8. The system asks the secretariat to enter a course-syllabus.<br>9.The secreatariat enters a course-syllabus.<br>10.While the secretariat enters a contained course-syllabus:<br>10.1 The system asks the secretariat to enter a non-contained course- syllabus.<br>10.2 The secretariat enters a course-syllabus<br>11.The system asks the secretariat to enter the year for the course between 1 and 5<br>12 The secretariat enters a year<br>13.While the secretariat enters a non valid year:<br>13.1.The system asks the secretariat to enter a year between 1 and 5<br>13.2 The secretariat enters a year<br>13.The system asks the secretariat to enter the semester of the course.<br>14. The secretariat enters a semester<br>14.While the secretariat enters a non-valid semester:<br>14.1 The system asks the secretariat to enter a valid semester.<br>14.2 The secretariat enters a semester for the course.<br>15 The system asks the secretariat to enter an instructorLogin for the course.<br>16The secretariat enters an instructorLogin<br>17.While the secretariat enter a non-existing instructorLogin:<br>17.1 The system asks the secretariat to enter another instructorLogin<br>17.2 The secretariat enters an instructorLogin.<br>18. The secretariat adds the course. |
| **Alternative flow 16** | 1.At any time the secreatariat may leave the system. |
| **Post conditions** | The secretariat adds a course  into the list of courses. |

## 3.11 <UpdateCourse>

| | |
|---|---|
| **Use case ID** | UC11 |

| **Actors** | Secretariat |
|---|---|
| **Pre conditions** | The Secreatariat has logged into the system with username and password. |
| **Main flow of events** | 1.The use case starts when the secretariat selects from the menu <<UpdateCourse>><br>2.The system outputs a list with all the courses and asks the secretariat to enter a coursed which is contained in the list of courses.<br>3. The secretariat enters a courseId<br>4.While the secretariat enters a non-contained courseId:<br>4.1 The system asks the secretariat to enter a contained courseId<br>4.2 The secretariat enters a courseId<br>5. The systems asks the secretariat to enter a non-contained name for the course<br>6. The secretariat enters a name for the course<br>7. While the secretariat enters a non-unique name<br>7.1 The system asks the secretariat to enter a non- unique name<br>7.2 The secretariat enters a name for the course.<br>8. The system asks the secretariat to enter a course-syllabus.<br>9.The secreatariat enters a course-syllabus.<br>10.While the secretariat enters a non-unique course-syllabus:<br>10.1 The system asks the secretariat to enter a unique course- syllabus.<br>10.2 The secretariat enters a course-syllabus<br>11.The system asks the secretariat to enter the year for the course between 1 and 5<br>12 The secretariat enters a year<br>13.While the secretariat enters a non valid year:<br>13.1.The system asks the secretariat to enter a year between 1 and 5<br>13.2 The secretariat enters a year<br>13.The system asks the secretariat to enter the semester of the course.<br>14. The secretariat enters a semester<br>14.While the secretariat enters a non-valid semester: |

| | |
|---|---|
| | 14.1 The system asks the secretariat to enter a valid semester.<br>14.2 The secretariat enters a semester for the course.<br>15 The system asks the secretariat to enter an instructorLogin for the course.<br>16The secretariat enters an instructorLogin<br>17.While the secretariat enter a non-existing instructorLogin:<br>17.1 The system asks the secretariat to enter another instructorLogin<br>17.2 The secretariat enters an instructorLogin.<br>18. The secretariat updates the course. |
| **Alternative flow 16** | 1.At any time the secreatariat may leave the system. |
| **Post conditions** | The secretariat updates a course from the list of courses. |

## 3.13 <DeleteCourse>

| | |
|---|---|
| **Use case ID** | UC13 |
| **Actors** | Secretariat |
| **Pre conditions** | The Secretariat has logged into the system with username and password. |
| **Main flow of events** | 1.The use case starts when the instructor selects from the menu <<DeleteCourse>><br>2.The system goes to step 2 of UC9.<br>2.The system asks the instructor to enter the courseId of the preferred course.<br>3.The instructor enters the courseId.<br>4.While the instructor types a non-existing courseId:<br>4.1 The system asks the instructor to enter one courseId from the above list of all courses.<br>4.2 The instructor types a courseId.<br>5.The Instructor types correct courseIdId from the above list.<br>6.The system deletes the course and outputs the updated list of of the courses. |

| | |
|---|---|
| **Alternative flow 1** | 1.At any time the instructor may leave the system. |
| **Post conditions** | The instructor deleted the course from the list of all courses. |

## 3.14 <ShowAllInstructors>

| | |
|---|---|
| **Use case ID** | UC14 |
| **Actors** | Secretariat |
| **Pre conditions** | The Secreatariat has logged into the system with username and password. |
| **Main flow of events** | 1.The use case starts when the secretariat selects from the menu <<ShowAllInstructors>><br>2.The system outputs a list with all the instructors |
| **Alternative flow 1** | 1.At any time the secretariat may leave the system. |
| **Post conditions** | The secretariat gets the list with all the instructors. |

## 3.15 <AddInstructor>

| | |
|---|---|
| **Use case ID** | UC15 |
| **Actors** | Secretariat |
| **Pre conditions** | The Secreatariat has logged into the system with username and password. |
| **Main flow of events** | 1.The use case starts when the secretariat selects from the menu <<AddInstructor>><br>2.The system outputs a list with all the instructors and asks the secretariat to enter an instructorName which is not contained in the list of instructors.<br>3. The secretariat enters an instructorName<br>4.While the secretariat enters a contained instructorName: |

| | |
|---|---|
| | 4.1 The system asks the secretariat to enter a non-contained instructorName<br>4.2 The secretariat enters an instructorName<br>5. The systems asks the secretariat to enter a non-contained username for the instructor<br>6. The secretariat enters a username for the instructor.<br>7. While the secretariat enters a contained username.<br>7.1 The system asks the secretariat to enter a non-contained username.<br>7.2 The secretariat enters a username for the instructor.<br>8. The system asks the secretariat to enter a password for the instructor.<br>9.The secreatariat enters a password for the instructor.<br>10.While the secretariat enters a non-valid password:<br>10.1 The system asks the secretariat to enter a valid password for the instructor.<br>10.2 The secretariat enters a password for the instructor.<br>11 The system asks the secretariat to enter an email for the instructor<br>12 The secretariat enters an email for the instructor.<br>13.While the secretariat enter an existing email:<br>13.1 The system asks the secretariat to enter another email for the instructor<br>13.2 The secretariat enters an email for the instructor.<br>14 The system asks the secretariat to enter a telephone for the instructor<br>15 The secretariat enters a telephone for the instructor.<br>16.While the secretariat enter an existing telephone:<br>16.1 The system asks the secretariat to enter another telephone for the instructor<br>16.2 The secretariat enters a telephone for the instructor. |
| **Alternative flow 16** | 1.At any time the secreatariat may leave the system. |
| **Post conditions** | The secretariat adds an instructor into the list of instructors. |

## 3.16 <DeleteInstructor>

| | |
|---|---|
| **Use case ID** | UC16 |
| **Actors** | Secretariat |
| **Pre conditions** | The Secretariat has logged into the system with username and password. |
| **Main flow of events** | 1.The use case starts when the instructor selects from the menu <<DeleteInstructor>><br>2.The system goes to step 2 of UC14.<br>2.The system asks the secretary to enter the name of the preferred instructor.<br>3.The secretary enters the name of the instructor.<br>4.While the secretary types a non-existing name:<br>4.1 The system asks the secreatariat to enter one name from the above list of<br>all the instructors.<br>4.2 The secretariat types a name.<br>5.The Secretariat types a name from the above list.<br>6.The system deletes the instructor and outputs the updated list of the<br>instructors. |
| **Alternative flow 1** | 1.At any time the instructor may leave the system. |
| **Post conditions** | The Secretariat deleted the instructor from the list of all instructors. |

## 3.17 <UpdateInstructor>

| | |
|---|---|
| **Use case ID** | UC17 |
| **Actors** | Secretariat |
| **Pre conditions** | The Secreatariat has logged into the system with username and password. |
| **Main flow of events** | 1.The use case starts when the secretariat selects from the menu <<UpdateInstructor>><br>2.The system outputs a list with all the instructors and asks the secretariat |

| | |
|---|---|
| | to enter an instructorName which is unique in the list of instructors.<br>3. The secretariat enters an instructorName<br>4.While the secretariat enters a non-unique instructorName:<br>4.1 The system asks the secretariat to enter a unique instructorName<br>4.2 The secretariat enters an instructorName<br>5. The systems asks the secretariat to enter a unique username for the instructor<br>6. The secretariat enters a username for the instructor.<br>7. While the secretariat enters a non-unique username.<br>7.1 The system asks the secretariat to enter a unique username.<br>7.2 The secretariat enters a username for the instructor.<br>8. The system asks the secretariat to enter a password for the instructor.<br>9.The secreatariat enters a password for the instructor.<br>10.While the secretariat enters a non-valid password:<br>10.1 The system asks the secretariat to enter a valid password for the instructor.<br>10.2 The secretariat enters a password for the instructor.<br>11 The system asks the secretariat to enter an email for the instructor<br>12 The secretariat enters an email for the instructor.<br>13.While the secretariat enter a non-unique email:<br>13.1 The system asks the secretariat to enter another email for the instructor<br>13.2 The secretariat enters an email for the instructor.<br>14 The system asks the secretariat to enter a telephone for the instructor<br>15 The secretariat enters a telephone for the instructor.<br>16.While the secretariat enter a non unique telephone:<br>16.1 The system asks the secretariat to enter another telephone for the instructor<br>16.2 The secretariat enters a telephone for the instructor. |
| **Alternative flow 17** | 1.At any time the secreatariat may leave the system. |

| | |
|---|---|
| **Post conditions** | The secretariat updates an instructor  from the list of instructors. |

## 3.18 <ShowAllStudents>

| | |
|---|---|
| **Use case ID** | UC14 |
| **Actors** | Secretariat |
| **Pre conditions** | The Secreatariat has logged into the system with username and password. |
| **Main flow of events** | 1.The use case starts when the secretariat selects from the menu <<ShowAllStudents>><br>2.The system outputs a list with all the students |
| **Alternative flow 1** | 1.At any time the secretariat may leave the system. |
| **Post conditions** | The secretariat gets the list with all the students. |

## 3.19 <AddStudent>

| | |
|---|---|
| **Use case ID** | UC19 |
| **Actors** | Secretariat |
| **Pre conditions** | The Secreatariat has logged into the system with username and password. |
| **Main flow of events** | 1.The use case starts when the secretariat selects from the menu <<AddStudent>><br>2.The system outputs a list with all the students and asks the secretariat to enter a studentName which is  not contained in the list of students. |

| | |
|---|---|
| | 3. The secretariat enters a studentName<br>4.While the secretariat enters a contained studentName:<br>4.1 The system asks the secretariat to enter a unique studentName<br>4.2 The secretariat enters an studentName<br>5. The systems asks the secretariat to enter a unique username for the student<br>6. The secretariat enters a username for the student.<br>7. While the secretariat enters a anon-unique username.<br>7.1 The system asks the secretariat to enter a unique username.<br>7.2 The secretariat enters a username for the student.<br>8. The system asks the secretariat to enter a password for the student.<br>9.The secreatariat enters a password for the student.<br>10.While the secretariat enters a non-valid password:<br>10.1 The system asks the secretariat to enter a valid password for the student.<br>10.2 The secretariat enters a password for the student.<br>11 The system asks the secretariat to enter an email for the student<br>12 The secretariat enters an email for the student.<br>13.While the secretariat enter an existing email:<br>13.1 The system asks the secretariat to enter another email for the student<br>13.2 The secretariat enters an email for the student.<br>14 The system asks the secretariat to enter a telephone for the student<br>15 The secretariat enters a telephone for the student.<br>16.While the secretariat enter an existing telephone:<br>16.1 The system asks the secretariat to enter another telephone for the student<br>16.2 The secretariat enters a telephone for the student.<br>17 The system asks the secretariat to enter a unique am for the student<br>18 The secretariat enters an am for the student.<br>19.While the secretariat enter an existing am:<br>19.1 The system asks the secretariat to enter another am for the student |

| | |
|---|---|
| | 19.2 The secretariat enters an am for the student.<br>20.The system asks the secretariat to enter the year for the student between 1 and 5<br>21 The secretariat enters a year<br>22.While the secretariat enters a non valid year:<br>22.1.The system asks the secretariat to enter a year between 1 and 5<br>22.2 The secretariat enters a year<br>23.The system asks the secretariat to enter the semester of the student.<br>24. The secretariat enters a semester<br>25.While the secretariat enters a non-valid semester:<br>25.1 The system asks the secretariat to enter a valid semester.<br>25.2 The secretariat enters a semester for the student.<br>26.The system asks the secretariat to enter the semester of the student.<br>24. The secretariat enters a semester<br>25.While the secretariat enters a non-valid semester:<br>25.1 The system asks the secretariat to enter a valid semester.<br>25.2 The secretariat enters a semester for the student.<br>26. The system asks the secretariat to enter an avgGrade between 5 and 10 for the student.<br>27. The secretariat enters an avg for the student.<br>28. While the secretariat enter a non valid avgGrade:<br>28.1 The system asks the secretariat to enter another avg.<br>28.2 The secretariat enters another grade.<br>29. The secretariat adds the student. |
| **Alternative flow 16** | 1.At any time the secreatariat may leave the system. |
| **Post conditions** | The secretariat adds a student  into the list of students. |

## 3.20 <DeleteStudent>

| | |
|---|---|
| **Use case ID** | UC20 |
| **Actors** | Secretariat |
| **Pre conditions** | The Secretariat has logged into the system with username and password. |
| **Main flow of events** | 1.The use case starts when the secretariat selects from the menu <<DeleteStudent>><br>2.The system goes to step 2 of UC18.<br>2.The system asks the secretariat to enter the name of the preferred student.<br>3.The secretary enters the name of the student.<br>4.While the secretary types a non-existing name:<br>4.1 The system asks the secreatariat to enter one name from the above list of<br>all the students.<br>4.2 The secretariat types a name.<br>5.The Secretariat types a name from the above list.<br>6.The system deletes the instructor and outputs the updated list of the<br>students. |
| **Alternative flow 1** | 1.At any time the secreatariat may leave the system. |
| **Post conditions** | The Secretariat deleted the student from the list of all students. |

## 3.21 <UpdateStudent>

| | |
|---|---|
| **Use case ID** | UC21 |
| **Actors** | Secretariat |
| **Pre conditions** | The Secretariat has logged into the system with username and password. |
| **Main flow of events** | 1.The use case starts when the secretariat selects from the menu <<UpdateStudent>><br>2.The system outputs a list with all the students and asks the secretariat |

| | |
|---|---|
| | to enter a studentName which is contained in the list of students.<br>3. The secretariat enters a studentName<br>4.While the secretariat enters a contained studentName:<br>4.1 The system asks the secretariat to enter a unique studentName<br>4.2 The secretariat enters a studentName<br>5. The systems asks the secretariat to enter a unique username for the student<br>6. The secretariat enters a username for the student.<br>7. While the secretariat enters a non-unique username.<br>7.1 The system asks the secretariat to enter a unique username.<br>7.2 The secretariat enters a username for the student.<br>8. The system asks the secretariat to enter a password for the student.<br>9.The secreatariat enters a password for the student.<br>10.While the secretariat enters a non-valid password:<br>10.1 The system asks the secretariat to enter a valid password for the student.<br>10.2 The secretariat enters a password for the student.<br>11 The system asks the secretariat to enter an email for the student<br>12 The secretariat enters an email for the student.<br>13.While the secretariat enters a non unique email:<br>13.1 The system asks the secretariat to enter another email for the student<br>13.2 The secretariat enters an email for the student.<br>14 The system asks the secretariat to enter a telephone for the student<br>15 The secretariat enters a telephone for the student.<br>16.While the secretariat enter a non unique telephone:<br>16.1 The system asks the secretariat to enter another telephone for the student<br>16.2 The secretariat enters a telephone for the student.<br>17 The system asks the secretariat to enter a unique am for the student<br>18 The secretariat enters an am for the student.<br>19.While the secretariat enter an existing am: |

| | |
|---|---|
| | 19.1 The system asks the secretariat to enter another am for the student<br>19.2 The secretariat enters an am for the student.<br>20.The system asks the secretariat to enter the year for the student between 1 and 5<br>21 The secretariat enters a year<br>22.While the secretariat enters a non valid year:<br>22.1.The system asks the secretariat to enter a year between 1 and 5<br>22.2 The secretariat enters a year<br>23.The system asks the secretariat to enter the semester of the student.<br>24. The secretariat enters a semester<br>25.While the secretariat enters a non-valid semester:<br>25.1 The system asks the secretariat to enter a valid semester.<br>25.2 The secretariat enters a semester for the student.<br>26.The system asks the secretariat to enter the semester of the student.<br>24. The secretariat enters a semester<br>25.While the secretariat enters a non-valid semester:<br>25.1 The system asks the secretariat to enter a valid semester.<br>25.2 The secretariat enters a semester for the student.<br>26. The system asks the secretariat to enter an avgGrade between 5 and 10 for the student.<br>27. The secretariat enters an avg for the student.<br>28. While the secretariat enters a non valid avgGrade:<br>28.1 The system asks the secretariat to enter another avg.<br>28.2 The secretariat enters another grade.<br>29. The secretariat adds the student. |
| **Alternative flow 16** | 1.At any time the secretariat may leave the system. |
| **Post conditions** | The Secretariat updated the student from the list of all students. |

## 3.22 <ShowAllCoursesPassed>

| | |
|---|---|
| **Use case ID** | UC22 |
| **Actors** | Student |
| **Pre conditions** | The Student has logged into the system with username and password. |
| **Main flow of events** | 1.The use case starts when the secretariat selects from the menu <<ShowAllCoursesPassed>><br>2.The system outputs a list with all the passed courses |
| **Alternative flow 1** | 1.At any time the student may leave the system. |
| **Post conditions** | The student gets the list with all the courses passed. |

## 3.23 <ShowCoursesCurrentSemester>

| | |
|---|---|
| **Use case ID** | UC23 |
| **Actors** | Student |
| **Pre conditions** | The Student has logged into the system with username and password. |
| **Main flow of events** | 1.The use case starts when the secretariat selects from the menu <<ShowCoursesCurrentSemester>><br>2.The system outputs a list with all the courses of the current semester. |
| **Alternative flow 1** | 1.At any time the student may leave the system. |
| **Post conditions** | The student gets the list with all the courses of the current semester. |

## 3.24 <RegisterCoursesCurrentSemester>

| Use case ID | UC24 |
|---|---|
| Actors | Student |
| Pre conditions | The Student has logged into the system with username and password. |
| Main flow of events | 1.The use case starts when the student selects from the menu <<RegisterCoursesCurrentSemester>><br>2.The system outputs a list with all the courses of the current semester and a list with all the courses and asks the student to add the preferred courses<br>3. The student selects the preferred courses.<br>4. The system adds the preferred courses outputs the updated list with the courses of the current semester. |
| Alternative flow 1 | 1.At any time the student may leave the system. |
| Post conditions | The student registers the preferred courses of the current semester. |

## 3.25 <DeleteRegisteredCoursesCurrentSemester>

| Use case ID | UC25 |
|---|---|
| Actors | Student |
| Pre conditions | The Student has logged into the system with username and password. |
| Main flow of events | 1.The use case starts when the student selects from the menu <<DeleteRegisteredCoursesCurrentSemester>><br>2.The system outputs a list with all the courses of the current semester and asks the student to add the preferred courses<br>3. The student selects the preferred courses.<br>4. The system deletes the preferred courses outputs the updated list with the courses of the current semester. |
| Alternative flow 1 | 1.At any time the student may leave the system. |
| Post conditions | The student deletes some of the registered courses of the current semester. |

## 3.26 <CalcualateMyAverageGrade>

| | |
|---|---|
| **Use case ID** | UC26 |
| **Actors** | Student |
| **Pre conditions** | The Student has logged into the system with username and password. |
| **Main flow of events** | 1.The use case starts when the student selects from the menu <<CalculateMyAverageGrade>><br>2.The system calculates the average grade of the student and outputs the Grade |
| **Alternative flow 1** | 1.At any time the student may leave the system. |
| **Post conditions** | The student calculates his average grade. |

## 3.27 <ShowHistoryWIthAllAttempts>

| | |
|---|---|
| **Use case ID** | UC27 |
| **Actors** | Student |
| **Pre conditions** | The Student has logged into the system with username and password. |
| **Main flow of events** | 1.The use case starts when the student selects from the menu <<ShowHistoryWithAllAttempts>><br>2.The system outputs the history with all the attempts of the student |
| **Alternative flow 1** | 1.At any time the student may leave the system. |

| | |
|---|---|
| **Post conditions** | The system shows history with all the attempts of the student. |

## 3.28 <ShowHistoryPerExamPeriod>

| | |
|---|---|
| **Use case ID** | UC28 |
| **Actors** | Student |
| **Pre conditions** | The Student has logged into the system with username and password. |
| **Main flow of events** | 1.The use case starts when the student selects from the menu <<ShowHistoryWithPerExamPeriod>><br>2. The system asks the student to type the exam-period he prefers.<br>3. The student types an exam-period<br>4.While the student types a non valid exam-period:<br>4.1 The system asks the student to type another exam-period<br>4.2 The student types an exam period.<br>5.The system outputs the history with the attempts of the student during the preferred exam-period. |
| **Alternative flow 1** | 1.At any time the student may leave the system. |
| **Post conditions** | The system shows history per exam period of the student. |

## 3.29 <Login>

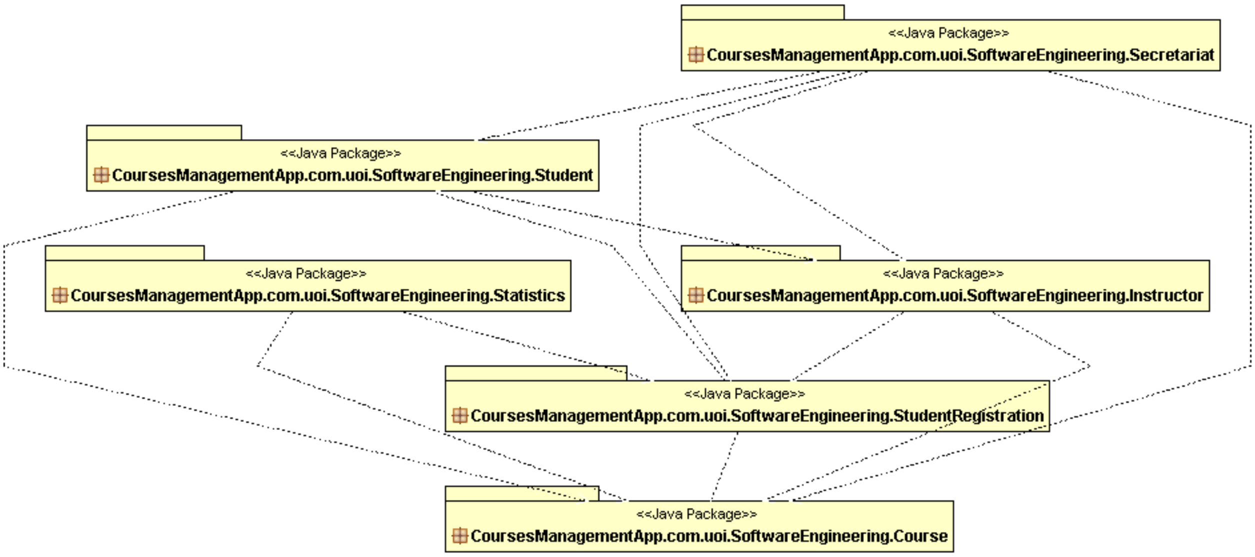
| | |
|---|---|
| **Use case ID** | UC28 |

| **Actors** | Student,Secretariat,Instructor |
|---|---|
| **Pre conditions** | The Student or the secretariat or the instructor has logged into the system with username and password. |
| **Main flow of events** | 1.The use case starts when the student or the secretariat or the instructor enters the system<br>2. The system asks the student/secretariat/instructor to type the username he prefers.<br>3. The student/secretariat/instructor types a username<br>4.While the student/secretariat/instructor types a non existing username:<br>4.1 The system asks the student/secretariat/instructor to type another username<br>4.2 The student types an exam period.<br>5.The system outputs the history with the attempts of the student during the preferred exam-period. |
| **Alternative flow 1** | 1.At any time the student may leave the system. |
| **Post conditions** | The system shows history per exam period of the student. |

# 4 Design

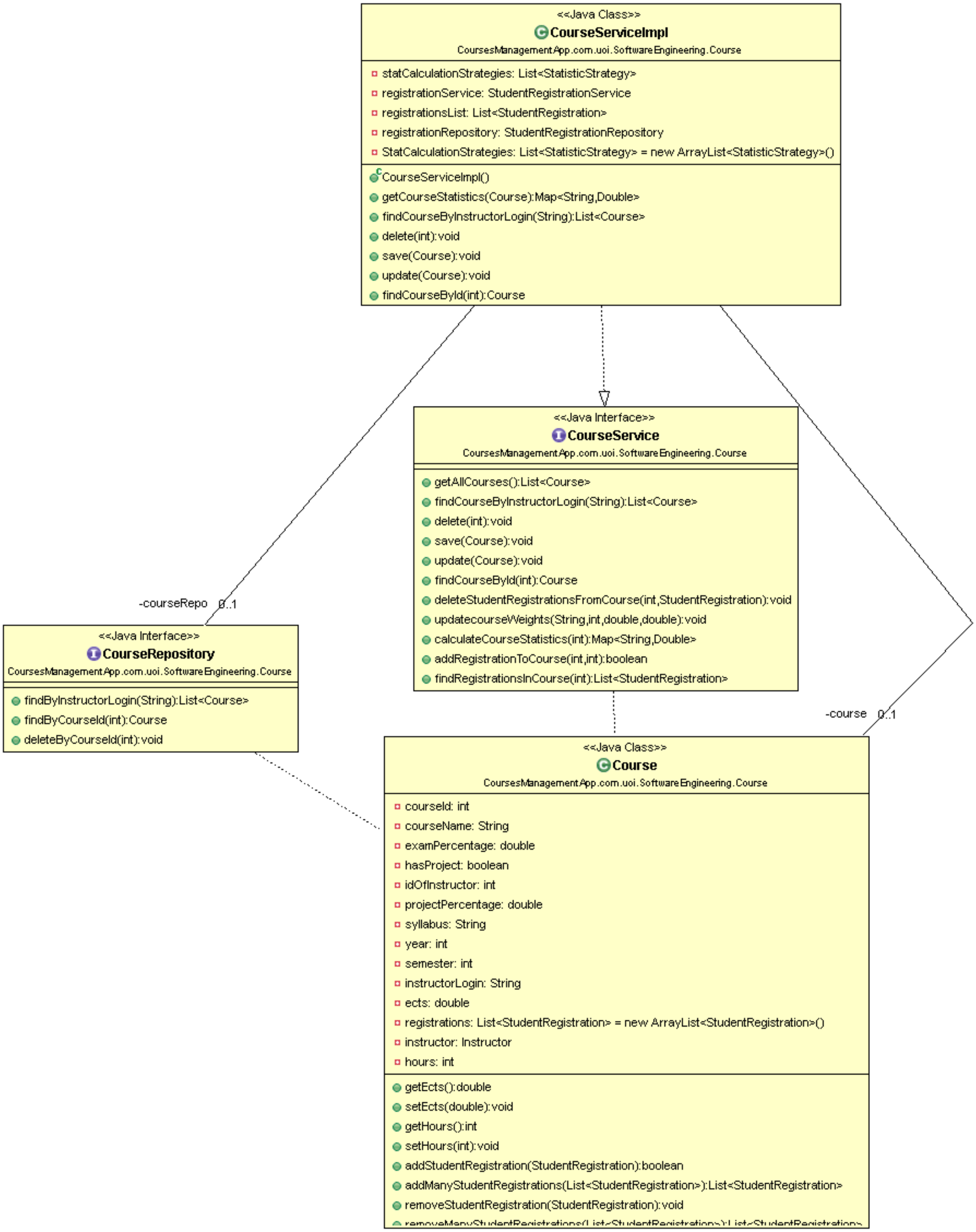
## 4.1 Architecture

<Specify the overall architecture for this release in terms of a **UML package diagram**.>

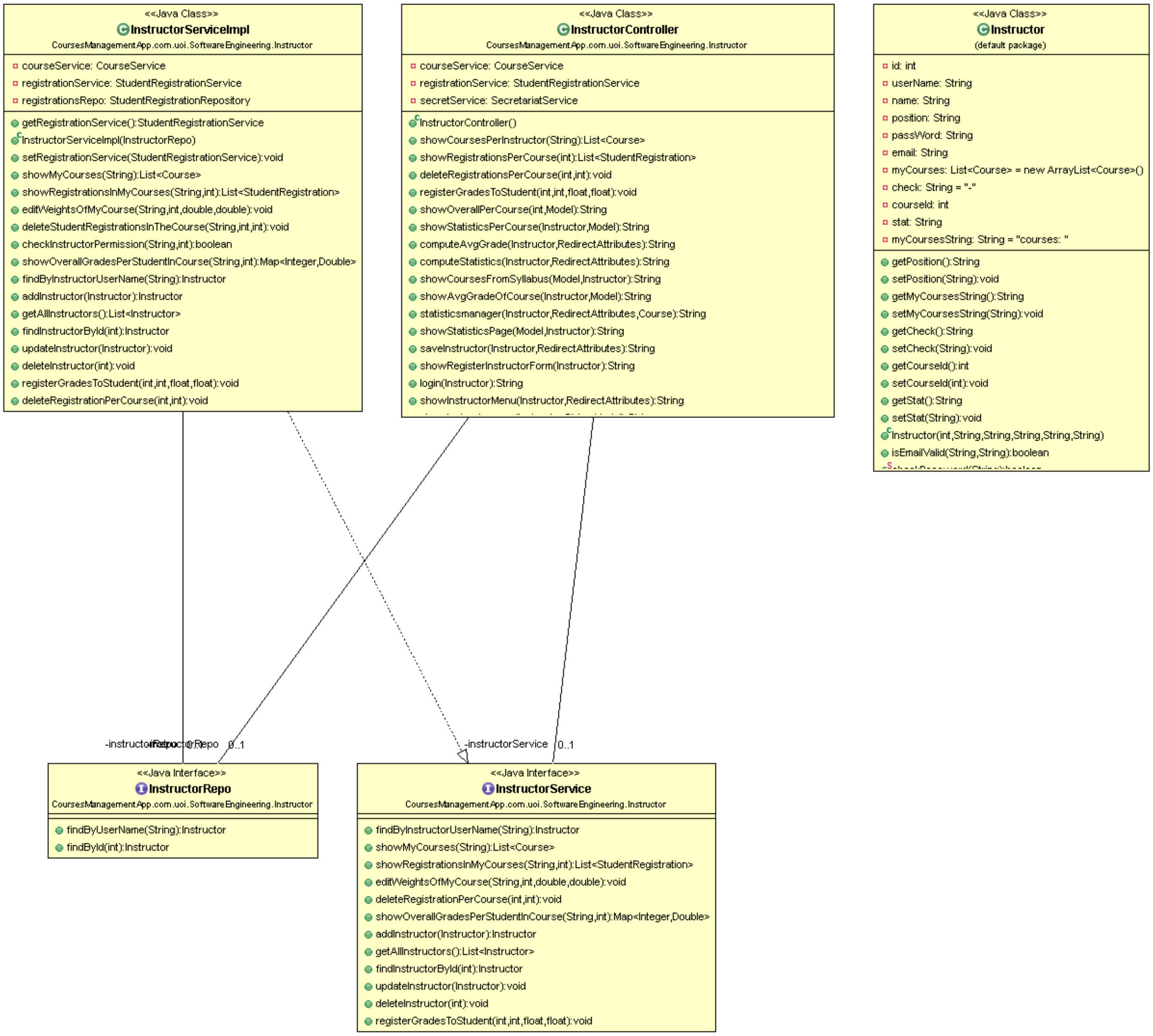

<<Java Package>>
CoursesManagementApp.com.uoi.SoftwareEngineering.Secretariat
<<Java Package>>
CoursesManagementApp.com.uoi.SoftwareEngineering.Student
<<Java Package>>
CoursesManagementApp.com.uoi.SoftwareEngineering.Statistics
<<Java Package>>
CoursesManagementApp.com.uoi.SoftwareEngineering.Instructor
<<Java Package>>
CoursesManagementApp.com.uoi.SoftwareEngineering.StudentRegistration
<<Java Package>>
CoursesManagementApp.com.uoi.SoftwareEngineering.Course

## 4.2 Design

<Specify the detailed design for this release in terms of **UML class diagrams**.>

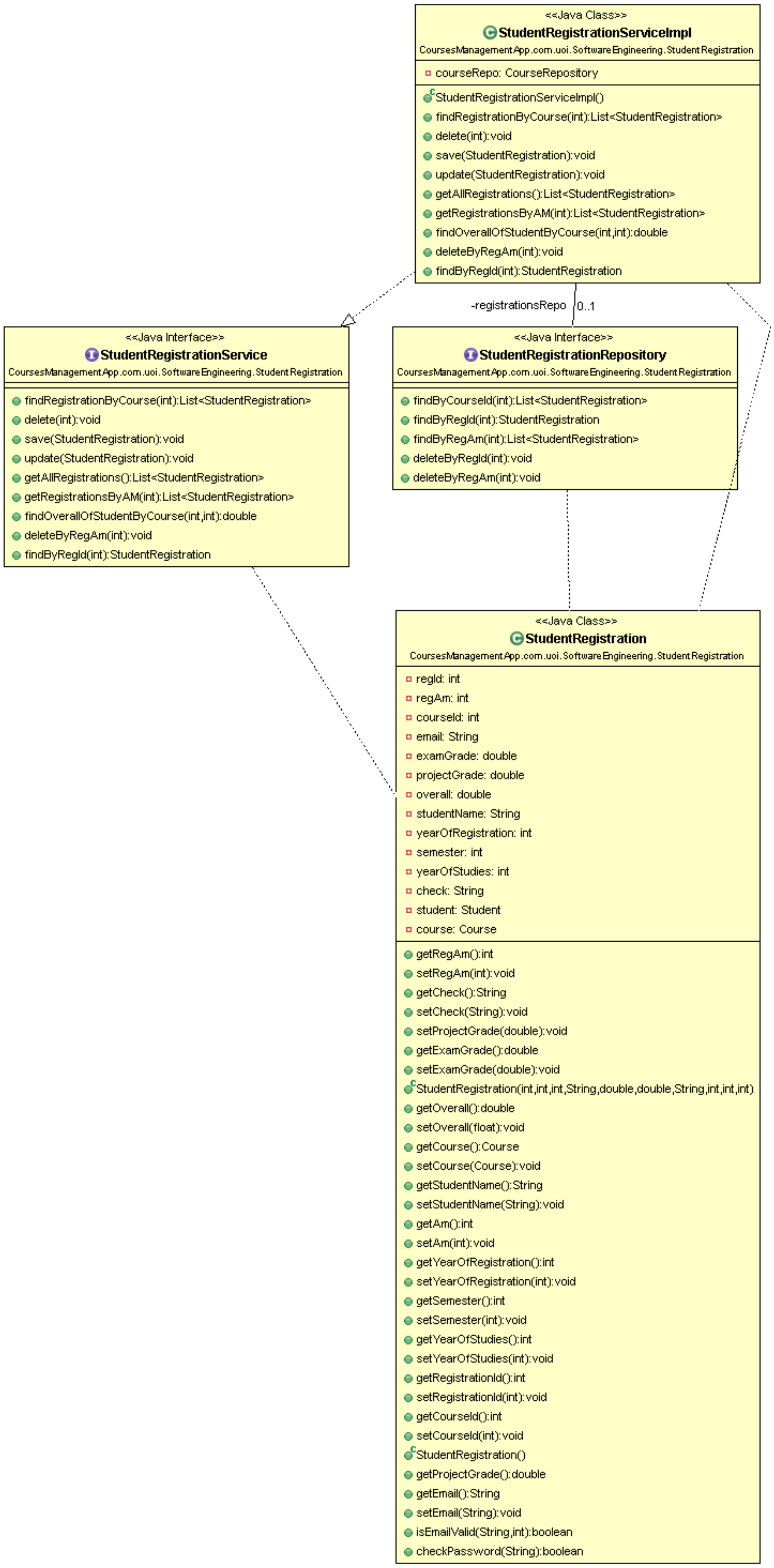


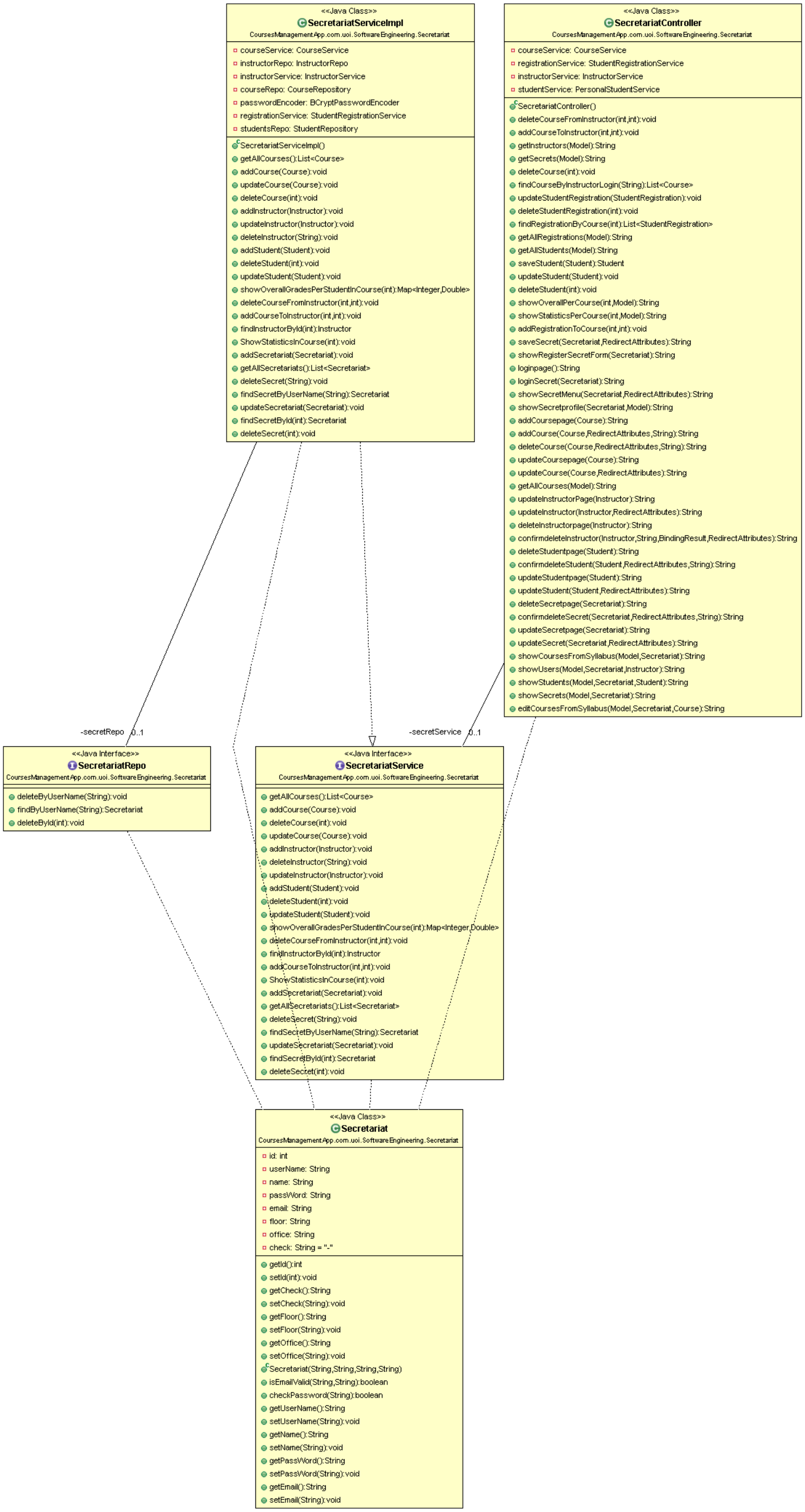
<<Java Class>>
CourseServiceImpl
CoursesManagementApp.com.uoi.SoftwareEngineering.Course
statCalculationStrategies: List<StatisticStrategy>
registrationService: StudentRegistrationService
registrationsList: List<StudentRegistration>
registrationRepository: StudentRegistrationRepository
StatCalculationStrategies: List<StatisticStrategy> = new ArrayList<StatisticStrategy>()
CourseServiceImpl()
getCourseStatistics(Course):Map<String,Double>
findCourseByInstructorLogin(String):List<Course>
delete(int):void
save(Course):void
update(Course):void
findCourseById(int):Course
<<Java Interface>>
CourseService
CoursesManagementApp.com.uoi.SoftwareEngineering.Course
getAllCourses():List<Course>
findCourseByInstructorLogin(String):List<Course>
delete(int):void
save(Course):void
update(Course):void
findCourseById(int):Course
deleteStudentRegistrationsFromCourse(int,StudentRegistration):void
updatecourseWeights(String,int,double,double):void
calculateCourseStatistics(int):Map<String,Double>
addRegistrationToCourse(int,int):boolean
findRegistrationsInCourse(int):List<StudentRegistration>
-courseRepo 0..1
<<Java Interface>>
CourseRepository
CoursesManagementApp.com.uoi.SoftwareEngineering.Course
findByInstructorLogin(String):List<Course>
findByCourseId(int):Course
deleteByCourseId(int):void
-course 0..1
<<Java Class>>
Course
CoursesManagementApp.com.uoi.SoftwareEngineering.Course
courseId: int
courseName: String
examPercentage: double
hasProject: boolean
idOfInstructor: int
projectPercentage: double
syllabus: String
year: int
semester: int
instructorLogin: String
ects: double
registrations: List<StudentRegistration> = new ArrayList<StudentRegistration>()
instructor: Instructor
hours: int
getEcts():double
setEcts(double):void
getHours():int
setHours(int):void
addStudentRegistration(StudentRegistration):boolean
addManyStudentRegistrations(List<StudentRegistration>):List<StudentRegistration>
removeStudentRegistration(StudentRegistration):void
removeManyStudentRegistrations(List<StudentRegistration>):List<StudentRegistration>



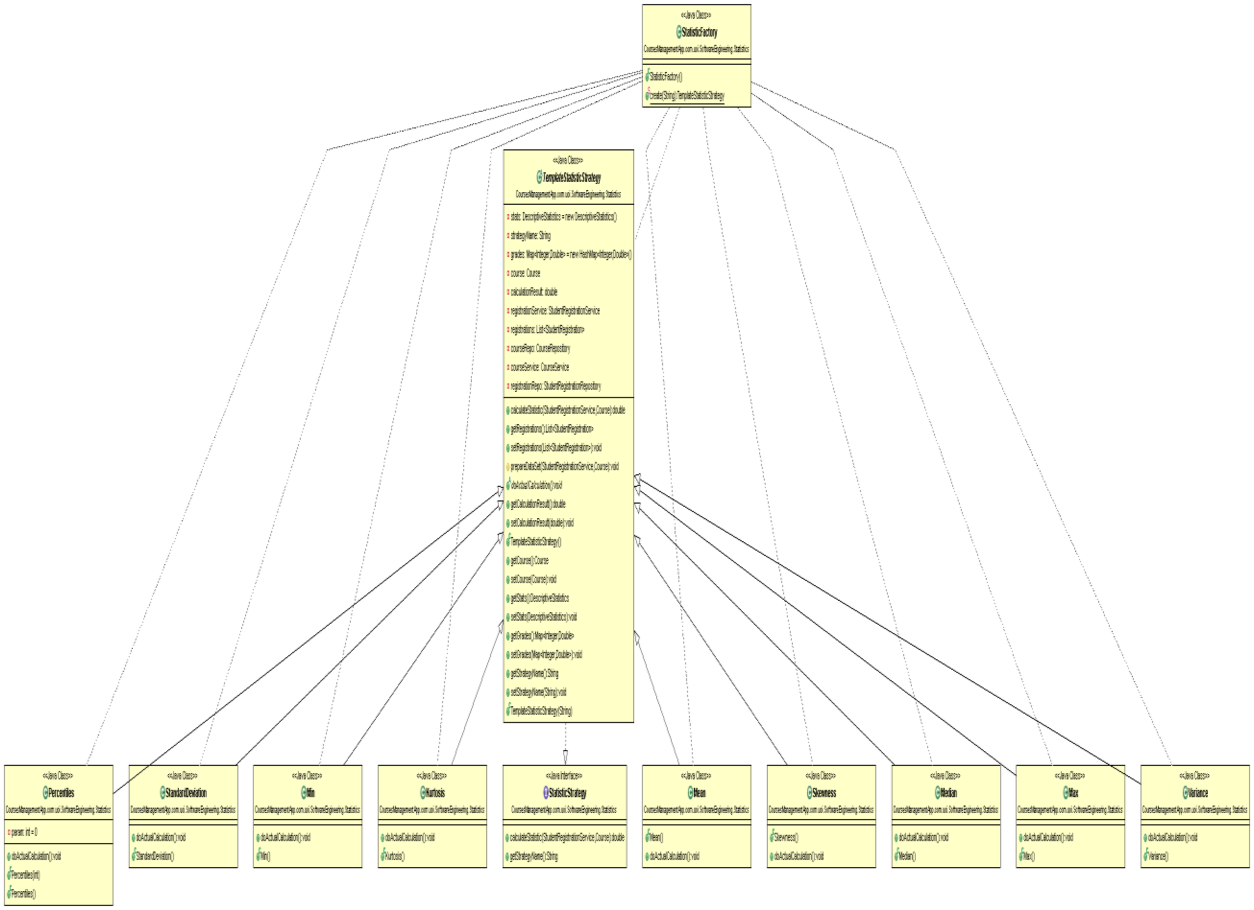
<<Java Class>>
InstructorServiceImpl
CoursesManagementApp.com.uoi.SoftwareEngineering.Instructor
courseService: CourseService
registrationService: StudentRegistrationService
registrationsRepo: StudentRegistrationRepository
getRegistrationService():StudentRegistrationService
InstructorServiceImpl(InstructorRepo)
setRegistrationService(StudentRegistrationService):void
showMyCourses(String):List<Course>
showRegistrationsInMyCourses(String,int):List<StudentRegistration>
editWeightsOfMyCourse(String,int,double,double):void
deleteStudentRegistrationsInTheCourse(String,int,int):void
checkInstructorPermission(String,int):boolean
showOverallGradesPerStudentInCourse(String,int):Map<Integer,Double>
findByInstructorUserName(String):Instructor
addInstructor(Instructor):Instructor
getAllInstructors():List<Instructor>
findInstructorById(int):Instructor
updateInstructor(Instructor):void
deleteInstructor(int):void
registerGradesToStudent(int,int,float,float):void
deleteRegistrationPerCourse(int,int):void
<<Java Class>>
InstructorController
CoursesManagementApp.com.uoi.SoftwareEngineering.Instructor
courseService: CourseService
registrationService: StudentRegistrationService
secretService: SecretariatService
InstructorController()
showCoursesPerInstructor(String):List<Course>
showRegistrationsPerCourse(int):List<StudentRegistration>
deleteRegistrationsPerCourse(int,int):void
registerGradesToStudent(int,int,float,float):void
showOverallPerCourse(int,Model):String
showStatisticsPerCourse(Instructor,Model):String
computeAvgGrade(Instructor,RedirectAttributes):String
computeStatistics(Instructor,RedirectAttributes):String
showCoursesFromSyllabus(Model,Instructor):String
showAvgGradeOfCourse(Instructor,Model):String
statisticsmanager(Instructor,RedirectAttributes,Course):String
showStatisticsPage(Model,Instructor):String
saveInstructor(Instructor,RedirectAttributes):String
showRegisterInstructorForm(Instructor):String
login(Instructor):String
showInstructorMenu(Instructor,RedirectAttributes):String
<<Java Class>>
Instructor
(default package)
id: int
userName: String
name: String
position: String
passWord: String
email: String
myCourses: List<Course> = new ArrayList<Course>()
check: String = "-"
courseId: int
stat: String
myCoursesString: String = "courses: "
getPosition():String
setPosition(String):void
getMyCoursesString():String
setMyCoursesString(String):void
getCheck():String
setCheck(String):void
getCourseId():int
setCourseId(int):void
getStat():String
setStat(String):void
Instructor(int,String,String,String,String,String)
isEmailValid(String,String):boolean
-instructorRepo 0..1
-instructorRepo 0..1
-instructorService 0..1
<<Java Interface>>
InstructorRepo
CoursesManagementApp.com.uoi.SoftwareEngineering.Instructor
findByUserName(String):Instructor
findById(int):Instructor
<<Java Interface>>
InstructorService
CoursesManagementApp.com.uoi.SoftwareEngineering.Instructor
findByInstructorUserName(String):Instructor
showMyCourses(String):List<Course>
showRegistrationsInMyCourses(String,int):List<StudentRegistration>
editWeightsOfMyCourse(String,int,double,double):void
deleteRegistrationPerCourse(int,int):void
showOverallGradesPerStudentInCourse(String,int):Map<Integer,Double>
addInstructor(Instructor):Instructor
getAllInstructors():List<Instructor>
findInstructorById(int):Instructor
updateInstructor(Instructor):void
deleteInstructor(int):void
registerGradesToStudent(int,int,float,float):void

## Instructor

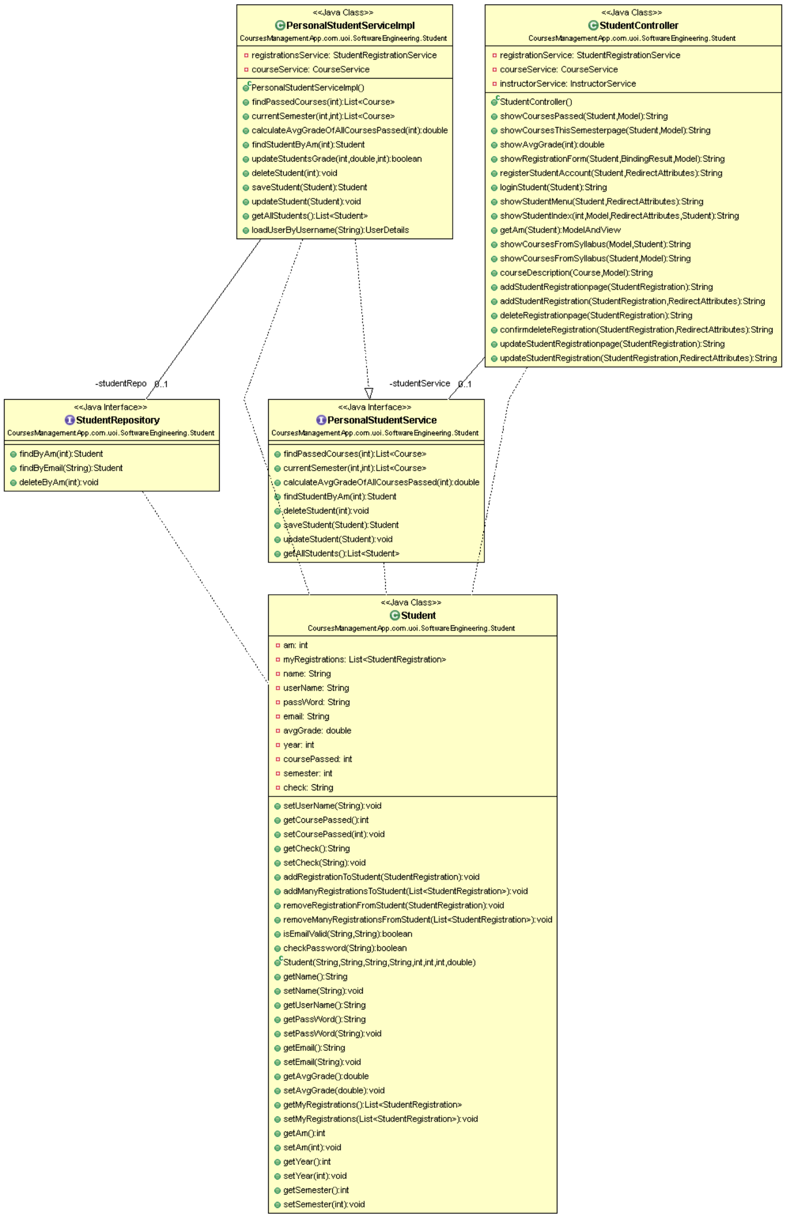

<<Java Class>>
StudentRegistrationServiceImpl
CoursesManagementApp.com.uoi.SoftwareEngineering.StudentRegistration
courseRepo: CourseRepository
StudentRegistrationServiceImpl()
findRegistrationByCourse(int):List<StudentRegistration>
delete(int):void
save(StudentRegistration):void
update(StudentRegistration):void
getAllRegistrations():List<StudentRegistration>
getRegistrationsByAM(int):List<StudentRegistration>
findOverallOfStudentByCourse(int,int):double
deleteByRegAm(int):void
findByRegId(int):StudentRegistration
-registrationsRepo
0..1
<<Java Interface>>
StudentRegistrationService
CoursesManagementApp.com.uoi.SoftwareEngineering.StudentRegistration
findRegistrationByCourse(int):List<StudentRegistration>
delete(int):void
save(StudentRegistration):void
update(StudentRegistration):void
getAllRegistrations():List<StudentRegistration>
getRegistrationsByAM(int):List<StudentRegistration>
findOverallOfStudentByCourse(int,int):double
deleteByRegAm(int):void
findByRegId(int):StudentRegistration
<<Java Interface>>
StudentRegistrationRepository
CoursesManagementApp.com.uoi.SoftwareEngineering.StudentRegistration
findByCourseId(int):List<StudentRegistration>
findByRegId(int):StudentRegistration
findByRegAm(int):List<StudentRegistration>
deleteByRegId(int):void
deleteByRegAm(int):void
<<Java Class>>
StudentRegistration
CoursesManagementApp.com.uoi.SoftwareEngineering.StudentRegistration
regId: int
regAm: int
courseId: int
email: String
examGrade: double
projectGrade: double
overall: double
studentName: String
yearOfRegistration: int
semester: int
yearOfStudies: int
check: String
student: Student
course: Course
getRegAm():int
setRegAm(int):void
getCheck():String
setCheck(String):void
setProjectGrade(double):void
getExamGrade():double
setExamGrade(double):void
StudentRegistration(int,int,int,String,double,double,String,int,int,int)
getOverall():double
setOverall(float):void
getCourse():Course
setCourse(Course):void
getStudentName():String
setStudentName(String):void
getAm():int
setAm(int):void
getYearOfRegistration():int
setYearOfRegistration(int):void
getSemester():int
setSemester(int):void
getYearOfStudies():int
setYearOfStudies(int):void
getRegistrationId():int
setRegistrationId(int):void
getCourseId():int
setCourseId(int):void
StudentRegistration()
getProjectGrade():double
getEmail():String
setEmail(String):void
isEmailValid(String,int):boolean
checkPassword(String):boolean


<<Java Class>>
SecretariatServiceImpl
CoursesManagementApp.com.uoi.SoftwareEngineering.Secretariat
courseService: CourseService
instructorRepo: InstructorRepo
instructorService: InstructorService
courseRepo: CourseRepository
passwordEncoder: BCryptPasswordEncoder
registrationService: StudentRegistrationService
studentsRepo: StudentRepository
SecretariatServiceImpl()
getAllCourses():List<Course>
addCourse(Course):void
updateCourse(Course):void
deleteCourse(int):void
addInstructor(Instructor):void
updateInstructor(Instructor):void
deleteInstructor(String):void
addStudent(Student):void
deleteStudent(int):void
updateStudent(Student):void
showOverallGradesPerStudentInCourse(int):Map<Integer,Double>
deleteCourseFromInstructor(int,int):void
addCourseToInstructor(int,int):void
findInstructorById(int):Instructor
ShowStatisticsInCourse(int):void
addSecretariat(Secretariat):void
getAllSecretariats():List<Secretariat>
deleteSecret(String):void
findSecretByUserName(String):Secretariat
updateSecretariat(Secretariat):void
findSecretById(int):Secretariat
deleteSecret(int):void
<<Java Class>>
SecretariatController
CoursesManagementApp.com.uoi.SoftwareEngineering.Secretariat
courseService: CourseService
registrationService: StudentRegistrationService
instructorService: InstructorService
studentService: PersonalStudentService
SecretariatController()
deleteCourseFromInstructor(int,int):void
addCourseToInstructor(int,int):void
getInstructors(Model):String
getSecrets(Model):String
deleteCourse(int):void
findCourseByInstructorLogin(String):List<Course>
updateStudentRegistration(StudentRegistration):void
deleteStudentRegistration(int):void
findRegistrationByCourse(int):List<StudentRegistration>
getAllRegistrations(Model):String
getAllStudents(Model):String
saveStudent(Student):Student
updateStudent(Student):void
deleteStudent(int):void
showOverallPerCourse(int,Model):String
showStatisticsPerCourse(int,Model):String
addRegistrationToCourse(int,int):void
saveSecret(Secretariat,RedirectAttributes):String
showRegisterSecretForm(Secretariat):String
loginpage():String
loginSecret(Secretariat):String
showSecretMenu(Secretariat,RedirectAttributes):String
showSecretprofile(Secretariat,Model):String
addCoursepage(Course):String
addCourse(Course,RedirectAttributes,String):String
deleteCourse(Course,RedirectAttributes,String):String
updateCoursepage(Course):String
updateCourse(Course,RedirectAttributes):String
getAllCourses(Model):String
updateInstructorPage(Instructor):String
updateInstructor(Instructor,RedirectAttributes):String
deleteInstructorpage(Instructor):String
confirmdeleteInstructor(Instructor,String,BindingResult,RedirectAttributes):String
deleteStudentpage(Student):String
confirmdeleteStudent(Student,RedirectAttributes,String):String
updateStudentpage(Student):String
updateStudent(Student,RedirectAttributes):String
deleteSecretpage(Secretariat):String
confirmdeleteSecret(Secretariat,RedirectAttributes,String):String
updateSecretpage(Secretariat):String
updateSecret(Secretariat,RedirectAttributes):String
showCoursesFromSyllabus(Model,Secretariat):String
showUsers(Model,Secretariat,Instructor):String
showStudents(Model,Secretariat,Student):String
showSecrets(Model,Secretariat):String
editCoursesFromSyllabus(Model,Secretariat,Course):String
-secretRepo
0..1
-secretService
0..1
<<Java Interface>>
SecretariatRepo
CoursesManagementApp.com.uoi.SoftwareEngineering.Secretariat
deleteByUserName(String):void
findByUserName(String):Secretariat
deleteById(int):void
<<Java Interface>>
SecretariatService
CoursesManagementApp.com.uoi.SoftwareEngineering.Secretariat
getAllCourses():List<Course>
addCourse(Course):void
deleteCourse(int):void
updateCourse(Course):void
addInstructor(Instructor):void
deleteInstructor(String):void
updateInstructor(Instructor):void
addStudent(Student):void
deleteStudent(int):void
updateStudent(Student):void
showOverallGradesPerStudentInCourse(int):Map<Integer,Double>
deleteCourseFromInstructor(int,int):void
findInstructorById(int):Instructor
addCourseToInstructor(int,int):void
ShowStatisticsInCourse(int):void
addSecretariat(Secretariat):void
getAllSecretariats():List<Secretariat>
deleteSecret(String):void
findSecretByUserName(String):Secretariat
updateSecretariat(Secretariat):void
findSecretById(int):Secretariat
deleteSecret(int):void
<<Java Class>>
Secretariat
CoursesManagementApp.com.uoi.SoftwareEngineering.Secretariat
id: int
userName: String
name: String
passWord: String
email: String
floor: String
office: String
check: String = "-"
getId():int
setId(int):void
getCheck():String
setCheck(String):void
getFloor():String
setFloor(String):void
getOffice():String
setOffice(String):void
Secretariat(String,String,String,String)
isEmailValid(String,String):boolean
checkPassword(String):boolean
getUserName():String
setUserName(String):void
getName():String
setName(String):void
getPassWord():String
setPassWord(String):void
getEmail():String
setEmail(String):void

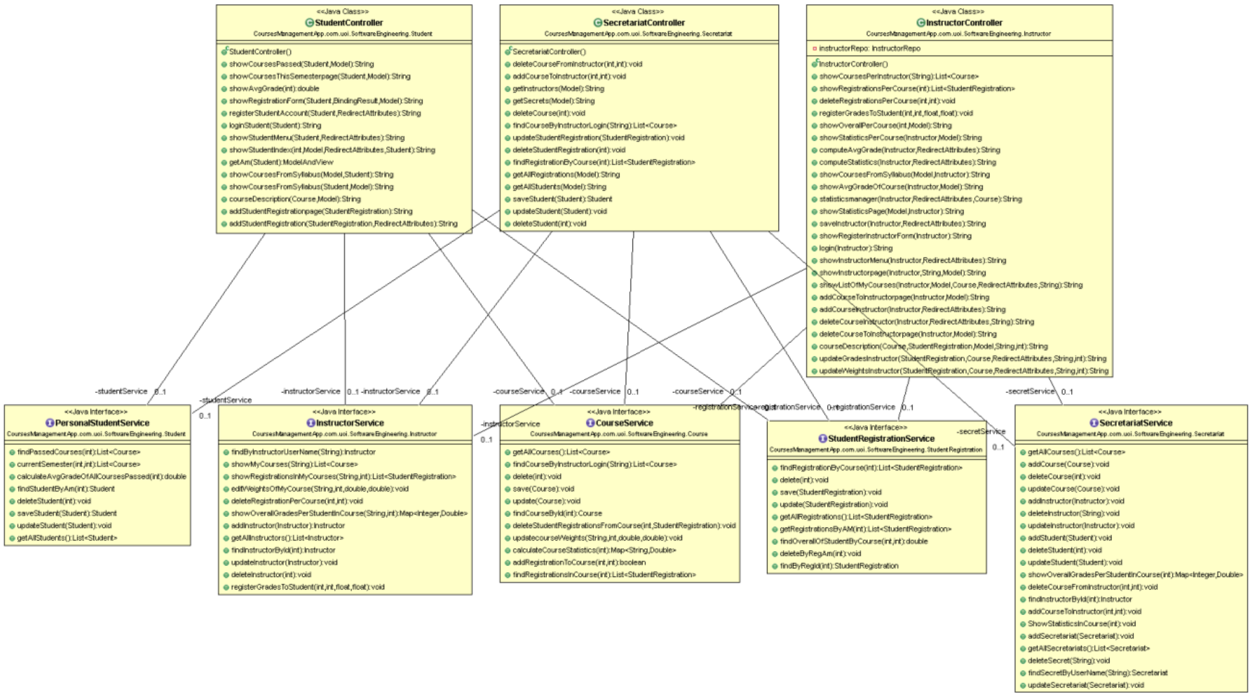
## Stats



<<Java Class>>
StatisticFactory
CoursesManagementApp.com.uoi.SoftwareEngineering.Statistics
StatisticFactory()
create(String):TemplateStatisticStrategy
<<Java Class>>
TemplateStatisticStrategy
CoursesManagementApp.com.uoi.SoftwareEngineering.Statistics
stats: DescriptiveStatistics = new DescriptiveStatistics()
strategyName: String
grades: Map<Integer,Double> = new HashMap<Integer,Double>()
course: Course
calculationResult: double
registrationService: StudentRegistrationService
registrations: List<StudentRegistration>
courseRepo: CourseRepository
courseService: CourseService
registrationRepo: StudentRegistrationRepository
calculateStatistic(StudentRegistrationService,Course):double
getRegistrations():List<StudentRegistration>
setRegistrations(List<StudentRegistration>):void
prepareDataSet(StudentRegistrationService,Course):void
doActualCalculation():void
getCalculationResult():double
setCalculationResult(double):void
TemplateStatisticStrategy()
getCourse():Course
setCourse(Course):void
getStats():DescriptiveStatistics
setStats(DescriptiveStatistics):void
getGrades():Map<Integer,Double>
setGrades(Map<Integer,Double>):void
getStrategyName():String
setStrategyName(String):void
TemplateStatisticStrategy(String)
<<Java Class>>
Percentiles
CoursesManagementApp.com.uoi.SoftwareEngineering.Statistics
param: int = 0
doActualCalculation():void
Percentiles(int)
Percentiles()
<<Java Class>>
StandardDeviation
CoursesManagementApp.com.uoi.SoftwareEngineering.Statistics
doActualCalculation():void
StandardDeviation()
<<Java Class>>
Min
CoursesManagementApp.com.uoi.SoftwareEngineering.Statistics
doActualCalculation():void
Min()
<<Java Class>>
Kurtosis
CoursesManagementApp.com.uoi.SoftwareEngineering.Statistics
doActualCalculation():void
Kurtosis()
<<Java Interface>>
StatisticStrategy
CoursesManagementApp.com.uoi.SoftwareEngineering.Statistics
calculateStatistic(StudentRegistrationService,Course):double
getStrategyName():String
<<Java Class>>
Mean
CoursesManagementApp.com.uoi.SoftwareEngineering.Statistics
Mean()
doActualCalculation():void
<<Java Class>>
Skewness
CoursesManagementApp.com.uoi.SoftwareEngineering.Statistics
Skewness()
doActualCalculation():void
<<Java Class>>
Median
CoursesManagementApp.com.uoi.SoftwareEngineering.Statistics
doActualCalculation():void
Median()
<<Java Class>>
Max
CoursesManagementApp.com.uoi.SoftwareEngineering.Statistics
doActualCalculation():void
Max()
<<Java Class>>
Variance
CoursesManagementApp.com.uoi.SoftwareEngineering.Statistics
doActualCalculation():void
Variance()


<<Java Class>>
PersonalStudentServiceImpl
CoursesManagementApp.com.uoi.SoftwareEngineering.Student
registrationsService: StudentRegistrationService
courseService: CourseService
PersonalStudentServiceImpl()
findPassedCourses(int):List<Course>
currentSemester(int,int):List<Course>
calculateAvgGradeOfAllCoursesPassed(int):double
findStudentByAm(int):Student
updateStudentsGrade(int,double,int):boolean
deleteStudent(int):void
saveStudent(Student):Student
updateStudent(Student):void
getAllStudents():List<Student>
loadUserByUsername(String):UserDetails
<<Java Class>>
StudentController
CoursesManagementApp.com.uoi.SoftwareEngineering.Student
registrationService: StudentRegistrationService
courseService: CourseService
instructorService: InstructorService
StudentController()
showCoursesPassed(Student,Model):String
showCoursesThisSemesterpage(Student,Model):String
showAvgGrade(int):double
showRegistrationForm(Student,BindingResult,Model):String
registerStudentAccount(Student,RedirectAttributes):String
loginStudent(Student):String
showStudentMenu(Student,RedirectAttributes):String
showStudentIndex(int,Model,RedirectAttributes,Student):String
getAm(Student):ModelAndView
showCoursesFromSyllabus(Model,Student):String
showCoursesFromSyllabus(Student,Model):String
courseDescription(Course,Model):String
addStudentRegistrationpage(StudentRegistration):String
addStudentRegistration(StudentRegistration,RedirectAttributes):String
deleteRegistrationpage(StudentRegistration):String
confirmdeleteRegistration(StudentRegistration,RedirectAttributes):String
updateStudentRegistrationpage(StudentRegistration):String
updateStudentRegistration(StudentRegistration,RedirectAttributes):String
-studentRepo
0..1
-studentService
0..1
<<Java Interface>>
StudentRepository
CoursesManagementApp.com.uoi.SoftwareEngineering.Student
findByAm(int):Student
findByEmail(String):Student
deleteByAm(int):void
<<Java Interface>>
PersonalStudentService
CoursesManagementApp.com.uoi.SoftwareEngineering.Student
findPassedCourses(int):List<Course>
currentSemester(int,int):List<Course>
calculateAvgGradeOfAllCoursesPassed(int):double
findStudentByAm(int):Student
deleteStudent(int):void
saveStudent(Student):Student
updateStudent(Student):void
getAllStudents():List<Student>
<<Java Class>>
Student
CoursesManagementApp.com.uoi.SoftwareEngineering.Student
am: int
myRegistrations: List<StudentRegistration>
name: String
userName: String
passWord: String
email: String
avgGrade: double
year: int
coursePassed: int
semester: int
check: String
setUserName(String):void
getCoursePassed():int
setCoursePassed(int):void
getCheck():String
setCheck(String):void
addRegistrationToStudent(StudentRegistration):void
addManyRegistrationsToStudent(List<StudentRegistration>):void
removeRegistrationFromStudent(StudentRegistration):void
removeManyRegistrationsFromStudent(List<StudentRegistration>):void
isEmailValid(String,String):boolean
checkPassword(String):boolean
Student(String,String,String,String,int,int,int,double)
getName():String
setName(String):void
getUserName():String
getPassWord():String
setPassWord(String):void
getEmail():String
setEmail(String):void
getAvgGrade():double
setAvgGrade(double):void
getMyRegistrations():List<StudentRegistration>
setMyRegistrations(List<StudentRegistration>):void
getAm():int
setAm(int):void
getYear():int
setYear(int):void
getSemester():int
setSemester(int):void

**Student**


<<Java Class>>
StudentController
CoursesManagementApp.com.uoi.SoftwareEngineering.Student
StudentController()
showCoursesPassed(Student,Model):String
showCoursesThisSemesterpage(Student,Model):String
showAvgGrade(int):double
showRegistrationForm(Student,BindingResult,Model):String
registerStudentAccount(Student,RedirectAttributes):String
loginStudent(Student):String
showStudentMenu(Student,RedirectAttributes):String
showStudentIndex(int,Model,RedirectAttributes,Student):String
getAm(Student):ModelAndView
showCoursesFromSyllabus(Model,Student):String
showCoursesFromSyllabus(Student,Model):String
courseDescription(Course,Model):String
addStudentRegistrationpage(StudentRegistration):String
addStudentRegistration(StudentRegistration,RedirectAttributes):String
<<Java Class>>
SecretariatController
CoursesManagementApp.com.uoi.SoftwareEngineering.Secretariat
SecretariatController()
deleteCourseFromInstructor(int,int):void
addCourseToInstructor(int,int):void
getInstructors(Model):String
getSecrets(Model):String
deleteCourse(int):void
findCourseByInstructorLogin(String):List<Course>
updateStudentRegistration(StudentRegistration):void
deleteStudentRegistration(int):void
findRegistrationByCourse(int):List<StudentRegistration>
getAllRegistrations(Model):String
getAllStudents(Model):String
saveStudent(Student):Student
updateStudent(Student):void
deleteStudent(int):void
<<Java Class>>
InstructorController
CoursesManagementApp.com.uoi.SoftwareEngineering.Instructor
instructorRepo: InstructorRepo
InstructorController()
showCoursesPerInstructor(String):List<Course>
showRegistrationsPerCourse(int):List<StudentRegistration>
deleteRegistrationsPerCourse(int,int):void
registerGradesToStudent(int,int,float,float):void
showOverallPerCourse(int,Model):String
showStatisticsPerCourse(Instructor,Model):String
computeAvgGrade(Instructor,RedirectAttributes):String
computeStatistics(Instructor,RedirectAttributes):String
showCoursesFromSyllabus(Model,Instructor):String
showAvgGradeOfCourse(Instructor,Model):String
statisticsmanager(Instructor,RedirectAttributes,Course):String
showStatisticsPage(Model,Instructor):String
saveInstructor(Instructor,RedirectAttributes):String
showRegisterInstructorForm(Instructor):String
login(Instructor):String
showInstructorMenu(Instructor,RedirectAttributes):String
showInstructorpage(Instructor,String,Model):String
showListOfMyCourses(Instructor,Model,Course,RedirectAttributes,String):String
addCourseToInstructorpage(Instructor,Model):String
addCourseInstructor(Instructor,RedirectAttributes):String
deleteCourseInstructor(Instructor,RedirectAttributes,String):String
deleteCourseToInstructorpage(Instructor,Model):String
courseDescription(Course,StudentRegistration,Model,String,int):String
updateGradesInstructor(StudentRegistration,Course,RedirectAttributes,String,int):String
updateWeightsInstructor(StudentRegistration,Course,RedirectAttributes,String,int):String
-studentService 0..1
-studentService 0..1
-instructorService 0..1
-instructorService 0..1
-instructorService 0..1
-courseService 0..1
-courseService 0..1
-courseService 0..1
-registrationService
-registrationService
0..1 registrationService 0..1
-secretService 0..1
-secretService 0..1
<<Java Interface>>
PersonalStudentService
CoursesManagementApp.com.uoi.SoftwareEngineering.Student
findPassedCourses(int):List<Course>
currentSemester(int,int):List<Course>
calculateAvgGradeOfAllCoursesPassed(int):double
findStudentByAm(int):Student
deleteStudent(int):void
saveStudent(Student):Student
updateStudent(Student):void
getAllStudents():List<Student>
<<Java Interface>>
InstructorService
CoursesManagementApp.com.uoi.SoftwareEngineering.Instructor
findByInstructorUserName(String):Instructor
showMyCourses(String):List<Course>
showRegistrationsInMyCourses(String,int):List<StudentRegistration>
editWeightsOfMyCourse(String,int,double,double):void
deleteRegistrationPerCourse(int,int):void
showOverallGradesPerStudentInCourse(String,int):Map<Integer,Double>
addInstructor(Instructor):Instructor
getAllInstructors():List<Instructor>
findInstructorById(int):Instructor
updateInstructor(Instructor):void
deleteInstructor(int):void
registerGradesToStudent(int,int,float,float):void
<<Java Interface>>
CourseService
CoursesManagementApp.com.uoi.SoftwareEngineering.Course
getAllCourses():List<Course>
findCourseByInstructorLogin(String):List<Course>
delete(int):void
save(Course):void
update(Course):void
findCourseById(int):Course
deleteStudentRegistrationsFromCourse(int,StudentRegistration):void
updatecourseWeights(String,int,double,double):void
calculateCourseStatistics(int):Map<String,Double>
addRegistrationToCourse(int,int):boolean
findRegistrationsInCourse(int):List<StudentRegistration>
<<Java Interface>>
StudentRegistrationService
CoursesManagementApp.com.uoi.SoftwareEngineering.StudentRegistration
findRegistrationByCourse(int):List<StudentRegistration>
delete(int):void
save(StudentRegistration):void
update(StudentRegistration):void
getAllRegistrations():List<StudentRegistration>
getRegistrationsByAM(int):List<StudentRegistration>
findOverallOfStudentByCourse(int,int):double
deleteByRegAm(int):void
findByRegId(int):StudentRegistration
<<Java Interface>>
SecretariatService
CoursesManagementApp.com.uoi.SoftwareEngineering.Secretariat
getAllCourses():List<Course>
addCourse(Course):void
deleteCourse(int):void
updateCourse(Course):void
addInstructor(Instructor):void
deleteInstructor(String):void
updateInstructor(Instructor):void
addStudent(Student):void
deleteStudent(int):void
updateStudent(Student):void
showOverallGradesPerStudentInCourse(int):Map<Integer,Double>
deleteCourseFromInstructor(int,int):void
findInstructorById(int):Instructor
addCourseToInstructor(int,int):void
ShowStatisticsInCourse(int):void
addSecretariat(Secretariat):void
getAllSecretariats():List<Secretariat>
deleteSecret(String):void
findSecretByUserName(String):Secretariat
updateSecretariat(Secretariat):void

**Controllers**

Η διαρκεια του βίντεο ξεπερνάει την προβλεπόμενη λόγω των τριών διαφορετικών χρηστών που υποστηρίζει και υις φορμες για register login