Dissertation Recommender System: Design & Development

Konstantina Vezirtzoglou
SID: 3306150014

SCHOOL OF SCIENCE & TECHNOLOGY
A thesis submitted for the degree of

Master of Science (MSc) in Mobile and Web Computing

DECEMBER 2017
THESSALONIKI – GREECE
Dissertation Recommender System: Design & Development

Konstantina Vezirtzoglou

SID: 3306150014

Supervisor: Dr. Merkourios Karaliopoulos
Supervising Committee Members: Prof. Georgios Evangelidis
Dr. Marios Gatzianas

SCHOOL OF SCIENCE & TECHNOLOGY
A thesis submitted for the degree of

Master of Science (MSc) in Mobile and Web Computing

DECEMBER 2017
THESSALONIKI – GREECE
Acknowledgements

I would like to express my deepest gratitude to my advisor, Dr. Merkouris Karaliopoulos, for his continuous support and guidance. I sincerely appreciate his valuable assistance and contribution to the successful completion of my dissertation and I would like to specifically thank him for inspiring me as my teacher and mentor, throughout my studies at the International Hellenic University.

My sincere thanks also go to Dr. Marios Gatzianas for his assistance with the collection of data that were used for the evaluation of my work. I highly appreciate his time and willingness to share information that was necessary for the implementation of my dissertation.

Finally, I wish to thank my family and close friends for their unending support and encouragement in everything I pursue.
Abstract

Dissertation topics are commonly undertaken by university students as a precondition for the completion of their studies. The way this process is coordinated in the International Hellenic University (IHU) has inspired the dissertation recommender system that is being proposed in this Thesis. The topic assignment to students is roughly a three-step process. First, staff with teaching assignments (referred to as *advisors* in the Thesis), local or adjunct, announce dissertation topics. Then, the students submit a ranked list of up to four dissertation topics they are interested in, while also being able to propose their own topics. Finally, a special committee in IHU centrally assigns topics to the students taking into account their preferences and additional resource constraints such as the resulting load per advisor.

In our Thesis, we have designed and developed a system that offers meaningful and useful recommendations to all involved parties in the dissertation assignment process (students, teaching staff, IHU committee) in order to automate it and make it faster and more efficient. Hence, the students get recommendations about topics lying closer to their interests, the *advisors* receive recommendations about students who are more appropriate for the topics they announce, and the committee gets recommendations about efficient assignments of students to topics.

Our system combines web and mobile app components. The website, implemented in HTML, CSS and JavaScript, is used by the advisors and the committee. The mobile app, developed in Swift for the Apple iOS platform is intended for use by the IHU students. Both components are supported by Google’s Firebase backend as a service (BaaS) platform. We tried to follow best practices in the design and development phase of the system such as a priori listing the requirements of the system, working with mockups in the design phase and following well established architectural patterns throughout the development process. From an algorithmic point of view, we have relied on standard recommendation techniques and elements of matching theory and properly customized them to derive the actual recommendations. We evaluated our system against anonymized data provided by IHU on this year’s dissertation topic assignments to find out that the system can derive assignments that satisfy more the students’ preferences than the current, manual, assignment process.
We were thus able to propose a fully-functional dissertation recommender system that is able to successfully serve as a consultant for students, assisting them in selecting dissertation topics that are fully aligned with their interests. Furthermore, our system is able to recommend the most suitable students for any topic that is being proposed by advisors and it is finally capable of facilitating the efficient assignment of dissertation topics to students by the involved university committees.

Our ambition is to further improve the proposed system and to see it embedded into the existing systems of the International Hellenic University, as well as into the systems of other universities, enhancing in this way their dissertation assignment mechanisms.

Konstantina Vezirtzoglou

December 2017
# Table of Contents

1. Introduction.........................................................................................................................8

1.1 Motivation..........................................................................................................................8

1.2 State of the art....................................................................................................................9

   1.2.1 Recommender systems in general..............................................................................9

   1.2.2 Recommender systems in education.........................................................................10

1.3 Objectives of the dissertation...........................................................................................11

1.4 Outline of the report.........................................................................................................12

2. Recommender system requirements....................................................................................14

2.1 User requirements............................................................................................................15

   2.1.1 Student requirements...............................................................................................15

   2.1.2 Advisor requirements...............................................................................................15

   2.1.3 Committee member requirements............................................................................16

2.2 Computational and Platform requirements......................................................................16

2.3 Algorithmic requirements...............................................................................................17

   2.3.1 The recommendation problem................................................................................18

   2.3.2 The matching problem.............................................................................................19

2.4 Basic vs. enhanced system versions................................................................................19

3. Design and Implementation...............................................................................................21

3.1 User Interfaces................................................................................................................21

   3.1.1 User Interfaces of the Mobile Application...............................................................21

   3.1.2 User Interfaces of the Website................................................................................23

3.2 User Experience................................................................................................................28

   3.2.1 Different screen sizes...............................................................................................28

   3.2.2 User interface colors...............................................................................................29

3.3 Tools and Languages used...............................................................................................31

   3.3.1 Firebase.....................................................................................................................32
3.3.2 iOS Mobile Application ................................................................. 37
3.3.3 Website ....................................................................................... 39
3.3.4 Version Control ........................................................................... 40
3.3.5 Tag extraction .............................................................................. 40

4. Developed Algorithms ........................................................................ 42
4.1 Algorithm for recommending topics to students .................................. 42
   4.1.1 Example .................................................................................... 47
4.2 Algorithm for recommendations to advisors ....................................... 49
   4.2.1 Example .................................................................................... 52
4.3 Algorithms for recommendations to committee members .................. 52
   4.3.1 Availability of student profiles: the Gale-Shapley Algorithm .......... 54
   4.3.2 Lack of information about student profiles: Random Serial Dictatorship algorithm .... 58

5. Evaluation ........................................................................................... 62
5.1 Usefulness of the developed system .................................................. 62
5.2 Performance of the system ................................................................. 67
   5.2.1 Mobile application performance ................................................. 67
   5.2.2 Website performance ................................................................. 69
5.3 Popularity of the system .................................................................... 71

6. Conclusions and future work ............................................................... 72
6.1 Conclusions ....................................................................................... 72
6.2 Future Work ...................................................................................... 73

Appendices ............................................................................................. 75
A.1 Mockups for website & mobile application ....................................... 75
   A.1.1 Website Mockups – Advisors .................................................... 75
   A.1.2 Website Mockups - Dissertation Committee ............................. 77
   A.1.3 Mobile Application Mockups – students .................................... 78
A.2 Licenses for icons used during the development process .................... 80
1. Introduction

This chapter will serve as an introduction to the topic of the dissertation by describing the motivation for it, the state-of-the-art and the dissertation’s objectives.

1.1 Motivation

In the end of their graduate or postgraduate studies, it is common for students to undertake a dissertation project as a prerequisite for the completion of their studies. The selection of the dissertation topic is often a time-consuming process that requires meetings between students and potential advisors, as well as the collection of useful information that can help both sides make “optimal” decisions about which topic will be undertaken by each student.

More specifically, in the majority of the universities, it is common practice for students to directly contact their professors and decide together about the dissertation topic that will be assigned to them, after consultation. It is a distributed tactic that requires direct contact between students and their potential advisors, before decisions about the final dissertation topics are reached.

Another less widespread approach is the one followed by the International Hellenic University (IHU) in its graduate teaching programs. It works along the following steps:

1. The teaching staff of IHU submit dissertation topics.
2. Once these topics become available to the students, they are asked to submit their preferences by choosing four of the suggested topics and ranking them according to their preferences. Alternatively, they are also able to propose their own topics.
3. Finally, it is the task of an IHU committee, hereafter called dissertation committee, to centrally assign topics to all students of the university in a way that respects their preferences to the maximum possible extent. This is almost always done heuristically, trying to respect a number of constraints such as the maximum load that will be delegated to each member of the teaching staff.

The above process serves as reference and case study for the dissertation recommender system that is developed in the context of this dissertation. The developed system includes a mobile
application and a website; thus, in the following, when we refer to this system, we provide information concerning both the mobile application and the website.

1.2 State of the art

This section discusses the existing general recommender systems, as well as recommender systems that specifically focus on the educational sector.

1.2.1 Recommender systems in general

The field of recommender systems has been researched extensively over the past decades and has advanced in parallel with the general progress of technology, in particular with the growth of the web. Recommender systems are software tools and techniques that are utilized in order to provide suggestions to users about the choices they will make on a number of areas ranging from which products they should buy to which online course they should attend. They almost always do so by filtering personalized information and analyzing it to infer their unique preferences. Among others, they might analyze the browsing history of users, the feedback they give to several products and distinct features of their behavior. [1]

One possible way to categorize recommender systems appears in Figure 1 and it classifies them into personalized and non-personalized. The first ones issue user-specific recommendations, whereas the recommendations of the latter are common for all users. [1] Personalized recommender systems are further divided into five categories. Three of those categories, namely content-based, collaborative filtering and demographic filtering systems, were initially the most common ones [2].

![Figure 1: Types of Recommendation Systems](image_url)
Systems that use content-based filtering, base their recommendations upon the preferences of users and the item descriptions. Collaborative filtering on the other hand leads to suggestions that are made based on the past behavior of users and on decisions that have been made by users with similar interests. Finally, demographic filtering systems utilize information that are related to the structure of the population, like a user’s age, gender and employment status. [1]

Despite of the category they belong to, recommender systems have been proven quite successful when used by well-known websites. Specifically, the e-commerce company Amazon.com uses recommender systems’ algorithms in various pages of their website, in order to suggest more items to be added in a user’s shopping cart. A Microsoft Research report estimated that 30% of Amazon’s page views came from such recommendations. [3] YouTube is another example of a company that uses such algorithms, so that different videos can be recommended to each viewer. [4] Likewise, Netflix used recommender systems extensively and its Chief Product Officer, Neil Hunt, stated that their recommender system influences the choices of users for around 80% of the hours that are streamed at Netflix [5]. [6]

1.2.2 Recommender systems in education

While the aforementioned examples could be considered as some of the most well-known cases of the successful use of recommender systems, there are several application domains where such systems are being implemented and utilized. More specifically, recommender systems have been developed in the domains of e-government, e-business, e-commerce/e-shopping, e-library, e-tourism, e-resource service, e-group activity and e-learning. Regarding the latter, educational institutions have been developing and using recommender systems since the early 2000s. Several e-learning recommender system applications are described in [7] and their main goal is to help students choose which courses, subjects and learning materials could fit best to their interests and learning activities, as well as to help lecturers choose the teaching strategies that better match their students’ needs. [7]

Another example of a system that has been developed for similar purposes is Erie, which is a system that was designed to solve a problem that is often encountered in academic conferences, namely the assignment of submitted papers to the most appropriate reviewers. This system deals with challenges that are analogous to the ones that are encountered by the system that we will be suggesting. More specifically, Erie had to overcome the problem of finding suitability scores
between submitted papers and the papers of reviewers in the same way that we will need to find such scores between students and the submitted topics. Furthermore, Erie dealt with the assignment of submitted papers to reviewers, while our system will offer suggestions for the assignment of topics to the university’s students. [8]

A system that has also been proposed for educational use and closely relates to our dissertation recommender system is RWAS, Research Work Area Recommender System. This system aims to suggest a research work area to students, by comparing their characteristics with the characteristics of other students that already have some work experience in a given research area. The characteristics that RWAS takes into consideration include the user’s hobbies, subjects of interests, programming skills and future objectives. This system is still under development, but we believe that, in the future, its recommendations could be used by a dissertation recommender system, like the one that we are suggesting, in order to provide more general recommendations to users about research areas that should also be considered for their dissertation topics. [9]

To the best of our knowledge, this is the first time that the design and development of a dissertation recommender system is being addressed.

1.3 Objectives of the dissertation

The ultimate objective of this dissertation is to make the assignment of dissertation topics to the students faster and more efficient. To this end, we aim at developing a recommender system that can be used by all involved parties and serve as an assistant to their tasks.

More specifically, our ambition is that the software designed and developed in this dissertation will be used by:

- University students: the system will recommend to each student the topics that lie closest to their preferences. This will be accomplished by taking into consideration all factors that are considered by students when selecting their dissertation topics, like their interests, academic background and the courses they have attended throughout their studies. Furthermore, the system will take into consideration requirements that are specific to each dissertation topic. The aim is to help students make more targeted and well-informed decisions that are aligned with their personal preferences and their future career goals.
University supervisors: the developed system will also be able to provide recommendations to the university’s supervisors, hereafter called advisors, as to which students are better fits to the topics that they have proposed, so that communication between those students and the advisors could be initiated as a result of the system’s recommendations. In the case of IHU, the term “advisors” points to members of the IHU academic staff, IHU academic tutors and adjunct faculty, who hold teaching assignments in the graduate teaching programs of the university.

University committee: The recommender system could also be helpful for the university’s committee that is responsible for assigning topics to students by following the aforedescribed selection process. By consulting the recommendations made by the system, this committee would be able to make the best decision faster when in doubt about which of the students that prefer a given topic will finally undertake it.

This high-level goal will be realized through a number of intermediate objectives. Hence, our work aims to:

- **Design** a dissertation recommender system; that is, specify all the system functionality available to each type of user, including the mobile application, the web interface and the backend system.

- Formulate and **solve the technical problems** that arise in the system, e.g., the way recommendations are determined and how students are best matched with dissertation topics.

- **Develop** the software components of this system in line with the design choices.

### 1.4 Outline of the report

The remainder of this document is organized as follows:

Chapter 2 discusses the requirements that served as guide for the development of the recommender system. These requirements are multifold and relate to different dimensions of the system: the functionality that the system must provide to different parties using it; computational and platform-related requirements; and the algorithmic problems that had to be efficiently addressed. Furthermore, in this chapter, we distinguish between those functionalities that are implemented in a basic version of the recommender system against what could be possible future
extensions of a future release. This was deemed mandatory given the tight time constraints of the dissertation.

Then, chapter 3 focuses on the design and development process of the recommender system, detailing the processes that have been followed for the development of both the mobile application and the website. This chapter reasons about the decisions that have been made in regard to the user interfaces of the system and the tools and languages that have been used.

Chapter 4 discusses the algorithms that have been designed and developed so that our system would successfully provide recommendations to all types of its final users. More specifically, this chapter extensively discusses the algorithms regarding the topic recommendations that can be made to each student of the university, the recommendations of most suitable students for each proposed topic, as well as regarding the recommended allocations of topics to the dissertation committee.

Chapter 5 presents the methodology that has been used in order to evaluate the developed system and it reports the produced results. This evaluation has three dimensions, focusing on the usefulness, the performance and the popularity of the developed system.

Finally, the document is concluded in chapter 6, where our work is being summarized and conclusions are presented together with our final thoughts and ideas for future work.
2. Recommender system requirements

This chapter will describe the functionalities of the recommender system that has been developed and the problems that have been addressed by it.

In order to assist the process of selecting and allocating dissertation topics, we developed a mobile application that could be used by university students and a website that could be used by their advisors and by the university’s dissertation committee.

Advisors are able to use the website in order to add to the system the dissertation topics they want to propose and in order to edit their descriptions. The system can then provide recommendations to the advisors by suggesting the students that fit best to the description of each one of their topics. This way, advisors could communicate with those students in order to discuss about the possibility of them undertaking a given topic or they could consult the system whenever there is a need for them to make a decision about selecting the most suitable student for one of their topics.

The developed website can also be used by the university’s committee. The system is able to provide recommendations to the committee about how to allocate the topics that have been proposed by advisors to the students of the university. The final decisions about the allocation of topics will be made by the committee, while our system can serve as a consultant providing recommendations for this allocation in order to help speeding up the process and making it more efficient.

Finally, the mobile application that has been developed can be used by the students of the university. The goal of the application is to help students with deciding which topic they want to undertake out of the ones that have been proposed by the university’s advisors. Students will have to first complete their profile by giving feedback about the content of the courses they have attended. Moreover, they are also able to provide information to the application about their interests. This way, the system can then recommend topics that better fit the created profile of each student.

The specific functionalities of the system that are available to each type of user, are described in more detail in the following section.
2.1 User requirements

This section presents the user roles of the mobile and web applications and it connects each user/role with the application functionalities that they are be able to use.

More specifically, there are three types of users for the mobile application and the website that have been developed:

- People who can be helped when selecting their dissertation topic – **students**
- People who manage the topics that are inserted in the system – **advisors**
- People who are responsible for allocating the topics to students – **committee members**

In order to satisfy the needs of all types of users, we developed the appropriate functionalities that can help us achieve the requirements of the project. These requirements are divided into three categories, depending on the type of the user, and they appear in the following subsections.

### 2.1.1 Student requirements

Students that use the developed software should be able to use the system and fulfil the following requirements:

SR1. Students shall be able to create a profile that matches their interests and personal preferences.

SR2. Students shall be able to rate the content of the courses they have attended throughout their studies.

SR3. Students shall be able to edit their profiles.

SR4. Students shall be able to view the available dissertation topics in the order which is recommended by the system.

### 2.1.2 Advisor requirements

The requirements that address the needs of the advisors for the developed recommender system appear below.

AR1. Advisors shall be able to add, edit and remove their proposed dissertation topics through the website.

AR2. Advisors shall be able to see which students are the most suitable for their proposed topics.
AR3. Advisors should get alerted when very similar topics are inserted to the system by other professors.

2.1.3 Committee member requirements
The requirements that are addressing the needs of the committee members that will be using our system are the following:

CR1. Committee members shall be able to get recommendations for the allocation of topics, through the system.
CR2. Committee members shall be able to view and remove proposed topics from the system.
CR3. Committee members should be able to see which students are the most suitable for each proposed topic.
CR4. Committee members should be able to manually allocate topics to students.

2.2 Computational and Platform requirements
The aforementioned requirements of users have been fulfilled by the development of the mobile application and website components of our system. We believe that advisors and committee members are most likely going to use devices with relatively big screens and keyboards when utilizing our software, since they will need to add and edit topics, as well as to get recommendations about their allocation. Thus, we decided to develop a website that could be easily used from any such device, regardless of its size and the operating system it has. Our goal was to develop a responsive website, so that the user experience would be flawless in both bigger and smaller screens.

Students on the other hand will not need to add any information to the software by typing their preferences. Thus, our goal was to develop a mobile application that can be used at all times by these users when they want to simply edit their profiles or see the dissertation topic recommendations that are made by the system. We believe that applications should be developed for both iOS and Android mobile devices, since as of the first quarter of 2017, these are the mobile operating systems that hold 99.7% of the market share [10].
For the purposes of this dissertation we focused on the design and development of an iOS mobile application, whose deployment target version is iOS 10.0, meaning that the application can be installed on devices running iOS version 10.0 or later. It is interesting to mention that in December 2017 iOS versions 10 and 11 were installed in 92% of all iOS devices [11].

Moreover, for the development of our system we needed to select a backend system that could work well with both the website and the mobile application that have been developed. The functionalities that should be supported by the selected backend system are the following:

- Authentication of users
- A database in order to keep information about the system’s assets, including the available topics and courses, the users of the system and their profiles

### 2.3 Algorithmic requirements

This section presents the algorithmic problems that needed to be tackled to enable the mobile application and website components of the system. These problems are the following:
The recommendation problem
The matching problem

Each problem will be discussed separately in the following subsections.

2.3.1 The recommendation problem
Two of the main goals of the recommender system that will be developed are the recommendation of dissertation topics to students and the recommendation of students for the proposed dissertation topics. The input that we get in order to make both types of recommendations is common: we have the description of each dissertation topic and the profiles of students, which comprise their individual general preferences. Thus, the algorithms that will be used can take advantage of the same information when recommendations to students and advisors need to be made.

One basic architecture for a recommendation system classifies them into a group of content-based systems, where we primarily focus on the properties of items in order to make recommendations to our users. This is the architecture that we followed for the development of the algorithms that we use in order to address the recommendation problem. [12]

The process that needed to be followed for the development of our algorithms included the creation of “item profiles” and “user profiles”. These profiles are collections of records about the characteristics of each proposed topic and about the preferences of students, respectively. In order to make recommendations, our system needs to first construct “utility matrixes” that will demonstrate the connection between students and topics. Utility matrixes depict the preferences of users over the items that they have tried and the goal of a recommender system is to indicate how much a user will like items for which no current preference information exist. In our case, as users we can consider the university’s students and topics can be considered as the items. The preferences of users over items is often expressed in the form of ratings, but since we do not want students rating the proposed topics, we decided to take into consideration the courses that students attend and we let users rate the contents of those courses (often mentioned as course ratings in the remainder of this document). This way, considering topics and courses as items with the same features, the utility matrix can have information about the student-course pairs and recommendations to both students and advisors can be made based on the similarity of each student-topic pair.
2.3.1.1 Item & User profile tags

One of the first problems that arose with the creation of the item and user profiles was the definition of the profile features. We could either use an open or a closed set of tags as the profile features. However, there is a tradeoff between having an open or a closed set of tags. More specifically, by having a closed set of tags, topics will be more easily comparable, but they will probably not be as well described as if we had an open set of tags. Nonetheless, an open set of tags would impose difficulties in the comparison of items and users. Thus, for the first version of the developed system, we decided to create the item and user profiles having a closed set of tags. In order to define these closed set of tags we followed the process that is described in section 3.3.5.

2.3.2 The matching problem

Another main goal of our recommender system is to offer recommendations to the committee members about the assignment of topics to students, given their submitted preferences. In order for our system to be able to provide such recommendations, we needed to develop algorithms that determine the eventual matching of topics to the students.

In section 4.3 we present the algorithms that have been selected in order for our system to successfully provide the topic allocation recommendations to the dissertation committee.

2.4 Basic vs. enhanced system versions

Most of the functionalities and requirements that have been mentioned in this chapter have been included in the basic versions of the mobile application and the website that we have developed. Specifically, students are able to add and edit their profiles, while advisors and committee members have the ability to add and edit dissertation topics. Additionally, all users are able to get distinct recommendations by the system either about which topics are most suitable for each student, which students are most suitable for each topic or about how topics should be allocated to the students of the university. In the following table, you can see which of the requirements have been included in the basic software versions and the requirements that could be considered as extensions to be accommodated in future releases. The table’s column names refer to the requirements of each type of user, as they appeared in section 2.1.
At this point, we would also like to propose some ideas about future extensions of our dissertation recommender system. Some of the ideas that could be considered are the following:

- An Android application could also be developed
- All types of users (students, advisors, committee members) could be able to use both the website and the mobile application(s)
- Topics that are checked a lot of times by a student could be added high on the recommendations made by the system
- Students could be able to bookmark topics that they find the most interesting
- Students could be notified by the application when new interesting topics are added to the system
- Students could be notified by the application in order to rate their courses, as soon as a semester ends
- Explanations about why the system makes its recommendations could be provided to all users
- Tags could be automatically suggested for each added topic, based on the TF-IDF of the terms that are present in the topic’s description
- Committee members could be notified when there are not enough topics that match well with the students’ profiles
- General suggestions about a research work area could be provided
- Current trends could be taken under consideration for topic recommendations
- Data could be cached to improve the quality of recommendations and to reduce any unnecessary network communication

### Table 1: Requirements included in basic and enhanced versions

<table>
<thead>
<tr>
<th></th>
<th>SR1</th>
<th>SR2</th>
<th>SR3</th>
<th>SR4</th>
<th>AR1</th>
<th>AR2</th>
<th>AR3</th>
<th>CR1</th>
<th>CR2</th>
<th>CR3</th>
<th>CR4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Basic Version</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>Enhanced Versions</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
</tr>
</tbody>
</table>
3. Design and Implementation

This chapter will present specific information about the system’s design and development process. More specifically, sections 3.1 and 3.2 focus on the justification of the system’s design decisions, while sections 3.3 and 3.4 present more thoroughly the development process. We will start by discussing the decisions that have been taken in regard to the user interfaces of the mobile application and the website.

3.1 User Interfaces

Before developing our mobile application and website, we created some mockup user interfaces so that the development process would be driven by them. The mockups have been made using the Balsamiq Mockups Tool\(^1\) and they appear at the appendices sections of this document. Following the general outlines that were suggested by these mockups; we developed our products and in the following subsections you can see them as they appear in their final form.

3.1.1 User Interfaces of the Mobile Application

The screenshots that appear in this section have been taken from an iPhone 8 Plus simulator. The first interface that users encounter when opening the application for the first time is the login screen, where they can insert their credentials in order to access the application.

When the users of the system enter false credentials when trying to log in or when non-student users try to enter the mobile application, appropriate error messages appear at the login screen.

As expected, after students enter correctly their credentials, they get access to the mobile application and they are being redirected to the “Courses” section of the application, where they can see a list of the courses they have attended throughout their studies. At this section, users are able to rate the content of each course, in order to get more personalized suggestions by the system.

\(^1\) [https://balsamiq.com/products/mockups/]
Through the bottom tab bar of the application, users can navigate to the other available sections. The “Topics” section includes the topics that have been proposed by the university’s advisors, sorted in the order that is suggested by the recommender system, having most suitable topics for each student appear at the top of the list. Notice here that the mobile application has been developed in such a way that it supports both portrait and landscape device orientations. The following screenshot of the “Topics” section has been taken in landscape mode.
Users are able to select each topic in order to see more information about it and they are finally, able to access their profiles, in order to define their interests or logout of the application. The details screen of a random topic and the profile screen of a user can be seen in the screenshots below.

Figure 8: Topic details - iOS

Figure 9: Profile section – iOS

3.1.2 User Interfaces of the Website

The system’s website is targeted to the university’s advisors and committee members. The initial page of the website is the login page, which is used in order to access each user’s page and can be seen in the following screenshot.
Furthermore, through this initial page users are able to move to the registration section, in order to register new members to the system.
Once advisors are logged into the system, they get redirected to their home page, where they can access all of the topics they have proposed, in order to view, edit or delete them.

**Figure 12: Advisor’s home page – Website**

By clicking on the icons that appear under the first column of the table next to each topic, users are able to see more information about a given topic, including the recommendations of students that are suggested by the recommender system as most suitable for this topic. Hereby follows a screenshot of the details of a random topic.

**Figure 13: Topic details & Student suggestions for topic – Website**
When selecting to edit one of their topics, advisors get redirected to the following page, where they are able to make the changes they need.

![Edit Topic Page](image1.png)

**Figure 14: Edit topic page – Website**

The last page that can be used by advisors is the page for adding new topics to the system. This page follows the same pattern as the page for editing topics and can be seen below.

![Add Topic Page](image2.png)

**Figure 15: Add topic page – Website**
Regarding dissertation committee members, their home page gives them access to all of the proposed topics, so that they can view or delete them.

![Figure 16: Committee member's home page – Website](image)

By clicking on the “Allocate topics” button committee members get redirected to the page where topic allocations are being recommended to them by the system.

![Figure 17: Recommendation of topic allocations](image)
Finally, in cases where a recommended allocation constitutes into one or more students that are left without an allocated topic, our system is able to offer the names of such students to the dissertation committee. In the following screenshot, we present an example of how such students appear when the “Students with no topic allocations” button is clicked.

![Figure 18: Students with no topic allocations](image)

### 3.2 User Experience

One of our ambitions was to provide a beautiful user experience to the end users of the system. Both for the mobile application and for the website, we want users to have a flawless experience regardless of the screen size of the device they will be using.

#### 3.2.1 Different screen sizes

Throughout the development of the mobile application, we took advantage of the tools that are provided to developers by Apple\(^2\), in order to create beautiful user interfaces for all iOS devices. More specifically, we used Xcode\(^3\) as an integrated development environment (IDE) and we were able to build our application in any iOS device simulator. Moreover, we were able to use Xcode’s “Interface Builder” while designing our user interfaces, in order to directly see how the interfaces would look in different devices and orientations. The storyboard designer of “Interface

---

\(^2\) [https://www.apple.com](https://www.apple.com)

\(^3\) [https://developer.apple.com/xcode/](https://developer.apple.com/xcode/)
“Builder” helps us easily create our user interfaces and below you can see a screenshot of our developed storyboard, with the bar on the bottom allowing us to navigate through the existing devices and orientations.

![Figure 19: Xcode storyboard](image)

Regarding the recommender system’s website, we followed a responsive design approach so that the web pages would be rendered adequately regardless of the size of the browser’s window or the device’s screen. In order to accomplish that, we applied specific CSS rules for different screen sizes, when needed, and we also made use of the Bootstrap\(^4\) web framework which encourages the development of responsive, mobile-first websites.

3.2.2 User interface colors

One major decision we had to make was about the colors that would be utilized to the user interfaces of the mobile application and the website. As you can see from the screenshots that have been provided, the system’s logo and its software components (i.e. the mobile application and the website), were designed minimally with shades of the blue color being evident in several places of the software.

\(^4\) [http://getbootstrap.com](http://getbootstrap.com)
We can justify our decision to select blue as our dominant color for several reasons. One of those reasons is that the developed system is built taking as an example the case of the International Hellenic University, whose logo dominant color is also blue.

Furthermore, our decision to use this color was more explicit due to the meaning of the blue color to many people, as well as due to the associations that are made by people who prefer this color. More specifically, a study that has been made by the Kutztown University of Pennsylvania regarding the perception of color in product choice among college students, showed that blue is considered as the safest color globally and it represents trust, truth, confidence, security and reliability, among others. [13] Since we built a recommender system, we believe that it is important for our users to associate it with all the above in order to feel more comfortable with using the system and to be more easily convinced about the trustworthiness of our recommendations.

Finally, the last crucial reason why we chose blue as our system’s dominant color; is because of its significance for color blind people. By choosing to use different shades of the blue color to our system we made sure that it would be more easily accessible by people with most types of color blindness, since they will be able to see this color and use the software without any difficulty. [14] Note here that “Color blindness affects approximately 1 in 12 men (8%) and 1 in 200 women in the world” [15].
3.3 Tools and Languages used

This section will present the tools and languages that have been used throughout the recommender system’s development process. Their logos appear in the following table and we will start by justifying our decision to use Firebase in the place of our backend system.

Table 2: Tools and languages logos

<table>
<thead>
<tr>
<th>Mobile Application</th>
<th>Website</th>
<th>Tag-Extraction</th>
</tr>
</thead>
<tbody>
<tr>
<td>Backend</td>
<td><img src="https://firebase.google.com/brand-guidelines/" alt="Firebase logo" /></td>
<td><img src="https://firebase.google.com/brand-guidelines/" alt="Firebase logo" /></td>
</tr>
<tr>
<td>Version Control</td>
<td><img src="https://github.com/" alt="GitHub logo" /></td>
<td><img src="https://github.com/" alt="GitHub logo" /></td>
</tr>
<tr>
<td>Programming Languages</td>
<td><img src="https://swift.org/" alt="Swift logo" /></td>
<td><img src="https://swift.org/" alt="Swift logo" /></td>
</tr>
</tbody>
</table>

5 [https://firebase.google.com/brand-guidelines/](https://firebase.google.com/brand-guidelines/)
6 [https://github.com/](https://github.com/)
7 [https://itunes.apple.com/](https://itunes.apple.com/)
9 [https://swift.org/](https://swift.org/)
10 [https://www.planet-source-code.com/](https://www.planet-source-code.com/)
11 [https://www.python.org/](https://www.python.org/)
3.3.1 Firebase\textsuperscript{12}

As mentioned in section 2.2, our goal was to have a common backend system both for the mobile application and for the website. Furthermore, the main functionalities that needed to be supported by this system were the following:

- The authentication of users
- The storage of information about the system’s assets, into a database

We first researched the options that were available, including the option of setting up our own server and hosting our backend on it, as well as existing products like Amazon Web Services\textsuperscript{13} and Microsoft Azure\textsuperscript{14}. After examining the available options, we decided that the best choice for the development of our system was the use of a backend as a service (BaaS) platform, like Firebase or Parse\textsuperscript{15}.

Our final decision was to use the Firebase platform, since it could support both the mobile and the web applications’ development, it is able to handle the authentication of the system’s users and it offers us a realtime database that can store our data. Other important advantages of Firebase involve the fact that it is currently widely used by companies and developers, and that community support and plenty of tutorials are broadly available online.

Firebase provides us with a “Console”, through which we can manage the products that are being used by our project. We named our project “dRecS”, as you can see in the following screenshot, and the Firebase products that are used by this project are the “Authentication” of users and the “Realtime Database”.

\textsuperscript{12} https://firebase.google.com
\textsuperscript{13} https://aws.amazon.com
\textsuperscript{14} https://azure.microsoft.com/en-us/
\textsuperscript{15} http://parseplatform.org
3.3.1.1 Authentication of users

For the authentication of users to our system we decided to use emails and passwords. Thus, users are able to select an email and a password in order to register and log in to our system. For the purposes of the dissertation this method would suffice, but in real-life scenarios; universities are able to use their existing account systems directly with Firebase, in order to let their users utilize their existing credentials when signing in to our system.

The way users that are registered to dRecS appear at the Firebase console can be seen in the screenshot below. As you can see, Firebase provides a unique identifier (UID) to each user and it gives us information about the registration date of each user and the last date when he/she signed in to the system.
3.3.1.2 Realtime Database

The other Firebase product that is being used by dRecS is the “Realtime Database”, through which we are able to store and synchronize data among all users and devices in real time. Our database is hosted on the cloud and it is NoSQL. More specifically, data are stored to the database as JSON objects and this is something that made easier the development process of our system, since our database was more flexible and we could make less effort whenever we needed to change its structure. By using the SDKs that are provided by Firebase, we are able to store and synchronize data both at our mobile application and at the system’s website.

The final structure of the database can be seen in the screenshot below. On the top-most level we have “courses”, the “currentTopicNumber”, “tags” and “users” objects.

![Database structure - Firebase](image)

Figure 24: Database structure - Firebase

The IDs of the topics that are added to the system follow this pattern “DS_X”, where X is a positive number. The database’s “currentTopicNumber” object keeps track of the number of the latest topic that has been added to the system. Thus, we can have our backend define the number part of the IDs of new topics that are added to the system, leading to unique IDs for each topic.

The “tags” object includes all the tags that are available for the topics that are added to the system, the university’s courses and each student’s interests. We will give more information about the selection process of these tags in section 3.3.5.

In the following screenshot, you can see how objects of courses are stored in our database. As you can see, each course includes information about the tags that characterize it, as well as the master programs where it is being taught.
Our last type of objects, the system’s users, have a different structure for each type of user. Students that are added to the system follow the structure of the example student that appears in the screenshot below. This way we are able to store all necessary information about students, like their master programme, the elective courses they have attended, the ratings they have given to their courses, the interests they have added to their profiles and the preferences they have expressed over the proposed topics. Note that the information about the elective courses and the topic preferences of students has been added manually to our database, although a commercial product would have to offer a user-friendly interface to handle such insertions, too.
Advisor objects follow a different structure, so that we can store their personal information and the topics that each one of them has proposed. A screenshot of the structure of advisor objects can be seen below.

![Database advisor objects' structure - Firebase](image)

**Figure 27: Database advisor objects’ structure - Firebase**

Finally, with Firebase we were able to impose safety rules to our database, in order to preserve the integrity of the stored data. For our system, the rules that we have imposed let us make sure that information about each user can be read only by authenticated users, while changes at a given user object can be made either by this user or by a committee member. Moreover, information about the latest topic’s number can only be read by the university’s advisors and committee members, while any authenticated user can access the tags and courses objects. All of the imposed rules appear in the following screenshot.
3.3.2 iOS Mobile Application

As mentioned in section 3.2.1, we used Xcode as our IDE for the mobile application’s development. We specified iOS 10.0 as our deployment target and we targeted all devices running iOS. Thus, any iPhone, iPod touch or iPad device can run our mobile application, provided that its operating system is iOS 10.0 or higher.

The programming language that we used for the mobile application’s development is Swift, a language that was introduced by Apple and has been open sourced since late 2015 [16]. Swift 4 has been officially released in September 2017 and it is currently the latest version of the language and the one that we used for our mobile application [17].

3.3.2.1 Software Architectural Pattern

The architectural design pattern that we followed for the development of our application is “Model-View-Controller” (MVC). We organized our code in such a way, so that each part would be independent and communication among the model, the view and the controller would respect the guidelines of the pattern, as they appear in the following figure.
Our model is associated with the logic behind our application, including code about the algorithm for the recommendation of topics to students, as well as code about how data should be retrieved from the database. The controller is responsible about managing the display of this information on the user’s screen, while the view consists of the components that the controller uses in order to display the information to the user interface (e.g. the buttons, labels and scroll views of the user interface).

In the following screenshot, you can see our mobile application’s files grouped under the “Model”, “View” and “Controller” folders. The name of each file that belongs to the model ends with “–brain”, since we wanted to indicate that the application’s logic is put inside these files. The view consists of only one “.storyboard” file, which has been presented in section 3.2.1 and consists of the UI components of the mobile application. Finally, files that serve as controllers to our application have “–controller” as their suffix, indicating that these files are responsible for managing the display of the information that we get through our model; to the user interfaces.
3.3.2.2 CocoaPods

Before concluding the section about the mobile application’s development, we would like to highlight the use of CocoaPods in our application. CocoaPods is a dependency manager for the Swift and Objective-C languages. Through this manager we were able to use some existing libraries for the development of our system’s mobile application.

More specifically, we initially used CocoaPods in order to install the libraries that were needed for using Firebase. Furthermore, we used the “HTagView” library for displaying the students’ tags in the “Interests” section of their profiles [18] and the “FloatRatingView” library for displaying the rating stars under each course that a student has attended [19].

3.3.3 Website

For the development of the recommender system’s website we used the Sublime Text source code editor and the languages in which our web application was built are the following:

- **HTML** – to define the structure of our website’s content
- **CSS** – to define how the content will be presented to browsers
- **JavaScript** – to handle the authentication of users and the communication with the database through Firebase, as well as to define the algorithms that are used by the website in order to make recommendations to the system’s users

![HTML, CSS, JavaScript](source: http://www.unwembi.co.za/)

Figure 31: HTML, CSS, JavaScript [source: http://www.unwembi.co.za/]

16 https://www.sublimetext.com
3.3.4 Version Control
Throughout the development process of the recommender system’s mobile and web applications we used a Version Control System (VCS) in order to keep track of the additions and changes in the source code files of both software components. The VCS of our choice was Git, since it is one of the major and most widely used VCSs [20]. Additionally, we maintained online repositories for the mobile and the web applications’ files. We chose GitHub¹⁷ as the hosting service for our Git repositories and we were thus able to maintain safe copies of our code files online and to revert our project’s files to previous states, whenever needed.

3.3.5 Tag extraction
In order to provide recommendations through our system we needed to be able to create the profiles of students and dissertation topics. Each student would be characterized by their interests, which would be presented as a set of tags, and by the ratings they would give to the content of their courses, which would also need to be characterized by tags. Moreover, advisors would have to characterize their topics by selecting the tags that describe best each one of them. Our goal was to be able to compare the student profiles with the topic profiles and thus we decided to have the same set of tags available for the characterization of both students and topics. In order to define this set of tags in advance, we used the information about the topics that were proposed in 2017 by the International Hellenic University’s advisors for the students of the following master programs: MSc in ICT Systems, MSc in Mobile and Web Computing, MSc in Communications and Cyber Security and MSc in e-Business and Digital Marketing. Furthermore, we used the descriptions of the courses that were being taught at these master programs, which we were able to find online at the university’s website¹⁸.

We automated part of the process of defining the set of tags, by developing some scripts in Python, using some text processing libraries from the Natural Language Toolkit (NLTK)¹⁹ and

¹⁷ https://github.com
¹⁸ https://www.tech.ihu.edu.gr
¹⁹ http://www.nltk.org
the OpenPyXL\textsuperscript{20} library. Through these scripts we first extracted the information that we needed from the Excel file that contained the topic proposals by the advisors of the university. Then, we computed the TF-IDF and TF scores of the words appearing in all topic descriptions and separately the same scores of words appearing in the descriptions of courses.

TF-IDF is an arithmetic statistic that serves as an indication about the importance of words to a document [21]. This statistic is computed by multiplying how many times a term appears in a document, which is the term frequency (TF), with the inverse document frequency (IDF), which indicates the uniqueness of a word among a given set of documents. In our scripts, we took into consideration the TFs of each word in all topics’ (or courses’) descriptions and we multiplied them with the IDF of the word among the available topics (or courses). Thus, we were able to find the most important and frequent words among all topics, compare them with the most important and frequent words among all courses and manually construct a list of the words that we believed would be good candidates for our set of tags.

In this way, we were able to define the final set of tags, after characterizing the courses and topics that would be used by our system. We thus added the courses and topics objects to our database by taking into consideration the aforementioned list of candidate words.

\textsuperscript{20} https://openpyxl.readthedocs.io
4. Developed Algorithms

The developed recommender system provides different recommendations to each type of user. More specifically:

- Students using the system’s mobile application get recommendations about topics that score high in their preferences. Topics are ranked in order of decreasing user preferences.
- Advisors, on the other hand, use the system’s website. The system recommends to them ranked lists of students for each one of the topics they propose.
- Finally, the university’s committee members, who also interact with the system through the developed website, get recommendations about how topics could be assigned to the students of the university.

In the first two cases (students, advisors), the system needs to implement algorithms that identify and assess the relevance between the students’ interests/preferences and the proposed topics. In the second case, the call is for matching algorithms that consider different criteria, including such relevance, for assigning topics to students. This chapter presents the algorithms that are chosen and properly adopted so that our system can provide the aforementioned recommendations to each type of user.

4.1 Algorithm for recommending topics to students

In order to make recommendations to a student we need a measure that quantifies the relevance of different topics to the student. In our case, this metric is the cosine similarity between each student-topic pair. Namely, for a given student we assign to each topic a score that is equal to the cosine similarity of the student-topic pair. Topics can then be ranked per student in order of decreasing cosine similarity value so that topics with higher scores appear higher in this ranked list.

The cosine similarity can be computed only when we have adequate descriptions of the student interests/preferences and the available dissertation topics. We assume that both are given as vectors of weights over a common set of \(n\) thematic tags (features) so that the cosine similarity of each student/topic pair is given by the following formula [12]:
\[
\text{cossim}(\mathbf{s}, \mathbf{t}) = \frac{\sum_{i=1}^{n} s_i t_i}{\sqrt{\sum_{i=1}^{n} s_i^2} \times \sqrt{\sum_{i=1}^{n} t_i^2}}
\]

where \( \mathbf{s} \) and \( \mathbf{t} \) are the vectors of weights of the \( n \) common tags that describe the student interests and topic scope, respectively. Hereafter, we refer to \( \mathbf{s} \) as the student feature vector and \( \mathbf{t} \) as the topic feature vector.

The topic feature vectors are binary. A topic either possesses a certain feature (tag) or not. In the former case, the value of the vector in the respective position is one and the topic vector practically looks like this: \( \text{topicVector} = [\text{“topicTag1”: 1, “topicTag2”: 1...}] \).

In our case, topic vectors are known since each topic that is added to the system needs to include information about the tags that describe it. Therefore, we needed to derive the student feature vectors, in order to compute the cosine similarities and determine the recommendations to the university’s students.

In order to compute the student feature vectors, we first make the following assumptions:

- Ratings of a course’s content equal to 1-2 should be considered as negative ratings.
- Interests/tags that are added by a student to their profile should be considered as having a rating that is slightly higher than the highest rating that this student has given, assuming that the student has provided course ratings to the system.

We then distinguish between different cases depending on the amount of information we have at our disposal regarding the student interests:

**Case 1: The student hasn’t rated any course and hasn’t added any interest to their profile.**

The student vector is empty: \( \text{studentVector} = [] \). Thus, there is no way to compute something and we can give the same score to all topics. In our application, we set as default topicScore = 0 and sort topics alphabetically since they all have identical scores.

**Case 2: The student has added one or more interests to their profile, but hasn’t rated any course.**

We compute the vector by giving the same positive weight (in our case, this is equal to one) to each interest/tag that is added to the student profile. Thus, topics that attract one or more of the student’s interests will appear higher in the topics’ ranked list. In this case, the student vector practically looks like this: \( \text{studentVector} = [\text{“interest1”: 1, “interest2”: 1...}] \). Thus, the cosine
similarity is 0 if we don’t have any common tags between the student and the topic and it is positive in any other case.

**Case 3:** The student has only rated one or more courses, but hasn’t added any interest to their profile.

**Case 3.1:**

If the student has only rated one course or if the student has rated more than one courses but gave the same rating to all of them:

**Case 3.1.1:**

- If the rating is >2, we add a positive weight to each one of the rated courses’ tags in the student feature vector. This positive weight is equal to 1. Thus, an example vector would look like this: `studentVector = [“ratedCourseTag1”: 1, “ratedCourseTag2”: 1 …]`. As a result, courses that have no common tags with the student have a similarity score = 0 and topics that have common tags with the student have positive similarity scores.

**Case 3.1.2:**

- If the rating is <3, we add a negative weight to the student’s vector for each one of the rated courses’ tags. This negative weight is equal to -1. Thus, an example student vector would look like this: `studentVector = [“ratedCourseTag1”: -1, “ratedCourseTag2”: -1 …]`. As a result, courses having no common tags with the student have a similarity score = 0 and topics having common tags with the student have negative similarity scores.

**Case 3.2:**

If the student has rated more than one courses, with different ratings, we compute the weight of each tag from all available topics that also appears in the rated courses as follows: `courseTagWeight = (sum of normalized ratings of courses that contain this tag)/(number of courses that contain the tag)`. Note that in order to get the normalized rating of a course, we subtract the average rating of the user from the rating of the given course [12]. For tags that are part of the feature space but don’t appear in any of the rated courses, we compute their weight as: `otherTagWeight = 3 – average_student_rating – 0.5`. By doing this, we make sure that tags that haven’t been rated will have a bigger weight than tags that have been...
rated negatively (i.e., the average rating of the courses containing these tags is <3). Hence, topics with tags that appear in positively rated courses are ranked higher than topics with tags that haven’t been rated at all. Finally, topics with tags that have been rated negatively, appear in the last places of the topics’ list.

In the following screenshot, we provide the code that is used for computing the weights in the student feature vector, for case 3.2.

```swift
Figure 32: Swift code for computation of tag weights – Case 3.2
```

**Case 4:** The student has rated one or more courses and they have added one or more interests to their profile.

**Case 4.1:**

If the student only rated one course or gave the same rating to the courses that they rated:

**Case 4.1.1:**

- If the rating is >2, we add a **positive** weight to the student’s vector to each one of the rated courses’ tags. To each one of the student’s interests, we add a **positive** weight which is slightly bigger than the weight of the tags of rated courses. The positive weight for the rated courses’ tags is equal to 1 and for the student’s interests it is equal to 1.5.

Thus, an example student vector would look like this: ```studentVector = ["interest1": 1.5, "interest2": 1.5, "ratedCourseTag1": 1, "ratedCourseTag2": 1 ...].``` 

As a result, courses that have no common tags with the student have a similarity score = 0 and topics that have common tags with the student have positive similarity scores, with student interests having a bit of a higher weight than positively rated courses’ tags.
Case 4.1.2:

- If the rating is <3, we add a negative weight to the student’s vector to each one of the rated courses’ tags. To each one of the student’s interests, we add a positive weight. The negative weight for the rated courses’ tags is equal to -1 and the weight for the student’s interests it is equal to 1.5.

Thus, an example student vector would look like this: studentVector = [“interest1”: 1.5, “interest2”: 1.5, “ratedCourseTag1”: -1, “ratedCourseTag2”: -1 …].

As a result, courses that have no common tags with the student have a similarity score = 0 and topics that have common tags with the student have positive or negative similarity scores, that depend on the common tags between the student and the topic.

Hereby follows a screenshot, where you can see the code that is used for computing the weights in the student feature vector, for case 4.1.

![Swift code for computation of tag weights – Case 4.1](image)

**Figure 33: Swift code for computation of tag weights – Case 4.1**

Case 4.2:

If the student rated more than one courses with different ratings:

- If a tag appears in the user’s interests, we assign it a relatively big weight in the student feature vector. Specifically, we make it equal to “maxRating – averageRating + 0.5” if the max rating of the user is >2 or equal to “3 – averageRating + 0.5” if the max rating of the user is <3, so that these inferred interests always get a bit bigger weight than all other tags.

- For any other tag, we compute its weight as in case 3.2.
4.1.1 Example

Based on the procedure that is explained above, we offer an example of how different topics would be sorted for a given student. Since case 4.2 is the most complex one, our example addresses this case, where a student, say Bob, has rated differently more than one courses and has added some interests to his profile.

The topics that we want to sort for Bob, are three and have the following feature vectors:

- topic1 = [“web”: 1, “mobile”: 1]
- topic2 = [“data-mining”: 1, “databases”: 1]
- topic3 = [“databases”: 1, “web”: 1]

We assume that Bob has added the tags “data-mining” and “data” to his interests and has rated the content of the course “Data Mining” with 5 and the content of the course “Advanced Database Systems” with 4. The tags of each course are the following:

- dataMiningTags = [“data-mining”, “data”]
- advancedDatabaseSystems = [“data”, “databases”]

The first step that we follow in order to compute Bob’s feature vector is to identify the maximum rating of Bob, which is 5, and compute the average rating he gave to his courses, which is equal to: (5 + 4)/2 = 4.5. Having this information, we can now compute the weights of all the tags that appear in the three topics.

For each one of the tags that appear in all topics as well as in Bob’s interests the student vector weights are set equal to “maxRating – averageRating + 0.5 = 5 – 4.5 + 0.5 = 1”. Thus, the student vector initially looks like this:

```
studentVector = [“data-mining”: 1]
```

Then, for tags that neither appear in the courses that were rated by Bob nor are part of the interests he added to his profile, namely for the tags “web” and “mobile”, the weights are set equal to “3 – average_student_rating – 0.5 = 3 – 4.5 - 0.5 = -2”. Therefore, the student vector now looks like this:

```
studentVector = [“data-mining”: 1, “web”: -2, “mobile”: -2]
```
Finally, since the tag “databases” appears in one rated course, we compute its weight as follows:

\[
\text{weight} = \frac{\text{sum of normalized ratings of courses that contain this tag}}{\text{number of courses that contain the tag}} = \frac{4 - 4.5}{1} = -0.5.
\]

Thus, the student vector finally looks like this:

\[
\text{studentVector} = \{\text{“data-mining”: 1, “web”: -2, “mobile”: -2, “databases”: -0.5}\}
\]

The square root of the sum of the weights of the student vector, which is needed in order to compute the cosine similarities, is the following:

\[
s = \sqrt{1^2 + (-2)^2 + (-2)^2 + (-0.5)^2} = \sqrt{9.25} \approx 3.04
\]

Since each topic has two vectors with weights equal to 1, the square root of the sum of the weights of all topic vectors, which is also needed in order to compute the cosine similarities, is the same and it is the following:

\[
t = \sqrt{1^2 + 1^2} = \sqrt{2} \approx 1.41
\]

The cosine similarity for each student/topic pair can now be computed and below we can see the results:

\[
\begin{align*}
\cos\text{sim}(\text{Bob, topic1}) &= \frac{-2-2}{3.04 \times 1.41} \approx -0.93 \\
\cos\text{sim}(\text{Bob, topic2}) &= \frac{1-0.5}{3.04 \times 1.41} \approx 0.12 \\
\cos\text{sim}(\text{Bob, topic3}) &= \frac{-0.5-2}{3.04 \times 1.41} \approx -0.47
\end{align*}
\]

Thus, topic2 will be the first recommendation of the system to Bob since it includes one of his interests and a tag that was included in a course that he rated positively. Topic3 will follow since it has one tag of a course that he rated positively and one tag for which we have no information. Finally, topic1 will be last since it is described by two tags for which we have no information about whether or not Bob would be interested in them.

Note that in order to make our example simpler, we did not include many tags to the descriptions of the student, the courses and the topics, but the algorithm has been tested and functions properly (i.e., its recommendations are in line with intuition) regardless of the number of tags.

The afore-explained procedure and scoring of the topics is student-centric: the computed student-topic scores for a given student should be considered only for the recommendations of topics to
this student. Thus, for the same topic, the scores of two different students cannot serve as good comparison values for the competence of each student for the given topic and a different algorithm should be used in order to make student recommendations to advisors for each one of their proposed topics.

4.2 Algorithm for recommendations to advisors

For each topic that is added to the system we want to make recommendations to the advisor that proposed it, indicating which students might be good fits for this topic. In order to do that, we created an algorithm that gives scores to each topic/student pair, for a given topic, and we recommend the students that have the highest scores.

The assumptions made in order to compute the aforementioned scores are the following:

- For each tag, we consider as positive the rating of at least one course containing this tag, with a rate that is greater than 2.
- We give very high significance to tags that have been added to the interests of a student.

The worst case that might be encountered when trying to recommend students for a given topic is described below.

**Worst Case:** None of the students has added even one of the topic’s tags as interest to their profile and none of them has rated positively even a single course containing one of the topic’s tags.

This is the single case, where we do not recommend any of the students for this topic since we have no evidence that they might be interested in it. In any other case, we are able to find students that could be recommended as good fits for the topic. Based on the above, we compute the score of a topic/student pair only if the student has added at least one of the topic’s tags to their interests or if they have at least rated positively a course that contains one of the topic’s tags. These scores will be computed by giving weights to each one of the topic’s tags separately for every student and by then summing up all these weights for each topic/student pair.

For each student, we need to give weights to all of the topic’s tags and for each one of them, we will be following the order below:

1. Students that added the given tag to their interests and that rated positively the courses that contain it should get the highest weight for this tag.
2. Students that just added the tag to their interests but haven’t rated any course containing the tag should get a weight that is a bit smaller.

3. Students that added the topic’s tag to their interests, but rated negatively the courses that contain this tag should get a weight which is even smaller.

4. Students that just rated positively courses that contain the tag, but haven’t added it to their interest should follow.

5. Students that rated negatively courses that contain the tag and haven’t added it to their interests should have the smallest weight for this tag. Notice that if some student falls into this category for all of the topic’s tags, then we shall not include this student to our recommendations.

For each one of the tags that describe a topic, students fall into one of the cases that are described below. Our goal is to compute the weights for all of the topic’s tags for each student, sum them in order to get the topic/student scores and thus make recommendations to the advisor, regarding their topic. In what follows, we list the cases within which a student might fall, when we need to compute the weight of each one of the tags of a topic:

**Case 1:** The student hasn’t added the topic’s tag as interest to their profile and hasn’t rated any of the courses containing this tag.

We give a neutral weight to this tag. In our algorithm, we decided that this neutral weight would be equal to 0.

**Case 2:** The student has added the topic’s tag as interest to their profile, but hasn’t rated any course containing this tag.

We give a positive weight to this tag. More specifically, any time a topic tag is added to the student’s interests, our algorithm gives it a weight equal to 6.

**Case 3:** The student has rated one or more courses containing the topic’s tag, but hasn’t added the tag as interest to their profile.

Based on the average rating that has been given from the student to the courses containing the given tag, we give different weights to this tag. Our goal was to count negatively average ratings that are between 1-2 and positively any other average rating. Thus, we heuristically came up with the following tag weights, based on the average ratings of courses containing the tag:

1. **Average rating: [1, 2] → tag weight: -1**
2. Average rating: (2, 3) → tag weight: 1
3. Average rating: (3, 4) → tag weight: 2
4. Average rating: (4, 5) → tag weight: 3

**Case 4: The student has rated one or more courses containing the topic’s tag and they have added it as interest to their profile**

We initially give a positive weight to the tag, equal to 6, since it has been added to the student’s interests. This weight is the same as the weight that has been selected for case 2. Also, depending on the average rating of courses containing the tag, we compute the weight that should be given to this tag based on case 3 and make the final weight of the tag bigger or smaller by adding or subtracting this weight to the initial weight that was equal to 6.

Note that for the above cases, the actual weights that are given to each tag have been defined heuristically, so that our recommendations would follow the logic that is described. The function that has been developed in JavaScript in order to compute the actual score of a student topic/pair, which is part of the whole algorithm that has been developed, appears in the following screenshot.

![JavaScript function for topic/student score computation](image)

**Figure 34: JavaScript function for topic/student score computation**

After computing the weight of each one of the topic’s tags for students that could be considered as recommendations for the given topic, we sum up the computed weights and get the scores of all topic/student pairs. Finally, we select the five students with the highest scores and offer them as recommendations to the advisor, for the topic they have proposed.
4.2.1 Example
We hereby present an example of a topic and how students would be selected as recommendations for it by our system.

We assume that we have a topic that has been described by the advisor that proposed it, with just two tags:

\[
\text{topic} = [\text{“data”, “data-mining”}]\]

**Student1** has not added any of the topic’s tags to their profile and hasn’t rated any course that contains at least one of the topic’s tags. Thus, this student **will not be recommended** by our system for the given topic.

**Student2** on the other hand has added the tag “data” to their interests, although they haven’t rated any course containing any of the topic’s tags. Thus, the topic/student **score** for student2 would be equal to 6.

**Student3** has rated courses that contain the tag “data” with an average rating of 3.4 and courses containing the tag “data-mining” with an average rating of 4.2. Furthermore, this student has also added both tags as interests to their profile. Thus, the topic/student **score** for student3 is equal to 17.

Finally, **student4** has only rated negatively courses containing the “data” tag with an average rating of 1.6. Thus, this student **will also not be recommended** by our system for the given topic.

Therefore, the only students that would be recommended by our system for the topic that was presented in our example would be student3 and student2. This is the actual order in which the system would provide its recommendations, since the score of student3 is greater than the one of student2.

4.3 Algorithms for recommendations to committee members
Our system is designed to provide recommendations to committee members that will assist them in making the final decisions about the assignment of topics to the university’s students. The goal of the system is to come up with an assignment that is most aligned with the preferences of students to the maximum extent possible.
In order to achieve the above goal, the system is designed to process two types of information. The first type, which is also the most crucial one, refers to the preferences of students over the topics that have been proposed by the university’s advisors. The International Hellenic University requires that their students submit such preference lists, as a prerequisite for assigning dissertation topics to them.\(^{21}\)

The second type of information processed by our system refers to the “preferences” of topics over the students of the university, which essentially reflect how well the interests and the profile of each student matches a given topic. In section 4.2, we have described the way in which such topic “preferences” could be extracted from the mobile application. More specifically, in order to recommend a student to the advisor of a given topic, we demanded that this student had rated positively at least one course containing a tag of this topic or that they had added one of its tags to their interests. To recommend an assignment of topics to the university’s committee members, we remove such limitations and consider a topic’s preferences also over students who do not satisfy the above criteria.

Out of these two types of information, the first one is always available to the system; namely, the IHU students always submit their lists of preferred topics. The maximum length of this list is four, but students are allowed to submit fewer topics. On the contrary, the second type of information is not guaranteed. Students may not be willing to share their interests through the mobile application or use it to rate courses. Hence, it may not be always possible to rank students on per-topic basis, according to the match of their profile to it.

Therefore, in our system, we explicitly distinguish and handle differently these two scenarios about the availability of student profiles from the mobile application. When an adequate number of student profiles is available so that we can rank students according to the relevance of their profile to each topic, we approach the topic assignment question as an instance of the stable matching problem with incomplete lists, referred to as SMI problem in the literature [22]. The

---

\(^{21}\) The university students are also allowed to propose their own dissertation topics and the committee decides on whether to accept the request on per case basis, considering factors like the availability of an advisor who could assist the student with the proposed topic. Our recommender system does not treat these cases, which do not lend to recommendations and need to be handled separately by the committee.
algorithm we draw upon to derive stable matchings is the celebrated Gale-Shapley algorithm. We explain the notion of stability and the use of the algorithm in our context in section 4.3.1.

In the baseline case, we assume that the only available information consists in the submitted students’ preferences over the topics. This assumption renders the problem of topic assignment to students an instance of the matching problem with one-sided incomplete preferences [23]. The objective is to get a topic assignment, which maximizes the number of students getting their most preferred topic, and then the number of students getting their second most preferred topic and so on. To this end, we rely on the Random Serial Dictatorship (RSD) [24]. We describe its use in our context in section 4.3.2.

4.3.1 Availability of student profiles: the Gale-Shapley Algorithm

When adequate information is available to the system regarding the students’ and topics’ preference lists, we can use the Gale-Shapley algorithm. This algorithm originally addressed what is called the Stable Marriage problem with complete preferences (SM), which was introduced in 1962 by David Gale and Lloyd S. Shapley [25].

In its basic form, the SM problem seeks to match a number of men with an equal number of women. Input to the algorithm is the explicitly indicated preferences of each person over all people of the opposite sex. It has been shown that for this problem, the Gale-Shapely algorithm can always derive a stable matching. A matching is called stable when it does not feature any blocking pair, i.e., a pair of man-woman, (m, w), who both prefer each other over their matched partners in M [22]. Whereas there may exist multiple stable matchings for a given problem instance, the one the Gale-Shapley algorithm produces is male-optimal; namely, it gives men the best possible female partner according to their preferences, over what any other stable matching would do.

In our problem, we could parallelize men with the students of the university, having preference lists over the dissertation topics; while women could be parallelized with the proposed dissertation topics, having preference lists over the students of the university [22]. Recall (ref. section 4.2) that these preference lists are artificial, reflecting how well each student’s profile matches the scope of each dissertation topic. However, our problem has three distinct differences with respect to the original SM problem:
a) students express their preferences over a small subset of the topics and not over all available topics.

b) two or more students might be equally suitable for a given topic, leading to ties among these students at the topic’s “preference” list.

c) the number of students is not identical with that of the dissertation topics; the topics rather outnumber the students.

We can resolve the ties in b), by arbitrarily positioning students having the same scores in a topic’s preference list. Then, the constrained preference lists in a) and the asymmetry in c) give rise to a special case that appears in literature as a variation of the basic SM problem and it is known as the Stable Marriage problem with Incomplete Lists (SMI) [26]. In SMI, the definition of the blocking pair (hence, the definition of stable matching) is slightly different yet intuitive given that it is not possible to match all men and women. Now, a blocking pair for a matching M is an acceptable pair \((m, w)\) such that \(m\) is either unmatched in M or prefers \(w\) to his partner in M, and likewise, \(w\) is either unmatched or prefers \(m\) to her partner in M. The term acceptable implies that \(m\) does appear within the preference list of \(w\), and so does \(w\) within the preference list of \(m\).

The Gale-Shapely algorithm produces a stable matching, the man-optimal one, for the SMI problem, as it did for the original SM problem. Its pseudocode representation is shown in Figure 35. Men take turns in proposing to women that appear at the top of their lists, after leaving out those they have already proposed to, and women accept or decline a given proposal based on whether they are available or they prefer this man to their existing partner [26].

```
M = ∅;
assign each person to be free; /* i.e., not a member of a pair in M */
while (some man m is free and has not proposed to every woman on his list)
    if (w is free)
        add \((m, w)\) to M; /* w accepts m */
    else if (w prefers m to her current partner m')
        remove \((m', w)\) from M; /* w rejects m', setting m' free */
        add \((m, w)\) to M; /* w accepts m */
    else
        M remains unchanged; /* w rejects m */
return M;
```

*Figure 35: Gale-Shapley SMI algorithm pseudocode [26]*
If we use this algorithm to solve the problem of allocating topics to students, by substituting men with students and women with dissertation topics, we are able to get student-optimal stable matchings that have each topic being assigned to at most one student. Note that in the SMI case, stable matchings do not guarantee that all students are assigned an acceptable topic, i.e., one in their preference list [26]. Thus, our system is designed to execute the algorithm multiple times, each time resolving ties in the topics’ preference lists in different ways, and choosing the “best” possible stable matching, i.e., the one that maximizes the number of students who get an acceptable topic allocation. In case there are two or more such matchings, we need a criterion to differentiate between them, reflecting the overall satisfaction of students over the assigned topics. One possibility is to consider the sum of \( \text{rnk}_m(s) \) scores over all students, also mentioned as fitness scores in this document, i.e., recommend the matching \( m_{rec} \).

\[
m_{rec} = \{ m : \sum_{s \in S} \text{rnk}_m(s) < \sum_{s \in S} \text{rnk}_{m'}(s) \}, m, m' \in M_1 \}
\]

We call this criterion for choosing the recommended matching the SUMRNK criterion.

So, if for example the system has produced an allocation where five students are assigned with the topics that appear first on their preference lists and five students are allocated their second preferences, the overall student satisfaction score would be equal to 15. Assuming that another allocation would feature eight students getting their first preference and only two students ending up with their second choice, we would get an overall student satisfaction score equal to 12. The allocation that would be recommended by our system, according to the SUMRNK criterion, is the second one.

An alternative would be to choose the recommended matching lexicographically; namely, pick the matching that maximizes the number of students who get their first choice. If there is a draw between two or more of the matchings produced by RSD in this respect, choose the matching that maximizes the number of students who get their second-best choice. If we cannot single out one matching as superior up to considering the \( R^{th} \) choices of students, we pick one of those competing in random. We refer to this criterion as LEXRNK.

For example, one of the produced allocations could involve 5 students being assigned with their first preference and 5 other students getting their second choice. Another allocation might suggest 6 students getting their first preference and only 4 students ending up with their second
choice. Based on the LEXRNK criterion, the system would recommend the latter allocation, since it involves the maximum number of students getting their first choice.

4.3.1.1 Example of Gale-Shapley operation

We present a simple example of the algorithm’s operation in a toy example with three students and five topics. The preferences of students over topics are given in Table 3 and those of topics over students in Table 4.

Table 3: Student preferences – Gale Shapley example

<table>
<thead>
<tr>
<th></th>
<th>1st topic</th>
<th>2nd topic</th>
<th>3rd topic</th>
<th>4th topic</th>
</tr>
</thead>
<tbody>
<tr>
<td>Student 1</td>
<td>Topic 1</td>
<td>Topic 3</td>
<td>Topic 2</td>
<td>Topic 4</td>
</tr>
<tr>
<td>Student 2</td>
<td>Topic 1</td>
<td>Topic 2</td>
<td>Topic 4</td>
<td>Topic 3</td>
</tr>
<tr>
<td>Student 3</td>
<td>Topic 3</td>
<td>Topic 2</td>
<td>Topic 5</td>
<td></td>
</tr>
</tbody>
</table>

Table 4: Topics' preferences – Gale Shapley example

<table>
<thead>
<tr>
<th></th>
<th>1st student</th>
<th>2nd student</th>
<th>3rd student</th>
</tr>
</thead>
<tbody>
<tr>
<td>Topic 1</td>
<td>Student 1</td>
<td>Student 3</td>
<td>Student 2</td>
</tr>
<tr>
<td>Topic 2</td>
<td>Student 2</td>
<td>Student 1</td>
<td>Student 3</td>
</tr>
<tr>
<td>Topic 3</td>
<td>Student 3</td>
<td>Student 1</td>
<td>Student 2</td>
</tr>
<tr>
<td>Topic 4</td>
<td>Student 2</td>
<td>Student 1</td>
<td>Student 3</td>
</tr>
<tr>
<td>Topic 5</td>
<td>Student 2</td>
<td>Student 3</td>
<td>Student 1</td>
</tr>
</tbody>
</table>

Applying the Gale-Shapley algorithm to these data, we would end up recommending the following stable matching, regardless of the order in which students would be proposing.
You can notice that students get allocated topics that appear in the 1st and 2nd positions of their lists. These topics are the best that could be allocated to them, while having each topic allocated to at most one student, with no blocking pairs appearing in the proposed matching.

Note that this would be the only stable matching, if no ties appeared at the topics’ preference lists or if the only way in which such ties would be resolved is the one that appears at table 4.

### 4.3.2 Lack of information about student profiles: Random Serial Dictatorship algorithm

The “Random Serial Dictatorship” (RSD) algorithm becomes relevant when we do not have enough information about student profiles to extract per-topic rankings of students (topic “preference” lists).

A simple way in which we could recommend the allocation of topics, would be by using a mechanism known as the “Simple Serial Dictatorship” (SSD). With SSD, students are first strictly ordered, according to some arbitrary criterion, and then take turns “asking” for a topic to be allocated to them. The first student would deterministically get their first choice. The remaining ones get the choice that ranks top in their lists among the remaining topics. Intuitively, the lower the position of a student in this ordered list, the worse their chances to get a topic within their top preferences. As a result, and despite its simplicity, this algorithm tends to result in allocations that are inefficient and unfair. The original ordering of students induces a strong discrimination between those lucky ones who make their choices first and those unlucky ones that choose in the end and, with high chance, may end up without a topic within their preference lists. [24]

One way to overcome the aforementioned inefficiency and induce higher fairness, is to use the randomized counterpart of the SSD algorithm, the Random Serial Dictatorship (RSD) procedure. RSD generates a random ordering of the students and then resorts to the SSD process. [24] In our

<table>
<thead>
<tr>
<th></th>
<th>1st topic</th>
<th>2nd topic</th>
<th>3rd topic</th>
<th>4th topic</th>
</tr>
</thead>
<tbody>
<tr>
<td>Student 1</td>
<td>Topic 1</td>
<td>Topic 3</td>
<td>Topic 2</td>
<td>Topic 4</td>
</tr>
<tr>
<td>Student 2</td>
<td>Topic 1</td>
<td>Topic 2</td>
<td>Topic 4</td>
<td>Topic 3</td>
</tr>
<tr>
<td>Student 3</td>
<td>Topic 3</td>
<td>Topic 2</td>
<td>Topic 1</td>
<td></td>
</tr>
</tbody>
</table>

__Table 5: Topic allocations – Gale Shapley example__

The **Gale-Shapley algorithm** is a procedure that ensures the stability of an allocation by iteratively proposing and accepting matches until a stable matching is achieved. A *stable matching* is one where no pair of students would prefer each other over their assigned topics. The process ensures that every student is assigned to a topic that they prefer over any topic that another student could have been assigned to.

The **proposed matching** (or any matching accepted by students) is considered stable because no pair of students would prefer each other over their assigned topics. The key idea is that if one student prefers a topic to their assigned one and the other student prefers the first student to their assigned topic, then the pair is a blocking pair. They would prefer to switch topics, making the matching unstable. A stable matching ensures that no such blocking pairs exist.

This algorithm is particularly useful in scenarios where there is a limited set of resources (e.g., topics) to be assigned among a set of agents (e.g., students). The Gale-Shapley algorithm guarantees that a stable matching will be found, and it is efficient in that it runs in polynomial time. The algorithm is symmetric in the sense that both students and topics are considered in the same manner.

Students and topics have their respective preference lists. Initially, all topics are unassigned, and students are unmatched. In each round, an unmatched student proposes to their top choice, and the topic either accepts or declines the proposal. If the topic is already matched to another student, the proposal is rejected, and the unmatched student moves down their preference list. If the topic is not matched, the proposal is accepted, and the student is matched with the topic. If the topic is already unmatched, it accepts the proposal. The process continues until all students are matched.

In the Gale-Shapley algorithm, a stable matching is always guaranteed because of the way proposals are made. Each student proposes to their top choice, and topics only accept proposals from students who prefer them over their current partners. This ensures that no blocking pairs form, and the matching remains stable. If a student's top choice declines their proposal, the student moves to their next preference, and the process repeats until all students are matched in a stable way.
system, we run RSD multiple times, producing each time a different random ordering of the students. We then recommend to the university’s committee members, the matching that assigns to students the topics that rank higher in their preferences. Note that the total possible number of student orderings is $n!$, where $n$ is the number of students. Therefore, we essentially sample a space that grows huge already with small values of $n$.

More specifically, the choice of the recommended matching is made by applying two criteria in a lexicographic manner. Formally, let $S$ be the set of students and $R(s)$ the preference list of student $s \in S$. Moreover, let $M$ be the set of matchings produced by the iterative run of the RSD algorithm, $m \in M$ being one of them, as produced by a single run of the algorithm, and $\text{rnk}_m(s)$ the rank of the topic assigned to student $s$ where $\text{rnk}_m(s) \in [1,r]$ ($r = 4$ for the IHU case).

We first single out the matching(s) that maximize the number of students who are assigned topics listed in their lists of preferences. Let $M_1$ be the set of these matchings. For example, say we want to allocate topics to 10 students and the algorithm produces eight matchings with only six students ending up with an assigned dissertation topic and two matchings with all 10 students being assigned a dissertation topic. Then $M_1$ includes the two latter matchings.

In general, if $M_1$ is singleton, then the single matching at hand is the recommended one. If more than one matchings score equally well in the first criterion, we resort to one of the criteria discussed in section 4.3.1, to choose among them.

In what follows, we offer a pseudocode representation of the RSD procedure that is being run multiple times by our system, before recommending a topic allocation to the committee members:

```plaintext
/* Initialize an empty set of allocations of topics to students */
A = ∅

randomly order the students;
for (each student s among the ordered students)
    if (s’s topic preference list in not empty and t is the first topic in that list)
        add (s, t) to A;
        remove t from the topic preference lists of the remaining students;
```
After each time that the above algorithm is run by the system, we check if we end up having a better topic allocation in order to select the final allocations that will be recommended to the committee members, as described above.

Note that the system that has been developed for the purposes of this dissertation would use data that have been provided to us by the International Hellenic University, including the topic preferences of the students of four MSc programs and the dissertation topics that have been proposed by their advisors. Since the available information would lead to our system following the RSD procedure, we implemented this procedure as part of our website. If the system was to be used in practice by the university’s students and staff, we would also need to implement the Gale-Shapley algorithm and the step that checks whether that or the RSD procedure would be finally used by the system.

4.3.2.1 Example of RSD operation

Using the same toy example of three students and five topics in 4.3.1.1, we present the different allocations that would be produced by the system for the $3! = 6$ different orderings of students, following the RSD procedure.

Remember that the order in which students were proposing did not change the produced result of the Gale-Shapley algorithm while the RSD algorithm produces different results depending on this order.

Table 6 shows the assignment that emerges under orderings (1,2,3), (1,3,2) and (3,1, 2) This allocation would result to a fitness score equal to 4.

<table>
<thead>
<tr>
<th></th>
<th>1st topic</th>
<th>2nd topic</th>
<th>3rd topic</th>
<th>4th topic</th>
</tr>
</thead>
<tbody>
<tr>
<td>Student 1</td>
<td>Topic 1</td>
<td>Topic 3</td>
<td>Topic 2</td>
<td>Topic 4</td>
</tr>
<tr>
<td>Student 2</td>
<td>Topic 1</td>
<td>Topic 2</td>
<td>Topic 4</td>
<td>Topic 3</td>
</tr>
<tr>
<td>Student 3</td>
<td>Topic 3</td>
<td>Topic 2</td>
<td>Topic 5</td>
<td></td>
</tr>
</tbody>
</table>

If students are ordered as (2,3,1) or (3,2,1) we would end up having the following topic allocation, that suggests a fitness score equal to 5.
Table 7: Topic allocations case 2 - RSD example

<table>
<thead>
<tr>
<th></th>
<th>1st topic</th>
<th>2nd topic</th>
<th>3rd topic</th>
<th>4th topic</th>
</tr>
</thead>
<tbody>
<tr>
<td>Student 1</td>
<td>Topic 1</td>
<td>Topic 3</td>
<td>Topic 2</td>
<td>Topic 4</td>
</tr>
<tr>
<td>Student 2</td>
<td>Topic 1</td>
<td>Topic 2</td>
<td>Topic 4</td>
<td>Topic 3</td>
</tr>
<tr>
<td>Student 3</td>
<td>Topic 3</td>
<td>Topic 2</td>
<td>Topic 5</td>
<td></td>
</tr>
</tbody>
</table>

Finally, if students choose in the order (2,1,3) the allocation that would be produced by the system would yield a fitness score equal to 5 and it would be the following:

Table 8: Topic allocations case 3 - RSD example

<table>
<thead>
<tr>
<th></th>
<th>1st topic</th>
<th>2nd topic</th>
<th>3rd topic</th>
<th>4th topic</th>
</tr>
</thead>
<tbody>
<tr>
<td>Student 1</td>
<td>Topic 1</td>
<td>Topic 3</td>
<td>Topic 2</td>
<td>Topic 4</td>
</tr>
<tr>
<td>Student 2</td>
<td>Topic 1</td>
<td>Topic 2</td>
<td>Topic 4</td>
<td>Topic 3</td>
</tr>
<tr>
<td>Student 3</td>
<td>Topic 3</td>
<td>Topic 2</td>
<td>Topic 5</td>
<td></td>
</tr>
</tbody>
</table>

Thus, since we end up having all three students undertaking a topic in all the above produced allocations, based on the SUMRNK criterion, the system would recommend the first one since it has the smallest fitness score. Note here that the reason why our system runs the RSD procedure multiple times, is because such different orderings of students result to better or worse recommendations by the system.
5. Evaluation

This chapter aims to present the evaluation results of the developed dissertation recommender system. The evaluation of the system addresses:

- the usefulness of the developed system
- the system’s performance, and

We also discuss how we could assess the popularity of the website and the mobile application, which is the ultimate evaluation of the system.

5.1 Usefulness of the developed system

The usefulness of the system is linked to the quality of the provided recommendations to the system’s users. More specifically, we evaluate our system based on real data that were provided to us by the International Hellenic University. These data include:

- the list of dissertations topics, as submitted by IHU and visiting staff with teaching assignments in four MSc programs of the School of Science and Technology, in 2016-2017. The aforementioned MSc programs are the following: MSc in ICT Systems, MSc in Mobile and Web Computing, MSc in Communications and Cyber Security and MSc in e-Business and Digital Marketing).
- the topic preferences, as submitted by the students attending these four MSc programs
- the final decisions of the IHU dissertations’ committee members regarding the assignment of topics to the university’s students.

The information obtained by IHU concerns 50 students in total. The data are anonymized with respect to the students’ names. Having all the above information, we could compare the topic allocations that have been made by the university’s committee, with the allocations that would be proposed by the system.

Out of the 50 students, 21 were allocated with their own proposed topics and the decisions about these topics being the final allocations to these students were made by the university’s committee in a way that cannot be automated by our system. Thus, we only included in our database the information that was needed for the remaining 29 students and we compared the allocation
recommendations of our system with the final topics that had been allocated to them by the university.

Moreover, since data about the course ratings and interests of these students were not available so as to build their profiles, we could only approach the topic assignment problem as an instance of matching problem with one-sided ranked preferences (ref. section 4.3) and apply the Random Serial Dictatorship (RSD) algorithm to derive the recommended topic allocations. The algorithm, which has been thoroughly explained in section 4.3.2, was used to derive all the results that appear in the following screenshots.

1\textsuperscript{st} Recommendation:

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{Figure36.png}
\caption{Recommended allocation of topics 1/3 - 1\textsuperscript{st} Recommendation}
\end{figure}

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{Figure37.png}
\caption{Recommended allocation of topics 2/3 - 1\textsuperscript{st} Recommendation}
\end{figure}
As can be noticed in the screenshots, the names of the university’s advisors have been replaced with the word “Instructor” followed by a numeric identifier that helps us distinguish between the different advisors. Also, random pseudonyms have been produced in order to maintain the anonymity of the university’s students.

Regarding the resulting topic allocations that are being recommended by the developed system, we can see that all 29 students would be allocated with a topic listed in their preferred topics. 22 of them have been allocated with their most preferred topics (ranking first in their submitted preference lists), while 7 students would end up undertaking the topics that appeared second in their preference lists. Thus, the fitness score of such an allocation of topics, would be equal to:

\[(22 \times 1) + (7 \times 2) = 36.\]

By repeating the system allocation process multiple times, we have noticed that there are alternative topic assignments that could be recommended and result in 23 students undertaking their first topic preference 5 students being allocated topics that rank second in their preference lists, and 1 student undertaking a topic that ranks third in their preferences. The fitness score that would characterize an allocation of the topics like the one described above, would be equal to:

\[(23 \times 1) + (5 \times 2) + (1 \times 3) = 36.\]

One of these allocations appears in the screenshots below.
2nd Recommendation:

Figure 39: Recommended allocation of topics 1/3 - 2nd Recommendation

Figure 40: Recommended allocation of topics 2/3 - 2nd Recommendation

Figure 41: Recommended allocation of topics 3/3 - 2nd Recommendation
Both recommended topic allocations that have been presented result in the same fitness score and thus can be considered as equally good, as far as the system bases its recommendations on the SUMRNK criterion.

The final allocations that have been decided by the university’s committee members resulted in 21 students being allocated topics ranking first in their lists; 5 students undertaking their second choices; 2 students getting topics that ranked third in their lists; and, 1 student being assigned their fourth choice. The fitness score of this allocation is \((21 \times 1) + (5 \times 2) + (2 \times 3) + (1 \times 4) = 41\).

Table 9 compares the university’s allocations with the two allocations recommended by our system.

<table>
<thead>
<tr>
<th>Table 9: Topic allocations comparison</th>
</tr>
</thead>
<tbody>
<tr>
<td>University’s allocation</td>
</tr>
<tr>
<td>Students that got their 1st topic preference</td>
</tr>
<tr>
<td>Students that got their 2nd topic preference</td>
</tr>
<tr>
<td>Students that got their 3rd topic preference</td>
</tr>
<tr>
<td>Students that got their 4th topic preference</td>
</tr>
<tr>
<td>Students with no topic allocation</td>
</tr>
<tr>
<td>Fitness score</td>
</tr>
</tbody>
</table>

Remember that, as we mentioned in section 4.3.2, we consider as better allocations the ones that result to a lower fitness score. Based on that, we would favor the allocations that have been recommended by the developed system, although the university’s fitness score is relatively close to the fitness scores of these recommendations. Notably, the system recommendations
outperform the assignment made by the university committee also when the LEXRANK criterion (ref. section 4.3.1) is considered.

It is important to mention that the final decisions that were made by the committee members might have also considered the resulting workload for each advisor, which is a factor that could not be automated by our system, since the exact maximum workload of each advisor could not be known. Thus, if a given advisor ended up being responsible for supervising the dissertation topics of too many students, the committee members would need to specifically resolve such problems by having one or more students assigned to new topics that appeared lower on their preference lists.

Nonetheless, since oftentimes different good recommendations are available and can be provided by the system, we believe that it would be useful for the committee members to briefly consider these alternatives, each time they consult the system in order to make the final decisions regarding the topic allocations to students. This way, they would be able to instantly produce several possible allocation recommendations and make the final decisions by also taking into consideration factors that we were not able to automate.

5.2 Performance of the system

Regarding the evaluation of the system’s performance, we used different tools for the mobile application and the website components. The following sections present the details of each component’s evaluation.

5.2.1 Mobile application performance

In order to evaluate the performance of the mobile application, we used “Debug Navigator”, which is one of Xcode’s debugging tools. Debug gauges are a main part of this tool and they are able to monitor an application that is running, providing insights to the performance of the application through its use of system resources. [27] To evaluate the developed mobile application, we ran it on an actual iPhone SE device and we performed the functionalities that are offered by the application, like the rating of courses, the insertion of user profile data and the examination of the available dissertation topics.
The first type of debug gauge that we examined was about the device’s CPU. The following graph shows a history of the CPU’s usage over the time the application was being used.

![Usage over Time](image.png)

*Figure 42: CPU usage graph*

Figure 42 suggests that the highest percentage of CPU usage was 40% and, as indicated in the following meter view, CPU utilization could reach up to 200%, since the iOS device that we used had a dual-core processor.

![Percentage Used](image.png)

*Figure 43: CPU usage meter view*

Through the memory gauge we were able to see that our application needed to use only 21.8MB, which constitutes the 1.1% of the system’s total memory. The results regarding the memory usage can be seen in the meter view that follows.

![Memory Use](image.png)

*Figure 44: Memory usage meter view*

Regarding the energy impact of the developed mobile application, the following pie chart indicates the average energy utilization by the different components. It can be seen that the overhead is responsible for the vast majority of the energy that is being used. This can be easily
explained by the fact that our application performs several network operations in order to send and receive data from the system’s database and such communications often lead to extensive overhead cost [28].

![Energy usage pie chart](image)

*Figure 45: Energy usage pie chart*

Finally, regarding the network activity throughout the performance of the aforementioned functionalities of the mobile application, we noticed that 0.4MB were received by the device and only 11.2KB were sent by it.

![Network activity](image)

*Figure 46: Network activity*

### 5.2.2 Website performance

For the evaluation of the system’s website component we used the “Web Inspector” tool, which is one of the development tools that are provided by the Safari web browser. We first performed the basic functionalities that an advisor would perform, including the addition and deletion of dissertation topics and the inspection of the detailed information and recommended students of topics. Through “Timelines”, which is part of “Web Inspector”, we were able to have a visual representation of the network requests, JavaScript & Events, as well as of the website’s memory usage.

In the following screenshot, we show the results that were produced while we were using the website as one of the university’s advisors. It can be seen that several network requests have
been made and that the system’s memory was mostly used by JavaScript’s objects and functions, as well as by all other memory that belongs to the “Pages” category and includes information about the DOM, styles, memory caches etc. [29]

Figure 47: Evaluation results based on advisor’s activity

Finally, we also used the website in the way committee members are able to utilize it, by viewing all available topics and producing a recommendation about the allocations of topics to the university’s students. The results of these activities appear in the screenshot below. Again, several network requests have been made by the website in order to communicate with the system’s database. Most of the system’s memory usage appeared once more at the JavaScript and Page categories, with JavaScript usage corresponding to a higher percentage this time.

Figure 48: Evaluation results based on committee member’s activity
5.3 Popularity of the system

The ultimate system evaluation would be made by actual users that would use the system and assess its overall performance. Over time, we would also be able to identify the popularity of the system among potential users. However, given the time-constraints of this dissertation, we are not in position to evaluate the popularity of the system at this point. Nevertheless, in the future we could evaluate the developed system by having actual users utilize it and offer their feedback and suggestions about how to further improve the system, so that it could be more useful for them and better meet their needs.
6. Conclusions and future work

This chapter presents the conclusions and final thoughts after the completion of this dissertation. Moreover, it offers ideas and directions for potential future work on the system.

6.1 Conclusions

The major objective of our work was to successfully design and develop a dissertation recommender system, involving a mobile and a web application. Our system is targeted to universities, considering as target users the students, advisors and dissertation committee members of such educational institutions. We used as a case study the dissertation selection and assignment process that is followed by the International Hellenic University and our aim was to offer relevant recommendations to the system’s users, in order to automate this process as much as possible and make it faster and more efficient.

Sophisticated algorithms have been designed and developed in the context of this dissertation to derive the required recommendations. We have given a detailed presentation of these algorithms in chapter 4 of this report. Their implementation enables the recommendation of dissertation topics to the university’s students, as well as the recommendation of the most suitable students for each topic that is being proposed by the university’s advisors. Finally, through the implemented algorithms, our system is able to offer recommendations to the university’s dissertation committee members, regarding assignments of topics that would contribute to highly satisfied students. In this way, complex decisions can be made quickly with the support of an automated system that is taking into consideration various factors that are usually examined manually, in ad hoc manner, by the system’s users.

The most valuable contribution and outcome of our work is the creation of a trustworthy, fast and efficient dissertation selection and allocation channel that provides scientifically proven recommendations, while analyzing multiple factors. Our system supports the selection of dissertation topics by university students and it can assist advisors and dissertation committees into making faster and more efficient decisions about the assignment of topics to the students of their institutions. By evaluating the system on real data that have been provided to us by the
International Hellenic University, we were able to verify the fact that the developed system is already able to offer allocation recommendations that are valid and utmost aligned with the preferences of students.

Our ambition for the developed dissertation recommender system is to see it embedded into the existing systems of the International Hellenic University, as well as into the systems of other universities that would be interested in enhancing their dissertation selection and assignment mechanisms. The system has been designed and developed in such a way that appropriate parametrizations could enable its incorporation to the systems of a wide-range of academic institutions. If successfully promoted to the universities’ students and academic staff, the capabilities of the system could be utilized to their full extent.

For the successful application of our system in real-life scenarios, adequate information and incentives need to be offered to potential users, so that they would be willing to share with the system the accurate information that is needed for its successful function. Since privacy concerns might arise due to the types of data that are handled by the system, we believe that appropriate security and privacy measures should be taken in order to successfully address such considerations before applying the system in real-life situations.

### 6.2 Future Work

Although the proposed dissertation recommender system is already able to offer the contributions that are mentioned above, we believe that additional work can be made in order to further evaluate the existing system and extend its functionalities, while generating added value in view of a potential commercial exploitation.

Particularly, it would be very useful to get the feedback of actual users about their experience with the system. This would help us create a better user experience for them. Also, based on their proposals we would be able to expand the recommendation criteria that are used in the system, further optimizing the dissertation selection and assignment process. At the same time, the actual use of the application would be most informative for evaluating the scalability of the system, especially if it is to be employed by universities with a big number of potential users.

Throughout this dissertation and particularly in section 2.4, we have mentioned several examples of potential system extensions that could be part of future system releases. We believe that if the
system is to be commercialized, it would be crucial to thoroughly consider the security and privacy principles that need to be followed and implemented. Besides, it would be important to implement an Android version of the mobile application and to ensure that all types of users would be able to use both the mobile and the website components of the system. Moreover, providing the ability to advisors to define their own tags when they need to characterize their topics would be one of our top priorities, as the case is with enabling students to have an open set of tags when they want to insert specific interests to their profiles. In this way, through the use of machine learning, we could provide even more enhanced recommendations to the users of the system.

Specifically for the deployment of the system in existing universities, we believe that administrative roles of users would need to be implemented, in order to manage the insertion of general information in the system, like for example the set of predefined tags, and in order to ensure its proper overall operation. By parametrizing the proposed system and embedding it into the current systems of IHU or any other university, students and academic staff would be able to login to the system using their existing credentials and to utilize the developed web and mobile applications.

Overall, we wish to further contribute to the enrichment and improvement of the system in the future, in order for it to be fully trusted when used as a main dissertation recommender system by universities.
Appendices

A.1 Mockups for website & mobile application

Hereby we present the mockups that have been created before the development of the proposed system.

A.1.1 Website Mockups – Advisors

The mockups that have been designed for the advisors that would be using the system appear in the following screenshots.

![Figure 49: Login page - Advisor Mockups](image)
Figure 50: Topics Page - Advisor Mockups

Figure 51: Add Topic Page - Advisor Mockups
A.1.2 Website Mockups - Dissertation Committee

The mockups that have been designed for the dissertation committee members that would be using the system are the following.

Figure 52: Topics Page - Committee Mockups

Figure 53: Topics Suggested Allocation Page - Committee Mockups
A.1.3 Mobile Application Mockups – students

Finally, the mockups that were designed for the mobile application and the university’s students, can be seen in the screenshots below.

Figure 54: Login Screen - Mobile App Mockups

Figure 55: Course Rating Screen - Mobile App Mockups
Figure 56: Topics Screen - Mobile App Mockups

Figure 57: Profile Screen - Mobile App Mockups
A.2 Licenses for icons used during the development process

Following, we mention the icons that have been used throughout the development process and we offer the licenses they came with.

Icon made by icomoon from www.flaticon.com

Icon made by Smashicons from www.flaticon.com

Icon made by Freepik from www.flaticon.com

Icon made by Freepik from www.flaticon.com
References


