The University of Sheffield
International Faculty
CITY College

DEPARTMENT OF COMPUTER SCIENCE

TSBU: The Syllabi Badges Utility

Petros Grammatikopoulos

July 2020

The University of Sheffield
International Faculty
CITY College

FINAL YEAR PROJECT

TSBU: The Syllabi Badges Utility

This report is submitted in partial fulfillment of the requirement for
the degree of Bachelor of Science with Honors in Computer Science by

Petros Grammatikopoulos

July 2020

Approved

Grade

____
Ms Anna Sotiriadou

____
Prof. Petros Kefalas

TSBU: The Syllabi Badges Utility

by

Petros Grammatikopoulos

Supervisor

Anna Sotiriadou

Abstract

This dissertation covers the study, conducted research and implementation of a system the author wanted to develop for the university. More specifically, starting from the proposition of recognising the Student Graduate Attribute representing badges, that are contained in our units' syllabi, a complete and expandable system to run previously completely manually created statistics about the units was developed to possibly aid the staff of the faculty and make the various programmes' formation more open to the students.

DECLARATION

Name (block capitals): PETROS GRAMMATIKOPOULOS

Signed: .... Date: ....

Acknowledgements

With this section, I would first like to thank my parents, who supported me throughout this journey in my academic career and gave me all the courage to keep going. Likewise, there are no fitting enough words to express my gratitude towards my peers, whom now I see more like friends rather than colleagues. We have been through some challenges, we had fun all along, we even fought to reach a better result in specific occasions. Our interaction the last 3 years will be unforgettable and formative to our new beginnings professionally. We could all agree though that our lecturers, mentors and supervisors guided us towards a rightfully well paved academic path and pushed us enough to achieve more. I stand amazed thinking of their openness and support throughout my studies, something that made me appreciate higher education a lot. Finally, a well deserved thanking needs to be passed to all the staff of the faculty, from the ILC facilities members to the cleaners, that all did their best to give us the most focused and helpful experience we could possibly have as students, removing all the challenging bits off our minds during our studies.

Contents

1 Introduction **1**

1.1 Project Description 1

1.2 Project Rationale 1

1.3 Project Aim 2

1.4 Objectives of the dissertation 2

1.5 Contents of the report 3

2 Functionality of the System **4**

2.1 An abstract idea of the system 4

2.2 Main user goals 5

2.3 Secondary features 6

2.4 Administration features 6

2.5 User roles 7

2.5.1 Visitor 7

2.5.2 Elevated user 7

2.5.3 Administrator 8

2.6 Use cases 8

3 PDF document parsing methodology **10**

3.1 Badge Recognition 11

3.1.1 A simplistic idea of an algorithm 11
3.1.2 Problems in badge recognition & their solutions 12
3.1.3 Results 14
3.2 Unit information parsing 14
3.2.1 Briefing of the case 14
3.2.2 Parsing Algorithms 14
3.2.3 Effective parsing 16

4 Structure of the System 17

4.1 Separation of concerns 17
4.1.1 Initial development 17
4.1.2 User-centered design 18
4.2 The clientside system 19
4.2.1 Utility execution 19
4.2.2 Handling of Documents 21
4.2.3 Handling of Images 22
4.3 The online system 23
4.3.1 Knowledge repository 23
4.3.2 Public sections of the interface 24
4.3.3 Elevated permission sections 28
4.3.4 Personal information & Authentication solution 33
4.3.5 User experience and Design decisions 34

5 Testing and Evaluation 39

5.1 Testing 39
5.1.1 Black-box testing 39
5.2 Evaluation 40

5.2.1 Project evaluation 40
5.2.2 User evaluation 40

6 Future work and how it is envisioned **41**
6.1 Bug in text extraction 41
6.2 Expansion of system 41
6.2.1 Master Programmes 42
6.2.2 Addition of badges 42
6.2.3 Addition of statistics 43
6.2.4 Operating system portability 43
6.2.5 Export functionality for graphs 43

7 Conclusions **45**

Bibliography **47**

List of Figures

2.1 Abstract system 4

2.2 Visitor Use Cases 8

2.3 Elevated user Use Cases 9

2.4 Administrator Use Cases 9

3.1 Visual representation of the PDF parsing procedures 10

3.2 An example of a badge's features. 11

3.3 Real comparison example 13

3.4 Real comparison example 2 13

3.5 Word-to-Word algorithm visual representation 15

3.6 Word-to-Delimiter algorithm visual representation 16

4.1 The execution of the utility in CLI mode 20

4.2 The simple execution of the utility 20

4.3 Ending output of the utility 21

4.4 Log of an execution 21

4.5 The schema of the knowledge repository 24

4.6 The categories of the system 25

4.7 The filters of Academic Year Stats 25

4.8 A snippet from the Academic Year Stats for a level of studies. 26

4.9 Choice of distribution 26

4.10 A snippet from the badge distribution based on pathways. 26
4.11 Choice of pathway 27
4.12 A snippet from the badge distribution based on levels of study. 27
4.13 An unpopulated badge sum graph. 28
4.14 A populated badge sum graph. 28
4.15 Header of interface without elevated user permissions 28
4.16 Header of interface with elevated user permissions 29
4.17 Instructions on what needs to be downloaded 29
4.18 Instructions on how to execute the utility 29
4.19 Example of a populated pathways' section 30
4.20 Creation or modification of a pathway 30
4.21 Unit deletion selection 31
4.22 Password prompt for administration panel 31
4.23 Elevated Users' editor 32
4.24 System reset option 32
4.25 System reset option confirmation 32
4.26 Notifications for missing badges 35
4.27 A different range of colours to that of figure 4.14 36
4.28 Example of scaling on mobile devices 36
4.29 A graph with more data scaled for mobile devices 37
4.30 Information tool-tips for graph details 38

Chapter 1

Introduction

This chapter serves as an introduction to the overall project. The project's description, rationale and aim are explained thoroughly.

1.1 Project Description

The International Faculty of the University of Sheffield complies with TUOS in its learning and teaching strategy. The top priority of this strategy is the Programme Level Approach [1]. Part of how this strategy is implemented is the introduction of the Sheffield Graduate Attributes. These constitute a system of validation of the fact that, the students, during their studies, developed specific skills employers would be interested in, how they developed them and when in their studies. The International Faculty has implemented an abstraction layer on top of this, which incorporates badges that represent those graduate qualities. Those are included in each unit's syllabus, to make it even more useful for the overall experience of the students, as well as to provide a sense of contiguity in their studies, helping them recognise what aspects of their employable persona are enhanced, when and why, in the specific unit structure and programme as a whole. The software that should be produced during this project is an enhancement of how the administrative board keeps track of the graduate attributes, in the form of badges, for each unit and a concrete method to validate the quality of education the graduates receive, based on TUOS's standards.

1.2 Project Rationale

The issue with the present solution of the syllabi badges is the fact that the whole process of identifying the unit attributes based on the badges is performed manually by the Administrative staff. To elaborate, their current responsibility is to gather all the syllabi proposed by the lecturers, get to the badges section, note down what badges appear on which units and then manually orchestrate statistics based on this data. This is a lengthy and tiresome procedure and there is room for error or even changes in the syllabi altogether. Therefore this project is introduced on top of that procedure, to eliminate some of the time needed organising all that data and help the users of the eventually produced system solely focus on what is important: A more suitable strategy to achieve a holistically beneficial result for the students' Programme. The interesting aspect of this project is the way this could be implemented so that its users can achieve more in a lot less effort than before and possibly open new paths of flexibility in their work-flow and making room for the introduction of more useful units for the students.

1.3 Project Aim

The course administrators of the faculty carry out the process of running statistics on the units' badges on spreadsheets each year, completely manually. As the badges are associated with certain Sheffield's Graduate Attributes, this procedure aids the Programme Level Approach [1]. The project's aim is to make the manual procedure the administrative staff is currently following fully automated, meaning that both the whole process of finding out what badges exist on which units, as well as the statistics themselves, need to be completely carried out by the system alone. This will be done by parsing the PDF documents of the syllabi and acquiring all the needed information. Using this system has to be less challenging and more efficient than the currently followed manual method, so that it opens the possibility for the administrative staff to focus on more important issues, regarding the holistic vision of the studies, instead of information gathering, organisation and presentation.

1.4 Objectives of the dissertation

The objectives of the project as a whole are the following:

1. It has to be more effective and efficient to use than the manual procedure statistics have been carried out with as of now.

2. It has to make the Programme Level Approach our faculty follows more open to the students, giving them greater insight over their studies.

3. The information the system carries has to be easily accessible by all kinds of devices (computers or mobile devices).

4. It has to be a solid foundation/basis for more functionality to be implemented in the future.

1.5 Contents of the report

This report starts with a briefing on the general functionality that the system is expected to incorporate, for the different kinds of users that access it. After that, the two main areas the system excels at (image recognition and textual parsing of PDF syllabi documents) and the algorithms behind them are presented. Subsequently, the implementation details and decisions of the whole system are analysed. The final two chapters of the dissertation's main body are dedicated to the future work, that the author believes would complement the system appropriately, as well as a briefing on the testing and the evaluation of the project. The conclusions of the author follow and conclude the paper.

Chapter 2

Functionality of the System

This chapter demonstrates the functionality of the system. This means that the contents of this chapter are concerned with what the system achieves to do as a holistic entity. This chapter is not concerned with the design decisions that took place during the development of the system. This chapter will also include the information about different user roles and what they can do. Diagrams of several use cases are present.

2.1 An abstract idea of the system

The solution the system introduces is mostly geared towards the course administrators, as it automates their work regarding the statistics that are carried out with the badges, which are included in different units' syllabi documents. Therefore the process the system tries to abstract is that depicted in the following figure (2.1):

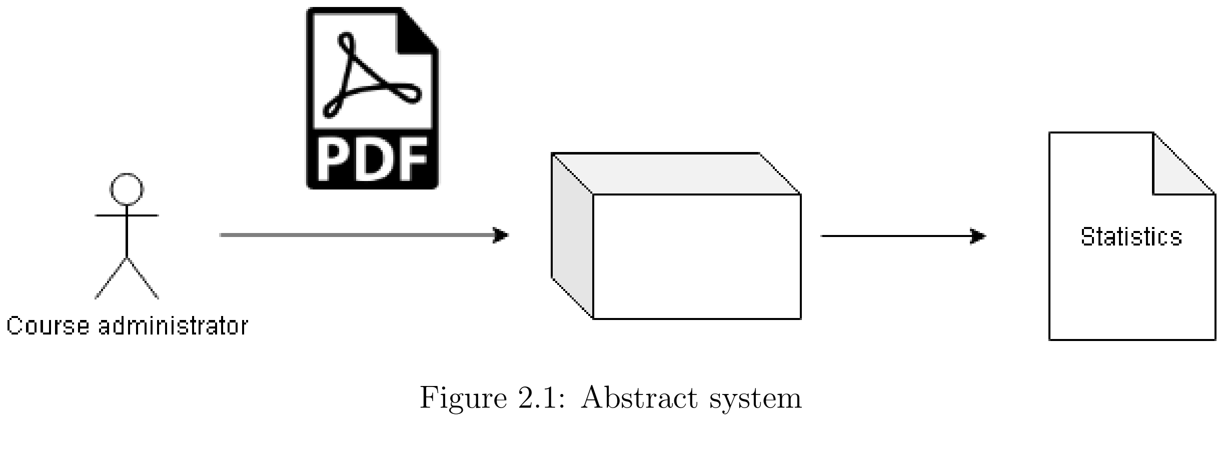


Figure 2.1: Abstract system

What figure 2.1 depicts is the following:

1. A course administrator inputs a PDF document to the system.

2. Some procedures take place internally.
3. The statistics are produced for the course administrator to view.

This is the highest level of abstraction of the system's operation, for the reader to understand its main utility. But moving away from that, this chapter is not dedicated to the utility itself, but rather the different kinds of functionalities the users that access the system have to be able to accomplish while using it, as well as who these users are.

2.2 Main user goals

As described in the introduction, the system tries to accommodate some specific needs the faculty's course administrators had before its development. More specifically, every year, after the units of the programmes were settled, course administrators ran statistics on the appearance of graduate attributes -in the form of badges- in the syllabi. This process took place manually with the creation of a spreadsheet, where all the badges were represented by a column and all the units by a row. This spreadsheet took into consideration the different undergraduate levels of study as well as the different paths the students would be able to choose from, if applicable. Paths, when applicable for a department, are the different groupings of specialization a student may choose to undergo during their final third year, which in turn affects their degree specialization title. A typical example from the Computer Science department would be that of the general path, the internet computing path and the business informatics path.

From all this information, four major categories of charts were devised by the staff to keep track of the balance, quality, engagement factor as well as usefulness of specific units being placed at a specific time during programmes:

1. The sum of badges per unit.
2. A distribution of the number of appearances of badges according to an academic year session (Autumn or Spring).
3. A distribution of badges based on the academic pathway, just for the third level of studies of a department.
4. A distribution of the overall badges across all three years of undergraduate studies, based on a single academic pathway.

The system, therefore, has as its ultimate goal the automation of the creation of all those graphs respectively. When a user of the system wants the same information, they have to somehow input the information derived from the syllabi to the system and then the system carries out the process they would normally do, to generate the above statistics. This is successfully achieved with the system having an online repository that is updated by the course administrators of the faculty and the rest of the procedures are solely carried out by the system itself. The online interface that the system incorporates serves as a visual representation of all the needed statistics, upon request.

2.3 Secondary features

While incorporating all the aforementioned, that are targeted mainly towards the staff of the faculty, the system always had as a purpose to make the whole procedure more open to the students. What the author realized after talks with fellow classmates was that, even though the most well informed peers already knew the purpose of the existence of the badges in syllabi we received, and how they relate to Sheffield's graduate attributes, they did not know anything about their very specific use, when taken into a holistic consideration by the administrative staff of the respective departments. This gave rise to the motivation of making the utility open to access for anyone in the faculty. In this way the students can have the chance to view those extra statistics that the faculty carries out every academic year and realize how the units are actually formatted to have the timing they do. With these stated, the author thought of two extra features that could help both staff and students alike, during their navigation of the online interface of the system.

Badges of Unit: This section of the interface is an "added" section, where the user can find a specific unit of a department and view the badges for that specific unit. This serves as a reference point of viewing the badges, without the need of going through the syllabi to find them.

Missing badges from distribution graphs: At the bottom of every distribution based graph, with this concerning the distribution based on academic year, academic year session or pathway, if a badge is actually missing from the graph, it is depicted with an informative message. This both means that the graph is not clouted with zero values but also that there is a very straightforward way for academics and students alike to spot that some part of the Sheffield Graduate Attributes is not adequately met, for a certain context during their studies, and possibly reach solutions quicker than they normally would have in the past. As an example, if for the whole 3rd level of the Internet Computing pathway of the Computer Science department there exists no unit that satisfies the Ethical attribute of a Sheffield Student, it is depicted clearly at the bottom, for students and academics to take action and propose overall beneficial solutions.

2.4 Administration features

It is obvious that, like every system, the system developed as part of this dissertation has to enable easier administration for its years in service while deployed. For that reason the author made sure to also include a sub-section in the overall online system, from where the systems' administrator of the faculty can perform two functionalities that are essential for the maintenance of the system. These two functionalities are those of assigning and revoking elevated permissions to the course administrators, that maintain the information in the system, as well as a reset functionality, that totally resets the system to a just initialized state, as if it just got deployed. For more information on user permissions the reader is prompted to 2.5.

2.5 User roles

This section enlists all the user types of the system and general information about them. It also includes their access privileges.

2.5.1 Visitor

A visitor of the system is a user that has the permission of viewing and navigating the system. That user is internal to the faculty, as they have to use a faculty account to log into the system. For more information on accounts and personal data, the reader is prompted to 4.3.4.

The specific kind of user can:

1. View the badges of a specific unit.
2. View the sum of badges for each unit, per department and level of studies.
3. View the distribution of badges as per learning session (Autumn or Spring).
4. View the distribution of badges among different third level pathways.
5. View the distribution of badges across all academic years based on a third level pathway they are interested in.

2.5.2 Elevated user

This user is a course administrator that has all the permissions of a Visitor but they also have access to the manipulation of the system's contents. This in detail means they have access to the utility that updates the system with syllabi and badge information, with instructions on its usage, they have access to the creation, modification and deletion of pathways for a department and finally, they have access to managing the units on the online platform, deleting units that are inaccurately represented.

The specific kind of user can:

1. Do all the actions a Visitor user role can.
2. Create, read and delete units of the system (CRD units).
3. Create, read, update and delete pathways of the system (CRUD pathways).
4. Access the clientside utility and the instructions on its usage.

2.5.3 Administrator

The Administrator has all the permissions an Elevated user has with the added benefit of access to the deployment solution of the online system, therefore knowledge of the password that is needed for the administration panel to be accessed. The main role of the administrator is that of resetting the system, as well as assigning and revoking Elevated user permissions to specific accounts.

The specific kind of user can:

1. Do all the actions an Elevated user role can.
2. Create, read, update and delete Elevated users (CRUD Elevated users).
3. Reset system to its initial state.

2.6 Use cases

The following figures visually represent the use cases that were mentioned in this chapter, and serve as a point of quick evaluation reference.

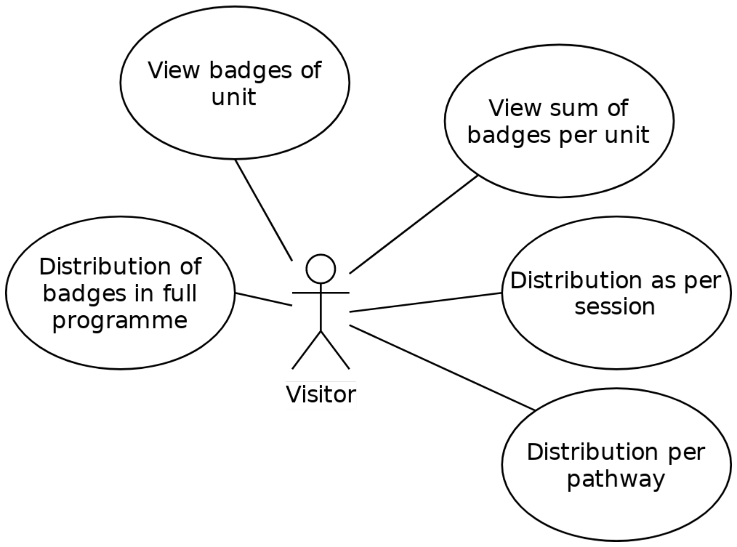

View badges of unit
View sum of badges per unit
Distribution of badges in full programme
Distribution as per session
Visitor
Distribution per pathway

Figure 2.2: Visitor Use Cases

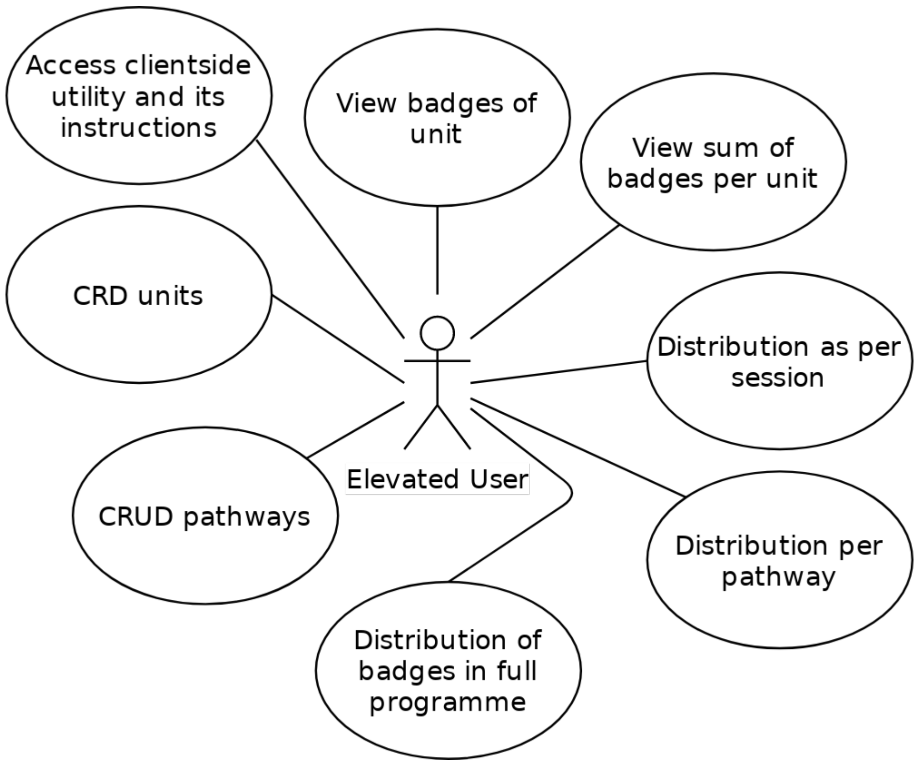

Access clientside utility and its instructions
View badges of unit
View sum of badges per unit
CRD units
Distribution as per session
CRUD pathways
Elevated User
Distribution per pathway
Distribution of badges in full programme

Figure 2.3: Elevated user Use Cases

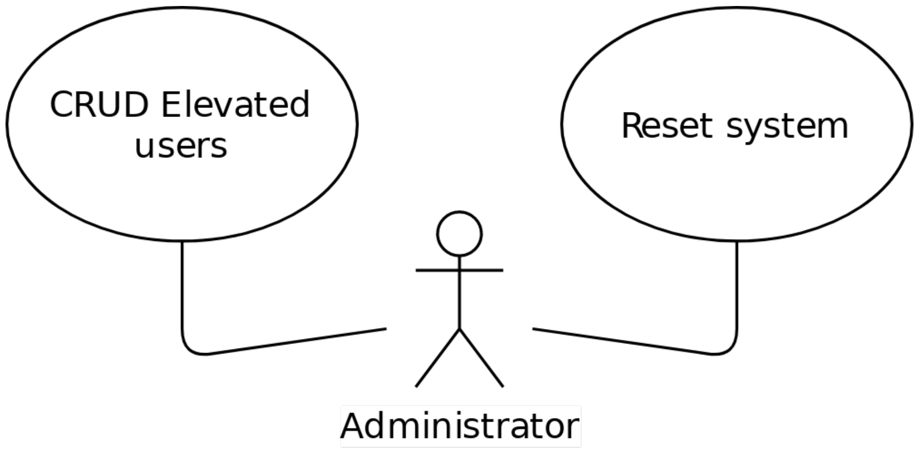


Figure 2.4: Administrator Use Cases

Chapter 3

PDF document parsing methodology

This chapter goes into the details of how the two main algorithms of the utility got conceptualised and work. Those two main algorithms are

1. the image recognition procedure, which is responsible for recognising the badges inside a syllabus document
2. the textual parsing of the document, which extracts all the information about the units, to associate them with their badges

Both procedures take place with PDF documents as an input.

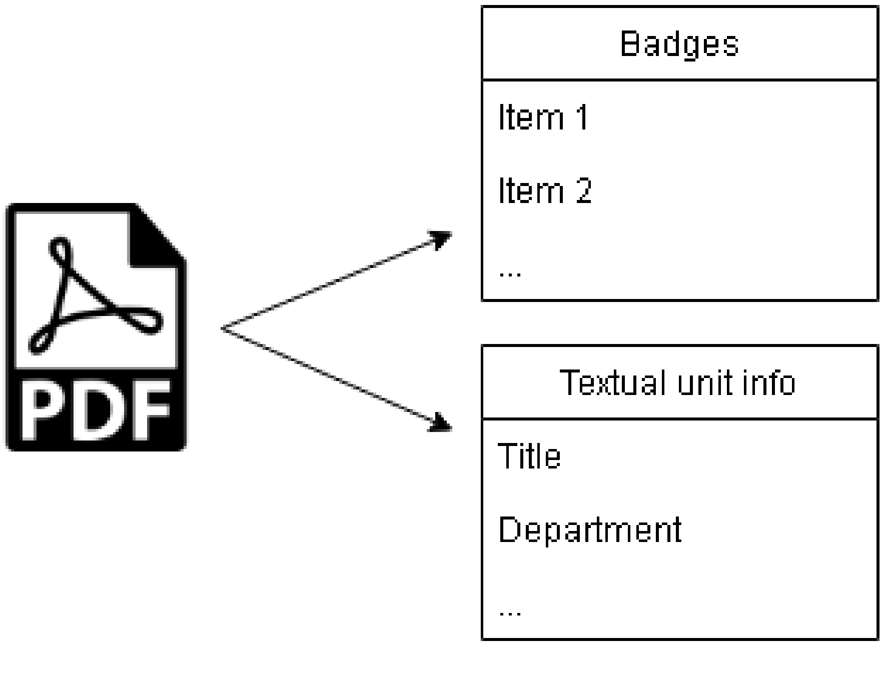


Figure 3.1: Visual representation of the PDF parsing procedures

3.1 Badge Recognition

Before the report gets deeper into what the badge recognition algorithm of choice was and how it was implemented, we have to examine the task at hand. Badges are images that are embedded into the syllabi. They have specific color schemes as well as distinct silhouettes. They are also supposed to be of equal dimensions and they all follow the principle of a circle as the base shape. There are 17 unique badges.

With the above information, upon inspection, the author immediately realized that those images follow a very specific pattern, but they are unique nonetheless, because of their distinctive characteristics both color-wise and content-wise. That subsequently made the need for an intelligent algorithm redundant, as there is no sophistication in recognising what is a badge and what is not, because they all have the same base shape. Therefore, as long as all badges have the same shape and as long as all of them are distinct, that means that their raster representation differs, but can still be classified as a representation of a badge. To make matters clearer, if an image in the PDF document has the shape of a badge, it can immediately be classified as one. After that, the whole recognition procedure can be focused on the internal differences of the badges.

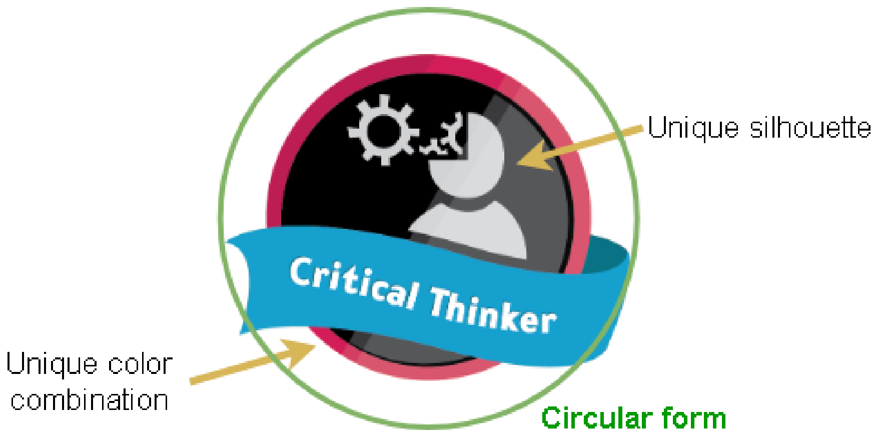


Figure 3.2: An example of a badge's features.

3.1.1 A simplistic idea of an algorithm

During their research, the author came across the algorithm of raster image comparison. This algorithm compares two images pixel by pixel and returns the percentage of difference [2]. After some experimentation and benchmarking, it was evident that this algorithm, with a very small footprint and with adequate performance, was a solution perfectly suited for the problem at hand. One could have the images of all the badges as assets in the backend and then the algorithm would work as follows: The utility would extract all the images from the PDF document as temporary files, compare them one by one with all the asset images in the backend, calculating all the percentages and, finally, if a percentage is lower than an arbitrary threshold, it would confirm that each respective image corresponds to a certain badge. It has to be noted that this arbitrary threshold would be chosen by the developer themselves after trial and error and optimization for the best results. So this threshold can otherwise be considered the bias level of this algorithm.

3.1.2 Problems in badge recognition & their solutions

The extraction of images from syllabi needed to be a high-performance process, to trim down execution time overall. The author was already aware of a utility that performs that operation in bulk. That utility is pdfimages by Glyph & Cog LLC [3]. Therefore the source code of the system would spawn instances of that utility per PDF document and extract all the images in PNG format.

The first problem with this would be that of size and color. Different syllabi might include the same badge images in different resolutions and/or dimensions. The colorspace might also differ among syllabi in 3 ways: Optimized PDFs might have compression artifacts on the images, in the case the images are lossy, limited color space and some might even include transparency layers around the badges, since PNG supports this as a format. To overcome this issue, the author implemented a solution using the Ffmpeg multimedia manipulation library [4]. The solution would be that of, after extracting all the images from the syllabi, an automatic procedure, which would convert all of them into GIF images of 200x200 dimensions. This would ensure 3 things:

1. The images would all be of the same color space (256 colors).
2. The images would all have the same dimensions.
3. The resulting files would be smaller due to the GIF format's compression, further lowering the overhead of the image comparison algorithm [5].

A final problem, though, still had to be combated: PNG images support transparency and while testing with various syllabi, the author found out that some word processors export PDFs preserving that transparency. Therefore, for the algorithm to work for such sets of exported images, the author would have to store as assets in the backend of the utility 2 sets of images. Those with the alpha layer present and those without it. And the comparison would therefore need to be executed twice per image, once for the first set of asset images and once for the second. After that, the smallest difference percentage would be kept, as that would be the closest value to recognising a badge. To elaborate: The exported image would be compared with the first and the second dataset of asset images from the backend. On each comparison, the comparison that results in the smallest difference would be the one kept by the algorithm, because that could only possibly be closest to recognising the correct badge. This procedure is visualized at figures 3.3 and 3.4. Finally, since only the smallest difference percentage is kept from each pair of comparisons, the algorithm still ends up with 17 "most likely" difference percentages, so the last step is to find the one that is below the arbitrary threshold that was described in 3.1.1. There is no way 2 different badges would have a value below that threshold because the raster image comparison is quite strict as a methodology and thus this eliminates the margin for error.

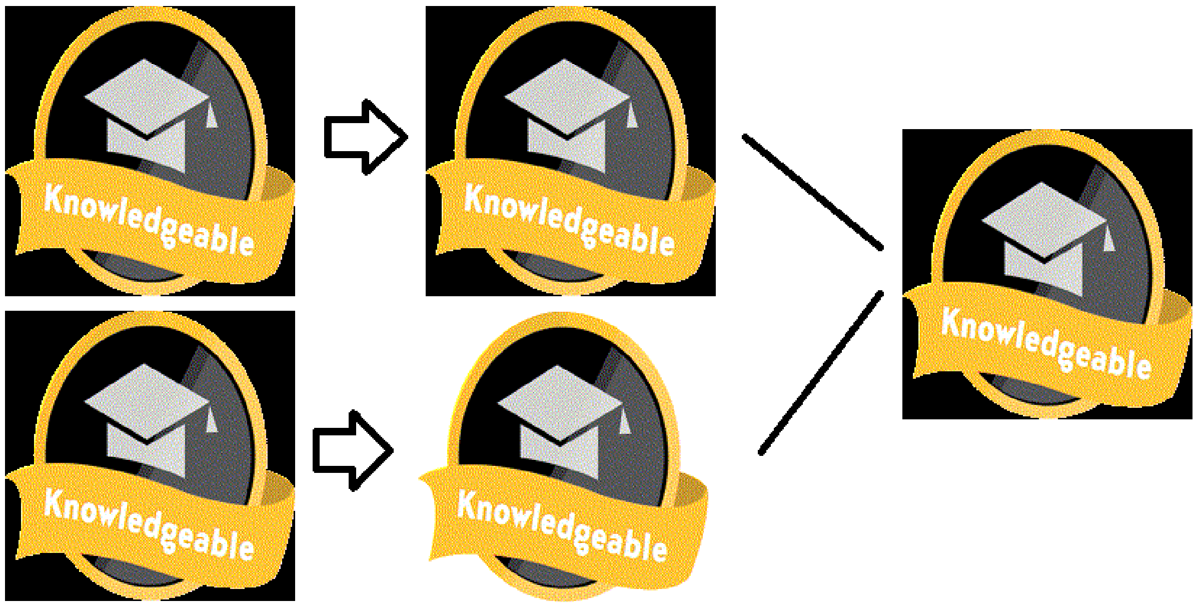


Figure 3.3: Real comparison example
The leftmost images are the exported image, which is then compared to the two types of asset images from the backend. Finally the correct image is picked. In this case we have a match of badges too.

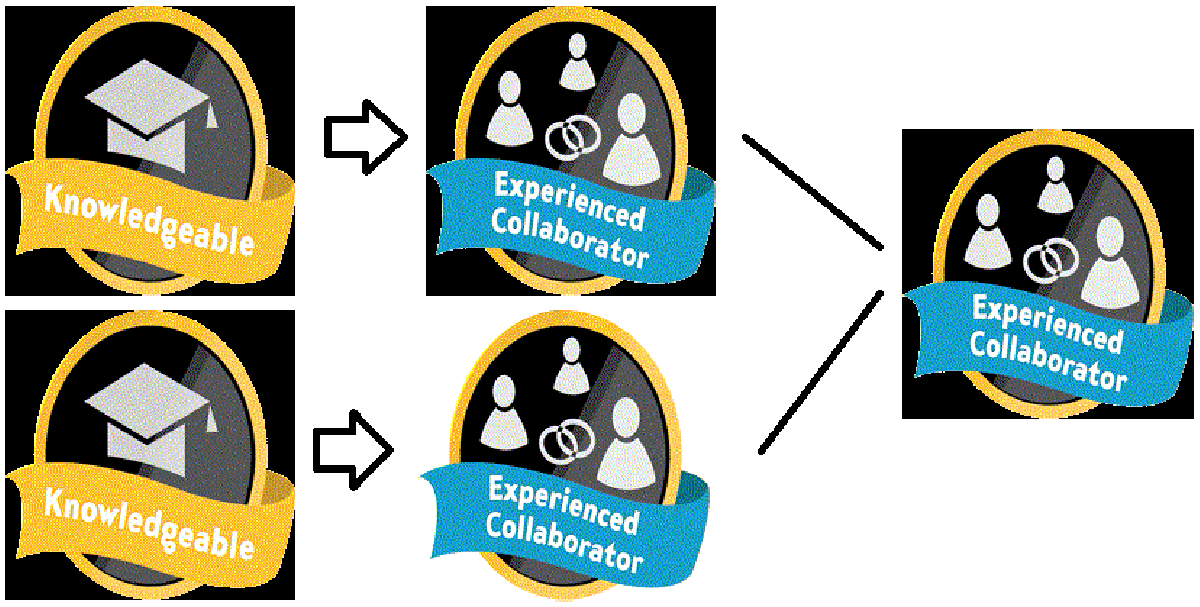


Figure 3.4: Real comparison example 2
In this case it is obvious that we could not possibly have a badge match. But still, the most likely image based on percentage is chosen. After all, the main percentage comparison will happen at a later stage.

3.1.3 Results

Taking all the steps described above into consideration, the final result was an algorithm that has an 100% success rate in recognising the embedded badges of a syllabus document, as all the factors that differentiate them were both pinpointed and exploited for the final result. The success of that algorithm is not only its small memory footprint and adequate performance. It also straightforwardly depends on the number of images existing in a syllabus document. Therefore, for the most typical cases of syllabi structure that were met by the author while testing among syllabi of different departments and professors, the algorithm is both effective and quick in recognising the correct badges. Moreover, because all the source code involved was produced or tweaked by the author, the size of the solution is small compared to more sophisticated image recognition algorithms, which require extensive data models, making it ideal to be downloaded as a local system.

3.2 Unit information parsing

All the information on the textual parsing of the PDF documents of the syllabi.

3.2.1 Briefing of the case

The text that the author cared about in the syllabi was that of the unit information table. This would give them the ability to get the information about the unit a specific syllabus describes, so they could afterwards pair the recognised badges with that specific unit in the system. As the author wanted to completely automate the procedure of inserting units in the system and make it quicker than the existing manual methods (described in 1 and 2.2), the parsing of the unit information was of great importance to the final product.

There is one fundamental problem that made the parsing of the text the syllabi contain difficult. That is the differences the professors have when they produce the mentioned syllabi. The author realized that the table of information is not uniform across syllabi of different professors. While it was already known that the professors create those documents based on some internal faculty template, some of the lecturers, mostly the Computer Science lecturers in particular, had the tendency, as users of the template, to omit fields or rearrange them. While the author managed to find a way to extract all the text of the first page from each document, in the correct order, with pdftotext by Glyph & Cog LLC, there was no hint regarding where any tables exist in the document and therefore no effective way to delimit the text of tables, to pick the values of interest [6].

3.2.2 Parsing Algorithms

The text parsing algorithms the author considered are discussed in this section.

Word-to-Word

The first algorithm the author thought of was that of delimiting based on surroundings. The text parsing mechanism would traverse the extracted text from the first page of the document and find the contents between a pair of words. For example, if there was some cell named “foo” and the next cell was named “bar”, anything in the middle would definitely be the information about “foo”. While this is a straightforward way and would eliminate all the trouble of parsing the text, it also implies that the author would have to arrange with the whole faculty that there would be a single template of a syllabus that all lecturers must use, for the system to work. That would be an extreme request to say the least, especially for a system that would be meant to get its data from all the future professors of the faculty. While conventions and protocols do indeed make the parsing of documents more concrete, we cannot expect that they should be followed that strictly by all future users. Therefore that solution gets immediately affected by the aforementioned problem of the user’s free will in 3.2.1.

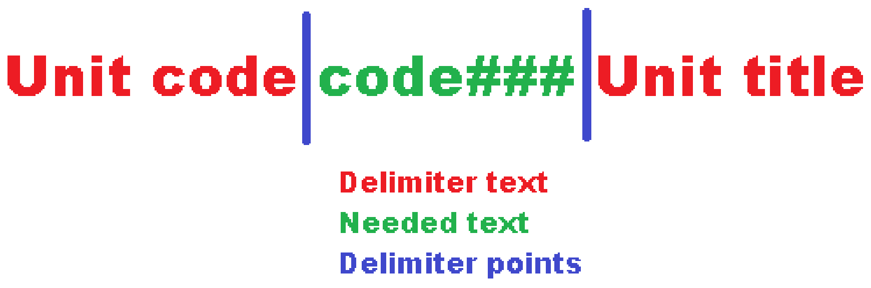


Figure 3.5: Word-to-Word algorithm visual representation

Word-to-Delimiter

Since the author realized that arbitrary text recognition is much harder than it seems, they referred to the design decisions of some older operating systems, before the meta-tagging of data was a thing in the industry. The obvious answer was that all fields had a finite array buffer and the whole buffer was stored as data per data field, after of course being trimmed to save space in more advanced implementations. But that solution was not applicable in the high level of abstraction the extracted text from a PDF has. The main concern the author again came across was that of delimiting arbitrarily long text.

And then the solution emerged: Since table cells are separated with tabulation characters, the extracted text from the document had extra spaces between the cells. Therefore if that spacing got replaced with some delimiting character, that cannot be met in an official legitimate syllabus, the author could pick the information they wanted simply by stating the title of the information and traversing all the way to the next delimiting character. To elaborate, if the information that the author wanted to extract was titled “foo”, the author would only have to program that the variable they want to extract in their source code exists between “foo” and “delimiting character”. For documentation purposes, the delimiting character the author used was the Full Block character from the Unicode encoding system(U+2588) [7]. This character exists in Unicode since version 1.1.0 back in 1993 and it definitely is a character one would not meet in the details of a formal syllabus document, even if that document had international characters, therefore overlapping with potential syllabus text on the first page is eliminated [8].

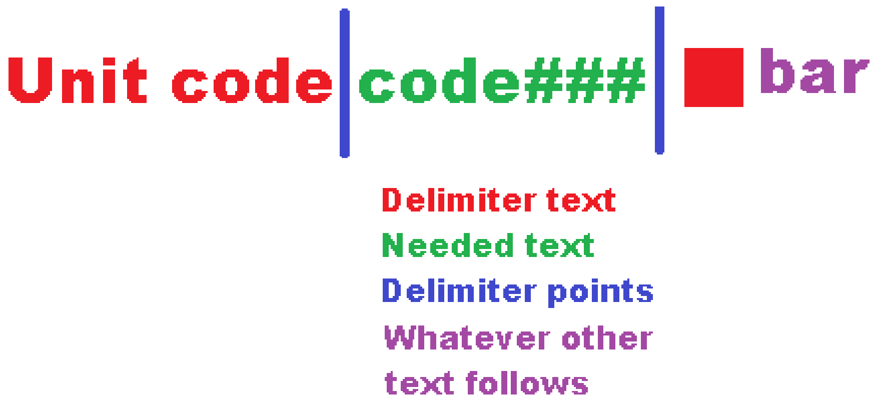


Figure 3.6: Word-to-Delimiter algorithm visual representation

3.2.3 Effective parsing

After the aforementioned details in the entirety of 3.2.1 and 3.2.2, the author noticed a new problem with the information about the units in the syllabi, that had nothing to do with text parsing. The lecturers that created those documents had differences in how they represent the unit’s details. And while this was manageable in the binary format of semester session, where the possible values are Autumn and Spring -and some professors filled in Fall instead, but that is easily fixable-, it was not so manageable to the not so finite ways of expressing the programme or the level of studies the unit was bound to. Therefore the author had a discussion with their supervisor to unfold the details of the only universally commonly patterned values in the document: the unit codes.

After the discussion, the author had a clear view that unit codes follow a specific format. The first three letters represent a department and the first number a level of studies. The codes of the current departments of the faculty are the following: “CCP” for Computer Science, “CBE” for Business Administration and Economics, “CPY” for Psychology and finally “CES” for English Studies. Numbers “1” to “3” are the respective undergraduate levels while “6” represents the master programmes. This information opened the possibility of parsing only the unit code, which is universally formatted the same in syllabi, and then deriving the rest of the information through that code. And this very thing was achieved, with the same certainty of success as that of the badges’ recognition. For more information on the single bug of the above solution, the reader is prompted to 6.1.

Chapter 4

Structure of the System

This chapter contains all the structural details and design decisions of the conceptualised and implemented software system that was produced as a result of the author's work. It is important to mention that no code excerpts are included in the report, as the code for the most important features of the system is too verbose. All source code can instead be found as part of the author's full submission.

4.1 Separation of concerns

This section explains how the system got conceptualised to incorporate two subsystems, one online and one clientside. It also justifies the need to have two subsystems and the advantages that dichotomy brings about to the overall user experience.

4.1.1 Initial development

From the conception of the idea for a system that would automate the statistics course administrators did manually, the author was concerned with the way whatever procedure would be developed would get abstracted and seamlessly "hidden" from the users. More specifically, the system's goal would be that of using the least amount of computational resources, while also achieve the goal quickly. For that matter, the author would have to find the middle ground between the different types of users of the system and how their actions affect it.

The first idea was that of a fully clientside system, that the users would have to download on their machines and then run the statistics individually, by selecting the syllabi they are interested in by themselves. This solution was a very simplistic approach that would only cover the needs of the course administrators. This idea focused on performance and was the basis upon the algorithms for PDF parsing were developed on.

After a few meetings with their supervisor though, the author decided that it would be in the interest of the overall faculty to have an understanding behind the procedures the programmes of the departments are realised. With enough consultation and ideas that emerged from those meetings, the system had to be converted in a more accessible format, that would benefit all users equally. That format of the system would have to be partly online for ease of access for the users that are just viewers of the statistics.

4.1.2 User-centered design

Moving on from the aforementioned, the author had at that point reached halfway through the academic year. There was a solid foundation for parsing syllabi and a very vivid idea of making the system as accessible as possible, to a wide range of users. There was one choice that had to be made that would critically affect the overall progress of the project as a whole. That choice was whether the system would be converted to be fully online, or be separated into two parts, an online interface and a clientside utility to update that interface.

Since the author chose to proceed with a user-centered approach while designing the system as a whole, an analysis of the key advantages and disadvantages of each solution had to be taken into account. Therefore a list of key system features that affect user experience got devised, to assist with picking the best approach. That list included perceived performance, bandwidth consumption, balance of user types and finally ease of hosting.

Perceived Performance: The perceived performance is how “snappy” different operations of the system feel when executed. After minor general benchmarking, the author realised that this would not be very different when comparing a fully online to a two-part system. The added latency of an online system though gave the full advantage to a system that includes a clientside subsystem for heavy operations, like the parsing of syllabi. Additionally, a clientside subsystem could utilize the performance gains of a higher performing programming language while staying simple in its design.

Bandwidth consumption: The major disadvantage of making the system fully online would be that of having to upload the syllabi PDF documents to the system, for the procedure of parsing. That would make the whole process of adding units to the system a lot slower for the course administrators, rendering the system ultimately opposed to its initial objective of a faster automated procedure. The difference of a clientside subsystem doing the parsing work would be the following: When the parsing of the syllabi finishes and all information about their badges is derived, the results can simply be sent as a short textual message, that updates the database of the online subsystem, without the need for any lengthy upload procedure.

Balance of user types: The amount of users that would actually alter the system's contents, which are the course administrators, is significantly lower than the amount of people that would be plain viewers of the generated statistics. Therefore, via the notion of separation of concerns, there is no need for the online part of the system to include all the functionality for its contents to be altered, when this can happen by a separate sub-module, that only a few specific people can have access to. This makes the online system independent and less bloated to some extend.

Ease of hosting: If the system in its entirety was fully online, then the hosting solutions for deploying it would be limited, as the core of the parsing algorithms was developed entirely in Java. This would lead to the negative scenario of being locked down to specific providers, as the options for Java web hosting are limited. On the other hand, having separate online and clientside subsystems, and distributing system concerns in this way, would mean that the online subsystem would be totally independent. This would open the possibility for it to be developed with a very common programming language in the industry, like PHP, and therefore would make its deployment straightforward and compatible with many different environments and providers. Finally, that solution could even be hosted from within the faculty's premises, as the heavy computations would not take place on the servers, but rather only on the machines of those updating the system via the clientside utility.

Taking all the above into consideration, the obvious choice of the author was to continue their project by splitting it into two separate subsystems. The first one would be the clientside system, that updates the stored information automatically, and the second one would be the online interface, which depicts the stored information.

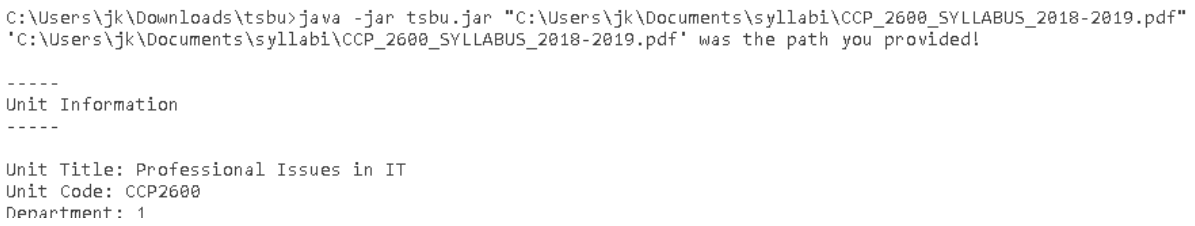
4.2 The clientside system

This section serves as an in-depth analysis of the core as well as the implementation of the clientside subsystem. The utility was developed entirely using the Java programming language and uses third-party binaries to achieve high performance and industry-tested reliability for certain procedures [3],[4],[6]. For that matter, for more information on the compatibility of the utility with various operating systems, the reader is prompted to 6.2.4. As a final note to this small introduction, the reason Java was used for the development of the core utility was a fusion of the author's familiarity with the language, their personal research interest in how it can be used as a scripting language, as well as the performance gains the last point could bring about to such a project, in place of a more common scripting dialect like Bash [9].

4.2.1 Utility execution

The utility's execution may be done in two forms:

CLI mode: The first is that of a CLI (command line interface) utility. If the executable is typed into the command line, along the path of a syllabus PDF document, the whole procedure of recognising the badges of that document, as well as the extraction of all the unit information, to finally update the online subsystem, takes place. This mode of execution was included by the author in an attempt to give the possibility of scripting with the utility, an option for more advanced users. An example of such an execution is depicted in figure 4.1.



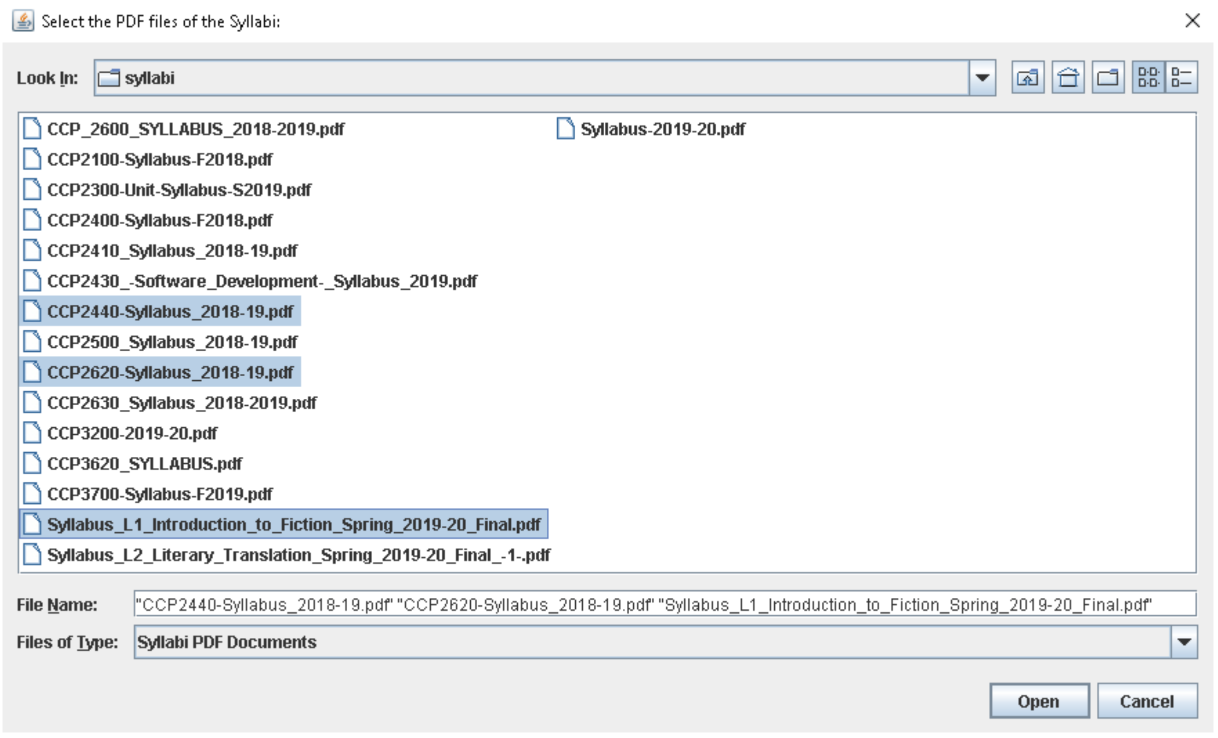
```
C:\Users\jk\Downloads\tsbu>java -jar tsbu.jar "C:\Users\jk\Documents\syllabi\CCP_2600_SYLLABUS_2018-2019.pdf"
'C:\Users\jk\Documents\syllabi\CCP_2600_SYLLABUS_2018-2019.pdf' was the path you provided!

-----
Unit Information
-----

Unit Title: Professional Issues in IT
Unit Code: CCP2600
Department: 1
```

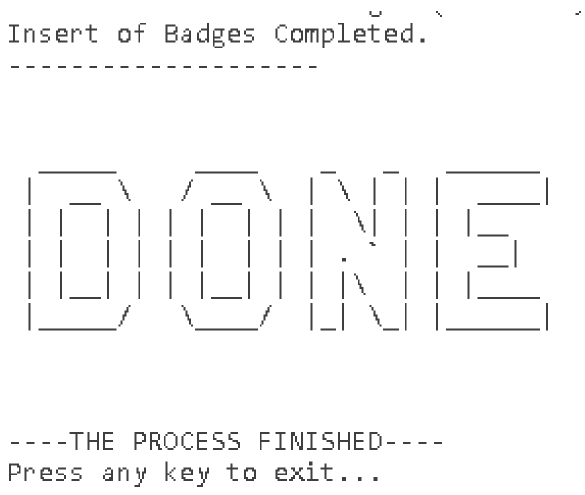
Figure 4.1: The execution of the utility in CLI mode
A cropped example of the subsystem's execution

GUI (graphical) mode: The second and main mode of execution though is the simple execution of the utility. In this case, the user just runs the executable program and a graphical representation of the user's filesystem aids them in picking the PDF documents of the syllabi they want to add to the online subsystem. Then the same procedure, as previously described, runs for all the documents automatically and informs the user when it finishes. This mode of execution can be seen in figures 4.2 and 4.3.



Select the PDF files of the Syllabi:
Look In:
syllabi
CCP_2600_SYLLABUS_2018-2019.pdf
CCP2100-Syllabus-F2018.pdf
CCP2300-Unit-Syllabus-S2019.pdf
CCP2400-Syllabus-F2018.pdf
CCP2410_Syllabus_2018-19.pdf
CCP2430_-Software_Development-_Syllabus_2019.pdf
CCP2440-Syllabus_2018-19.pdf
CCP2500_Syllabus_2018-19.pdf
CCP2620-Syllabus_2018-19.pdf
CCP2630_Syllabus_2018-2019.pdf
CCP3200-2019-20.pdf
CCP3620_SYLLABUS.pdf
CCP3700-Syllabus-F2019.pdf
Syllabus_L1_Introduction_to_Fiction_Spring_2019-20_Final.pdf
Syllabus_L2_Literary_Translation_Spring_2019-20_Final_-1-.pdf
Syllabus-2019-20.pdf
File Name:
"CCP2440-Syllabus_2018-19.pdf" "CCP2620-Syllabus_2018-19.pdf" "Syllabus_L1_Introduction_to_Fiction_Spring_2019-20_Final.pdf"
Files of Type:
Syllabi PDF Documents
Open
Cancel

Figure 4.2: The simple execution of the utility
A graphical representation helps the user to pick all the syllabi on their machine.



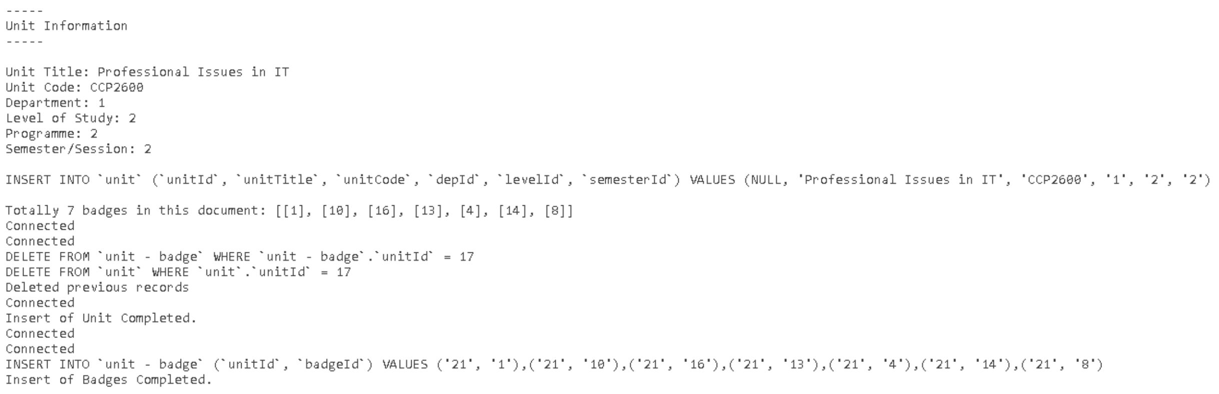
```
Insert of Badges Completed.
-------------------

 _____   ____  _   _ ______
|  __ \ / __ \| \ | |  ____|
| |  | | |  | |  \| | |__
| |  | | |  | | . ` |  __|
| |__| | |__| | |\  | |____
|_____/ \____/|_| \_|______|


----THE PROCESS FINISHED----
Press any key to exit...
```

Figure 4.3: Ending output of the utility
A message that informs the user the execution has ended.

It needs to be noted that both types of execution give a set of outputs to the user. These are informative, to show what the system is working on at each given moment, as well as a debugging console log, which the users can include, when contacting the developers of the system in the future, in case they come across any problems during the execution of the utility. Another important thing to take into consideration is that of contextual changes in the execution of the utility. As depicted in figure 4.4, the unit was already present on the online subsystem. Therefore it got updated with the new information.



```
-----
Unit Information
-----

Unit Title: Professional Issues in IT
Unit Code: CCP2600
Department: 1
Level of Study: 2
Programme: 2
Semester/Session: 2

INSERT INTO `unit` (`unitId`, `unitTitle`, `unitCode`, `depId`, `levelId`, `semesterId`) VALUES (NULL, 'Professional Issues in IT', 'CCP2600', '1', '2', '2')

Totally 7 badges in this document: [[1], [10], [16], [13], [4], [14], [8]]
Connected
Connected
DELETE FROM `unit - badge` WHERE `unit - badge`.`unitId` = 17
DELETE FROM `unit` WHERE `unit`.`unitId` = 17
Deleted previous records
Connected
Insert of Unit Completed.
Connected
Connected
INSERT INTO `unit - badge` (`unitId`, `badgeId`) VALUES ('21', '1'),('21', '10'),('21', '16'),('21', '13'),('21', '4'),('21', '14'),('21', '8')
Insert of Badges Completed.
```

Figure 4.4: Log of an execution
The utility recognises the unit and then the badges. Then inserts the information to the online subsystem.

4.2.2 Handling of Documents

Moving on to more detailed specifications about the clientside subsystem, the whole execution procedure, as hinted previously, is document-centric. This means that the main concern of the subsystem is to receive a syllabus PDF document as an input and give back its conceptual unit, alongside the recognised badges, as an output. This is done within the handler class `Document`. What this class is responsible of is the creation of objects of type Document. Those objects hold all the attributes of a single document. Those attributes are:

1. The path to the document in the user's filesystem.
2. The badges of the document, as a collection.
3. The unit's title.
4. The unit's faculty-given code.
5. The department of the unit.
6. The level of study this unit is taught on(legacy feature).
7. The programme of the unit.
8. The semester this unit is taught within.
9. The ID of the unit on the online subsystem(if it already exists).

These values initially used to be textual and would display all this information on the terminal, for the user to experience how the utility works. The open source version of the utility available on the author's GitHub account was the one that also got presented in the mid-year progress presentations. The textual representation of information got replaced, with the identifier value representation, that matches the information with pre-populated data on the online subsystem. This was done to save bandwidth and processing power, by using less text per updating message from the clientside subsystem to the online interface. Document handling also includes a separate "helper" class, `UnitCodeHelper`, through which the system identifies most of the attributes enlisted previously in respect to the unit code of a unit.

Apart from containing all the information about a syllabus PDF document as an ontology, the `Document` class also includes all the methods associated with operations that are done on documents. That means that all the operations that take place for a PDF document can, intuitively enough, be called from an initialised temporary document object, which also holds the information that will be needed by the operations. The produced data is not stored in any way on the user's system: The primary goal is to formulate the appropriate SQL-based operations to update the remote knowledge repository. So all the data for each document is purged after the utility's execution.

4.2.3 Handling of Images

The handling of images is separated in two parts on the clientside utility, because there are two kinds of images conceptually. The `DataImage` class is concerned with the creation of objects of type "data image". These are the images that reside as assets in the backend of the utility. Every object contains the title of the data image and the two paths for the two different versions of the image, as per the algorithm in 3.1.2. The title attribute got modified to mean the identifying element of the image, as the title already resides on the online remote knowledge repository and sending it over would be excess bandwidth. On the other hand we have the `ScannedImage` class, which is responsible for the creation of objects of type "scanned image". These are all the exported images from the PDF documents, that somehow have to be kept tracked for the algorithms to handle them. There is also an extension of the `ArrayList` java class called `ScannedImageCollection`, which only provides a few macros to make the operations conducted on exported images more intuitive and readable code-wise.

As for the algorithm of image recognition, the functionality is implemented in two classes. The `Scaler` class is responsible for the preparation of exported images, before they undergo the comparison process, as described in 3.1.2. It resizes and encodes the images in GIF format, for them to be valid candidates for comparison with the backend asset images. The comparison and recognition of badges takes place in the `Comparator` class of the utility, with a modified version of the "image difference percentage" algorithm, which facilitates three images instead of two [2]. All the information is fed back into the document object, for it to hold the full collection of badges that got recognised in it.

4.3 The online system

This section serves as an in-depth analysis of the core as well as the implementation of the online subsystem. The online interface, that was developed for access by all members of the faculty, was developed in pure PHP, HTML with HTML5 elements and pure Javascript. The knowledge repository of the system was implemented in MariaDB, a free and open source fork of MySQL [10]. The reasons the author chose those technologies were due to familiarity as well as to keep complexity minimal, as the more dependencies a system carries the more the points of failure become. The system can also be hosted on a wider range of deployment providers this way, as discussed in 4.1.2.

4.3.1 Knowledge repository

The knowledge repository of the system is the only module that both the online and clientside subsystems have access to mutually and it is constantly online. It is the place where all the information about the units is stored. Every unit, as an entity, only contains identifiers for repetitive values, which reference to their complete textual representation. This saves space in the database overall. Another key element that needs to be noted is that of the many-to-many relationships that exist, both for badges with units, as well as for units with third level pathways. This signifies the fact that there are numerous and diverse possibilities for the system that need to be accommodated, as it will scale to include all potential combinations of units, badges and pathways. While the system looks empty with a small, demonstration-geared sample of data, the author estimates that the configuration chosen for the database will be the middle ground for performance and size requirements, with all the units being inserted into the system on a live deployment. The schema of the database is included for reference purposes as figure 4.5.

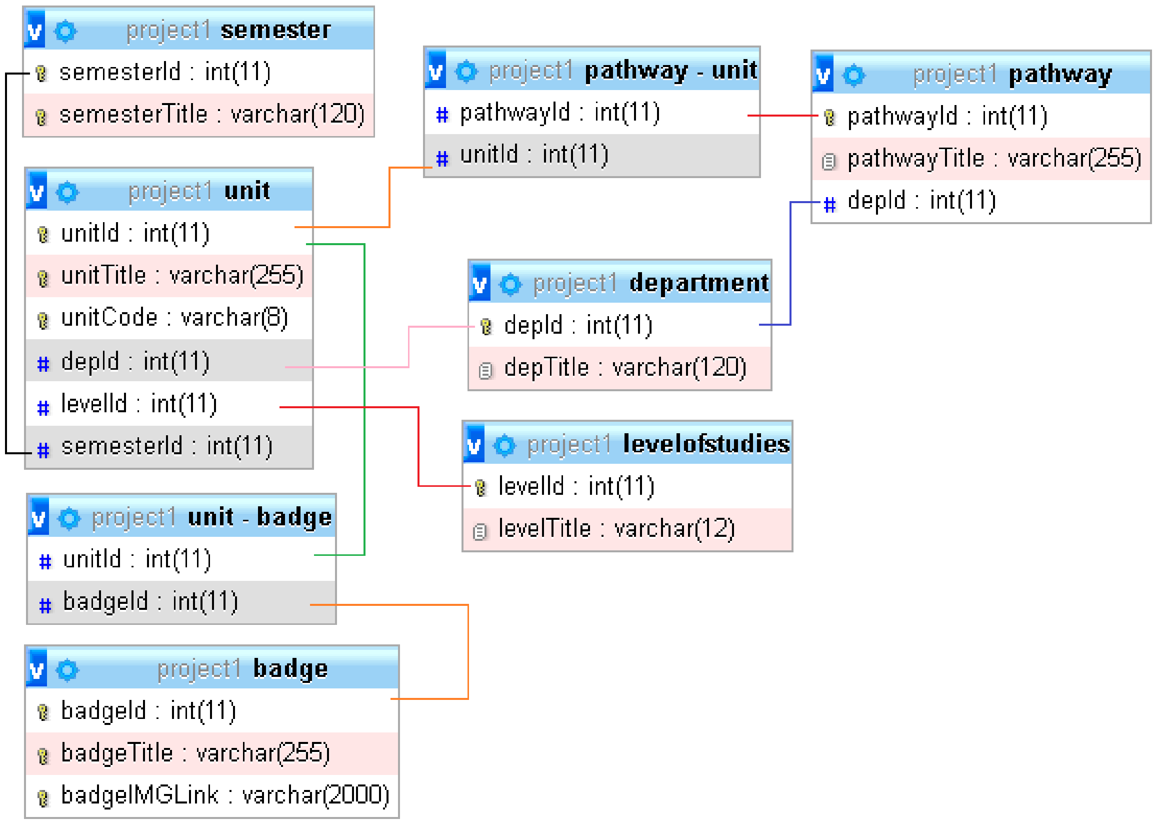


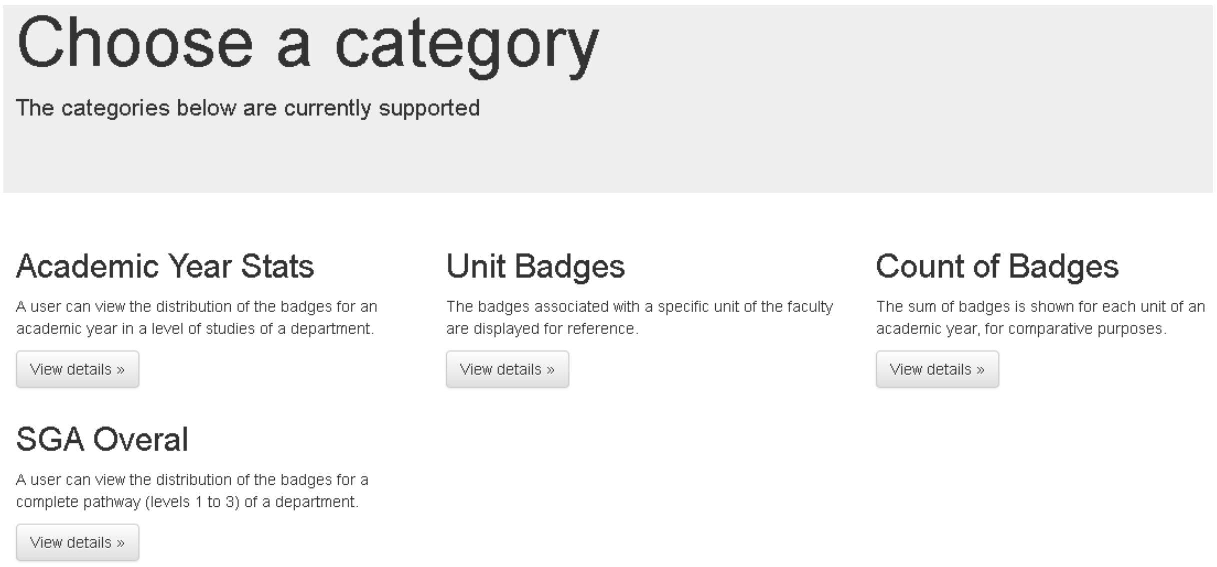
Figure 4.5: The schema of the knowledge repository

4.3.2 Public sections of the interface

The online interface of the system is sectioned, for ease of use. The sections are called "Categories", because they categorise the different kinds of statistics the viewers are able to see. Those sections are available to all users of the system, as they only require the permissions for viewership, as discussed in 2.5.1. The categories included in the current version of the system are the following:

1. Academic Year Stats
2. SGA Overall
3. Unit Badges
4. Count of Badges

Each of the categories has a small disclaimer piece of text under it, that explains what the statistics it includes are, with more precision, to help first time visitors.



Choose a category

The categories below are currently supported

Academic Year Stats

A user can view the distribution of the badges for an academic year in a level of studies of a department.

View details »

Unit Badges

The badges associated with a specific unit of the faculty are displayed for reference.

View details »

Count of Badges

The sum of badges is shown for each unit of an academic year, for comparative purposes.

View details »

SGA Overal

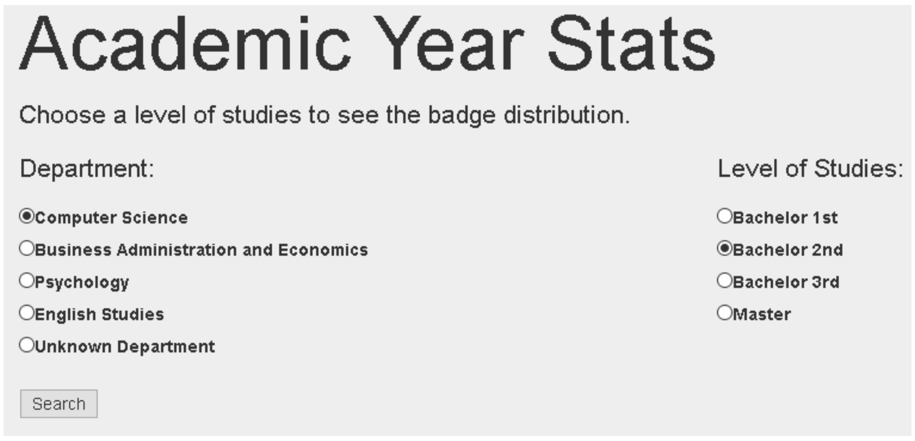
A user can view the distribution of the badges for a complete pathway (levels 1 to 3) of a department.

View details »

Figure 4.6: The categories of the system
A sample from the online interface.

Academic Year Stats

The academic year stats category initiates with the user having to choose the department and level of studies of their preference. Afterwards, they are able to view a distribution of each badge individually across sessions of an academic year. The sessions are Autumn and Spring.



Academic Year Stats

Choose a level of studies to see the badge distribution.

Department:

- [✓ Computer Science]
- Business Administration and Economics
- Psychology
- English Studies
- Unknown Department

Level of Studies:

- Bachelor 1st
- [✓ Bachelor 2nd]
- Bachelor 3rd
- Master

Search

Figure 4.7: The filters of Academic Year Stats

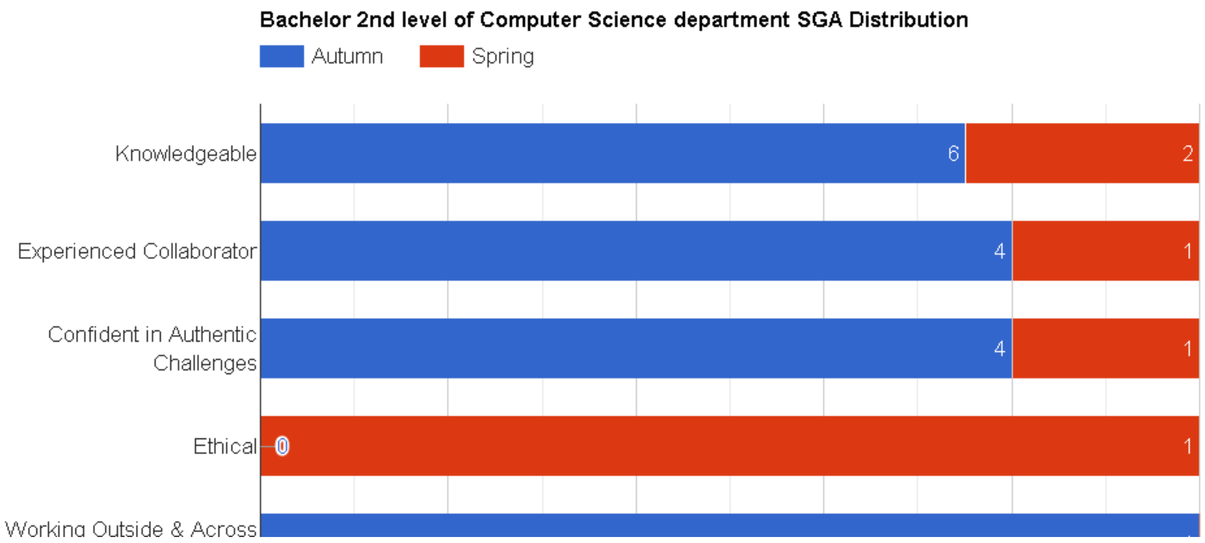


Figure 4.8: A snippet from the Academic Year Stats for a level of studies.

As the third year of undergraduate studies is the year where programmes are separated into pathways, the system handles this occasion differently. The users have the opportunity to choose between the distribution of badges across sessions, or across pathways. The later is depicted in the following figures:

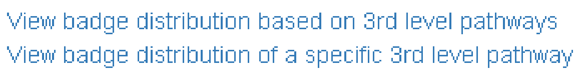

View badge distribution based on 3rd level pathways
View badge distribution of a specific 3rd level pathway

Figure 4.9: Choice of distribution

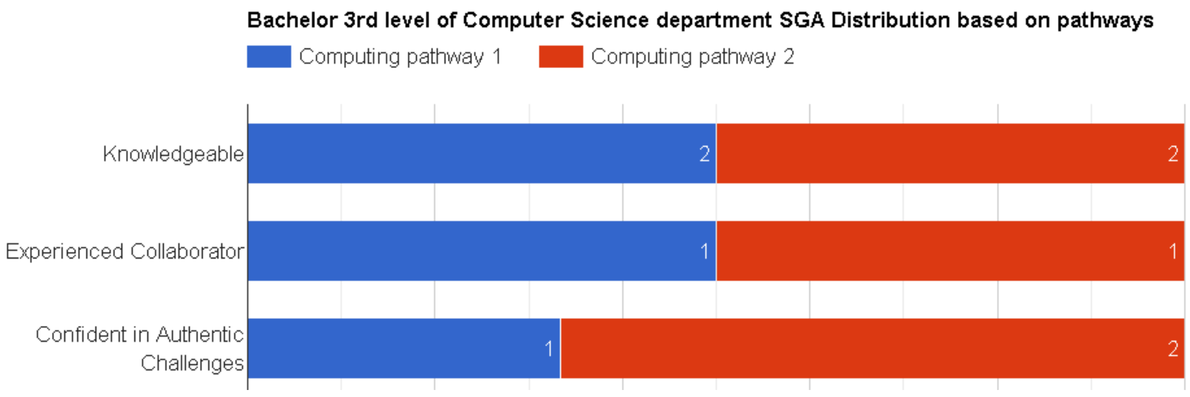


Figure 4.10: A snippet from the badge distribution based on pathways.

This was a challenging graph to implement, because the content changes depending on how many pathways are registered in the system, and this can be infinite, as discussed in 4.3.3.

This statistic overall is a way for the course administration to balance the experience of an undergraduate, by mixing different types of attributes across semester sessions, making each academic year engaging.

SGA Overall

The "student graduate attributes overall" category is where a user can find the distribution of badges for a complete pathway. This means that the distribution of badges across all levels of study of an undergraduate programme of a department are depicted. The user is prompted to choose the department they are interested in and after that a list of all the pathways of that department is depicted. Choosing one pathway, the user is presented with a distribution of the badges in a per-undergraduate-level fashion.

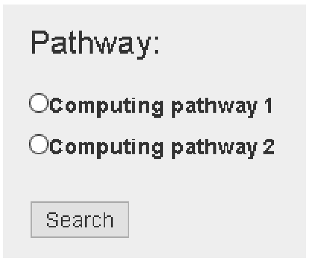


Figure 4.11: Choice of pathway

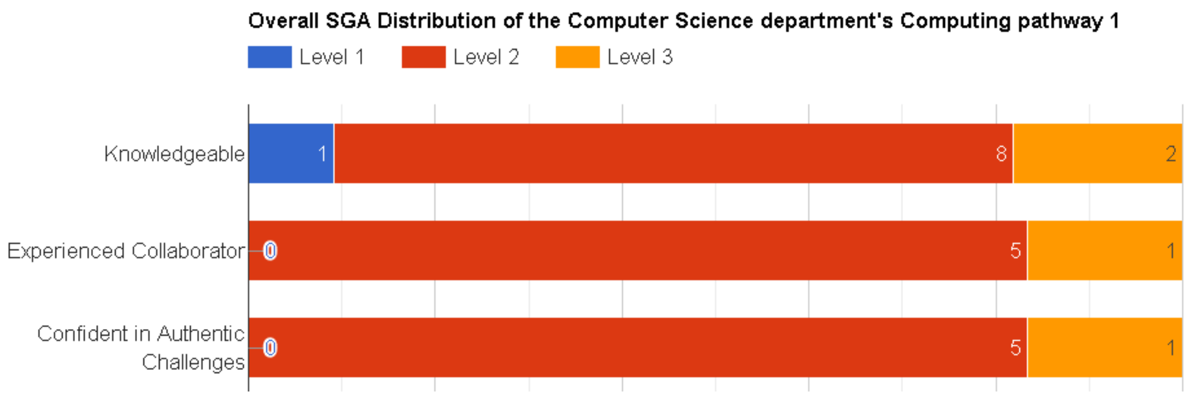


Figure 4.12: A snippet from the badge distribution based on levels of study.

This statistic overall is a way for the course administration to identify the distribution of achievements an undergraduate reaches, during the course of their studies, and when in particular. According to the Programme Level Approach, this aids in the holistic design and delivery of programmes, making each iteration meaningful for the students [1].

Unit Badges

The "unit badges" category of the online interface is a section of the website where the user is able to find a specific unit and view the badges that are associated with it. This section was only created to aid the users of the system. For more information about this the reader is prompted to 4.3.5.

Count of Badges

The "count of badges" category of the online interface is a section that shows the sum of badges per unit, for a level of studies of a department. The user chooses the department and level of studies through the filters and then the graph is populated with the data. The colours of the columns are randomly generated.

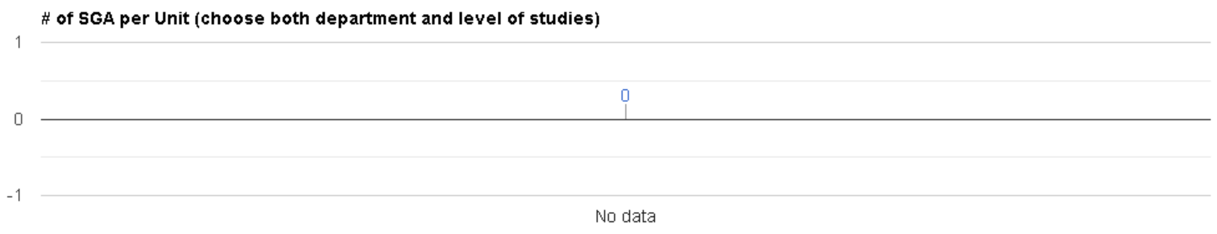


Figure 4.13: An unpopulated badge sum graph.

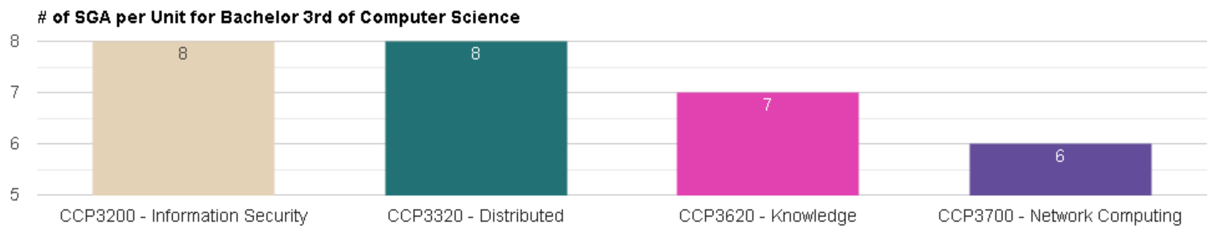
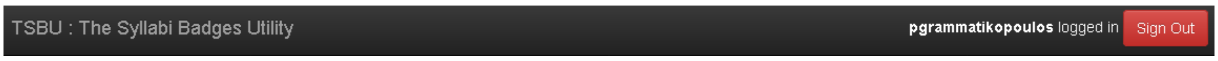


Figure 4.14: A populated badge sum graph.
(Not all units of the level are present in this example)

4.3.3 Elevated permission sections

The online interface of the system includes hidden functionalities, that only users with elevated permissions can see or access, as mentioned in 2.5.2. This section provides a thorough look on their implementation.



TSBU : The Syllabi Badges Utility
pgrammatikopoulos logged in
Sign Out

Figure 4.15: Header of interface without elevated user permissions

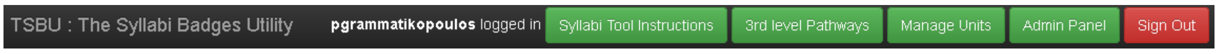
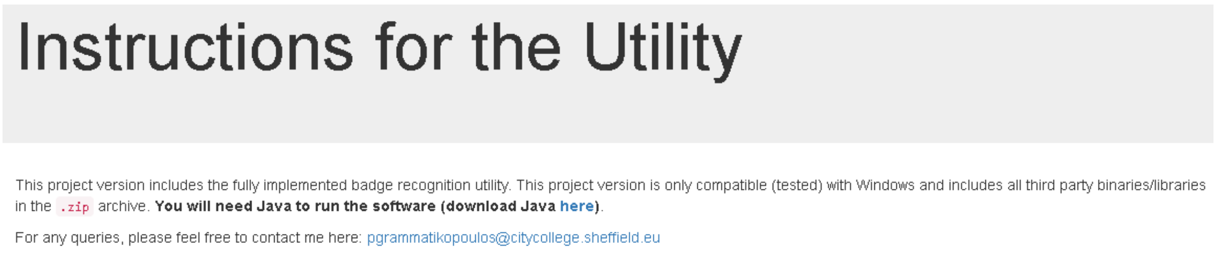

TSBU : The Syllabi Badges Utility
pgrammatikopoulos logged in
Syllabi Tool Instructions
3rd level Pathways
Manage Units
Admin Panel
Sign Out

Figure 4.16: Header of interface with elevated user permissions

Syllabi Tool Instructions

This section of the online interface includes the instructions to download, as well as execute, the utility that updates the online subsystem of the project. The download button on the page does not link directly to the file to be downloaded, but rather checks if the user is a user with elevated permissions. This prevents simple viewers of the website to have any form of access to the tool that updates the unit and badge information, which are stored in the online knowledge repository.

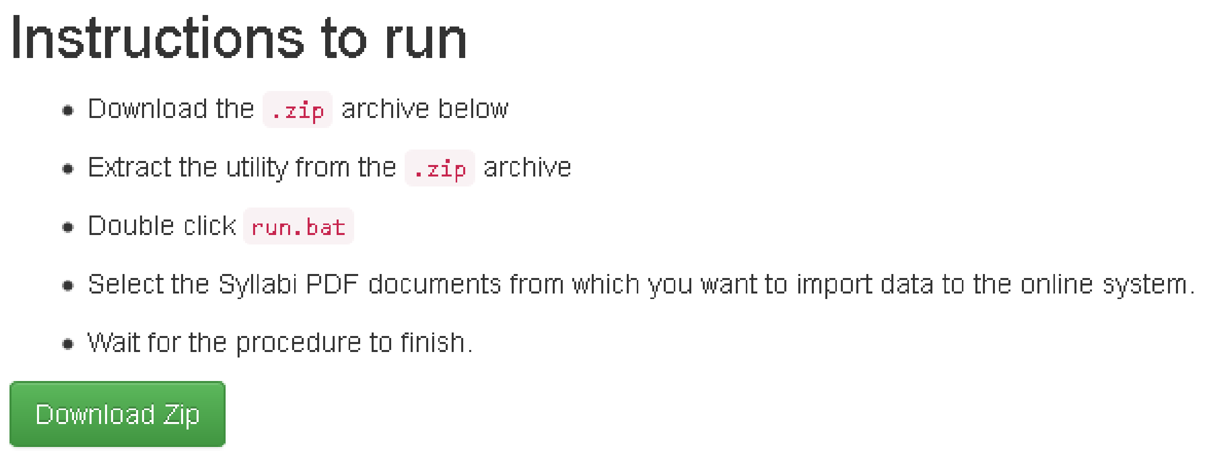


Instructions for the Utility

This project version includes the fully implemented badge recognition utility. This project version is only compatible (tested) with Windows and includes all third party binaries/libraries in the `.zip` archive. **You will need Java to run the software (download Java here).**

For any queries, please feel free to contact me here: pgrammatikopoulos@citycollege.sheffield.eu

Figure 4.17: Instructions on what needs to be downloaded



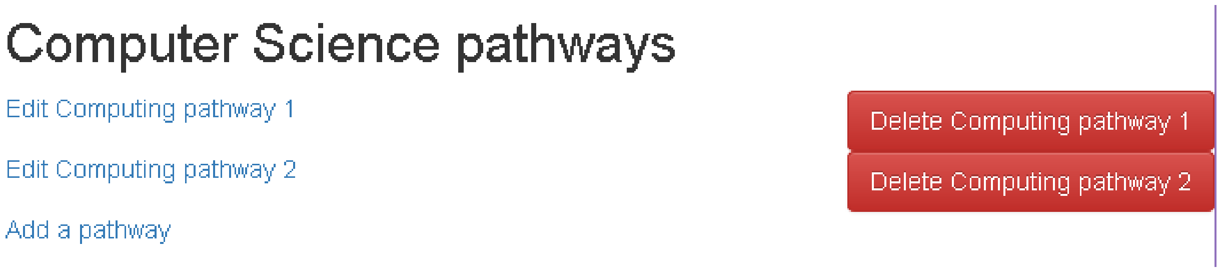
Instructions to run

- Download the `.zip` archive below
- Extract the utility from the `.zip` archive
- Double click `run.bat`
- Select the Syllabi PDF documents from which you want to import data to the online system.
- Wait for the procedure to finish.

Figure 4.18: Instructions on how to execute the utility

Third level Pathways

The pathways of the third undergraduate level propose a system that separates the students into specialisation paths, with their unique choice of units for the final year. Those specialisation paths have a close relation to the student's degree title. While the information about pathways is released some time before each academic year, the pathways are not in any way represented in syllabi and therefore cannot be extracted from them. The author's solution to this was to create a section where course administrators can create and manage an infinite number of unique pathways per department. By choosing a department, the course administrators have the chance to view already existing pathways and edit or remove them, or create new ones. The creation of a new pathway takes them to a new page, where they choose the third level units that will be included in the pathway.

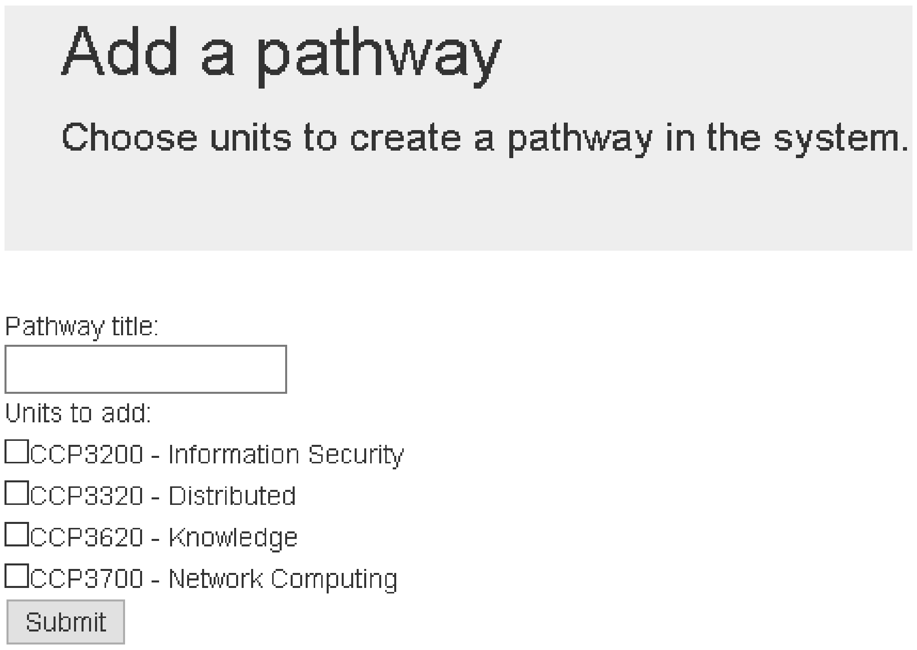

Computer Science pathways

Edit Computing pathway 1

Edit Computing pathway 2

Add a pathway

Figure 4.19: Example of a populated pathways' section


Add a pathway

Choose units to create a pathway in the system.

Pathway title:

Units to add:

☐CCP3200 - Information Security

☐CCP3320 - Distributed

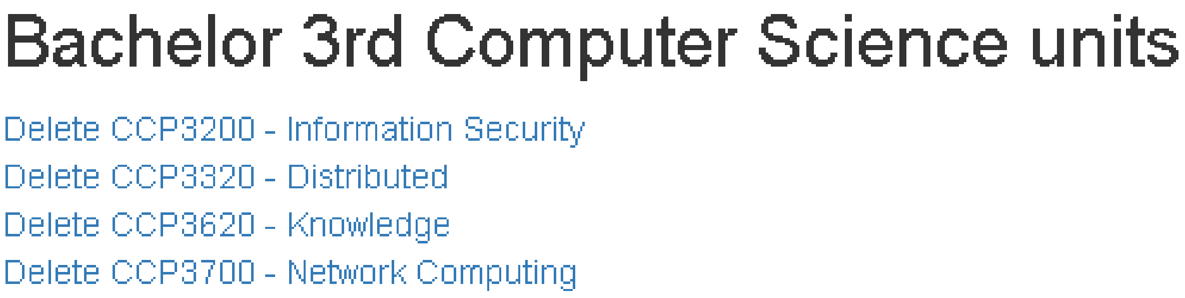
☐CCP3620 - Knowledge

☐CCP3700 - Network Computing

Figure 4.20: Creation or modification of a pathway

Manage Units

This section provides the users with elevated permissions with an interface to remove units from the system, based on the department and level of studies. This would be a valid use case when the information for some units was inaccurately represented in their respective syllabi and therefore there is a need for them no to be depicted and affect the statistics throughout the online interface. After their removal, the units can be inserted back to the system again, in their correct form, using the same automated clientside utility.

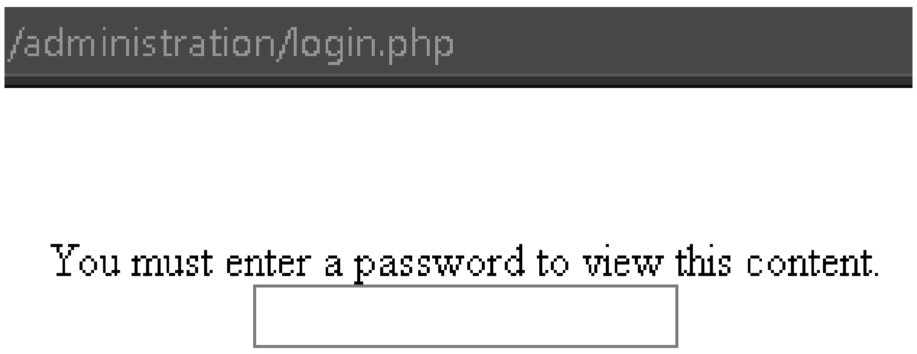

Bachelor 3rd Computer Science units

Delete CCP3200 - Information Security
Delete CCP3320 - Distributed
Delete CCP3620 - Knowledge
Delete CCP3700 - Network Computing

Figure 4.21: Unit deletion selection

Admin Panel

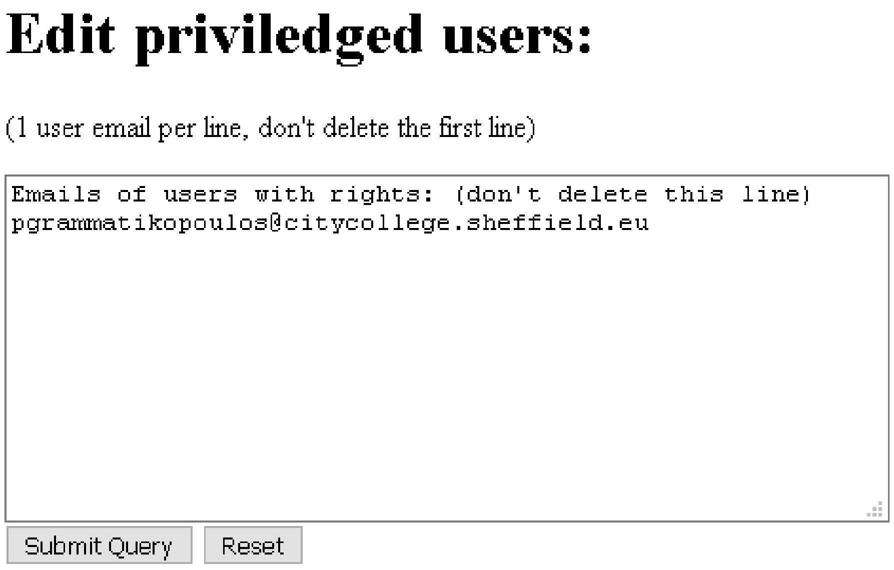
The administration panel is a section accessible only to users with “site Administrator” level permissions, as it requires a password only known by them for authentication. This section provides an easy way to manipulate the list of users with elevated permissions. It also includes a website reset functionality, that can be only ran if the administrator is also a user included in the users with elevated permissions. This action resets the online interface back to its empty deployment state: All the information about units, badges and pathways is flushed from the system.


/administration/login.php

You must enter a password to view this content.

Figure 4.22: Password prompt for administration panel

When the user inputs the correct password, they are transferred to a page that includes the following:

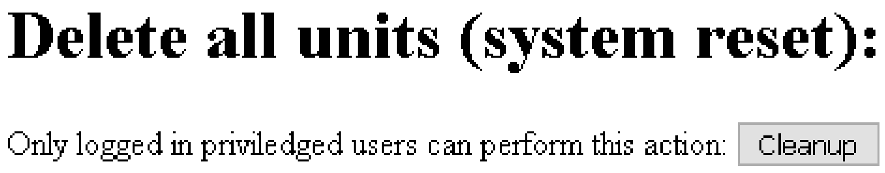


Edit priviledged users:

(1 user email per line, don't delete the first line)

```
Emails of users with rights: (don't delete this line)
pgrammatikopoulos@citycollege.sheffield.eu
```

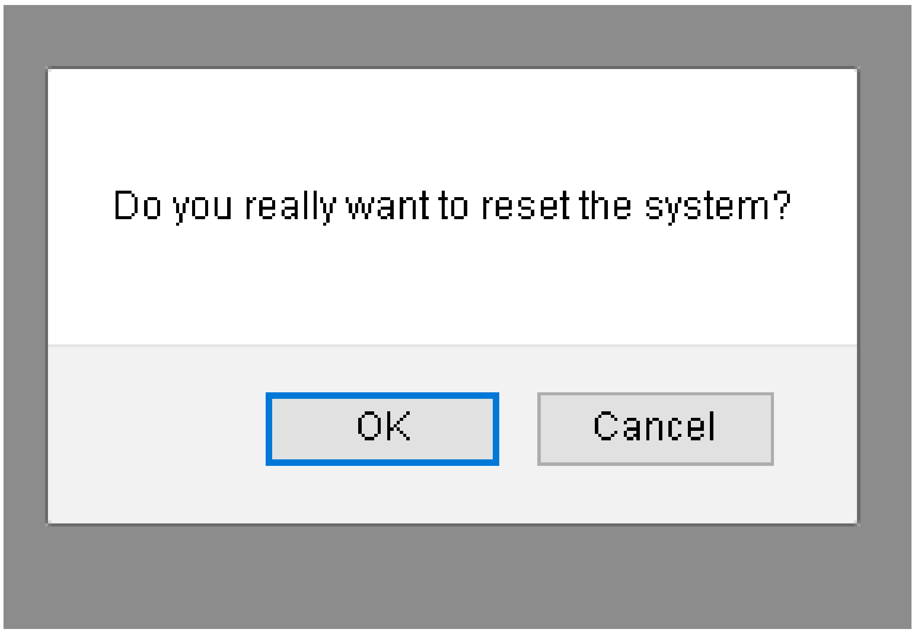
Figure 4.23: Elevated Users' editor



Delete all units (system reset):

Only logged in priviledged users can perform this action:

Figure 4.24: System reset option



Do you really want to reset the system?

Figure 4.25: System reset option confirmation

4.3.4 Personal information & Authentication solution

From the moment the author realised that the system would have to have one of its parts being online, a single goal became a rule around its access methods: The system would not, in any way, store personal user data. The apparent reason for this is the fact that the system is a mere interface to view information about the faculty, like any normal website would be. Making it so the users would have to register themselves, and therefore have personal information about them stored, just to identify themselves and access the system, would be plainly unethical. There is no reason for such a tool to hold the combination of someone's email address and a password they devised, which might be similar to other passwords they use. This only introduces the responsibility that if the system is somehow compromised, there need to be background mechanisms to ensure the security of its user data. In an era where technology giants cannot fight back online crime effectively, even if it comes down to social engineering techniques, it is not the kind of burden any computer scientist wants to put on their work without an explicit fair reason.

With the above being stated, this system would only have to use user information for their authentication, so that external users to the faculty do not have access to the information the system holds. Therefore the author thought that, since the staff and students of the faculty have other portals, where they can use their university-provided credentials to login, the project could just somehow fork their authentication services and get verification of the users' identity via their third-party procedures.

To elaborate, the author first thought of the Mole platform as the key to authentication: The user of the online interface of this dissertation would type in their credentials, those would be safely be transported to Mole and initiate a login there. Then, if the response was a successful one, the system would log the user in on the dissertation system too. This means that the login process would really take place on Mole's platform and the dissertation interface would only be a wrapper of that procedure. While this solution ensured that the author would be in complete control of the development of the module that performs the authentication procedure, and could therefore take full responsibility of its flaws, if any, a meeting with one of their supervisor's colleagues stopped this idea from manifesting any further. What they mentioned was that not all staff members necessarily have a Mole account right when they join the faculty. What they instead insisted on was that every member of the faculty has a Google account.

The author was not entirely in favour with this solution, as it adds a dependency to Google's obfuscated login implementation, that would be hard to diagnose in its entirety. Also the size of this implementation makes the dependency quite big in size, in relation to the rest of the project. The benefits of this solution that made the author consider it and finally implement the authentication mechanisms around it were those of

1. Convenience for the system's users, as they are already accustomed to Google's services.
2. Separation of concerns, as the author would not have to develop the authentication process from scratch.

3. Goal achievement, as the solution still works without storing any of the users' personal data server-side.

4. Burden lifting, as the authentication mechanism's security is a responsibility of Google, when used correctly by developers.

4.3.5 User experience and Design decisions

This section briefly points at all the details that the author believes make the overall system pleasant to use. While the performance of the system was the primary goal from the beginning, the author made sure to compromise whenever possible to achieve a more usable and understandable experience. These cases can be classified as either user experience cases or design decision cases. Design is indeed how a product functions and communicates its functionality to the users, but some details tend to fall into the practicality realm, while others towards aesthetics.

Practical decisions

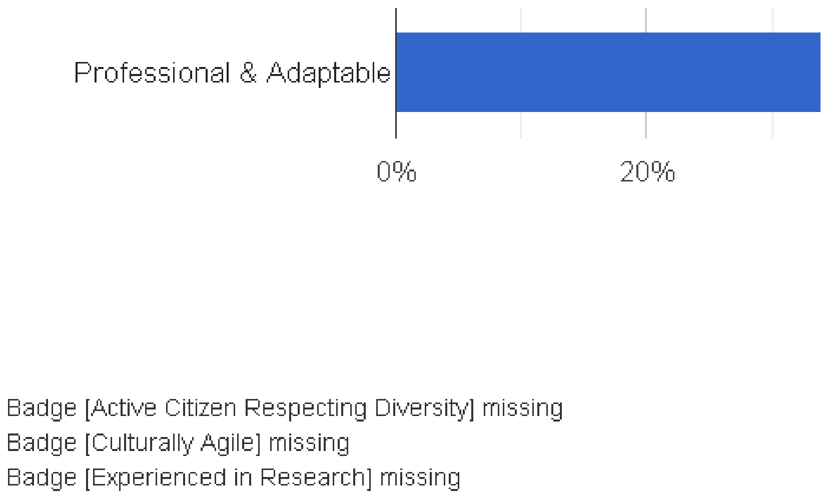
The practical decisions can be seen as the points that make the online interface more understandable. Starting with the way the website is laid out, the first thing a user sees, when accessing the website, is the landing page. This serves two purposes:

1. The blue button links to the introduction of this dissertation. As a color that symbolises trust and loyalty, it is the perfect way for an intrigued user to learn more about the project, if they really need to.

2. The green login button, that includes Google's logo, is a straightforward and familiar way to safely guide the user towards logging in the online part of the system.

Moving into the internal first screen of the website, the user is greeted with the categories of statistics they can view. Those categories are arranged according to the type of graphs depicted: The left-hand side shows categories with distribution graphs while the right-hand side the category of simple bar graphs. In the middle the reference-related category is depicted for the user to quickly discover the various badges of units, without the need to have the syllabi.

All categories include short descriptions both on the initial page as well as inside them, with brief instructions on what the user should do to view the content they need. When there are no user selections on the filters, the pages are either empty or show an empty graph, to signify that an action needs to be taken, as well as to keep the screen real estate free. A very important detail is that of the "missing badge" notifications at the very bottom of the pages. This is a way to ensure that graphs are not cluttered with zero values while also informing the users that information is missing from the graph, for appropriate action to be taken.



Badge [Active Citizen Respecting Diversity] missing
Badge [Culturally Agile] missing
Badge [Experienced in Research] missing

Figure 4.26: Notifications for missing badges

The functionalities for elevated permission users are depicted as green buttons across their website header, giving the feeling of option menus, just like in traditional desktop software. This should intrigue their interest to explore the functionality that lies underneath them. The instructions' section specifically makes sure to hint visually all the important points about the setup and use of the clientside utility, so the users do not skip steps that are important.

As for the clientside utility that updates the online knowledge repository, the goal was to make it as simple as possible for course administrators to find the syllabi on their machines and with a few clicks let the utility itself do the rest of the work automatically. After all the whole project was meant to be created to automate every bit of the manual procedure they had to follow initially.

Aesthetics

The aesthetics of the website can be separated in two approaches: The contrast approach and the mobile friendliness of the interface. The contrast approach can be seen as an example on every page of the website. The top slightly shadowed compartment contains a big title with a short description or instruction, while the rest, lightly coloured area, is the area of results. The graphs themselves are rendered with vivid colours, that make different pieces of information stand out, without tiring the eyes of the users. The easiest-to-develop graph of the website, which only features classic vertical chart bars, also has a feature of random colour generation per graph rendering, giving a taste of playfulness to a project that was overall the fun development experience of a university alumnus.

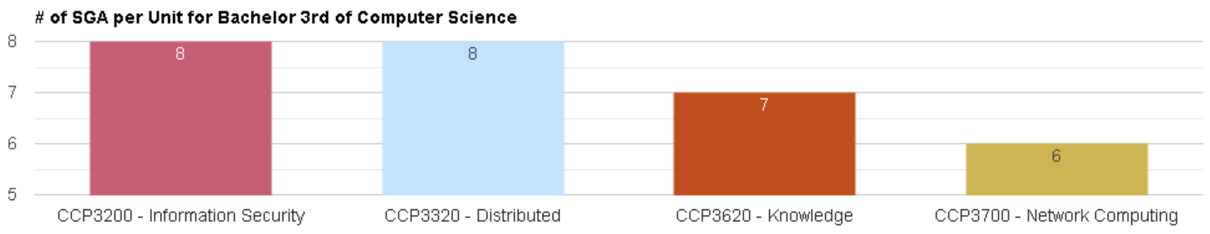
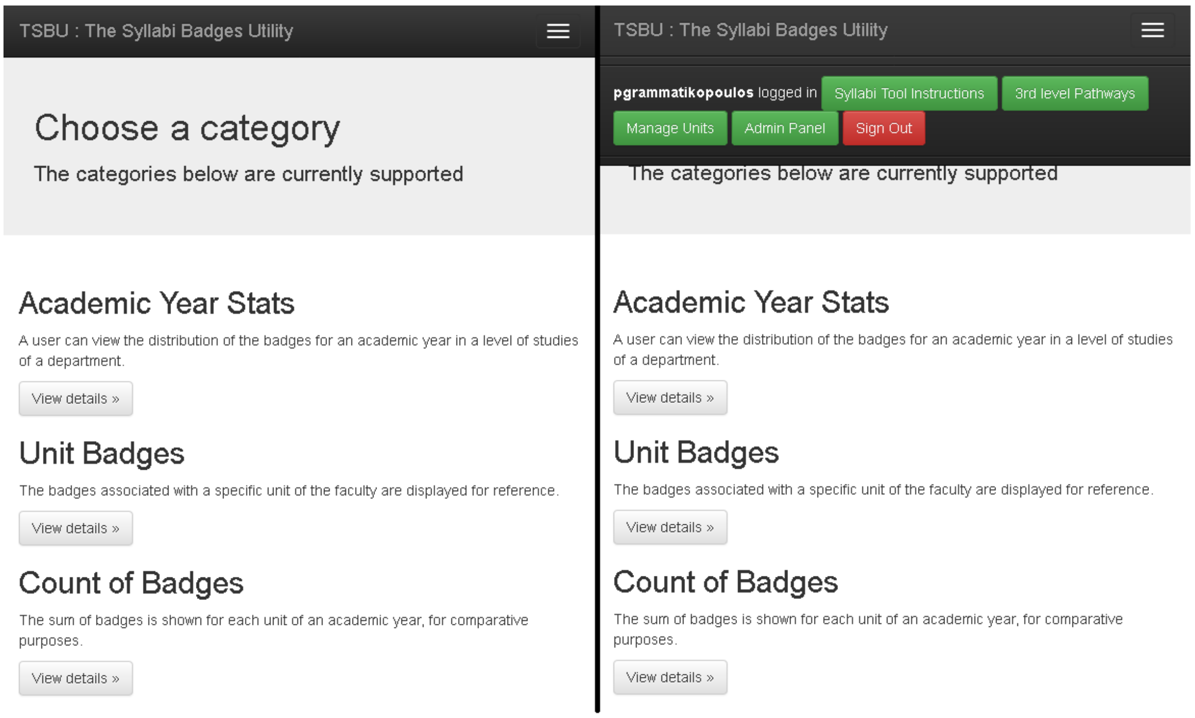


Figure 4.27: A different range of colours to that of figure 4.14

Moving to mobile friendliness, the entirety of the online subsystem was developed utilising Bootstrap's functionality to make it scale well for mobile devices with smaller resolutions and screen dimensions, making it accessible and usable from the users' smartphones [11]. Added to that, Google Charts, the library used for the rendering of graphs, produces graphs that scale to the dimensions of the users' screen sizes and also includes tool-tips that pop out, if the chart bars are hovered over with a mouse or pressed continuously on a touchscreen [12]. This gives accessibility to the whole range of the details that are missing from the abstracted graphs. All these features were tested both on the developer tools web browsers provide as well as on actual mobile devices.


TSBU : The Syllabi Badges Utility

Choose a category

The categories below are currently supported

Academic Year Stats

A user can view the distribution of the badges for an academic year in a level of studies of a department.

View details »

Unit Badges

The badges associated with a specific unit of the faculty are displayed for reference.

View details »

Count of Badges

The sum of badges is shown for each unit of an academic year, for comparative purposes.

View details »

TSBU : The Syllabi Badges Utility

pgrammatikopoulos logged in Syllabi Tool Instructions 3rd level Pathways Manage Units Admin Panel Sign Out

The categories below are currently supported

Academic Year Stats

A user can view the distribution of the badges for an academic year in a level of studies of a department.

View details »

Unit Badges

The badges associated with a specific unit of the faculty are displayed for reference.

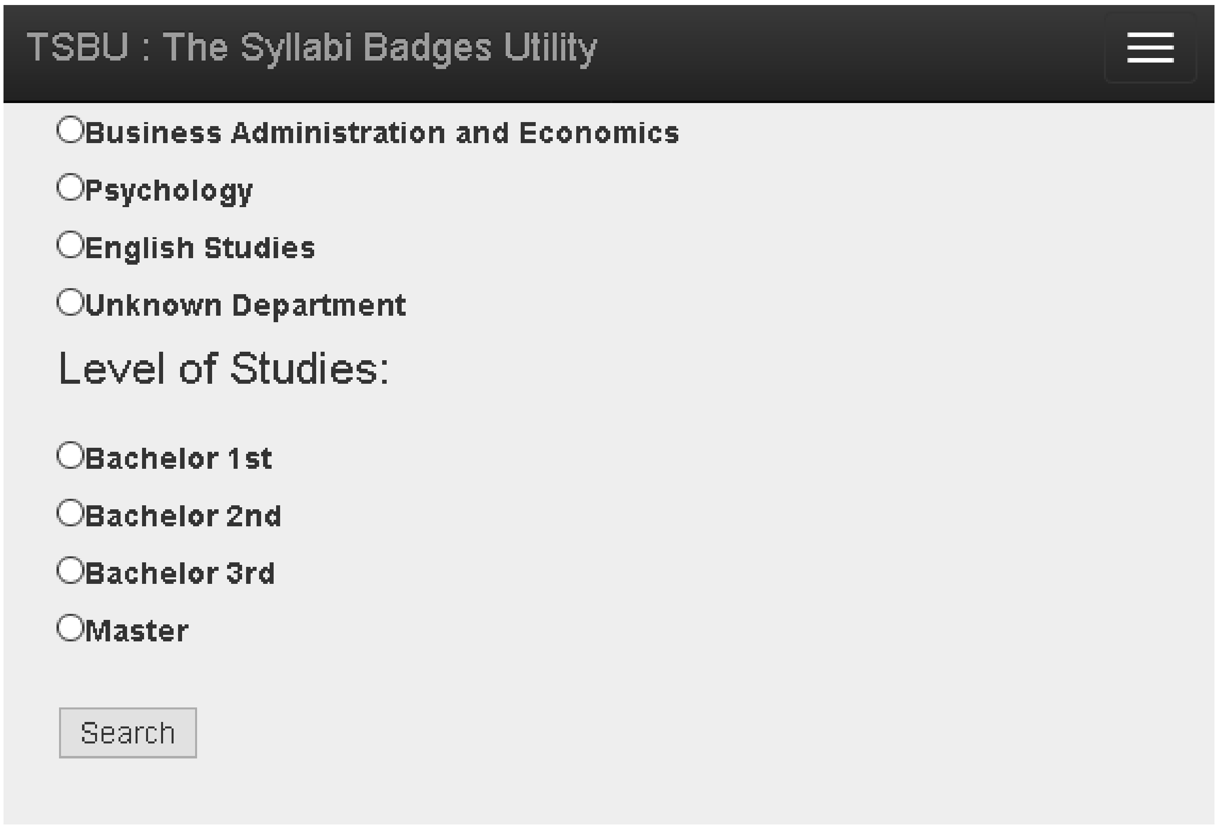
View details »

Count of Badges

The sum of badges is shown for each unit of an academic year, for comparative purposes.

View details »

Figure 4.28: Example of scaling on mobile devices


TSBU : The Syllabi Badges Utility

○Business Administration and Economics

○Psychology

○English Studies

○Unknown Department

Level of Studies:

○Bachelor 1st

○Bachelor 2nd

○Bachelor 3rd

○Master

Search

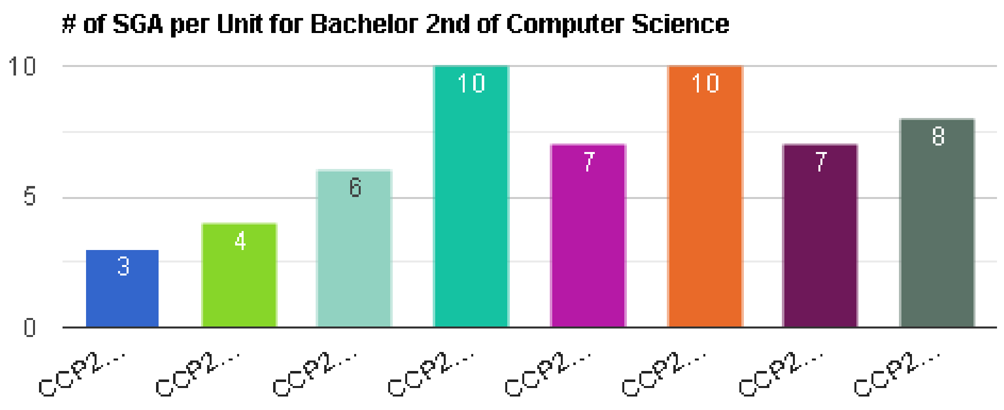


Figure 4.29: A graph with more data scaled for mobile devices
The details for each unit can be viewed when the user holds on a bar to display the tool-tip, like in figure 4.30

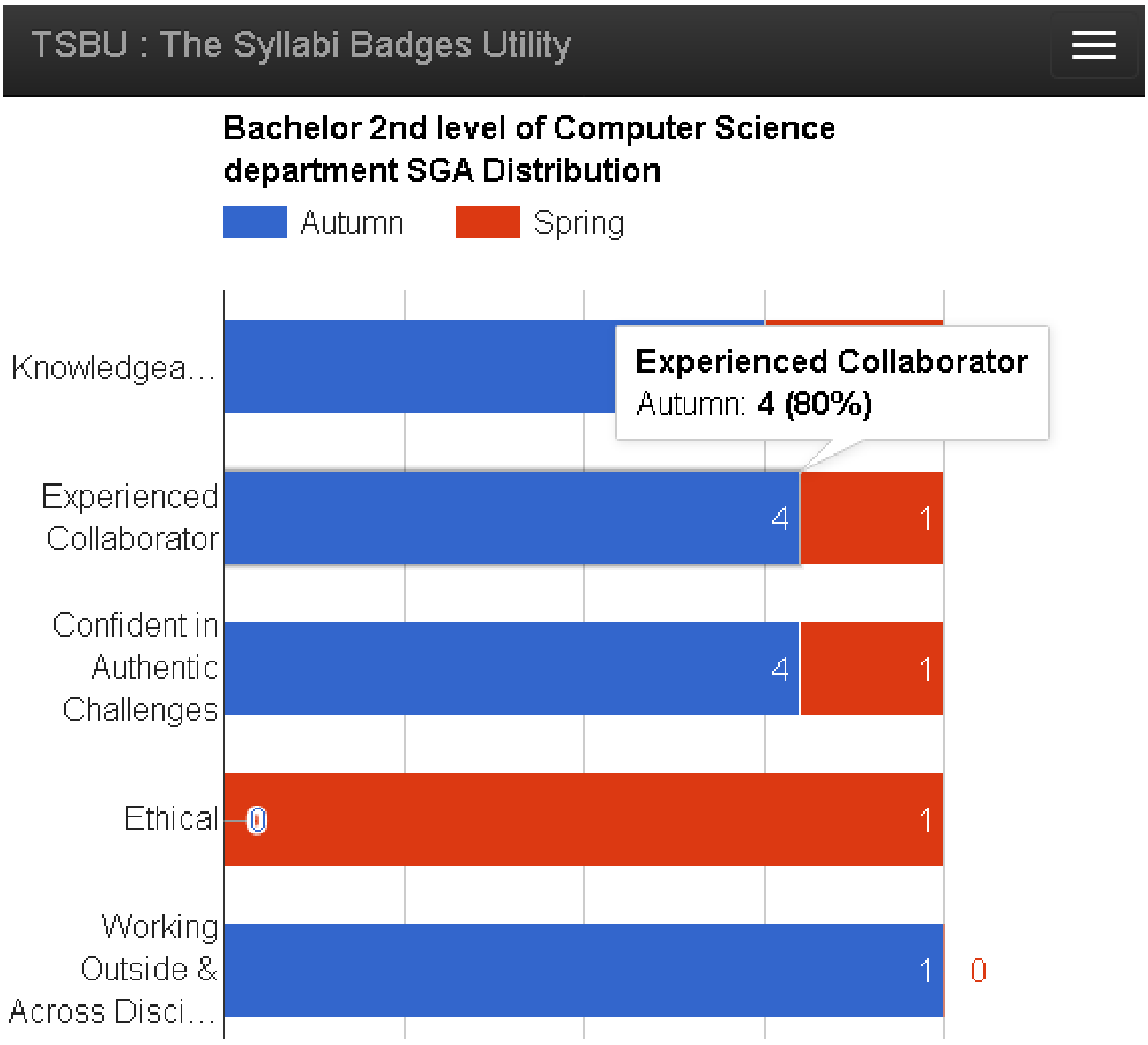

TSBU : The Syllabi Badges Utility
Bachelor 2nd level of Computer Science department SGA Distribution
Autumn
Spring
Knowledgea…
Experienced Collaborator
Confident in Authentic Challenges
Ethical
Working Outside & Across Disci…
Experienced Collaborator
Autumn: 4 (80%)
4
1
4
1
0
1
1
0

Figure 4.30: Information tool-tips for graph details

Chapter 5

Testing and Evaluation

As part of a formal development procedure, the system had to be thoroughly tested and evaluated by the author, to meet the appropriate standards to be ready for use. This chapter is dedicated to the above concepts.

5.1 Testing

Testing is the sum of procedures a developer chooses, to objectively assess the quality of their products. Testing can be an exhaustive procedure as part of debugging, or a mere usability test of the proposed system.

5.1.1 Black-box testing

Black-box testing is the specific type of testing the author used throughout the development of the final product. This type of testing goes through the functionality of the system, to verify and validate it, without actually getting in the deeper parts of the code implementation. This is ideal for systems with linear functionality, like that of this dissertation, as the procedures followed are predictable and without varying outcomes, if, of course, they are well thought-out during the design process.

To be more elaborate, the author put a lot of effort into the algorithmic designing process, to take care of all the edge cases they could think of. After that, they implemented the solution in their programming language of choice, by transpiling their pseudo-code. Finally, black-box testing was conducted on each proposed software module to verify its behaviour and assure its validation in relation to the requirements of the rest of the system.

5.2 Evaluation

The evaluation of a system is the procedure of checking whether the final product accomplishes all the objectives it started with. The author splits this section into two evaluation areas: The project evaluation and the user evaluation.

5.2.1 Project evaluation

The project evaluation of this dissertation is only concerned with whether the final product satisfies the initial objectives, discussed in the introduction. The objectives will be enlisted and discussed:

- It has to be more effective and efficient to use than the manual procedure statistics have been carried out with as of now: This is achieved completely, as the solution is fully automated and of high performance. It also uses system resources effectively.
- It has to make the Programme Level Approach our faculty follows more open to the students, giving them greater insight over their studies: This is fully achieved, as the system is open for access to all students of the faculty.
- The information the system carries has to be easily accessible by all kinds of devices(computers or mobile devices): Fully achieved, as the main interface of the system is always online and scales to different screen resolutions.
- It has to be a solid foundation/basis for more functionality to be implemented in the future: It is, as all the information is stored in a relational database, making different types of queries possible. Also the basic algorithms that the clientside utility is composed of take care of irregular user behaviour, which makes the system versatile.

5.2.2 User evaluation

While no formal user evaluation was conducted by the developer, as it was not their main concern over delivering a complete product, they are confident that they have succeeded into delivering a high quality system for the following reasons:

- Their supervisor gave them great insight in the processes course administrators follow, as well as the semantics of different aspects of the faculty, for example the unit codes.
- Their software demonstrations were received favourably and all questions had a legitimate answer to be given, that was in favour of their solution.
- They conducted informal user testing with their peers throughout their studies, to ensure the user experience of the system is as fluid as possible.

Chapter 6

Future work and how it is envisioned

This chapter covers all the information the author has thought of and gathered for future developers of the system. Specifically it hints missing functionality that could be added to either of the subsystems, under-performing aspects of the overall system, as well as ideas for future dissertation proposals that would be based on this system.

6.1 Bug in text extraction

The only bug in the implementation of the extraction of the unit information, on the clientside utility, is triggered by the possibility of extra spacing between words in the unit title field. In that case, since the text is delimited based on the spacing, that is replaced by a delimiter(seen in 3.2.2), only the first word of the unit title is being kept. This proved to be challenging to resolve as a unit title can have a completely arbitrary amount of words and the next field might not always be the same to use that as a delimiter. Therefore this is definitely a matter of future work that needs to take place in the backend of the client-side utility.

6.2 Expansion of system

While the system developed, as part of this dissertation, covers all the existing needs of the faculty for undergraduate programmes, the author kept in mind the possibility of its expansion by future developers. By “future developers”, what is implied could be the author themselves, students that would be assigned this system as a future dissertation or even professionals hired by the faculty, in case the system got deployed live. There are three key areas that the author recognised that the system would be subject to change.

6.2.1 Master Programmes

The client-side utility of the system, as of authoring the present report, manages to recognise Master level unit syllabi, as well as their respective badges. The online repository, where the information about those units is stored, is also present. The main issue is that the programmes Master level units belong to manifest similarly to the pathways of the third level undergraduate programmes in real-world scenarios: Their content could change or new programmes could be introduced, rather frequently in the macro-scale of the faculty's operating history. While the core functionality of recognising the information about the units and storing it is present, the way to facilitate their depiction in the online system is not finalised. The author, though, can advise future developers that the underlying implementation of the online subsystem for Master levels would be the exact same with that of pathways. The course administrators will have a new area in the website, that will work exactly like the third level undergraduate pathways' management does, but it will instead show Master level units on the selection screen, when creating a new Master programme. The database would have the addition of new tables, but instead of them being geared towards pathways, they would be for Master programmes. The logic and functionality of the rest of the implementation would be kept identical.

6.2.2 Addition of badges

This is a hypothetical scenario, that was posed as a question by the supervisor of this dissertation, that relates to the expansion of the system based on new badges. The system both on the client-side utility, as well as the online repository of information, uses identification numbers for syllabi. These act as reference pointers to derive the rest of the information that is stored in each subsystem. Specifically, on the clientside utility, the pointers reference the two versions of backend asset images the badges are associated with, while on the online subsystem, the identification number is a reference to the title of a badge, as well as where its image is stored online. For the addition of a new badge therefore, three parts of the system would have to change:

1. The online repository to include the new badge, with its image source link
2. the implementation of the clientside system in the `Comparator` class, for the inclusion of the new badge
3. the two versions of the backend asset images that represent the badge in the image recognition procedure.

It needs to be noted that, for the best recognition results, as well as to keep the overall clientside subsystem small in size, the images of the third step would have to be in GIF format, 200x200 dimensions and 256 colors. Also the two versions of the image are basically one with a white background and one with a black background (to emulate the alpha layers of an exported PNG image with transparency layers). After all the above modifications, the system would include the new badges.

6.2.3 Addition of statistics

The online subsystem includes all the types of statistics that the staff ran with their spreadsheet methodology. In the case that more statistics need to be added to the system, it is ultimately a matter of creating new sections on the online subsystem and performing the right operations with the knowledge that exists in the information repository(database). This whole scenario would definitely be more suited to become a future dissertation topic, both for the improvement of the already existing graphs, if needed, as well as the introduction of a whole new range of statistics with a solid knowledge foundation already present and automatically updated. What the author envisions about such a dissertation would be a student going through the procedures the course administration staff does, define their motives and goals and derive new meaningful statistics for them by the end of the year, as well as implement them in the existing system. This would be a research-based endeavour that would complement the existing system or even improve it.

6.2.4 Operating system portability

The current clientside utility is compatible with the Windows operating system, as the third-party binaries provided are for Windows. There are two solutions to execute it on non-Windows hosts:

1. The compatibility layer of Wine can be installed on a POSIX-compliant operating system, such as those of Linux, macOS and BSD. The utility has been successfully tested on all three major unix and unix-like distributions this way [13].
2. A group of separate versions of the utility could be introduced that would basically include the correct binaries for each respective target operating system.

Since the majority of the user base that will use this utility are Windows users, after consulting with the author's supervisor, only the Windows based version was implemented and the portability and repackaging of the utility could be part of future work on the project. The author though made sure that their dependencies could be easily found as pre-compiled compatible binaries on most modern operating systems, so packaging a new version of the utility for a different operating system should not be a big concern for a future developer.

6.2.5 Export functionality for graphs

This system did not have as a goal the exporting procedure of the generated graphs, as it would be implemented with performance in mind, and therefore the graphs could be generated upon request by the users on the online interface, and always be accessible when needed. The author though noticed that in the work of an alumnus, that had as a topic only the textual parsing of the PDF syllabi documents, one of the main features was the exporting of the unit data [14]. While their solution could be algorithmically improved with the author's work in this paper in real-world scenarios, with a broader set of "faulty" syllabi, as they used a variation of the word-to-word algorithm discussed in 3.2.2 on their paper, the feature of data extraction would be a nice addition as a sub-feature, in a future dissertation based on this dissertation's product. As discussed throughout the report, the knowledge base is there to be queried and its structure allows for a versatile set of possible outcomes.

Chapter 7

Conclusions

With this final chapter of the dissertation, the author would like to go through what their work has achieved. The software that got developed, as part of this dissertation, is a highly technical proof of concept of how automation of procedures could have a positive effect towards the faculty's functioning. Since the project had as its main goal a product that serves a specific purpose within the faculty's needs, the options for research were limited, but the motivation for systematic and effective work had to be kept at high levels. By breaking down the problem of how course administrators orchestrate statistics about the units of their respective departments, a modular system design got devised and implemented. This system manages to recognise the badges present in the syllabi of the units, associate them with the units and then provide a clear interface, that facilitates all the needs of the faculty members, regarding programme planning.

As an aim of this project was to make the system as easy to use as possible, so that it would be incorporated seamlessly in the workflow of the faculty's staff, there was a clear intention to design the system around its users and their needs. To make this possible, the author had to figure the balance between the performance and intuitiveness associated with the final product, as well as a fair way to spread the use of resources. This paved the path for a system that strives to provide simplicity and abstracts the details of its implementation well enough, so that they seem hidden. The only thing the faculty staff should ever do, for this system to reach its full potential, will be to provide it with the PDF documents of all units. The rest of the processes need not be attended or handled by the users and the time frame in which they conclude is far shorter than that of the initial problem. Therefore, with a lot more time in their hands, the course administrators can focus on more important aspects relating to course planning and delivery, while exploring numerous new possibilities.

Another goal the system manages to accomplish is that of opening up the Programme Level Approach of our faculty to a wider audience, and specifically that of the students themselves. Students should be able to have a greater insight than previously into the structure of their studies, as well as their personal development throughout them. They could also help in the planning process, with more constructive feedback towards the staff of the faculty, creating an opportunity for their meaningful engagement in the shaping of their academic careers. This is a valuable option the author believes any student should have the power to acquire, during what is their stepping stone towards a successful professional future.

Finally, the author would need to point out that this tool is only the basis for something more holistic in the years to come. New types of statistics, as well as new information about the learning approach the University of Sheffield achieves its high quality standards with are hopefully part of the future of the online interface that got developed. There is room for improvement in the current implementation of the system, but overall the key goals, as of now, are being met, and it would be a merry outcome to see its expansion taking place in the future, while looking back at what it started as.

Bibliography

[1] University of Sheffield, "Learning and teaching." [Online]. Available: https://www.sheffield.ac.uk/staff/learning-teaching/our-approach/programme-level

[2] "Percentage difference between images." [Online]. Available: https://rosettacode.org/wiki/Percentage_difference_between_images#Java

[3] "pdfimages." [Online]. Available: https://www.xpdfreader.com/pdfimages-man.html

[4] "FFmpeg." [Online]. Available: https://ffmpeg.org/

[5] Welch, "A technique for high-performance data compression," *Computer*, vol. 17, no. 6, pp. 8–19, Jun. 1984. [Online]. Available: https://doi.org/10.1109/mc.1984.1659158

[6] "pdftotext." [Online]. Available: https://www.xpdfreader.com/pdftotext-man.html

[7] "Unicode Character 'FULL BLOCK' (U+2588)." [Online]. Available: https://www.fileformat.info/info/unicode/char/2588/index.htm

[8] "Unicode® 1.1." [Online]. Available: https://www.unicode.org/versions/Unicode1.1.0/

[9] "GNU Bash." [Online]. Available: https://www.gnu.org/software/bash/

[10] "MariaDB." [Online]. Available: https://mariadb.com/

[11] Otto, Mark and Thornton, Jacob, "Bootstrap." [Online]. Available: https://getbootstrap.com/

[12] "Charts — Google Developers." [Online]. Available: https://developers.google.com/chart/

[13] "What is Wine?" [Online]. Available: https://www.winehq.org/

[14] Ramadani, Elijon, *FINAL YEAR PROJECT: Creating a database of module syllabi from documents.* The University of Sheffield, 2019.