DiploMate: A Web Application for Posting and Searching Diploma Thesis Topics

Petros Zampelis

Diploma Thesis

Supervisor: Apostolos Zarras

Ioannina, June, 2022

ΤΜΗΜΑ ΜΗΧ. Η/Υ & ΠΛΗΡΟΦΟΡΙΚΗΣ
ΠΑΝΕΠΙΣΤΗΜΙΟ ΙΩΑΝΝΙΝΩΝ

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
UNIVERSITY OF IOANNINA

Acknowledgements

I would like to thank my supervisor, Professor Apostolos Zarras, who was an inspiration to me in the field of programming, for a wonderful collaboration.

Above all, I want to thank my family, for their faith in me and their constant support.

June, 2022

Petros Zampelis

Abstract

This web application aims to be a space where professors can upload diploma thesis topics and their details, and students can search for diploma thesis topics. Students can either apply for a topic and the application will add them in a list of candidates or they can request from the application to offer them an available topic. Professors can schedule interviews with the students on the list, and grade their outcome, so that they can finally choose the right one for the topic. In this space, professors and students will be able to be in constant consultation about everything related to their cooperation in a thesis topic, from scheduling meetings and exchanging files, to scheduling and grading its examination.

Keywords: web application, diploma thesis, topic, professor, student, automated assignment, candidates, exchanging files, scheduling, interview, meeting, examination, grading

Περίληψη

Η διαδικτυακή αυτή εφαρμογή, έχει ως στόχο να αποτελέσει έναν χώρο στον οποίον οι καθηγητές θα μπορούν να ανεβάζουν θέματα διπλωματικών, και οι φοιτητές να τα αναλαμβάνουν. Οι φοιτητές μπορούν είτε να υποβάλουν αίτηση για ένα θέμα και οι αιτήσεις τους να προσθέσουν σε μια λίστα υποψηφίων, είτε μπορούν να ζητήσουν από την εφαρμογή να τους προσφέρει ένα διαθέσιμο θέμα. Στους καθηγητές δίνεται η δυνατότητα να προγραμματίσουν συνεντεύξεις με τους φοιτητές της λίστας, και να βαθμολογήσουν την έκβαση αυτών, ώστε να μπορέσουν τελικά να επιλέξουν τον κατάλληλο. Στον χώρο αυτόν, καθηγητές και φοιτητές θα μπορούν να βρίσκονται σε συνεχή συνεννόηση σχετικά με οτιδήποτε αφορά τη συνεργασία τους σε κάποιο θέμα διπλωματικής εργασίας, από τον προγραμματισμό συναντήσεων και την ανταλλαγή αρχείων, έως τον προγραμματισμό και τη βαθμολόγηση της εξέτασης αυτού.

Λέξεις κλειδιά: διαδικτυακή εφαρμογή, διπλωματική εργασία, θέμα, καθηγητής, φοιτητής, αυτοματοποιημένη ανάθεση, υποψήφιοι, ανταλλαγή αρχείων, προγραμματισμός, συνέντευξη, συνάντηση, εξέταση, βαθμολόγηση

Table of Contents

Chapter 1. Introduction

The basic motivation of this thesis is the typical process that is followed in our department for the assignment of diploma thesis to students. In particular, this process involves issues concerning the posting of diploma thesis topics by the professors, the search for diploma thesis topics by the students, and the coordination between the supervisors and the students during the elaboration and the examination of the diploma theses.

At a glance, the assignment of diploma theses to students involves the following tasks:

- A professor has an idea and writes down a corresponding description of the diploma thesis topic.
- The professor gives a certain number of such topics to the secretary.
- The secretary posts these topics to the Department's web site.
- Students looking for a topic have to visit the web site and browse all the available descriptions. In general, the search is not supported by any dedicated facility.
- A student who finds an interesting topic has to contact the professor with some means like phone call, email, etc.
- The processor usually has to arrange a meeting with the candidate; there may be multiple candidates for a topic.
- After the meeting the processor decides whether to assign the topic to a candidate.
- Then the professor should contact the candidate somehow to let him know that he can take over the particular diploma thesis.
- During the elaboration of the thesis the supervisor regularly meets with the student. Moreover, they may have to exchange / share information about the thesis topic.

- Finally, the professor should arrange the examination of the diploma thesis, by finding a date and two more examiners that will be the member of the examination committee, together with the supervisor.
- After the examination, the committee members should grade the diploma thesis.

At the beginning of this thesis, we contacted several professors and students of the department. We discussed with them the abovementioned process and asked them what they think can be improved during several open interviews. The key outcome of these discussions was that overall, the whole process is quite informal and lacks automation. The professors and students brought out the following high-level needs:

- More automation when posting a thesis topic.
- More automation when looking for a topic.
- Easy coordination during the thesis.
- Easy arrangement of the thesis examination and grading.

This is where this thesis comes in with the development of DiploMate, an application that facilitates the assignment of diploma thesis to students by automating to a certain degree all the tasks that constitute this process.

More specifically:

- DiploMate allows a professor to post a diploma thesis topic.
- DiploMate lets a professor to accept a candidate for his / her topic.
- DiploMate allows a student to search for a topic based on his / her preferences.
- DiploMate lets a student to apply for a topic.
- DiploMate allows a student to request a topic from it.
- DiploMate lets a professor to schedule an interview / meeting / examination.
- DiploMate allows a professor to rate an interview.
- DiploMate lets a professor and a student upload some files to a topic's repository.
- DiploMate allows a committee's professors to score the examination of a topic.

The rest of this thesis is structured as follows. Chapter 2 briefly discusses the key technologies used for the development of DiploMate. Chapter 3 details the requirements analysis. Chapter 4 describes the project's design and analyzes the project's packages and classes. Chapter 5 discusses the testing process. Chapter 6 shows DiploMate's user

interface. Chapter 7 presents the results of DiploMate's evaluation. Chapter 8 summarizes the thesis, mentions some future work that could be apply to it. Finally, Chapter 9 contains the thesis references.

Chapter 2. Technologies Used

Clearly, the use of web applications today is becoming more and more common, as they have many advantages, some of which are that they are cross platform compatible, highly deployable, and more manageable. With these advantages in mind, we decided to develop DiploMate as a web application. Following, this is a list of the technologies that we used for developing DiploMate.

2.1 Java 11

Java [1] is one of the most popular programming languages. It is an object-oriented language that is famous for creating programs that are cross-functional and portable. There are also many libraries available for it, with which a developer can achieve almost anything imaginable.

2.2 Spring Boot 2.7.0

Spring Boot [2] is an open-source Java-based framework developed by Pivotal Team. It makes developing web application and microservices with Spring Framework faster and easier while these applications are also stand-alone and production ready. Below are the parts of spring framework that we used for DiploMate.

2.1.1 Spring Web

Spring Web is a dependency that contains common web specific utilities for both Servlet and Portlet environments.

2.1.2 Spring Security

Spring Security is a framework that acts as a highly customizable authenticator and access-controller for Spring Boot applications.

2.1.3 Spring Data JPA

Spring Data JPA focuses on using JPA to store data in a relational database. It can create repository implementations automatically, at runtime from a repository interface, and it is used to store and retrieve data in relational databases.

2.3 Lombok

Lombok [3] is a java annotation library that makes a developer's work easier while keeping the code clean and elegant. By annotating for example, a class with @Getters saves the developer from writing a standard java "getter" by hand and reduces the lines of code in the class file.

2.4 PostgreSQL

PostgreSQL [4] is an open-source object-relational database and is used in this project instead of MySQL or any other database. It is one of the most popular relational database management systems (RDBMSs) and is used in this project due to the author's prior experience with this particular DBMS.

2.5 HTML5

HTML [5] (HyperText Markup Language) is the widely known standard markup language for documents designed to be displayed in a web browser. We use HTML to write DiploMate's front-end.

2.5.1 Thymeleaf

Thymeleaf [6] is a Java template engine for web environments. It is better suited for serving HTML5 at the view layer of MVC-based web applications. It provides full Spring framework integration and so it is a very good way for us to display any information returned from our Spring Boot back-end to the front-end HTML pages.

2.6 Maven

Maven [7] is a software project management tool hosted by the Apache Software Foundation. It basically handles things like the dependency injections, builds, processing, etc., so that the developer's work gets easier.

2.7 Spring Initializr

Finally, all the above-mentioned technologies were selected and put together in a basic project structure by using the Spring Initializr [8]. It is a web application that can generate a Spring Boot project structure for you just by filling some name fields, selecting the Spring Boot version (2.7.0), the language (Java) and the project type (Maven) you prefer. You can also select any dependencies such as those mentioned above, and procced to generate a zip file that contains the project's structure that you will be working on.

Chapter 3. Requirements Analysis

Below are some tables, containing the User Stories and then the Use Cases that were created during the design phase of DiploMate.

3.1 User Stories

User Stories are short sentences that contain information about the features of an application, described from the end-user's point of view. They are very important, since in addition to showing the features of the application, also describe them in simple words and help to understand why the user needs them. Our User Stories are divided into categories, for their better understanding.

3.1.1 Basic Features

These are the features of DiploMate that a user can access from the login page, registration page and forgot password page, including a simple logout feature. Table 1 contains these features' User Stories.

User story	As a …	I want to be able to …	so that …
1	user	create an account	I can manage some personal information and secure access to this information.
2	user	login to DiploMate	I can use it.
3	user	logout from DiploMate	I can stop using it.

4	user	reset my password	I now have secure access to my account.

Table 1 User Stories of the basic DiploMate features

3.1.2 Administrator-Related Features

These are the features of DiploMate that the administrator can access after he / she logs in to the application. Table 2 contains these features' User Stories.

User story	As an …	I want to be able to …	so that …
5	administrator	view the administrator home page	I can access all the administrator-related features of DiploMate.
6	administrator	access the university's courses list	I can manage the list's contents.
7	administrator	access the university's tags list	I can manage the list's contents.

Table 2 User Stories of the administrator-related DiploMate features

3.1.3 Profile-Related Features

These are the features of DiploMate that a professor / student can access from the profile page. Table 3 contains these features' User Stories.

User story	As a …	I want to be able to …	so that …
8	user	view my profile page	I can manage the profile information.
9	user	change my profile	I can manage the profile information.

Table 3 User Stories of the profile-related DiploMate features

3.1.4 Topic-Related Features

We should explain right here that in DiploMate we divide the topics into two main categories, the standard, and the open ones.

A standard topic is a topic that has a candidates list, is displayed to the students during search, any student can apply for it, and the professor who added it has the last word on who will this topic be assigned to. The professor can also select a minimum score for any course that he / she deems necessary, so this information is visible to the students who might be interested in applying for the topic.

On the other hand, an open topic's main differences are that the professor cannot set a minimum score as required to any course. The topic will not be returned to the students as a search result and will only be offered to a student that requests a topic from DiploMate. This is how we try to solve the problem of students not finding a topic, but it is also at the discretion of the professors to add at least some open topics each. It is worth mentioning that when DiploMate offers an open topic to a student, it gives to the student exactly one day to accept or reject it, and then makes it available again for other students, while adds it in a list of unwanted topics for the student who rejected it, so it is not offered to him / her again. This is happening so the maximum number of open topics is always available for the students requesting them, and to not allow a student to keep a topic for a long time to finally not accept it.

Tables 4, 5 and 6 contain these features' User Stories, divided by the user's role.

User story	As a …	I want to be able to …	so that …
10	professor	view the topics page	I can see the topics I have added.
11	professor	add a new standard topic	a new topic is now available for all students to apply for.
12	professor	add a new open topic	a new topic is now available for DiploMate to offer automatically.
13	professor	change a topic's	I can manage the

		details	topic's information.
14	professor	see the candidates applied for a topic	I can schedule interviews with them.
15	professor	schedule an interview with a candidate of a thesis topic	I can assess whether the candidate is suitable for the topic.
16	professor	see a topic's completed interviews	I can select the most appropriate candidate.
17	professor	rate candidates' completed interviews	I can select the most appropriate candidate.
18	professor	assign a topic to the most appropriate candidate	the topic is now taken and is no longer available for application.
19	professor	delete a topic	I can manage my topics list.
20	professor	see a list of my old topics	I can access their repositories.
21	professor	receive an email notification about a student who took over one of my topics from the open topics list	I can now start my cooperation with the student.

Table 4 User Stories of the topic-related DiploMate features for professors

User story	As a …	I want to be able to …	so that …

22	student	view the topics page	I can see the topic I have taken or search for one if I have not taken any.
23	student	I want to receive an email notification about the addition of a new topic	I am always aware about which topics are available.
24	student	receive an email notification about my acceptance / rejection for a topic	I know that I now have a topic / I am no longer a candidate for it.
25	student	receive an email notification about the removal of a topic, if I am a candidate for it, has been offered to me, or I used to work on it	I know that the topic no longer exists.
26	student	search for a topic based on a professor's full name and / or some tags	so that I can find the topics that match my preferences and apply for them.
27	student	apply for a topic	I can let a professor know that I am interested to take over the topic.
28	student	see the topics I have applied for	I can keep track of my applications and possibly cancel one or more of them.
29	student	cancel my	I can let the prof

		application for a standard topic	know that I am no longer interested in the topic.
30	student	automatically assign a topic to me from the open topics list	I can take over a diploma thesis topic as soon as possible.
31	student	accept a topic that was automatically assigned to me from the open topics list	let DiploMate know that I am willing to take over the topic that was automatically assigned to me.
32	student	reject a topic that was automatically assigned to me from the open topics list	let DiploMate know that I am not willing to take over the topic that was automatically assigned to me.

Table 5 User Stories of the topic-related DiploMate features for students

User story	As a …	I want to be able to …	so that …
33	user	view another user's profile	I can get some information about him / her.
34	user	see a topic's detailed description	I can get more information about what this topic is about.
35	user	see one of the topics' I have added / the topic's I have taken repository	I can see the uploaded files and download / delete them or add more.

36	user	upload a PDF file to a topic's repository	I can add content that is relevant to the topic.
37	user	receive an email notification about a new file's upload in a topic's repository	I am aware of the addition of new content related to the topic.
38	user	delete a PDF file I uploaded to a topic's repository	I can remove content that is not anymore relevant to the topic.
39	user	download a PDF file from a topic's repository that the topic's supervised / supervisor uploaded	I can read the file's content and store it to my personal computer.

Table 6 User Stories of the topic-related, not user role specific, DiploMate features

3.1.5 Calendar-Related Features

These are the features of DiploMate that a user can access from the calendar page, we divide them into not user specific and professor specific. Tables 7 and 8 contain these features' User Stories, divided by the user's role.

User story	As a …	I want to be able to …	so that …
40	user	view the calendar page	I can access the calendar-related features of DiploMate.
41	user	receive an email notification about a session that a professor has	I can join the professor for an interview / meeting / examination.

		scheduled	
42	user	get notified via email about a session's cancelation	I know that this interview / meeting / examination is no longer scheduled.
43	user	receive an email notification about the completion of the scoring of (one of) my topic(s)	I know the final topic's score.

Table 7 User Stories of the calendar-related, not user role specific, DiploMate features

User story	As a …	I want to be able to …	so that …
44	professor	schedule a meeting with one or all my students	keep track of his / her / their progress.
45	professor	schedule an examination for one of the students supervised by me	me and two more professors can rate his / hers work on the topic.
46	professor	re-schedule an interview / meeting / examination I have scheduled	the session's date and / or time are now changed.
47	professor	cancel a session	the interview / meeting / examination is no longer scheduled.
48	professor	see all the unrated examinations I have participated in	I can score them.
49	professor	score an examination of a	the student work's scoring is

		student's work on a topic that I scheduled or got invited to	completed.
50	professor	receive a notification about an examination that I participated in, and I have not scored yet	I can remember it and go score it.

Table 8 User Stories of the calendar-related DiploMate features for professors

3.2 Use Cases

Use Cases expand on the features mentioned in the User Stories section, by adding more details and explaining the basic scenarios in which the user uses each feature. This is achieved by describing the sequence of steps the user follows. In Use Cases we also state the conditions that must be met for the execution of each action and the results. In contrast to the categorized way in which we presented our User Stories, we will follow a more sequential approach in presenting the features of DiploMate in this chapter, to emphasize the actions that begin after the end of others.

RegisterUser
ID: UC1
Actors: User
Preconditions: The user is on DiploMate's login page

Flow of events:
1. The Use Case starts when the user clicks on the "Sign up" link
2. The system displays on the screen some fields for the user to fill in (name, email, role, etc)
3. The user fills in the fields
4. The user clicks the "Sign up" button
5. If the information provided is valid
  1. The system creates the account
  2. The system sends an email with a verification link to the user's email
  3. The system redirects the user to the login page and displays a message saying that a verification email has been sent to him / her and has 20 minutes to activate the account.
  4. The user goes to his / her email and clicks on the link

5. If the link's token is valid 1. The system enables the user's account 2. The system redirects the user to the login page and displays a message saying that the verification was completed successfully 6. Else 1. The system displays an error message on the screen 6. Else 1. The system displays an error message on the screen
Postconditions: The user has created an account

LoginUser
ID: UC2
Actors: User
Preconditions: The user has an enabled account
Flow of events: 1. The Use Case starts when the user goes to DiploMate's website 2. The system displays on the screen an email and a password field for the user to fill in 3. The user types his / her email and password 4. If the information provided its valid 1. The system logs the user in to DiploMate and redirects him / her to the profile page 5. Else 1. The system displays an error message on the screen
Postconditions: The user is logged in to DiploMate

LogoutUser
ID: UC3
Actors: User
Preconditions: The user is logged in to DiploMate
Flow of events: 1. The Use Case starts when the user clicks on the "Logout" button 2. The system displays a window asking the user if he / she really wants to log out from DiploMate 3. If the user selects "Yes" 1. The system logs the user out of DiploMate and redirects him / her to DiploMate's login page
Postconditions: The user is logged out of DiploMate

ResetUserPassword
ID: UC4
Actors: User
Preconditions: The user is on DiploMate's login page

Flow of events:
 	
1. The Use Case starts when the user clicks on the "Forgot password" link
2. The system displays an email field for the user to fill in
3. The user types his / her email and clicks "Send"
4. The system sends an email to the user with a link and displays a message on the screen informing the user about it
5. The user goes to his / her email and clicks on the link
6. If the link's token is valid
  1. The system redirects the user to a window with a new password and a reentered new password field
  2. The user types the password and clicks "Reset"
  3. The system checks that the passwords match and updates the user's password
7. Else
  1. The system displays an error message on the screen

Postconditions: The user's password has been updated

ViewAdminHomePage
ID: UC5
Actors: Administrator
Preconditions: The administrator is logged in to DiploMate
Flow of events: 1. The Use Case starts after UC2 2. The system displays on the screen the administrator's home page for the administrator to access Diplomate's administrator-related features
Postconditions: The administrator's home page is displayed on the screen

UpdateCoursesList
ID: UC6
Actors: Administrator
Preconditions: The administrator is logged in to DiploMate
Flow of events: 1. The Use Case starts when the administrator clicks on the "Update courses list" button 2. The system displays on the screen a list of courses for the administrator to fill in / edit

3. The administrator fills in / edits the list and clicks “Update”
4. If the information provided is valid
  1. The system updates the courses list, and redirects the administrator to the administrator’s home page
5. Else
  1. The system displays an error message on the screen

Postconditions: The courses list has been updated

UpdateTagsList
ID: UC7
Actors: Administrator
Preconditions: The administrator is logged in to DiploMate

Flow of events:
1. The Use Case starts when the administrator clicks on the “Update tags list” button
2. The system displays on the screen a list of tags for the administrator to fill in / edit
3. The administrator fills in / edits the list and clicks “Update”
4. If the information provided is valid
  1. The system updates the tags list, and redirects the administrator to the administrator’s home page
5. Else
  1. The system displays an error message on the screen

Postconditions: The tags list has been updated

ViewProfilePage
ID: UC8
Actors: User
Preconditions: The user is logged in to DiploMate
Flow of events: 1. The Use Case starts after UC2 or when the user clicks on the “Profile” button 2. The system displays on the screen the user’s profile page for the user to access DiploMate’s profile-related features
Postconditions: The user’s profile page is displayed on the screen

UpdateProfileInfo
ID: UC9
Actors: User

Preconditions: The user is logged in to DiploMate and on the profile page

Flow of events:
 	
1. The Use Case starts when the user clicks on the “Update profile” button
2. The system displays on the screen the user’s information
3. The user updates the information that he / she wants and clicks “Update”
4. If the information provided is valid
  1. The system updates the information and redirects the user to the profile page
5. Else
  1. The system displays an error message on the screen

Postconditions: The user’s information has been updated

UpdatePassword
ID: UC10
Actors: User
Preconditions: The user is logged in to DiploMate and on the profile page

Flow of events:
 	
1. The Use Case starts when the user clicks on the “Update password” button
2. The system displays on the screen three password fields for the user to fill in (old, new, reentered new)
3. The user fills in the fields and clicks “Update”
4. If the passwords are valid
  1. The system updates the user’s password, and redirects him / her to the profile page
5. Else
  1. The system displays an error message on the screen

Postconditions: The user’s password has been updated

ViewTopicsPage
ID: UC11
Actors: User
Preconditions: The user is already logged in to DiploMate
Flow of events: 1. The Use Case starts when the user clicks on the “Topics” button 2. The system displays the topics page and the topic(s) he / she has added / taken for the user to access DiploMate’s topic-related features
Postconditions: The user’s topics page is displayed on the screen

AddStandardTopic

ID: UC12
Actors: Professor
Preconditions: The professor is on the topics page

Flow of events:
1. The Use Case starts when the professor clicks on the “Add new standard topic” button
2. The system displays on the screen some fields for the professor to fill in, some tags to select and some courses to set minimum score for
3. The professor fills in the fields, selects some tags, and sets some minimum scores for the courses that he / she deems required, and clicks “Add”
4. If the information provided is valid
  1. The system adds the new standard topic and redirects the professor to the topics page
5. Else
  1. The system displays an error message on the screen

Postconditions: A new standard topic has been added

AddOpenTopic
ID: UC13
Actors: Professor
Preconditions: The professor is on the topics page

Flow of events:
1. The Use Case starts when the professor clicks on the “Add new open topic” button
2. The system displays on the screen some fields for the professor to fill in and some tags to select
3. The professor fills in the fields and selects some tags and clicks “Add”
4. If the information provided is valid
  1. The system adds the new open topic and redirects the professor to the topics page
5. Else
  1. The system displays an error message on the screen

Postconditions: A new open topic has been added

NotifyStudentForNewTopic
ID: UC14
Actors: Professor
Preconditions: A professor has just added a new standard topic
Flow of events: 1. The Use Case starts after UC12

2. The system calculates the similarity of the new standard topic's tags, with the interest tags that all the students without a topic have selected 3. If the similarity is above a threshold   1. The system sends an email to the fitting students that the topic might interest them
Postconditions: An email has been sent to some students

ViewTopicDetails
ID: UC15
Actors: User
Preconditions: The user is on the topics page and there is at least one topic displayed
Flow of events: 1. The Use Case starts when the user clicks on the "See details" button of a topic 2. The system displays on the screen the topic's details
Postconditions: The topic's details are displayed on the screen

UpdateTopicDetails
ID: UC16
Actors: Professor
Preconditions: The professor is on the topics page and there is at least one topic displayed

Flow of events:
1. The Use Case starts when the professor clicks on the "Update details" button of one of his / her topics
2. The system displays on the screen the topic's details
3. The professor changes the details that he / she wants and clicks "Update"
4. If the information provided is valid
  1. The system updates the topic's details, and redirects the professor to the topics page
5. Else
  1. The system displays an error message on the screen

Postconditions: The topic's details have been updated

ViewTopicCandidates
ID: UC17
Actors: Professor
Preconditions: The professor is on the topics page and there is at least one topic

displayed
Flow of events: 1. The Use Case starts when the professor clicks on the “See candidates” button of one of his / her topics 2. The system displays on the screen the topic’s candidates
Postconditions: The topic’s candidates are displayed on the screen

ScheduleInterview
ID: UC18
Actors: Professor
Preconditions: The professor is on a topic’s candidates page and there is at least one candidate displayed

Flow of events:
1. The Use Case starts when the professor clicks on the “Schedule interview” button of one of his / her topic’s candidates
2. The system displays on the screen a date and a time for the professor to select
3. The professor selects the date and time and clicks “Schedule”
4. The system verifies that none of the participants has something other scheduled for this date and time and if it is true
  1. The system schedules the interview
  2. The system sends a notification email to the candidate
  3. The system redirects the professor to the calendar page and displays a message on the screen saying that the session has been scheduled successfully
5. Else
  1. The system displays an error message on the screen

Postconditions: The interview has been scheduled

ViewCompletedInterviews
ID: UC19
Actors: Professor
Preconditions: The professor is on the topics page and there is at least one topic displayed
Flow of events: 1. The Use Case starts when the professor clicks on the “Completed interviews” button of one of his / her topics 2. The system displays on the screen all the completed interviews for this topic
Postconditions: The topic’s competed interviews are displayed on the screen

RateCompletedInterview
ID: UC20
Actors: Professor
Preconditions: There is at least one completed interview displayed on the screen

Flow of events:
1. The Use Case starts after the completion of the UC19
2. The system displays on the screen one or more completed interviews, with a score field on the unrated ones for the professor to fill in
3. The professor fills in the score
4. If the score is valid
  1. The system saves the interview's score, and orders it in the correct place on an ordered list
5. Else
  1. The system displays an error message on the screen

Postconditions: The interview has been rated and ordered

AcceptCandidate
ID: UC21
Actors: Professor
Preconditions: There is at least one candidate for a professor's topic

Flow of events:
1. The Use Case starts when the professor clicks on the "Accept candidate" button on the candidates or the completed interviews page
2. The system displays a window on the screen, asking the professor if he / she really wants to accept this candidate for the topic
3. If the professor selects "Yes"
  1. The system assigns the topic to the candidate
  2. The system sends an email to notify the student that he / she got accepted
  3. The system removes all other candidates for the topic from the candidates list
  4. The system sends an email to notify the rest of the students that they are no longer candidate for this topic

Postconditions: The topic has been assigned to the student

NotifyCandidateAboutAcceptanceOrRejection
ID: UC22
Actors: Professor
Preconditions: The professor has just accepted a student for one of his / her topics

Flow of events: 1. The Use Case starts after UC21 2. The system sends an email to notify the accepted student that he / she got accepted 3. The system sends an email to notify the rest of the students that they are no longer candidates for the topic
Postconditions: All the topic's candidates have been notified via email about their acceptance / rejection

DeleteTopic
ID: UC23
Actors: Professor
Preconditions: The professor is on the topics page and there is at least one topic displayed

Flow of events:
1. The Use Case starts when the professor clicks on the "Delete" button of one of his / her topics
2. The system displays a window on the screen, asking the user if he / she really wants to delete this topic
3. If the professor selects "Yes"
  1. The system deletes the topic
  2. The system sends an email to notify the student who owned the topic, or the topic's candidates, or the student the topic was offered to, that it has been deleted

Postconditions: The topic has been deleted

NotifyStudentAboutTopicDeletion
ID: UC24
Actors: Professor
Preconditions: The professor has just deleted a topic
Flow of events: 1. The Use Case starts after UC23 2. The system sends an email to notify the student who took over the topic, or the topic's candidates, or the student the topic was offered to, that it has been deleted
Postconditions: All the students related to the topic have been notified about its deletion

ViewOldTopics

ID: UC25
Actors: Professor
Preconditions: The professor is on the topics page
Flow of events: 1. The Use Case starts when the professor clicks on the “Old topics” button 2. The system displays on the screen all the professor’s old topics
Postconditions: All the professor’s old topics are displayed on the screen

SearchTopics
ID: UC26
Actors: Student
Preconditions: The student is on the topics page and has not a topic yet

Flow of events:
1. The Use Case starts when the student clicks on the “Search topics” button
2. The system displays a dropdown menu with all the professors and a list of tags for the student to select
3. The student selects his / her preferences and clicks “Search”
4. If the tags selection is valid
  1. The system calculates the similarity between the student preferences and all the available topics’ tags
  2. The system displays on the screen an ordered list of topics
5. Else
  1. The system displays an error message on the screen

Postconditions: Available topics matching the student’s preferences are displayed on the screen

ViewOtherProfile
ID: UC27
Actors: User
Preconditions: The user sees a “View profile” button next to a name on the screen
Flow of events: 1. The Use Case starts when the user clicks on the “View profile” button, next to a user’s name 2. The system displays on the screen the user’s profile page
Postconditions: The selected user’s profile is displayed on the screen

ApplyForATopic

ID: UC28
Actors: Student
Preconditions: The student has an available standard topic's details displayed on the screen after UC15

Flow of events:
1. The Use Case starts when the user clicks on the "Apply as candidate" button
2. If the user has uploaded a PDF file with his / her completed scoring on the university's courses and has not already applied for the topic
  1. The system adds the student to the topic's candidates list
  2. The system displays on the screen a message saying the application has been completed
3. Else
  1. The system displays an error message on the screen

Postconditions: The student has been added to the topic's candidates list

ViewMyTopicApplications
ID: UC29
Actors: Student
Preconditions: The student is on the topics page
Flow of events: 1. The Use Case starts when the student clicks on the "My topic applications" button 2. The system displays on the screen all the active applications that the student has made
Postconditions: All the active applications that the student has made are displayed on the screen

CancelTopicApplication
ID: UC30
Actors: Student
Preconditions: The student has made an application and UC29 has been completed
Flow of events: 1. The Use Case starts when the student clicks on the "Cancel" button of one of his / her applications 2. The system asks the student if he / she really wants to cancel the application to this topic 3. If the student selects "Yes"   1. The system deletes the student from the topic's candidates list

Postconditions: The student has been removed from the topic's candidates list

AutomaticallyAssignTopicFromOpenTopicsList
ID: UC31
Actors: Student
Preconditions: The student is on the topics page and has not a topic yet

Flow of events:
1. The Use Case starts when the student clicks on the "Offer me a topic" button
2. The system calculates the similarity between the student interest tags and the available open topics tags, decides the best one for him / her and if there is an available open topic
  1. The system displays on the screen the topic's details and gives the student one day to accept or reject it
3. Else
  1. A message is displayed on the screen, notifying the student that there is no available topics to be offered right now

Postconditions: The student has a topic offered to him / her and one day to accept or reject it

ReplyToAutomaticTopicOffer
ID: UC32
Actors: Student
Preconditions: A topic has been offered to the student and the student is seeing its details

Flow of events:
1. The Use Case stars when the student clicks on the "Accept" or "Reject" button
2. If the student clicks "Accept"
  1. The system assigns the topic to the student
  2. The system sends an email to the professor who added the topic, to notify him / her about the topic's assignment
3. Else
  1. The system adds the topic to a list, so it does not offer it again to this student
  2. The system makes the topic available again

Postconditions: The topic is now assigned to the student or has been made available again

NotifyProfessorAboutAutomaticTopicAssignment

ID: UC33
Actors: Student
Preconditions: A student has just accepted an offered topic
Flow of events: 1. The Use Case starts after UC32 2. The system sends an email to the professor who added the topic, to notify him / her about the topic's assignment
Postconditions: An email has been sent to the professor

ViewTopicRepositoryPage
ID: UC34
Actors: User
Preconditions: There is a topic on the topics page
Flow of events: 1. The Use Case starts when the user clicks on a topic's "Repository" button 2. The system displays on the screen all the files uploaded to this topic's repository and a form for the user to upload a new one
Postconditions: The topic's repository content is displayed on the screen

UploadFile
ID: UC35
Actors: User
Preconditions: The user is on a topic's repository page after UC34

Flow of events:
1. The Use Case starts when the user clicks on the "Add new file" button
2. The system opens an explorer window
3. The user selects a file and clicks "Open"
4. The system gives the proper name to the file, checks if a file with the same name exists and if not
  1. The system adds the file to the topic's repository
  2. The system sends an email to notify the other user related to the topic that a new file has been added to its repository
5. Else
  1. The system displays an error message on the screen

Postconditions: A file has been added to the topic's repository

DeleteUploadedFile

ID: UC36
Actors: User
Preconditions: The user is on a topic's repository page and has uploaded a file
Flow of events 1. The Use Case starts when the user clicks on the "Delete" button next to a file's name 2. The system displays a window asking the user if he / she really wants to delete the file 3. If the user selects "Yes"   1. The system removes the file from the topic's repository
Postconditions: The file has been removed from the topic's repository

DownloadUploadedFile
ID: UC37
Actors: User
Preconditions: The user is on a topic's repository page and there is one file displayed
Flow of events: 1. The Use Case starts when the user clicks on the "Download" button next to the file's name 2. The system downloads the file to the user's computer
Postconditions: The file has been downloaded to the user's computer

ViewCalendarPage
ID: UC38
Actors: User
Preconditions: The user is logged in to DiploMate
Flow of events: 1. The Use Case starts when the user clicks on the "Calendar" button 2. The system displays on the screen the calendar page with any scheduled session divided in categories
Postconditions: The calendar page is displayed on the screen

ScheduleMeeting
ID: UC39
Actors: Professor

Preconditions: The professor is on the calendar page and supervises at least one student

Flow of events:
 	
1. The Use Case starts when the professor clicks on the “Schedule meeting” button
2. The system displays on the screen a dropdown menu with the professor’s students and an option to select all of them at the same time and a date and time for the professor to select
3. The professor selects the participants, date and time and clicks “Schedule”
4. The system verifies that none of the participants has something other scheduled for this date and time and if that is true
  1. The system schedules the meeting
  2. The system sends a notification email to all the participants
  3. The system redirects the professor to the calendar page and displays a message on the screen that the session has been scheduled successfully
5. Else
  1. The system displays an error message on the screen

Postconditions: The meeting has been scheduled

ScheduleExamination
ID: UC40
Actors: Professor
Preconditions: The professor is on the calendar page and has at least one student

Flow of events:
 	
1. The Use Case starts when the professor clicks on the “Schedule examination” button
2. The system displays on the screen three dropdown menus, for the professor to select a student and two other professors, and a date and time for him / her to also select
3. The professor selects the participants, date and time and clicks “Schedule”
4. The system verifies that none of the participants has something other scheduled for this date and time and if that is true
  1. The system schedules the examination
  2. The system sends a notification email to all the participants
  3. The system redirects the professor to the calendar page and displays a message on the screen that the session has been scheduled successfully
5. Else
  1. The system displays an error message on the screen

Postconditions: The examination has been scheduled

RescheduleSession
ID: UC41
Actors: Professor
Preconditions: The professor has already scheduled an interview, meeting, or

examination

Flow of events:
1. The Use Case starts when the professor clicks on the “Reschedule” button of one of his / her scheduled interviews, meetings, or examinations
2. The system displays on the screen the session’s participants and a date and time for the professor to select
3. The professor selects the date and time and clicks “Reschedule”
4. The system verifies that none of the participants has something other scheduled for this date and time and if that is true
  1. The system reschedules the session
  2. The system sends a notification email to the participants
  3. The system redirects the professor to the calendar page and displays a message on the screen that the session has been rescheduled successfully
5. Else
  1. The system displays an error message on the screen

Postconditions: The session has been rescheduled

CancelSession
ID: UC42
Actors: Professor
Preconditions: The professor has already scheduled an interview, meeting, or examination

Flow of events:
1. The Use Case starts when the professor clicks on the “Cancel” button of one of his / her scheduled interviews, meetings, or examinations
2. The system displays a window asking the professor if he / she really wants to cancel this session
3. If the professor selects “Yes”
  1. The system deletes the session
  2. The system sends an email to the invited participants that the session has been canceled
  3. The system redirects the professor to the calendar page and displays a message on the screen that the session has been canceled successfully

Postconditions: The session has been canceled

SeeUnratedExaminations
ID: UC43
Actors: Professor
Preconditions: The professor is on the calendar page
Flow of events:

1. The Use Case starts when the professor clicks on the “Unrated examinations” button 2. The system displays on the screen all the passed unrated examinations that the professor has participated in and not scored yet
Postconditions: The passed unrated examinations are displayed on the screen

ScoreExamination
ID: UC44
Actors: Professors Committee
Preconditions: The professors, members of the committee, are seeing the unrated examinations that they have participated in and not scored yet, on the screen

Flow of events:
1. The Use Case starts when a professor fills in a score in an unrated examination’s score field
2. If the score is valid
  1. The system saves the score
  2. if all three members of the professors committee have scored the examination
  1. The system calculates the examination’s average score
  2. The system notifies the professor and the student related to the topic about the final score
  3. The system redirects the professor to the calendar page

Postconditions: The examination has been scored by the professors committee members

NotifyProfessorForUnratedExamination
ID: UC45
Actors: Professor
Preconditions: The professor has participated in an examination that has not scored yet
Flow of events: 1. The Use Case starts when the professor logs in to DiploMate 2. The system checks if the professor has any examination that has passed but not scored yet and if there is   1. The system sends an email notifying the professor about it
Postconditions: An email has been sent to the professor

Chapter 4. Design

In this chapter we explain the architectures we used to develop DiploMate. In addition, we talk about all DiploMate's packages, and the classes each one of them contains. We also include UML diagrams of the packages and classes, for the application's better understanding.

4.1 MVC Architecture

MVC [10] (Model-View-Controller) is the architecture pattern that we use in DiploMate. More specifically, MVC is a pattern, which divides the program into three kinds of elements. Model represents the data we used in the application. View contains the application's user interface. Controller combines the two previous ones and sends information between them. The use of this architecture pattern grands the developer better division of work and improved maintenance of the program.

4.2 Controller – Service – Repository Architecture

Most Spring Boot applications also use the Controller - Service - Repository architecture that is a kind of Multitier architecture [11]. This architecture is the perfect way to handle a complex application. The Controller is where the front and the back end of an application communicate. From the Controller we access the Service. The Service is where all the business logic of an application is happening. From the Service we access the Repository. The Repository is from / to where we retrieve / store an application's data. This architecture is also used in our application to achieve better code management.

4.3 Packages

In Figure 1 we give an overview of DiploMate's architecture in terms of the packages that constitute it.

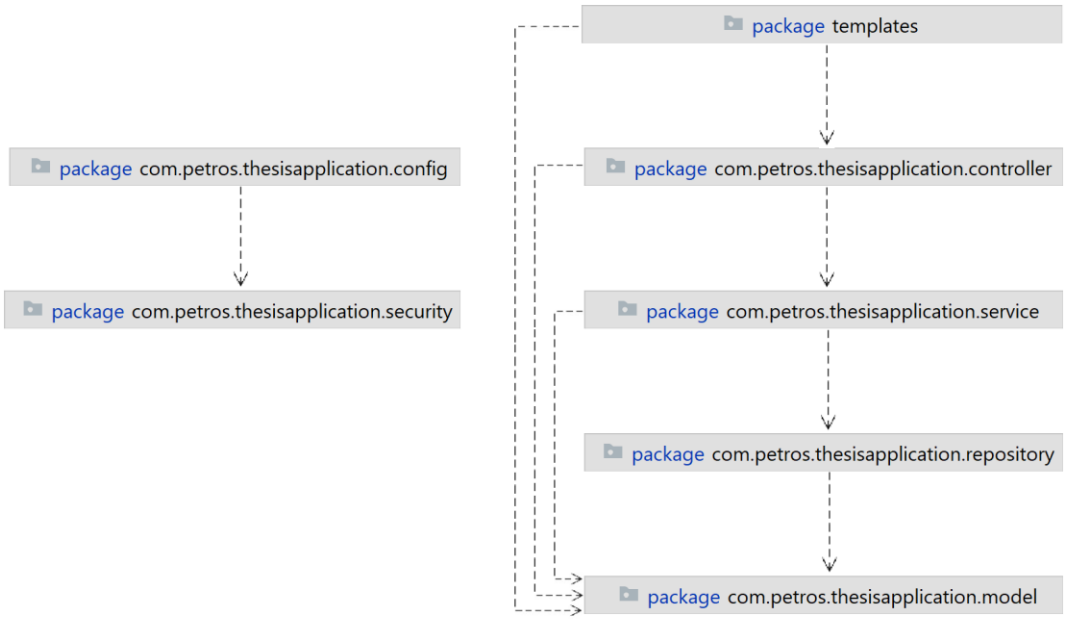

Figure 1 DiploMate packages overview

4.3.1 Model

In the model package we have the domain model of DiploMate. In more detail, in this package we keep the classes that represent the data used in our application. These classes are practically the entities of our program. Within these classes we include the variables of each, together with some methods to get / set them. This package is obviously related to the Model element of the MVC architecture pattern. Figure 2 and Figure 3 that follow below, give the classes that constitute the domain model, in the form of UML diagrams. The diagrams also contain the relations between the classes. We also describe each class in more detail.

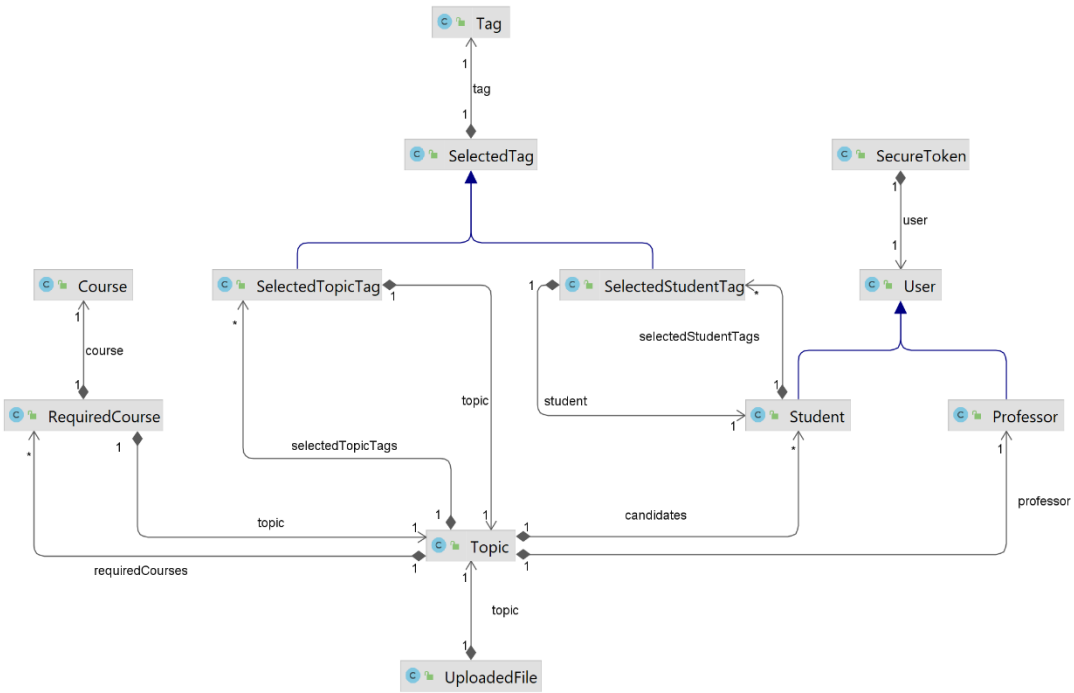

Figure 2 User and Topic related classes, including Tag and Course

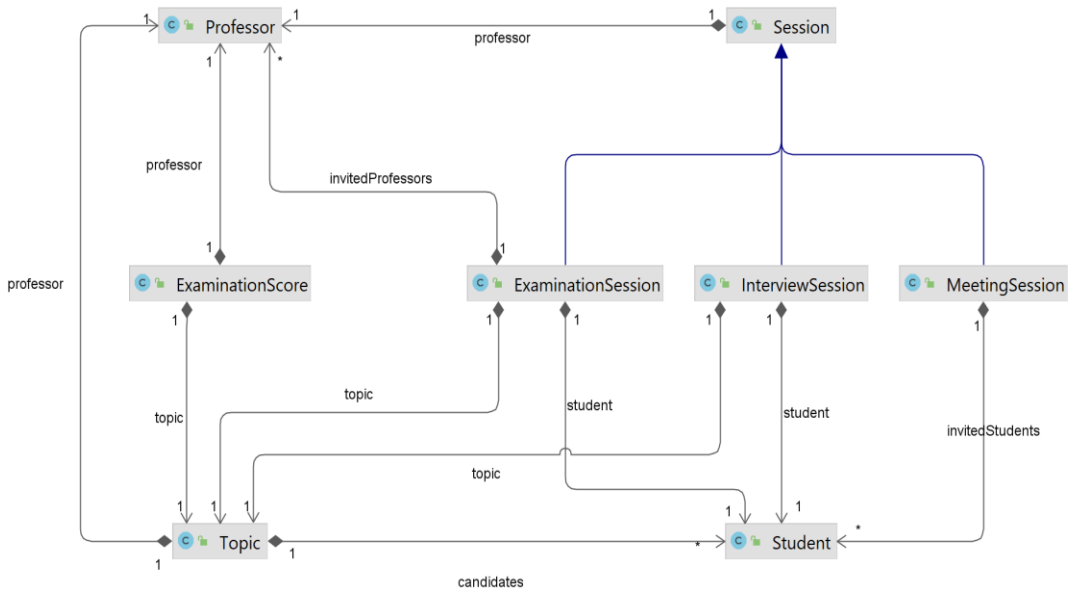

Figure 3 Session and Examination Score, related to Professor, Student and Topic

User

User defines the basic properties of Diplomate's users, like a unique identifier, their first and last name, a unique email address, a password and a role (administrator, professor, student), a profile image path, a GitHub link, some information text about them and also an “is activated” variable that is true or false base on if their account is activated or not.

It is worth mentioning that we include the "role" property as we use a common repository for all the users, and we also use it to our front end. User has two extensions, Professor and Student. In addition to the abovementioned properties, these two classes define even more properties, based on the user's role. In more detail, in Professor we define a phone number, some office information and a web page link. In Student, we define a path for his / her grades PDF and a list of tags.

Topic

Topic defines the properties of topics. Specifically these properties include a unique identifier, a unique name, a type (standard, open), a score, a status (application period, interviews period), an application deadline, an acceptance deadline and a list of students not interested in it (if the topic is open), the professor who added it, the student who has taken it, a list of candidates and a list of required courses (if the topic is standard), and finally a list of tags. The list of students not interested in the topic, is included in the case of an open topic, so that we don't offer the same topic twice to a student that does not like it.

SecureToken

We assign a secure token to every user that creates an account in DiploMate. SecureToken defines the properties of this secure token. These properties include a unique identifier, a token, an expiration date, and a redeemed variable (that is either true or false). We fill the user secure token's token variable with a random generated string, every time we need to verify the user's validity. More specifically we fill and send the token to the user when he / she needs to verify his / her account for the first time, and when he / she forgets his / her password and wants to reset it. The token is accessed via the link we send to the user's email. DiploMate verifies the token's validity, and the user can proceed. It is worth mentioning that we chose to make the token valid for only twenty (20) minutes, after that time has elapsed, the link will no longer work, and the user will have to repeat the process. We finally empty this token so it cannot be used again.

Course

Course defines the properties of a university's course, like a unique identifier and a unique name.

RequiredCourse

RequiredCourse defines the properties of a topic's required course. These properties include a unique identifier, a course, a topic, and a minimum score for the required course.

Tag

Tag defines the properties of a university's tag, like a unique identifier and a unique text.

SelectedTag

SelectedTag defines the properties of a selected tag. These properties include a unique identifier, a type (student tags, topic tag), a tag, and an "is selected" variable (that is either true or false). It is worth mentioning that we include the "type" property as we use a common repository for all the selected tags, and we also use it to our front end. We should also explain that the "is selected" variable exists, so that a student / topic has even the tags that is not selected for him / her / it stored in the database. This happens to make the update of a student / topic easier, in the case another tag gets selected / unselected for him / her / it. SelectedTag has two extensions, SelectedStudentTag and SelectedTopicTag. In addition to the abovementioned properties, these two classes define even more properties, based on the selected tag's type. In more detail, in SelectedStudentTag we also define the student that has selected the tag, while in SelectedTopicTag we also define the topic that the tag has been selected for.

UploadedFile

UploadedFile defines the properties of an uploaded file, like a unique identifier, a unique name, the uploader's role (professor, student), the date that the file was uploaded and the topic in whose repository the file was uploade.

RegistrationRequest

RegistrationRequest defines the properties of the request that a user makes to register to DiploMate. These properties include a first and last name, an email address, a password and a reentered password, and finally a user role (professor, student).

Session

Session defines the basic properties of Diplomate's sessions, like a unique identifier, a type (interview, meeting, examination), a date, a time and the professor who scheduled it. Session has three extensions, InterviewSession, MeetingSession and

ExaminationSession. In addition to the abovementioned properties, these three classes define even more properties, based on the session's type. In more detail, in InterviewSession, we define a student, a topic and a score with which the professor rated the interview. In MeetingInterview, we define a list of students invited to it. Finally, in ExaminationSession, we define a student a topic and a list of professors that got invited to it.

ExaminationScore

ExaminationScore defines the properties of an examination's scoring, like a unique identifier, a topic and the professor who gave the score, and also a presentation, a text, a methodology and preparation score.

4.3.2 Repository

In the repository package, we have a repository for each one of our entities. A repository is a place in which we store our entities' data. Our repositories extend the JpaRepository, so we can use the preexisting methods defined in it. Some basic examples are the findById and findAll methods, from which we get an entity based on its identifier, if the entity exists, and a list of all stored entities respectively. In addition, in most of these repositories, we have added some extra methods that we can call. By using all the above-mentioned methods, we can save, update, and retrieve the data from these repositories.

4.3.3 Service

In the service package we have a service for each one of our entities. Inside the service classes, we have written all the business logic of DiploMate. This means that all the methods included in these classes, are methods that perform a specific task, related to the entity they include in their titles.

UserService

In this service we have written methods related to the users. Some examples are the methods that we use to sign up a user to DiploMate, load a user from the database, save a user to the database, update a user's information, etc.

TopicService

In this service we have written methods related to the topics. For example, we have methods here that we use to create a topic, load a topic from the database, save a topic

to the database, update a topic's details, accept a topic's candidate, calculate the topic's tags similarity with a student's interest tags, etc.

SecureTokenService

In this service we have written methods related to the secure tokens. Some examples are the methods that we use to initialize a secure token, load a secure token from the database, save a secure token to the database, delete a secure token from the database, update and empty a secure token, etc.

CourseService

In this service we have written methods related to the courses. For example, we have methods here that we use to get all the courses in different formats, check a courses list's validity, update the courses list, etc.

RequiredCourseService

In this service we have written methods related to the required courses. Some examples are the methods that we use to initialize the required courses for a topic, update the required courses, etc.

TagService

In this service we have written methods related to the tags. For example, we have methods here that we use to get all the tags in different formats, check a tags list's validity, update the tags list, etc.

SelectedTagService

In this service we have written methods related to the selected tags. Some examples are the methods that we use to add a student's / topic's selected tag to the database, update a student's / topic's selected tags, get a student's / topic's selected tags from the database, etc.

UploadedFileService

In this service we have written methods related to the uploaded files. For example, we have methods here that we use to upload and save a file to the database, get a file from the database, delete a file from the database, etc.

RegistrationService

In this service we have written a method related to the registration requests. Specifically, through this method, we send the verification link to a newly registered user.

SessionService

In this service we have written methods related to the sessions. Some examples are the methods that we use to create a new session, delete a session from the database, get all the upcoming sessions of a user, etc.

ExaminationScoreService

In this service we have written methods related to the examination scores. Specifically, we have a method to get the examination score base on the professor and the topic, a method that rates a topic, and a method that completes a topic's scoring.

EmailService

In this service we have written methods related to the emails that DiploMate sends. Some examples are the methods that we use to send the email, to build the emails content based on the occasion, etc.

EmailValidatorService

In this service we have written a method related to the email's validity test. Specifically, in this class we implement the Predicate interface, and override the test method. Through this method we can check an email (if for example it ends with an organization's domain) by using some regex. Although we call it, it returns always true, as we have not set a specific regex. We include it for completeness reasons.

4.3.4 Controller

In the controller package we have several controller classes. A controller is the part of the Spring Boot program, where the back end, communicates with the front end. More specifically, in a controller, we select which HTML page the user sees, based on his / her actions. We also get the information that the user provides in some instances (like when a professor wants to add a new topic). We access the service layer, in which we perform the required business logic. From there we access the repository layer, so we can save, update and / or retrieve the require data. Finally, from the controller, we redirect the user to the correct HTML page, and possibly give him / her feedback.

Below we provide more information about each one of the controller classes we have in DiploMate. We also include some diagrams that illustrate the controllers, and their relations with the service and repository layers.

RegistrationController

By using the RegistrationController, we display DiploMate's registration page to the user. We take the data the user provides to register to the application, and store the user in DiploMate's database. This controller uses the SecureTokenService, UserService and RegistrationService. SecureTokenService, and UserService access the SecureTokenRepository and UserRepository respectively, as shown in Figure 4. Also, in Table 9 we provide even more details for the controller.

Use Cases	Services	Repositories	Entities
UC1	SecureTokenService, UserService, RegistrationService	SecureTokenRepository, UserRepository	SecureToken, User

Table 9 Registration controller details

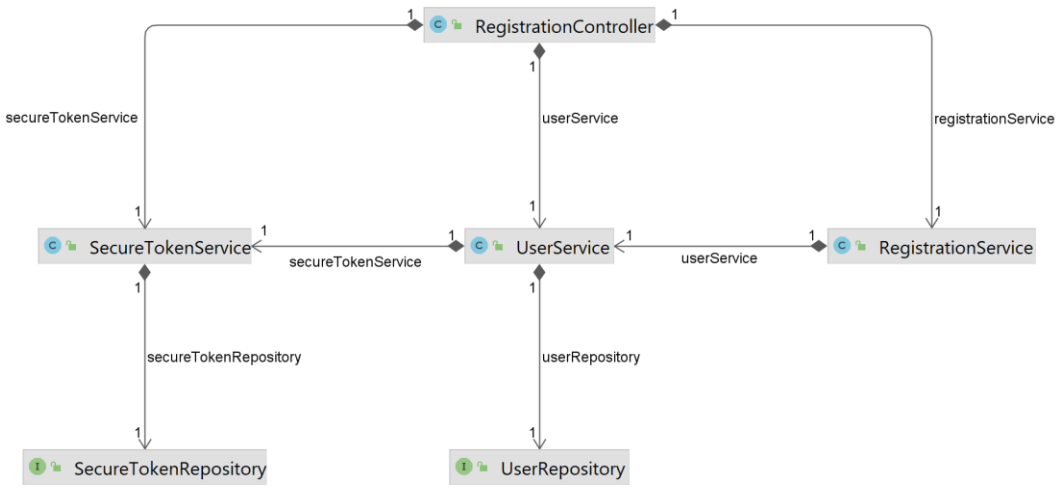

Figure 4 The controller responsible for the user's registration to DiploMate, and its relations

LoginLogoutController

By using the LoginLogoutController, we display DiploMate's login page to the user. We take the data the user provides to login to the application, verify them and redirect the user to his / her profile page. We also realize the logout action and redirect the user to the login page when he / she logs out of DiploMate. This controller uses the UserService

and SessionService. These services access the UserRepository and SessionRepository respectively, as shown in Figure 5. Also, in Table 10 we provide even more details for the controller.

Use Cases	Services	Repositories	Entities
UC2, UC3, UC45	UserService, SessionService	UserRepository, SessionRepository	User, Session

Table 10 Login-Logout controller details

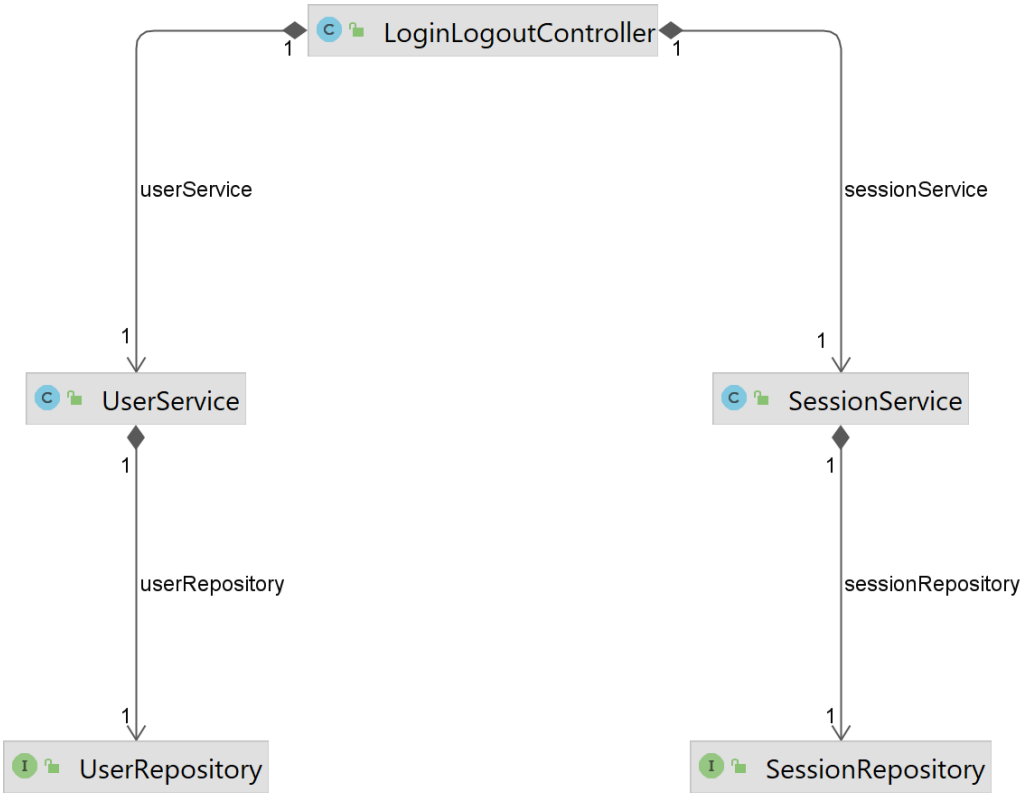

Figure 5 The controller responsible for the user's log in and log out of DiploMate, and its relations

ForgotPasswordController

By using the ForgotPasswordController, we display DiploMate's forgot password page to the user. We take the data the user provides to reset his / her password and help him / her complete the process. This controller uses the UserService and SecureTokenService. These services access the UserRepository and SecureTokenRepository respectively, as shown in Figure 6. Also, in Table 11 we provide even more details for the controller.

Use Cases	Services	Repositories	Entities
UC4	UserService, SecureTokenService	UserRepository, SecureTokenRepository	User, SecureToken

Table 11 Forgot password controller details

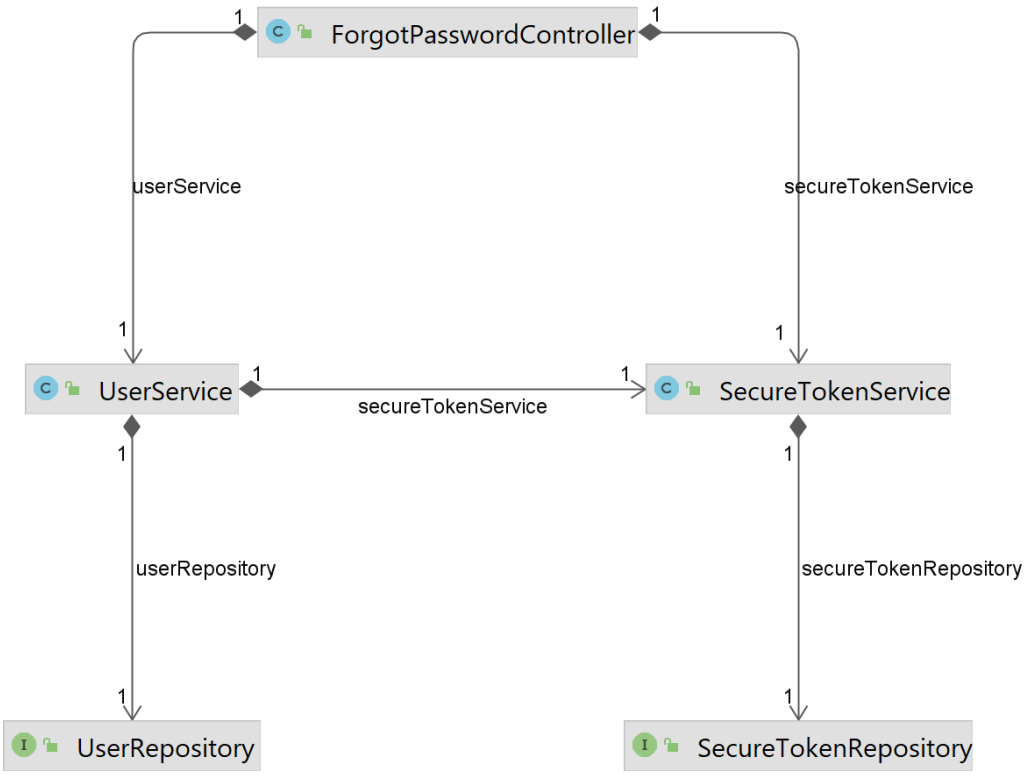

Figure 6 The controller responsible for resetting the user's password in DiploMate, and its relations

UserController

By using the UserController, we display to the user all DiploMate's pages that are not specific to the user's role. We also take the data the user provides and perform the requested not user specific actions. This controller uses the UserService, TopicService, TagService, SelectedTagService, and UploadedFileService. While these services use each other when needed, they also access the UserRepository, TopicRepository TagRepository, SelectedTagRepository, and UploadedFileRepository. respectively, as shown in Figure 7. Also, in Table 12 we provide even more details for the controller.

Use Cases	Services	Repositories	Entities

UC8, UC10, UC15, UC27, UC34, UC35, UC36, UC37	UserService, TopicService, TagService, SelectedTagService, UploadedFileService	UserRepository, TopicRepository, TagRepository, SelectedTagRepository, UploadedFileRepository	User, Topic, Tag, SelectedTag, UploadedFile

Table 12 User controller details

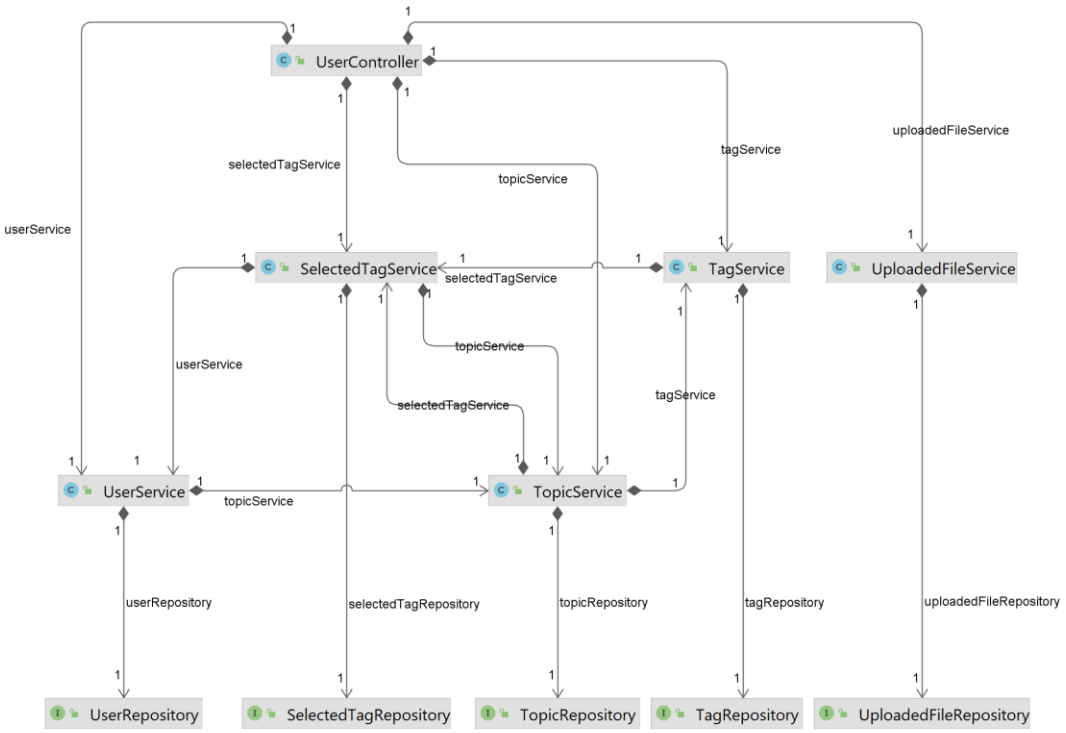

Figure 7 The controller responsible for all the not user specific features of DiploMate, and its relations

AdminController

By using the AdminController, we display to the user all DiploMate's pages that can be accessed by the administrator. We also take the data the administrator provides and perform the requested administrator specific actions. This controller uses the UserService, CourseService and TagService. While these services use each other when needed, they also access the UserRepository, CourseRepository and TagRepository respectively, as shown in Figure 10. Also, in Table 13 we provide even more details for the controller.

Use Cases	Services	Repositories	Entities
UC5, UC6, UC7,	UserService,	UserRepository,	User, Course, Tag

UC10	CourseService, TagService	CourseRepository, TagRepository	

Table 13 Administrator controller details

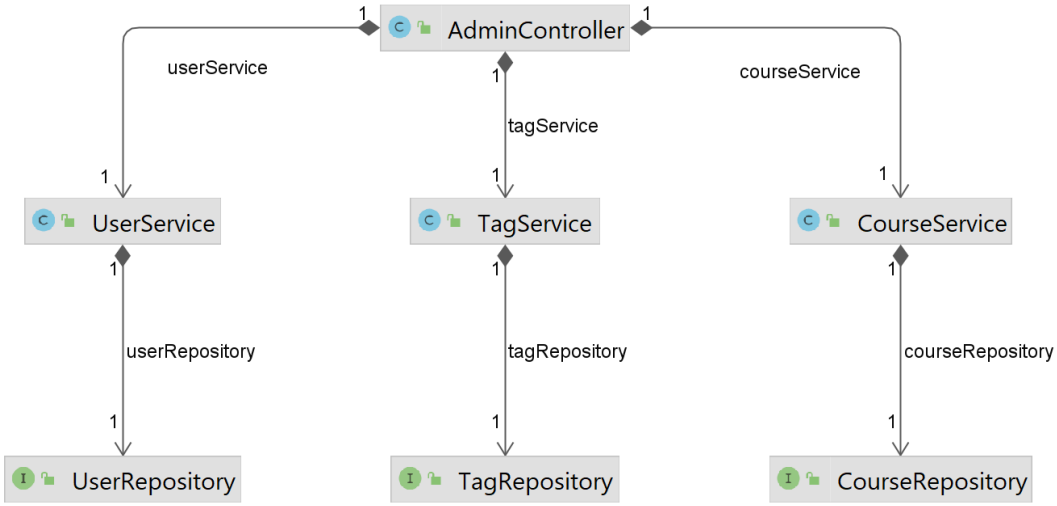

Figure 8 The controller responsible for all the administrator specific features of DiploMate, and its relations

ProfessorController

By using the ProfessorController, we display to the user all DiploMate's pages that can be accessed by professors. We also take the data the professor provides and perform the requested professor specific actions. This controller uses the UserService, TopicService, TagService, SelectedTagService, CourseService, RequiredCourseService, UploadedFileService, SessioinService and ExaminationScoreService. While these services use each other when needed, they also access the UserRepository, TopicRepository TagRepository, SelectedTagRepository, CourseRepository, RequiredCourseRepository, UploadedFileRepository, SessioinRepository and ExaminationScoreRepository respectively, as shown in Figures 9 and 10. Also, in Table 14 we provide even more details for the controller.

Use Cases	Services	Repositories	Entities
UC11, UC12,	UserService, TopicService, TagService,	UserRepository, TopicRepository,	User, Topic, Tag, SelectedTag,

UC13, UC14, UC16, UC17, UC18, UC19, UC20, UC21, UC22, UC23, UC24, UC25	SelectedTagService, CourseService, RequiredCourseService, UploadedFileService, SessionService, ExaminationScoreService	TagRepository, SelectedTagRepository, CousreRepository, RequiredCousreRepository, UploadedFileRepository, SessionRepoository, ExaminationScoreRepository	Course, RequiredCourse, UploadedFile, Session, ExaminationScore

Table 14 Professor controller details

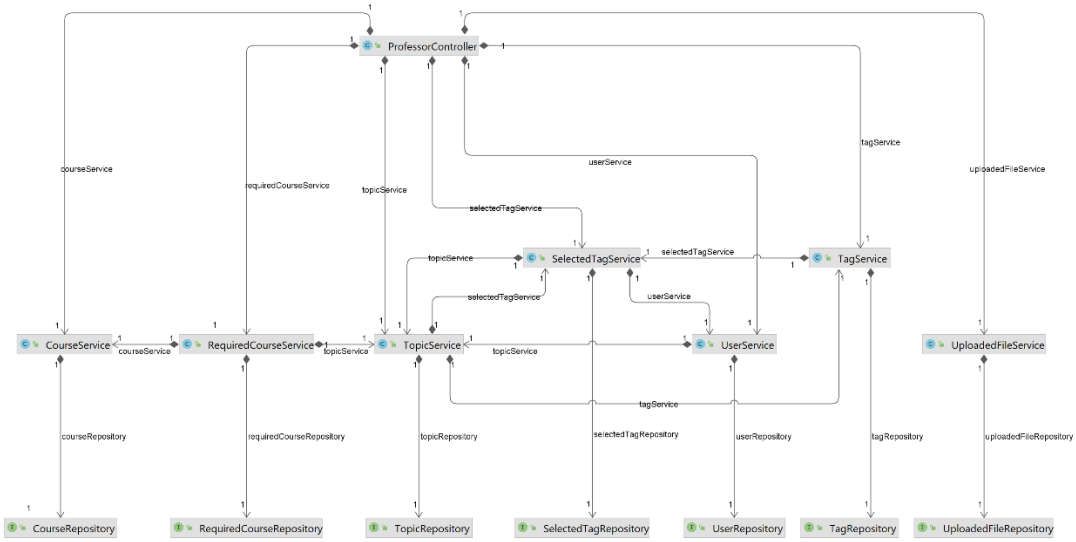

Figure 9 The controller responsible for all the professor specific features of DiploMate, and its relations (diagram1)

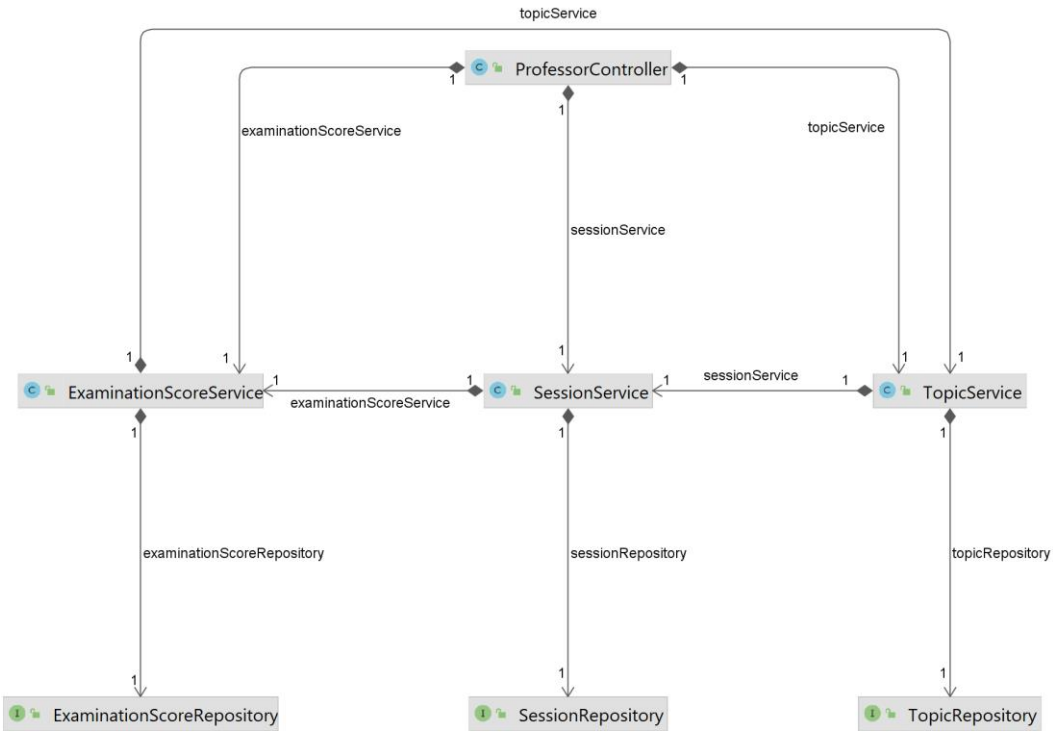

Figure 10 The controller responsible for all the professor specific features of DiploMate, and its relations (diagram2)

StudentController

By using the StudentController, we display to the user all DiploMate's pages that can be accessed by students. We also take the data the student provides and perform the requested student specific actions. This controller uses the UserService, TopicService, TagService, SelectedTagService, UploadedFileService and SessioinService. While these services use each other when needed, they also access the UserRepository, TopicRepository, TagRepository, SelectedTagRepository, UploadedFileRepository, and SessioinRepository respectively, as shown in Figure 11. Also, in Table 15 we provide even more details for the controller.

Use Cases	Services	Repositories	Entities
UC11, UC26, UC28, UC29, UC30, UC31, UC32, UC33	UserService, TopicService, TagService, SelectedTagService, UploadedFileService, SessionService	UserRepository, TopicRepository, TagRepository, SelectedTagRepository, UploadedFileRepository, SessionRepoository	User, Topic, Tag, SelectedTag, UploadedFile, Session

Table 15 Student controller details

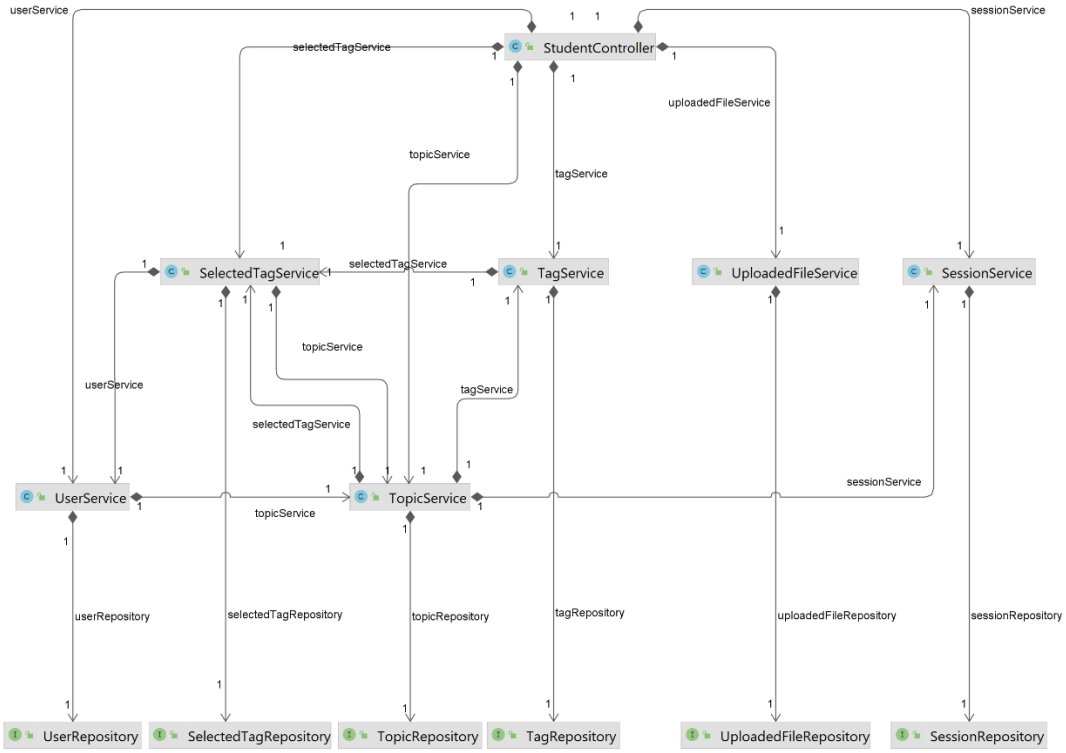

Figure 11 The controller responsible for all the student specific features of DiploMate, and its relations

4.3.5 Templates

This package contains multiple HTML5 files, which constitute the front-end of DiploMate. Being contained in the resources folder, Spring Boot can access them at runtime, and display them on the user's screen. This package constitutes the View in MVC architecture.

4.3.6 Config

The config package contains two classes, ResourceConfig and WebServiceConfig, shown in Figure 12.

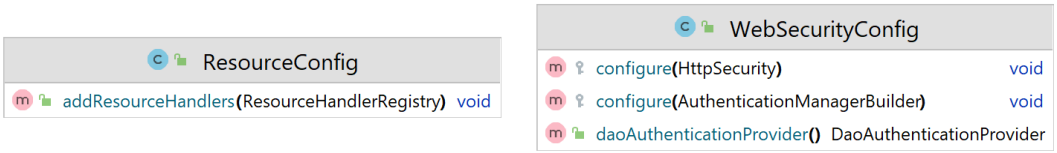

Figure 12 Config package classes and its methods

ResourceConfig

DiploMate needs to access the uploads folder that we have already created in our project's directory. Inside this folder, DiploMate saves and retrieves files like profile images and PDFs. ResourceConfig is a configuration class, that defines the uploads folder as a folder that contains useful resources. In this way, DiploMate can access the uploads folder, and use its contents at runtime.

WebServiceConfig

We use the WebServiceConfig class, to configure everything related to the user's access on DiploMate, from setting the default login and logout paths, to allowing and denying access to users depending on their roles and ids.

We'll expand a little further on the last part about accesses, by providing an example on a DiploMate's link. DiploMate's links look something like this:

http://localhost:8080/api/professor/1/profile

This link shows that the user with the role of a professor and the identifier number 1, is on the profile page. This link can be accessed only by the abovementioned user. Any other user with a different role and / or identifier, is forbidden from accessing the link by the configurations in this class.

4.3.7 Security

The security package also contains two classes, PasswordEncoder and Guard, shown in Figure 13.

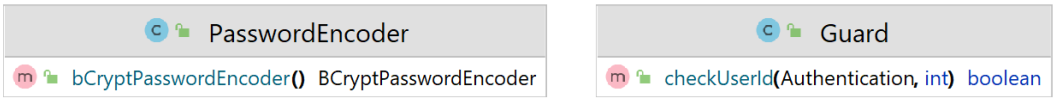

Figure 13 Security package classes and its methods

PasswordEncoder

PasswordEncoder defines the encoding used in DiploMate users' passwords. In this way, the passwords that the users use and are stored in the database, are secure and cannot be seen by anyone else.

Guard

Guard is responsible for verifying that the page the user wants to access, can be accessed based on his identifier. In this way the user cannot access another user's data by manually changing the link in the address bar.

4.4 Expanding on Some of DiploMate's Features

4.4.1 Administrator's Profile

DiploMate currently creates a user with the administrator role automatically at runtime, so this type of user cannot be created by the sign-up process.

This happens to ensure that there cannot be more than one administrator created and is a temporary solution just for the DiploMate's administrator-related features to be accessed and used.

4.4.2 University Courses

DiploMate allows the administrator to add up to one hundred (100) courses. This number is checked and has been selected for mainly visualization purposes. Duplicate courses (with the same name) are not allowed, and Diplomate checks and informs the administrator in this case.

While the administrator adds or deletes a new or pre-existing course, DiploMates also adds or removes this course from / to any new or pre-existing topic's courses, for it to be available to set a minimum score to or not.

4.4.3 University Tags

DiploMate allows the administrator to add up to fifty (50) tags. This number is also checked and has been selected for mainly visualization purposes. Duplicate tags (with the same name) are also not allowed, and Diplomate checks and informs the administrator in this case.

While the administrator adds a new tag, DiploMate also adds this new tag and makes it available to be selected as a new and pre-existing topic's tag and a student's interest tag.

But for the administrator to delete a tag, DiploMate checks if this tag has been selected as a topic's and / or student's interest tag, and in the case it is not, it then removes it. In other case, DiploMate keeps the tag in the tags list, to avoid leaving topics and / or students with no tags selected.

It is worth mentioning that a professor must select at least one (1) and up to five (5) tags for a topic that he / she want to add / update. On the other hand, a student may not select any tag if he / she wishes but can still select up to five (5) tags as his / her interest tags.

4.4.4 Session's Duration

We have set the session duration to one (1) hour by default. This means that when DiploMate checks the availability of an invited user, it checks that the user has nothing scheduled from the time the session is scheduled until one (1) hour later.

4.4.5 Similarity Check

In terms of DiploMate's features that use a similarity check, this similarity is calculated using the Jaccard Similarity algorithm.

Jaccard Similarity

Jaccard Similarity [9] (or Jaccard Index) is a similarity metric which calculates the similarity between two datasets, in our case two lists of tags, the one containing a topic's tags and the other containing the student's interest tags. The formula used follows below.

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|}$$

Where:

A, B are the two sets,

$A \cap B$ is the number of common items between the two sets and

$A \cup B$ is the number of all the items in both sets minus the common ones.

Chapter 5. Testing

In this chapter we explain how we tested DiploMate. Below we present the technologies used, as well as the exact way we used them to achieve our goal.

5.1 Technologies

5.1.1 JUnit 5

A JUnit test [12] is a small method, which we include in a test class. We use them to test the methods we have written inside our application's classes. We specifically use the JUnit 5 version.

5.1.2 Mockito

Mockito [13] is a framework that we use inside our JUnit tests. More specifically, by using Mockito, we manage to "mock" the behavior of some of our methods contained in the Service layer. By doing this, we can get the returned results we want from a method without having to set up and establish a real database for our application during the testing process.

5.2 Testing Process

DiploMate is a big project, that contains multiple classes and multiple methods. We emphasized on testing some of the more important methods of each class. More specifically, we tested methods that are contained in the classes of the Service and Controller layers. Overall, we developed 75 tests encapsulated in 16 test classes. Below we elaborate on these tests.

5.2.1 Application Test

The application test is a small and basic test. By calling this test, we basically make sure that the Spring application context is loaded successfully at the start of the application. So, we are now sure that our Spring Boot application starts correctly. In Figure 14 we see the successful execution of this test.

Figure 14 Application test's successful execution

5.2.2 Service Tests

As we established so far, the Service layer is where the business logic of DiploMate happens. So, each one of the methods inside the Service layer classes, perform a specific task for DiploMate. We emphasized on testing the methods that could throw an exception, in the case of some invalid input. To do so, we wrote small JUnit tests for these methods, and provided them valid as well as invalid inputs. We finally tested the output of these methods, to make sure that they performed the correct task when we provided the valid inputs, and that they threw the correct exception in the case of the invalid inputs. We summarize the details in the Table 16. In Figure 15 we also see the successful execution of these tests.

Service	Test Class	Tested Methods
UserService	UserServiceTest	signUpUser, resetPassword
TopicService	TopicServiceTest	createdTopic, updateTopic, deleteTopicById, acceptCandidate
RegistrationService	RegistrationServiceTest	registerNewUser
CourseService	CourseServiceTest	updateCoursesList, checkCoursesListValidity

TagService	TagServiceTest	updateTagsList, checkTagsListValidity
SessionService	SessionServiceTest	scheduleInterviewSession, scheduleMeetingSession, scheduleExaminationSession, cancelSession
UploadedFileService	UploadedFileServiceTest	saveNewFile, uploadFile
ExaminationScoreService	ExaminationScoreServiceTest	rateExamination, completeExaminationScoring

Table 16 Service tests details

- ✔ ExaminationScoreServiceTest
  - ✔ whenExaminationScoringIsComplete_shouldCallEmailSendTwoTimesAndSetTopicScoreToTheRightScore()
  - ✔ scoringAnExamination_whenUsingRateExamination_shouldCallSaveOneTime()
- ✔ UploadedFileServiceTest
  - ✔ givingAnExistingName_whenSavingANewFile_shouldCallSaveOneTime()
  - ✔ uploadFile_shouldCallTransferToOneTime()
  - ✔ givingANewName_whenSavingANewFile_shouldCallSaveOneTime()
- ✔ CourseServiceTest
  - ✔ checkingCoursesListValidity_whenGivenCoursesListIsNotValid_shouldReturnException()
  - ✔ addingTwoCourse_whenUpdatingAnEmptyCoursesList_shouldCallSaveTwoTimes()
  - ✔ checkingCoursesListValidity_whenGivenCoursesListIsValid_shouldNotReturnException()
  - ✔ addingACourse_whenUpdatingAnEmptyCoursesList_shouldCallSaveOneTime()
- ✔ TagServiceTest
  - ✔ checkingTagsListValidity_whenGivenTagsListIsNotValid_shouldReturnException()
  - ✔ checkingTagsListValidity_whenGivenTagsListIsValid_shouldNotReturnException()
  - ✔ addingTwoTags_whenUpdatingAnEmptyTagsList_shouldCallSaveTwoTimes()
  - ✔ addingATag_whenUpdatingAnEmptyTagsList_shouldCallSaveOneTime()
- ✔ TopicServiceTest
  - ✔ creatingAnOpenTopic_whenCreatingATopic_shouldReturnAnOpenTopic()
  - ✔ creatingAStandardTopic_whenCreatingATopic_shouldReturnAStandardTopic()
  - ✔ updatingATopicDescription_whenUpdatingATopic_shouldUpdateItsDescription()
  - ✔ deletingATopicOfferedToAStudent_whenDeletingATopic_shouldCallEmailSendOneTime()
  - ✔ deletingATopicWithTwoCandidates_whenDeletingATopic_shouldCallEmailSendTwoTime()
  - ✔ deletingATopicWithAStudent_whenDeletingATopic_shouldCallEmailSendOneTime()
  - ✔ acceptingACandidate_whenTheTopicHasTwoCandidates_shouldCallEmailSendTwoTimesAndSetTheStudentToTheTopic()
  - ✔ givingANameThatAlreadyExists_whenCreatingATopic_shouldThrowException()
- ✔ RegistrationServiceTest
  - ✔ givingSamePasswords_whenRegisteringNewUser_shouldCallEmailSendOneTime()
  - ✔ givingDifferentPasswords_whenRegisteringNewUser_shouldThrowException()
- ✔ SessionServiceTest
  - ✔ schedulingInterviewSession_whenInterviewIsAlreadyScheduled_shouldThrowException()
  - ✔ schedulingAnExaminationSession_whenEverythingIsOk_shouldCallSendEmailThreeTimes()
  - ✔ cancelingAnInterviewSession_shouldCallEmailSendAndDeleteByIdOneTimeEach()
  - ✔ cancelingAMeetingSessionWithTwoParticipants_shouldCallEmailSendTwoTimesAndDeleteByIdOneTime()
  - ✔ schedulingAnExaminationSession_whenExaminationIsAlreadyScheduled_shouldThrowException()
  - ✔ schedulingInterviewSession_whenEverythingIsOk_shouldCallSendEmailOneTime()
  - ✔ schedulingAMeetingSessionWithTwoStudents_whenEverythingIsOk_shouldCallSendEmailTwoTimes()
  - ✔ cancelingAnExaminationSession_shouldCallEmailSendTreeTimesAndDeleteByIdOneTime()
- ✔ UserServiceTest
  - ✔ givingAnInvalidEmail_whenAskingToResetPassword_shouldThrowException()
  - ✔ givingAnEmailThatHasBeenAlreadyGivenButNotVerifiedYet_whenSigningUpANewUser_shouldThrowException()
  - ✔ givingAValidVerifiedEmail_whenAskingToResetPassword_shouldUseEmailSendOneTime()
  - ✔ givingAnEmailThatHasBeenAlreadyGivenButExpirationHasPassed_whenSigningUpANewUser_shouldNotThrowException()
  - ✔ givingANewEmail_whenSigningUpANewUser_shouldCallSave()
  - ✔ givingAnEmailThatHasBeenAlreadyGivenAndVerified_whenSigningUpANewUser_shouldThrowException()
  - ✔ givingAValidNotVerifiedEmail_whenAskingToResetPassword_shouldThrowException()

Figure 15 Service tests' successfull execution

5.2.3 Controller Tests

As we established so far, the Controller layer is where the front end and the back end of DiploMate communicate. The methods that are contained in the Controller classes, perform a task requested by the user, by calling the corresponding method from a Service class, and return / redirect the user to the correct HTML page based on if the action was completed or not. We have already checked that the Service methods

correctly perform the tasks or throw a exceptions, based on the abovementioned tests. So this time, we called the methods contained in the Controller classes giving emphasis on these classes that could have more than one outcomes based on the task performed. We "mocked" the Service methods that they call, and made them to either complete the task successfully, or to throw an exception. We finally checked that the Controller methods returned / redirected the user to the correct HTML page. We summarize the details in the Table 17. In Figure 16 we also see the successful execution of these tests.

Controller	Test Class	Tested Methods	Use Case
RegistrationController	RegistrationControllerTest	showRegistrationForm	UC1
LoginLogoutController	LoginLogoutControllerTest	redirectToHomePage	UC2
ForgotPasswordController	ForgotPasswordControllerTest	showForgotPasswordForm, processForgotPasswordForm, processResetPasswordForm	UC4
UserController	UserControllerTest	processUpdatePasswordForm	UC10
AdminController	AdminControllerTest	processUpdatePasswordForm, processUpdateCoursesForm, processUpdateTagsForm	UC6, UC7, UC10
ProfessorController	ProfessorControllerTest	processProfessorUpdateInfoForm, processAddTopicForm, processUpdateTopicForm, processScheduleInterviewSessionForm, processScheduleMeetingSessionForm, processRescheduleSessionForm	UC9, UC12, UC13, UC16, UC18, UC39, UC41
StudentController	StudentControllerTest	processStudentUpdateInfoForm,	UC9, UC31

		showOfferedTopicPage	

Table 17 Controller tests details

- ✔ ForgotPasswordControllerTest
  - ✔ providingInvalidToken_whenResettingPassword_shouldRedirectToLoginPage()
  - ✔ accessingForgotPasswordPage_shouldDisplayForgotPasswordPageAtGetAndRedirectToItAtPost()
  - ✔ providingValidToken_whenResettingPassword_shouldRedirectToLoginPage()
  - ✔ providingValidTokenAndWrongPasswords_whenResettingPassword_shouldRedirectToResetPasswordPage()
- ✔ UserControllerTest
  - ✔ providingValidParams_whenUpdatingProfessorPassword_shouldRedirectToProfessorProfilePage()
  - ✔ providingValidParams_whenUpdatingStudentPassword_shouldRedirectToStudentProfilePage()
  - ✔ providingInvalidParams_whenUpdatingStudentPassword_shouldRedirectToUpdatePasswordPage()
  - ✔ providingInvalidParams_whenUpdatingProfessorPassword_shouldRedirectToUpdatePasswordPage()
- ✔ ProfessorControllerTest
  - ✔ providingInvalidParams_whenUpdatingSession_shouldRedirectToUpdateSessionPage()
  - ✔ providingInvalidParams_whenUpdatingProfileInfo_shouldRedirectToUpdateProfileInfoPage()
  - ✔ providingValidParams_whenUpdatingSession_shouldRedirectToCalendarPage()
  - ✔ providingValidParams_whenUpdatingProfileInfo_shouldRedirectToProfilePage()
  - ✔ providingInvalidParams_whenSchedulingMeeting_shouldRedirectToScheduleMeetingPage()
  - ✔ providingValidParams_whenAddingTopic_shouldRedirectToTopicsPage()
  - ✔ providingValidParams_whenSchedulingInterview_shouldRedirectToTopicCandidatesPage()
  - ✔ providingValidParams_whenSchedulingMeeting_shouldRedirectToCalendarPage()
  - ✔ providingInvalidParams_whenUpdatingTopic_shouldRedirectToUpdateTopicPage()
  - ✔ providingValidParams_whenUpdatingTopic_shouldRedirectToTopicsPage()
  - ✔ providingInvalidParams_whenSchedulingInterview_shouldRedirectToScheduleInterviewPage()
- ✔ AdminControllerTest
  - ✔ providingValidParams_whenUpdatingAdminPassword_shouldRedirectToAdminHome()
  - ✔ providingInvalidList_whenUpdatingTagsList_shouldReturnUpdateTagsPage()
  - ✔ providingValidList_whenUpdatingCoursesList_shouldRedirectToAdminHome()
  - ✔ providingValidList_whenUpdatingTagsList_shouldRedirectToAdminHome()
  - ✔ providingInvalidParams_whenUpdatingAdminPassword_shouldRedirectToUpdatePasswordPage()
  - ✔ providingInvalidList_whenUpdatingCoursesList_shouldReturnUpdateCoursesPage()
- ✔ LoginLogoutControllerTest
  - ✔ redirectToHomePage_whenUserRoleIsAdmin_shouldRedirectToAdminHomePage()
  - ✔ redirectToHomePage_whenUserRoleIsProfessor_shouldRedirectToProfessorProfilePage()
  - ✔ redirectToHomePage_whenUserRoleIsStudent_shouldRedirectToStudentProfilePage()
- ✔ RegistrationControllerTest
  - ✔ goingToRegistrationPageLink_whenWantingToRegisterForFirstTime_shouldReturnRegisterPage()
  - ✔ registeringUser_whenUserIsValidAndDoesNotAlreadyExists_shouldRedirectToLoginPage()
  - ✔ goingToRegistrationPageLinkUsingTheBackButton_whenAlreadyMadeRegistrationRequest_shouldRedirectToLoginPage()
  - ✔ registeringUser_whenUserIsNotValidOrAlreadyExists_shouldReturnRegistrationPage()
- ✔ StudentControllerTest
  - ✔ providingInvalidParams_whenUpdatingProfileInfo_shouldRedirectToUpdateProfileInfoPage()
  - ✔ providingValidParams_whenUpdatingProfileInfo_shouldRedirectToProfilePage()
  - ✔ askingForTopicOffer_whenThereIsNoOpenTopicAvailable_shouldRedirectToTopicPage()
  - ✔ askingForTopicOffer_whenThereIsOpenTopicAvailable_shouldReturnTopicDetailsPage()

Figure 16 Controller tests' successfull execution

Chapter 6. User Interface

In this chapter we present some pictures of DiploMate's user interface.

6.1 Login – Register – Forgot Password Pages

Through DiploMate's login page, a user can either login, or access the registration or forgot password pages. Figure 17 shows these pages.

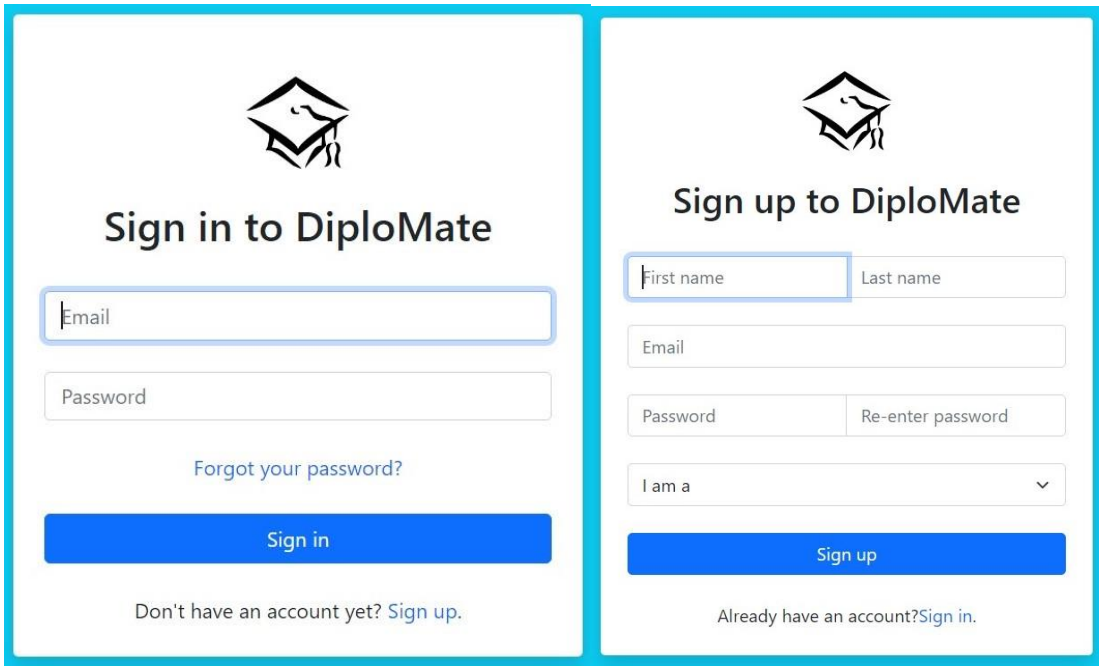

Figure 17 Login and registration pages

If a user forgot his / her password, he / she can access the forgot password page. The user must provide a valid email address, and DiploMate will redirect him / her to the reset password page. Figure 18 shows both pages.

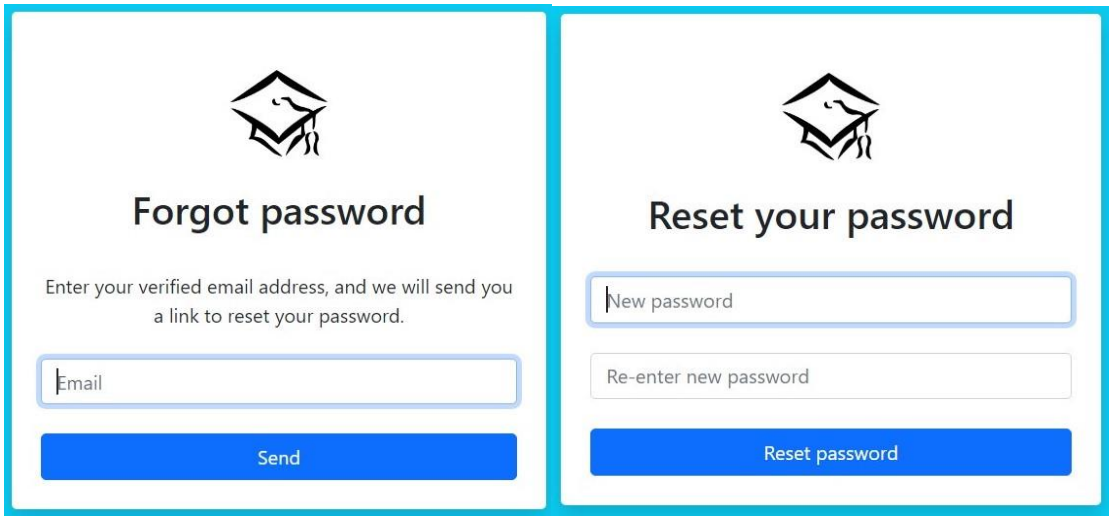

Figure 18 Forgot and reset password pages

6.2 Admin Pages

After the admin logs in to DiploMate, he / she sees the Administrator home page as shown in Figure 19.

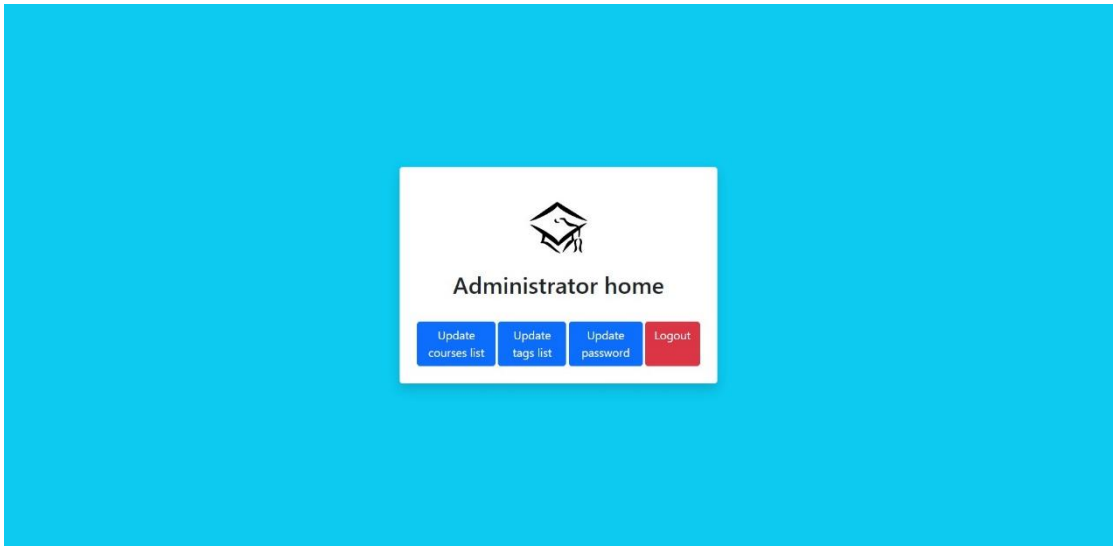

Figure 19 Administrator home page

If the administrator selects to update either the tags or the courses list, DiploMate redirects him / her to the corresponding update pages shown in Figure 20.

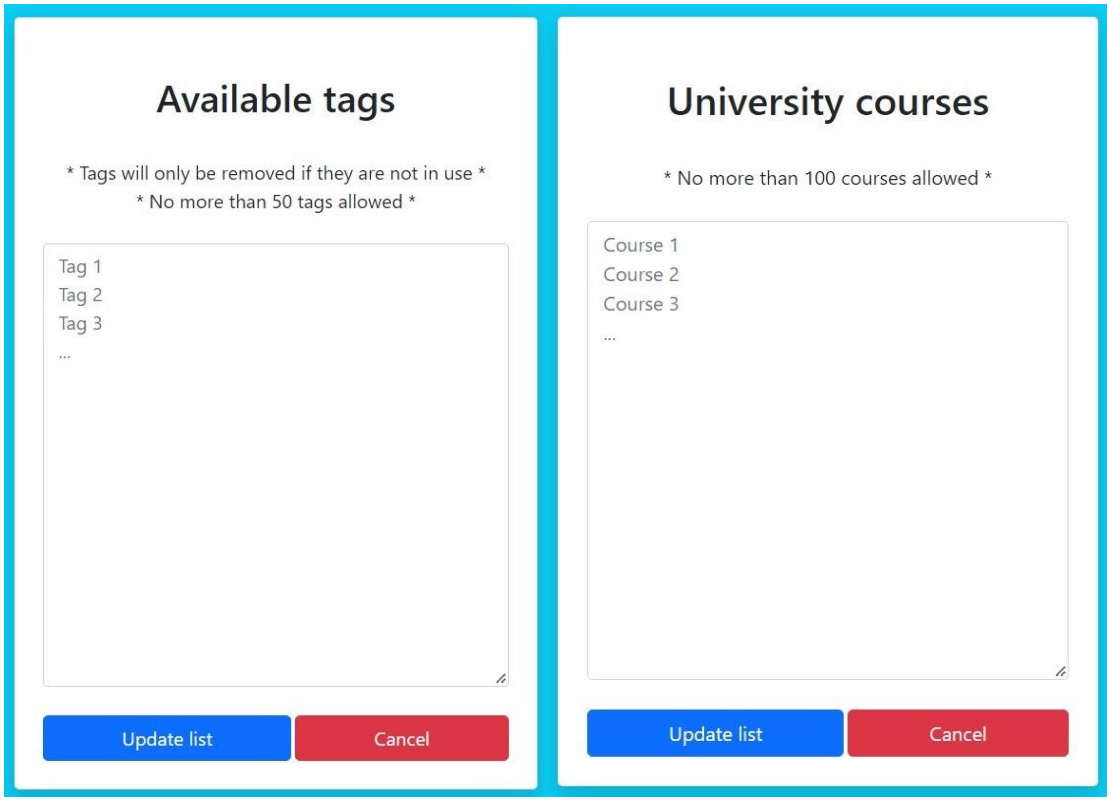

Figure 20 Update tags and courses list pages

6.3 Profile Pages

After a user logs in to DiploMate, he / she sees the profile page. This page differentiates based on the user's role. Figures 21 and 22 show these pages.

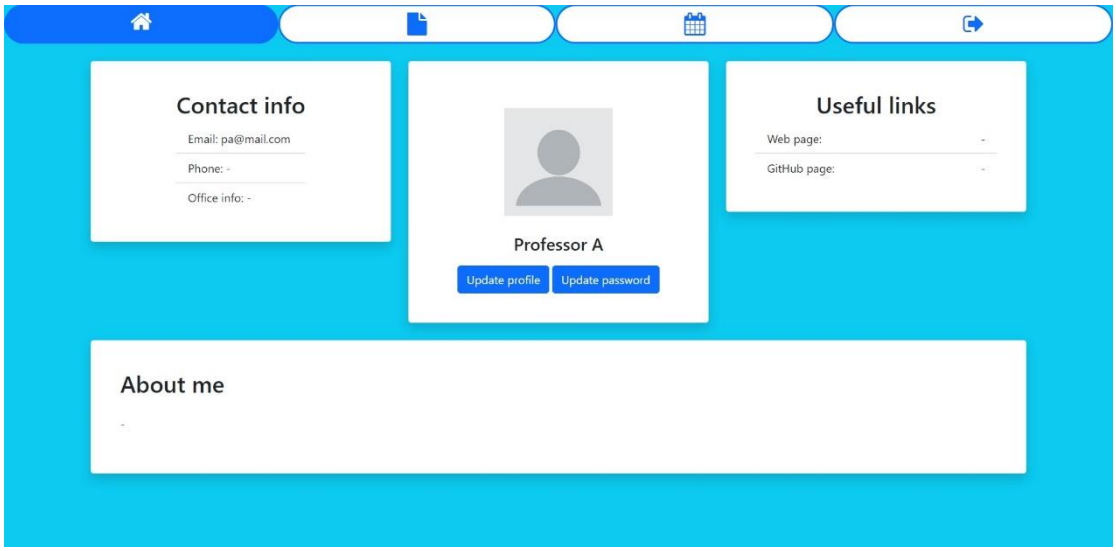

Figure 21 Professor's profile page

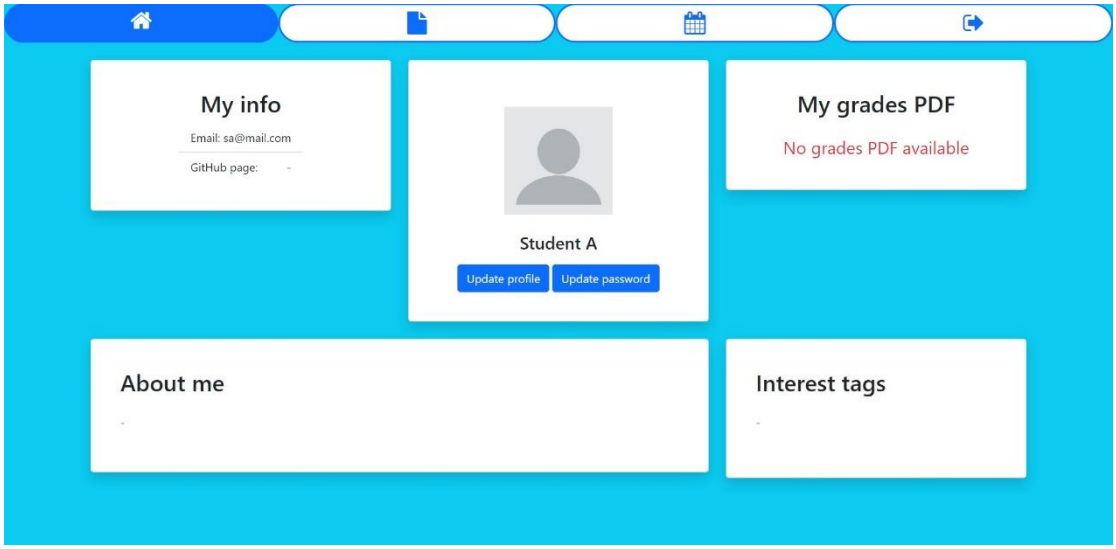

Figure 22 Student's profile page

It is worth mentioning that if the student has provided a PDF with his / her grades, DiploMate changes the top right corner of Figure 22 allowing both the student and every professor that visits his / her profile, to download the PDF. Figure 23 shows this change.

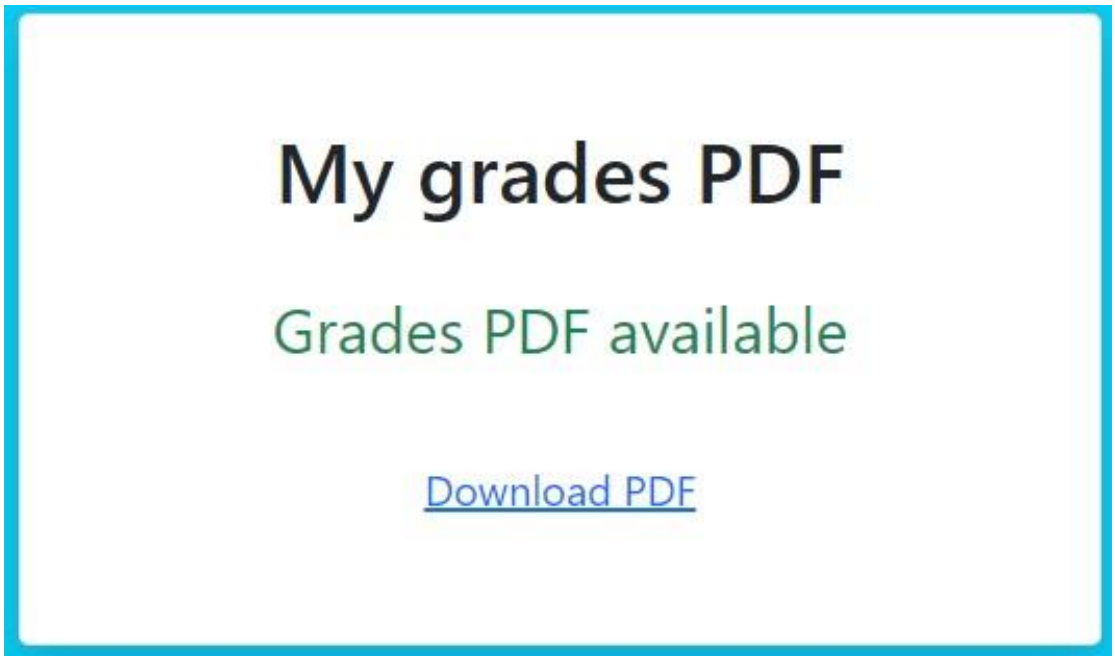

Figure 23 Available grades PDF on student's profile

When a user selects to update his / her profile information, DiploMate redirects him / her to the update information page, based on the user's role. Figures 24 and 25 show these pages.

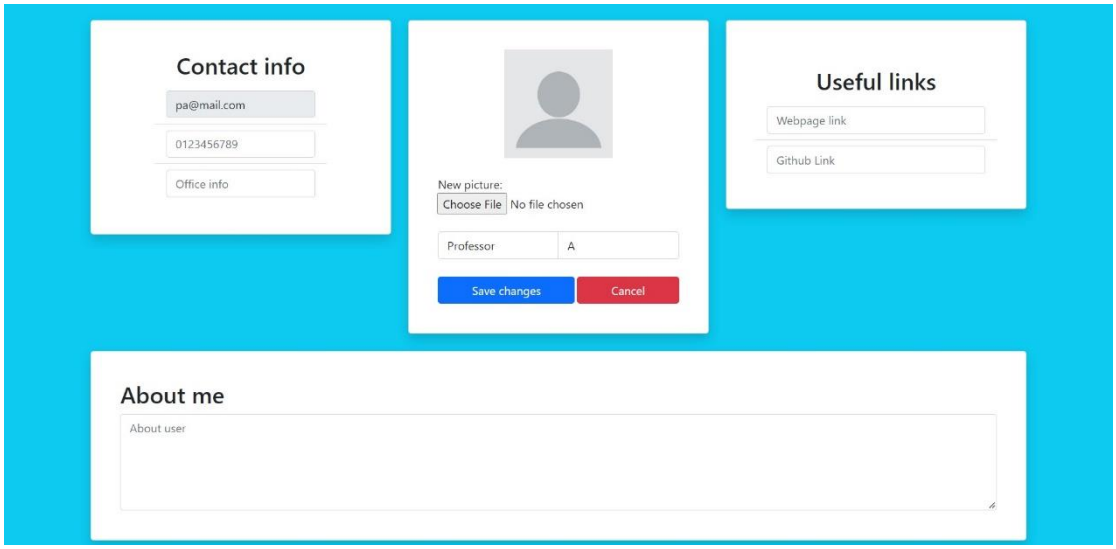

Figure 24 Professor's update information page

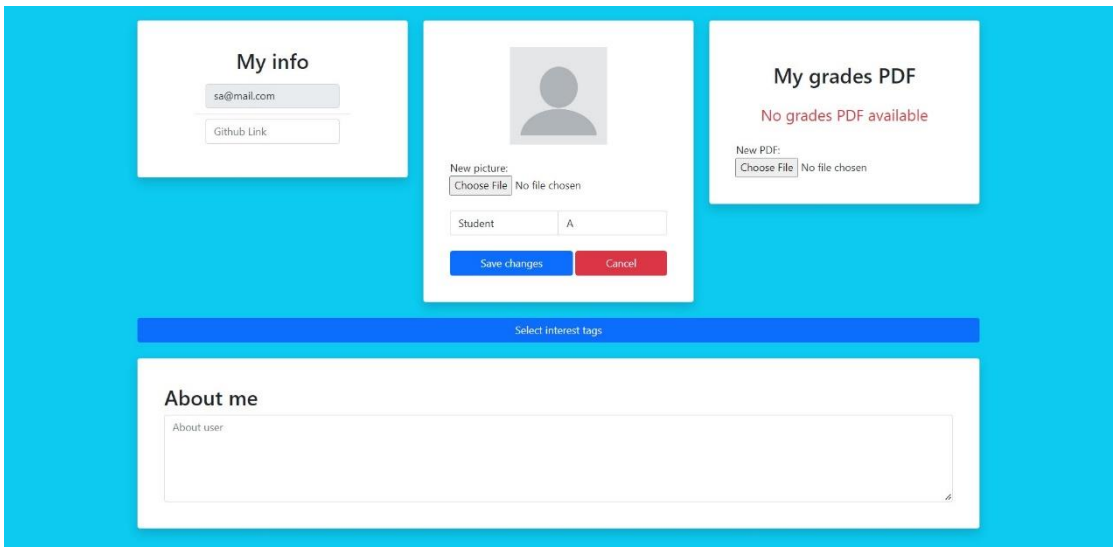

Figure 25 Student's update information page

Here the student has an extra option to update his / her interest tags by clicking on the corresponding button. This button displays the available tags for the student to select as shown in Figure 26.

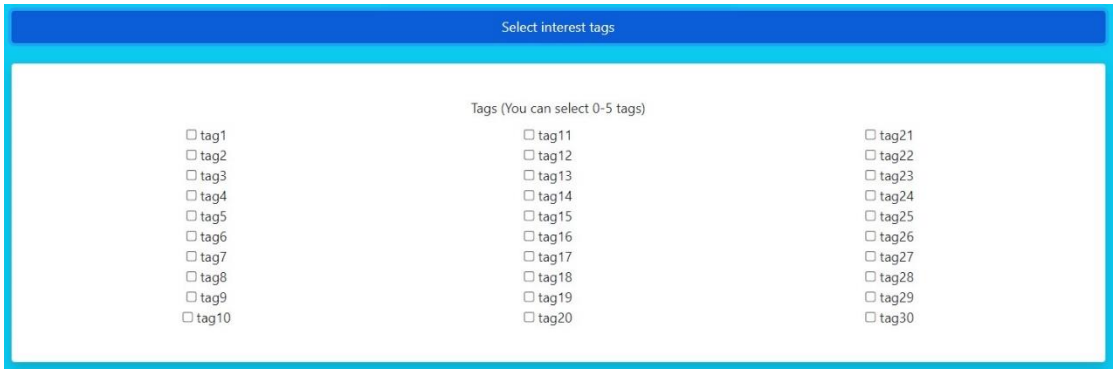

Figure 26 Student's update information available tags

Finally, if a user wants to update his / her password, DiploMate displays the corresponding page shown in Figure 27.

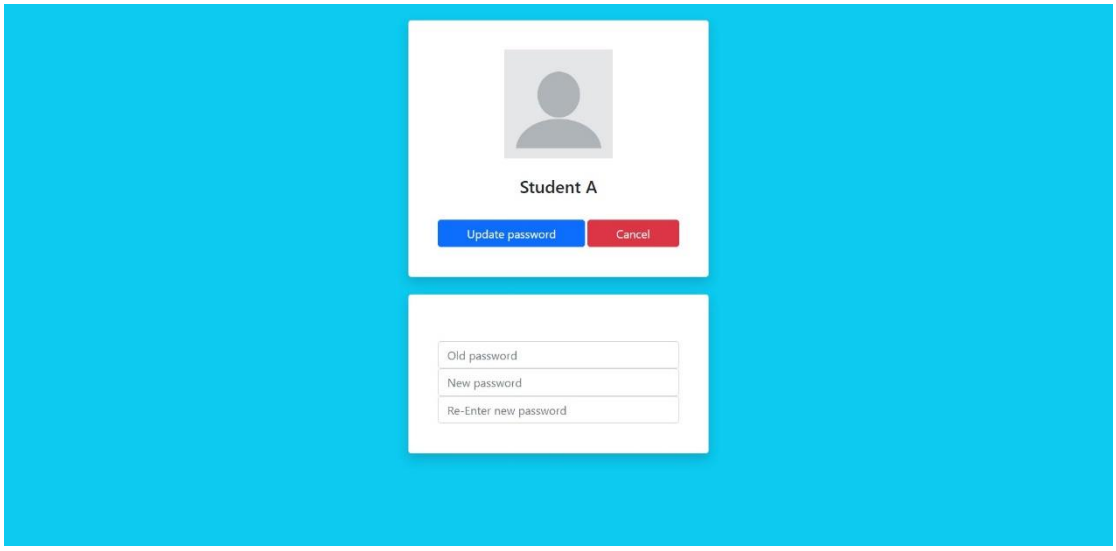

Figure 27 Update password page

6.4 Topic Pages

For better understanding the topic-related pages of DiploMate, we have to divide them based on the user's role.

6.4.1 Professor Topic Pages

We see a general overview of a professor's topic page in Figure 28. In it we can see an available standard and open topic, as well as a taken topic.

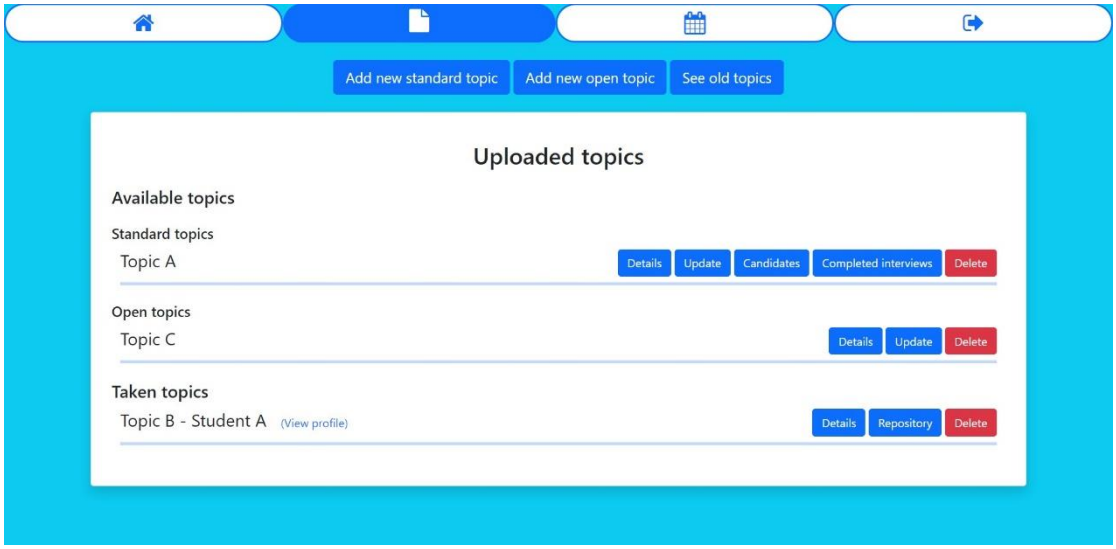

Figure 28 Professor's topic page

A professor can add a standard or an open topic, by clicking the corresponding button. Then DiploMate displays the corresponding page shown in Figures 29 and 30.

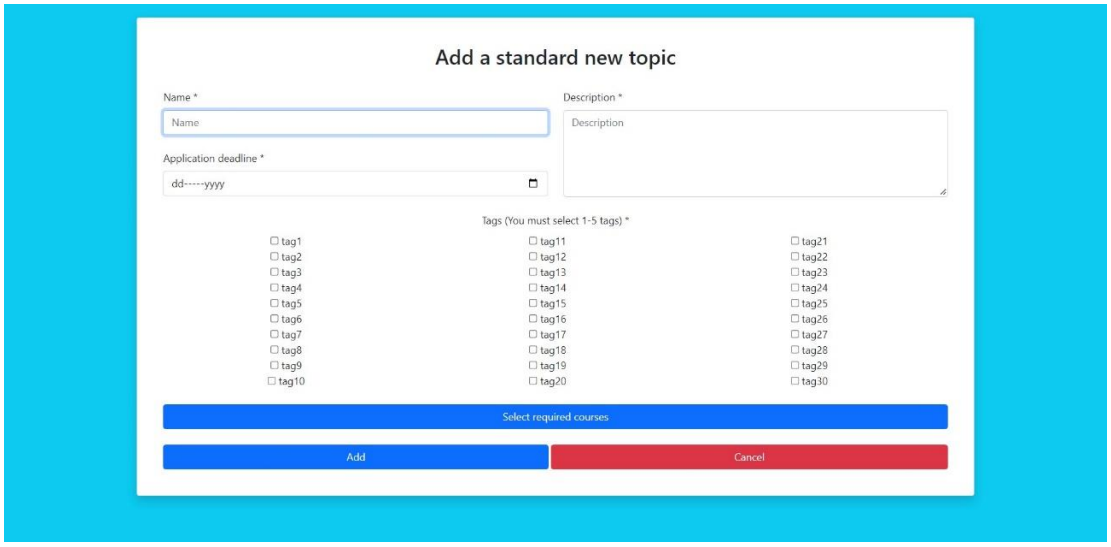

Figure 29 Add standard topic page

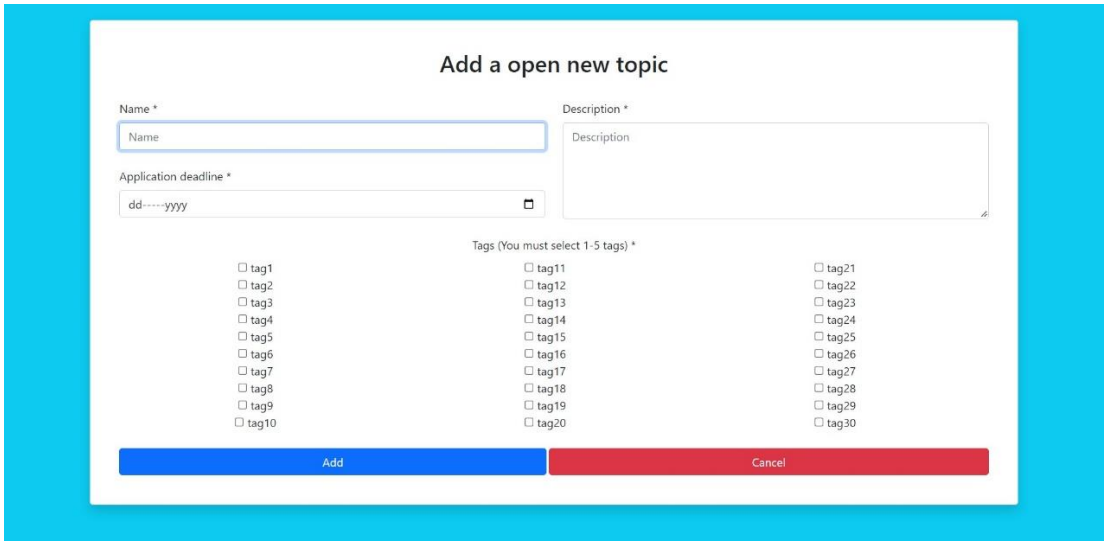

Figure 30 Add open topic page

It is worth mentioning that in the case of a standard topic's addition, DiploMate allows the professor to set a minimum required score to some courses. If a professor clicks on the corresponding button, DiploMate displays the available courses with a number field as shown in Figure 31.

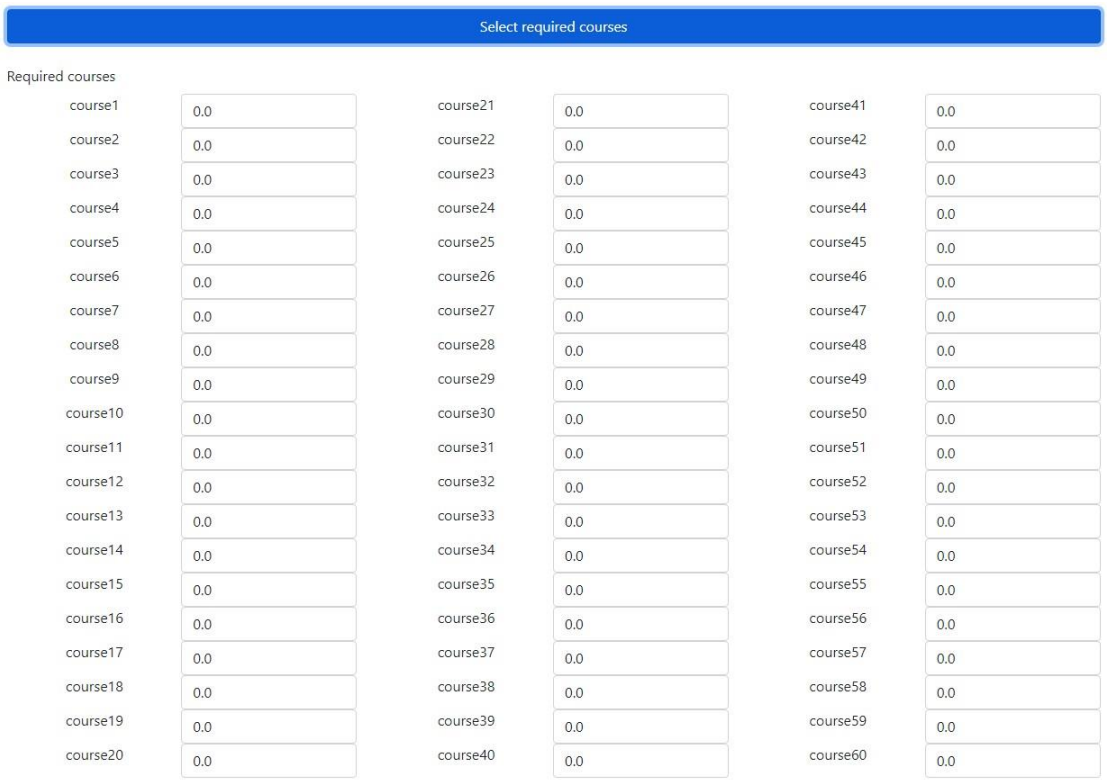

Figure 31 Available courses to set minimum required score for

A professor can see a standard topic's candidate and the upcoming interviews he / she has scheduled with some of them. Figure 32 shows a topic's candidates page.

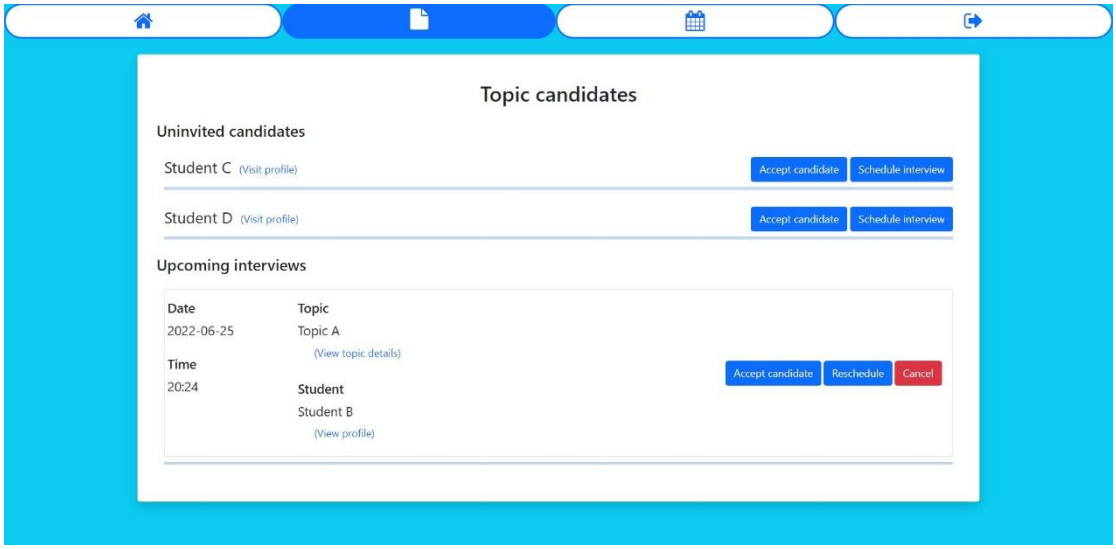

Figure 32 Topic's candidates list

While the professor can also schedule a new interview and see / score the completed ones by accessing the pages shown in Figure 33.

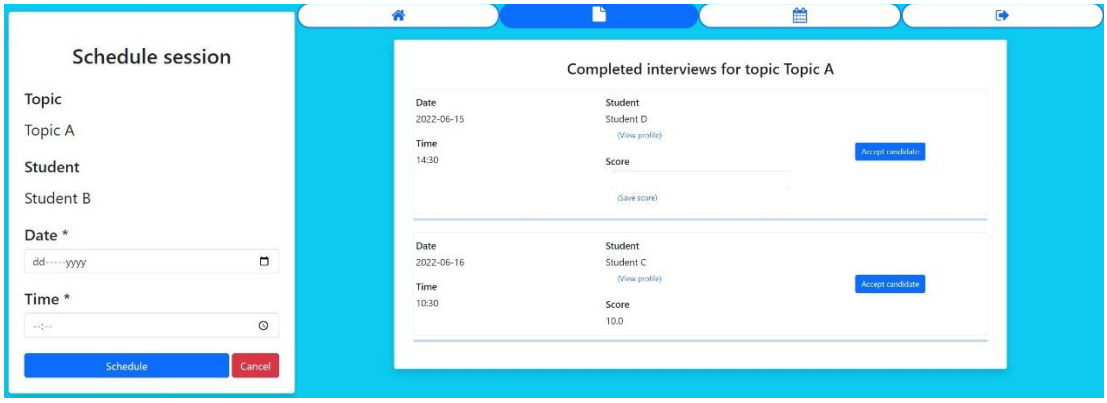

Figure 33 Schedule interview and completed interviews pages

6.4.2 Student Topic Pages

Figure 34 shows a general overview of a student's topics page. In it we can see that the student has also asked DiploMate to offer him / her a topic, and the student has now a deadline to accept it or not.

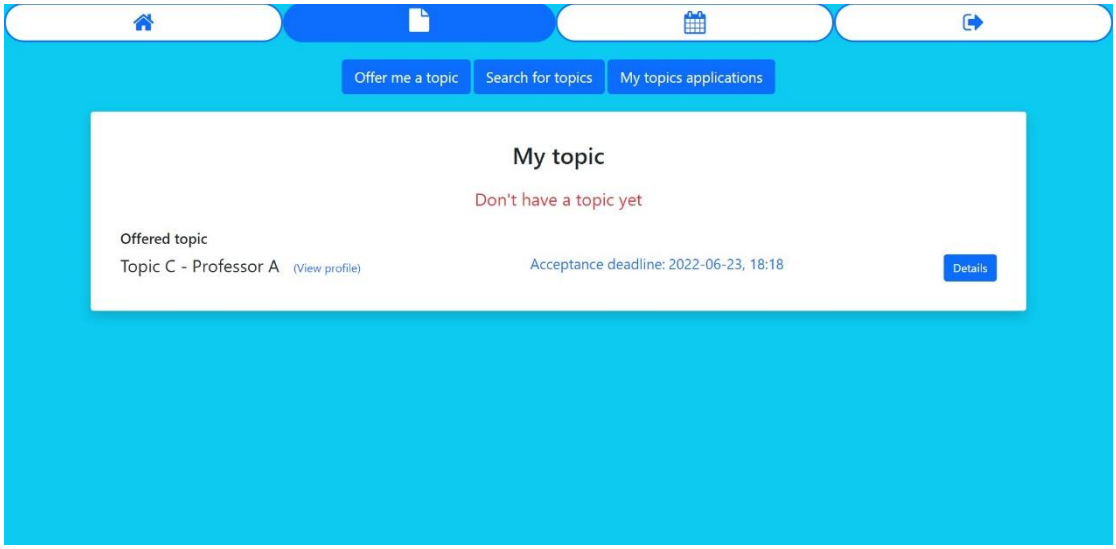

Figure 34 Student's topics page with an offered topic

A student can also search for some topics by accessing the search page shown in Figure 35. And DiploMate will display the results in the page show in Figure 36.

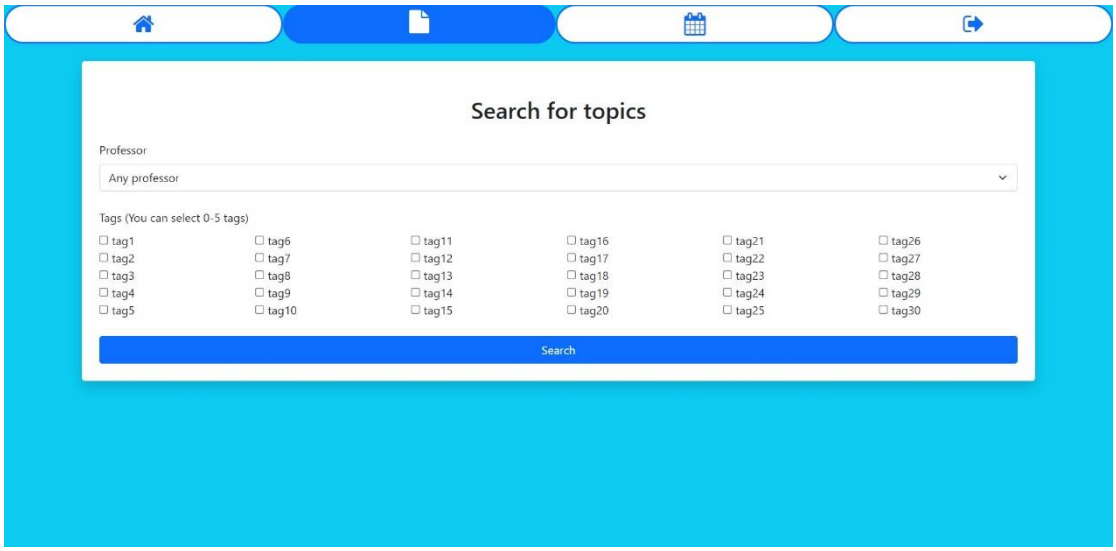

Figure 35 Search topics page

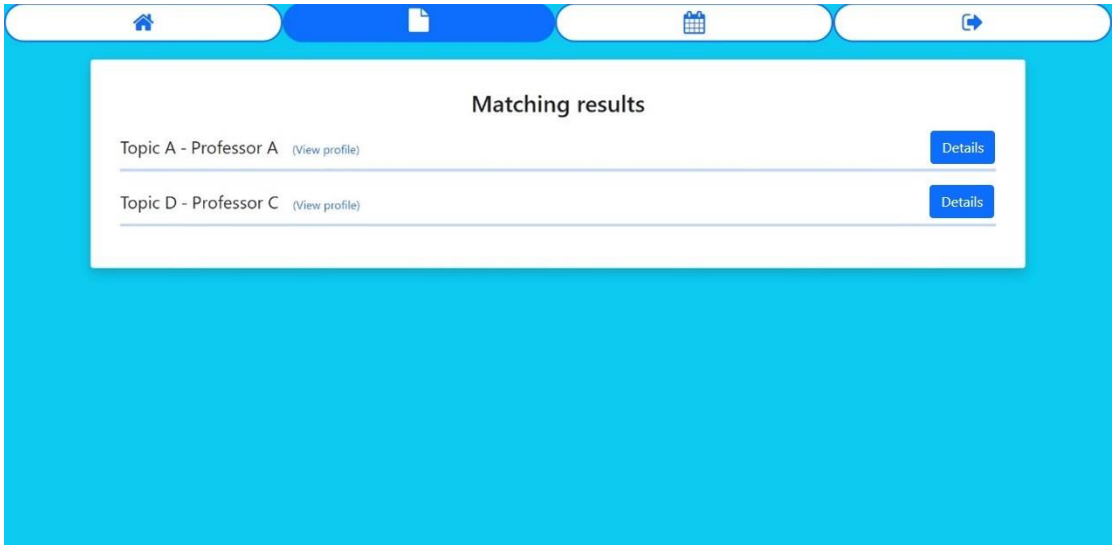

Figure 36 Search topics results

If a student asks DiploMate to offer him / her a topic, and there is one available, DiploMate displays the topic's details as shown in Figure 37.

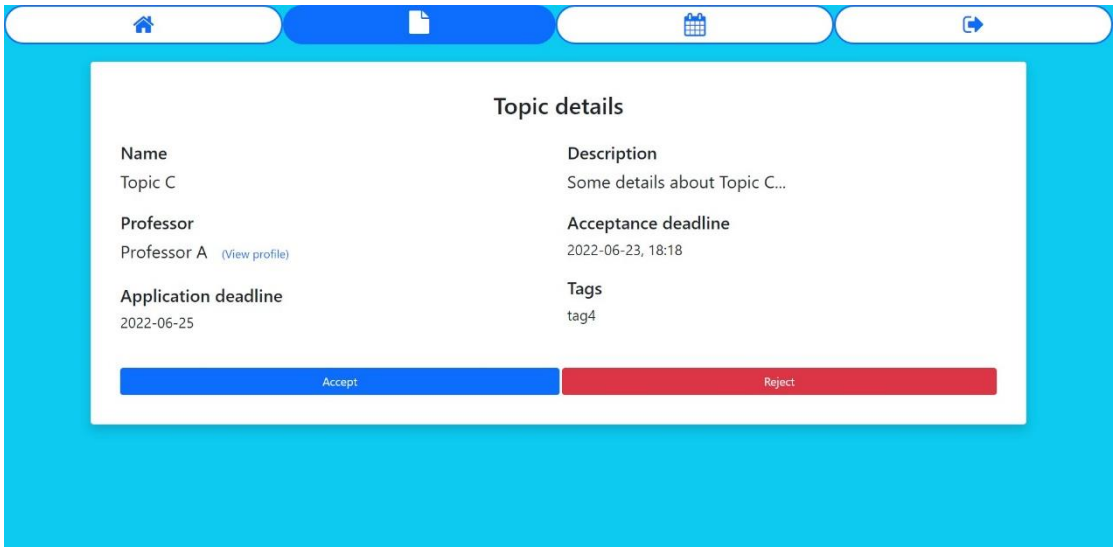

Figure 37 Offered topic's details

A student can see his / her applications to topics and cancel some of them if he / she wants to. Figure 38 shows this applications page.

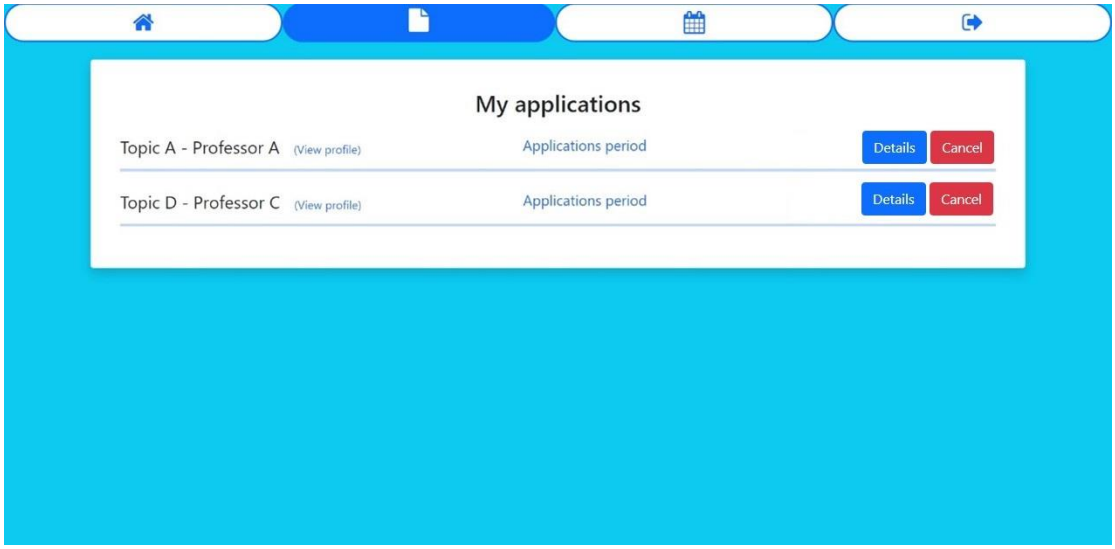

Figure 38 A student's topic applications page

After a student takes over a topic, his / her topics page looks like Figure 39.

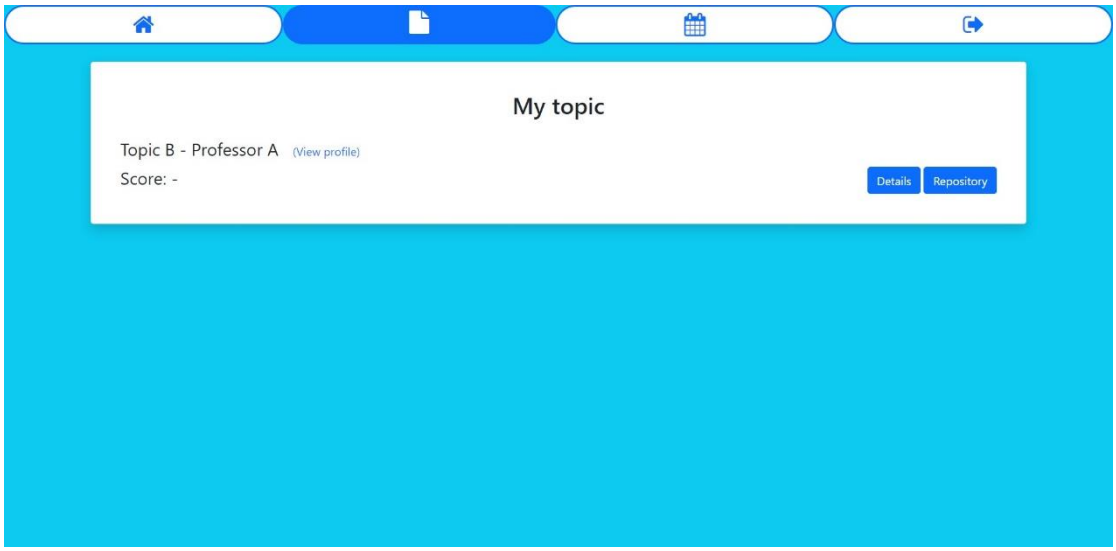

Figure 39 A student's topic page after taking over a topic

Finally, a student and a professor can access a topic's repository that is shown in Figure 40. The user can download all the files but can delete only those that he / she has added (in this case the student).

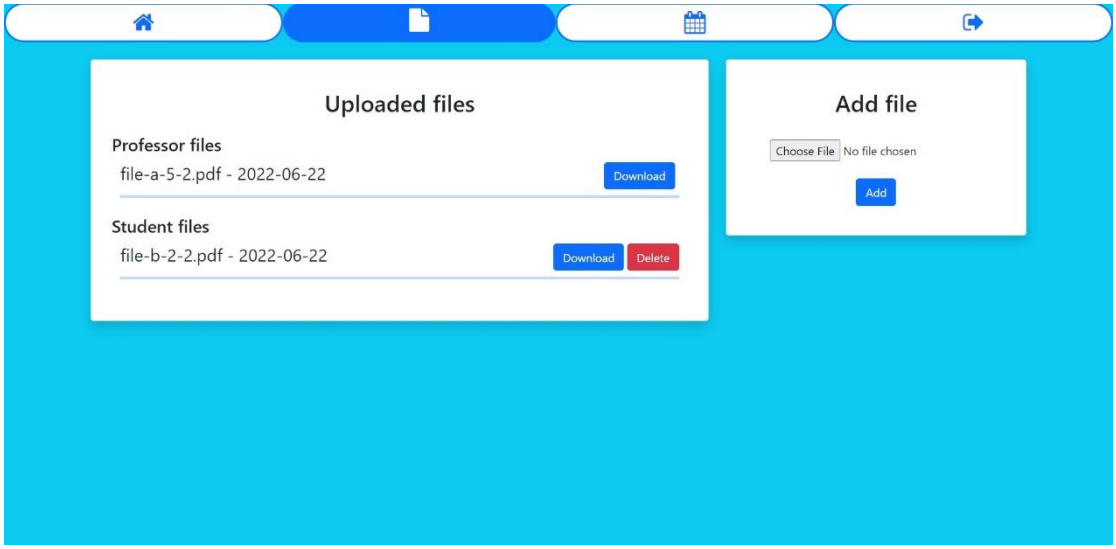

Figure 40 Topic's repository page

6.5 Calendar Pages

Figures 41 and 42 show a general overview of a professor's calendar page. In them we see that the professor just scheduled an examination ("Examination scheduled successfully" message) and has an interview, a meeting and an examination scheduled. This page is very similar to a student's calendar page, excluding that the student's page has not the scheduling and unrated examinations buttons.

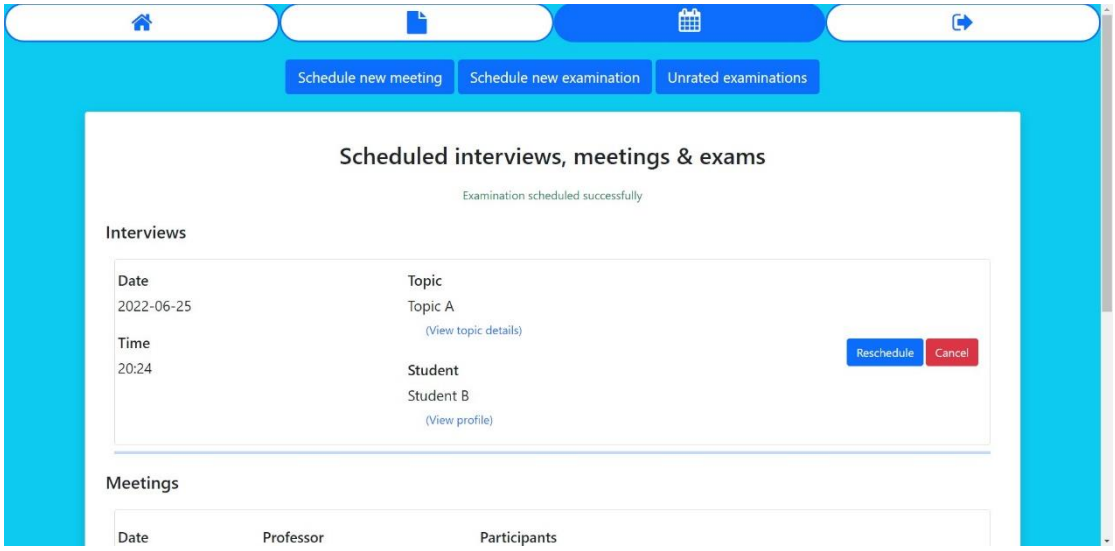

Figure 41 A professor's calendar page (1)

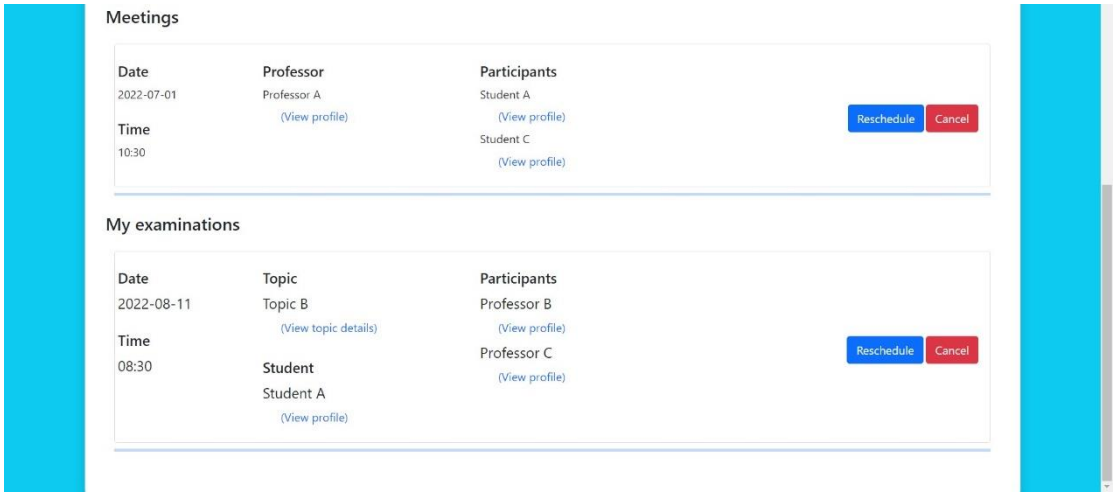

Figure 42 A professor's calendar page (2)

A professor can schedule a meeting or an examination by accessing the pages show in Figure 43.

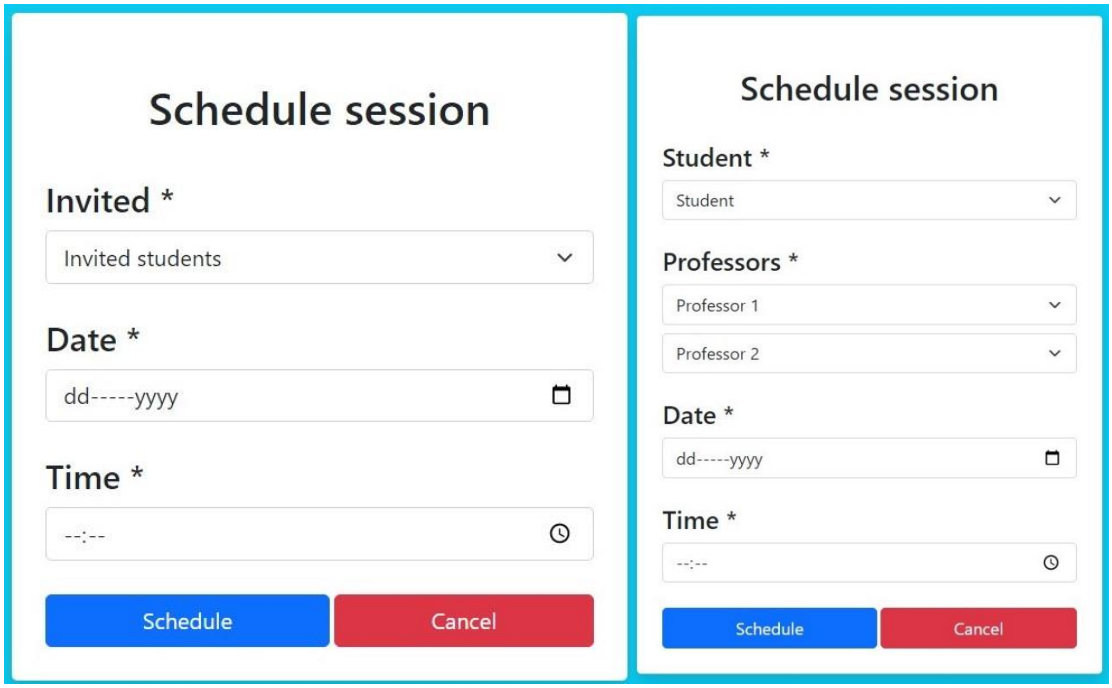

Figure 43 Schedule meeting and examination pages

Finally, a professor can see the unrated examinations he / she has participated in and score them. Figure 44 show the unrated examinations page.

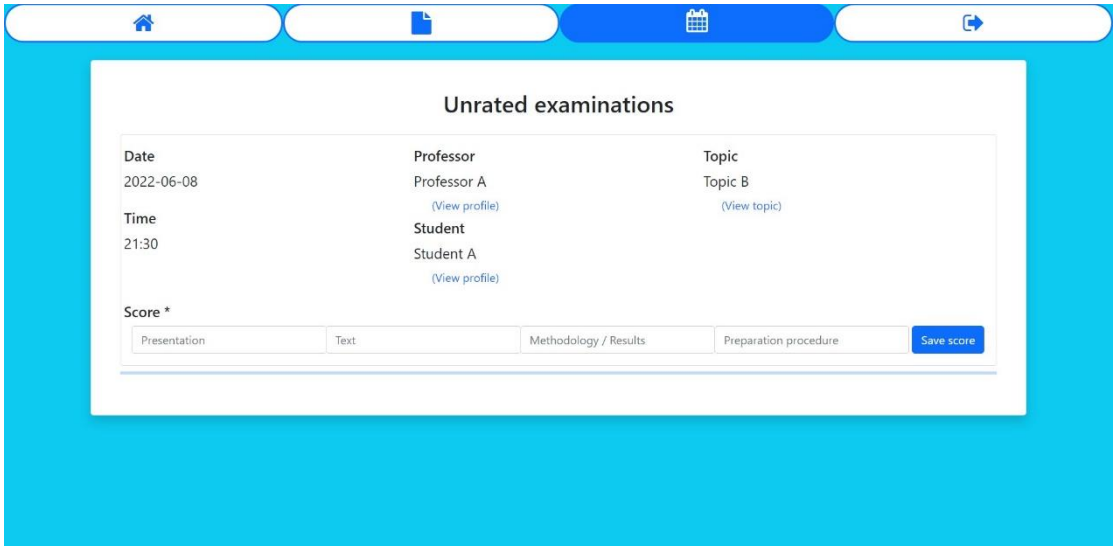

Figure 44 A professor's unrated examinations

Chapter 7. Evaluation

Following the development of DiploMate, we considered it necessary to be evaluated by some prospective users. To achieve this evaluation, we created a questionnaire with 15 questions. Due to the nature of DiploMate, it was difficult to be used widely by many people. For this, it would be required to setup a server and have some users pretending to be professors and students to satisfy our use case. Instead, we presented the application to a small group of 10 students. We showed them both the use of the application from a professor as well as from a student standpoint. Our goal is to extract information about the familiarity and usability of DiploMate's Interface, as well as the utility of its features. The questions of the questionnaire are based in the search of this information. Below, we present the questions, some graphs based on the evaluators' answers, and some comments on them.

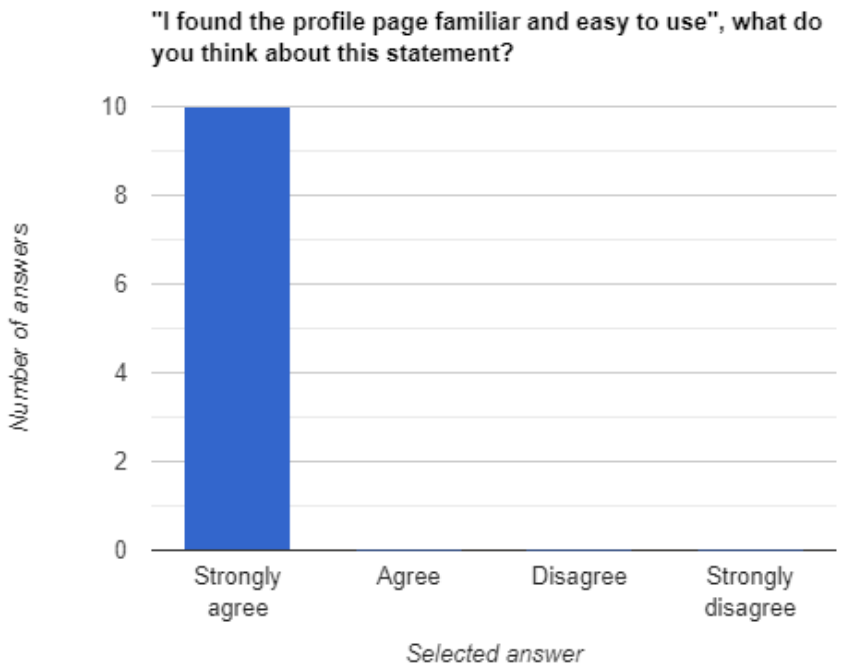

Figure 45 Profile page evaluation

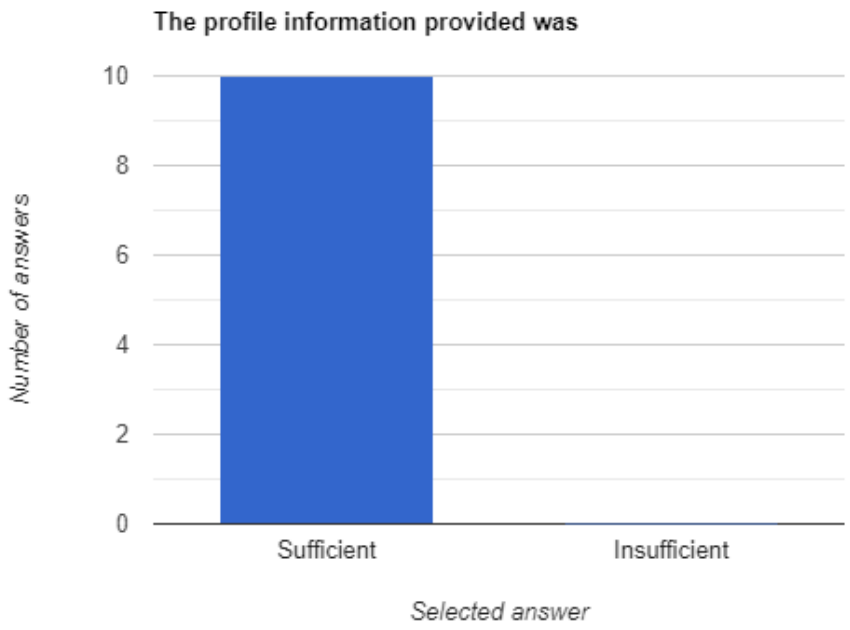

Figure 46 Profile information evaluation

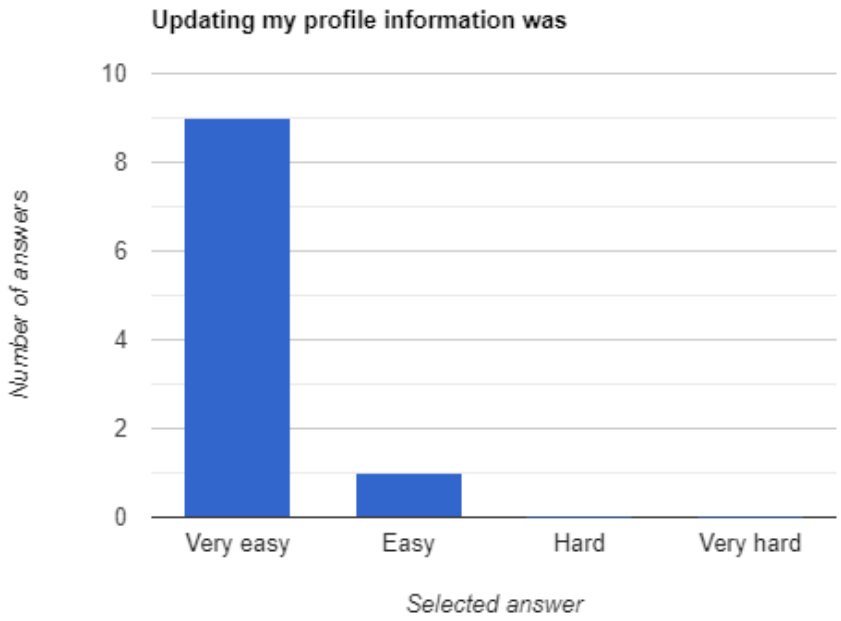

Figure 47 Updating profile evaluation

It is obvious from the Figures 45, 46 and 47, that the evaluators found DiploMate's profile page, to be very familiar, and therefore very easy to update as well. It is worth

mentioning that we also included a question, asking the evaluators that found the profile's information insufficient, to suggest some more information. As all evaluators found it to be sufficient, there were no answers in this question.

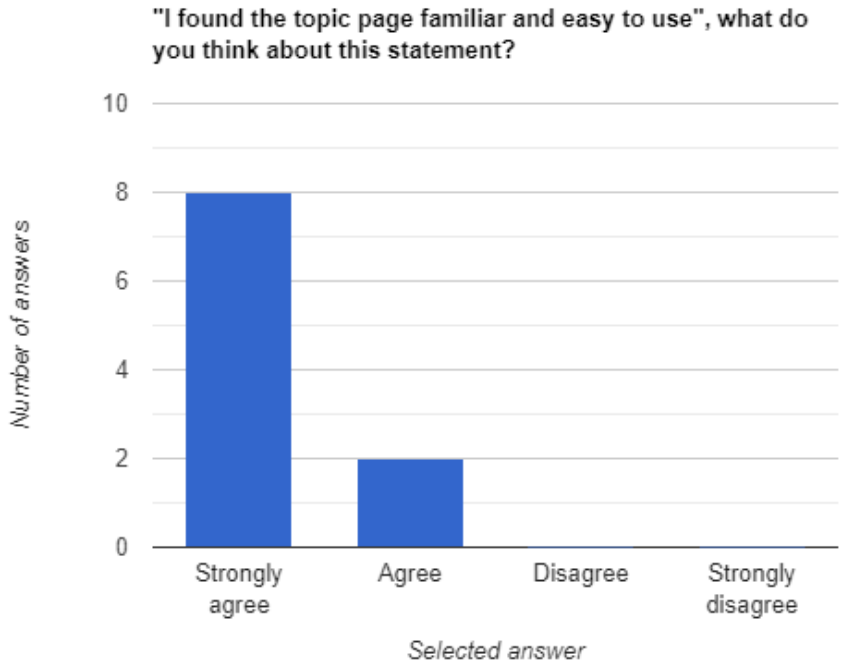

Figure 48 Topic page evaluation

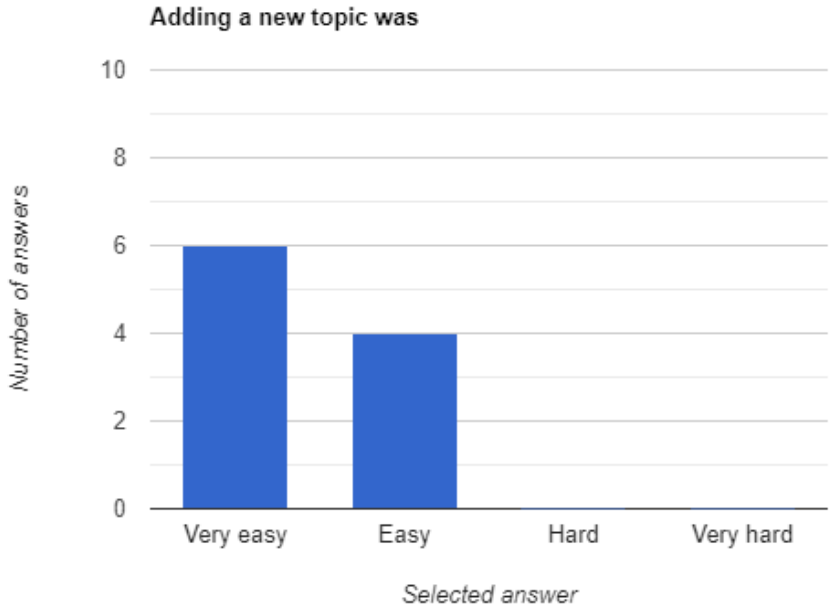

Figure 49 Adding a topic evaluation

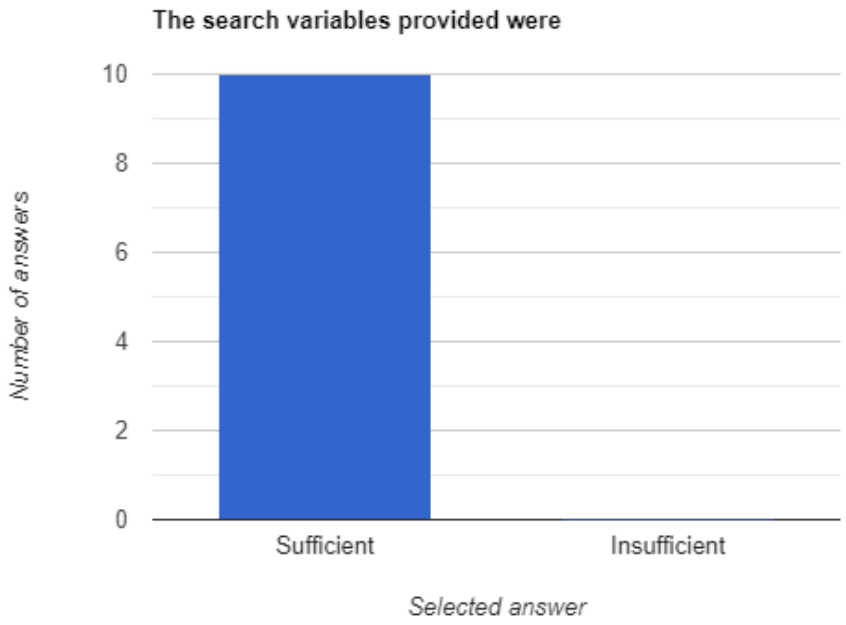

Figure 50 Searching topic evaluation

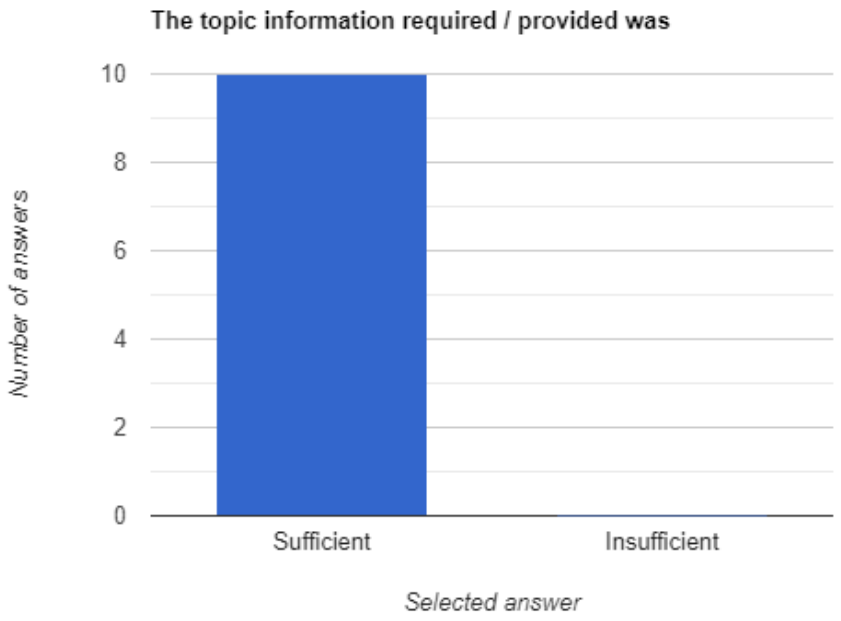

Figure 51 Topic information evaluation

From the Figures 48, 49, 50 and 51, we can see that while DiploMate's topic page may not be as familiar as its profile one, it is also easy to use, either to add or to search for topics. It is also worth mentioning that we included two questions, asking the evaluators

that found the search variable and the topic information insufficient, to suggest some more variables / information. As all evaluators found them to be sufficient, there were no answers in these questions.

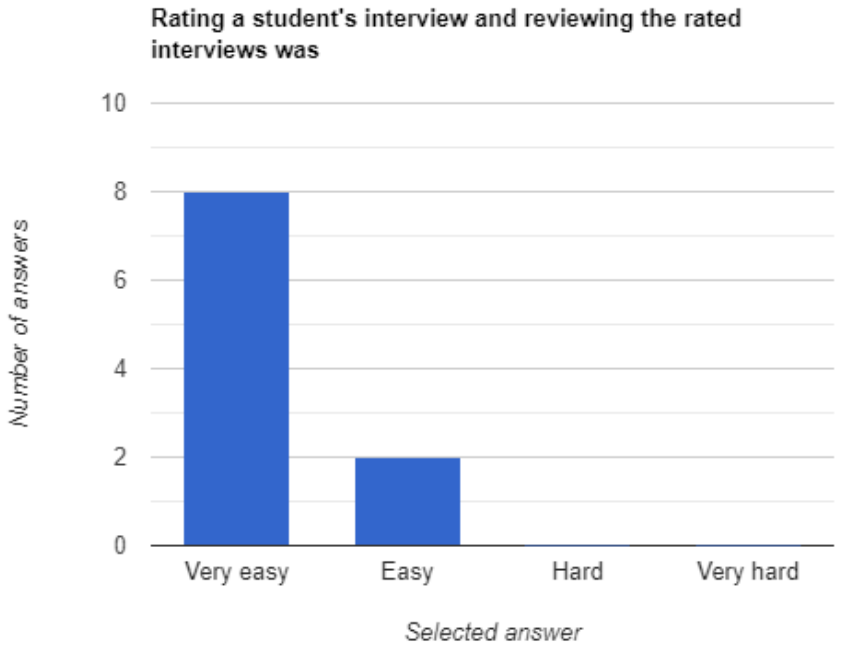

Figure 52 Rating a user's interview evaluation

Rating a student's interview was also found to be easy by our evaluators as shown in Figure 52.

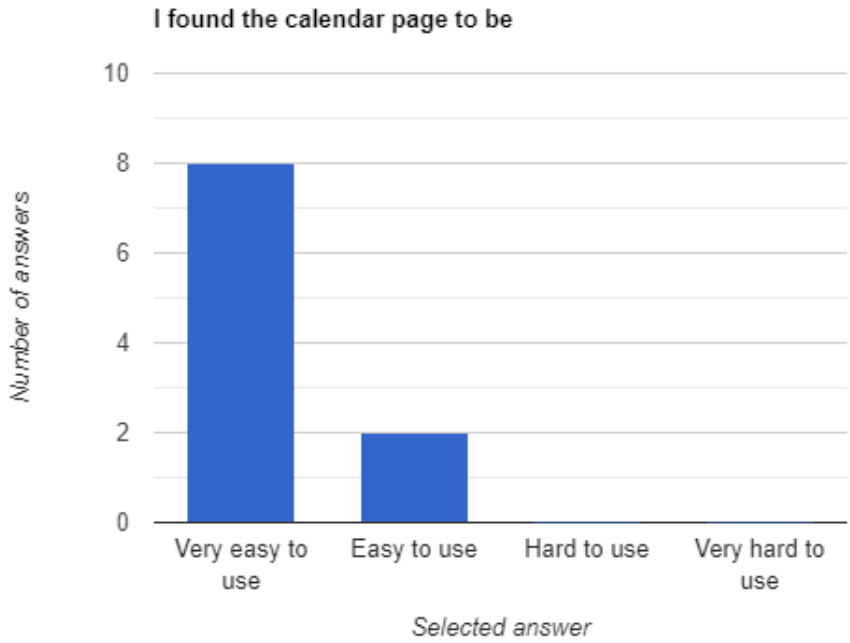

Figure 53 Calendar page evaluation

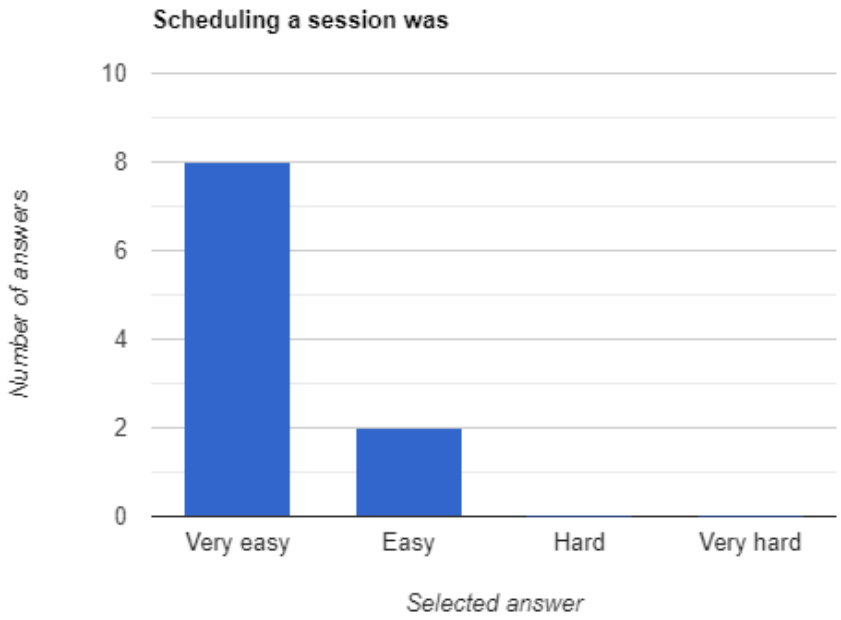

Figure 54 Scheduling a session evaluation

Moving on to the DiploMate's calendar page, we can also tell by Figure 53 and 54, that while it might be unfamiliar to some evaluators, it is still pretty easy to use to either see what is scheduled or to schedule a new session.

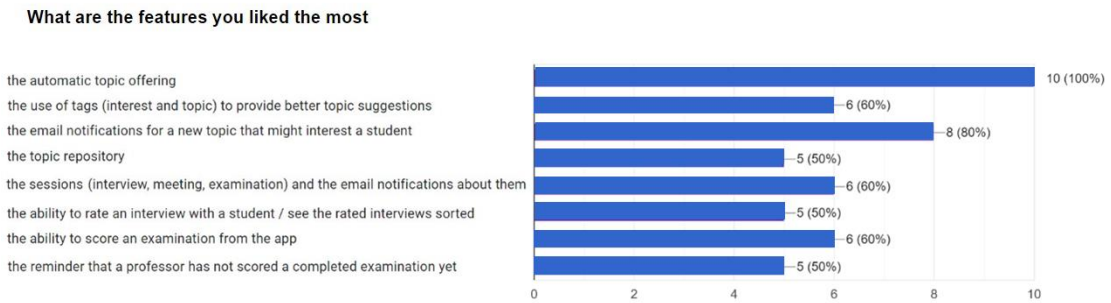

Figure 55 DiploMate's features evaluation

Finally, we asked the evaluators to select one or more features of DiploMate, they liked the most. We can clearly see in Figure 55, that the automatic topic offering is the most favored DiploMate's feature, having been selected by all 10 evaluators. Closely second comes the email notification for a topic that might interest the student. So, we can understand that DiploMate's features that help a student find a topic, are a really good reason to use such an application. We can also tell that DiploMate has no unnecessary features, as even the least selected features listed in this question, were liked by at least half of the evaluators. We should also mention that we included a question, asking the evaluators to suggest more features they would like to see in DiploMate. No such

features were provided by any of them, so we can probably say that DiploMate's features as is, cover a wide range of use cases.

Chapter 8. Summary and Future Work

8.1 Summary

Having developed DiploMate for some time, we can say that such an application is very interesting to work on. We have to mention how easy it is for anyone to use a web application such as DiploMate. We also cannot ignore the lack of such an application, and the positive feedback we got from the evaluators that answered our questionnaire. The problems we try to solve with DiploMate, are getting solved in a good-enough way, but there is always room for improvement. While the evaluators may not have suggested any features missing from DiploMate, we as developers, have some ideas for the future work of the project. These ideas were not included by default due to lack of time, or because they were conceived pretty late into DiploMate's development. We present them below.

8.2 Future Work

8.2.1 User's Feed

As of now, DiploMate redirects a user that logs in to the application, to his / her profile page. After this suggested modification, while the user's profile page will still be available, DiploMate will now redirect the users to a feed page. The feed page will contain all new notifications concerning the user. In this page the user will be able to remove a specific notification or remove all of them in one click. The user will now also have the option to select if he / she wants to get notifications via email, by checking or unchecking a checkbox in the update profile page.

8.2.2 Response to Session

In the current state of DiploMate, when a professor schedules a session including some other professors and / or students, DiploMate checks that all the participants are available, schedules the session and notifies them. While this solves the sessions overlapping problem in some extend, it does not exactly reflect in the real world. In the real world a user, while might not has some session scheduled in DiploMate, might still not be available at a certain date and / or time. This problem can potentially be solved by completing the scheduling of a session only when all participants have verified their availability for it.

Chapter 9. References

[1] https://openjdk.java.net/projects/jdk/11/

[2] https://spring.io/projects/spring-boot

[3] https://projectlombok.org/

[4] https://www.postgresql.org/

[5] https://html.spec.whatwg.org/

[6] https://www.thymeleaf.org/

[7] https://maven.apache.org/

[8] https://start.spring.io/

[9] https://en.wikipedia.org/wiki/Jaccard_index

[10] https://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93controller

[11] https://en.wikipedia.org/wiki/Multitier_architecture

[12] https://junit.org/junit5/

[13] https://site.mockito.org/