Diploma Projects Management App

Sprint Report

Bytewave
George Didaskalou, 3217
Marios Mpellos, 3288
Miltos Euthimiou, 4055

VERSIONS HISTORY

Date	Version	Description	Author
30/04/2023	1	Initialize use cases	All
14/05/2023	2	Finalize use cases and add design	All

1 Introduction

This document provides information concerning the **4** sprint of the project.

1.1 Purpose

1.2 Document Structure

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

2 Scrum team and Sprint Backlog

2.1 Scrum team

Product Owner	Goerge Didaskalou
Scrum Master	George Didaskalou
Development Team	George Didaskalou Marios Mpellos Miltos Euthimiou

2.2 Sprints

Sprint No	Begin Date	End Date	Number of weeks	User stories
1	**2/04/2023**	**15/04/2023**	**2**	**U1, U2, S1, P1**
2	**16/04/2023**	**29/04/2023**	**2**	**S2, S3, S4, P2, P3, P4**
3	**30/04/2023**	**13/05/2023**	**2**	**P5, P6, P7**
4	**14/05/2023**	**18/05/2023**	**1**	**P8, P9**

3 Use Cases

3.1 <Use Case 1>

Use case ID	UC1
Actors	The user
Pre conditions	-
Main flow of events	1. The use case starts when the user wants to sign up to the system 2. The user writes the username, password and the role he want
Alternative flow 1	If user select a username that already exists, the system inform the user to select other username
Post conditions	The system completes the sign up with the given information

3.2 <Use Case 2>

Use case ID	UC2
Actors	The user
Pre conditions	-
Main flow of events	1. The use case starts when the user wants to login to the system 2. The user writes the username and his password
Alternative flow 1	If user gives wrong credentials, the system inform the user
Post conditions	The system connects the user

3.3 <Use Case 3>

Use case ID	UC3
Actors	The student
Pre conditions	The student have to sign in to the system
Main flow of events	1. The use case starts when the student select to update his personal information 2. The system shows a form to be completed by the student 3. The student select to update his personal information when he has finished filling out the form
Post conditions	The system updates the personal information of the student

3.4 <Use Case 4>

Use case ID	UC4
Actors	The student
Pre conditions	The student have to sign in to the system and must have completed his personal information
Main flow of events	1. The use case starts when the student wants to see the available diploma thesis subjects 2. The system shows all the available diploma thesis subjects in a list
Post conditions	-

3.5 <Use Case 5>

Use case ID	UC5
Actors	The student
Pre conditions	The student have to sign in to the system and must have completed his personal information
Main flow of events	1. The use case starts when the student wants to see more detailed description of a diploma thesis subject 2. The system shows the details of the selected diploma thesis subject
Post conditions	-

3.6 <Use Case 6>

Use case ID	UC6
Actors	The student
Pre conditions	The student have to sign in to the system and must have completed his personal information
Main flow of events	1. The use case starts when the student wants to apply for a specific diploma thesis subject 2. The system shows a confirmation window
Post conditions	The system creates an application for this diploma thesis subject

3.7 <Use Case 7>

Use case ID	UC7
Actors	The professor
Pre conditions	The professor have to sign in to the system
Main flow of events	1. The use case starts when the professor select to update his personal information 2. The system shows a form to be completed by the professor 3. The professor select to update his personal information when he has finished filling out the form
Post conditions	The system updates the personal information of the professor

3.8 <Use Case 8>

Use case ID	UC8
Actors	The professor
Pre conditions	The professor have to sign in to the system and must have completed his personal information
Main flow of events	1. The use case starts when the professor select to see the available diploma thesis subject that he offer 2. The system shows all diploma thesis subject that he offer in a list
Post conditions	-

3.9 <Use Case 9>

Use case ID	UC9
Actors	The professor
Pre conditions	The professor have to sign in to the system and must have completed his personal information
Main flow of events	1. The use case starts when the professor want to add a new diploma thesis subject 2. The system shows a form for the subject to be completed by the professor 3. The professor select to add the subject when he has finished filling out the form
Post conditions	The system adds the subject

3.10 <Use Case 10>

Use case ID	UC10
Actors	The professor
Pre conditions	The professor have to sign in to the system and must have completed his personal information
Main flow of events	1. The use case starts when the professor want to delete a diploma thesis subject 2. The professor selects the diploma thesis subject that he want to delete
Post conditions	The system deletes the subject

3.11 <Use Case 11>

Use case ID	UC11
Actors	The professor
Pre conditions	The professor have to sign in to the system and must have completed his personal information
Main flow of events	1. The use case starts when the professor want to see the list of applications from the students who want to take over a diploma thesis subject 2. The system show all the application for the available diploma thesis subjects
Post conditions	-

3.12 <Use Case 12>

Use case ID	UC12
Actors	The professor
Pre conditions	The professor have to sign in to the system and must have completed his personal information
Main flow of events	1. The use case starts when the professor want to assign a diploma thesis subject to one of the students who applied 2. The professor selects the strategy for the assignment from the below: 2.1 Random choice 2.2 Student with the best average courses grade 2.3 Student with the fewest remaining courses for graduation 2.4 Student with average courses grade greater than a given threshold Th1 and number of remaining courses for graduation less than a given threshold Th2 3. If professor select the fourth strategy 3.1 The professor fulfill the Th1 and Th2 threshold
Post conditions	The system assign the diploma thesis subject to the selected student

3.13 <Use Case 13>

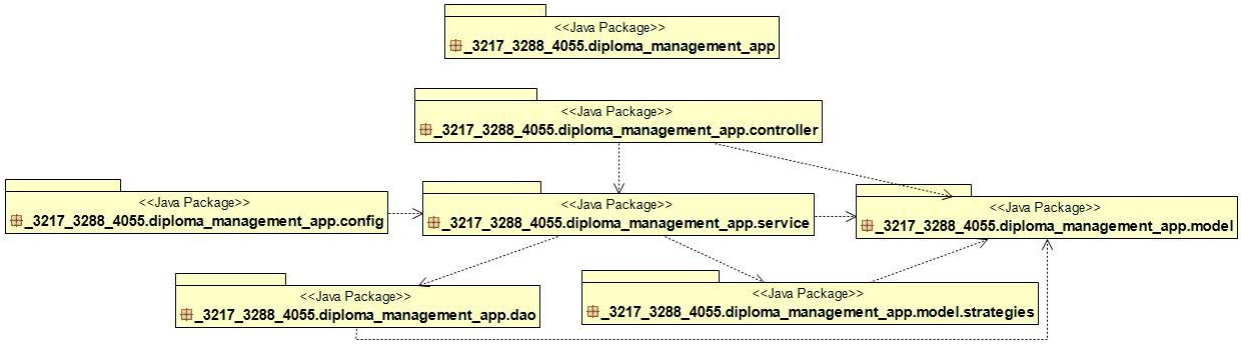
Use case ID	UC13
Actors	The professor
Pre conditions	The professor have to sign in to the system and must have completed his personal information
Main flow of events	1. The use case starts when the professor want to see the list of assigned diploma thesis projects that he supervise 2. The system show the assigned diploma thesis projects
Post conditions	-

3.14 <Use Case 14>

Use case ID	UC14
Actors	The professor
Pre conditions	The professor have to sign in to the system and must have completed his personal information
Main flow of events	1. The use case starts when the professor want to set the grades for an assigned diploma thesis subject 2. The professor select the diploma thesis subject he want to add the grades 3. The system shows a form 4. The professor completes the implementation, the report and the presentation grade
Post conditions	The system adds the given grades to the selected diploma thesis subject and calculate the overall grade

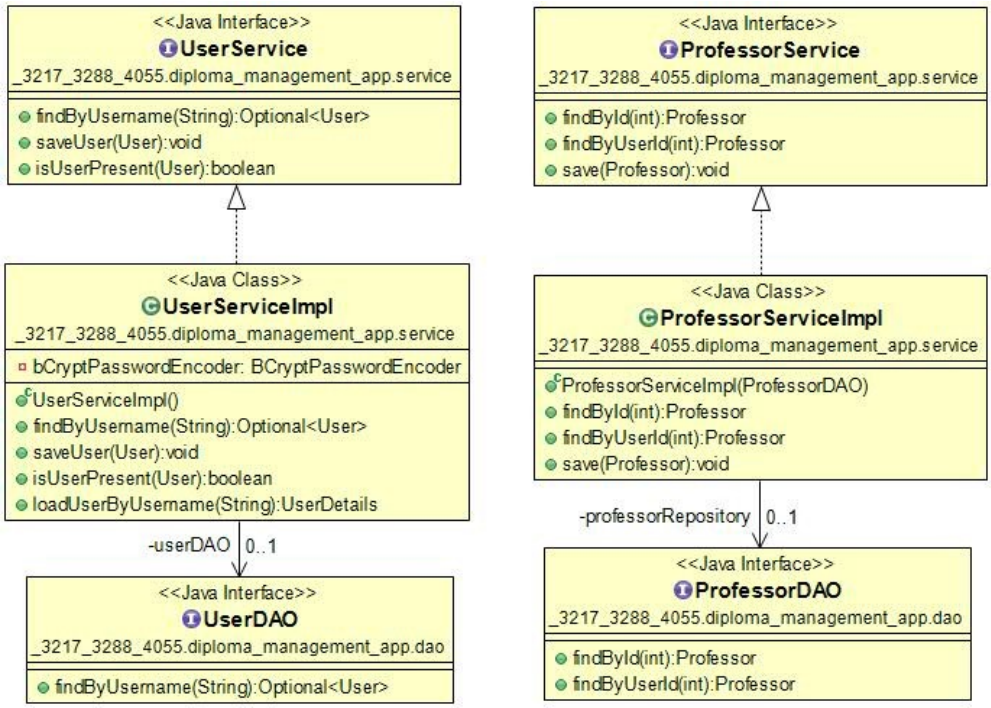
4 Design

4.1 Architecture

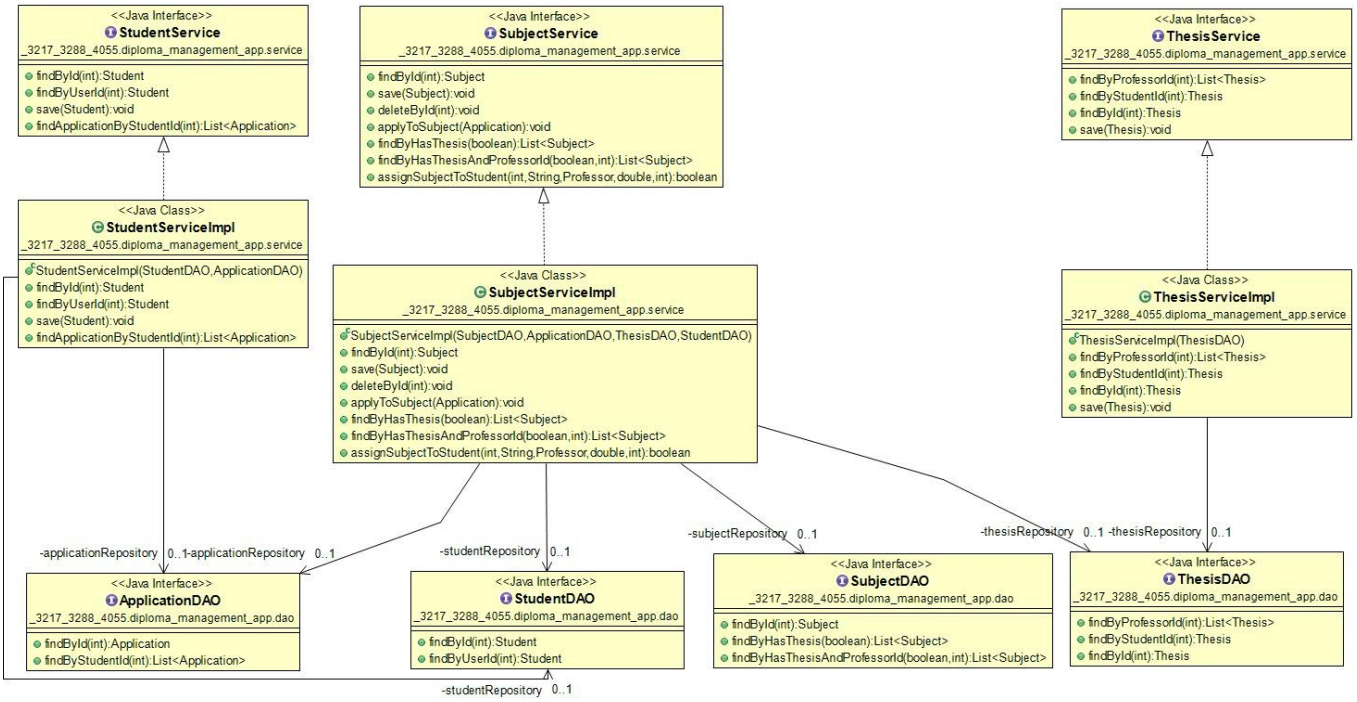


4.2 Design

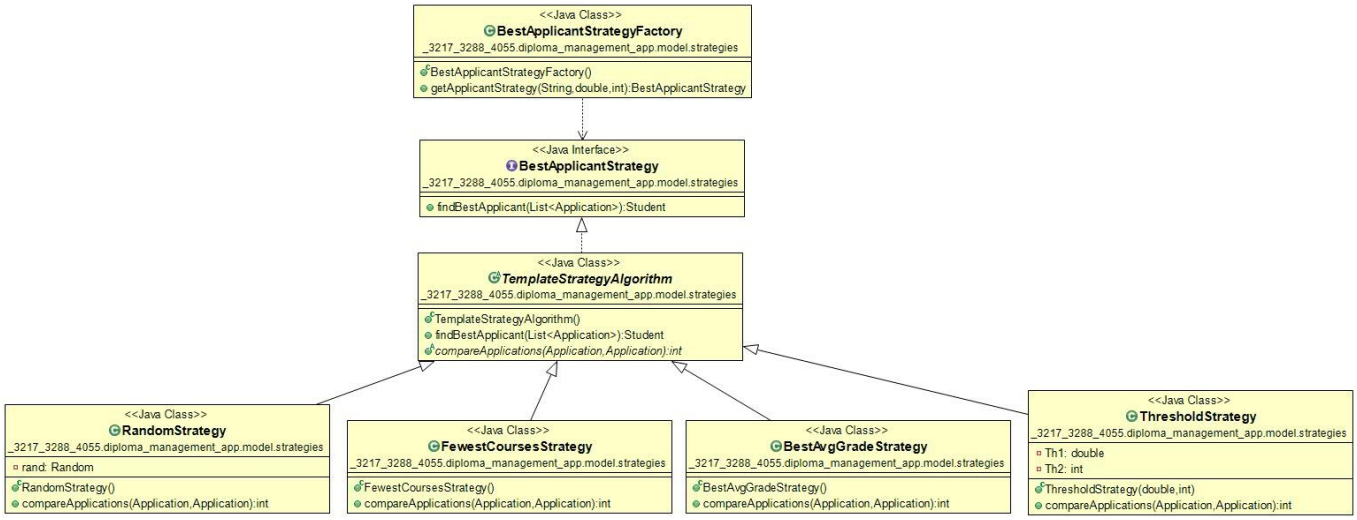
Services (part1)



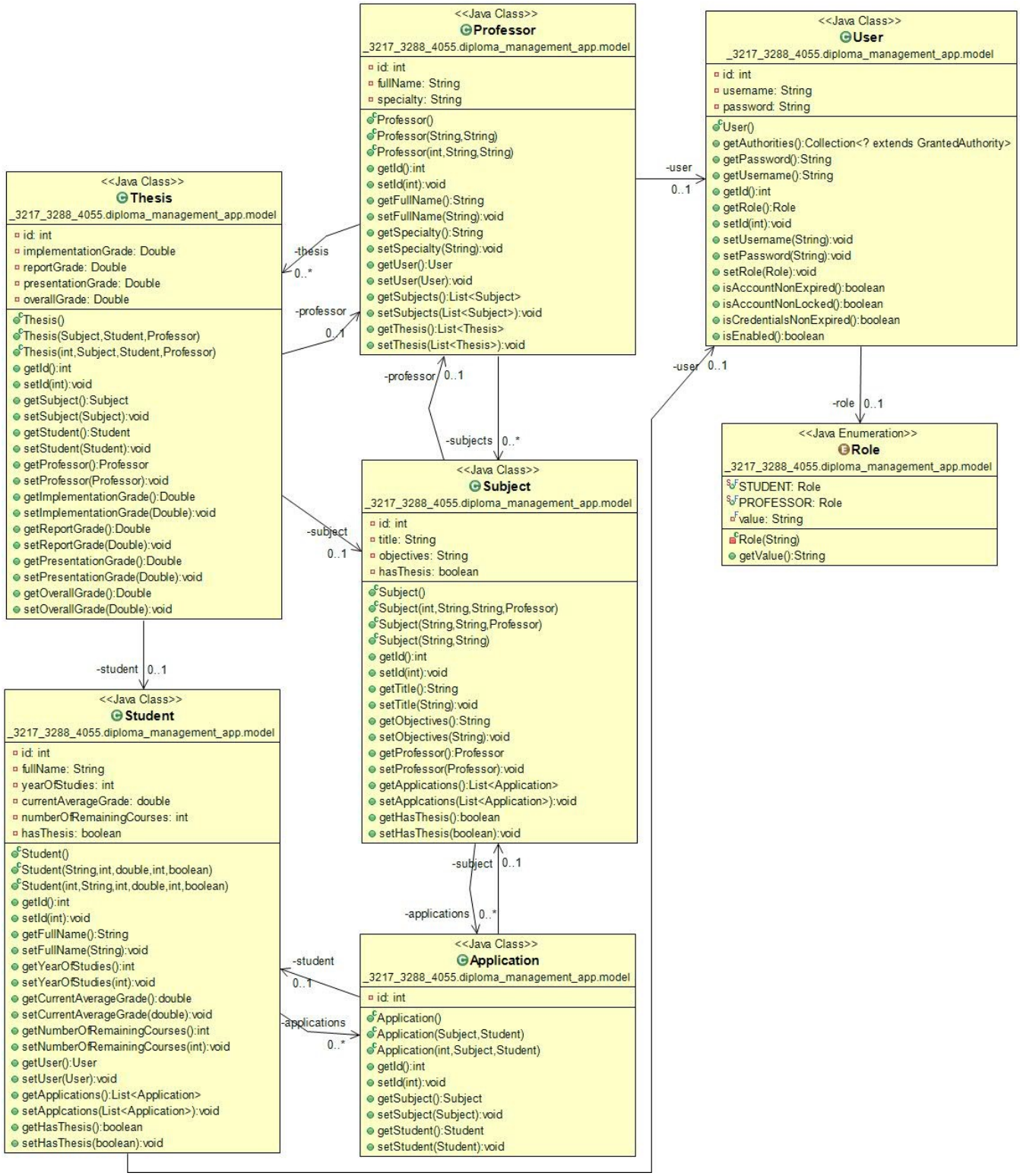
Services (part2)



Strategies



Model


<<Java Class>>
Professor
_3217_3288_4055.diploma_management_app.model
id: int
fullName: String
specialty: String
Professor()
Professor(String,String)
Professor(int,String,String)
getId():int
setId(int):void
getFullName():String
setFullName(String):void
getSpecialty():String
setSpecialty(String):void
getUser():User
setUser(User):void
getSubjects():List<Subject>
setSubjects(List<Subject>):void
getThesis():List<Thesis>
setThesis(List<Thesis>):void
<<Java Class>>
User
_3217_3288_4055.diploma_management_app.model
id: int
username: String
password: String
User()
getAuthorities():Collection<? extends GrantedAuthority>
getPassword():String
getUsername():String
getId():int
getRole():Role
setId(int):void
setUsername(String):void
setPassword(String):void
setRole(Role):void
isAccountNonExpired():boolean
isAccountNonLocked():boolean
isCredentialsNonExpired():boolean
isEnabled():boolean
-user
0..1
<<Java Class>>
Thesis
_3217_3288_4055.diploma_management_app.model
id: int
implementationGrade: Double
reportGrade: Double
presentationGrade: Double
overallGrade: Double
Thesis()
Thesis(Subject,Student,Professor)
Thesis(int,Subject,Student,Professor)
getId():int
setId(int):void
getSubject():Subject
setSubject(Subject):void
getStudent():Student
setStudent(Student):void
getProfessor():Professor
setProfessor(Professor):void
getImplementationGrade():Double
setImplementationGrade(Double):void
getReportGrade():Double
setReportGrade(Double):void
getPresentationGrade():Double
setPresentationGrade(Double):void
getOverallGrade():Double
setOverallGrade(Double):void
-thesis
0..*
-professor
0..1
-professor
0..1
-subjects
0..*
-user
0..1
-role
0..1
<<Java Enumeration>>
Role
_3217_3288_4055.diploma_management_app.model
STUDENT: Role
PROFESSOR: Role
value: String
Role(String)
getValue():String
<<Java Class>>
Subject
_3217_3288_4055.diploma_management_app.model
id: int
title: String
objectives: String
hasThesis: boolean
Subject()
Subject(int,String,String,Professor)
Subject(String,String,Professor)
Subject(String,String)
getId():int
setId(int):void
getTitle():String
setTitle(String):void
getObjectives():String
setObjectives(String):void
getProfessor():Professor
setProfessor(Professor):void
getApplications():List<Application>
setApplcations(List<Application>):void
getHasThesis():boolean
setHasThesis(boolean):void
-subject
0..1
-student
0..1
<<Java Class>>
Student
_3217_3288_4055.diploma_management_app.model
id: int
fullName: String
yearOfStudies: int
currentAverageGrade: double
numberOfRemainingCourses: int
hasThesis: boolean
Student()
Student(String,int,double,int,boolean)
Student(int,String,int,double,int,boolean)
getId():int
setId(int):void
getFullName():String
setFullName(String):void
getYearOfStudies():int
setYearOfStudies(int):void
getCurrentAverageGrade():double
setCurrentAverageGrade(double):void
getNumberOfRemainingCourses():int
setNumberOfRemainingCourses(int):void
getUser():User
setUser(User):void
getApplications():List<Application>
setApplcations(List<Application>):void
getHasThesis():boolean
setHasThesis(boolean):void
-subject
0..1
-applications
0..*
-student
0..1
-applications
0..*
<<Java Class>>
Application
_3217_3288_4055.diploma_management_app.model
id: int
Application()
Application(Subject,Student)
Application(int,Subject,Student)
getId():int
setId(int):void
getSubject():Subject
setSubject(Subject):void
getStudent():Student
setStudent(Student):void

Class Name: AuthController	
Responsibilities: ▪ login ▪ register	**Collaborations:** ▪ UserService

Class Name: ProfessorController	
Responsibilities: ▪ Get professor home page ▪ Get professor profile ▪ Show form for add professor profile ▪ Show form for update professor profile ▪ Save professor profile ▪ Get list of professor's subjects ▪ Show form for add subject ▪ Save subject ▪ Delete Subject ▪ Get list of applications ▪ Show details of a subject ▪ Show profile of a student ▪ Select strategy to assign a subject ▪ Assign subject to student ▪ Show list of thesis ▪ Show form for update thesis ▪ Show threshold form ▪ Save thesis	**Collaborations:** ▪ ProfessorService ▪ StudentService ▪ SubjectService ▪ UserService ▪ ThesisService

Class Name: StudentController	
Responsibilities: ▪ Get student home page ▪ Get student profile ▪ Show form for add student profile	**Collaborations:** ▪ StudentService ▪ SubjectService ▪ UserService

▪ Show form for update student profile ▪ Save student profile ▪ Show available diploma thesis subjects ▪ Show details of a subject ▪ Apply to subject	▪ ThesisService

Class Name: ApplicationDAO	
Responsibilities: ▪ Get application by id ▪ Get application by student id	**Collaborations:** ▪ JpaRepository ▪ Application

Class Name: ProfessorDAO	
Responsibilities: ▪ Get professor by id ▪ Get professor by user id	**Collaborations:** ▪ JpaRepository ▪ Professor

Class Name: StudentDAO	
Responsibilities: ▪ Get student by id ▪ Get student by user id	**Collaborations:** ▪ JpaRepository ▪ Student

Class Name: SubjectDAO	
Responsibilities: ▪ Get subject by id ▪ Get subjects by has thesis ▪ Get subjects by has thesis and professor id	**Collaborations:** ▪ JpaRepository ▪ Subject

Class Name: ThesisDAO	
Responsibilities: ▪ Get thesis by professor id ▪ Get thesis by student id ▪ Get thesis by id	**Collaborations:** ▪ JpaRepository ▪ Thesis

Class Name: UserDAO	
Responsibilities: ▪ Get user by username	**Collaborations:** ▪ JpaRepository ▪ User

Class Name: Application	
Responsibilities: ▪ Keeps the information of an application	**Collaborations:** ▪ Subject ▪ Student

Class Name: Professor	
Responsibilities: ▪ Keeps the information of a professor	**Collaborations:** ▪ User ▪ Subject ▪ Thesis

Class Name: Role	
Responsibilities: ▪ Keeps the information of the role	**Collaborations:** ▪

Class Name: Student	
Responsibilities: ▪ Keeps the information of a student	**Collaborations:** ▪ User ▪ Application

Class Name: Subject	
Responsibilities: ▪ Keeps the information of a subject	**Collaborations:** ▪ Professor ▪ Application

Class Name: Thesis	
Responsibilities: ▪ Keeps the information of a thesis	**Collaborations:** ▪ Subject ▪ Student ▪ Professor

Class Name: User	
Responsibilities: ▪ Keeps the information of a user	**Collaborations:** ▪ UserDetails ▪ Role

Class Name: BestApplicantStrategy	
Responsibilities: ▪ Interface for strategies	**Collaborations:** ▪

Class Name: BestApplicantStrategyFactory	
Responsibilities: ▪ Creates the selected strategy object	**Collaborations:** ▪ RandomStrategy ▪ BestAvgGradeStrategy ▪ FewestCoursesStrategy ▪ ThresholdStrategy

Class Name: BestAvgGradeStrategy	
Responsibilities: ▪ Selects the application based on the best average grade	**Collaborations:** ▪ TemplateStrategyAlgorithm ▪ Application

Class Name: FewestCoursesStrategy	
Responsibilities: ▪ Selects the application based on the fewest courses	**Collaborations:** ▪ TemplateStrategyAlgorithm ▪ Application

Class Name: RandomStrategy	
Responsibilities: ▪ Selects the application random	**Collaborations:** ▪ TemplateStrategyAlgorithm ▪ Application

Class Name: TemplateStrategyAlgorithm	
Responsibilities: ▪ Selects the best application	**Collaborations:** ▪ BestApplicantStrategy ▪ Application

Class Name: ThresholdStrategy	
Responsibilities: ▪ Selects the application based on thresholds	**Collaborations:** ▪ TemplateStrategyAlgorithm ▪ Application

Class Name: ProfessorService	
Responsibilities: ▪ Get professor by id ▪ Get professor by user id ▪ Save professor	**Collaborations:** ▪ Professor

Class Name: ProfessorServiceImpl	
Responsibilities: ▪ Get professor by id ▪ Get professor by user id ▪ Save professor	**Collaborations:** ▪ ProfessorService ▪ ProfessorDAO ▪ Professor

Class Name: StudentService	
Responsibilities: ▪ Get student by id ▪ Get student by user id ▪ Save student ▪ Get applications by student id	**Collaborations:** ▪ Student ▪ Application

Class Name: StudentServiceImpl	
Responsibilities: ▪ Get student by id ▪ Get student by user id ▪ Save student ▪ Get applications by student id	**Collaborations:** ▪ StudentService ▪ StudentDAO ▪ ApplicationDAO ▪ Student ▪ Application

Class Name: SubjectService	
Responsibilities: ▪ Get subject by id ▪ Save subject ▪ Delete subject by id ▪ Apply to subject ▪ Get subject by has thesis ▪ Get subject by has thesis and professor id	**Collaborations:** ▪ Subject ▪ Application

Class Name: SubjectServiceImpl	
Responsibilities: ▪ Get subject by id ▪ Save subject ▪ Delete subject by id ▪ Apply to subject ▪ Get subject by has thesis ▪ Get subject by has thesis and professor id	**Collaborations:** ▪ SubjectService ▪ SubjectDAO ▪ ApplicationDAO ▪ ThesisDAO ▪ StudentDAO ▪ Subject ▪ Application

Class Name: ThesisService	
Responsibilities: ▪ Get thesis by professor id ▪ Get thesis by student id ▪ Get thesis by id ▪ Save thesis	**Collaborations:** ▪ Thesis

Class Name: ThesisServiceImpl	
Responsibilities: ▪ Get thesis by professor id ▪ Get thesis by student id ▪ Get thesis by id ▪ Save thesis	**Collaborations:** ▪ ThesisService ▪ ThesisDAO ▪ Thesis

Class Name: UserService	
Responsibilities: ▪ Get user by username ▪ Save user ▪ Check if user present	**Collaborations:** ▪ User

Class Name: UserServiceImpl	
Responsibilities: ▪ Get user by username ▪ Save user ▪ Check if user present	**Collaborations:** ▪ UserService ▪ UserDetailsService ▪ BCryptPasswordEncoder ▪ UserDAO ▪ User