



university of  
 groningen

university college  
 groningen

# **Rate Your Search**

**Capturing Architectural Information from Search  
 Engine results**

## **Bachelor Thesis**

Tiffany Meijer

Faculty of Science and Engineering  
 University of Groningen

7 February 2021

**Supervisors:**

Dr. Mohamed Soliman

Dr. Paris Avgeriou

# Abstract

Because of the rapid growing of the amount of information and alternatives, searching and finding software architecture information is difficult for software engineers. To ease this burden on the software engineers, others have tried to re-rank the search results in Stack Overflow by classifying, filtering or applying machine learning algorithms. In addition, others have created architecture knowledge repositories. Furthermore, others tried to eliminate the use of search engines by creating a plugin for IDEs. However, in this research we would like to support in finding the best solution by conducting an experiment to determine the best ways to capture this information. This experiment may be the foundation of the future of research in software architecture knowledge. However, to conduct this experiment we are required to acquire some information.

As a result, this thesis presents an extension for browsers named Rate Your Search that can capture this information from Google when a programmer searches for software architectural information. This extension extracts the search results and user input as data and is developed using ordinary web-page development languages (HTML, CSS, and JavaScript). This plugin will help answer the research question: "How to capture search results for software architecture information?". Thus, while others have worked with the search engine in Stack Overflow, IDEs or static repositories, we introduce a way to use one of the most used search engines, Google, and to utilize user input.

Rate Your Search is merely satisfactory when it can indeed support the experiment by gathering all the correct information and making it user friendly. In our evaluation, we demonstrate the different tests and their results to illustrate it captures all the necessary information and is straightforward for users.

# Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
<b>2</b>	<b>Background</b>	<b>4</b>
2.1	Software Architecture . . . . .	4
2.2	Search Engines . . . . .	4
2.3	Plugins . . . . .	5
<b>3</b>	<b>Related Work</b>	<b>6</b>
3.1	Search Tasks . . . . .	6
3.2	Architecture Knowledge Repositories . . . . .	7
3.3	Search Engines . . . . .	7
3.4	User Ranking . . . . .	9
3.5	Plugins . . . . .	9
3.6	Search Process . . . . .	10
<b>4</b>	<b>Architecture</b>	<b>11</b>
4.1	Database . . . . .	12
4.2	Chrome Extensions . . . . .	12
4.3	RESTful API . . . . .	16
<b>5</b>	<b>The Implementation</b>	<b>17</b>
5.1	Database . . . . .	17
5.2	Chrome Extension . . . . .	18
5.3	RESTful API . . . . .	19
5.4	Deployment . . . . .	22
<b>6</b>	<b>The Evaluation</b>	<b>24</b>
6.1	Methods . . . . .	24
6.2	Usability . . . . .	25
6.2.1	Results . . . . .	27
6.2.2	Evaluation . . . . .	29
6.3	Functionality . . . . .	30

<i>CONTENTS</i>	iii
6.3.1 Experiment . . . . .	31
6.3.2 Results . . . . .	32
6.3.3 Evaluation . . . . .	33
<b>7 Conclusion</b>	<b>36</b>
<b>8 Future Work</b>	<b>37</b>
<b>A User's guide</b>	<b>39</b>
<b>B Programmer's guide</b>	<b>47</b>
B.1 Database . . . . .	47
B.2 Rest API . . . . .	47
B.3 Web extension . . . . .	49

# 1

## Introduction

Before and during the process of software engineering, programmers need to perform architectural tasks such as deciding which technologies are suitable, comparing two technologies, search for possible architectural principles, patterns, components, and much more. As a result, programmers need to research software architecture information [1, 2], which can be interpreted as the architectural elements and the organization of them to constitute the design and development of any complex software system [3]. However, finding this software architecture information is difficult [1] and is accepted as one of the critical issues in software engineering [3] due to the exponential growth of information. Thus, we will need to research what this information consists of, and the directions that have already been explored to improve on this. Based on this research, our desire is to explore software architectural knowledge further and support the software engineers searching on the web.

Accordingly, an experiment is to be conducted to determine which resources are most effective, which websites provide better information than others, and how satisfactory the search engines themselves are. This experiment is conducted as such: multiple practitioners will be asked to perform architectural tasks comparable to the ones mentioned above. Before they start searching, they need to classify their search as one of the architectural tasks given. Once that is done, we want to keep track of their search queries executed in Google and the search results (URLs) given. In addition, during the experiment they can select a relevance score ranging from no relevance to high relevance and the knowledge types in the web page for each of their search results. We also want to monitor which search results they click on.

In order to conduct this experiment, it calls for a tool which captures the users' search

process (the task, their queries with results, the relevance, knowledge types and clicks) in finding architectural information. The research of this paper intends to produce this tool named Rate Your Search. As a result, Rate Your Search will accumulate all the captured data in a database suited to reach our goal.

In fact, this research will try to answer the question: "How to capture search results for software architecture information?". To answer this, the research will expand on the architecture of Rate Your Search and the information in the database to determine the effectiveness of existing search engines, and information on plugins. This results in the following sub-questions:

- What is the architecture for a system that captures search results for architecture information?
- What languages and frameworks should we use?
- Can the plugin be utilized universally across all browsers and search engines?
- How can we assess the effectiveness of existing search engines to search for architecture information?
- Which architectural information is useful for each architectural task?

Consequently, we propose to develop a plugin, also called an extension, for web browsers which captures the keywords of the search query, the top 10 search results (URLs), the relevance of the search result and the architectural task the user is trying to perform. The keywords and search results can be extracted from the web page, whereas the user will need to interact to select an architectural task and score the relevance of a web page. Extensions are built using conventional web development technologies such as HTML, CSS, JavaScript and APIs.

Moreover, by using a MySQL database and a REST API to communicate with this database from the plugin, we can store the captured data. This data will provide the possibility to measure search relevance using Normalized Discounted Cumulative Gain (NDCG). Additionally, using user accounts, the system is able to count the number of search queries a user applied to obtain their desired information. This helps determining the quality of the search results and search engine.

The main contributions of this paper include:

- A plugin which captures the search queries and results from existing Search Engines (SE) such as Google.
- The plugin allows users to determine the relevance of the search results on the Search Engine Result Page (SERP).
- The plugin can easily be accessed on the (Chrome) browser.

Therefore, this plugin has an impact on the field of information retrieval. It can direct and enhance the future of (the research in) software architecture knowledge and support the software engineers and software architects.

**Structure of this Thesis** The structure of the remainder of this document is the following:

In Chapter 2 (Background) , the general research field is described with the recurring terms.

In Chapter 3 (Related Work), descriptions of how others tried to solve the problem are described and evaluated.

In Chapter 4 (Architecture), the architecture of the tool Rate Your Search is outlined.

In Chapter 5 (Implementation), the details of how the architecture is applied to implement the tool are presented.

In Chapter 6 (Evaluation), test cases are presented with the results. These results and the rest of the tool is evaluated.

In Chapter 7 (Conclusion), we summarize our work and present what we learned.

In Chapter 8 (Future Work), we suggest possible future work to extend or build upon this research.

In Appendix A, we introduce a user guide and a programmer's guide to the tool.

In Appendix B, we introduce programmer's guide to the tool to augment Chapter 5.

# 2

## Background

To completely understand the contents of this thesis, it is important to comprehend some key aspects.

### **2.1 Software Architecture**

Software architecture consists of the organization of architectural elements that constitute it. Having a solid architecture implies the production of a system's properties such as reliability, flexibility, etc., is facilitated. It can be the foundation and guide during the development of such a system. It determines the levels abstraction, levels of expressions, structure and behavior of the system. Therefore, without a solid architecture, there can be great consequences [3].

### **2.2 Search Engines**

Almost everyone has used a search engine as they are the modern way of finding information easily. Technically, a search engine uses algorithms to find and collect information about web pages [4]. Visually, it is a user interface with a search bar and a search button, which triggers the algorithm and then displays the search results on the search engine result page (which we will call the SERP). When we talk about search engines, we call them SEs. The results on the SE mostly consist of the URL of the page, and some information which indicates the content of the page. A well known SE is



Google. However, there are also search engines within websites such as Stack Overflow and Wikipedia which apply the algorithm within the contents of its own web pages.

## 2.3 Plugins

Plugins, in general, are software components that enhance the user experience. There are many different types of plugins, ranging from ones for media players to ones for web browsers. The main advantages of plugins lie in their ability to easily add new functionality, allow third party developers to extend the possibilities. We will profit from these advantages by creating a plugin for web browsers. Namely, one for the Chrome web browser. These are called web extensions and their characteristics allow users to customize and personalize their experience [5].

# 3

## Related Work

Software engineers use search engines to search for software architecture information [1, 2]. However, the selection of software architecture information remains complex [1], by reason of the rapidly growing number of alternatives [2]. For this reason, earlier research has been conducted, which is reviewed in this section.

### 3.1 Search Tasks

In our research, the architectural tasks will augment the research by analyzing the results per task, so that we can deduce a more comprehensive conclusion. Therefore, we look at other literature to receive more insight on how they categorize their research and how they obtain their conclusions.

In fact, Xia et. al [6] also deduced that search engines have become one of the most important tools to complete different type of software engineering tasks. Their research aimed to get a better understanding of some of the problems developers face throughout the software development process by researching what developers search for on these search engines. This is in line of what we are trying to achieve, because this research aims to comprehend the problems the search engines potentially cause.

This literature [6] identified 34 tasks which they classify in seven categories: general search, debugging and bug fixing, programming, third party code reuse, tools, database, and testing. In their study, they found that Google does not support software engineers well, since the special characters are not allowed in the search queries even though during coding a multitude of special characters are used.

Unfortunately, we cannot assess this literature based on similarities or dissimilarities, but this literature does already provide insight on the different search tasks which we could use to describe the results. Also, we can take their discussions into account such that we might not be able to fetch the special characters in the search queries.

Thus, Xia et. al [6] demonstrate a different method on how to assess the effectiveness of existing search engines which could greatly augments the experiment and other future research with the tool of this research. Nonetheless, our research will focus on software architecture information specifically.

## 3.2 Architecture Knowledge Repositories

As an attempt to work around the complexity of inquiring architecture information, multiple tools have been created.

For instance, Gorton et al. [1] have built QuABaseBD, "a repository of semantically structured knowledge for big data software systems" [1]. It introduces a new feature of classifying and comparing distributed database systems and their features. However, the creators of QuABaseBD have already made assessments of the relevance of the top 10 URL recommendations of each feature [1].

The former resembles an architecture knowledge repository where software engineers can browse through the different possibilities [2]. However, these repositories are to be manually updated. While in fact, most information is shared by software engineers through different knowledge sharing tools [2]. As a result, the repositories should accumulate their knowledge differently than manually, as this would also cause the information to be out of date.

Compared to QuABaseBD, the tool of this research will enable users to make the assessment of the relevance themselves which allows for more opinions of different software engineers. Likewise, however, the tool of this research will also capture the results in a repository. And although the tool will use a repository to capture results, it will be automatically updated with each use to keep our data up to date.

## 3.3 Search Engines

Besides architecture knowledge repositories, others have attempted to re-rank the search results on search engines. Hence, Soliman et al. [2] have "developed a new search approach to search for architecturally relevant information in Stack Overflow"[2]. The tool utilizes a new method to filter and rank the search results based on their suitability for architecture design activities. Also, it classifies the Stack Overflow posts to separate architecture-relevant and programming-related posts. Furthermore, it also classifies the posts in related sub-categories [2].

Clearly, this is a better solution than the architecture knowledge repositories. However, the limitations of this tool are that it is exclusively for Stack Overflow, and even though it is the most popular developer community [2], it does not work on search engines such as Google, whereas the proposed tool will. On the other hand, we propose to classify search queries (as different tasks) and, likewise, also recognize architecture-relevant information.

Besides examining the different approaches of searching for software architecture information, it might also be useful to look at how searching for other information in the software engineering field is conducted. This includes an approach to improve search engines on software forums by Gottipati et al. [7]. Their approach is proposed in a framework which classifies the posts as answers, relevant answers, junk and other classifications, and which includes a semantic search engine which uses the semantic tags to retrieve relevant answers in the threads [7]. Similarly, the tool of this research will also need to classify and find relevant information, as a web link in this case, though on search engines such as Google instead of on software forums. In contrast, this research will focus on software architecture information.

Similarly, Beyer et. al [8] created a classification module which applies machine learning algorithms for Stack Overflow Questions. Their aim was to automate the classification of questions of Stack Overflow questions into seven question categories. To apply the machine learning algorithms, an initial data set of 500 Stack Overflow questions was curated. These were manually classified into the seven categories, The seven categories are specified as: API change, API usage, Conceptual, Discrepancy, Learning, Errors and Review. This study, as well as the study by Xia et. al [6] described in the Search Tasks section, gives us insight into the information that might used for each architectural task. In addition, this literature demonstrates a different approach to classify each search.

Due to the exponential growth of information, it is becoming more complex for search engines to meet the user's information requirements and, therefore, provide the sought after results[9]. As a result, Indumathi et. al [9] present an approach to expand the user query and to re-rank the search results. The query expansion utilizes query processing, which denotes the evaluation of a user's query to expand the query and make it more precise. The process fetches when a user clicks on a snippet and applies that information to reflect the interest of the user on the concept and will provide a user preferred concept to expand the query. The similarity of this approach lies in the fetching of the clicks of the user, which the tool of this research will also capture. This literature, however, applies it to compute an expanded search query, whereas our research will apply it to compute the relevance score of the search results.

### 3.4 User Ranking

So far, we have not seen many studies where user input is used to either categorize the search queries or compute the relevance of a search result. In our research, however, user input signifies our results. Therefore, we will look at the study that Opoku-Mensah et. al [10] conducted. In contrast to other studies, this study, as well as ours, presents the need to include the user's relevance in the Search Engine Results Page (SERP) ranking. Their motivation, similarly, lies in improving the relevancy of rankings for a better user satisfaction of the search results. The research also mentions that the NDCG is ultimately based on the structural and content features of retrieved documents. From this, we can conclude that our research will be one of the firsts to base the NDCG score on user relevance and input. Their literature review mentioned that the overall users assessment of a search engine is a result of user's interaction with the SERP, while not explicitly asking the user for their assessment. The former is what this research implements.

Another study, which resulted in Rankbox proposed "an adaptive ranking system for mining complex relationships on the Semantic Web" [11]. In Rankbox, each user has their own ranking function to represent their specific preferences. The users can continuously adapt their preferences and strive for a user friendly experience by allowing the user to interact the system through just a few clicks. In contrast to Rankbox, the research of this paper will allow the user to interact on the already existing Search Engine Google, whereas Rankbox implements their own search engine. We can also draw inspiration from their user-friendly user interface (UI) such that we also make sure our users can access our plugin with just a few clicks. Rankbox allows users to "like" or "dislike" a result, whereas our research will use the Likert scale to rank the relevance of a search result to retrieve a more detailed result.

### 3.5 Plugins

Also, since we propose to develop a plugin, reviewing other plugins should be relevant to our research. As an example, Ponzanelli et al. [12] created a plugin for Eclipse called Seahawk. Seahawk automatically formulates queries from the changes in the code, which will generate a list of relevant results from Stack Overflow, and developers can also drag and drop code samples from Stack Overflow into their source code [12]. However, this plugin is not closely related to our research since it does not handle the search results such as the other tools which classify and re-ranks the results, but instead presents them and makes them available for use. This tool also suffers from the limitation that it solely operates with Stack Overflow and it merely construct search queries from the code context (e.g. code under editing) whereas we propose to utilize keywords from the search queries.

On the other hand, we have another plugin created by Rahman et al. [13] called

Surfclipse, "an IDE-based web search solution" [13]. The plugin executes searches on three main search engines (Google, Bing and Yahoo) and another Q&A site (e.g. Stack Overflow). The result is a representation of the search result with relevance scores based on the result of the search results from the different search engines. In addition, the plugin extracts solutions from a number of developer communities [13]. In comparison, the tool for this research will base the relevance score on user inputs, but, similar to this one, it should also work on popular search engines. However, this plugin is deployed in the IDE and will base the search on the errors presented by the IDE, whereas the proposed plugin will use keywords from the user on a web search engine as mentioned before.

### **3.6 Search Process**

The references for this literature review have been obtained by reading through Soliman's dissertation and researching other relevant papers from Soliman as he is part of the team that will conduct the experiment. Some of the references were also found using relevant references in Soliman's dissertation. Later, we used the keywords: Software Architecture, and Search Engine on IEEE Xplore.

# 4

## Architecture

In this chapter, we answer the question: "What is the architecture for a system that captures search results for architecture information?", by presenting our research into the architecture of web plugins.

Our web plugin should capture keywords, search results, the relevance of the search results and which architectural task the search query is categorized as. Consequently, the plugin needs these functional requirements to be able to answer the remaining research questions:

1. It captures keywords from Google.
2. It captures the URLs from the top 10 search results or all search results on the first SERP, if those are less than 10.
3. The user is able to decide on the relevance for each URL on the search engine but also on the website itself denoted by using a Likert scale.
4. The user is able to decide which knowledge type is included for each URL on the search engine.
5. The user is being presented the tasks which they can select and hide. The user can also change their selected task.
6. It captures whether the user has clicked on a search result.
7. The database is structured.

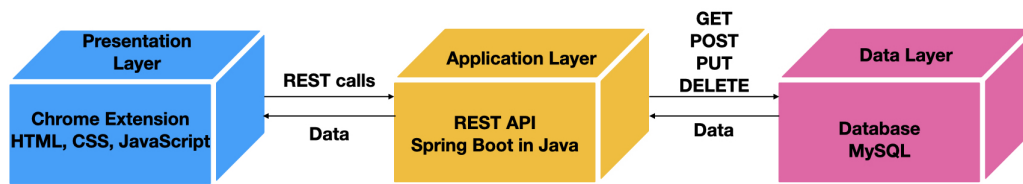


Figure 4.1: The representation of the three tier architecture we use, created by us.

8. It has secure accounts for users, so that we can compute how many queries it took for them to find their desired information.
9. It measures the search relevance with NDCG in the database.
10. It works in all countries.
11. It is secure.

In addition, as a non-functional requirement, the plugin should be user-friendly. Finally, some optional requirements, such as categorizing the URLs and allowing the user to distinguish a part of a web page as most relevant, would enhance our research even more.

## 4.1 Database

From the requirements, we recognize that the plugin will capture keywords and search results (URLs), as well as store the relevance and an architectural task. It will also contain a few user accounts. This information will be stored in a MySQL database. For this research, we will use a 3-tier database architecture. The three tiers consist of the database (Data Layer), the application (Application Layer) and the user tier (Presentation Layer). Firstly, the database defines the tables with its columns, and the relationships between each table. Secondly, the application is, in this case, the REST API which communicates with the database and displays an abstract view of the database. The REST API is intermediary between the database and the user tier. Lastly, the user tier in this case is the web extension. Our use of the 3-tier database architecture can be seen Figure 4.1.

## 4.2 Chrome Extensions

The plugin will be implemented as a browser extension on Google Chrome. Therefore, we will need to consider the architecture of extensions to decide on the architecture of



the tool and on the languages and frameworks. Essentially, a browser extension adds features to a browser. Extensions are built using familiar web development technologies such as HTML, CSS, JavaScript and other open APIs. In the end, an extension is a package of files [14]. According to [14], the architecture of an extension consists of the manifest, background scripts, UI elements, content scripts and an options page.

### Manifest

All extensions are required to have a JSON-formatted manifest file, named `manifest.json` [15]. It contains metadata which the browser employs to load up the extension. The metadata includes the files and the capabilities the extension might use. The files include the scripts mentioned, but also the styling sheets, markup files, and the icons shown in the browser. The manifest supports multiple manifest fields, but the three required fields are "manifest\_version", "name" and "version".

Additionally, there are two recommended fields called "description" and "icons". The icons represent the extension, and the developer should provide different sizes of the icon for the various uses, namely, the icon in the Chrome Web Store, in the extensions management page (`chrome://extensions`) and the favicon [16].

The browser action or the page action in the manifest places the favicon in the main Google Chrome toolbar, to the right of the address bar [17, 18]. The page action is specifically for a few pages, whereas it would make more sense to use a browser action for actions that could be applied to all pages. The decision is up to the developer.

The additional fields will not be discussed in this chapter, but they mostly describe the settings, APIs, and files used by the extension.

### Background Scripts

Background scripts, written in JavaScript, are utilized as an event handler as it contains listeners for browser events [19]. The events are browser triggers, such as navigating to a new page, removing a bookmark, or closing a tab. These events are monitored in the background script which handles accordingly. The extension is also able to trigger the events by, for example, message passing from the content script, or by calling a background function from another view of the extension.

Background pages are also utilized to maintain long-term state or perform long-term operations regardless of the lifetime of a web page or browser window. This is because a background page will stay running as long as it is performing an action and will not unload until all visible views of the extension and message ports are closed. As a side note, message passing will be discussed more alongside the content script.

### UI elements

The extensions' user interface (UI) elements allow for an expanded user experience without distracting from the browser experience [14]. The page or browser action, which are

mostly in charge of displaying the icon, mentioned in the manifest section are examples of the UI elements.

Including badges, which are purely allowed with browser actions, display a colored banner with up to four characters on top of the browser icon [20], are an example of the possible UI elements for chrome extensions.

An additional UI Feature is the popup, which is a small-scale HTML web page, displayed in a special window when the user clicks on the icon in the toolbar [20]. The popup allows for an enhanced user experience as it could be used for users to set their preferences or display other options the extension allows for. The popup can be developed using conventional web development technologies with HTML, CSS and JavaScript.

Furthermore, chrome web extensions allow for the possibility of the use of a tooltip to give a short description when hovering over the browser icon. In addition, users can invoke the extension's functionality in the omnibox, known as the chrome address bar [21], by typing in keywords designated by the developer. The extension can trigger events by the user's input in the omnibox.

Also, Chrome possesses the `chrome.contextMenus` API to add items to the Google's context menu [22], which you get by selecting a part of the web page and pressing the right mouse button. This context menu is created in the background script. Even shortcuts, also known as commands, can trigger the extension's functionality. These commands are declared in the manifest, but implemented in the background script [20].

Lastly, an extension can override and replace the History, New Tab or Bookmarks web pages with a HTML page the extension desires. This web page and the custom HTML page are specified in the manifest, but developed using, again, conventional web development technologies.

### **Content Scripts**

Content scripts contain JavaScript that access and manipulate the DOM of the web pages in the browser window. It communicates through messages with their parent extension. The advantage of content scripts is they live in an isolated world so that it enables the content script to implement functionality that should not be accessible to the web page but also allows for the content script to make changes to the web pages' DOM and JavaScript environment without conflicting with the already existing functionalities and other content scripts [23].

The content scripts can be injected declaratively using the manifest, or programmatically in the background script with the combination of some settings in the manifest. Within the grammatical injection option, there lies two other options, where you could inject the content script as a few lines of code or an entire JavaScript file. The extension specifies on which pages the content script should be injected.

However, since the content scripts live in an isolated world, they are not as easily

accessible to the rest of the extension. Therefore, the script can communicate with the background script or the popup by means of message passing. Both sides of communication listen for messages, written in JSON. There are two APIs. On the one hand, there exists an API for simple one-time requests, where you send a message and have a listener on the other side. On the other hand, there exists a more complex API for long-lived connections. They make use of a port, which the extension has to connect to. Messages can also be passed from one extension to another, and, similarly, the extension can receive messages from web pages. This message passing is where security is fragile[24].

### Options Page

Lastly, the options page allows for modification of the extension. On the options page, the user is able to customize the extensions for the user's need. This page is also developed using conventional web development. It is accessible to users by either right-clicking on the icon in the toolbar and selecting options or by navigating to details and to the options page in the extensions page of chrome [25]. Eventually, the architecture will look like in Figure 4.2.

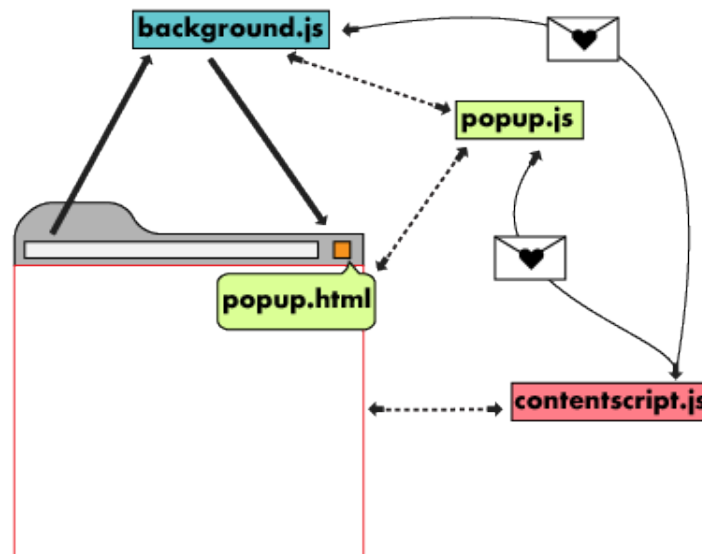


Figure 4.2: The architecture of a chrome extension from [14].

## 4.3 RESTful API

As mentioned before, we need an intermediary web app for the extension to be able to communicate with the database. For this research, we will use a RESTful API. REST is acronym for REpresentational State Transfer. RESTful APIs are also referred to as REST APIs, which we will use in the rest of the paper, or as a RESTful web service. The API uses HTTP request that the extension calls to GET, PUT, POST and DELETE data [26]. The advantage of REST is that it uses less bandwidth than Simple Object Access Protocol (SOAP), and, therefore, is more efficient.

REST APIs can be written in many different ways and languages. In this research, we utilize Spring Boot. Spring Boot is a project of Spring, a Java framework designed to make Java programming easier for everyone [27]. This is because we could embed the Tomcat server directly [28], which facilitate the deployment the REST API. It requires Java 8 or higher and support the build tools Maven and Gradle.

Every Spring Boot application written in Maven requires a POM file, named `pom.xml`, as a recipe to build the application. This pom file to the application is comparable to the manifest of the extension. It specifies some fields, but also the dependencies are declared in this file [28].

Furthermore, Spring Boot uses the Spring MVC (Model, View, Controller) framework and the annotations from that framework. Those annotations provide information to the reader about the code. On the other hand, the annotations also provide information necessary for the functionality of the application and handling of the HTTP requests.

Also, Spring has the Spring Data project, which provides data access technologies. It has multiple subprojects that are specific to a given database [29]. By adding the frameworks as dependencies in the POM, the application is able to access them. Using JPA, Java Persistence API, the Spring Data JPA project, and the Spring DATA JDBC project, we can use annotations in the code and declare properties to allow the application to access the data [30].

Namely, the `spring-boot-starter-data-jpa` dependency provides three key dependencies named Hibernate, Spring Data JPA, and Spring ORMs. They are one of the most popular JPA implementations, simplify the implementation of JPA-based repositories, and are a Core ORM support from the Spring Framework, respectively [30].

Additionally, the `spring-boot-starter-data-jdbc` dependency adds the Spring Data's JDBC repositories. JDBC automatically generates SQL for the methods in the CRUD (Create, Retrieve, Update, Delete) Repositories, and allows the developer to provide a `@Query` annotation for customized, and/or more advanced queries.

The requirements can be met when we combine these technologies and their architecture.

# 5

## The Implementation

Using the requirements and the architectural information of the technologies provided to us in the Architecture chapter, we will now elaborate on the implementation to specifically fulfill the requirements and develop our web extension Rate Your Search.

### 5.1 Database

We have designed our database in such a way to make it easy for the researchers to get the correct information to reach their goal to determine the effectiveness of resources, websites and the search engines themselves.

With the help of classifying the result in architectural tasks, the researchers can deduct which architectural task has a higher complexity to perform.

Then, with the search queries the users use in Google and the URL results, we can research elements such as which type of search queries perform best and which results are most common. Then, in combination with sessions which denote which user, from the user accounts in the database, has selected what architectural task, and the search queries within that session, we can evaluate how many queries and how long it took to reach a satisfactory result.

Additionally, we want to store the relevance scores of the results entered by the users and whether or not they clicked on the results to be able to perform a better evaluation of the quality of the results. Particularly, we can use those inputs to calculate the Normalized Discounted Cumulative Gain (NDCG) score of the search results, which, in turn, can be utilized to evaluate the effectiveness of the search engines themselves.

A more detailed description can be found in Appendix B, and a database structure design can be seen in Figure B.1, Appendix B.

## 5.2 Chrome Extension

Our Chrome extension is the aspect of the product the user will interact with. Firstly, the background script of Rate Your Search will initialize the data members for our widget to be functional. Secondly, since Rate Your Search is for a specific use, an options page is not needed.

### UI elements

Thirdly, our UI elements are provided by the popup and content script. The important factor for UI elements is their deliberate lack of distraction from the workflow. Therefore, our main priority was that the user would acquire the targets of the UI elements as fluently as possible. Therefore, we have decided to have the popup as the place where the users can login and logout, whereas the rest of the user input is done on the web pages themselves to make it more user friendly.

The popup has two states (Figure 5.1). On the one hand, we have the login screen in which the users can enter their assigned credentials. On the other hand, we just have a button for them to logout. The login button executes a REST call to our database to validate the user's input. Consequently, if their credentials are incorrect or incomplete, we notify the user with the message "Incorrect username or password". Moreover, correct credentials allow the user to utilize the rest the extension has to offer. This is because the content script checks if the user is logged in before it displays the other UI elements and before it fetches the user's actions.

In fact, the content script consist mostly of two elements: 1) the relevance and knowledge form in the Google SERP (Figure 5.2d and Figure 5.2e), where users select the relevance score of the results, and 2), the widget (Figure 5.2a-5.2c). The widget displays the task segment on Google's web pages. Whereas it also displays the relevance form the web pages from search results. The widget can be collapsed (Figure 5.2c, so that the user will not be distracted from their browser experience. Moreover, Figure 5.2a shows two segments: the task segment on top, and the relevance segment at the bottom. Firstly, the task segment allows the user to select the task and also view the task's description. Secondly, the relevance segment is solely visible on web pages from search results, and this is where the user selects or updates the relevance of a web page. In addition, to notify the user of their choices, the content scripts also displays a notification bar with their choice. One example of when this happens is when the user selects a task, which can be seen in Figure 5.2f.

### Content scripts

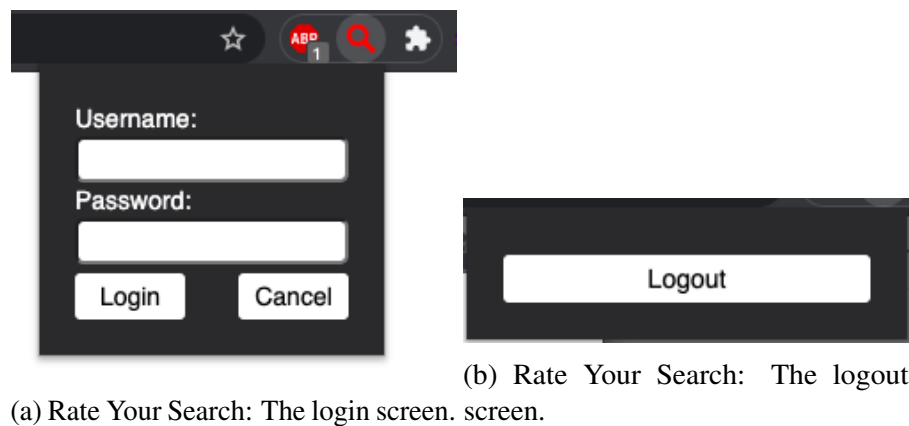


Figure 5.1: The two states of the popup screen.

Fourthly, the content script carries out a majority of the work of the chrome extension. It can access and manipulate the DOM, and, therefore, is responsible for listening to user interactions with the DOM, as well as with the widget. Admittedly, creating the UI elements mentioned above is also one of its responsibilities besides inserting them in the DOM.

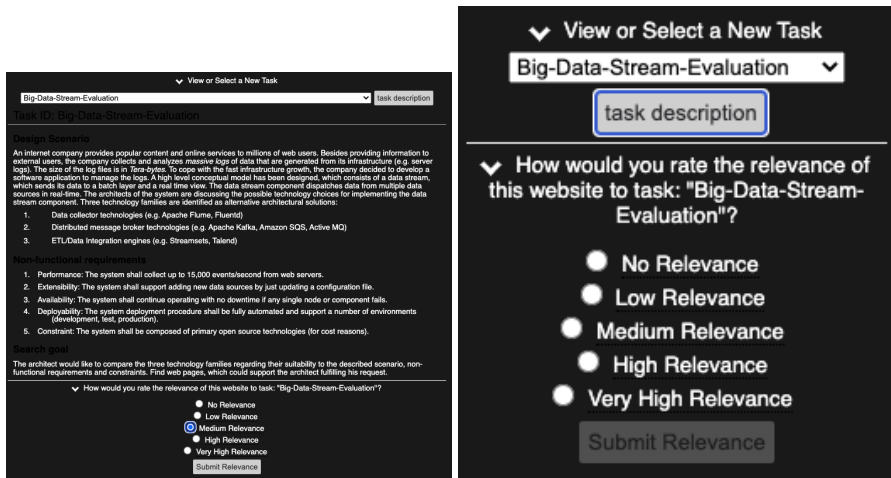
Once the content script has created these elements, it adds event handlers to the widget as well as to the anchor elements so that the URLs of those anchor elements can be stored as search results. These event handlers oftentimes result in the need for REST calls to store the fetched information. For example, once the content script has been injected and it discovers that the user executed a query, it should create REST calls that ensures the query with its search results are stored in the database. In addition to when the user selects a relevance, the content script should retrieve the correct entry from the database and update the relevance score of the record.

Furthermore, the content script is responsible for revealing the correct UI elements. As mentioned before, the widget is exclusively visible when the user is logged in. Therefore, the content script checks this condition. In addition, once the user is logged in, the content script examines if we are on Google, on a search result, or neither to see which segments of the widget should be visible.

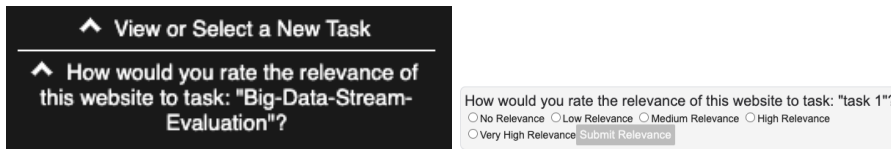
### 5.3 RESTful API

As mentioned before, the REST API is implemented using Spring Boot, which means we create it in Java with the MVC framework. The MVC framework consists of three components: Model, View and Controller.

#### MVC



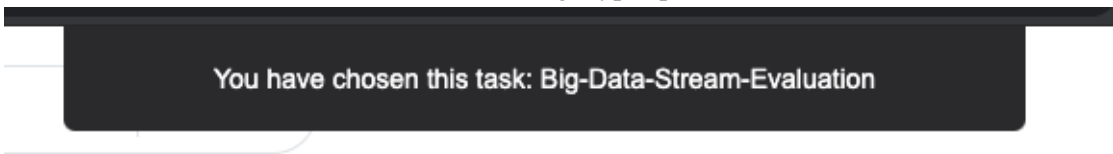
(a) Rate Your Search: Expanded wid- (b) Rate Your Search: select relevance  
get. in web page.



(c) Rate Your Search: Collapsed wid- (d) Rate Your Search: ask relevance of  
get. search results in the SERP.



(e) Rate Your Search: ask the knowledge types pf a search result in the SERP.



(f) Rate Your Search: notification bar.

Figure 5.2: The different UI elements the content script provides.



However, since we did not create an interface for our API, we did not implement any View components. Although, our Models correspond to the tables we have in our database and are the Resource Representation classes. Thus, the classes represent the table, while the attributes represent the columns in the table. Therefore, our models are: Account, Searchquery, Session, Task and Url.

Consequently, these models also all have a Resource Controller, which handles the HTTP requests and other service interactions. For example, the Controller contains a function which handles GET request for `/account/{id}`, which returns an Account with the id from the id parameter. Using the correct annotations, we can assure our Controller comprehends which mapping should be mapped to which functions and which parameter should be bound to which parameter in the mapping.

In addition, since most of the functions should return an entity of the class, we have created an Exception class for each of the Models, which is thrown when we cannot find an entity with the parameter the user passes.

### Mappings

Each of the controller handles the mapping for `table_name/{id}`, `table_name/all`, `table_name/add`. The mapping for `table_name/{id}` can be a GET, PUT or DELETE method. The `table_name/all` retrieves all records in the equivalent table, and is, therefore, a GET mapping. Last but not least, we have the POST method for all models which is handled when `table_name/add` is used.

Specifically, the Account Controller also implements a GET mapping for `accounts/username={username}`, so that it can try to retrieve an Account with the corresponding username. We consume this mapping when users try to log in.

Moreover, the Searchquery Controller requires a method for getting search queries with specific keywords that correspond to a certain session. We implemented a method for `searchqueries/searchquery={searchquery}/sessionId={sessionId}`, so that we can retrieve those search queries. This is utilized when we want to check if we already have that search query in our system and do not store the same data twice. Also, when the search query also exists, we can also retrieve the already selected relevance score of the search results and present them to the users for if they change their minds.

Furthermore, the Url Controller also has custom GET mappings supporting functionality such as checking if the web page the user is currently on is a search result. This is achieved by checking if we have an entity with the current URL in the current session with a mapping for `urls/url={url}/sessionId={sessionId}`. In addition, we often require an update of an entity of a URL, such as when the user selects a relevance, or clicks on the URL. Therefore, we need to ascertain retrieving the correct URL entity, since multiple search queries can supply some of the same search results. Thus, when retrieving those URLs, we search by URL and by the Searchquery entity which requires us to implement

a GET mapping for

`urls/url={url}/searchqueryId={searchqueryId}`. Finally, to compute the NDCG score for a search query, we need to retrieve all the URL records for that search query. Hence, there also exists a mapping for `urls/searchqueryId={searchqueryId}`.

### Security

To secure the Rest API, we acquired an SSL certificate to be able to use the HTTPS protocol. As a result, the data transferred over the network is encrypted [31].

To implement this, our REST API enforces the request to be with the HTTPS protocol by redirecting all incoming HTTP requests to the HTTPS port in the Application class and adding a configuration class.

Admittedly, during testing, we noticed that the HTTP request were being blocked by the website themselves to ensure security. Accordingly, we updated our security.

## 5.4 Deployment

In fact, one essential component of the implementation of our extension is the deployment. Namely, the users should be able to access the plugin. Whereas the researchers need to be able to access the database and REST API as well. Furthermore, the plugin needs to access the REST API.

Hence, we need to deploy our extension, REST API and database.

### Web extension

The web extension is designed for Google Chrome. Therefore, we could upload the web extension to the Chrome Web Store. However, since this extension will be used by just a few practitioners participating in the experiment, another possibility is to transfer them the folder with the scripts, style sheets and HTML files, and, most importantly, the manifest. Then, in `chrome://extensions/`, they can upload the folder including the manifest in developer mode. This automatically deploys the extension.

### REST API & Database

To deploy the REST API and the database, we want to containerize them both. Therefore, we use Docker. A docker container standardizes the development environment (frameworks, dependencies, etc.) and packages it into one container so that you can run it easily on any environment [32].

Then we need a server to store all the information and run the docker container in. For this reason, we use Amazon Elastic Compute Cloud (Amazon EC2). It is a web service that provides a secure, compute capacity in the cloud [33]. Our instance has an instance ID, and an domain name where we can send our REST calls to.

To conclude, we upload our web extension in Chrome and package our database and REST API in a Docker container, which we then deploy on our AWS EC2 instance, which we can utilize for our REST calls as well, so that they can be executed from different locations.

# 6

## The Evaluation

Now that Rate Your Search has been explained in detail, this chapter is designed to demonstrate that our approach was successful by evaluating the tool.

### 6.1 Methods

There are two components of Rate Your Search we will evaluate.

1. The usability of the tool.
2. The functionality of the tool.

The usability evaluation is done in two parts: testing out the tool at different milestones during the development to uphold usability fundamentals, but also by conducting a usability evaluation survey. This survey will be distributed after performing an experiment with practitioners who might be using the tool. We determine the questions of this survey by researching existing usability evaluation methods and surveys and selecting the questions that are most applicable to our tool.

To evaluate the functionality of the tool, we refer back to the requirements established in Chapter 4. We can analyze if the requirements are met with the provided functionality.

## 6.2 Usability

### Usability principles

Firstly, during development, we assured to maintain the best usability possible. We knew our effort should be integrated in the UI design to accomplish this. Therefore, we followed the 10 UI design principles from [34] to prove Rate Your Search is user friendly.

The 10 design principles according to [34] are:

1. *Aim at an Almost Invisible User Interface* by particularly showing essential elements and using clear language.
2. *Keep it consistent.*
3. *Be Purposeful with Page Layout* by strategically placing the elements.
4. *Use Color and Texture Strategically.*
5. *Use Familiar UI Elements: One of the Key Rules of Good UI Design.*
6. *Put the User in Control of the UI* by informing the users about their actions and have the actions be reversible.
7. *Minimize Cognitive Load: Recognition over Recall* by having “Task-relevant information only”.
8. *Stick to One Primary Action per Screen.*
9. *Use Typography to Create Visual Hierarchy* by using different font styles.
10. *Stick to a Small Number of Gestures*

Therefore, we have made these decisions in our design to adhere to the principle:

1. Placing the login and logout screen in the popup, since that will merely be used twice, once at the start and once at the end. Exclusively displaying the widgets that are applicable to the state of the user’s browser actions and choices. This is also explained in Chapter 5. Rate Your Search also aims to use clear language by using “submit” or “update” in the buttons, using descriptive questions/choices. Such as asking them “How would you rate the relevance of this website to task: task name?”, as well as using the description of the relevance scores (“No relevance”, “Low Relevance”, etc) instead of numbers. Additionally, the tool also presents the description of the relevance scores when you hover over the label.

2. Each of the widgets use the same layout. Similarly, the textual elements on the Google SERP and the widget in which the user selects the relevance are exactly the same.
3. The decision to keep the widget at the bottom right was deliberate as this will presumably not cover up any other content of a web page and is easily accessible.
4. Rate Your Search uses light grey on the Google SERP, and black and white for the widgets and the popup. This is so that it stands out but does not distract you from the rest of the web page. We also make sure the buttons have a different background color than the rest to emphasize that their role.
5. The familiar UI elements are the radio buttons that suggest the user they need to make a choice, the check boxes to suggest the user can select multiple choices, the arrow besides the titles in the widget to imply they can toggle the visibility of the widget. Additionally, we made sure the buttons had a different background from the rest to imply its function. Furthermore, we made sure that the button's opacity slightly changes when the user hovers over it like most modern buttons.
6. We notify the user with the notification bar or a text change (from "submit" to "update") whenever they made a decision or something went wrong in the process of confirming their decision in the back-end. They are also in control because of the collapsible widgets.
7. As already mentioned before, Rate Your Search particularly displays the widgets necessary. Also, once the user has selected their task, the task widget is automatically collapsed, and stays collapsed until the user specifies their need to open it. In addition, the task name is inserted in the header so that the user is reminded which task they are solving.
8. Each widget or box on the Google SERP has one specific intent. It could be for selecting the relevance of the search result, or for the knowledge types in the search result, or for selecting the task, or even logging in and out.
9. The questions in the header of the widgets are larger to draw attention. The relevance score strings have a dotted line underneath them to hint at that there is more to it than meets the eye, which are the tooltip containing the descriptions of the relevance scores. The buttons have another typography to distinguish them from the rest of the tool.
10. To improve on usability, Rate Your Search makes some of the choices for the users, so that the users do not have to click any buttons. This is evident in that the user automatically selects their choice of task by changing the selection of the drop-down without the need of a button. This is because the users are assigned a task,

and they can go back to the description if they need it, but, in this way, we removed the necessity of another click. Also, once they have achieved this, the widget collapses and the notification appears to notify them and which automatically closes. We also disable the buttons if they have not made any changes, so that they realize what choice they have previously made.

### **Usability evaluation survey**

Secondly, we created a usability evaluation survey. To do this, we researched existing surveys and assessed which one were most applicable to Rate Your Search. We looked at UEQ [35], SUS [36] and the surveys of which the validity and the reliability have been established according to Gary Perlman [37]. Our goal was to create a survey that would give more insight into the general usability of the system. Therefore, we eliminated the more detailed questions such as the questions about system speed. Additionally, since Rate Your Search is not to improve on any service any software or process yet, but to enhance future research, we dismissed questions that would examine whether it would.

After reading all the potential questions, we selected these three questions, where the respondents had to answer based on a five-point scale from 1 to 5, where 1 was "strongly disagree" and 5 was "strongly agree":

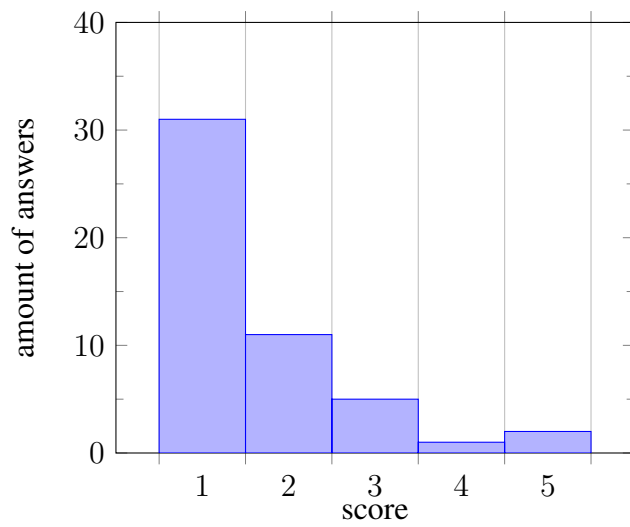
- I needed to learn a lot of things before I could use the plugin.
- I felt comfortable using the plugin.
- I found the various functions in the plugin were well integrated.

This survey was conducted after 50 practitioners conducted a experiment with Rate Your Search and the instructions from Appendix A.

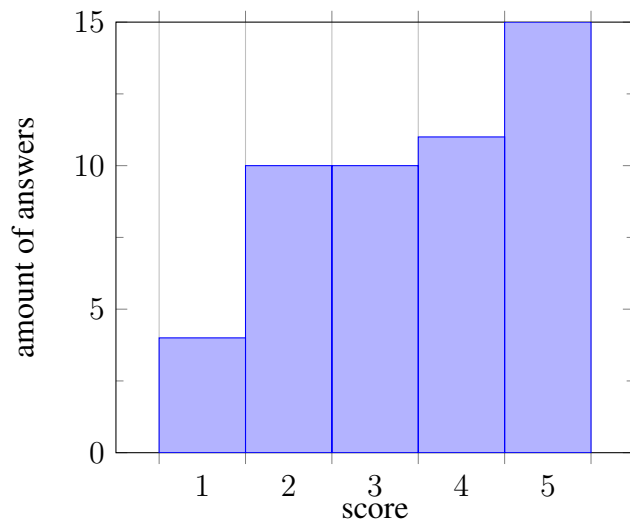
## **6.2.1 Results**

These are the results of the survey:

**I needed to learn a lot of things before I could use the plugin.**

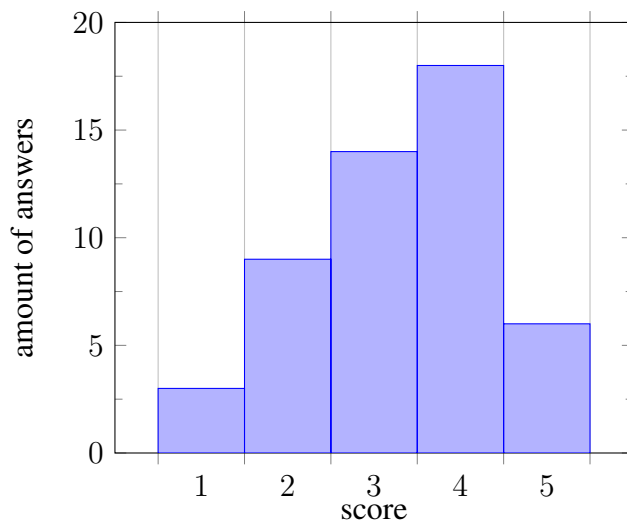


**I felt comfortable using the plugin.**



**I found the various function in the plugin were well integrated.**





### 6.2.2 Evaluation

From our efforts and improvements during the development phase mentioned in section 6.2, we can say we adhere to most design principles.

However, we could improve in our consistency by also including widget for the knowledge types since the form is included on the Google SERP, but not displayed in a widget so that the users could easily access the form. This would improve our consistency as well as improve our efforts to adhere to the principle to stick to a small number of gestures.

Additionally, we adopted this list as the principles to keep in mind since we think that these summarize most UI usability principles, but there are a lot of other principles on the internet that we might have overlooked during the development process on which we could have improved and even enhanced our tool with. Nevertheless, we can conclude the tool adheres to most UI usability principles.

From the usability survey, we first see that most people did not need to learn a lot of things before they could use the plugin, which means Rate Your Search is suitable for the experiment and is easy for first time users to pick up.

Secondly, the responses for the statement "I felt comfortable using the plugin" was divided, as 16 respondents (strongly) agreed, 14 respondents (strongly) disagreed and 10 respondents were conflicted. This means that there is definitely something to improve in the tool to make the users more comfortable. Consequently, we should ask these respondents their reason for their answer before we can know what to improve on.

Thirdly, the respondents were more positive than negative about the integration of the various functions in the plugin. Thus, Rate Your Search has integrated most functions well, but can improve on other functions. Similarly, we should ask the respondents for their justification for their answer to recognize which functions should be better

integrated. However, we do expect it also refers to the knowledge type form as it is not fully integrated.

### 6.3 Functionality

To evaluate the functionality of the tool, we will conduct our own a little experiment which will acknowledge all of our requirements in and see if the results are as we expected and in Chapter 4. For readability, we reiterate the requirements here:

1. It captures keywords from Google.
2. It captures the URLs from the top 10 search results or all search results on the first SERP, if those are less than 10.
3. The user is able to decide on the relevance for each URL on the search engine but also on the website itself denoted by using a Likert scale.
4. The user is able to decide which knowledge type is included for each URL on the search engine.
5. The user is being presented the tasks which they can select and hide. The user can also change their selected task.
6. It captures whether the user has clicked on a search result.
7. The database is structured.
8. It has secure accounts for users, so that we can compute how many queries it took for them to find their desired information.
9. It measures the search relevance with NDCG in the database.
10. It works in all countries.
11. It is secure.

In addition, as a non-functional requirement, the plugin should be user-friendly. Finally, some optional requirements, such as categorizing the URLs and allowing the user to distinguish a part of a web page as most relevant, would enhance our research even more.

### 6.3.1 Experiment

These are the steps we should follow during the experiment:

1. Login using the username: 'test', and the password: 'test'. This should give us an error message: 'Incorrect username or password'.
2. Then, we are going to login using the username: 'user1', and the password: '123456'. These are valid login credentials.
3. We are going to open Google.com.
4. We are going to select one of the tasks.
5. Then, we are going to execute a search query on google.
6. Open all the results from the first SERP of Google.
7. Select a relevance for all results on their web page. We will select the relevance score for the results in this order: 1, 5, 2, 4, 5, 1, 2, 3, 5, 2, where 1 is "No Relevance", 2 is "Low Relevance", 3 is "Medium Relevance", 4 is "High Relevance", and 5 is "Very High Relevance". If there are less than ten results on the first SERP, we will select the first scores.
8. Then, we will update the relevance score for the first, third and fifth result on their web page to 2, 1, 3, respectively.
9. Then, we will update the relevance score of the second and fourth result on the Google SERP to 4 and 5, respectively.
10. Additionally, on the Google SERP, we should submit that the first result has the knowledge of "Solution's Description", "Solution drawbacks" and "Technical background" in "Others". We should also submit that the second result has the knowledge of "UseCase".
11. Then, we should update that the second result also has the knowledge of "Development and Implementation Guide".
12. Then, we should execute another search query.
13. Then, we should select another task.
14. Then, we should execute a search query for that task.
15. Then, we should select select the relevance of the search results in this order: 1, 2, 3, 4, 5, 1, 2, 3, 4, 5 in the same way as step 7.

16. We should also open the second SERP of Google to make sure Rate Your Search does not capture those results.
17. Then, we should logout.

This should result in our user having two sessions, where one session should start as soon as we select the other one, where each session should have the correct user. All search queries and all results from the first SERP, besides the ads, should be stored with the correct foreign key to the sessions. The URLs should have the final relevance scores, and the search query should have a computed NDCG. All URLs of the first search query should demonstrate that they have been clicked on. The first URL of the first search query should have "true,false,false,false,true,false,Technical background" as the cause. Whereas the second URL of the first search query should have "false,true,false,false,false,true," as the cause.

### 6.3.2 Results

The incorrect login credentials indeed displayed the error message "Incorrect username or password". Once we were logged in with valid credentials, we could indeed see the task widget in Google. We chose the JSON-Search task. We executed the search query "JSON syntax". On the first SERP of Google, there were 10 search results. These were from w3schools, tutorialspoint, json.org, en.wikipedia.org, digitalocean, medium, phphulp, developer.mozilla, javaee.github.io and www-db.deis.unibo.it. Thus, we expect all of them to be in the database.

The json.org result returned a 500 status code and could not be retrieved from the database, so we selected the relevance score on Google instead.

After following steps 7 through 11, we executed the search query "JSON content type". This search query also had 10 search results on the first SERP. These consisted of results from stackoverflow, geeksforgeeks, freecodecamp, developer.mozilla.org twice, developer.atlassian.com, github.com, geoforum, Wikipedia, and symfonycasts.com.

Afterwards, we selected the Physical-Design task and executed the search query "web app design". This search query had 9 search results consisting of results from nl.pinterest.com, printerest.com, dribbble twice, budibase, designfounders, designmodo, codica, and webapphuddle. The second SERP included results from fuselab twice, fluidui, thedroidsonroids, printerest.dk, awwwards, clearbridgemobile, behance, proto.io. However, these should not be inserted in the database.

The account used in the experiment has the id "1" in the database. The tasks from the database, without description since it is not applicable to this experiment, can be seen in Figure 6.1. Figures 6.2, 6.3, and 6.4 represent the session, searchqueries and URLs that were created during the experiment. The URLs are sometimes cut off since they were too long. These URLs are depicted by the URLs that end with "...". Moreover, some

id	taskname
1	Physical-Design
2	Big-Data-Stream-Evaluation
3	Conceptual-Design
4	Middleware-Search
5	JSON-Search
6	Messaging-Evaluation

Table 6.1: The tasks in the database.

id	userid	taskid	datetimestart	datetimefinish
372	1	5	2020-12-01 12:41:54	2020-12-01 12:51:10
373	1	1	2021-12-01 12:51:12	2020-12-01 13:00:22

Table 6.2: The sessions recorded from the experiment in the database.

column names use aliases for the same reason, e.g. rank for rankingoogle, cl for clicked, and sqid for searchqueryid.

In those figures, we see that a new session is started two seconds after the first one ended. This is as expected. We also see that all the URLs and their rank, relevance and cause that we anticipated are recorded.

### 6.3.3 Evaluation

The results of our experiment demonstrate that the plugin meets the requirements 1, 2, 3, 4, 5, 6, and 9. However, requirement 6 did not work for the <https://www.json.org> website.

Admittedly, it should be noted that this experiment was extremely small. We, undoubtedly, tested the plugin during development extensively, where we discovered that we could improve the performance by including threads on the server, so that more users can use the plugin at the same time. Additionally, the usability experiment also demonstrated that the tool is functional.

Unfortunately, evaluating requirement 7 is complex. Although, we did adhere to the principles we have been taught in our career such as making sure the columns and tables are logical and concise.

id	searchquery	sessionid	ndcg
1055	JSON syntax	372	0
1056	json content type	372	NULL
1057	web app design	373	0

Table 6.3: The search queries recorded from the experimented in the database.

id	url	rank	relevance	cl	sqid	cause
15179	<a href="https://www.w3schools.com/js/js_json_syntax.asp">https://www.w3schools.com/js/js_json_syntax.asp</a>	1	2	1	1055	true,false,false,true,false, Technical background
15180	<a href="https://www.json.org/">https://www.json.org/</a>	3	1	0	1055	NULL
15181	<a href="https://www.tutorialspoint.com/json/json_syntax.htm">https://www.tutorialspoint.com/json/json_syntax.htm</a>	2	4	1	1055	false,true,false,false,true
15182	<a href="https://en.wikipedia.org/wiki/JSON">https://en.wikipedia.org/wiki/JSON</a>	4	5	1	1055	NULL
15183	<a href="https://www.digitalocean.com/community/tutorials/an-introduction-to-json">https://www.digitalocean.com/community/tutorials/an-introduction-to-json</a>	5	3	1	1055	NULL
15184	<a href="https://www.phpuhp.nl/php/tutorial/php-algemeen/json/810/json-syntax/2249/">https://www.phpuhp.nl/php/tutorial/php-algemeen/json/810/json-syntax/2249/</a>	7	2	1	1055	NULL
15185	<a href="http://www-db.deis.unibo.it/courses/TW/DOCS/w3schools/json...">http://www-db.deis.unibo.it/courses/TW/DOCS/w3schools/json...</a>	10	2	1	1055	NULL
15186	<a href="https://medium.com/omarelgabrys-blog/json-in-a-nutshell-7d638dfea7cc">https://medium.com/omarelgabrys-blog/json-in-a-nutshell-7d638dfea7cc</a>	6	1	1	1055	NULL
15187	<a href="https://jvavee.github.io/tutorial/jsonp001.html">https://jvavee.github.io/tutorial/jsonp001.html</a>	9	5	1	1055	NULL
15188	<a href="https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global...">https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global...</a>	8	3	1	1055	NULL
15189	<a href="https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Content-Type">https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Content-Type</a>	4	NULL	0	1056	NULL
15190	<a href="https://stackoverflow.com/questions/477816/what-is-the-correct-json-content...">https://stackoverflow.com/questions/477816/what-is-the-correct-json-content...</a>	1	NULL	0	1056	NULL
15191	<a href="https://geoforum.nl/t/content-type-van-json-response-is-text-plain/300">https://geoforum.nl/t/content-type-van-json-response-is-text-plain/300</a>	8	NULL	0	1056	NULL
15192	<a href="https://www.geeksforgeeks.org/what-is-the-correct-json-content-type/">https://www.geeksforgeeks.org/what-is-the-correct-json-content-type/</a>	2	NULL	0	1056	NULL
15193	<a href="https://github.com/fnproject/fdk-python/issues/37">https://github.com/fnproject/fdk-python/issues/37</a>	7	NULL	0	1056	NULL
15194	<a href="https://symfonycasts.com/screencast/rest/application-problem">https://symfonycasts.com/screencast/rest/application-problem</a>	10	NULL	0	1056	NULL
15195	<a href="https://www.freecodecamp.org/news/what-is-the-correct-content-type-.../">https://www.freecodecamp.org/news/what-is-the-correct-content-type-.../</a>	3	NULL	0	1056	NULL
15196	<a href="https://developer.mozilla.org/en-US/docs/Web/HTTP/Basics_of_HTTP/...">https://developer.mozilla.org/en-US/docs/Web/HTTP/Basics_of_HTTP/...</a>	5	NULL	0	1056	NULL
15197	<a href="https://developer.atlassian.com/server/crowd/json-requests-and-responses/">https://developer.atlassian.com/server/crowd/json-requests-and-responses/</a>	6	NULL	0	1056	NULL
15198	<a href="https://en.wikipedia.org/wiki/JSON">https://en.wikipedia.org/wiki/JSON</a>	9	NULL	0	1056	NULL
15199	<a href="https://nl.pinterest.com/vahndabruka/web-app-and-ui-design/">https://nl.pinterest.com/vahndabruka/web-app-and-ui-design/</a>	1	1	0	1057	NULL
15200	<a href="https://www.pinterest.com/weitseng/web-application/">https://www.pinterest.com/weitseng/web-application/</a>	2	2	0	1057	NULL
15201	<a href="https://dribbble.com/tags/web_application">https://dribbble.com/tags/web_application</a>	4	4	0	1057	NULL
15202	<a href="https://www.budibase.com/blog/5-examples-of-web-application-design/">https://www.budibase.com/blog/5-examples-of-web-application-design/</a>	5	5	0	1057	NULL
15203	<a href="https://www.codica.com/blog/progressive-web-app-design-7-tips-...">https://www.codica.com/blog/progressive-web-app-design-7-tips-...</a>	8	3	0	1057	NULL
15204	<a href="https://designmodo.com/web-application-interface/">https://designmodo.com/web-application-interface/</a>	7	2	0	1057	NULL
15205	<a href="https://designforfounders.com/web-app-ux/">https://designforfounders.com/web-app-ux/</a>	6	1	0	1057	NULL
15206	<a href="https://dribbble.com/tags/web_app">https://dribbble.com/tags/web_app</a>	3	3	0	1057	NULL
15207	<a href="https://webapphuddle.com/step-by-step-web-app-design/">https://webapphuddle.com/step-by-step-web-app-design/</a>	9	4	0	1057	NULL

Table 6.4: The results from the experiment in the urls table.

Rate Your Search meets requirement 8 partially. The database contains accounts for all users. Though, they could be more secure as the accounts information should be encrypted. Everyone with access to the database can see the account names and passwords.

We made sure the plugin could work in all countries, by specifying in the manifest that every URL is allowed to work with the plugin. Additionally, we do not apply a limit on IP addresses on the server for our REST API and database, as well as not mentioning any domain in the content script to meet requirement 9.

Lastly, we should certainly improve on our security. Despite our efforts, such as that we have made it so that our REST API uses a SSL certificate and HTTPS, the data should be more secured. Particularly, the accounts should be encrypted. Another effort we implemented is using as little message passing in the web extension as possible and validating each message.

Another security flaw is that once a user knows the URL of the REST API requests and server, they can enter the URL in their browser and see the results. Since the experiment gives the participants the source code to upload to Google Chrome, they could easily find this in the source code. This could be improved upon by uploading the plugin to the Chrome Store so that the participants do not have access. Additionally, by restricting the HTTP requests from an unknown user by potentially using login credentials, we could solve the problem of being able to use the browser to look into the data. Also, Spring offers the Spring Security framework to enhance the security.

The non-functional requirement for the tool to be user-friendly is evaluated in Section 6.2.

In conclusion, the tool is functional and is suitable to be used in further research in a larger experiment, but it can be improved on by enhancing the security and testing more in other countries.

# 7

## Conclusion

The goal of this thesis was to develop a plugin which captures architectural information from search engine results on a browser by capturing the search queries, results and allowing the user to determine the relevance of a search result. This plugin will enhance future research to solve the problem of finding software architecture information. In this paper, we have looked at previous solutions to this problem, and implemented our own plugin to try and solve this problem.

In fact, we have developed a web extension for Chrome which can capture the relevant and required information and request the user for their input which uses a Spring Boot REST API to store this information in a database. The REST API and database are deployed in a docker container on the Amazon AWS, a cloud computing service. The web extension itself is implemented using HTML, CSS, and JavaScript.

In this paper, we also evaluated the tool's usability and functionality. We found that the tool's usability is satisfactory but could be improved upon. Consequently, more research should be conducted to accurately determine where specifically. Furthermore, we determined the tool is functional but its biggest concern is the security. However, since the tool is aimed for an experiment with trusted practitioners and not for widespread usage, the security we have implemented is enough for us to utilize the plugin.

In conclusion, we have successfully developed a functional plugin which captures the search queries and results Google in Google Chrome. It also allows users to determine the relevance of the search results on the SERP and on their web page. The plugin can certainly be used in the experiment we mentioned in the introduction as such an experiment is already conducted before conducting the user evaluation survey. We hope that this plugin will augment the research in the field of information retrieval.



# 8

## Future Work

Our evaluation concluded that there were elements to improve or enhance on, which were:

- Generally, we could augment the usability survey with more questions, where the user would have to justify their answer with feedback. In this way, we could improve the usability, particularly, assure the users are comfortable, and the functions of the plugins are well integrated.
- The knowledge types form should have its own widget, so that users can submit the knowledge types form on the web page instead of needing to submit it on the Google SERP.
- From the functionality evaluation, we concluded that we should integrate more security measures into the tool. This could be done by uploading the tool to the Chrome Web Store, using the Spring Security framework.
- There was one functional requirement that we did not meet. This was to allow the user to distinguish a part of a web page as most relevant. Adding this would enhance the plugin as well.

Also, from the development and our own testing, we also notice a couple of points of improvement:

- The plugin could be made more efficient, since it creates a form for each result, and we could already store the URL id in a class name in the DOM, so that we

can retrieve the URL object more efficiently with the id instead of URL and search query id.

- To make the plugin more universal, it should support more browsers and search engines.
- There could be more error code handling in the REST API.



## User's guide

### **Searching User Guide**

#### **Goal of the Experiment**

Explore the effectiveness of the web search engine (Google), and its abilities to search for architectural information in the web during the architectural design process.

#### **Overview on the Experiment**

Software architecture design tasks have a big scope, which involve functional and non-functional requirements, as well as constraints. You will be asked to search for architectural information on Google to solve six different design tasks. For each task, you will write queries (keywords) in Google, and evaluate the relevance of each web page to a task. During your searching, you will be using a Google chrome extension to support specifying the relevance of each web page to the tasks. The Google chrome extension captures in background your queries, the searching results, the specified relevance of each web-page and the specified types of information in each web-page. During the experiment, you will be asked to specify the relevance and the types of relevant information in each web-page. We explain both in the following sub-sections.

### Web-pages relevance

The relevance of each web-page (in the list of pages listed by Google) to help complete the task is defined on five levels:

- **Very High Relevance (VH):** The web page discusses a similar problem to the task and contains useful information. The web page provides an answer to the searching goal, and helps with fulfilling more than one requirement of the task.
- **High Relevance (H):** The web page addresses a similar problem to the task and contains useful information. The post provides an answer to the searching goal, and helps with fulfilling at least one requirement of the task.
- **Medium Relevance (M):** The web page addresses another problem not similar to the task at hand, but it provides some relevant information to the task, which could be an answer to the searching goal. Nevertheless, the provided information does not consider specifically the task's requirements.
- **Low Relevance (L):** The web page contains relevant information, which is only remotely relevant to solving the given task, but might help for refining the search.
- **No Relevance (N):** The web page has nothing to do with the task. It has no relevant information.

### Types of relevant information in web-pages

To facilitate specifying the types of relevant information in web-pages, we designed seven types from which you will select one or more type for each relevant web-page. The types of relevant information are described below:

- **Solution's description:** The web page contains general information on an architectural solution. For example, general information on the supported features of a technology or descriptions of architectural patterns.
- **Development and implementation guide:** The web page contains information on how to implement an architectural solution (examples, development guides, installation guides, code examples).
- **Solution alternatives:** The web page contains multiple (alternative) solution options for a certain design issue. For example, lists of broker technologies, lists of SOA patterns. The architectural options could be listed in text or as a comparison of different solution options.

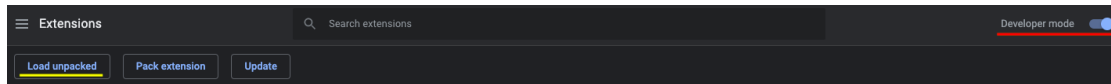


Figure A.1: Upload your own Chrome Extension.

- **Solutions benefits:** The web page contains information about the advantages of certain architectural solutions. For example, the advantages of using a certain technology regarding its performance, which can be supported with benchmarks or tests. Those advantages might be part of a comparison with other options.
- **Solutions drawbacks:** The web page contains information about the disadvantages of certain architectural solutions, or even discourages its application. For example, the disadvantages of using a certain technology regarding its security or Information on when not to decide on an architectural solution.
- **UseCase:** The web page contains explanations about the architecture of a specific system. This includes description for an existing architectural design of a specific system, or explanations about certain design decisions of a specific system. For example, a web page explains the software architecture of Facebook.
- **Others:** The web page contains other relevant architectural information. Please write down which other information you found.

## How to install Google chrome extension?

Before solving the tasks, you should install the attached Google chrome extension: "Rate Your Search". You can install the Google chrome extension using the following steps:

1. You will be given a zip file with the needed source code. You need to extract the zip file in a folder.
2. Open a new Google Chrome window.
3. Go to `chrome://extensions/`.
4. Turn on Developer mode at the top right (see the red line in Figure A.1).
5. Press "Load unpacked" at the top left (see the yellow line in Figure A.1).
6. Choose the folder, where the zip file is extracted.
7. Now the plugin should be in the list of extension.

## Experimental Procedure

When completing the tasks, you can follow several steps:

1. *Log in*: Log in within the Google chrome extension with your assigned user name and password. To reach the login page, press the extensions icon on the top right of your chrome toolbar. It should look like a red "magnifying glass" as presented in Figure A.2. If it does not show up, press the puzzle piece as presented in Figure A.2 instead and pin the "Rate Your Search" extension chrome extension.



Figure A.2: The icon of Rate Your Search is the red "magnifying glass".

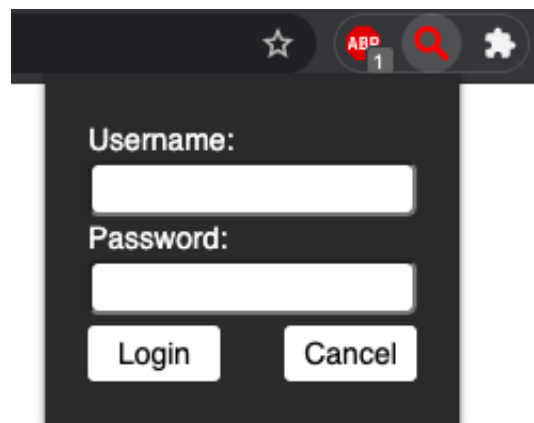


Figure A.3: The login screen of Rate Your Search.

2. *Read task*: Read the task carefully to understand the requirements and the goals of the search.
3. *Select task in plugin*: Go to Google.com and select the task you are conducting. Once you reach Google, there should appear a widget at the bottom right corner looking like Figure A.4. Here, you can select your task from the drop-down box. Moreover, you can view the description of the task in the widget at the bottom right (Figure A.5).
4. *Conduct Search*: Start searching for the relevant web-pages and information that could help achieving the goal and fulfilling the requirements of a given task. During the search, follow these steps:

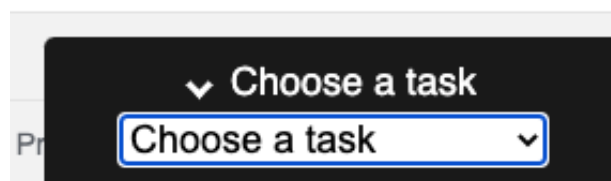


Figure A.4: The task widget of Rate Your Search

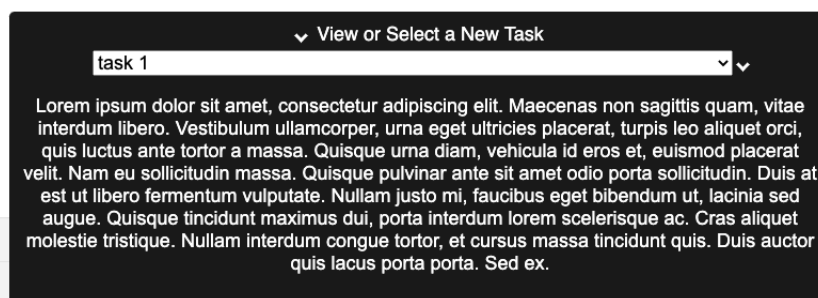


Figure A.5: Rate Your Search: ability to read task description in widget.

- Execute **at least three queries** in Google for each task. Try to find as many web-pages as possible relevant to the goal and requirements of a task, and try to execute different queries to cover all the aspects of the task (e.g. different requirements, solutions, constraints). The executed queries must be written in English.
- For each query, assess the **top 10 results**, which are returned from Google. During assessment, **open each web-page, and read it carefully** to make sure that it is relevant to the problem, and do not assess the relevance or the types of information just based on the title of a web-page. Please do not assess any other web pages (e.g. advertisements), which are not from the top 10 results. For each web-page, you need to specify the relevance of a web-page. The relevance assesses the relevance between the content of a web-page and the task (not the query). For relevant web-pages (i.e. web-pages with low or medium or high or very high relevance), please specify the types of knowledge in each web-page.

On the Google results, you can specify the relevance score of each result web-page directly. This should be done using the box just below each result, as can be seen in Figure A.6. Web-pages to be assessed should have the evaluation box in Figure A.6. Any other web-pages (e.g. advertisements) will not have the evaluation box.

Moreover, you can specify the types of architecture knowledge for each web-page as part of the Google results. You can select multiple types of

How would you rate the relevance of this website to task: "task 1"?

No Relevance
  Low Relevance
  Medium Relevance
  High Relevance
  Very High Relevance

Figure A.6: Rate Your Search: select the relevance score for a search result on the search engine result page.

Which knowledge exist in this website, which support solving task: "Physical-Design"?

Solution's description
  Development and implementation guide
  Solution alternatives
  Solutions benefits
  Solution drawbacks
  UseCase

Others

Figure A.7: Rate Your Search: select the cause of the relevance score for a search result on the search engine result page.

knowledge for each web-page directly. This should be done using the box just below each result, as can be seen in Figure A.7.

5. *Go to the next task:* After finishing one task and executing enough queries (at least 3 queries), you can go to the next task by selecting the task from the tasks widget as shown in Fig A.4. You can then conduct the search similarly to step 4. For each task, you need to execute new queries. You cannot re-use queries from previous tasks.
6. *Logout:* After you have completed all the tasks in their sequence, you can log out using the same popup you used to log in. Click on the icon shown in Figure A.2, and press the logout button (Figure A.9).
7. *Remove extension:* After logging out, you can remove the extension by clicking "Remove" for the extension at `chrome://extensions/` (Figure A.10).

## Important notes to consider during the experiment

During solving the tasks, please consider the following:

- Solve the tasks in their provided sequence.
- You cannot use any other search engines or web browsers other than Google and Google Chrome.



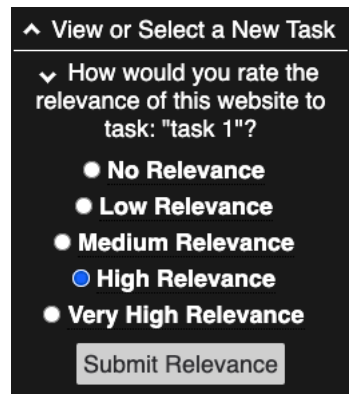
A dark-themed user interface element. At the top, it says "View or Select a New Task" with an upward-pointing arrow. Below that, a downward-pointing arrow indicates a dropdown menu. The main text asks, "How would you rate the relevance of this website to task: 'task 1'?". There are five radio button options: "No Relevance", "Low Relevance", "Medium Relevance", "High Relevance" (which is selected with a blue dot), and "Very High Relevance". At the bottom of the form is a button labeled "Submit Relevance".

Figure A.8: Rate Your Search: select the relevance score for a search result on its web page.

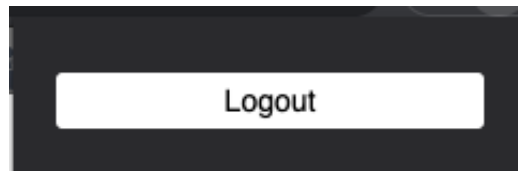


Figure A.9: Rate Your Search: log out button.

- You should not open any web-pages, which are not part of Google's results.
- You should execute the experiment in one Google chrome window.
- Only rate the web page found by google - Do not rate the associated web pages.

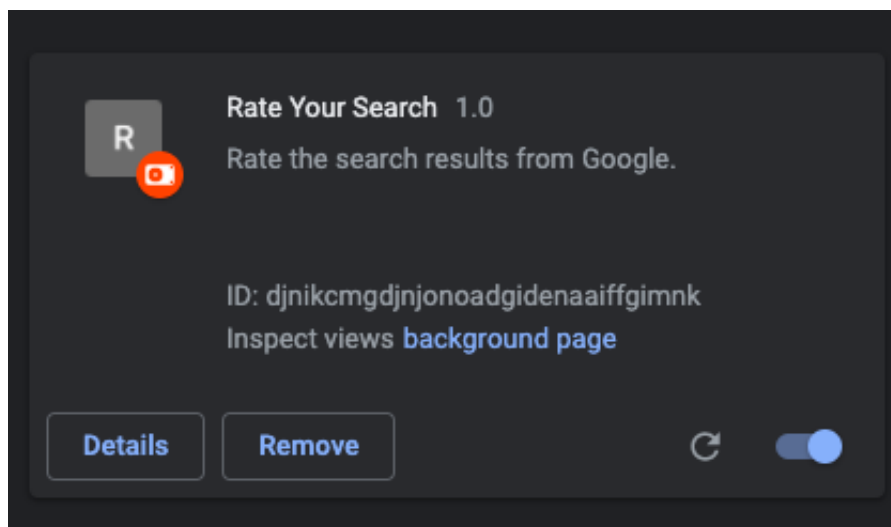


Figure A.10: This is what you will see in `chrome://extensions/` once you have installed the extension.

# B

## Programmer's guide

### B.1 Database

Before we designed our database, we decided on MySQL as our database type. The reason for this was that it was easily compatible with the rest of our tool.

From here we started to design our database. We made sure that each element from our tool had their own table with their corresponding attributes. The elements we treat are `accounts`, `tasks`, `sessions`, `searchqueries`, `urls`. We kept each element as straightforward as possible by naming each attribute appropriately. Additionally, each constraint such as not null or foreign keys maintain data integrity. The full database structure can be seen in B.1.

The database structure is implemented automatically by Spring Boot in the REST API and the annotations at each data member. The specifics of this are mentioned below.

### B.2 Rest API

As mentioned, our Spring Boot REST API automatically implements the database structure. We achieve this by implementing each database table as a class in the model directory since we follow a MVC design pattern. The model is named as the singular version of the table name. Thus, for example, the model for the `accounts` table is named `Account`.

Each model has the `@Entity` annotation and a `@Table` annotation, in order for Spring Boot to recognize it as a database table. Each attribute from the database structure

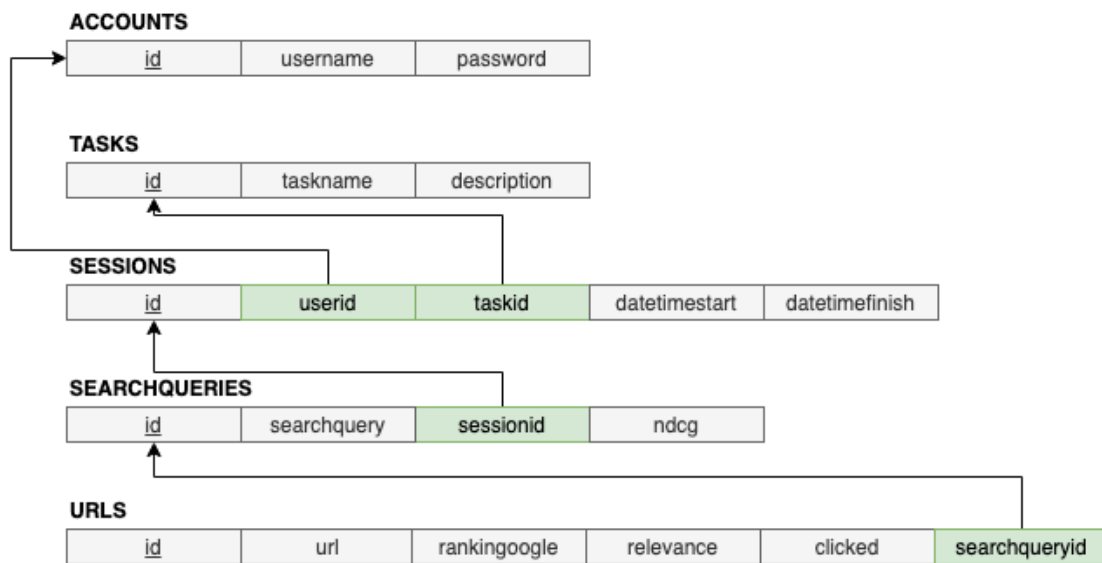


Figure B.1: The complete database structure, created in draw.io

is its own private data member with a corresponding getter and setter method. The data members have a `Column` annotation which specifies the name of the column and the nullability in the database. Foreign keys are also automatically recognized by their naming.

As a note, we would like to mention that having underscores as well as capital letters in the data members brought multiple complications with the implementation of the REST API. Therefore, each attribute is written in lower case and does not contain underscores.

We did not include a view for each model since we do not need one.

On the contrary, we do require a controller for each model. As these controllers handle all the endpoints. The controllers have the annotations `@RestController` to recognize them as the handler for the endpoints, `@CrossOrigin` to enable Cross Origin requests by including a headers for Cross-Origin Resource Sharing (CORS) in the response, which we needed to make the tool functional. As well as the annotation `@RequestMapping` to specify the path of each controller. We made the path of each controller, the name of the table. Thus, for example, the accounts controller has the path `"/accounts"`. However, each endpoint can specify their own path that is appended to the path.

Each controller implements the most basic CRUD (Create, Retrieve, Update, Delete) methods. These are the basic CRUD each controller has:

- **GET request `"/all"`**: to retrieve all entities of a model.
- **POST request `"/add"`**: to create a new entity where a request body in JSON

format is passed with the information of the new entity.

- **GET request "/{id}"**: to retrieve a specific entity with that id.
- **PUT request "/{id}"**: to update a specific entity with that id. This request also requires a request body.
- **DELETE request "/{id}"**: to delete a specific entity with that id.

Some controllers implement more essential GET requests for the tool.

All controllers use a repository to implement these methods. This is also implemented by Spring Boot with the use of the `@Repository` annotation and the `CrudRepository` parent class. We define supplementary query methods in the repositories for those controllers with custom GET requests.

The retrieve methods from these repositories can throw an exception when such an entity is not found. We implemented these exception for each entity accurately.

Moreover, we needed an endpoint to retrieve a file located in our target folder. The `FileController` and the `FileException` classes implement this endpoint.

Besides these models, controllers, repositories and exceptions, we have two more files: the `WebConfig` and the `RestApiApplication`. The latter runs the application in main and has two functions implementing the HTTPS redirect. The HTTPS redirect enforces secure requests due to exclusively allowing requests using the HTTPS protocol and redirecting the HTTP requests to use HTTPS. Whereas the former achieves the enabling of Cross Origin requests.

Importantly, the REST API is deployed using Docker on our AWS EC2 web service. We have two container, the mysql container containing our database and the app container for our REST API. Therefore, the URL of our EC2 web server is also the URL of our HTTP requests. This ensures the ability of creating requests from our Chrome web extension and operates the handling of many requests from multiple users.

Consequently, the app container depends on the mysql container as the REST API is not functional without the database.

### B.3 Web extension

Our web extension is implemented using HTML, CSS and mostly JavaScript. The extension is designed for Chrome, therefore, the manifest specifies the pages and the developer tools utilized to make our plugin functional on Chrome. The rest of the plugin implementation consists of the popup, the background script and the content script.

Firstly, the manifest properties speak for themselves. However, we would like to mention that `manifest_version` is required to be 2, since version 1 is deprecated. Additionally, we added the JQuery script to our manifest so that our other JavaScript scripts can utilize JQuery.

Secondly, the background script does not much more than initializing the storage values once the extension is installed.

Thirdly, the popup is relatively straightforward. It has two div containers that toggle between visibility depending on the state. The first container is the loginContainer with a generic login form. The second container is a container with a logout button. The popup JavaScript handles the back-end which entails partly by starting/ending sessions by executing an AJAX request to the REST API. Additionally, it determines which container to display. We store the user id and whether the user is logged in in the localStorage Chrome API. This enables our other scripts to easily access these data values from here as well. Consequently, logging out resets the localStorage values. Additionally, the popup script also uses message passing from Chrome to relay a message to the content script to reset and close all the widgets.

Lastly, the content script does most of the work. It creates the HTML elements for all the widgets, and creates all necessary event handlers of those widgets which deal with the REST calls or front-end updates. One substantial function in the content script is the `showCorrectWidgets` function. The function determines which widgets to display according to the current state of the input given or the page the user is on if and only if the user is logged in. The workflow is as follows:

1. It first makes sure everything is hidden. This is because most of the widgets get created before this function is called, and their default visibility is set to visible.
2. Then, it checks if the user is on a Google SERP.
  - (a) If so, it displays the task widget. It also retrieves from the local storage whether the user already has chosen a task.
  - (b) If the user has chosen task, it selects that task in the drop-down, but also collapses the task widget since it is not needed.
  - (c) Additionally, it has to check whether the user is on the first SERP of Google, so that it also retrieves the search results if necessary and displays the relevance widget and the cause widget at the first 10 search results, with the chosen relevance and cause, if applicable. Otherwise, the relevance widgets stay hidden, whereas the task widget stays displayed.
3. If the current page is on a Google SERP, we need to check whether we are on a search result.
  - (a) If so, it displays the task widget, but collapses it.
  - (b) Also, it displays the relevance widget. If a relevance score has already been selected, it collapses the relevance widget as well.

The function extracts necessary information to determine some of the conditions from the URL, or from the local storage as other information is not available yet.

On the other hand, we store some of the information in the attributes of an HTML element to extract the information quickly and effortlessly. This information will later be used to either be able to execute a REST call or determine the style of an element. For example, the class list for the following elements are formatted as such (where elements within `{}` are different on most pages):

- For the relevance container at a search result on the Google SERP, the class list is: `relevanceSERPForm plugin {searchqueryid}`. This container also stores the title of the search result in the `title` attribute, which will be used in the notification text to clearly notify the user of their update.
- For the actual relevance form container within the former container, the class list is: `inpRelevance {rankingoogle in words} {URL} {chosen relevance}`. The rank in google is to differentiate each form on the SERP from each other to ensure that each input element is connected to their search result. The chosen relevance is to determine which text to display and the disability of the submit/update relevance button.
- The name for the radio buttons are `relevance with {rankingoogle}` appended. The id for the radiobuttons are `rel with {relevance score as a number}` and `{rankingoogle in words}` appended. For example, the radio button for the first search result with the "No Relevance" will have "relevanceone" as their name and "rellone" as their id. Consequently, the label for the radio will use "rellone" for the `for` attribute.
- The submit relevance button, for these form containers, has `btnSubmitRelevance {rankingoogle in words}` as its class attribute, and `btnSubmitRelevance with {rankingoogle in words}` appended as its id.
- Although the knowledge types form uses the same style guide, instead of "Relevance" and "rel", it uses "Cause" and "cause", respectively.
- In contrast to the search result elements, the widget element uses a different guide. Particularly, the widget at the top right has this as the `class` attribute: `plugin {searchqueryId} {rankingoogle}`, where the rank is merely included if we are on a search result's page.
- The radio buttons in the relevance widget are distinguished using the same method, but without the `{rankingoogle in words}` appended, as there are no other relevance forms on a search results' page to distinguish from.

- Moreover, the relevance form in the relevance widget does not need the rank or URL either, but it does need the chosen relevance. Therefore the class list for that element is equal to: `inpRelevance {selected relevance}`, if the user has selected a relevance.

The rest of the content script is split into four types of functions: helper functions, REST call functions, event handler functions, create widget functions.

The helper functions are simple functions for us to clean up code, for example during condition checks. Furthermore, the REST call functions are just functions which execute the REST calls with the object and id that it is passed to since the content script executes these REST calls often with different objects. These function are often called upon by the event handlers, which are added to the radio buttons, drop-down as well as buttons and anything else. The event handlers are also sometimes simple functions to either toggle the collapsed state of a few widget elements. Lastly, the create widget functions create the HTML elements and append them to the correct DOM element.

This web extension is deployed by uploading the zip with the manifest to your local chrome extensions in developer mode. More details can be found in Appendix A.



# Bibliography

- [1] I. Gorton, R. Xu, Y. Yang, H. Liu, and G. Zheng, “Experiments in curation: Towards machine-assisted construction of software architecture knowledge bases,” in *2017 IEEE International Conference on Software Architecture (ICSA)*, April 2017.
- [2] M. Soliman, A. Rekaby Salama, M. Galster, O. Zimmermann, and M. Riebisch, “Improving the search for architecture knowledge in online developer communities,” in *2018 IEEE International Conference on Software Architecture (ICSA)*, April 2018, pp. 186–18 609.
- [3] M. C. Oussalah, *Software Architecture 1*. John Wiley & Sons, Incorporated, 2014.
- [4] J. L. Ledford, *Search Engine Optimization Bible*. Wiley, Vol. 2nd ed. 2009.
- [5] “Chrome extensions overview,” <https://developer.chrome.com/extensions/overview>, accessed: 2020-07-17.
- [6] X. Xia, L. Bao, D. Lo, P. S. Kochhar, A. E. Hassan, and Z. Xing, “What do developers search for on the web,” *Empirical Software Engineering*, vol. 22, pp. 3149–3185, 2017.
- [7] S. Gottipati, D. Lo, and Jing Jiang, “Finding relevant answers in software forums,” in *2011 26th IEEE/ACM International Conference on Automated Software Engineering (ASE 2011)*, Nov 2011, pp. 323–332.
- [8] S. Beyer, C. Macho, M. Di Penta, and M. Pinzger, “Automatically classifying posts into question categories on stack overflow,” in *2018 IEEE/ACM 26th International Conference on Program Comprehension (ICPC)*, 2018, pp. 211–21 110.
- [9] D. Indumathi and A. Chitra, “A collaborative search with query expansion and result re-ranking,” in *2011 World Congress on Information and Communication Technologies*, 2011, pp. 985–989.
- [10] E. Opoku-Mensah, F. Zhang, F. Zhou, and P. K. Kittur, “Understanding user situational relevance in ranking web search results,” in *2017 8th IEEE Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON)*, 2017, pp. 405–410.

- [11] N. Chen and V. K. Prasanna, “Rankbox: An adaptive ranking system for mining complex semantic relationships using user feedback,” in *2012 IEEE 13th International Conference on Information Reuse Integration (IRI)*, 2012, pp. 77–84.
- [12] L. Ponzanelli, A. Bacchelli, and M. Lanza, “Seahawk: Stack overflow in the ide,” in *2013 35th International Conference on Software Engineering (ICSE)*, May 2013, pp. 1295–1298.
- [13] M. M. Rahman, S. Yeasmin, and C. K. Roy, “Towards a context-aware ide-based meta search engine for recommendation about programming errors and exceptions,” in *2014 Software Evolution Week - IEEE Conference on Software Maintenance, Reengineering, and Reverse Engineering (CSMR-WCRE)*, Feb 2014, pp. 194–203.
- [14] “Chrome develop extensions,” <https://developer.chrome.com/extensions/devguide>, accessed: 2020-03-11.
- [15] “Chrome manifest file format,” <https://developer.chrome.com/extensions/manifest>, accessed: 2020-07-17.
- [16] “Chrome manifest file format,” <https://developer.chrome.com/extensions/manifest/icons>, accessed: 2020-07-17.
- [17] “chrome.pageaction,” <https://developer.chrome.com/extensions/pageAction>, accessed: 2020-07-17.
- [18] “chrome.browseraction,” <https://developer.chrome.com/extensions/browserAction>, accessed: 2020-07-17.
- [19] “Manage events with background scripts,” [https://developer.chrome.com/extensions/background\\_pages](https://developer.chrome.com/extensions/background_pages), accessed: 2020-07-17.
- [20] “Design user interface,” [https://developer.chrome.com/extensions/user\\_interface](https://developer.chrome.com/extensions/user_interface), accessed: 2020-07-17.
- [21] “chrome.omnibox,” <https://developer.chrome.com/extensions/omnibox>, accessed: 2020-07-17.
- [22] “chrome.contextmenus,” <https://developer.chrome.com/extensions/contextMenus>, accessed: 2020-07-17.
- [23] “Content scripts,” [https://developer.chrome.com/extensions/content\\_scripts](https://developer.chrome.com/extensions/content_scripts), accessed: 2020-07-17.
- [24] “Message passing,” <https://developer.chrome.com/extensions/messaging>, accessed: 2020-07-17.

- [25] “Give users options,” <https://developer.chrome.com/extensions/options>, accessed: 2020-07-17.
- [26] “What is a restful api (rest api) and how does it work?” <https://searchapparchitecture.techtarget.com/definition/RESTful-API#:~:text=A%20RESTful%20API%20is%20an,to%20communicate%20with%20each%20other>.
- [27] “Spring — why spring,” <https://spring.io/why-spring>, accessed: 2020-07-17.
- [28] “Getting started — spring boot,” <https://docs.spring.io/spring-boot/docs/current/reference/html/getting-started.html>, accessed: 2020-07-17.
- [29] “Spring data,” <https://spring.io/projects/spring-data>, accessed: 2020-07-17.
- [30] “Spring boot features — working with sql databases,” <https://docs.spring.io/spring-boot/docs/current/reference/html/spring-boot-features.html#boot-features-sql>, accessed: 2020-07-17.
- [31] A. Dasari, “The https protocol explained! - the basics,” <https://medium.com/@anushadasari/the-https-protocol-explained-the-basics-5ae33b9d651f>, July 2019, accessed: 2020-07-17.
- [32] “Docker what is a container?” <https://www.docker.com/resources/what-container>, accessed: 2020-07-30.
- [33] “Amazon ec@,” <https://aws.amazon.com/ec2>, accessed: 2020-08-10.
- [34] “What are the 10 rules of good ui design? what is good ui/ux design?” <https://medium.com/@OPTASY.com/what-are-the-10-rules-of-good-ui-design-what-is-good-ui-ux-design-3ecc0d575c8f>, accessed: 2020-08-10.
- [35] “User experience questionnaire,” <https://www.ueq-online.org/>, accessed: 2020-08-10.
- [36] “System usability scale (sus),” <https://www.usability.gov/how-to-and-tools/methods/system-usability-scale.html>, accessed: 2020-08-10.
- [37] “User interface usability evaluation with web-based questionnaires,” <https://garyperلمان.com/quest/>, accessed: 2020-08-10.