Electronics and Computer Science

Faculty of Physical Sciences and Engineering

University of Southampton

Kwong Chi Tam

April 26, 2016

Novel Visualization Models for Exploring Academic Research Papers

Project supervisor: Dr. Long Tran-Thanh

Second examiner: Dr. John N Carter

A project report submitted for the award of

MEng in Computer Science
with Image and Multimedia Systems

UNIVERSITY OF SOUTHAMPTON

Abstract

FACULTY OF ENGINEERING, SCIENCE AND MATHEMATICS
SCHOOL OF ELECTRONICS AND COMPUTER SCIENCE

A project report submitted for the award of MEng in Computer Science

Novel Visualization Models for Exploring Academic Research Papers

by Kwong Chi Tam

This project was inspired by the daily challenges that we are confronted with as undergraduate students when searching and filtering relevant scholarly literature. The objective was to build an innovative user interface that clearly displays core metadata from academic papers and features user interaction techniques as well as useful visualisations which enable pleasant search and discovery experiences. Current scholarly search engines and relevant work were analysed as an attempt to understand what users are looking for. Once the proposed system, ScholarWiz, had been implemented, a user experience evaluation was carried out in order to assess and validate the success of the web application.

Contents

List of Figures

List of Tables

Acknowledgements

Firstly, I would like to express my sincere gratitude to my supervisor, Dr Long Tran-Thanh, for his constant support and encouragement to work on a project that truly appealed to me.

I would also like to thank Elsevier and their Integration Support Coordinator, who kindly allowed me to use the Scopus APIs for the purposes of this project, and for quickly responding to all of my queries.

Finally, I wanted to thank all of the anonymous participants who took part in my user study and provided invaluable feedback on the system that I developed.

Chapter 1

Introduction

1.1 Overview and Problem

During the course of a university degree, a student is expected to write a large number of reports or essays regardless of the field that their course belongs to. Lecturers are expecting high quality pieces of writing that are backed up with suitable and relevant academic references. However, finding useful and reliable sources of information that can be included in one's work is nowhere near an effortless task.

Scholarly search engines such as Google Scholar, Elsevier's Scopus or Thomson Reuters' Web of Science have large databases of academic literature that facilitate the exploration and discovery of interesting publications. Yet, these tools may not be intuitive enough for users with limited experience. Students are overwhelmed by the immense volume of search results retrieved by these services, making it particularly difficult to confirm a paper's importance or relevance [10]. Even if the user has previous experience using such technologies, it is inevitable to end up with an uncontrollable number of tabs and windows during research.

Essentially, the problem can be defined as being presented with a great quantity of results, but interfaces that leave much to be desired and little way of determining which of these results are actually relevant to us.

1.2 Project Goals

The objective is to construct an innovative user interface that clearly displays relevant information including publication title, authors, abstract and references amongst others. Such design will be successful if it allows users to seamlessly navigate through search results and evaluate the relevance of each publication without having to read the whole paper or having to access external websites where they are hosted.

I wish to tackle this challenge by building a web application in which the user has a greater degree of interaction with the data available from bibliographic libraries. Information visualisation should be a suitable instrument for reducing the complexity of data and presenting it in a way that maximises usability and ease of understanding. Illustrating citation networks should allow users to view multiple papers at once and their relationships within a single-page application. This should provide a trivial way of discovering subject-related papers based on their references and citations. Finally, I will carry out a user experience evaluation in order to conclude whether students find the proposed system to be convenient for their research purposes.

Chapter 2

Background Research

Before design and implementation, background knowledge regarding current systems is necessary to understand what is being done right, what areas comprise weaknesses and how can they be improved.

2.1 Scholarly Search Engines

This section presents two of the most popular, subscription-based (Web of Science) and free (Google Scholar), academic literature collections that are currently active.

2.1.1 Google Scholar

According to the study About the size of Google Scholar [1], the database is estimated to host roughly 160 million documents as of May 2014, making it the largest digital library by far [1]. Nonetheless, this figure is not considered to be of significant importance within the academic community, since Google Scholar is highly criticized by library professionals for its coverage [2] and is widely known to lack the reliability and accuracy of other digital libraries [3].

Despite its controversial reputation, it still remains as the most popular tool amongst students and faculty staff [2, 4], largely due to the Google brand, familiar interface and matchless speed [3, 5]. This is no surprise, since Google Scholar is also access free, meaning that it is open to a larger audience and word of mouth has a high repercussion. Since its launch in 2004, it has emerged as the

clear frontrunner amongst freely accessible libraries such as Microsoft Academic Search, CiteSeerX or Scirus, some of which are defunct now.

2.1.2 Web of Science

Web of Science, previously known as Web of Knowledge, is a subscription based scientific citation-indexing service, which can be accessed by anyone within a subscribing institution. It is a popular choice amongst academics, featuring a database containing over an estimated 20,000 journals and 90 million records [6]. One of its main advantages over Google Scholar is the quality control and normalization of its catalogue [1]. When science and engineering researchers at the University of California Santa Cruz were surveyed in 2010 about their scholarly database usage, Web of Science was the most popular choice followed by Google Scholar and PubMed, which shared the second place [7].

The user interface and search speeds were some of the weaker aspects of Web of Science, positioning Google Scholar as the preferred alternative due to its performance and ease-of-use [7]. However, in recent years, Thomson Reuters has focused on redesigning the user interface in order to provide an "improved layout and clear directions to enhance the search experience" [8]. The result of this is a modern-looking website, which provides convenient features such as list of references and citations for each paper, allowing intuitive navigation between linked articles.

2.2 User Experience Design

User experience takes into consideration a user's internal state, the features of the proposed system and the environment within which interaction takes place [4]. This means that my proposal must firstly address the user's main purpose, which is finding relevant literature. Then, by designing an interface that reduces the complexity of data and maximises usability, I can offer a pleasant research experience to the user.

It has already been concluded that Google Scholar's dominant popularity is highly dependant on its minimalist and familiar user interface [3, 7, 4]. The average user simply wants to perform a plain search query. Google addresses this well by hiding any advanced features and offering a clean, uncluttered homepage consisting of a simple search box.

Other digital libraries provide a greater variety of functionality. But despite these features being potentially useful, they are complex to use and often require time to master [5]. According to Shultz, Google Scholar is successful because it allows users to "find some resources they can use rather than be frustrated by a database's search screen" [5].

After reviewing other academic search engines, one realises that flaws can also be identified across Google Scholar's interface. For example, it does not allow users to view the whole abstract of a paper without accessing the website where it is hosted. This makes it hard for one to determine the relevance of a paper exclusively from the results page.

2.3 Data Visualisation and User Interaction

Information visualization attempts to visually represent abstract data in order to enhance human cognition. It has been proven to successfully increase the efficiency and effectiveness of decision makers, allowing them to separate relevant information from the redundant clutter [9]. Visualisations and user interaction that take advantage of the human eyes allow users to understand large amounts of information at once [10].

Interactive interfaces provide a more active learning experience by enabling a higher level of engagement with the user [11]. Nevertheless, it is also worth pointing out that interfaces with excessive interactivity implicate distractions that can cause the user to lose attention from the actual information and focus on interacting with the visualisation models instead [11]. This should be taken into consideration at the time of designing the proposed system, ensuring that users become engaged with the application as long as this interaction provides significant value to the research experience.

2.4 Available Data

It is important to be aware of what data is available and free to use for the purpose of this project. During my research, I have gathered several accessible bibliographic databases and APIs, but they all have unique benefits and drawbacks.

The initial plan for this project was to use the metadata provided by Google Scholar, however, the lack of an official API meant that the only way of extracting the data would be web scraping. This being far from ideal, I found out about other alternatives. For example, CiteSeerX offer their data freely for download and use under a license of creative commons. The issue with this source was the colossal size of its database, which required downloading over an exaggeratedly long period of time. A good alternative was the the small dataset released by the InfoVis 2004 Contest, which is an event that promotes the development of benchmarks for information visualisation [12].

Nevertheless, for the scope of this project I concluded that the APIs offered by Elsevier's Scopus were the most suitable source of data, despite some limitations and restrictions. The Scopus database contains over 21,000 journals [13] and benefits from daily updates, extremely reliable content and accurate citation indexing [14]. It offers the functionality and most of the metadata that I wish to extract from academic publications in JSON format (example in appendix B). It is worth noting that the purpose of this project is not to create a new set of academic metadata or develop an alternative ranking algorithm. Hence, the most reasonable approach to gathering data is to use an existing database. This should save time as well as providing the application with genuine scholarly content.

2.5 Relevant Projects

Projects involving the visualisation of academic data have been attempted before. However, most of them either have inelegant and complicated user interfaces, or limited functionality due to the lack of official library APIs. The following tools and applications will serve as an inspiration for the design and functionality of the proposed system.

Google Scholar Citation Visualisation Tool by Kalki Rose (Defunct)

Browser bookmark which executes JavaScript that bootstraps a web application inside the current page [15]. It allows users to quickly navigate through citation information by offering interactive citation mining, static citation graphs and citation chronology graphs for each journal.

PaperCube by Peter Bergström (Prototype)

Experimental tool for exploring the visual navigation of academic citation networks that relies on the CiteSeerX database [16]. It offers several visualization modes including paper details, circle view (citation mapping), tree map, publications per year and paper graph. It allows the user to easily navigate through the metadata of a paper without having to access external websites or open new tabs.

The proposed design benefits from an easy learning curve, enabling first-time users to easily assess the relevance of a paper and providing more advanced users with a wide variety of visualisations that aid the discovery of other related publications.

CiteWiz by Niklas Elmqvist (Prototype)

CiteWiz is a desktop application for bibliographic visualization of scientific networks, including three different types of visualisation that involve time, influence and keywords. The author claims that it illustrates the hierarchies of articles with potentially very long citation chains [17]. Although Citewiz's visualisations contain plenty of information, the interface was found to be difficult to use by the surveyed users [17]. This shows the importance of finding the ideal balance between simplicity and information quantity.

Allen Institute's Semantic Scholar (Beta)

Semantic Scholar is a free service for searching scientific literature that was only launched recently in November 2015 [18]. Emerging as a potential Google Scholar competitor, Semantic Scholar has already shown promising features that could change the tides in the field of academic search engines. Allen Institute states that their product applies methods such as data mining, natural-language processing and computer vision [18] in order to retrieve the best quality of results.

Semantic Scholar has been selected as a relevant project because of its state-of-the-art user interface, which is particularly clean and trivial in comparison to any other search engines. Search results provide an ideal balance of core metadata and functionality, meaning that users will not be overwhelmed by the amount of information and tools. Additionally, the interface includes innovative design aspects that can be of valuable use during one's research, including live filtering, automatic extraction of figures and tables, and citation graphs on hover.

Chapter 3

Design

In this segment, an analysis and specification of the solution to the problem will be presented. The detailed design will also include a series of wireframes, illustrating the different features that are planned to be included in the system.

3.1 Requirements Gathering

In order to achieve the proposed goals, the system needs to satisfy a set of functional and non-functional requirements. Once I have completed building the application, I can then use this list of requirements to evaluate its functionality. The system shall:

3.1.1 Functional Requirements

- Provide search functionality so the user can search for academic publications and authors.
- Allow the user to select a paper from the search results and display its metadata such as title, authors, references, direct link, etc.
- Provide saving functionality so the user can quickly access certain papers that he/she found interesting.
- Allow the user to click on a referenced or citing paper to directly display the metadata from that paper.

- Provide a graph of the citations of the selected paper per year.
- Provide a visualisation of the citation network of the selected paper.
- Allow the user to select a node from the visualisation in order to view the metadata from that paper.
- Provide metrics and data visualisations involving the search results.
- Display a list of search results when a user clicks on authors or keywords.

3.1.2 Non-Functional Requirements

- Be contained within a single page application
- Offer a clean, intuitive and minimalist user interface
- Provide smooth navigation between papers via references or citations
- Provide easy scalability in case the system is extended in the future

3.2 Identifying Useful Technologies

After deciding that the proposed system was going to be implemented as a web application, the following technologies have been identified to be useful.

Java (Spark Framework)

Java was selected for developing the back-end architecture due to high level of familiarity as well as being an industry standard with plenty of resources on the web. JetBrain's IntelliJ IDEA is the IDE of choice, as it provides full Maven integration as well as excellent version control support. Maven is a software tool that facilitates dependency management.

After experimenting with web frameworks such as Spring MVC and Play 2, I opted for Spark due to its simplicity and straight-forwardness, making it fitting for a small-scale application.

JavaScript, HTML and CSS (AngularJS and Material Design Lite)

A web application is the suitable approach for the implementation of the proposed system, as it provides easier and wider access for users. HTML5, CSS3 and JavaScript will be used in order to build a state-of-the-art and aesthetically pleasing user interface. JavaScript should enhance the user experience by enabling powerful visualisations and interactivity within the system [19].

AngularJS is the most popular JavaScript web application framework [20] and a common choice for the front-end of complex single page applications. It also offers a wide range of useful libraries, including D3, which is used for the implementation of powerful data visualisations. Using the Material Design Lite (MDL) framework will increase the speed of development by saving time from writing large amounts of boilerplate code as well as providing consistency to the design.

AJAX, REST and JSON

REST (Representational State Transfer) is an architecture style for designing networked applications, using HTTP to make calls between client-side and server-side. HTTP calls will be used to transfer JSON (JavaScript Object Notation) data efficiently from the back-end to the front-end and vice versa. JSON is one of the most common formats for storing and exchanging data, and benefits from a human-readable syntax as well as being extremely light-weight.

The use of AJAX (Asynchronous JavaScript and XML) techniques (AngularJS $http) will enhance the interactivity of the system by dynamically updating page content. Also, as I have mentioned before, the proposed system will be obtaining data from the Scopus database through their RESTful APIs.

3.3 Web Application Architecture and Flowchart

At the most basic level there will be a simple client-server split, with a light-weight server and a thick client Single Page Application (SPA) which communicate through a simple RESTful API. Moreover, the system will be encapsulated within a global level Model-View-Controller (MVC) pattern. This is based on industry standard software design architecture, ensuring easier scalability and maintainability [21].

One noticeable difference however, is that rather than building my own database, I will be using an external API in order to retrieve bibliographic items from an existing database. The following diagram illustrates how the different components are connected, forming the overall software architecture.

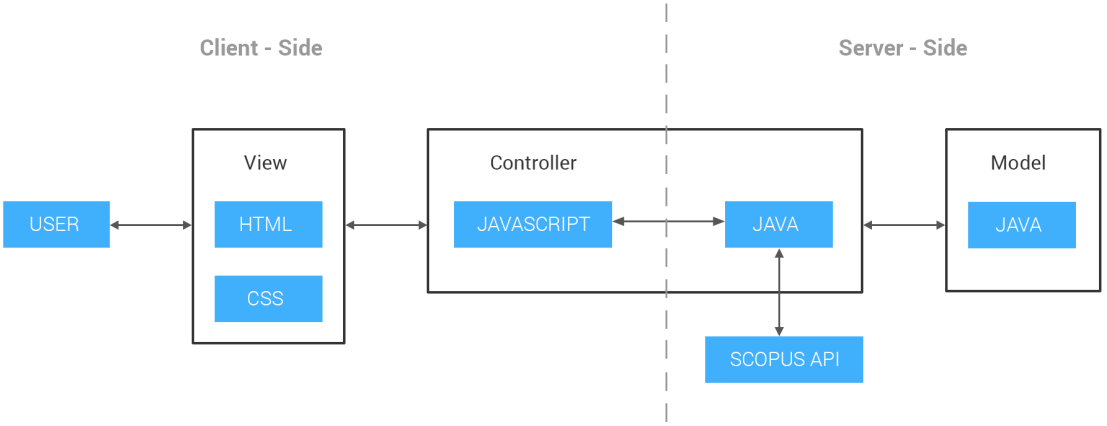


FIGURE 3.1: System component diagram

Given the requirements that have been gathered, generating a flow diagram will help to visualise the navigation between the different interface views.

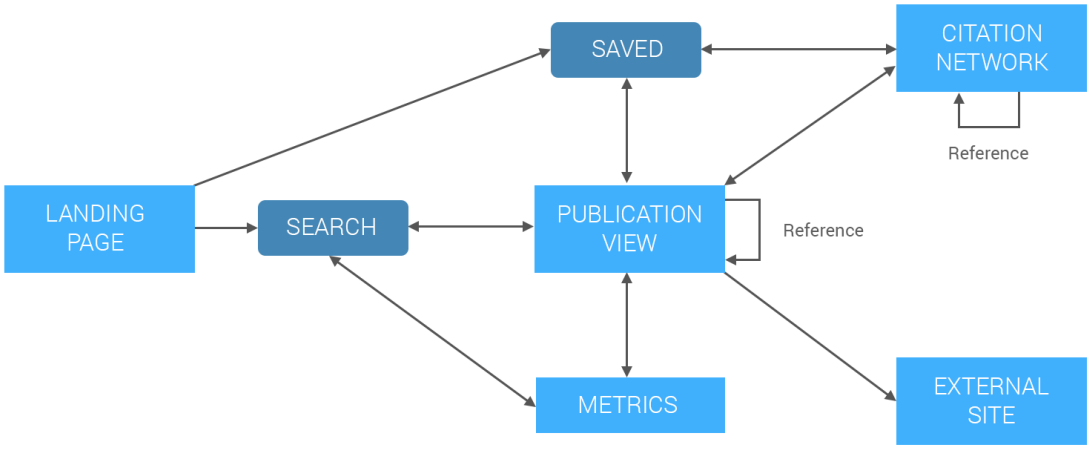


FIGURE 3.2: Application flow chart

When a user accesses the application for the first time, he or she should only be able to use the 'Search' functionality. Once an article has been selected from the list of search results, the application should update the view by displaying all the information regarding that article. From this view, the user can either access its citation network or the metrics page, which should host a series of graphs and data

visualisations. The 'Search' and 'Saved' components should always be accessible regardless of the page that is currently being viewed.

3.4 Website Wireframes and Design Decisions

After reviewing PaperCube and Semantic Scholar, I have grasped the effectiveness of minimalist design. I am therefore aiming to achieve the highest possible signal-to-noise ratio, displaying all the essential information that a user would be interested in (signal), and reducing any extraneous information that could result distracting (noise). As an attempt to make the interface feel familiar to the user, I applied a material design look and feel which is extremely common in modern applications.

The following wireframes represent certain views that the web application is going to feature. They have been designed with reasonably high fidelity to obtain a valuable first impression of whether the proposed interface is intuitive enough.

3.4.1 Search Drawer Menu

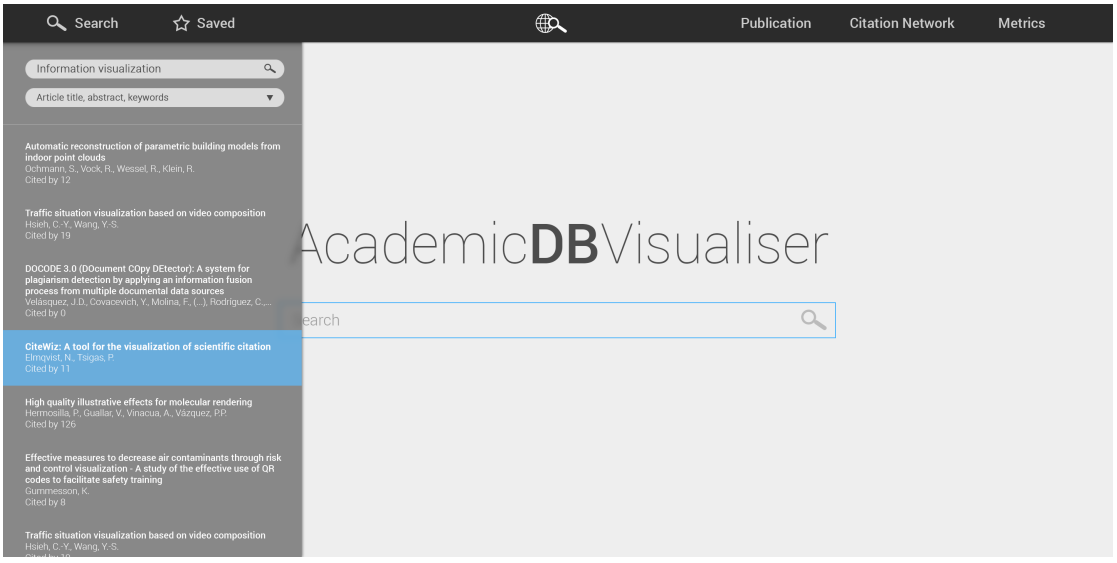


FIGURE 3.3: Wireframe showing the overlaying search drawer menu

The search drawer menu will slide-in when the user clicks on the search icon, and slide-out by clicking anywhere outside it. Usually, academic search engines will display a list of search results in a whole page, and once a result is clicked, the user will either be redirected to a new page or a new tab will open. However, in

my suggested design, the list of search results and the contents of a paper will be displayed in a single page view. This should allow users to rapidly assess the relevance of multiple papers without having to open additional tabs and windows.

The user will be able to save documents that he or she found particularly interesting or may want to revisit in the future. Once the user clicks on the 'Saved' icon, the content on the sidebar will be refreshed and show a list of saved articles instead. Likewise, clicking on one of these articles will update the view with the information regarding the selected paper.

3.4.2 Publication View

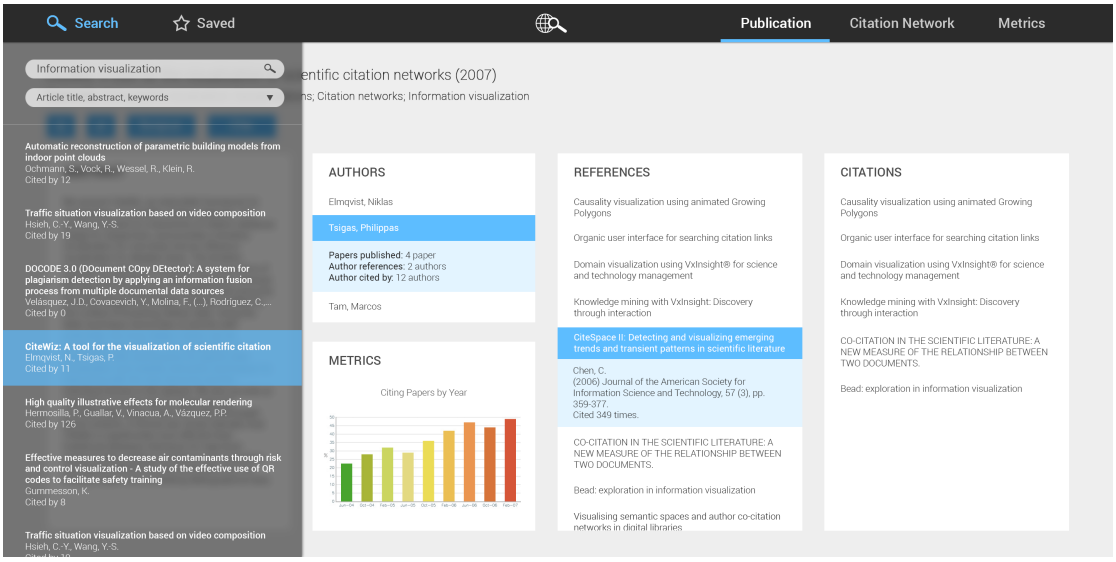


FIGURE 3.4: Wireframe showing the 'Publication' view

The 'Publication' view will present all the raw information regarding an academic article, including title, authors, publication, year, abstract, references and citations. This can be considered as the standard view of the system, and will be shown once a user selects a paper from the list of search results.

Unlike most scholarly search engines, I aim to display all the content within a single page application, eliminating the inconvenience of having to constantly scroll down to access all the information. Hence, I consider the table-based format used by Peter Bergström in his PaperCube prototype to be an instinctive way of visualising this academic metadata. Displaying data in a consistent structure will allow users to clearly identify specific information that they might be looking for.

A list of keywords should be clearly displayed, as these represent an accurate summary of the paper's content and will allow users to quickly assess if it is relevant to their research. When a 'Keyword' is clicked, the application should open the search drawer menu and perform a search query of that word or phrase. Similarly, when a user hovers over a reference or citation, it should respond by changing colour, indicating that it can be clicked. This will cause the page to be updated with the data of the paper that has been selected.

3.4.3 Citation Network

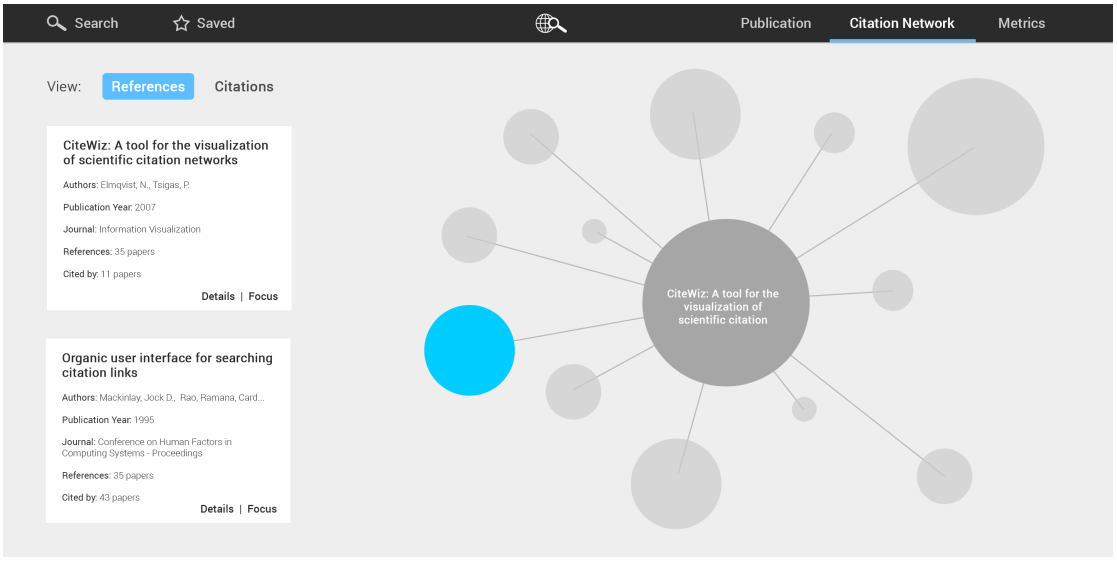


FIGURE 3.5: Wireframe showing the 'Citation Network' view

The undirected graph illustrated in figure 3.5 represents a citation network. The vertex in the centre embodies the currently selected article, and the vertices that are connected by edges represent the papers that are referenced by it. If the view is changed to 'Citations' rather than 'References', then the graph will show the publications that cite this paper instead of the ones that are referenced by it. The radius of a vertex depends on the popularity of the paper that it represents, and the length of its connecting edge depends on how far it is from the centre paper in terms of publication time.

Once users find a paper that they feel particularly interested in, this visualisation model can be useful for exploring other papers that are strongly related to that subject matter.

I decided to limit the citation network to only display the references or citations for the selected paper. In previous data visualisation projects such as CiteWiz,

developers have attempted to illustrate a complete citation map, with a much more extended network that shows the references of the referenced papers and so on. However, from my literature review, I concluded that users do not find these type of interfaces to be intuitive to use [17], due to an excessive overload of information. Hence, the design that I propose must have an ideal balance between ease-of-use and information volume. This should provide useful functionality without provoking a stressing learning experience or distracting users from their original research goals.

3.4.4 Metrics

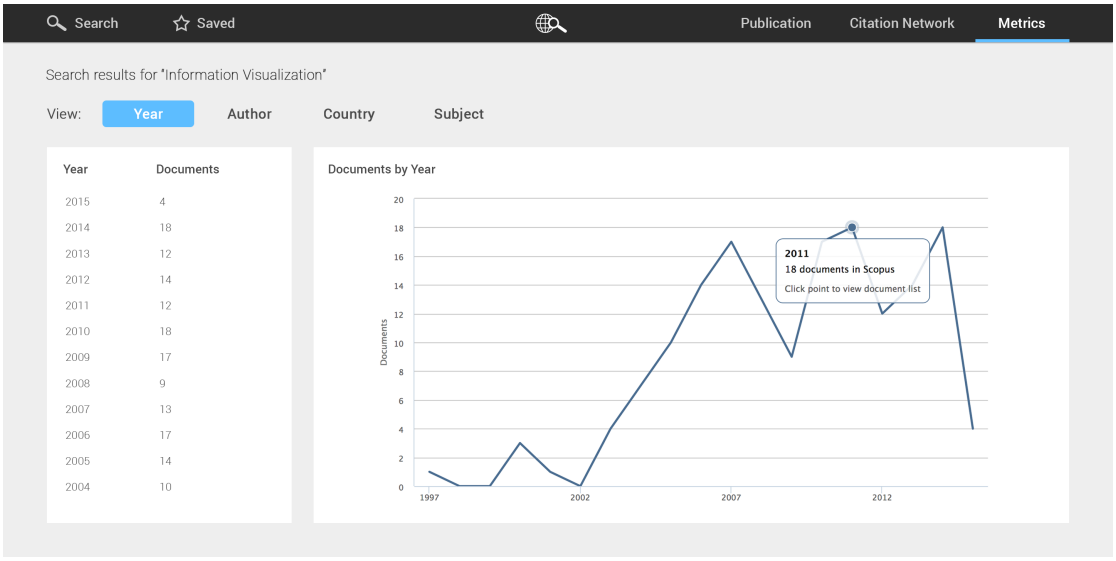


FIGURE 3.6: Wireframe showing the 'Metrics' view

The metrics page, inspired by Scopus' "Analyse search results" tool, will allow users to perceive an overview of the trends regarding their search queries. Although this feature has no immediate effect on the search experience, we believe that students can benefit from the data visualisations, as they provide insightful information that lead to a better understanding of their research area. For example, a graph showing the number of documents retrieved per year for a specific search query can help identify periods of time in which that topic was particularly popular.

Chapter 4

Implementation

The development process can be clearly split into two major components, the server-side and the client-side. In this section I will be discussing their implementation details as well as justifying the use of certain technologies for the development of the system, which I will be referring to as 'ScholarWiz'.

4.1 Server-side

Consisting of a series of Java classes, the server-side is responsible for retrieving bibliographic items from the Scopus database through their APIs, processing that data and sending it to the client-side as a JSON object, again, through a RESTful API.

4.1.1 HTTP Clients and Scopus API

In order to invoke a GET request from the Scopus APIs, the application needs to establish an HTTP connection. After trying Java's HttpUrlConnection, Apache's HttpClient and OkHttp, I decided to use the latter HTTP client due to its neat syntax and popularity amongst modern applications. In terms of performance, there was no considerable difference amongst the three. The following code shows how to perform a GET request and obtain a response object using OkHttp.

```
OkHttpClient client = new OkHttpClient();
```

```
Request request = new Request.Builder()
        .url(url)
        .addHeader("X-ELS-APIKey", API_KEY)
        .addHeader("Accept", CONTENT_TYPE)
        .build();

Response response = client.newCall(request).execute();
```

Listing 4.1: Sample of Java code showing how to perform an HTTP GET request via Scopus APIs

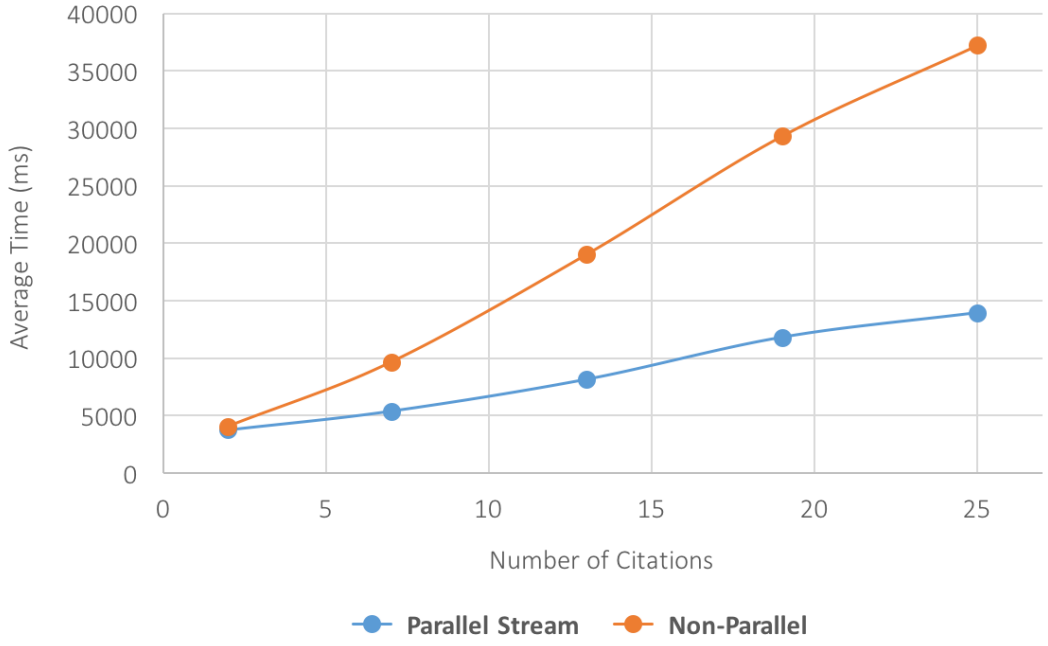
This request returns a JSON object, which I then process by extracting the required data from it. The 'url' parameter varies depending on the data that I wish to obtain from the Scopus database. The 'API_KEY' header stores the key that enables Scopus API usage and the 'CONTENT_TYPE' is 'application/json', which is the official Internet media type for JSON.

Unfortunately, some of the of the information that I wished to display required performing multiple GET requests to obtain several bibliographic items and merging them together into a single list. This was the case for the retrieval of citations, which refers to the list of papers citing a specific article. I was forced to perform a separate API call to obtain each individual document. This had an extremely negative impact on the performance of the system, as popular papers that have been cited frequently take much longer to load. In this case, I decided to sacrifice performance for the presence of such information, as I considered this to be invaluable for the user. Plus, the performance of the system is not included as one of the main objectives of this project. Since there are no plans to deploy the built application, it is worth showing the potential that the system is capable of featuring, despite affecting its performance negatively.

As an attempt to accelerate the retrieval of data, I implemented the use of Java streams to perform multiple HTTP requests in a parallelised manner. Parallelism offers the possibility of faster execution when more cores are available. However, their utility is highly criticised, as there is a common debate discussing whether their benefits surpass their downsides. In the case of my computer, I definitely noticed an improvement in speed when I benchmarked the system with and without the use of parallel streams (table D.1). For example, a publication which usually

took 19 seconds to load, it loaded as fast as just over 8.1 seconds, reflecting an almost 60% improvement in terms of performance.

FIGURE 4.1: Average time required to retrieve a document depending on the number of citations



4.1.2 Models

When a GET request is invoked from the Scopus API, it returns a large response object in JSON format (appendix B). This contains a lot of data that is not required on the client-side. Therefore, I pre-process the response object by extracting only the relevant metadata (e.g. publication title, authors, abstracts, etc.) before returning it to the front-end. This procedure is achieved by simple string matching, which looks for the unique names of certain attributes and copies their value. Whilst this procedure does add an extra step of complexity to the retrieval of data, it means that it will save the client-side from the hassle of processing and filtering out large quantities of irrelevant data.

Therefore, I created three classes that encapsulate scholarly metadata based on varying degrees of detail: result, node and publication. For example, a 'result' object contains the minimum amount of data, as it is only used to represent the individual results obtained by performing a search query. On the other hand, a 'publication' object encapsulates much more data, but it is only used once the

user clicks on an article that he or she is interested in. The motivation behind this approach is to optimise the performance of each individual system feature, enabling a smoother and more pleasant user experience. Rather than making the user wait for all the data to load at the beginning, the application only loads the necessary data at each stage.

The following list shows the attributes and describes the use of the objects that I mentioned earlier.

Result

- **Attributes:** id, title, year and cited-by-number.
- **Use:** returned in the form of a list when users perform a search query.

Node

- **Attributes:** id, title, publication-name, year, cited/referenced-by-number and authors.
- **Use:** represents the documents from the citation network of a specific paper.

Publication

- **Attributes:** id, title, abstract, publication-name, volume, page-range, link, year, authors, keywords, references and citations.
- **Use:** contains all the relevant metadata regarding an academic article and is returned when a user wants to analyse a specific document in greater detail.

4.1.3 Spark Framework

As I have mentioned before, Spark is a simple and light-weight framework that facilitates the creation of web applications in Java 8. It was used to create a set of routes through which the front-end communicates with the back-end in order to request data. Due to the nature of ScholarWiz, it only has to perform GET requests, since it only displays information and users are not allowed to create or update any objects. The following example is used to get a list of search results.

```
get(API_CONTEXT + "/searchResults/:query/:limit/:sort", (request,
    response) -> {
    try {
        return service.searchPapers(request.params(":query"), Integer.
            parseInt(request.params(":limit")), request.params(":sort"));
    } catch (Exception e) {
        System.err.println("Error: Failed to retrieve results for the
            given query (" + request.params(":query") + ")");
    }
    return null;
}, new JsonTransformer());
```

Listing 4.2: Sample of code showing how the server deals with a GET request

Once a route matches the defined path, including the three different parameters (query, limit and sort), the request is invoked. In this case, it will call the 'searchPapers' method that is hosted in the 'Service' class. The parameters include the actual search query, the limit of results and the sorting order. If the request is successful, the API will return a JSON object consisting of the list of papers retrieved from the given search query. Otherwise, it will print an error message on the server-side console and return null to the client-side.

The 'Service' class contains all the methods involving the retrieval of data from the Scopus API. These methods establish an HTTP connection and send a GET request for a specific piece of data, as I described in section 4.1.1. The most important ones include 'getPaperDetails', which returns a 'Publication' object, and 'searchPapers', which returns an ArrayList of 'Result' objects.

4.2 Client-side

The majority of the development time has been dedicated to the front-end, where I use HTML, CSS and JavaScript to create an interactive web application that is capable of loading data dynamically. A lot of attention has been placed to create a user interface that is truly intuitive and benefits from powerful data visualisation tools that facilitate the perception of bibliographic metadata.

4.2.1 AngularJS Framework

The whole client-side is encapsulated within the popular AngularJS framework, which enforces a clear Model-View-Controller structure and augments HTML's syntax for the creation of more interactive user interfaces. Due to its data binding and routing capabilities, I was able to develop my project as a single page application that enhances user interaction.

Routing

AngularJS provides an 'ngRoute' module which enables different content in a web application to be addressed with a unique URL. These are known as 'routes', and they are defined after the hashtag (#) in an URL. This means that a website can be built as a single page application, but its different views can still be addressed individually.

To configure this behaviour, I had to define the URLs in the '$routeProvider', which decides what HTML template is being displayed depending on the current route. Apart from the homepage, the only views available are publication, citation-network and metrics, each containing an ':arg' parameter at the end of the URL which depends on the unique Scopus ID of the paper that has been selected (e.g. /#/citation-network/36749009665). This means that if the web browser is given a direct link, including an existing ID, it will access that specific paper, otherwise, it will redirect to the homepage.

Controllers

Each HTML view has its own controller. These are used to augment the AngularJS scope, enabling the definition of dynamic scope variables which can be accessed directly from the HTML mark-up. This is an extremely useful feature since it allows me to easily change the values of HTML elements programmatically.

Asynchronous JavaScript and XML (AJAX)

AngularJS provides AJAX technology in the form of $http control, which allows the request and reception of data from the server without having to reload a whole web page. This feature enables a much smoother and fluid user experience, as it allows the application to dynamically update information on the current page.

```
getSearchResults: function (query, limit, sort, callback) {
    $http.get('/api/searchResults/' + query + "/" + limit + "/" + sort)
        .success(function (data) {
            if(data !== "null")
                callback(data);
            else
                callback();
        });
},
```

LISTING 4.3: Sample of JavaScript code showing how to perform a GET request via the server API

This example shows how the front-end can perform a GET request by calling the 'getSearchResults' method, in order to obtain a list of search results from the server. It attaches the corresponding parameters to the API address. It can be observed how AngularJS makes the use of promises (success) to determine the following action once the GET request has been responded. Then, only return a data object if the server managed to retrieve search results for the given query.

4.2.2 Material Design Lite (MDL)

Material Design Lite is a front-end framework that helps building modern websites with a Material Design look and feel. This facilitates the task of designing a user interface that is attractive and consistent. Despite offering a limited number of strictly crafted components, they resemble those commonly found in mobile applications, giving users a higher sense of familiarity. Screenshots of every single view from the built application can be found in appendix C.

With MDL I was able to easily create an expandable sidebar and implement a search bar inside it (figure 4.2). When a query is performed, the list of results is shown within the sidebar, without reloading the current page nor altering the content in the current view. Using MDL's default components, I was still able to easily override their CSS styling and re-use them. An example of this can be seen in figure 4.3, where MDL's default 'Card' component has been modified to become a table of clickable 'References'.

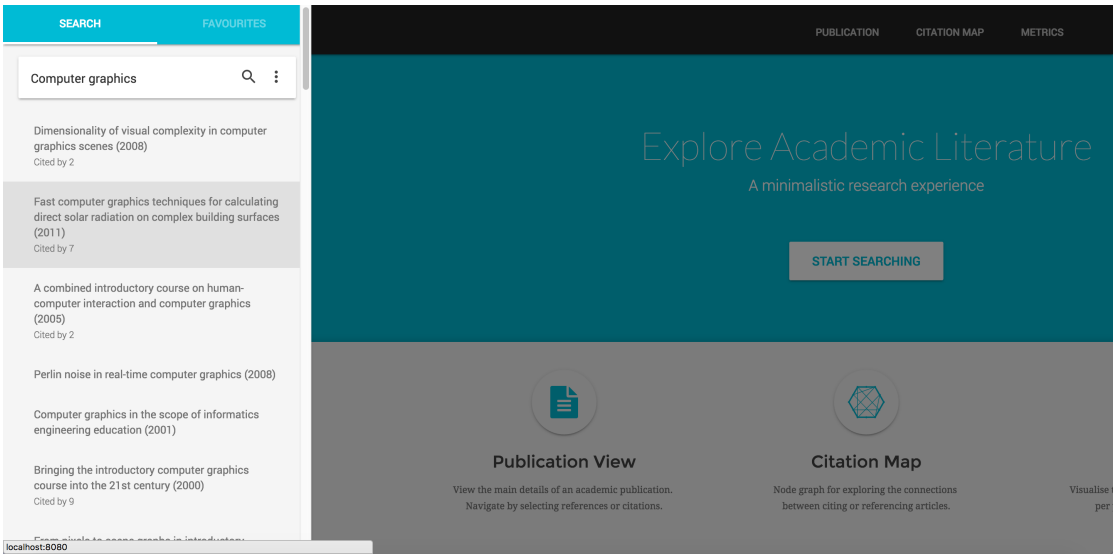


FIGURE 4.2: Screenshot of ScholarWiz's Search drawer menu

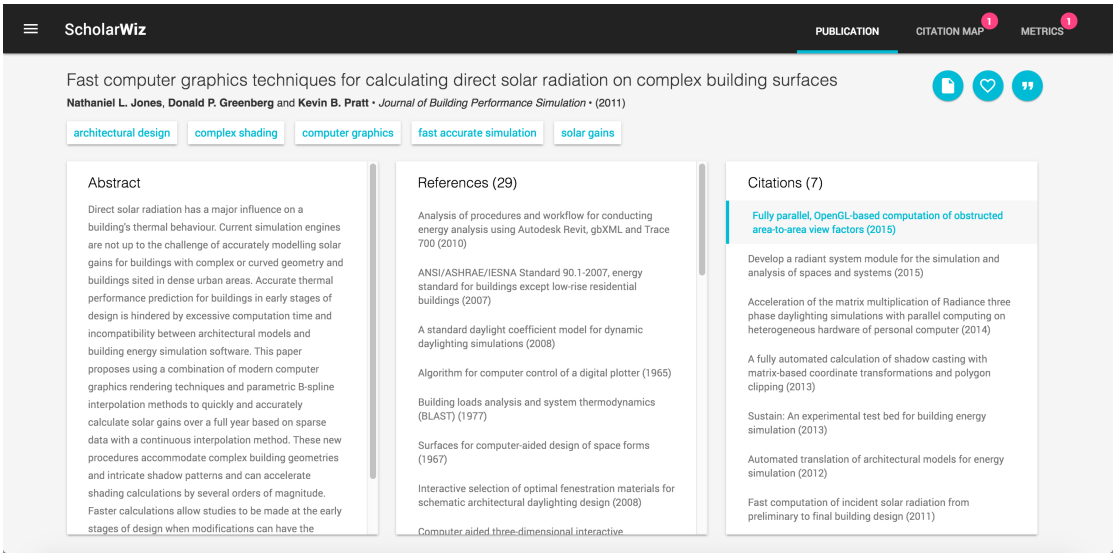


FIGURE 4.3: Screenshot of ScholarWiz's Publication view

4.2.3 Data-Driven Documents (D3)

D3 is a JavaScript library that enables users to interact with data through powerful data visualisations. One of its stronger advantages over other libraries is the high level of customisation, giving programmers almost total control of the aspect and behaviour of the SVGs (Scalable Vector Graphics). This makes it the ideal tool for implementing the citation network, which consists of an undirected graph with each paper represented by a vertex.

In order to arrange the data to be read by D3, the application needs to pass a list of nodes and a list of links, which represent the vertices and edges of a graph respectively. For the nodes, a list of citing or referenced papers is passed, which I obtain by performing a GET request on the server. Then, a list of links between the currently selected document ('source') and its references ('target') is manually generated. This should connect each vertex to the centre vertex, which represents the currently selected paper.

The size (radius) of each node depends on the popularity (number of citations or references) of the paper that it represents, and the length of each link depends on the difference in publication years. A simple sizing function and size boundaries are used to ensure that the radius of extremely popular papers are not infinitely large, and papers without any citations or references are not infinitely small. I also had to take into account that for some records in Scopus's database, there is missing citation information. This was mitigated by setting a default size to those documents with missing data.

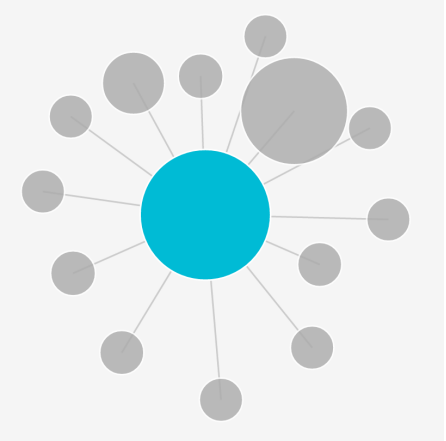


FIGURE 4.4: Vertex-edge graph generated with D3

Finally, to enhance the user experience and interaction of the citation map, I added responsiveness to mouse hovering and clicking. If the user hovers over a node, the vertex will change colour and expand in size, as well as displaying a text container on the side, including information regarding that specific paper. If the user clicks on a node, the application will reload the citation map to display the connections from that paper. Additionally, the vertices can be dragged around in case they overlap each other when the graph is rendered.

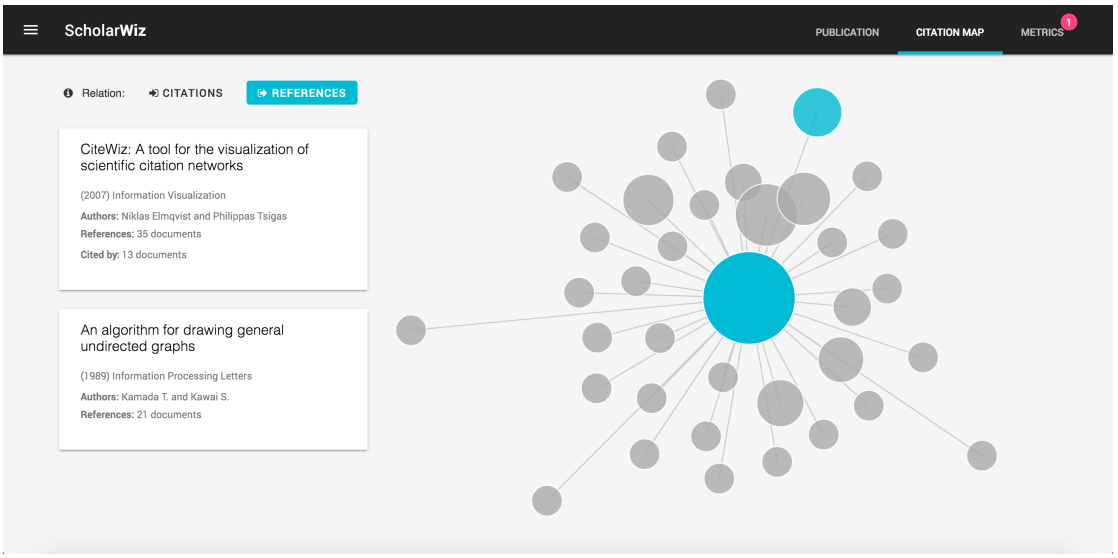


FIGURE 4.5: Screenshot of ScholarWiz's Citation Map view

4.2.4 Google Charts

Google Charts is a recent library that provides a large number of easily-implementable chart types, including line charts and bar charts. Whilst not offering near as much level of control as D3, Googles charts are already responsive and interactive by default.

I used Google's column chart to illustrate the distribution of referenced papers by year. Given a list of referenced papers including their respective publication years, it was a case of creating a mapping between a year and the number of documents from the list that had been published that year. This results in the following chart.

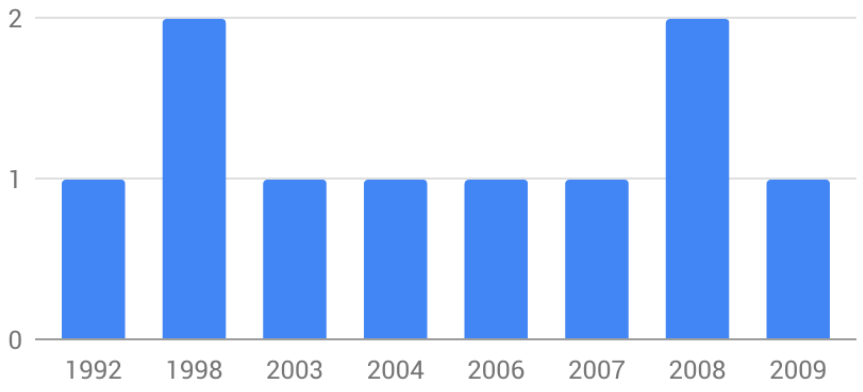


FIGURE 4.6: Column chart showing the references of an article by year

As we can observe, this does not provide a very useful perception of the distribution of references from this particular paper, as it only displays the years that are mapped to at least one referenced paper. Hence, the years without any publications have also been included for a clearer observation of how the references are distributed per year.

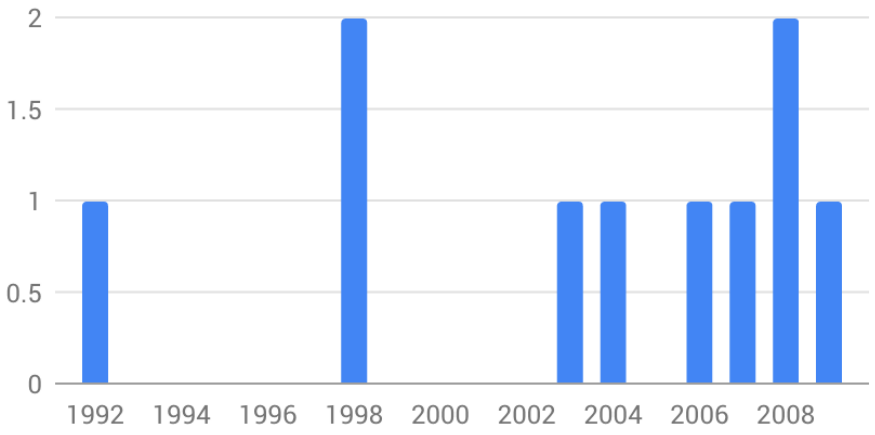


FIGURE 4.7: Column chart including years without references

Chapter 5

Testing

The following section describes the measures that have been taken to ensure the correct operation of the system. However, testing a system which obtains its data from external sources and is strongly based on the front-end, can sometimes result in a bit of a struggle. Therefore, a combination of automated testing and manual testing in particular, has been applied since early stages of the development process.

5.1 Unit Testing

Retrieving data from a third party API means that the application's functionality relies on that data being provided in a consistent manner. At the lowest level of the application, there is a Service class enclosing a series of methods used to request scholarly metadata and extract certain parts of it to created self-defined objects. These functions need to work flawlessly for the corresponding content to be displayed on the front-end. Thus, a series of unit tests have been written using the JUnit testing framework, in order to repeatedly check whether the server-side provides the necessary data to the client-side.

Figure 5.1 shows the test cases that have been defined. This allowed me to easily check whether new changes made to the application disrupted any of the already working functionality. The tests accurately compare attributes such as publication title with the values retrieved from the Scopus API, and are successful if they match the expected output.

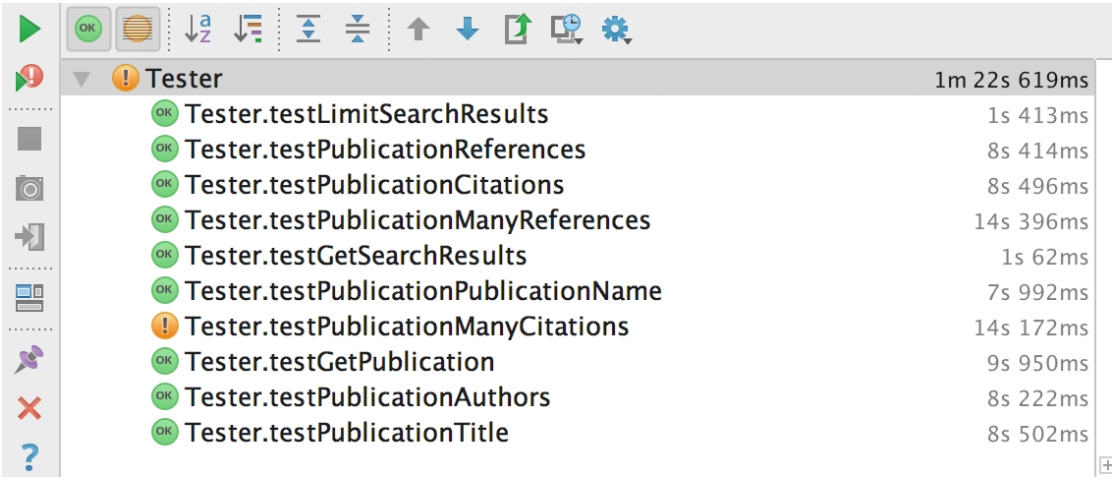


FIGURE 5.1: Unit tests written with JUnit in JetBrains IntelliJ IDEA

One of the tests shown in figure 5.1 actually fails. This involves the retrieval of the list of documents that cite a specific publication. The reason for this is that the Scopus API only provided a maximum number of 25 documents when requested for a list of citations, whilst the publication that I used for that test had been cited by 26 documents. Unfortunately, I could not mitigate this limitation, thus, the application will only display the first 25 cited-by articles.

5.2 Manual Testing

Whilst automated testing was an ideal technique to ensure that the server-side behaved as expected, manual testing became the most suitable procedure for testing the client-side. This is where I play the role of an end user and use all the features of the application to guarantee correct behaviour. The list of functional requirements that was defined in section 3.1.1 is representative of the test cases that the application needs to satisfy on the front-end (appendix D.2). This method allowed me to discover some obvious faults as well as some more discreet issues.

5.2.1 User Input

The only component of the system that allows user input is the search bar, which means that it must be tested for a variety of possible inputs that may harm the system. Therefore, I have produced the following table, showing possible inputs that were tested and their respective outcome.

Test Case (query)	**Outcome**	**Mitigation**
"Graphics"	As expected	n/a.
"Computer graphics"	As expected	n/a
" "	Error	Prevent searching if input query is null
"-"	As expected	n/a
"/"	Error	Remove all forward slash occurrences from query
"\"	Error	Remove all back slash occurrences from query

TABLE 5.1: Table showing search query test cases

By manually testing the search functionality with some unexpected queries, I was able to detect some threats that would cause the system to halt. The reason for this is that certain characters such as empty spaces and forward slashes modify the API address (/api/searchResults/query/limit/sort) in a way such that it is read incorrectly. To avoid this kind of issue, the user is prevented from performing a search if the query is empty, and illegal characters are replaced with empty spaces.

5.2.2 Graph Rendering

FIGURE 5.2: Rendering error

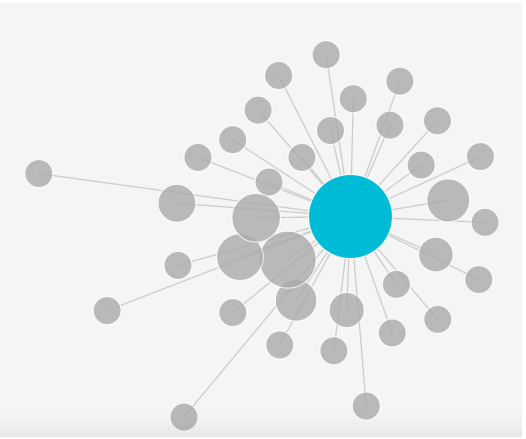


FIGURE 5.3: Correct rendering

Figure 5.2 shows one of the system errors that I managed to find through manual testing and consequently fixed. The source of the problem was the shallow copying of an array that contained the vertex objects being displayed in the graph. This caused D3 to throw an infinite series of errors and render the graphic as shown in figure 5.2. The solution was to simply apply deep copying to the object array.

Although this was an imprudent mistake and fixing it was straight-forward, the difficulty involved discovering the rendering issue in the first place. This bug only seemed to have an effect when I quickly switched between tabs in the application, seemingly corrupting the data that is provided to the D3 graphing library. Hence, manual testing served practical for uncovering an issue that could have easily gone unnoticed otherwise.

5.3 Usability Testing

This evaluation methodology required the approval of an ethical committee as it involved participants. The complete raw results and their respective graphs can be found in appendix E. I will also be evaluating the system by discussing the outcome of some of the survey questions.

5.3.1 Aim

The purpose of this user study was to obtain valuable feedback regarding the usability of the proposed user interface as well as the discovery of minor system defects and bugs. The results serve as an accurate indication of the success of the system. In particular, I want to assess its ease-of-use and how it compares to existing scholarly search engines.

5.3.2 Method

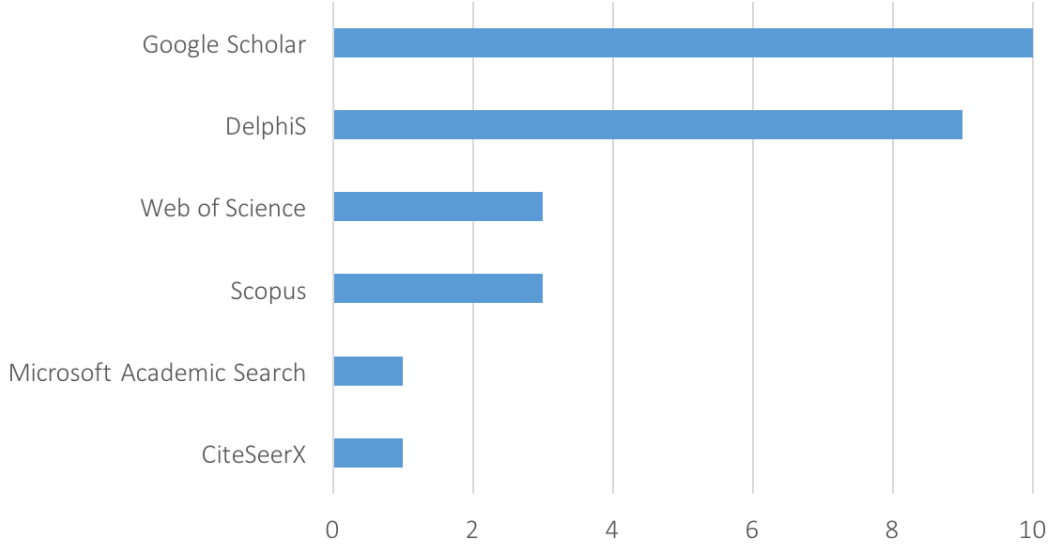
Ten students were asked to spend around 5 to 10 minutes trying out the features offered by the web application. A list of simple instructions was provided which walked participants through most aspects of the system's functionality, although this was optional and some preferred to test the application by their own likings. Once they felt familiar enough with the system, I asked them to complete a short

questionnaire (appendix E.1) evaluating their experience. This includes a mixture of multiple choice, rating and feedback questions which examine both their previous experiences with existing scholarly systems and their thoughts about the system that I propose.

5.3.3 Results

To begin with, participants were asked about their previous experience with academic databases. This shows some extreme, yet expected, outcomes. One hundred percent of participants were aware of, and have used Google Scholar before, whilst the University of Southampton's own service is almost as recognised. However, this contrasts with the lack of awareness of every other scholarly search engine.

FIGURE 5.4: Scholarly search engines that you have previously heard about



Given that participants possessed varying levels of experience involving academic search engines, it is then fair to compare their opinions regarding ScholarWiz and other systems. From figure 5.6 we can infer that there was not a single participant that struggled to learn to use the ScholarWiz system.

Some students even believed that the learnability of the interface was excellent, which contrasts with the ratings shown in figure 5.5 for their most commonly used service. This confirms the achievement of a very important goal, which is designing a web application that students find intuitive to use.

FIGURE 5.5: Rate the learnability of your favourite engine's user interface

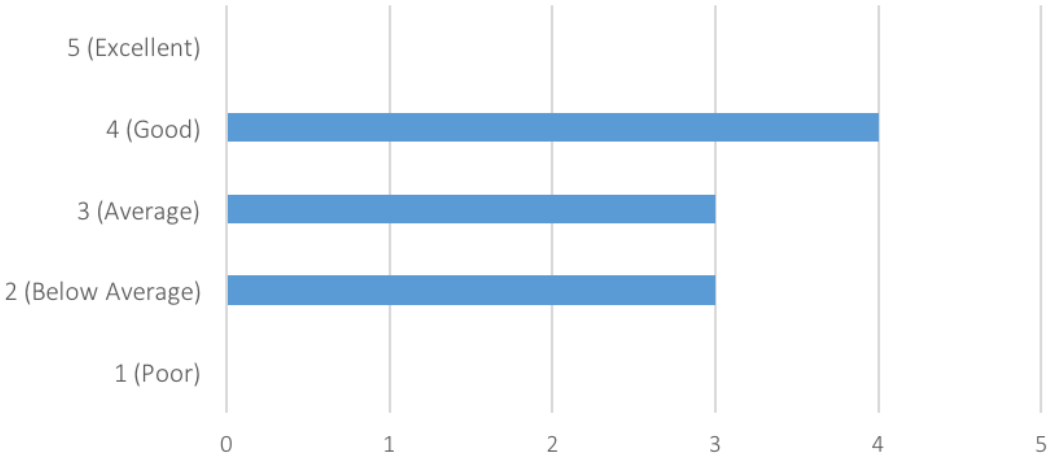
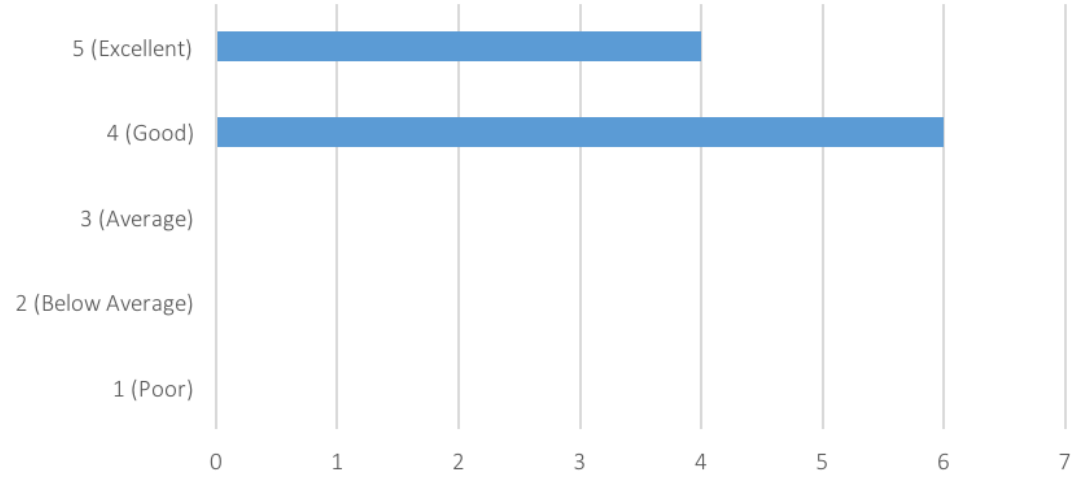
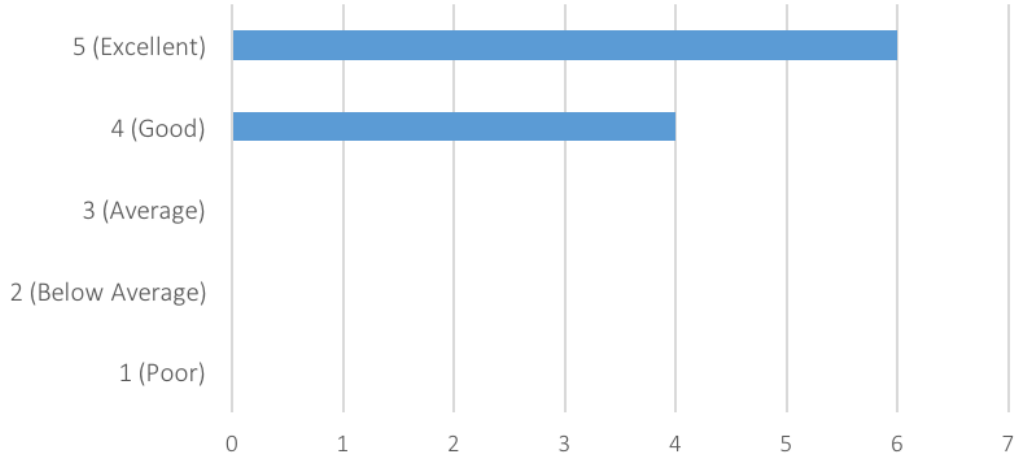


FIGURE 5.6: Rate the learnability of ScholarWiz's interface and ease of navigation



As I have mentioned before, ScholarWiz has been designed with the purpose of reducing the cognitive load upon users. This means that the lack of important information suddenly becomes a possible risk. Nevertheless, usability testing has established that the information that I have decided to display, seems to be both sufficient and useful. When participants were asked what additional academic data they would like to see on the application, most of them stated that the current information shown includes everything they would ever need (figure 5.7).

FIGURE 5.7: Rate the quantity of information presented for a paper by ScholarWiz



5.3.4 Written Feedback

The survey was also used to ask participants to provide their own opinions about the application, including their favourite features as well as suggestions for future improvements (appendix E.2).

Most of them agreed that the citation map was an interesting, instinctive and informative technique for exploring either the citing or referencing connections between academic papers. Students also seemed to appreciate some of the more technical aspects of the user interface, including the use of colour, sizing and length in the citation network as means for conveying information. One of the participants pointed out "abstract, citations and references on a single contained page" as a positive feature, referring to the attempt of fitting all the information within a single page application. Again, this was one of my objectives and it seems to have been accomplished.

In terms of improvements, participants managed to point out small bugs which could easily go unnoticed. These have been fixed right after the user study ended. Many commented that they would appreciate a direct link to a PDF. Unfortunately, I was not able to get hold of publications' PDFs, instead, users are redirected to the Scopus website, where the document is hosted. Other proposals from individual students included more customisation to the 'Favourite' feature and more referencing formats such as BibTeX to be added to the 'Cite' tool.

Lastly, participants were asked to suggest potential features that they would like to see in the future. These included the addition of a history of viewed articles, and surprisingly, only a few mentioned the ability to search on the basis of authors.

Chapter 6

Project Management

The following section describes the progress made throughout the course of this project as well as mentioning the tools that helped to manage it, including skills audit, risk assessment and Gantt charts.

6.1 Skills Audit

From the start, generating an audit of personal skills (table F.1) served as a good indication of what technologies and academic skills required prior mastering. Although I had no previous experience with some of the technologies used, one of the modules that I undertook in the first semester strongly helped me to gain experience involving the development of web applications in JavaScript. This became an ideal form of preparation, since it facilitated me the opportunity to improve all the skills that I particularly lacked experience with, but were necessary for the development of the system during the second semester.

6.2 Risk Management

It was of vital importance to carry out a risk analysis (table F.2) in order to identify threats likely to arise during the completion of the project, as well as possible mitigations to either eliminate them or reduce their effects.

Initially, the workload and deadlines from other modules were considered to be substantial risks across the whole term. Nonetheless, these resulted to not have

such a severe impact on the project as predicted. The main reason for this was excellent time planning and work efficiency during less-busy periods of time. Setting up a GitHub repository from the beginning of the implementation stage ensured that I always had a back-up of the code, as well as providing the benefits of version control, which helped me to identify what changes in the code were causing certain system errors.

Problems with third-party APIs were accounted for, but they still resulted to be a significant issue which caused rare downtimes, occasionally requiring customer support. Elsevier's Scopus was my final choice as the source of data, but in the case that this suddenly became unfeasible, I had searched for other alternatives that could replace this API, as mentioned in section 2.4.

It is important to note that one of the risks that I had not accounted for, was the downtime of the Southampton VPN, which prevented me from accessing Scopus content whilst I was back home during the Easter holidays. This meant that I could not test some of ScholarWiz's functionality, as it requires an institutional IP address to access certain Scopus data. Fortunately, this issue was only temporal, but it is factor that I will want to take into account in future projects, as it can become severely troubling.

6.3 Planning, Progress and Time Management

Gantt charts were used to ensure that the scope of the project was adequate for the time available. This helped me to allocate estimated time periods for certain tasks, allowing the creation of an initial project schedule. The Gantt chart was updated weekly, as the work progressed, providing a rough estimate of the work efficiency.

In general, the actual progress was relatively smooth and well-spread, resulting in only a few differences when compared with the initially set schedule (Figures 6.1 and 6.2). Although the time spent on certain tasks varied from the initial plan, I still managed to complete the essential report writing milestones slightly ahead of schedule.

During the first semester, most of the work consisted in reviewing existing systems and past projects in order to to identify ways in which the proposed system could be designed and built. The literature review and identification of useful

technologies took longer than expected. The reason for this is the immense quantity of online resources that are available for the development of web applications. Several tools, frameworks and APIs had to be tested before identifying which of them would be used for certain. On the other hand, designing the user interface and system architecture was a quick process after gaining valuable knowledge and inspiration from the research stage. Finally, report writing during term-time, in the middle of other deadlines, explains why the interim report required almost as much time as the final report, which was written during the Easter break.

As planned, I started implementing the server-side before the end of the first semester. However, no time was spent on the project throughout the Christmas break due to revision and other upcoming deadlines. After the exams, there was a period of time during which workload from other modules was minimal, allowing me to make considerable progress involving the development of the system. Whilst most of the work during the second semester fell under implementation, the final week before the Easter break was dedicated to carrying out a user study, as well as testing the system thoroughly. As a result of this, I was able to allocate the following weeks exclusively for report writing. Despite beginning to write the final report almost 4 weeks later than planned, I felt that the remaining time was still sufficient to complete the writing and producing the final product in LaTeX.

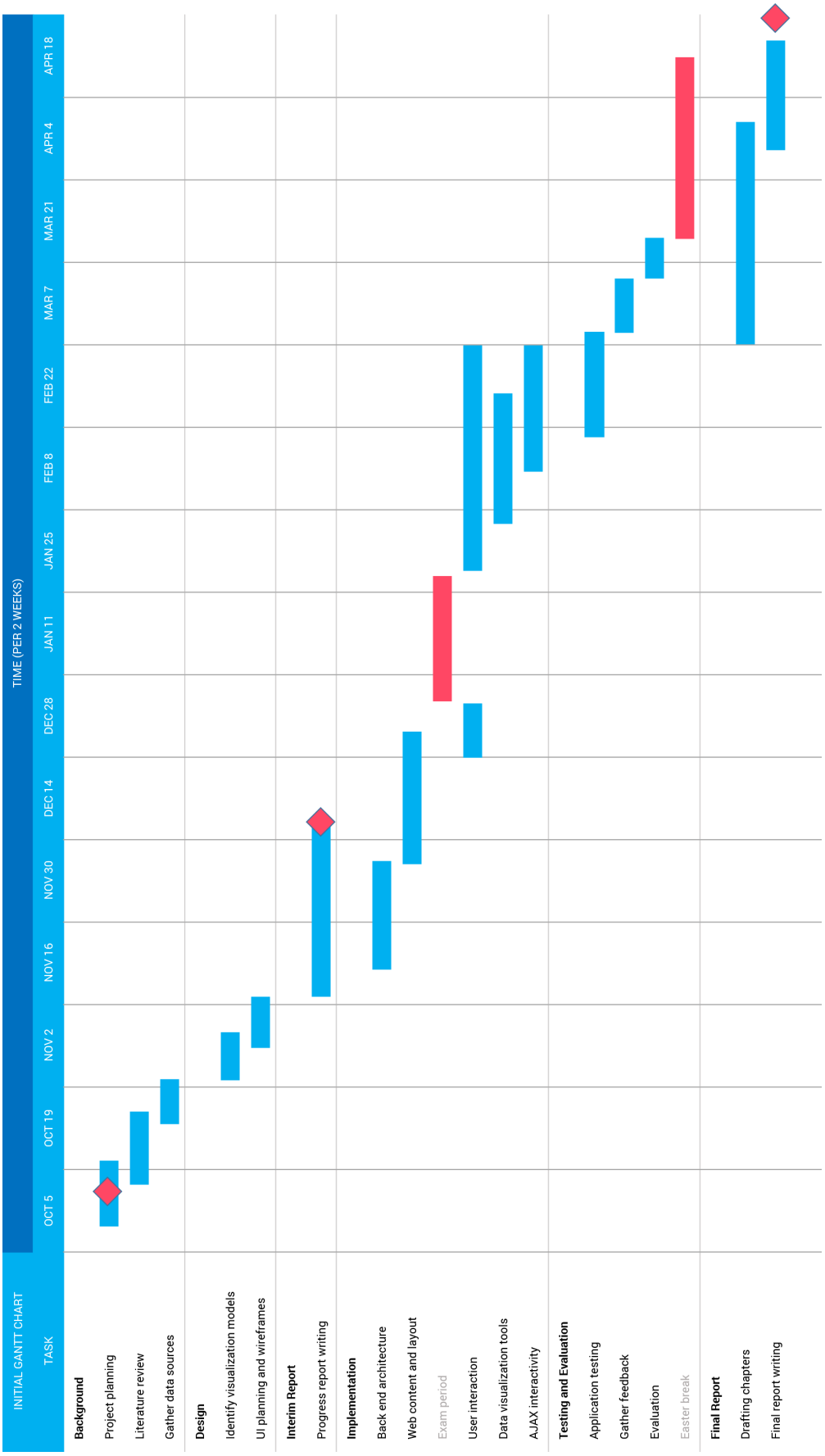


FIGURE 6.1: Estimated Gantt chart

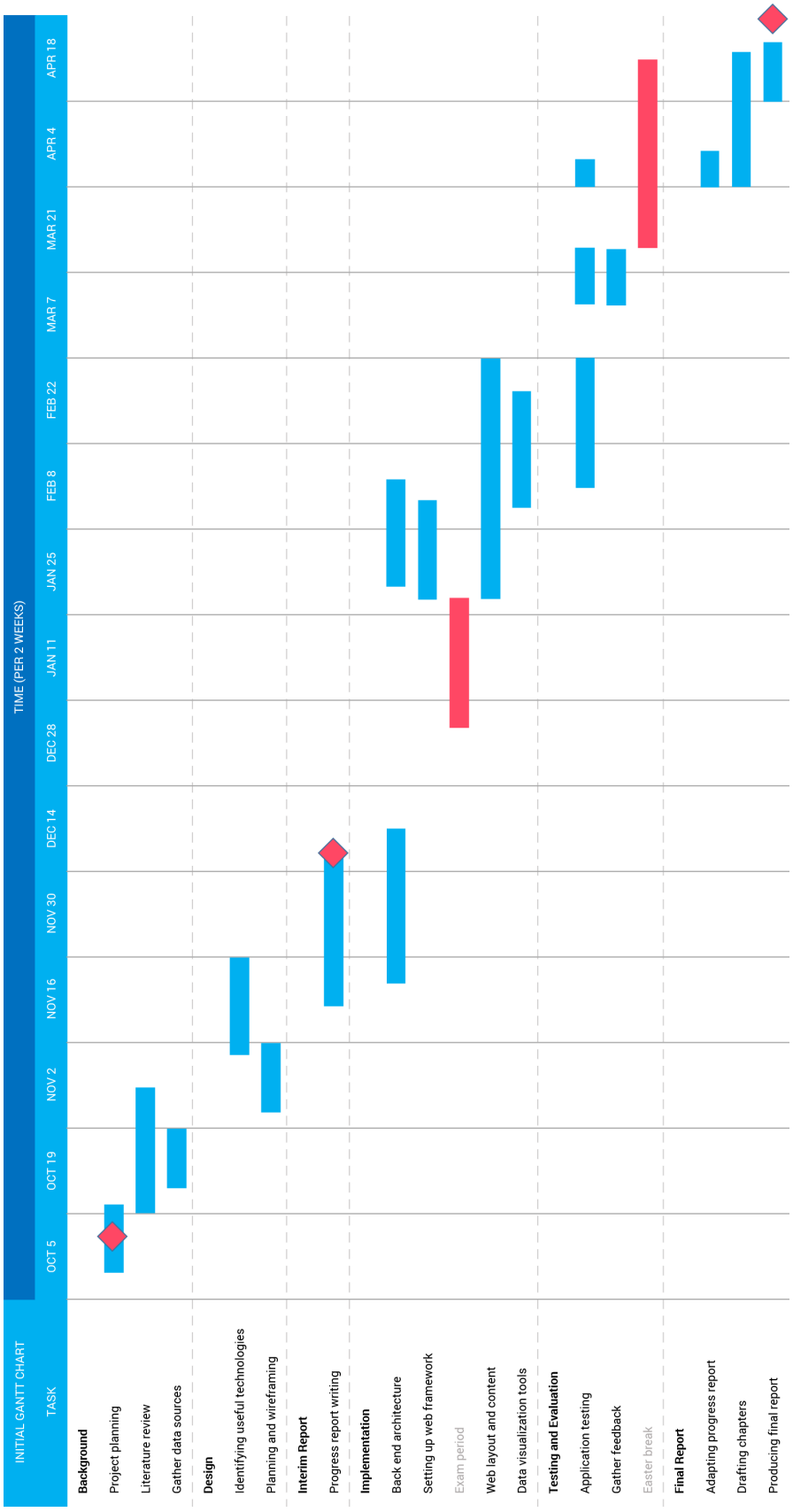


FIGURE 6.2: Actual progress Gantt chart

Chapter 7

Conclusions

This final segment will summarise the findings of the user study, discuss some of the possibilities for future work and present a personal reflection about the project as a whole.

7.1 Findings and System Evaluation

After reviewing all the opinions provided by the participants who took part in the user study, I can conclude that the proposed system has been successful and the initial goals have been achieved. It is worth noting that 90% of the students said that they would use the ScholarWiz system if it became a deployed application, which reflects the extensive work and attention to detail that has been put during the development and testing stages to perfectly polish the system.

Firstly, participants provided mixed opinions about their past experiences with scholarly search engines. Overall, they rated existing interfaces to be fairly average. Unsurprisingly, Google Scholar was the most commonly used digital library, followed by the University of Southampton's in-house service called DelphiS. However, I did not expect such little repercussion from other sites such as CiteSeerX, which only one out of the ten participants had previously heard about. Despite the high quality content and wide-ranging features offered by subscription-based databases such as Web of Science and Scopus, students admitted that they felt Google Scholar and Google's standard search engine were sufficient for their research purposes.

From the ScholarWiz results, it can be stated that users found the interface to be intuitive, practical and innovative. I placed a lot of attention specifically selecting what metadata I believed students would be interested in. This resulted to be effective, as participants affirmed that they were able to find all the information they would possibly need.

One of the functional requirements that was not implemented was author-based searching. Yet, only two participants suggested this feature to be included in the future. The lack of demand for such a common feature could be explained by low academic author awareness amongst students. This means that they are much more likely to search by topic and keywords, which is where the focus was placed.

One of the goals that I set from the beginning and managed to achieve was to provide bidirectional navigation amongst papers. By this I mean being able to view the references of a specific paper, as well as the list of papers citing it. This functionality is not provided by Google Scholar, which does not index the references of academic articles.

On top of this, participants found the citation map to be their favourite feature. They implied that it was a quick way of perceiving large amounts of information and a novel method for discovering potentially interesting articles. This reflects the advantages of using visual graphics and interactions within them to augment the exploration of scholarly content.

7.2 Limitations and Future Work

The decision to rely on Scopus' API for the access to a bibliographic database can be considered to be successful. This way, I have been able to focus on the development of the user interface with the guarantee of having a source of data that could provide genuine academic content, including connections amongst papers. In spite of that, it means that the system is subject to some of the common performance and content limitations imposed by the use of third-party APIs. For example, my intention was to show the complete list of documents citing a paper, but I was only able to retrieve the first twenty-five.

Moreover, the use of Scopus data also meant that the system is restricted from being deployed. Elsevier kindly allowed me to use their APIs for the purposes of this project, given that it would remain internal within the university and the

application would not become an actual product. Despite this, I believe that I have investigated the implementation of a design and visualisation models that have proven to succeed amongst students.

In the future, the server-side could be remodelled to rely on its own database. This could allow for more efficient retrieval of data and wider data visualisation possibilities. For example, the information displayed by the graphs in the metrics page is limited, but can certainly be augmented given a database that is more adapted to the ScholarWiz system.

Additionally, the creation of user accounts is certainly a feature that should be considered in the future. I decided to ignore this possibility for the current prototype in order to reduce complexity and reduce the time spent developing functionality that is outside the project scope. But this also carried some limitations within the system, for example, favourite articles are only stored in the current session. Once the user closes the browser tab, the favourite list will disappear. Therefore, providing users with the ability to create a profile would enable the 'Favourite' feature to be much more useful, as well as facilitating the implementation of a history of articles viewed, which was one of the ideas suggested in the user study.

One of the current issues that some users complained about was the unavailability of certain documents. Currently, when a user clicks on one of these unavailable documents, the system attempts to access the document and then spawns an error dialogue box if it is unavailable. Ideally, missing documents should be greyed out and clicking on them should be disabled. Unfortunately, I was not able to achieve this without affecting the performance of the system, as there was no way of telling whether a document was missing or not without making an attempt to access it in the first place. This is an interface defect that should be dealt with as well as providing direct links to PDFs.

Once the current flaws are fixed and refined, it is time to start thinking on ways of augmenting the system in the future. This could implicate adapting the system for wider audiences than just students. But it is worth noting that the requirements for academics certainly differ from those for students, which means that some of the information and tools that were not considered to be relevant, suddenly become demanded by the audience. For example, the lack of focus that I gave to author-related functionality was not a problem amongst students. However, this is believed to be an important aspect of scholarly search engines for researchers, as they have published many of these papers themselves.

Whilst augmenting ScholarWiz with more content and features seems a way to attract wider audiences, it is also important to reconsider the original motivations and goals of this project. The excess of content and features were initially identified as the major issues that these kind of systems suffer from. Thus, I wanted to reiterate the importance of excellent user experience, and that it must not be sacrificed for the addition of potentially irrelevant information.

7.3 Personal Reflection

In terms of planning, the system was completely built within the set schedule, proving that it had an adequate scope. The objectives that were set initially have been achieved, although some of the functionality defined in the requirements has been omitted. In a more personal level, I have gained indispensable experience involving the design and building of web applications.

The project investigated a topic that affects students during the course of their university degree, but that is commonly overlooked. Although there are plenty of scholarly search engines on the web, it seems that none of them have achieved to build an interface that truly suits one of their key audiences, students. Hence, I have explored the importance of good user experience and how it reduces the cognitive load put on a student, enabling them to assimilate large amounts of information.

The approach taken gave me the opportunity to learn several technologies which I had little to no experience with, previous to commencing this project. Being a proficient Java developer, it was a safe choice for developing the back-end in a programming language that I knew I could be productive with since day one. On the other hand, learning JavaScript to implement the front-end allowed me to build an application that features a state-of-the-art interface and uses powerful libraries such as D3 and Google Charts. Also, the extensive testing procedures undertaken have enabled the application to be effectively polished. Now that I have more experience involving JavaScript, I would strongly contemplate an implementation featuring a Node.js server-side, if I were to carry out this project again.

In conclusion, the findings obtained from the user study demonstrate that I have tackled the presented problem with the right approach and have built an application that has true potential.

References

[1] Enrique Orduña-Malea, Juan Manuel Ayllón, Alberto Martín-Martín, and Emilio Delgado López-Cózar. About the size of google scholar: playing the numbers. *arXiv preprint arXiv:1407.6239*, 2014.

[2] Susanne Mikki. Google scholar compared to web of science. a literature review. *Nordic Journal of Information Literacy in Higher Education*, 1(1), 2009.

[3] Dean Giustini and Maged N Kamel Boulos. Google scholar is not enough to be used alone for systematic reviews. *Online journal of public health informatics*, 5(2):214, 2013.

[4] Jillian R Griffiths and Peter Brophy. Student searching behavior and the web: Use of academic resources and google. 2005.

[5] Mary Shultz. Comparing test searches in pubmed and google scholar. *Journal of the Medical Library Association: JMLA*, 95(4):442, 2007.

[6] Thomson Reuters. The citation connection, 2014.

[7] Christy Hightower and Christy Caldwell. Shifting sands: science researchers on google scholar, web of science, and pubmed, with implications for library collections budgets. *Issues in Science and Technology Librarianship*, (63):4, 2010.

[8] Thomson Reuters. Web of science: Preliminary release notes v5.13, 2014.

[9] David P Tegarden. Business information visualization. *Communications of the AIS*, 1(1es):4, 1999.

[10] Kristin A Cook and James J Thomas. Illuminating the path: The research and development agenda for visual analytics. Technical report, Pacific Northwest National Laboratory (PNNL), Richland, WA (US), 2005.

[11] Mary Hegarty. Dynamic visualizations and learning: Getting to the difficult questions. *Learning and Instruction*, 14(3):343–351, 2004.

[12] Jean-Daniel Fekete, Georges Grinstein, and Catherine Plaisant. Ieee infovis 2004 contest, the history of infovis. *Retrieved from www.cs.umd.edu/hcil/iv04contest*, 2004.

[13] Elsevier. Elseviers scopus content, 2015.

[14] Judy F Burnham. Scopus database: a review. *Biomedical digital libraries*, 3(1):1, 2006.

[15] Rose Kalki. Google scholar citation visualisation tool, 2011.

[16] Peter Bergström and Darren C Atkinson. Augmenting the exploration of digital libraries with web-based visualizations. In *Digital Information Management, 2009. ICDIM 2009. Fourth International Conference on*, pages 1–7. IEEE, 2009.

[17] Niklas Elmqvist and Philippas Tsigas. Citewiz: a tool for the visualization of scientific citation networks. *Information Visualization*, 6(3):215–232, 2007.

[18] Allen Institute. Semantic scholar, 2015.

[19] Nick Nikiforakis, Luca Invernizzi, Alexandros Kapravelos, Steven Van Acker, Wouter Joosen, Christopher Kruegel, Frank Piessens, and Giovanni Vigna. You are what you include: large-scale evaluation of remote javascript inclusions. In *Proceedings of the 2012 ACM conference on Computer and communications security*, pages 736–747. ACM, 2012.

[20] Nilesh Jain, Priyanka Mangal, and Deepak Mehta. Angularjs: A modern mvc framework in javascript. *Journal of Global Research in Computer Science*, 5(12):17–23, 2015.

[21] Dragos-Paul Pop and Adam Altar. Designing an mvc model for rapid web application development. *Procedia Engineering*, 69:1172–1179, 2014.

Appendix A

Original Project Brief

1 Problem

Scholarly search engines such as Google Scholar, Microsoft Academic Search or CiteSeerX are commonly introduced to university students as a resourceful way of getting hold of academic papers and literature for homework and project research. Google Scholar is the largest and most updated bibliographic library on the internet, and does a better job than any of its other competitors, which are mostly defunct now. These tools can be extremely useful when used correctly, however, students that do not have much previous experience searching for academic publications might struggle to understand how to use the features provided by these search engines in order to find relevant articles. Therefore, we believe that students could benefit from an improved user interface which makes use of information visualization techniques as an alternative way of presenting the data.

2 Goals

Our objective is to visualise the metadata of scholarly literature provided by web search engines such as Google Scholar in a more user friendly and efficient way. To do so, we have to understand what kind of metadata can be extracted from such search engines. This might include basic information such as publication title, number of citations, PDF link, number of results, etc. Based on this set of data, the next step is to design a new graphical presentation of the results. We hope to use data visualization tools in order to illustrate the relationship between different papers as citation graphs. With the novel graphical presentation in the form of a web application, we aim to maximise the level of usability for navigating between scholarly articles.

3 Scope

Initially, the project will focus on the creation of appropriate visualizations and graphics that can be useful for presenting the data offered by Google Scholar and its academic papers. A prototype will be built as a web app and developed mostly in JavaScript. Depending on the metadata that is currently available from Google Scholar, the application will be adapted to either work using real data or self-generated data sets, and will be tested with a limited number of papers.

Appendix B

Scopus Search Example

http://api.elsevier.com/content/search/scopus?query=citewiz&sort=relevancy&count=1

```
"prism:url": "http://api.elsevier.com/content/abstract/scopus_id
   /36749009665",
"dc:identifier": "SCOPUS_ID:36749009665",
"eid": "2-s2.0-36749009665",
"dc:title": "CiteWiz: A tool for the visualization of scientific
   citation networks",
"dc:creator": "Elmqvist N.",
"prism:publicationName": "Information Visualization",
"prism:issn": "14738716",
"prism:eIssn": "14738724",
"prism:volume": "6",
"prism:issueIdentifier": "3",
"prism:pageRange": "215-232",
"prism:coverDate": "2007-12-01",
"prism:coverDisplayDate": "December 2007",
"prism:doi": "10.1057/palgrave.ivs.9500156",
"pii": "9500156",
"citedby-count": "13",
"affiliation": [
   {
     "@_fa": "true",
     "affilname": "Chalmers University of Technology",
```

```
      "affiliation-city": "Gteborg",
      "affiliation-country": "Sweden"
    },
    {
      "@_fa": "true",
      "affilname": "Goteborgs Universitet",
      "affiliation-city": "Goteborg",
      "affiliation-country": "Sweden"
    },
    {
      "@_fa": "true",
      "affilname": "Universite Paris-Sud XI",
      "affiliation-city": "Orsay",
      "affiliation-country": "France"
    }
    ],
"prism:aggregationType": "Journal",
"subtype": "ar",
"subtypeDescription": "Article",
"source-id": "4700151613"
```

Appendix C

ScholarWiz Views

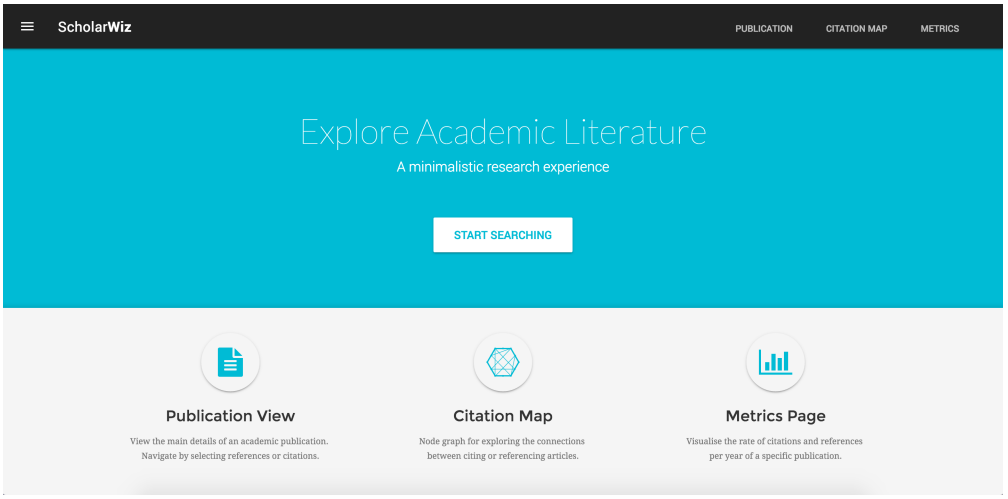


FIGURE C.1: Landing page

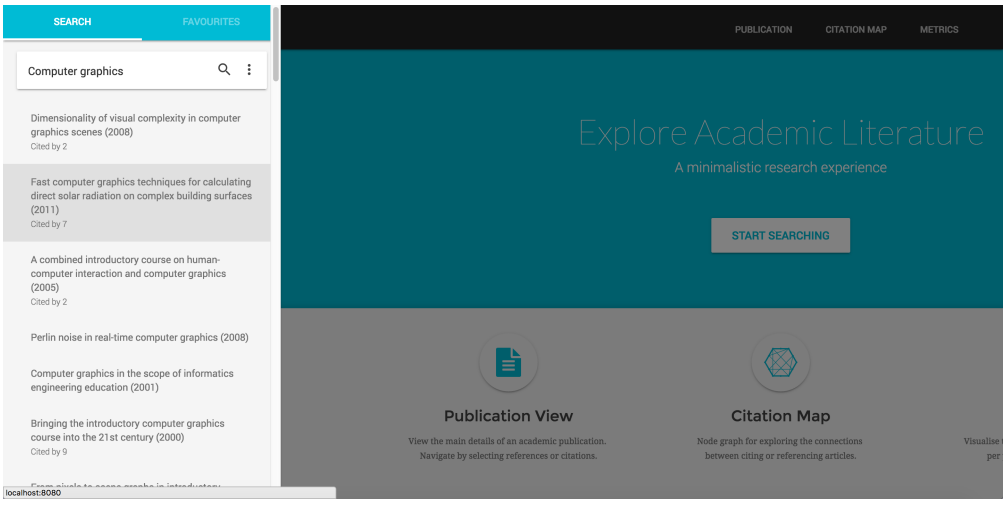


FIGURE C.2: Landing page with Search drawer

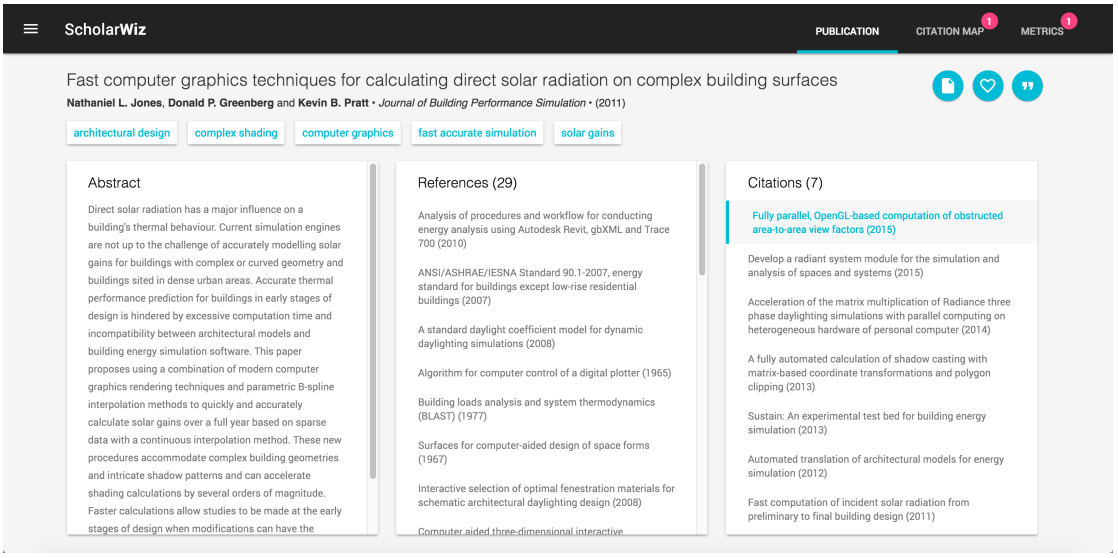


FIGURE C.3: Publication view

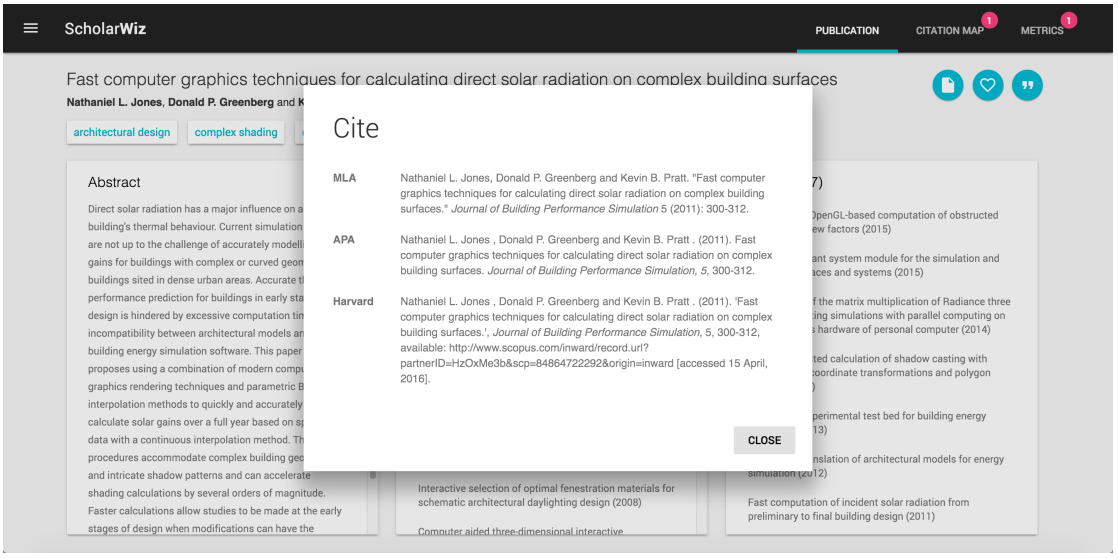


FIGURE C.4: Publication view with Cite feature

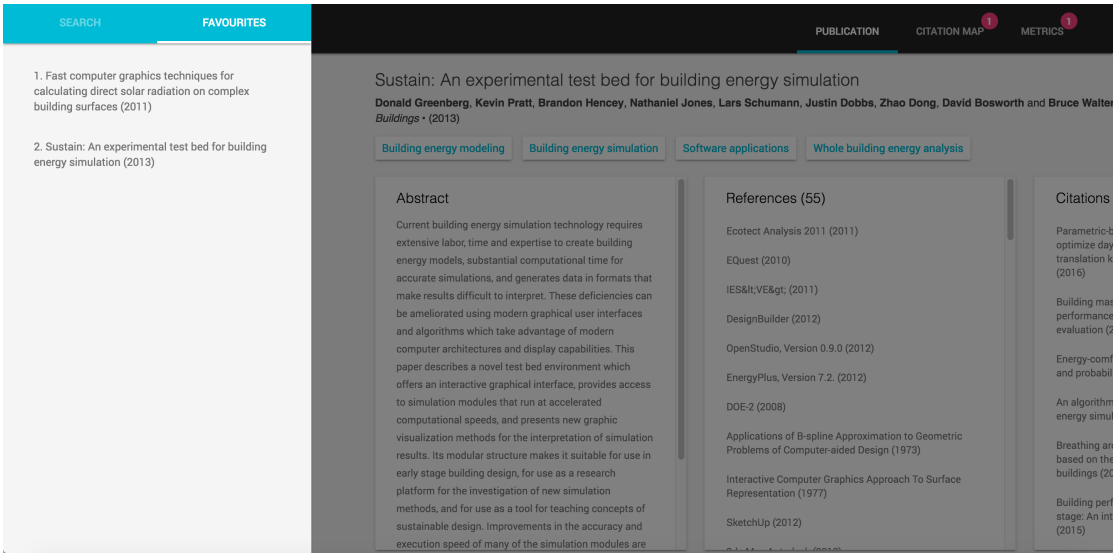


FIGURE C.5: Publication view with Saved drawer

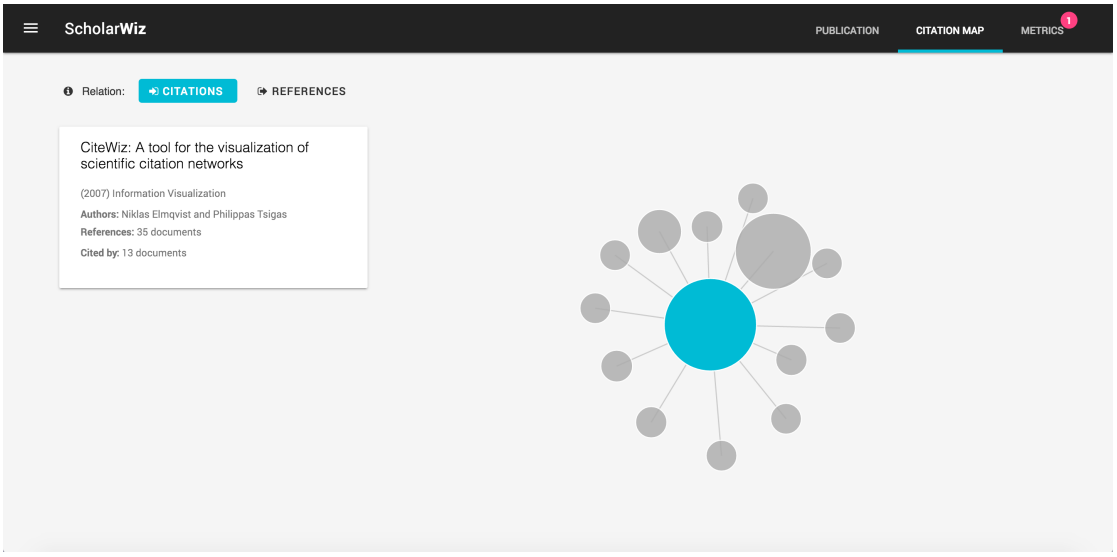


FIGURE C.6: Citation Map displaying cited-by documents

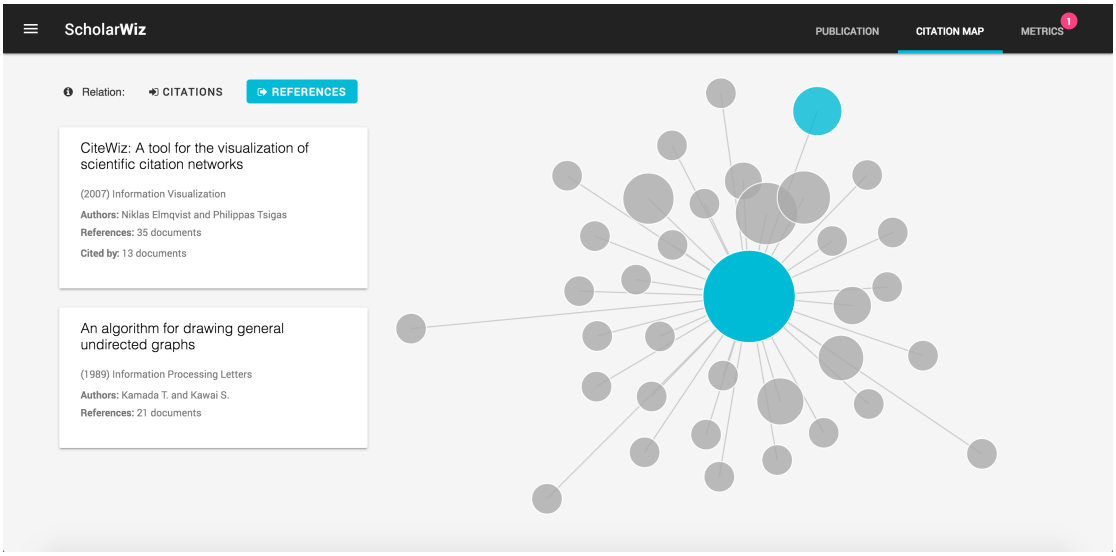


FIGURE C.7: Citation Map displaying referenced papers

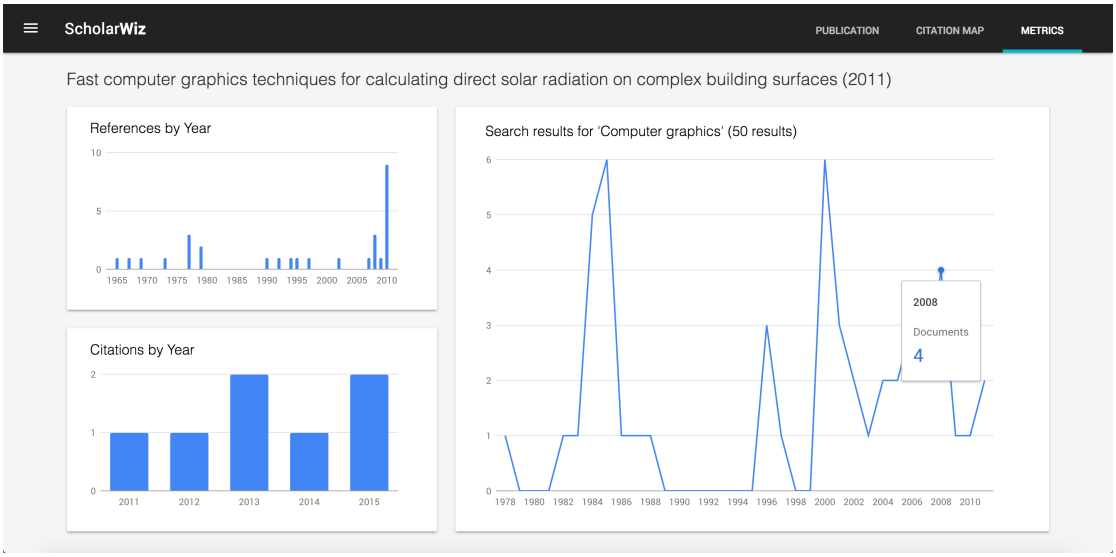


FIGURE C.8: Metrics view

Appendix D

Testing

D.1 HTTP Benchmark

Number of Citations	Average Time (ms)	
	Non-Parallel	**Parallel Stream**
2	4052,8	3745,4
7	9647,4	5381,0
13	19013,0	8155,2
19	29305,0	11815,8
25	37212,6	13956,8

Table D.1: Average times required to retrieve a document depending on the number of citations

D.2 Manual Testing

Test Case	**Outcome**
Provide search functionality so the user can search for academic publications	As expected
Provide search functionality so the user can search for authors	Not implemented
Allow the user to select a paper from the search results and display its metadata	As expected
Provide saving functionality so the user can quickly access certain papers that he/she found interesting	As expected
Allow the user to click on a referenced or citing paper to directly display the metadata from that paper	As expected
Provide a graph of the citations of the selected paper per year	As expected
Provide a graph of the references of the selected paper per year	As expected
Provide a visualisation of the citation network of the selected paper	As expected
Allow the user to select a node from the visualisation in order to view the metadata from that paper	As expected
Display metadata when a user hovers over a node in the citation network	As expected
Provide metrics and data visualisations involving the search results	As expected
Display a list of search results when a user clicks on authors or keywords	As expected

Table D.3: Audit of personal skills

Appendix E

User Study

E.1 Survey Questionnaire

E.1.1 Consent Information

Participants were shown this page prior to commencing usability testing.

Appendix (iii) Consent Form template

Consent Information

Ethics reference number: **ERGO/**FPSE/18131	Version: 3	Date: 2016-01-04
Study Title: Novel Visualization Models for Exploring Academic Research Papers		
Investigator: Kwong Chi Tam		

Participants are asked to indicate their agreement to the following statements.

I have read and understood the Participant Information and have had the opportunity to ask questions about the study.

I agree to take part in this study.

I understand my participation is voluntary and I may withdraw at any time and for any reason.

FIGURE E.1: Consent information

E.1.2 Instructions

Participants were shown this page prior to commencing usability testing.

Novel Visualization Models for Exploring Academic Research Papers

This project is inspired by the daily challenges that we are confronted with as undergraduate students when searching and filtering relevant scholarly literature. The objective is to build an innovative user interface that clearly displays core metadata from academic papers and features user interaction techniques as well as useful visualisations which enable pleasant search and discovery experiences. Once the proposed system has been implemented, a user experience evaluation will be carried out in order to assess the success of the web application.

Notes

- You may encounter some articles that are not available in the external academic database that we are using (Scopus by Elsevier).
- Speed is limited by the number of external API calls that the interface needs to make in order to obtain data, so it might take longer to load specific articles.

Optional Guidelines

1. **Search** for scholarly articles on the topic you are researching on, for example “computer vision".
2. **Select** a document from the list of results (preferably one that has been cited).
3. Add this publication to your **favourites** and **cite** it by copying the formatted citation.
4. Check if the publication appears in your list of **favourites**.
5. Navigate to one of the **citing articles** from the list of citations.
6. Click on one of the **keywords** and select a new new scholarly article.
7. Switch to the **Citation Map** page of the current document.
8. Click on one of the of the **citing articles** from the node graph.
9. Switch to the **Metrics** page of the current document.
10. Can you tell what is the year with the most **referenced articles**?

FIGURE E.2: Testing instructions

E.1.3 Scholarly Search Engines

These questions examine participants' previous experiences with existing scholarly search engines such as Google Scholar.

Appendix (ii) Questionnaire

1. **Select all the scholarly search engines that you have previously heard about.**

   - ○ Google Scholar
   - ○ Microsoft Academic Search
   - ○ CiteSeerX
   - ○ DelphiS (University of Southampton)
   - ○ Scopus (Elsevier)
   - ○ Web of Science (Thomson Reuters)
   - ○ Others

2. **Select all the scholarly search engines that you have used.**

   - ○ Google Scholar
   - ○ Microsoft Academic Search
   - ○ CiteSeerX
   - ○ DelphiS (University of Southampton)
   - ○ Scopus (Elsevier)
   - ○ Web of Science (Thomson Reuters)
   - ○ Others

3. **Rate your level of experience with your favourite / most used scholarly search engine.**
   (e.g. 1 – I used it once for an essay ... 5 – I use advanced search to find relevant publications)

   1 2 3 4 5

4. **Rate the learnability of the user interface of your favourite / most used scholarly search engine.**
   (e.g. 1 – I struggle to find PDF links ... , 5 – Very intuitive to use, even during the first time)

   1 2 3 4 5

5. **Rate the ease of finding relevant publications for your research with your favourite / most used scholarly search engine.**
   (e.g. 1 – I spend a lot of time before finding a paper that is worth referencing ... 5 – The first search results tend to be relevant for my research)

   1 2 3 4 5

FIGURE E.3: Questions involving existing scholarly search engines

E.1.4 ScholarWiz

These questions examine participants' experience with the ScholarWiz system.

6. Rate the learnability of the user interface and the ease of navigation in the proposed system.
(e.g. 1 – I struggle to find favourite papers… , 5 – Very intuitive to use, even the first time)

1 2 3 4 5

7. Rate the quantity and quality of features offered by the proposed system.
(e.g. 1 – Lacking essential tools …5 – Provides all the tools that I would possibly use)

1 2 3 4 5

8. Rate the quantity of information presented for a paper by the proposed system.
(e.g. 1 – Lacking essential information … , 5 – Provides all the information that I would possibly need)

1 2 3 4 5

9. Mention a system feature that you particularly liked and briefly explain why.
(e.g. Search bar, list of citations and references, citation map, paper details…)

10. Mention a system feature that requires improvements and briefly explain why.
(e.g. Search bar, list of citations and references, citation map, paper details…)

11. Mention a potential feature or features that you would like to be included in the future.

12. Would you use this tool if it became a real service in the future?

No Only after improvements Yes

FIGURE E.4: Questions involving ScholarWiz

E.2 Results

Existing scholarly search engines					
Select all the scholarly search engines that you have previously heard about					
Google Scholar	DelphiS	Web of Science	Scopus	MAS	CiteSeerX
10	9	3	3	1	1
Select all the scholarly search engines that you have used					
Google Scholar	DelphiS	Web of Science	Scopus	MAS	CiteSeerX
10	7	2	1	0	0

Table E.1: Questions regarding existing scholarly search engines

Favourite scholarly search engine				
Rate your level of experience with your favourite / most used scholarly search engine				
1 (Poor)	2 (Lacking)	3 (Average)	4 (Good)	5 (Excellent)
1	2	3	3	1
Rate the learnability of the user interface of your favourite / most used scholarly search engine				
1 (Poor)	2 (Lacking)	3 (Average)	4 (Good)	5 (Excellent)
0	3	3	4	0
Rate the ease of finding relevant publications for your research with your favourite / most used scholarly search engine				
1 (Poor)	2 (Lacking)	3 (Average)	4 (Good)	5 (Excellent)
0	3	5	1	11

Table E.2: Questions regarding particpant's favourite scholarly search engine

ScholarWiz				
Rate the learnability of the user interface and the ease of navigation in the proposed system				
1 (Poor)	2 (Lacking)	3 (Average)	4 (Good)	5 (Excellent)
0	0	0	6	4
Rate the quantity and quality of features offered by the proposed system.				
1 (Poor)	2 (Lacking)	3 (Average)	4 (Good)	5 (Excellent)
0	0	0	4	6
Rate the quantity of information presented for a paper by the proposed system				
1 (Poor)	2 (Lacking)	3 (Average)	4 (Good)	5 (Excellent)
0	0	0	4	6
Would you use this tool if it became a real service in the future				
No	Only after improvements			Yes
0	1			9

TABLE E.3: Questions regarding the ScholarWiz system

Written feedback
Mention a system feature that you particularly liked and briefly explain why
Fluidity, references / citations GUI.
Citation Map.
The graphical display of references, and of other articles which reference it.
The citation map makes it very easy to understand how many times and precisely when something was cited. The whole system is very easy to use and navigate across.

The citation map, the colour contrast made it very easy to use. Being able to favourite plus seeing who else cited it and navigate directly.
List of citations and references as it lets you easily find out more potentially relevant papers including historic and newers ones, which is very helpful.
Abstract, citations and references on a single self-contained page
Citation map for exploring further. Search is simple, but provides good amount of information.
Citation map. I enjoy dragging the nodes around. The 'cite' button. I like how easy it is to get the citation.
Citation Map. Made the act of looking for references/citations of a lot less boring, as scrolling through a list would be. Size and distance alluded to relevance in an intuitive and easy-to-process way.

Mention a system feature that requires improvements and briefly explain why
Missing lots of papers, system should inform the user whether a paper is available or not.
The link only takes the user to the article website rather than the pdf - although this would need much more data to work.
List of citations and references stylings could be improved by adding more options for the user to pick from. Searching speed could potentially be improved.
Citation map could use some explanation as to what it means.
Citation map would benefit from indications of what each node is (maybe by keywords to avoid cluttering).
When you click on a keyword, the search results from the previous search should be immediately removed. Being able to select from more referencing styles for the citation.
Misleading "References per year" graph title. Someties nodes get hidden underneath others. Maybe include more citation formats.

Mention a potential feature or features that you would like to be included in the future
Downloads
BibTex code for cite feature.
Link to pdf.
Tell the papers they have already seen or highlight them for later use. Ability to save session and search on the basis of authors.
Maybe click on author and see more papers by them.
Allow easy navigation of search history (selected/viewed papers). Author search would be useful. Allowx exporting citations (e.g. Mendeley or BibTex format).
Personalised dashboard. PDF links.

TABLE E.5: Written feedback regarding the ScholarWiz system

E.3 Graphs

E.3.1 Existing scholarly search engines

FIGURE E.5: Select all the scholarly search engines that you have previously heard about

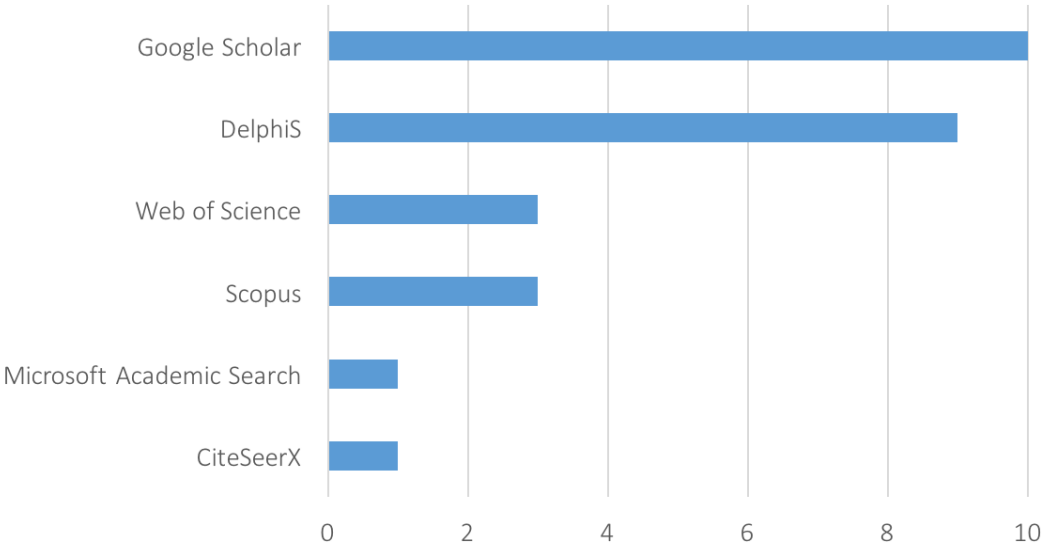


FIGURE E.6: Select all the scholarly search engines that you have used

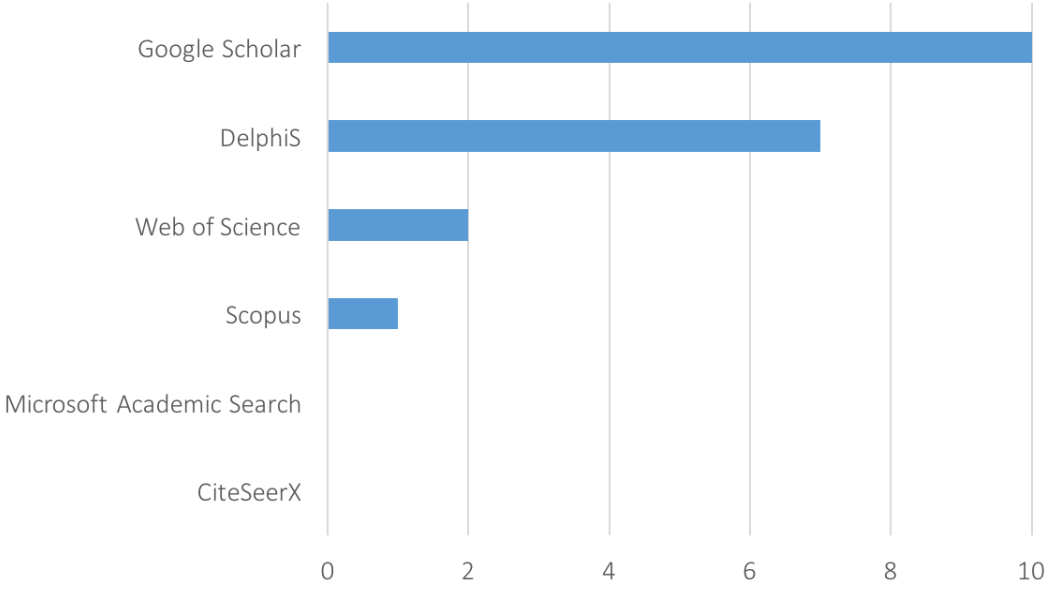


FIGURE E.7: Rate your level of experience with your favourite / most used scholarly search engine

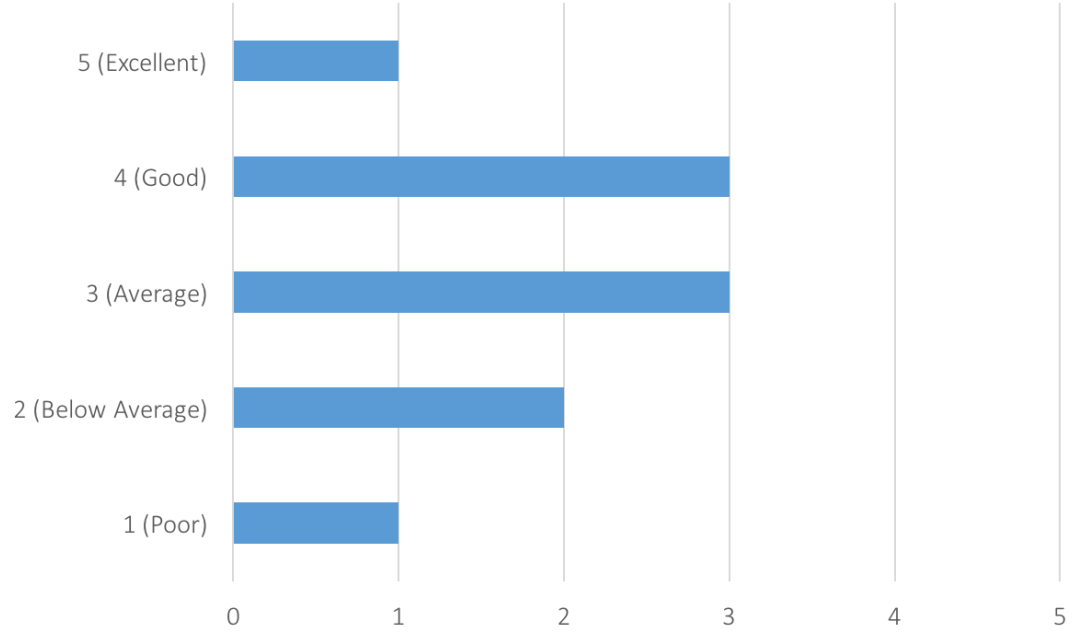


FIGURE E.8: Rate the learnability of the user interface of your favourite / most used scholarly search engine

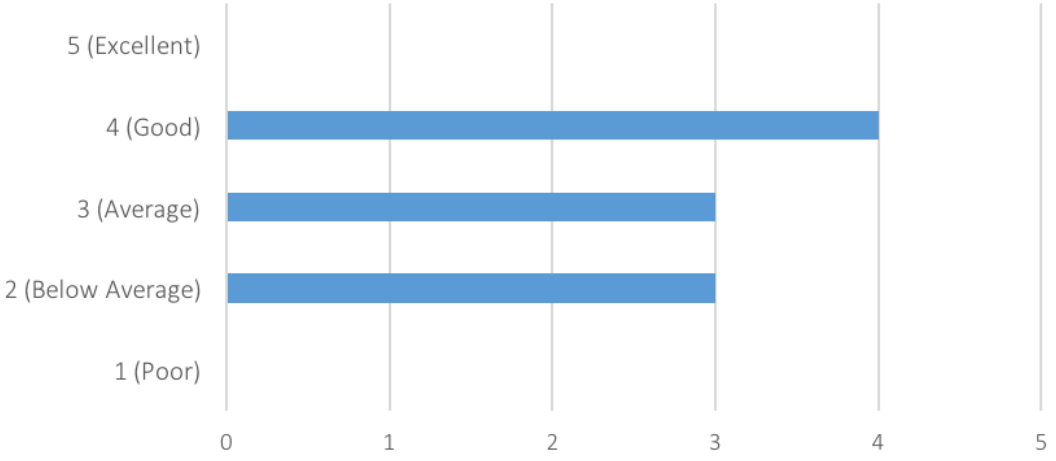
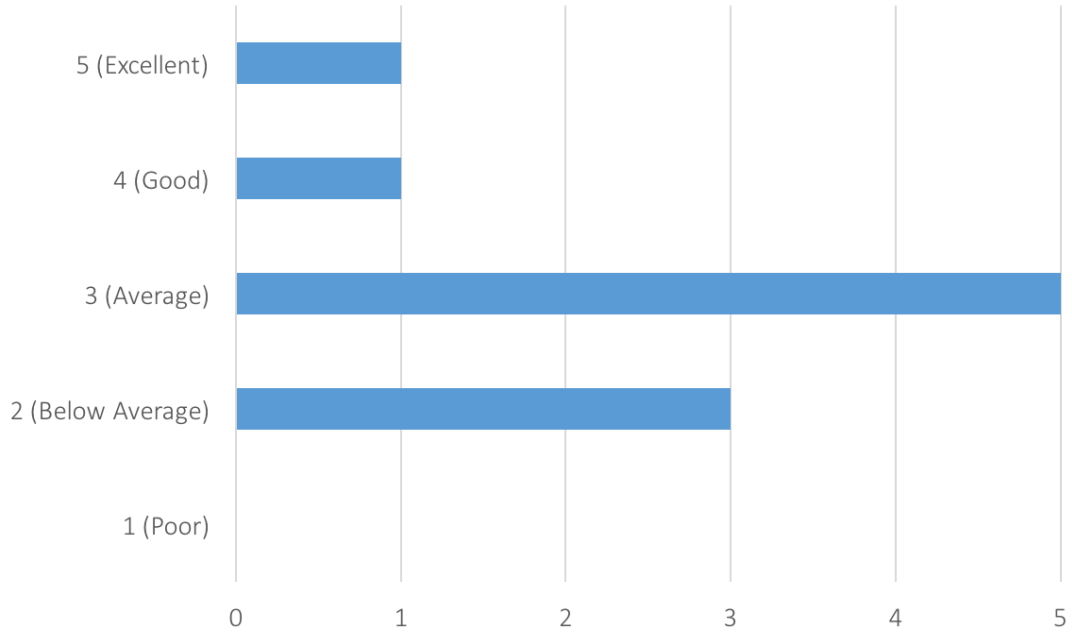


FIGURE E.9: Rate the ease of finding relevant publications for your research with your favourite / most used scholarly search engine



E.3.2 ScholarWiz

FIGURE E.10: Rate the learnability of the user interface and the ease of navigation in the proposed system

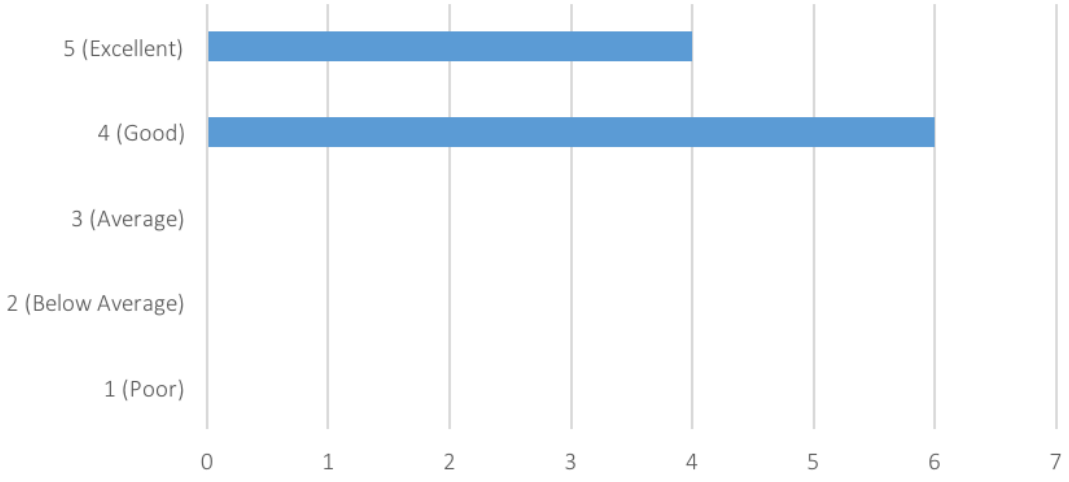


FIGURE E.11: Rate the quantity and quality of features offered by the proposed system

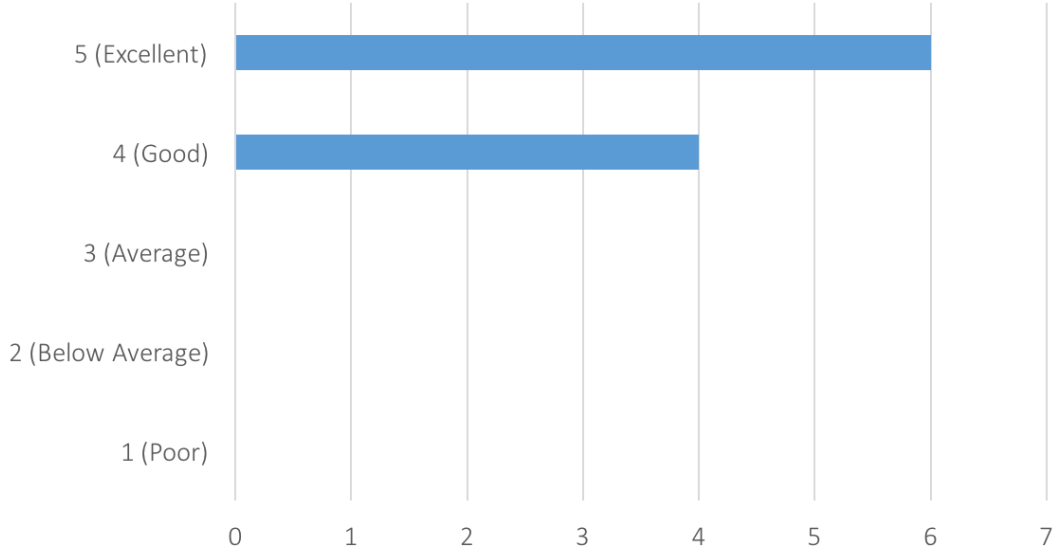
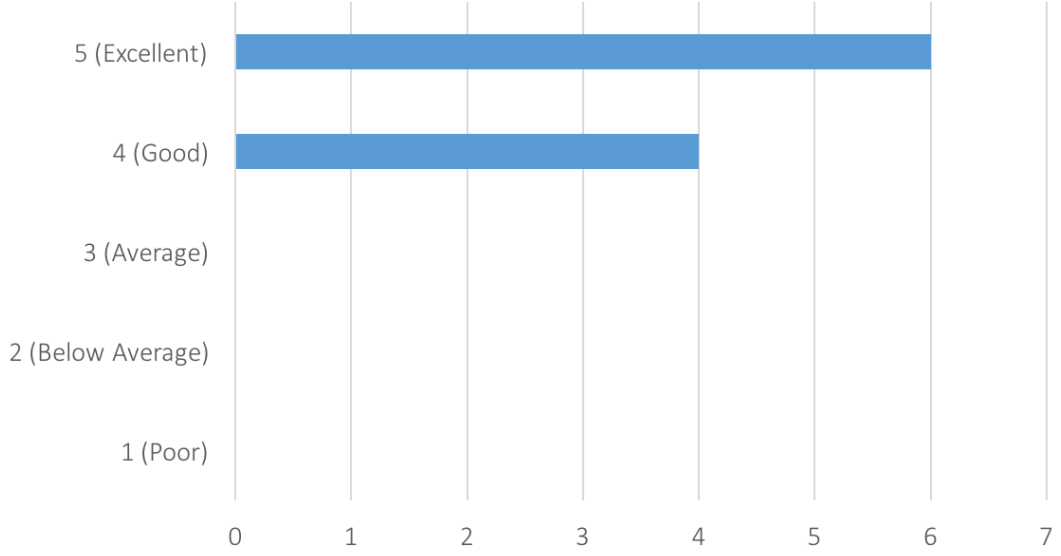


FIGURE E.12: Rate the quantity of information presented for a paper by the proposed system



Appendix F

Project Management

F.1 Skills Audit

Skill	**Rating (1-5)**
Technical Skills	
Java	5
JavaScript	1
HTML and CSS	3
REST and AJAX	1
Web Frameworks	1
LaTeX	1
Communication and Analysis Skills	
Writing	3
Researching	3

Table F.1: Audit of personal skills

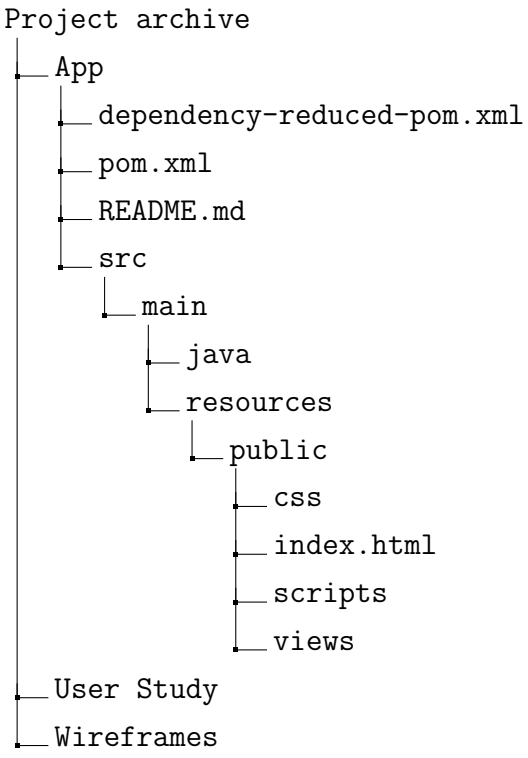
F.2 Risk Assessment

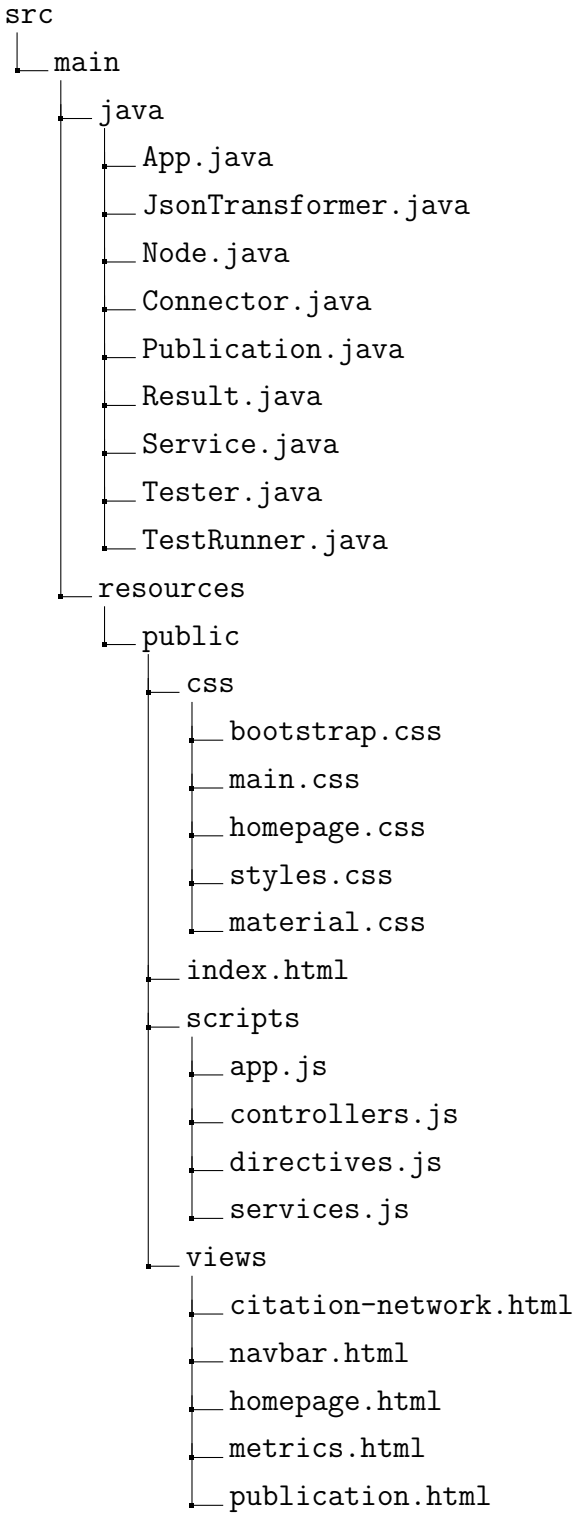
Risk	**P (1-5)**	**S (1-5)**	**P * S**	**Mitigation Strategy**
Conflict with other deadlines	4	3	12	Keep up-to-date with all the assignments that have been set and work on them progressively. Adapt Gantt chart to have less workload on deadline weeks.
Difficulty learning how to use technologies	4	3	12	Currently developing a cloud application that uses AngularJS for the front-end.
Lack of official API or project violates terms and conditions	4	3	12	Contact Scopus regarding the use of their APIs. Found a small academic database saved in an XML file.
Loss of data by theft or failure	2	5	10	Use a private Git repository to store project and make weekly backups on an external hard drive.
Inaccurate time management	3	3	9	Confirm workload with supervisor and assign longer times for the harder tasks in the Gantt chart.
Emergency unavailability or illness	1	5	5	Try to be slightly ahead of schedule whenever it is possible and assign longer times for the harder tasks in the Gantt chart.
Supervisor unavailable	1	4	4	Arrange meetings on a weekly basis for constant communication and feedback. Use emails in case supervisor is unavailable

Table F.2: Risk assessment

Appendix G

Design Archive




src
main
java
App.java
JsonTransformer.java
Node.java
Connector.java
Publication.java
Result.java
Service.java
Tester.java
TestRunner.java
resources
public
css
bootstrap.css
main.css
homepage.css
styles.css
material.css
index.html
scripts
app.js
controllers.js
directives.js
services.js
views
citation-network.html
navbar.html
homepage.html
metrics.html
publication.html