The International Journal of Computational Science, Information Technology and Control Engineering (IJCSITCE) Vol.6, No.1, January 2019

# Library Information Retrieval (IR) System of University of Cyprus (UCY)

Antreas Dionysiou[1], Michalis Papaevripides[1], and Andreas Charalampous[1]

[1]Department of Computer Science, University of Cyprus, P.O. Box 20537, 1678 Nicosia, Cyprus

adiony01@cs.ucy.ac.cy,mpapae04@cs.ucy.ac.cy,achara28@cs.ucy.ac.cy

**Abstract.** Building an effective Information Retieval (IR) System for such a complex sector like a library, is indeed a challenging task. Creating this kind of applications, one must be aware of basic IR methodologies and structures used, as well as the User's Experience requirements. Combining different technologies for creating different kinds of applications has been one of the major problems in software reuse. However, in recent years, many frameworks that offer a complete suite for developing cross-platform applications have been developed. In this paper, we present a combination of breakthrough technologies and frameworks for developing a Python based RestAPI, an Administrator's side desktop application and a cross-platform application powered by Ionic, Angular, HTML and CSS, as the main IR tool of UCY's library.

**Keywords:** Information Retrieval, Web Technologies, Elasticsearch, User Experience, Search Engine

## 1 Introduction

Displaying the most relevant results according to a user's query is still an open problem. However, many software companies (ex. Google, Yahoo etc.) have been working on the improvement of their IR algorithms as well as the ranking of the retrieved results. There are some well known big issues in IR like (a) relevance of each document retrieved (i.e. relevance mesaurement), (b) evaluation methodologies used (i.e. precision, recall, datasets used), and (c) users and information needs (search evaluation must be user-centered). On the other hand, some search engine issues that need to be considered by all field experts are: (a) performance (efficiency in any aspects), (b) dynamic data (updates, additions and deletions), (c) scalability (make it work with millions of users) and (d) adaptability (tuning search engine components) [1]. The "de facto" data structure used for storing a document's contents is positional inverted index [2]. By using this type of data structure, one can perform various types of queries like wildcard queries, phrase queries and proximity queries, using different techniques like biwords, permuterm or k-gram indexes. Furthermore, the tokenization, lemmatization and stemming techniques have to be language specific.

DOI: 10.5121/ijcsitce.2019.6102

The Big Data that exist nowadays, make the process of analysing documents and constructing inverted indexes much more difficult [3]. The scientific community has been straggling with parallel programming and the fact that a single machine has limited hardware capabilities. The main challenges that need to be tackled are: (a) distribute computation, (b) the hardness of Distributed/parallel programming and (c) the fact that machines fail. Due exactly to these reasons Dean J. and Ghemawat S. have proposed the groundbreaking Map-Reduce framework [4] that addresses all of the above. This framework enables programmers to work with Big Data in an elegant and efficient way, while it also limits the excessive need of parallel programming experts. Smartphones and mobile devices in general, have been the top selling electronic devices for the last twenty years. According to www.wearesocial.com the total world's population is 7.6 billion whereas the number of unique mobile users is approximately 5.2 billion [6]. This means that the number of unique mobile users is about the 70% of world's population. According to www.statista.com, applications in Google's Play Store and Apple's App Store are approximately four million [5]. Today, almost any task can be controlled through a smart device. This, is an extremely important fact, that broadens the mobile application market. A RESTful API is an application program interface (API) that uses HTTP requests to GET, PUT, POST and DELETE data. A RESTful API (also referred to as a RESTful web service) is based on REpresentational State Transfer (REST) technology, an architectural style and approach to communications often used in web services development. The REST used by browsers can be thought of as the language of communication in the internet. With cloud use on the rise, APIs are emerging to expose web services. REST is a logical choice for building APIs that allow users to connect and interact with cloud services. RESTful APIs are used by sites such as Amazon, Google, LinkedIn and Twitter [7].

In this paper we present a combination of state-of-the-art technologies, interacting with each other, to create the IR system of UCY's library. The architecture as well as the implementation will be examined in later sections.

## 2 Methodology and IR System's Architecture

In order to determine the exact system's architecture and requirements needed for such a problem, a team composed by three members performed an excessive number of face to face meetings with the library staff (almost all kind of staff) as well as on site inspections of the building architecture and book shelves in order to gather a general in the beginning and a more specific idea in a later stage regarding the mapping of books' categories with the shelves. An enormous number of interviews (especially with the IT staff) has been also accomplished, for gathering information regarding the already working website and Application Programming Interface (API) used, as the main library website for UCY, as well as different updates that the new IR system should have.

After gathering the exact user's requirements, the team moved forward by determining the technologies that will be used in order to develop the library's

IR system backbone architecture. After an excessive research, the team has came up with a final architecture backbone solution consisting of the following six (6) parts: (a) a central RESTful API [7], (b) the UCY's library API named Sierra, (c) the Google's Books API, (d) an administration site application for configuring books' categories - shelves locations mapping, (d) an open source search engine and (e) a mobile application. An illustration of the library's IR system backbone architecture is shown in Figure 1.

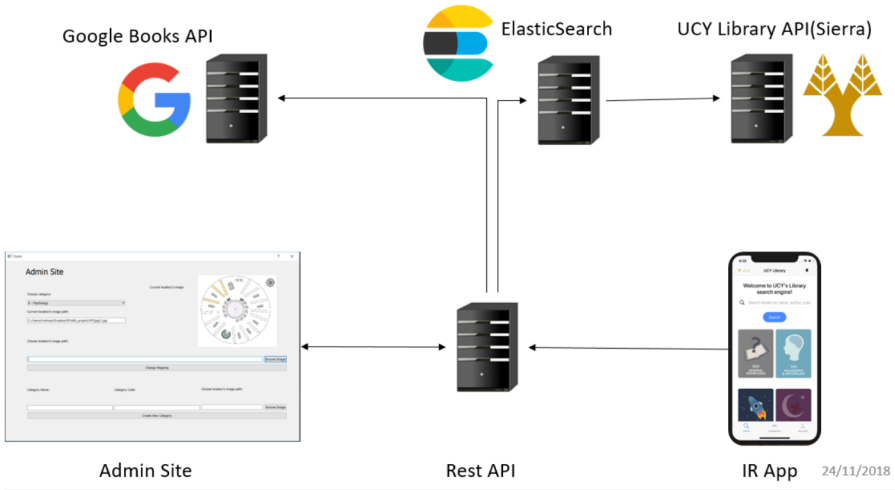


**Fig. 1. UCY's Library IR System Backbone Architecture:** In this figure, the combination of technologies used to create the library's IR system can be seen.

## 3 Architecture Components

### 3.1 Central RESTful API

The Python based restful API was built using the flask framework. Flask made the communication between mobile application with elasticsearch and Google's Books API easy enough using the HTTP protocol. The application is performing HTTP requests via HTTP methods (GET, POST, DELETE, PUT) in order to communicate and retrieve data from the specified resources. The client submits an http request to the server and then using Elasticsearch API and Google's Books API the data are collected and analysed in the central API component.Then the server (Flask based) returns a response to the client which contains the status information about the specific request and the requested content. The client has the ability to search by string or by requesting books related to specific categories. The appropriate endpoints are called for the proper use of elasticsearch and Google's Books API.

### 3.2 UCY's library API (Sierra) and Google's Books API

The current library's IR system is basically a simple webpage that retrieves data from a main API called Sierra. The Sierra API is used to retrieve information regarding the library's books. Nevertheless, it has to be noted that the book's information accessible through Sierra API was limited. The lack of information about library's books, like the publisher, description or the book's cover image, was an extremely important problem for the IR system development team as the information regarding any book is considered to be a critical aspect related to IR system's performance. Many fundamental operations like searching for a specific book require a full information record for each specific book of the library. This is the main reason that the team has decided to integrate the Google's Books API [8]. Integrating Sierra API with Google's Books API through a central python based RESTful API lead to a complete, in terms of available books' information, API.

### 3.3 Administrator Site Application

A main request from the UCY's library staff, was to be able to somehow guide people to books' shelves locations. More specifically, the main requirement of library's staff was to create a visual mapping of books categories and on site locations, so for library's users to be able to track a book's location as fast as possible. This requirement was stated as extremely important by the development team, due to the enormous size and same architectural design library's floors. In other words, not having a guidance to find a specific book's location, one can easily get lost in the enormous number of books in UCY's library. The development team has immediately requested the architectural plans to design a mockup images regarding books' locations, in order to determine the exact requirements regarding books' shelves location guidance. An example of a mockup image is shown in Figure 2.

Each book is located to a specific range of shelves in the building based on it's category and each category is represented by the first letter or letters of the call-number code. As the UCY's library staff requested an efficient way to modify and determine each category's mapping, the development team developed a user-friendly graphical interface, that allows administrator users to: (a) create new categories, (b) delete old ones, as well as (c) change the mapping of each category. When adding a new category, the administrator's site application (based on Python and more specifically PyQt4) requests as input the name and call-number of the new category and an image file path which represents its physical shelve location. It has to be noted that the physical book shelve - category mapping can later be modified due to obvious reasons. Finally, each time a change is made regarding categories (adding, removing or editing), a dynamic JSON file containing the latest book shelve - category mapping is constructed and passed to RESTful central API.

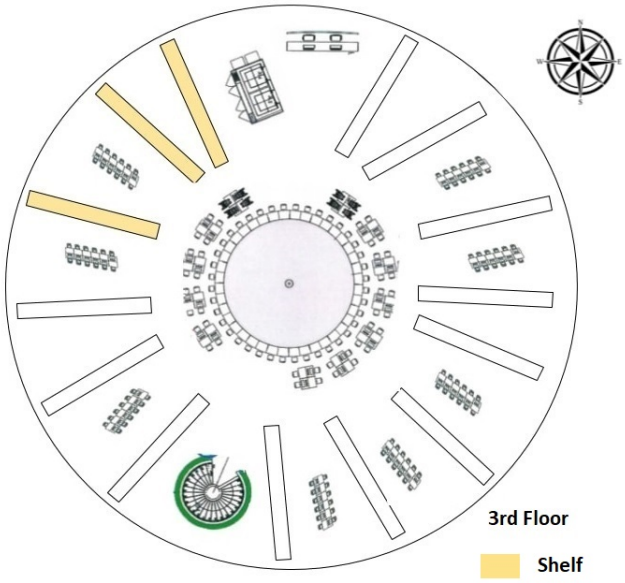


**Fig. 2. UCY's Library Mockup Guidance Image:** In this figure, the 3rd floor of UCY's library is shown. The yellowish lines represent the shelves that a specific selected book might be located.

### 3.4 Elasticsearch

As stated in introduction, the main challenges in the new Big Data age are: (a) distribute computation, (b) the hardness of distributed/parallel programming and (c) the fact that machines fail [3]. Elasticsearch [13] is a distributed [12], RESTful search and analytics engine capable of solving a growing number of use cases. Elasticsearch is free, open-source, document (json) oriented search and analytics engine built on top of Apache Lucene [14]. This extremely breakthrough tool is used for full-text search, structured search, analytics, or all three in combination. It has the ability of near real-time searching meaning that there is a slight latency ($\tilde{1}$ sec) from the time someone indexes a document until the time it becomes searchable. The development team has decided to use Elasticsearch tool as the responder to mobile application HTTP requests. In particular, the development team has retrieved all the data from UCY's Sierra API, integrated them with Google's Books API data, and finally indexed the aggregated results in an Elasticsearch server. In this way, the development team has managed to create a distributed search engine containing all UCY's library books information available and searchable in the class of milliseconds. The central RESTful API communicates directly with Elasticsearch server for every mobile application request. The UCY's library mobile application created is thus independent of Google Books API and Sierra API.

### 3.5 Mobile Application

As stated in the beginning of this paper, the number of unique mobile users is about the 70% of world's population. Every person nowadays, owns at least one mobile device. This is an important fact that lead the development team in deciding that the main contact point of library's users and library's books information will be a cross-platform application. Cross-platform mobile development is the creation of software applications that are compatible with multiple mobile operating systems. Originally, the complexity of developing mobile apps was compounded by the difficulty of building out a backend that worked across multiple platforms. Although it was time-consuming and expensive, it was often easier to build native applications for each mobile Operating System (OS). The problem was that the code built for one operating system could not be repurposed for another OS [9]. Developing a cross-platform mobile application as the main contact point between library's users and library's books information was one way road. The development team has concluded on using the state-of-the-art technologies and frameworks for performing such a task. The Ionic framework for developing the mobile application has been chosen due to the fact that is well maintained, free (paid pro features exist though) and open source [10]. The Ionic framework is powered by Apache Cordova that lets programmers target multiple platform with one codebase. Apache Cordova produces native code for each platform that programmer wants to deploy his mobile application on [11]. The programming languages that Ionic framework supports are the state-of-the-art programming languages regarding web technologies like TypeScript, Angular, HTML and CSS. The mobile application will communicate with Google's Books API [8] and UCY's Sierra API for retrieving books' information, through the central python based RESTful API [7]. It has to be noted that the mobile's application usability has been tested according to the state-of-the-art industry standards.

## 4 Evaluation and Discussion

### 4.1 Requirements Satisfaction

The first step for performing a basic evaluation is checking that all the users' requirements have been satisfied. This is a critical task as the initial purpose of developing any kind of application is to serve specific users' requirements. In requirement analysis phase, a list containing all users' requirements has been constructed in order for the evaluation to be performed in a later stage. The development team has been assured that all requirements specified in that list have been satisfied. Furthermore, a face to face meeting with the library's staff has been scheduled, in order to present the final solution and gather any comments regarding requirements satisfaction or future updates and corrections.

### 4.2 Usability Testing

The usability of the library's mobile application has been one of the main concerns since the beginning of the project. The development team had to build a mobile application that could be used with ease from all UCY's members. The graphical representation of the mobile application as well as the actions that a user can perform have been the top two requirements in the users' requirements list. After making an initial mockup design of the mobile application, the development team has been recursively collecting feedback from random users regarding their user experience using the application. Finally, the development team has created a questionnaire based on the well known System Usability Scale (SUS). The SUS provides a reliable tool for measuring the usability. It consists of a 10 item questionnaire with five response options for respondents; from Strongly agree to Strongly disagree. Originally created by John Brooke in 1986 [15], it allows you to evaluate a wide variety of products and services, including hardware, software, mobile devices, websites and applications. The SUS has become an industry standard, with references in over 1300 articles and publications. The noted benefits of using SUS include that it: (a) is a very easy scale to administer to participants, (b) can be used on small sample sizes with reliable results and (c) is valid – it can effectively differentiate between usable and unusable systems. The results of SUS are as follows: (a) the 87.50% of users think that they would like to use the mobile application frequently (Figure 3), (b) the 82.50% of users disagree with the fact that the system is unnecessarily complex (Figure 4), (c) the 90% users found that the system was easy to use (Figure 5), (d) the 87.50% of users strongly disagree with the fact that they will need the support of a technical person to be able to use the system (Figure 6), (e) the 87% of users found that the various functions in the system were well integrated (Figure 7), (f) the 87.5% of users strongly disagree with the fact that there was too much inconsistency in the system (Figure 8), (g) the 92.50% of users think that most people would learn to use this system very quickly (Figure 9), (h) the 95% of users disagree with the fact that the system was very cumbersome to use (Figure 10), (i) the 90% of users felt very confident using the system (Figure 11) and (j) the 90% of users disagree with the fact that they needed to learn a lot of things before they could get going with this system (Figure 12). The graphs from the SUS study are shown at the end of this paper (section 4.5).

### 4.3 Discussion

As we have seen from the feedback and SUS results collected, the system seems to be in a great shape for the time being in order to be used as the main IR and search engine tool from UCY's library users. This is extremely important as one of the main concerns in software development and engineering is the user experience. The applications are built by programmers for casual users and this is one of the main facts that every development team has to keep in mind. Furthermore, the administrator site application for dynamically configuring the books' shelves mapping has been presented to the IT staff of UCY's library and they are very

happy with it. Of course performing the SUS testing for the administrator site application would be an unnecessary and redundant action as this tool would be used by experts of the field such as the specialized UCY's IT staff, so the development team just skipped this step. As stated in the introduction of this paper, the development of such a complex interconnected IR systems that will be used from people with some or none technical skills is indeed a great challenge. Due exactly to this fact, all the aspects, including the design of the mobile application and user experience, have to be considered. In addition, the main big issues in IR and search engines development (also stated in introduction), have to be tackled in an efficient way in order for the overall IR system to be usable. If any of these issues could not be resolved properly the final IR system may be thrown in the trash. In other words, one expert of the field should deal with all these aspects or the final IR system may not be usable at all. In conclusion, the presented IR system for UCY's library will be deployed in a short period of time letting in this way the development team, to gather more statistics about the usability of the system and any updates/upgrades that should be made. The usability testing of all information systems (including IR systems) is directly related with their performance and efficiency of use. The next section (section 4.5) represents the results taken from the SUS testing over a sample of forty (40) random UCY's members.

### 4.4 Future Work

In this paper, a state-of-the-art library IR system was explained in detail. Furthermore, the detailed architecture as well as the major components that an IR system must have, were given. Finally, the proper methodology for examining the performance and usability of such systems was explained and issued. The results of this evaluation can be seen in section 4.5. Nevertheless, many other tasks could be performed in the future in order to provide a complete methodology and architecture for IR systems. Nowadays, many library IR systems are based on very different architectures and they need to be evaluated in order to have a comparison between different implementations. Moreover, the IR system performance should be measured in detail in order to have a full view about the implementation methodology. Also, the same usability test (SUS) could be performed on a much larger sample in order to obtain more accurate measurements regarding the usability of the whole IR system.

### 4.5 Graphs

As stated above, SUS testing has become the main industry standard for evaluating the usability of an information system. This section shows the graphs for the ten (10) questions of SUS testing over a random sample of forty (40) UCY's members, where for each graph 1 denotes "Strongly Disagree" and 5 denotes "Strongly Agree".

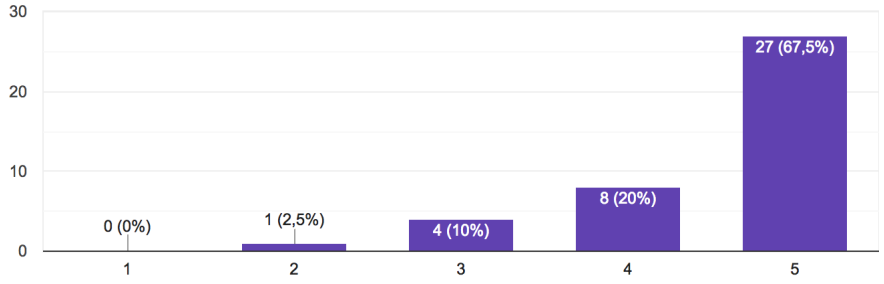


**Fig. 3. SUS Question:** I think that I would like to use this system frequently.

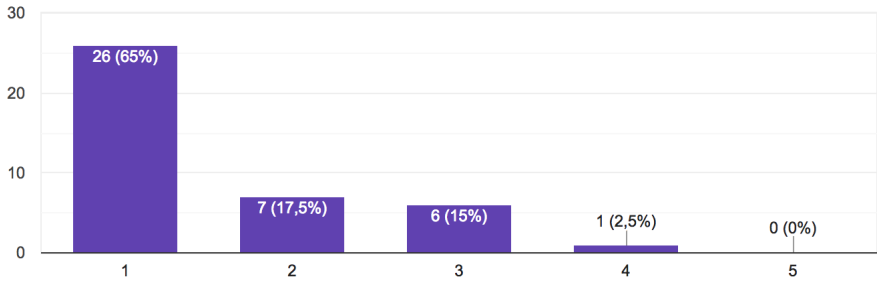


**Fig. 4. SUS Question:** I found the system unnecessarily complex.

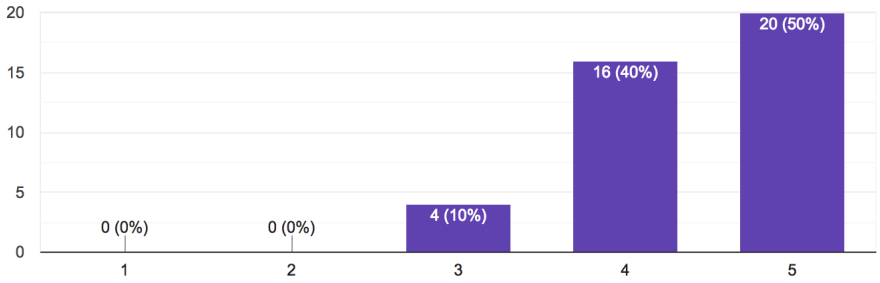


**Fig. 5. SUS Question:** I thought the system was easy to use.

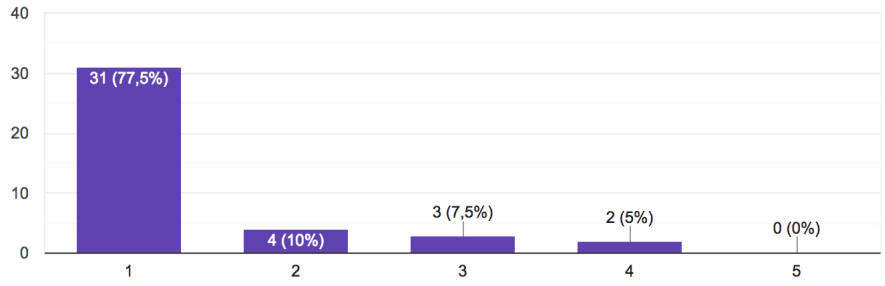


**Fig. 6. SUS Question:** I think that I would need the support of a technical person to be able to use this system.

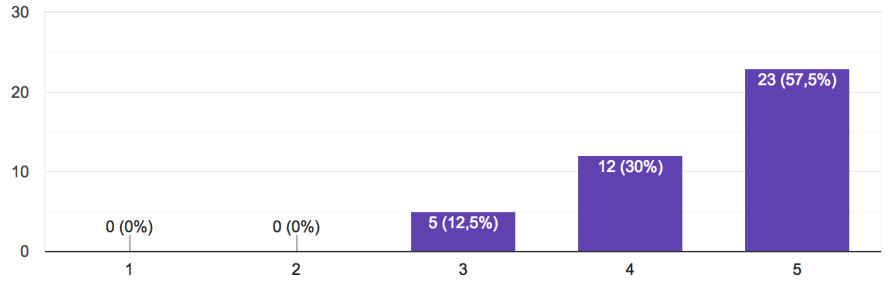


**Fig. 7. SUS Question:** I found the various functions in this system were well integrated.

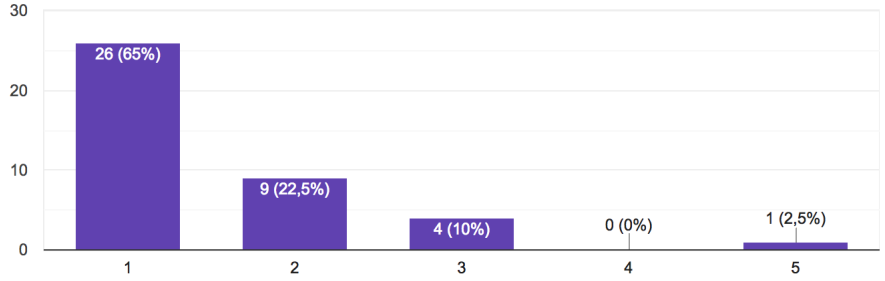


**Fig. 8. SUS Question:** I thought there was too much inconsistency in this system.

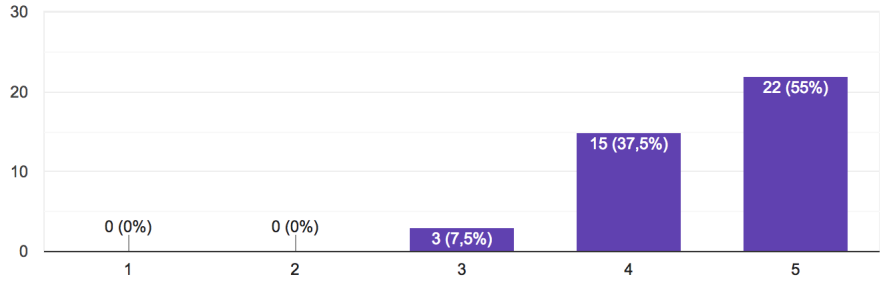


**Fig. 9. SUS Question:** I would imagine that most people would learn to use this system very quickly.

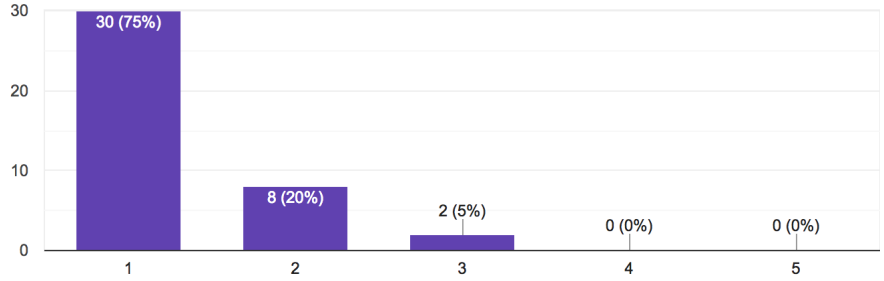


**Fig. 10. SUS Question:** I found the system very cumbersome to use.

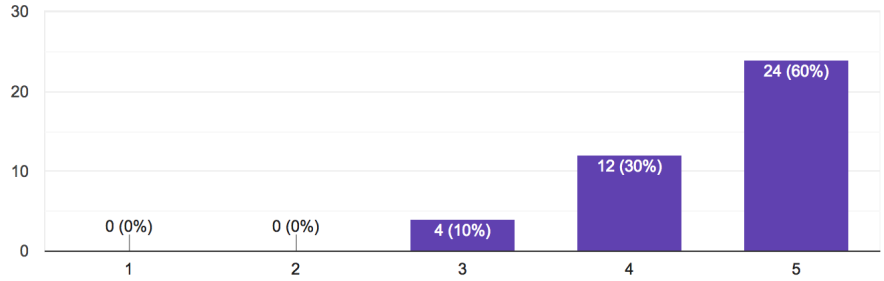


**Fig. 11. SUS Question:** I felt very confident using the system.

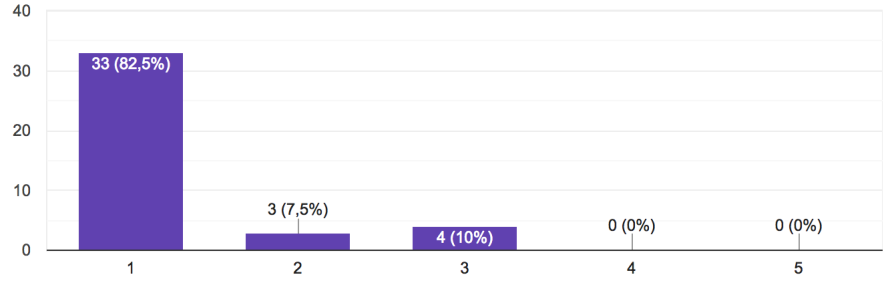


**Fig. 12. SUS Question:** I needed to learn a lot of things before I could get going with this system.

## References


1. Manning, C. D., Raghavan, P. & Schütze, H. (2008). Introduction to information retrieval (Vol. 39). Cambridge University Press.
2. Zobel, J., Moffat, A., & Ramamohanarao, K. (1998). Inverted files versus signature files for text indexing. ACM Transactions on Database Systems (TODS), 23(4), 453-490.
3. John Walker, S. (2014). Big data: A revolution that will transform how we live, work, and think.
4. Dean, J. & Ghemawat, S. (2008). MapReduce: simplified data processing on large clusters. Communications of the ACM, 51(1), 107-113.
5. Statista. (2018). App stores: number of apps in leading app stores 2018 — Statista. [online] Available at: https://www.statista.com/statistics/276623/number-of-apps-available-in-leading-app-stores/ [Accessed 21 Nov. 2018].
6. We Are Social USA. (2018). Digital in 2018: World's internet users pass the 4 billion mark - We Are Social USA. [online] Available at: https://wearesocial.com/us/blog/2018/01/global-digital-report-2018 [Accessed 21 Nov. 2018].
7. Richardson, L. & Ruby, S. (2008). RESTful web services. " O'Reilly Media, Inc.".
8. Google Developers. (2018). Google Books APIs — Google Developers. [online] Available at: https://developers.google.com/books/ [Accessed 22 Nov. 2018].
9. Heitkötter, H., Hanschke, S. & Majchrzak, T. A. (2012, April). Evaluating cross-platform development approaches for mobile applications. In International Conference on Web Information Systems and Technologies (pp. 120-138). Springer, Berlin, Heidelberg.
10. Ionic Framework. (2018). Build Amazing Native Apps and Progressive Web Apps with Ionic Framework and Angular. [online] Available at: https://ionicframework.com/ [Accessed 22 Nov. 2018].
11. Cordova.apache.org. (2018). Apache Cordova. [online] Available at: https://cordova.apache.org/#getstarted [Accessed 22 Nov. 2018].
12. Croft, W. B., Metzler, D. & Strohman, T. (2010). Search engines: Information retrieval in practice (Vol. 283). Reading: Addison-Wesley.
13. Elastic.co. (2018). Elasticsearch: RESTful, Distributed Search & Analytics — Elastic. [online] Available at: https://www.elastic.co/products/elasticsearch [Accessed 23 Nov. 2018].
14. McCandless, M., Hatcher, E. & Gospodnetic, O. (2010). Lucene in action: covers Apache Lucene 3.0. Manning Publications Co..
15. Brooke, J. (1996). SUS-A quick and dirty usability scale. Usability evaluation in industry, 189(194), 4-7.