INTERACTION SYSTEM FOR A MOOC SEARCH ENGINE

Design, implementation and summative empirical evaluation of the User Interface

Eugenia Schneider

Master Thesis

Aufgabensteller: Prof. Dr. François Bry
Betreuer: Prof. Dr. François Bry,
M. Sc. Inform. Yingding Wang

Abgabe am: 01. Februar 2018
Hiermit versichere ich, dass ich die vorliegende Arbeit selbständig verfasst habe und keine anderen als die angegebenen Hilfsmittel verwendet habe.

München, den 01. Februar 2018

Eugenia Schneider
To date, there is a huge amount of resources for education on the Web. Students can utilize these to obtain further knowledge about a desired topic or close subject-related gaps, thus getting more successful in their studies. A popular way of self-learning is through attending Massive Open Online Courses (MOOCs). But the huge amount of MOOCs is both a blessing and a curse. Finding the right course for one’s need is often like searching for a needle in the haystack – or more precisely – in a great number of haystacks, since there are many different platforms that provide courses from various sources. This is why a cross-provider vertical search application would be of great help for students.

Developing such a vertical search application is the purpose of this master thesis. A platform shall be created that offers not only an intelligent search functionality, but also useful personalized recommendations. Built with state-of-the-art web developing techniques the application shall have a user interface that corresponds to modern usability guidelines and provides a satisfying user experience. This includes, on the one hand, full responsiveness because mobile web usage is an indispensable part of our modern everyday life and thus an appropriate design of web applications is expected by today’s users. On the other hand the application shall offer all functionalities which are common for a modern, comprehensive vertical search engine. To increase the benefits of the application, the users shall be given the opportunity to adapt the system to their own interests and use the platform as a personalized tool for their own professional development.

A major part of this thesis deals with the monitoring of the application usage. Extensive metadata about application usage shall provide detailed information about how the employed search and recommendation engines can be improved to offer students a high quality service. Furthermore, with the aid of the gathered metadata, the content- and design-based concept of the platform shall be proven effective regarding design choices of several key aspects, such as the use of a grid layout for search results and their individual presentation. The concept shall be continuously improved and extended further.

A first comprehensive evaluation of the application has been conducted and has yielded many important findings to improve the layout as well as the content of the platform. Several of these suggestions for improvement have been also implemented. The evaluation results have proven the design concept of the layout to be successful. The employed search engine shall be further adapted to increase the efficiency of its usage. The core concept of the recommendation engine was very well received by users. With the extensive metadata gathered over time the recommendation engine can be further developed and adapted to the individual user in the future.

Overall, a solid basis has been created for a successful, high quality search platform which can be further developed and improved with the aid of the provided comprehensive tracking functionality. The first version of the vertical search application is available and ready to support students in their search for the right online courses.


Damit wurde die Basis für eine erfolgreiche, hochwertige Suchplattform geschaffen, die in Zukunft mithilfe der bereitgestellten umfangreichen Tracking-Funktionen weiterentwickelt und verbessert werden kann. Die erste Version der vertikalen Suchmaschine ist verfügbar und einsatzbereit, um Studenten bei ihrer Suche nach den richtigen Online-Kursen zu unterstützen.
I would first like to thank my thesis supervisor Prof. Dr. François Bry at the Institute for Informatics, Ludwig-Maximilian University, Teaching and Research Unit of Programming and Modeling Languages (PMS). He has consistently encouraged this thesis to be my own work and supported me with professional advice as well as his broad knowledge of the human perception regarding modern web user interfaces.

Especially I would like to thank my advisor and second reader of this thesis M. Sc. Inform. Yingding Wang, who gave me the chance to contribute to project IROM, supported me throughout the whole process, and whose door was always open whenever I ran into a trouble spot. I am gratefully indebted to him for his invaluable comments on this thesis and for the chance to learn so much in this field.

I am also grateful to all participants who were involved in the evaluation study for this research project: Georg Bottenhofer, Johannes Hammersen, Christophor Kantscharidi, Johanna Kondrasch, Katharina Popov (from the first test phase) and all other anonymous testers (from the second test phase), who invested their time in testing the application. Their input has helped me a lot to improve the quality of the application and allowed me to come up with new ideas. Without their passionate participation and feedback, the evaluation study would not have been successfully conducted.

Finally, I must express my very profound gratitude to my parents and to my husband Viktor for providing me with unfailing support and continuous encouragement throughout my years of study and especially the process of researching and writing this thesis. This accomplishment would not have been possible without them. Thank you.

Author
Eugenia Schneider
# Contents

1 Introduction ................................. 1  
1.1 Motivation ................................. 1  
1.1.1 Searching the depths of the web ........ 1  
1.1.2 Finding the right online course .......... 2  
1.1.3 Concepts behind IROM ................. 3  
1.2 Objective ................................. 5  
1.3 Related Work ............................. 6  
1.3.1 Generic and Vertical search ............ 6  
1.3.2 Design guidelines for search applications . 14  

2 Conception ................................. 27  
2.1 Overview of the backend structure ....... 27  
2.1.1 Search engine components ............. 27  
2.1.2 Data scheme ........................... 30  
2.2 Specification of the User Interface ....... 32  
2.2.1 Page layout types ..................... 32  
2.2.2 Common components ................... 34  
2.2.3 Connections between pages .......... 35  
2.3 Gathering metadata ...................... 37  

3 Implementation ............................ 39  
3.1 Developing the application .............. 39  
3.1.1 Structure of the rich client ........... 39  
3.1.2 Backend communication ............... 40  
3.1.3 Design of the user interface ......... 42  
3.2 Metadata collection ..................... 50  

4 Evaluation .................................. 51  
4.1 The basics of web analytics ............. 51  
4.2 Testing approach ......................... 53  
4.3 Evaluation results ....................... 55  
4.3.1 Insights from the lab study .......... 55  
4.3.2 Field study results ................... 62  
4.4 Lessons learned .......................... 71  

5 Conclusion and Future work ............... 73  
5.1 Achievements .............................. 73  
5.2 Further enhancements ................... 75  
5.3 Additional features ...................... 78  

Bibliography ................................. 83
INTRODUCTION

1.1 Motivation

1.1.1 Searching the Depths of the Web

At the time of writing there exist over a billion different websites on the web – with upward tendency. And this number stands only for unique domains, not taking the often countless sub-pages of each website into account. Therefore it is often quite difficult to find comprehensive information about a desired topic. This is why specialized search engines emerged which concentrate on providing results from a specific area, and their number continues to grow. Such vertical search applications, as they are referred to, search only a handful of websites that are often selected manually in a meticulous process. This way niche websites can be represented which are not considered by generic search engines, and a more profound information can be offered for the search results, because the content structure of the respective websites is often known to the vertical search engine.

Well-known examples for vertical search engines are all sorts of (price) comparison portals such as Booking.com (https://www.booking.com/), employment search sites such as Indeed (https://www.indeed.com/), and platforms specialized on images such as Pinterest (https://www.pinterest.com/). By now vertical search applications for nearly every conceivable area have been created, be it local businesses reviews, car sales or link collections of reliable scientific medical resources. Verticals are advancing fast and even Google has started to adopt vertical material in its famous general-purpose search engine – providing image search, a concrete news section, and Google Scholar, a search functionality specifically for scientific material, to name only a few.

The specialized search services make life easier for the modern information seeking process. They help the user to maintain an overview of the huge amount of content and not getting lost in the course of exploring several sources. The search results are also presented uniformly, showing the most important aspects at a glance, and provide further information on demand if needed. Thus the searcher saves a lot of time and does not miss valuable content, provided that the vertical search engine has a high quality standard.

1 http://www.internetlivestats.com/total-number-of-websites/, all URLs last accessed 2018-01-09
1.1.2 FINDING THE RIGHT ONLINE COURSE

To date the requirements for future specialists in all areas are higher than ever. We are evolving into a meritocracy, in which one has to succeed in various fields in order to stand out from the average. Therefore students are under strong achievement pressure, so they try to learn new things on their own. Luckily the possibilities to learn something new have also developed to a greater extent in the recent past. With the advances in content presented on the World Wide Web more knowledge on almost any subject is getting freely available for the public.

For students, the trend goes to Massive Open Online Courses (MOOCs), often offered by universities or institutions. MOOCs provide free access to knowledge on various topics and give the ability to connect to the network built around a certain topic on the web. Furthermore they allow to meet and communicate with other people interested in that field. Often it is also possible to get the credit from an institution by paying a certain fee for the certification, which proofs the acquired skills. MOOCs are therefore more than simple online learning videos: they offer interactive discussion boards, textual material with appropriate exercises and digital tests. These ways of digital interaction provide a new and interesting learning experience.

The good and the bad side

The advantages of MOOCs are obvious: free knowledge without access restrictions, provided independently of time and location, and easily accessible via the Internet, to mention only the most important. Since the first open online course "Connectivism and Connective Knowledge", held by Stephen Downes and George Siemens in 2008, the number of MOOCs has constantly risen. By the time of writing this thesis there are over a thousand courses on Coursera (https://www.coursera.org/) and edX (https://www.edx.org/), respectively. And these are not the only platforms where you can find various kinds of MOOCs. On the one hand the number of these platforms opens many opportunities for further education. But on the other it makes it very hard for students to find the right course for their needs in the immense amount of offers.

An example of the information overload

Here is a fictional case:

There is a computer science student Max in the first term of his bachelor study. After a couple of weeks he decides that he has to learn programming on his own in order to succeed in certain courses. He heard about a platform where online courses for various topics exist, so he visits its website and types the search query term "programming for beginners" in the search bar. This results in a list of some courses for the basics of programming, which Max is going to enroll in. Thus he becomes familiar with the topic.

---

2https://www.mz.itstz.tum.de/elearning/moocs/was-ist-ein-mooc/
1.1. MOTIVATION

But this knowledge is not enough for him to understand complex programming logic. Someone, who has a deeper insight in that field, would advise our student to look into the subject of principles of algorithmics. But the simple full-text search engine of the platform cannot give such tips, because the word “programming” doesn’t match the word “algorithmics”. Max decides to search in a general-purpose search engine such as Google Search for further courses on this topic, and gets a lot of results from different platforms with all types of subfields of programming. Trying to follow all the links that look promising he spends a lot of time comparing the results and loses the overview of the wide range of courses.

Vertical search can help

Such situations are quite common and are relativizing the benefits of a broad MOOCs palette. A vertical search engine for online courses, however, would help the student to save time by showing all relevant courses from many different platforms. But it doesn’t have to stop there: this search engine could also provide more intelligent results based on a deeper analysis of the course descriptions and on personalized recommendations. This is the idea behind project IROM (Intelligent Recommender of MOOCs) initiated by Yingding Wang. With deep textual analysis of the course descriptions to provide best results and helpful recommendations, and a rich web client based on modern design principles and an intelligent user interface, this search platform is aimed to support students by finding the right MOOC for their needs and thus to help them succeed in their studies.

1.1.3 CONCEPTS BEHIND IROM

The range of online courses searched by IROM is quite limited yet. Up until now three different MOOC platforms are being crawled to gather their contents: the well-known course provider Coursera that focuses on material from renowned universities, the professional business-driven service Udacity (https://www.udacity.com/) and lastly the university-related Open2Study platform (https://www.open2study.com/), an initiative of the Open Universities of Australia. The search application is going to be extended by other platforms in the near future. For starters the project concentrates only on MOOCs in English, but support of other languages is already in the planning stage.

To get a deeper understanding of what the user is searching for, a classic full-text search engine is not sufficient, therefore other methods of parsing and searching are needed to provide intelligent search results. The textual descriptions of the crawled MOOCs are being analyzed by clustering the term-document vectors derived from the descriptions, thus finding related courses. This approach can help building clusters of courses and derive respective categories. The courses are further processed by the already developed search and recommendation engines. This way relevant courses can be searched by textual queries and similar courses can be retrieved to provide useful recommendations.

---

3 For more details see http://irom.pms.ifl.lmu.de/
**Bringing it all together**

The textual analysis as well as the development of the search and recommendation engines are still in progress, but their results can already be used to begin with the next task of project IROM: Provide a rich web client for students to let them benefit from the search engine and enhance it with the help of direct and indirect user feedback. The web client is meant to be more than just a simple access point to the search functionality. It shall be designed in a way that fulfills usability principles and encourages users to interact with the search platform. Therefore it shall provide additional personalized features to support the user with individual information.

Furthermore it is aimed to gather metadata about users' interactions in the form of click-through data. The analysis of this metadata along with the direct feedback of students is necessary to evaluate the functionality, quality and success of the system. This way especially the recommendations can be enhanced to provide personalized, intelligent suggestions for the users. With the gathered knowledge from this data the entire system of project IROM can be constantly improved to provide a superior search experience in the domain of online courses.
This master thesis aims to design, implement and evaluate a search web application for project IROM, thus addressing the third task of the project described in section 1.1.3, namely to provide a rich web client for students, so that they can benefit from the search engine. This new web client shall be based on the search and recommendation engines which have already been developed, to enable the search functionality of the platform and to offer recommendations for viewed courses. Thus the development of the search and recommendation algorithms is out of scope of this thesis. Instead they are being provided to the web client through a REpresentational State Transfer (REST) API connection.

The main tasks include:

1) **Functionality conception** First a concept of the application shall be elaborated, with all necessary interfaces and functionalities that meet functional and non-functional requirements. This concept shall follow approved design and usability guidelines, gathered from various sources during an extensive literature review.

2) **Design concept** The next important step is to choose an appropriate theme for the future web client, including the general layout and content organization, the main colors, the font style and other common attributes of the desired look and feel.

3) **Implementation** After finishing the concept the web application can be implemented using modern web development techniques and frameworks. During this step a connection to the REST API shall be established to enable search functionality and to offer recommendations. Afterwards the web application is fully functioning and ready for use.

4) **Enhancement with tracking code** To enable a functionality analysis of the platform, the implementation code shall be enhanced further by more logic in order to gather user and click-through data with an appropriate information gain.

5) **Evaluation** After testing in a two-phase approach, in which during the first phase five controlled lab studies with individual users are being conducted and the second phase is composed of an open field study with more participants, the collected analysis data shall be evaluated to draw conclusions regarding the functionality, efficiency and usability of the web search application and possible improvements.

After finishing these tasks project IROM shall have a fully functioning web search client, which collects valuable user data. This data shall be used in future to improve the application and especially the recommendation engine, to provide personalized, intelligent suggestions for the users. Furthermore it can be simply extended by other search or recommendation algorithms in order to evaluate their effectiveness and efficiency. Also, students and other interested users shall be finally given the opportunity to benefit from the already developed algorithms.
1.3 Related Work

1.3.1 Generic and Vertical Search

— How search works —

Every search engine on the web follows mostly the same process pattern, whether it is a generic search service such as Google.com or a specialized one like Booking.com. Figure 1.1 shows the complete search engine process as it was proposed by Manning et al. [21, pp. 146-147]. The process consists of the following 5 steps:

1) Gather documents (see figure 1.1) A spider (also named "crawler" or "robot") gathers documents from the web by following the links pointing to those documents or web pages. In the case of a general-purpose search engine, the spider starts at one particular web page and collects the outgoing links of this page. These collected links are then added to the spider’s "URL frontier", from which the next page link to crawl will be taken [21, p. 444]. Eventually the spider visits every link that it has found on the web. A vertical search engine concentrates only on a specific topic, therefore the links that have to be visited are preselected and limited at least to the domain level. The visited pages are being evaluated in order to gain information about their contents.
1.3. RELATED WORK

For the web servers, which deliver web pages, spiders are the same as normal users who access the page through a browser. The only difference is that spiders cannot interact with a page in the same way. Spiders are not able to type in a certain URL or to fill out a form in order to access some content. They can only discover pages by following links, which are pointing to those pages, and process content only, if it’s already there without having to submit a form. This is one of the reasons why there’s a huge part of the web that is not being crawled by search engine spiders and therefore cannot be found, unless the searcher knows where to look, meaning the exact URL of a page. This part is often called the Invisible Web or “deep Web”, because it cannot be retrieved by search tools, the most common way of information seeking among web users. For most content of the Invisible Web, however, spiders choose to exclude it based on their own policies. But sometimes it is just not possible for them to crawl the content. Sherman and Price [31] have extensively examined the various reasons why certain pages are not being crawled. The reasons are:

- **Restricted access**: There are three ways to keep spiders from crawling a web page: The webmaster can create a `robots.txt` file with files or directories listed in it that should not be crawled. The webmaster can also request the spider not to crawl a certain web page by specifying the “noindex” metatag in the “head” section of this page. It works the same way as the `robots.txt` but only for that specific page, whereas the `robots.txt` can prevent the entire website from being crawled. These two methods, however, do not guarantee that the spider won’t crawl the content. Nevertheless most search engine spiders respect the protocol and stay away from such restricted pages. The third way is to protect the page by a password, which is the safest method from being crawled, because the spider simply cannot get to the content. But the drawback of this method is that only users, who know the password, can access the page – unless it is intended to protect it from the public as well.

- **Dynamically generated content**: Web content gets generated increasingly on the fly, i.e. on request from the user. This happens mostly on submitting a form with the desired parameters, with which information from the underlying database is retrieved. Examples of such applications are library catalogs or statistical databases and all kinds of search services. Very often such databases contain highly valuable information. But since there is almost an infinite number of possibilities to fill out a form, spiders cannot efficiently access the content behind it and therefore it stays hidden from the search engine. Dynamically generated content often has a unique URL with the parameters given after the "?". Therefore such URLs can be provided as ready-to-use links without the need of form submission. This would open the way for spiders, but most of them are nevertheless ignoring such links. The reason is that malicious websites often provide dynamic URLs, which effectively point to the same content and trap the spider in an endless loop. But search engines are evolving and spiders are being programmed to learn how to correctly fill in forms and which dynamic pages are safe to be crawled. Still research has not progressed that far to overcome many of the difficulties.

- **Non-textual data**: Search engines deal almost entirely with text. Text is easily understandable compared to other forms of data such as images, audio or video material. Such forms of content can be made more understandable by parsing the text surrounding the content, like the filename, metadata or descriptive headings, tags and similar. But if a page contains little or no text at all, not even the descriptive text of the non-textual content is enough to gain high relevance in the search index. Other formats, which are not being crawled, are for example
executables (\texttt{.exe}) or compressed files (\texttt{.zip}, \texttt{.rar}, etc.). This is not because it’s not possible but rather because the limited interest in those contents does not justify the high effort to crawl them.

- \textbf{Real-time data:} Many websites exist that serve real-time information, such as weather updates, all sorts of live tickers or other fast-changing data. There is no point in crawling such information by traditional spiders because it would take up too many resources to stay up-to-date with the changes by crawling the pages repeatedly in a short amount of time. It is not feasible for the spiders to crawl a page more frequently due to the large amount of pages on the web. The Google search service deals with such content in its own way: for example a search for the current stock information of a certain company results in a compact overview of the requested data, which comes from a partner information provider.

- \textbf{Disconnected URLs:} A page, that has no incoming links, meaning links from other pages pointing to that page, cannot be discovered by the spider. Such pages are called "disconnected" and the only way to make them accessible to the spider, is to submit their URLs directly to the search engines.

- \textbf{Irrelevant content:} Spiders crawl according to their own algorithms. These algorithms decide, whether a certain page contains valuable data or is not worth crawling. Additionally, spiders usually do not follow all subpages of a website but only to a certain depth level, generally not further than six clicks down the site tree. Therefore a great number of pages does not get indexed at all only because of the spider algorithm, thus they remain invisible for the searcher. On the one hand this approach helps the searcher by filtering out irrelevant content, this way reducing the number of results, but on the other hand it can also hinder her from finding the needed information.

Content on the Invisible Web does not have to stay invisible forever. The search tool providers can always change their exclusionary policies as well as the webmasters can always decide to grant public access to the former hidden content. Also the techniques of the spiders are constantly evolving to deal with content that was not processable before. Therefore both, the visible and the Invisible Web are continually changing.

2) \textbf{Parse content (see figure 1.1)} When the content of the visited page is considered valuable, it has to be parsed in order to build an index. Parsing means splitting the content of the document in small preprocessed chunks, often single words or phrases of two or more words. These chunks are called tokens and are later used for matching with the user query.

The first step of parsing a document is \textit{tokenization}. In this process the text of the document gets split up into tokens, mostly into single words. During this step most of the punctuation is being omitted [21, p. 22]. Here is an example of a simple tokenization:

\begin{verbatim}
Some birds are sitting on the tree, tweeting cheerily.
\end{verbatim}

\begin{verbatim}
Some | birds | are | sitting | on | the | tree | tweeting | cheerily
\end{verbatim}

But simply cutting on whitespaces is not always a good idea, for example expressions such as "San Francisco" should not be split at all. The stripping of punctuation is also a tricky task, e.g. when it comes to phrases like "isn’t": "isn’t" is not a real word, but separating it to "isn" and "t" is by far worse. Tokenization algorithms have to deal with such exceptional cases.
Earlier versions of search engines filtered the generated tokens in order to remove very common words, e.g. "the" or "in", that have no specific meaning for the content. Such words are also called stop words. In most cases these words are redundant but sometimes they make a difference for the meaning. Therefore, and because processing time and data storage have become less of a problem in modern technology, most search algorithms don't discard stop words anymore [21, p. 27].

The next step of parsing a document is normalization. Some words or terms can be written differently, for example acronyms with optional use of periods (U.S.A. versus USA) or terms that can contain a hyphen (meta-data and metadata). Nevertheless the terms have the same meaning and should be found with either spelling. This also applies to synonyms such as "taxi" and "cab", which often have to be added manually because they cannot be generated automatically. But the largest differences come from the user’s query: in order to get results as fast as possible queries are typed completely in lowercase and diacritics are mostly discarded. There are different ways for dealing with various spelling options, which generally result in listing a term under all kinds of spellings in order to get a match independently from the writing style [21, pp. 28-30].

The last parsing step is stemming or lemmatization. This task aims to generalize the tokens to their base form (stem) in order to match different forms of a word with one query term, for example to match "various", "varies" and "variation" with the query term "vary". Stemming refers to a technique which cuts off the end of a word (e.g. "-sses" to "-ss") or replaces it with the base form ("are" to "be"), according to language-specific rules. The previous example sentence could result in the following stemmed tokens:

Some birds are sitting on the tree, tweeting cheerily.

```
some | bird | be | sit | on | the | tree | tweet | cheer
```

Mostly this method produces useful results, but sometimes too much of a word is being cut off, so that words with different meanings result in the same stem. Lemmatization, however, refers to a vocabulary and does a morphological analysis of the word, resulting in grammatically correct stems. Therefore it is more precise than stemming, but it also slows down the performance [21, pp. 32-34]. Different implementations of both techniques have been developed so far, trying to achieve the best results with minimum costs. It has to be decided depending on the situation, which algorithm meets the requirements best.

The same techniques and algorithms that are used to parse a document should also be used for parsing the user query, in order to get an identical token for the same term. This ensures that a term of a user query always matches the same term in a document.

3) Build document caches and indexes (see figure 1.1) After parsing the documents two things have to be done: Fill the document cache with a summary of the content and build indexes for the previously generated tokens. The purpose of the document cache is to present a snippet of the content to the user, when the document is returned in the results list. This short summary helps the user to assess the relevance of the content for her information needs without having to visit the website first. This way she has to spend less time analyzing the results and can easily compare them. There are various ways to extract the summary snippet from the content, with the purpose to make it most useful for the user [21, pp. 170-173]. The two basic concepts are:
• **Static generation:** Static means that the snippet is generated beforehand regardless of the user query. The summary text is always the same and gives an overview of the content as a whole. Typically it is extracted from the beginning of the document and from its metadata description. The resulting text snippet is then stored in the document cache at index time.

• **Dynamic generation:** Dynamic summaries on the other hand depend on the query and therefore are generated on the fly. The snippet shows the query terms in their context at some points of the document, in order to explain why the document might be relevant. To generate the snippet dynamically the document (or some parts of it, if the content is too large) has to be stored in the cache at index time and processed at query time. In order to be most useful, such *keyword-in-context* snippets have to be informative, self-contained and short enough to fit in the often limited length of the summary. This requires more computation at query time but is also much better in terms of usability.

Building indexes for the generated tokens means creating an (inverted) index, which maps each token to the documents where it occurs [21, pp. 6-9]. Starting with a list of pairs of a token and its associated document ID (a unique integer assigned during index construction) these pairs are first sorted alphabetically by the token. Multiple instances of the same token are then grouped into a dictionary and the result is a list of token-postings pairs, where postings are sorted lists of all document IDs in which the token occurs. This index structure reduces required storage space, because a token usually appears in more than one document. Here is an example indexing result:

**Doc 1:** The sun shines brightly.  **Doc 2:** The sun goes down.

<table>
<thead>
<tr>
<th>dictionary</th>
<th>postings</th>
</tr>
</thead>
<tbody>
<tr>
<td>bright</td>
<td>1</td>
</tr>
<tr>
<td>down</td>
<td>2</td>
</tr>
<tr>
<td>go</td>
<td>2</td>
</tr>
<tr>
<td>shine</td>
<td>1</td>
</tr>
<tr>
<td>sun</td>
<td>1,2</td>
</tr>
<tr>
<td>the</td>
<td>1,2</td>
</tr>
</tbody>
</table>

The dictionary can also contain additional information such as document frequency, which tells how many documents contain the token ( = length of the postings list). The postings on the other hand can store the term frequency, which is the number of token occurrences in each document. Such statistics are later used for ranking the results.

These indexing processes are called the indexers. For large data collections such as the web the indexers have to be distributed over computer clusters. Once all indexes are built, the entire index process starts again, because documents are changing constantly, therefore their indexes have to be updated regularly. It is possible to maintain several indexes at a time, e.g. the main (old) index and a new "auxiliary" index for new documents, which is merged into the main index every time it gets too large [21, p. 78]. However, large search engines typically reconstruct the index from scratch periodically, because a dynamic approach with multiple indexes would make almost all processes of the search engine more complex.
4) **Process the user query (see figure 1.1)** At this point, when the index is built, a user can make queries to the search engine. The query is first being parsed using the same methods as for parsing the crawled web pages. Often the search engine is intelligent enough to propose spell corrections for misspelled queries in order to help the user find the needed information, if she doesn’t know how the searched term is spelled correctly or if she simply mistyped the term without noticing it. The parsed and optionally spell-corrected tokens are then looked up in the dictionary of the index and the corresponding postings are returned [21, p. 49].

Spell correction, if it is done, is not a trivial task. Beside the complex algorithms, which are used to provide corrections for misspellings, the errors have to be detected first. This is especially difficult when the query terms are seen as isolated, meaning that each term is checked without regarding its context. Such isolated-term correction won’t detect the error in the query “master salve architecture”, in which “salve” should be corrected to “slave” but won’t be since “salve” in isolation is a correctly spelled word [21, pp. 57-58]. This is where context-sensitive spelling correction comes into play. When the search for such a query retrieves only few documents, the search engine could offer an alternative corrected query [21, p. 62]. Thereby all possible corrections of every term in the query are being considered, e.g. “faster” for ‘master” and ‘slave’ for ‘salve’. Then combinations of these corrections are checked and the most common are proposed to the user. Instead of computing the alternatives, the search engine could also just select the most common similar query from the query logs that contain all previous user queries.

5) **Rank and present the results (see figure 1.1)** The number of retrieved documents for the user query can be far too large to let the user evaluate all of them. Furthermore not every document is equally relevant for the user’s needs. Therefore the results have to be sorted by relevance, or ranked, to make the search for and evaluation of information easier for the user.

The general idea is that if a query term occurs more often in a document, then this document is more relevant than the others and should be ranked higher. Therefore the already mentioned term frequency is needed, which was calculated during the indexing process and is now used for weighting the document relevance [21, pp. 117-119]. The term frequency of a term \( t \) in a document \( d \) is denoted as \( tf_{t,d} \). But considering the term frequency alone is not sufficient. If a query contains very common words such as “the”, earlier mentioned as stop words, it is very likely that almost every document gets a high rank. This example shows that not every term is equally relevant for the document rank. Therefore the term frequency has to be relativized in order to properly weigh the document’s relevance. This is where the so called document frequency comes into play. The document frequency \( df_t \) is defined as the number of documents \( N \) in the collection (in our index) containing the specified term \( t \), so with growing document frequency the weight of the term frequency should be reduced. To scale the term frequency by the proper factor the inverse document frequency is used:

\[
\text{idf}_t = \log(N/df_t)
\]

This means that a high idf stands for a rare term and a low idf for a rather common term. Combining \( tf \) and \( idf \) results in a scaled weight of a term \( t \) in a document \( d \):

\[
\text{tf-idf}_{t,d} = tf_{t,d} \times idf_t
\]
The \( tf-idf_{t,d} \) is then

- **highest**: when a term occurs frequently in a small number of documents
- **lower**: when a term occurs less frequently in a document or occurs in a higher number of documents
- **lowest**: when a term occurs in every document

For a query \( q \) consisting of multiple terms the score of a document is calculated as the sum of the \( tf-idf \) weights of each term in the document:

\[
\text{score}(q,d) = \sum_{t \in q} \text{tf-idf}_{t,d}
\]

This score calculation is fundamental for various algorithms, which have to deal with ranking documents, but is by far not sufficient for addressing requirements such as the accepted distance of the query terms in the document, or the preservation of the term order (which is ignored by the \( tf-idf \) calculation), and also the authority and informational value of the document (consider spam web pages, which overload their content with a specific term only to rank higher in search results).

To address the latter issue one method that web search engines often use is the so-called **PageRank** [21, pp. 464-465], [5]. The idea behind it is that when many pages link to a specific page, this certain page is likely to be more authoritative than pages which have only a few incoming links. The quality of the pages that link to a specific page also influences the rank of that page: the higher the authority of the referring pages, the higher the score of the referred page. The **PageRank** of a page can have a value between 0 (when the page has no incoming links from the nodes in the web graph) and 1 (when all nodes of the web graph point to that page). Clearly it is not just a simple calculation. Search engines often have to deal with link spammers, which are a group of pages specifically designed to link to a certain page in order to boost its **PageRank**.

Along with the **PageRank** algorithm another approach has been developed: the so-called **Hyperlink-Induced Topic Search (HITS)** algorithm [21, pp. 474-475], [5]. It is assumed that web pages can be categorized in two types: hubs and authorities. Hub pages are pages, which link to several authority pages but are otherwise not relevant. Authority pages have many incoming links from hub pages (similar to the **PageRank** idea). For every page both scores (hub and authority) are computed. Resulting hub pages are then used to find highly relevant authority pages. Both scores are depending on each other: Pages with links to high authority pages get a high hub score and pages, which have incoming links from good hubs, get a high authority score.

The described scoring methods along with some other techniques are used to calculate a final score for the retrieved search results. The scored documents are then sorted by the highest rank and presented to the user.

This description of a search engine is only a short overview of the most important components and steps. It is by no means a complete representation of all the techniques and algorithms that are used in a modern search engine, but rather a roundup of the crucial concepts of computational search.
1.3. RELATED WORK

— DIFFERENCES BETWEEN GENERIC AND VERTICAL SEARCH —

After having discussed the concepts of a search engine in general, the differences between generic and vertical search are elaborated in this section in details. The most important difference is obvious:

- **Topic-limited results:** Whereas a generic search engine attempts to provide results from various domains and topics which have something to do with the search query, a vertical or focused search engine limits the results to only one topic. Such specialized search engines aim to offer only highly relevant content that is often not found at all using a generic search.

There are several ways of how to provide only topic-relevant results in a vertical search engine. The mostly used method is crawling. However, the crawler does not index the entire Web like a generic crawler, but instead only topic-relevant pages. To achieve this, the crawler is being controlled to either only visit preselected web domains or filter the pages based on their content [5].

Which of these approaches is more advantageous for a certain vertical search engine depends on the case, because both have drawbacks. With the first approach new or unknown relevant web pages won’t be found, while the second method is quite inefficient because it requires a full web crawling and a good filtering algorithm (e.g. based on a domain-specific terminology). Several such filtering or crawling algorithms were developed in the past, many of them aimed to get the best results with efficient crawling by combining content-based and link-based analysis of the pages [5, 1]. Content-based analysis means that the text of the visited page is being checked for important topic-related words, whereas link-based analysis considers the web link structure, taking into account three criteria: the anchor text of the link pointing to a specific page, the authority of the link provider and the number of incoming links to that page (e.g. the PageRank algorithm).

Other methods beside crawling are meta-searching and crawling at query time [34]. Meta-search means to search for a query in databases of several websites and show the combined results. While this is an easy way to extend the search coverage, such a system is quite difficult to extend and maintain due to differences and frequent changes in the interfaces of the used websites. Crawling at query time means to search in real-time when no results are found in the index. This is not feasible for generic search engines because there are too many pages to crawl. But a vertical search engine could do that, when there is a limited number of pages to visit.

- **More detailed information:** Another difference of the two search engine types is that a vertical search explores websites more deeply and that way provides further information, which is not searchable by a generic search engine because it crawls websites only to a certain level [34].

- **More up-to-date content:** Vertical search engines also provide more up-to-date information because they can crawl the specific, rather small, part of the web more frequently. This feature is particularly useful for frequently changing topics like news [34].

- **Topic-specific functionalities:** A major advantage of vertical search engines is their ability to provide more structured results with additional information and advanced functionalities to explore the content. Because of the deep knowledge of the topic and the indexed pages such search engines can offer specialized interfaces with more functionalities such as topic-specific filters and categories [34].
The main hurdle of vertical search engines is to be found by the user. Their large and steadily growing number makes it difficult to find an appropriate search platform that examines its topic completely and provides highly relevant results. But once it is found and its usefulness is proven, the user will always return to its service.

Both types of search engines have their strengths: whereas a generic search is optimal to find some facts or general information in a very short time, such as looking something up in an encyclopedia, a vertical search provides more in-depth specific knowledge that is highly relevant and might not be found otherwise.

### 1.3.2 Design Guidelines for Search Applications

There are many design guidelines for user interfaces of all kinds. The purpose of these guidelines is to create an application with high usability and therefore an enjoyable experience for the user. Usability is very important because the user always comes first. Without the user there is no need for an application and its interface.

An important factor of a good usability is the learnability of the system, as Russel-Rose et al. put it [28, p. 11]. When the system is easy to understand, then the user will quickly learn all the functionalities of it and become an expert, so that she can fulfill her task in a shorter time. The learnability of an application can be enhanced through various methods, ranging from unambiguous instructions and descriptions of actions, e.g. a descriptive placeholder in an input field, to animations, which draw attention to helpful actions. Given a good learnability, the other four components of usability, defined by Shneiderman and Plaisant [33], are also easily achieved:

The better the learnability,

- the more efficient the use
- the easier the memorability of the functionalities
- the less errors are made
- and the higher the user’s satisfaction.

Apart from having a learnable user interface, a search application shall support the users at different stages of their task, as stated by Hearst [13, p. 1]:

"The job of the search user interface is to aid users in the expression of their information needs, in the formulation of their queries, in the understanding of their search results, and in keeping track of the progress of their information seeking efforts."

This reflects the main steps of a search application, which should be maximally optimized for usability: query specification, assessment of the search results, analysis of the most appealing results in detail and an optional new search through query reformulation. These basic steps have also been described in several models of information seeking. The most known is the standard model based on findings of various authors ([29, 22, 32, 35]) and specified by Broder [4]. It describes a cycle of identifying the information need, specifying the query, evaluating the results and, if needed, reformulating the query followed by a new search.
1.3. RELATED WORK

Other models exist, which go further than this simple process. For example the dynamic model (also known as the berry-picking model) proposed by Bates [2] points out that the information need often doesn’t remain static but rather changes throughout the search process, as new insights and knowledge are gathered from previous search results. But these assumptions don’t change the basic steps described earlier: Regardless of the nature of the information need the basic search process consists of querying a search engine and analyzing the retrieved results. Therefore in this section some important design guidelines for the steps of information seeking are described, and how they are implemented in the IROM search application.

— Querying —

Every search process starts with formulating a search query. Already at this step the user can be supported by an intelligent user interface. First of all it should be made clear to the user, what kind of query to type in the search box. This can be achieved by a descriptive placeholder in the search input box, which indicates what results can be expected after submitting the search query [13, p. 18]. The placeholder on The Movie Database (https://www.themoviedb.org/) for example states clearly that the user can “search for a movie, tv show, person” on their website. Especially vertical search engines like that offer a great opportunity for distinct placeholders as they are topic-limited and therefore the placeholder text can be more specific. Such a simple feature can increase the user’s confidence and result in a more effective search [28, p. 11]. To follow this guideline the search box of the IROM application has got the placeholder text “What topic are you interested in?”.

Another way to support the searcher at this step is to offer query suggestions dynamically as the user types in the search box [13, pp. 11-13]. For example Youtube shows a box with auto-completed queries as the user types in the search input field. Such suggestions can come from an internal database containing the titles of the items that can be searched, like on The Movie Database. Google’s search suggestions on the other hand are mostly based on frequent queries already made by other users (collective behavior) together with the user’s individual search history [28, p. 111]. The benefits of this functionality, as stated by Russel-Rose et al. [28, p. 109] and experimentally confirmed by White et al. [37], are mainly to reduce the user’s mental workload to remember how some term is spelled correctly (following the principle of recognition over recall in human-computer interaction), especially when it is a difficult name of e.g. some famous person, and to save her the effort of typing the entire query term.

A special case of autosuggestion is autocorrection, where the system detects a spelling mistake and suggests a correction automatically. Spell correction can also be offered in another way: If the misspelled query results in too few matches or the system believes that the term is misspelled, the results can be enhanced by matches for the spell-corrected query [13, pp. 17-18], [28, pp.115-118]. It is furthermore a common practice to show a “Did you mean...” tip with a direct link to the corrected query in order to get only meaningful results. This feature is particularly helpful for the user to understand the cause of a “No results” search answer. Dynamic query term suggestions provide effective feedback and, as Hearst puts it, “are a promising intermediate solution between requiring the user to think of terms of interest (and how to spell them) and navigating a long list of term suggestions” [13, p. 12]. In the IROM application the autocomplete feature derives its suggestions from the existing course titles as well as from collective user queries (as soon as there has been gathered enough user data).
Less important but also relevant things to keep in mind, when designing a search platform, are rather small details of the application design. Such minor things are mostly not noticed directly, but they can make a huge difference in the user experience. Franzen and Karlgren for example found out that a wider search input field encourages users to formulate a longer and more specified query and therefore get better matches from the system [9]. Furthermore it is also appreciated by users to see their search query on the results page together with the number of search results. Showing the query reduces the searcher’s mental workload to recall his query, and keeping it in the search box makes it easier to reformulate the original query during the search process loop described earlier [28, p. 101]. All these details are implemented in the IROM search application.

--- SERPs ---

After submitting the search query the user gets a list of results for her query. This view is often called a Search engine result page (SERP). Here many design aspects have to be considered.

**List vs. grid interface**

First of all it is important to decide, how the results should be arranged on the page. There are two main methods for how to represent the search results: either using a one-dimensional rank-ordered list view such as the Google search engine and various other general-purpose search engines, or using a two-dimensional grid view with multiple rows and columns, which is highly preferred for example by search engines for images (e.g. Pinterest (https://www.pinterest.com/), or the Google image search) or the presentation of articles in online shopping portals such as Etsy (https://www.etsy.com/).

Kammerer and Gerjets [16] analyzed the effects which the different visualizations have on user perception. They found out that with a list interface users focused significantly longer on the top results of the list and mostly ignored the results at the bottom. This confirmed earlier research outcomes according to which result lists are generally consumed in a Golden Triangle or an F-shape order [24, 27, 10] and mostly only the top results are being selected [11, 7]. The reason for such behavior is that users trust the ranking algorithm of the search engine so much that they consider only the first few results to be relevant for their search query. Even reordering the results in an ascending order, in which the most relevant result is at the bottom of the list, did not change this behavior, showing how much users rely on the ranking algorithm of the search engine [14, 17, 26].

Representing SERPs in a grid interface, however, led to a wider spread focus on the results with equal focusing times for nearly all result items. This is because such a representation implies that the results are not strictly ordered by rank and can be equally important. Thus a grid interface reduces the influence of result position on the user’s perception, enhances her willingness to explore more items ([28, pp. 141-142]) and encourages her to judge their relevance by own understanding rather than blindly trust the ranking algorithm of the search engine. Also a grid interface usually can show more results above the fold (that is the visible area of a page without scrolling) than a list interface, as long as each result has a reasonable size. This way more results can be examined without the need to scroll down.
However, this does not mean that a list interface is a bad choice. The right representation of SERPs depends on the situation. When searching for a simple fact a list interface is most efficient because it helps the user to find the answer very quickly without the need to examine several results. But a more complex search task, e.g. the pros and cons of a controversial subject such as the right medical treatment method, requires from the user to gain transparency in the subject and therefore examine several opinions from different sources. In such a complex search task it would definitely not be enough to only read the first two results from the company that invented this specific treatment method.

Regarding the presentation of MOOC search results, the user should be encouraged to explore the various possible courses and not to automatically pick the first result. This way she gains more from the search engine and possibly encounters appropriate topics of which she had not thought before. This is the ulterior motive behind arranging the IRROM SERPs in a grid structure.

**Result information**

Having discussed the arrangement of results on a SERP in the application, the next step to think about is, which information should be displayed for every result item. This depends on the type of search the application is intended for: A search for some minor information like the birthday of a famous person differs greatly from a search for the right refrigerator to buy. In his taxonomy of web search Broder defined three different types of search tasks [4]:

- **Navigational**: searching for a particular web page
- **Informational**: searching for some general information (often on multiple websites)
- **Transactional**: intending to perform some activity, e.g. purchase or download the desired material.

Granka et al. [10] as well as Cutrell and Guan [7] conducted eye-tracking studies of online search and found out that users scan search results differently in each search type. For informational and transactional search tasks users focus more on the text summary of items and less on their source. Also adding information to the summary snippet improved performance for informational tasks in Cutrell’s and Guan’s study. Therefore title and summary of a result are the most important elements in informational search that help the searcher to judge their relevance, and thus should be clear and descriptive [28, p. 29].

According to Hearst, “[e]ven the most relevant document is unlikely to be selected if the title is uninformative or misleading” [13, p. 120]. The importance of a descriptive summary is explained by the fact that users evaluate the relevance of an item by the given information before they decide to go to this item and examine it in more detail [26].

To make it easier for the user to scan the results the title of each item should be displayed prominently. Furthermore the summary text, which is typically an excerpt or description of the item, should preferably contain the query words (if not already present in the result title) in order to explain, why this particular result matters for the user’s search [6]. Such query-oriented summaries are also called keyword-in-context (KWIC) extractions [13, pp. 122-124] because just those particular fragments of the document, which contain the keywords, are picked for the summary. To help the searcher evaluate the context and proximity of the keywords at a glance, they also should be highlighted to produce an eye-catching effect [13, pp. 8-9, 128], [28, p. 30]. Example techniques for effective highlighting are a bold font style, a different font color or a colored background behind the word. Such KWIC excerpts have become the de facto standard on SERPs of general search applications.
In the IROM application each result is represented with a picture taken from the course website during the crawl and the title beneath it. Having pictures for the results has the benefit of refinding a previously discovered item more quickly than only by its title [28, p. 132]. Images also act as eye-catchers and bring diversity into the otherwise monotonous look of the items. To save space and not overwhelm the user with too much text, the description of a course result is initially hidden and can be easily shown by hovering over (or tapping on when using a touch device) the information icon beside the course title.

Use of categories

When there is a category system on the website, it can be helpful to group the results by their category. This way the user can scan the SERP more rapidly and gets a better idea about the affiliation of an item to a specific topic [28, p. 30]. Dumais et al. experimented with different ways of showing the category of a result. From their findings they concluded that showing results grouped by categories led to a better search performance than showing grouped results without category labels and much better than showing a regular, not grouped list of results with just the category name beneath each item [8]. The search performance was measured by the time a participant needed to find the answer and the large time differences in each representation were artificially induced by using very ambiguous queries for specific search questions, according to Hearst [13, p. 179].

A similar study conducted by Kules and Shneiderman revealed that organizing results in categories for more realistic searches had not that much impact on the searchers’ performance [18]. However, results grouping was highly appreciated by participants and led to a better exploration of the results. The idea of grouped search results by category can be seized in the IROM application as soon as the crawled course data contains information about categories.

Furthermore categories can be used to give the searcher the ability to filter items by a category of interest, this way providing a browsing functionality in the search application [13, pp. 20, 181-182]. Filtering by category means retrieving only those items from the entire item set that meet the filtering criteria. This way the searcher can focus just on items of interesting categories and does not get distracted by non-relevant results. With filters also a scoped search can be provided by the system, enabling the user not only to retrieve category-specific results from the entire set, but also searching with specific keywords inside the category-specific subset.

Beside filtering a common practice is to sort the search results either by categories (which makes sense only when the categories are ordinal or metrical), or based on more general criteria such as price or date [28, p. 152]. These criteria depend heavily on the type of results and must be picked individually for every search engine. The sorting functionality, as well as filtering, is seen as an effective way of direct user control over the result set and is recommended by Hearst, because it makes the system more transparent, understandable and controllable for the searcher [13, pp.10, 14-15].

Empty results page

A case, which shouldn’t be left out, is a SERP with no results. In this situation it is important to explicitly point out that the query provided no results, and to explain to the user why there are no matches and how she should proceed. This is a good opportunity to offer related search queries, thus showing a way out of the dead end [28, pp. 120-121, 148].
1.3. RELATED WORK

But an empty SERP should be avoided as far as possible [13, p. 24]. Users often over-constrain the search application with too long queries or select too many filters, which just don’t match anything [28, p. 119]. Here partial matches can help. It should be made clear to the searcher that her current query or filters don’t provide any results, and then results for partial matches should be offered, preferably with a hint, which query word has been omitted to match a particular result.

If there are still no results to show, the empty space can be used to promote certain items, e.g. the most popular or newly added items [28, p. 148]. The search engine which is used in the IROM project currently does not support partial matches and always returns a specified number of results sorted by highest relevance. This missing feature is part of the list of future improvements.

Pagination

To ensure an enjoyable user experience the platform should be maximally fast, particularly in showing the results. If the search engine cannot yield the results immediately, even if it is only a delay of milliseconds, a loading hint of some kind, typically a graphical animation, should be displayed to counteract user’s impatience and ensure her that the system is working on the query and will respond shortly [13, p. 14]. But such a loading hint won’t be of much help, if the SERP needs too long to render the results. Therefore often pagination or infinite scroll are used that load and show the results in chunks.

Both types have their advantages and drawbacks. A classic pagination helps the user to keep track of how far she has come and how many results or pages are still left to discover [28, p. 150]. This is the disadvantage of the infinite scroll method: The user has no idea of how long she has to scroll to view all the results, which is often frustrating. But this method outperforms standard pagination with regard to the “first page problem”: The majority of searchers doesn’t go further than the first results page but rather reformulates the query [14, 26]. This behavior hinders the exploration of results. With infinite scroll on the other side the searcher often loses herself in the process of scrolling and explores the results further and further. The system’s ranking of the results has also a minor impact on the user’s own perception because there is just no “Page 2” with “less relevant” results using infinite scroll. And lastly this type offers a more seamless experience and eliminates the need of distracting page reloads [28, p. 151].

To lessen the annoying effect of drowning in the amount of results with endless scroll it is helpful to show the number of results directly at the top of a SERP. But this information is also helpful for the searcher to assess the quality of her query: When too many results are returned, then the query was too ambiguous, and vice versa. This knowledge helps her reformulating the query and get better results [28, p. 139].

Due to the currently rather small number of courses that have to be searched, the IROM application is fast enough without the use of pagination or infinite scrolling. Besides, the search engine currently does not support paginated results. As soon as there are more courses in the database one of the possible pagination methods must be implemented. To find out which method works best for this specific platform, A/B testing could be performed to analyze the effects on user behavior of both pagination types.
**CHAPTER 1. INTRODUCTION**

**Saving results**

Another important feature not only on SERPs but also on any listing of items (e.g. when browsing on category pages) is the ability to save a specific item for later. When browsing many items, users can find several relevant results but don’t have the time to explore them all in detail. This is where such a saving feature comes in handy: a logged in user can save a specific item to find it easily later on and view its details or act on it, when she needs it or finds the time for it. For example, a student searches from her smartphone for a topic and finds some interesting course during the tube ride on her way to a lecture. She decides to enroll in the course later when she is at home sitting comfortably in front of her laptop and just clicks the save button of the course. Being then at home she logs in to her IROM account and finds the saved course instantly in her personal "Your saved courses" list.

---DETAILS---

When the searcher finally finds an interesting item in the results and clicks on it, in vertical search applications she mostly won’t land directly on the external source website of that item, like it is with general search platforms such as Google, but rather on an internal page of the search application that shows details of the item and has a link to the external source website. There are some reasons for such behavior. First of all many vertical search engines have multiple sources for the same item, e.g. price comparison systems. Another reason is that the vertical search application provides the ability to directly act on a found item, like booking a found hotel directly from the search website.

**Content- and user-based recommendations**

In the case of IROM an extra detail page offers the opportunity to further satisfy the searcher’s information need by showing related items to the one selected. This way the user can discover items, which she haven’t thought of or just didn’t know about. Such recommendations of items can have various sources [13, p. 212]. The most common kind are content-based suggestions, based on similarity of the items, which is often calculated by applying machine learning algorithms and clustering. Another way of finding related items that gains popularity is through Collaborative Filtering (social recommender systems) [13, pp. 218-219] resulting in user-based recommendations. The idea behind this algorithm is to suggest items that like-minded users with similar preferences also found interesting. User-based suggestions seem to work well for items with an aspect of taste, such as movies, music and books. For satisfying a searcher’s information need it makes more sense to make content-based recommendations.

The drawback of purely content-based suggestions is that such a filtering can overspecialize [30] and return only suggestions, which exactly match some predefined features. This can lead to a very narrow recommendation set that does not contain other similar items, which just have another wording in their description, e.g. synonyms. Collaborative Filtering on the other hand can result in unexpected items, give the searcher new ideas and encourage further exploration of the subject.

To benefit from both types of recommendations, content-based suggestions can be enhanced by users’ knowledge, such a method is known as item-based collective recommendations [13, pp. 220-221]. When the user views a certain item, she gets suggestions for
items, which were also preferred by other users who preferred the current item. Such preferences can have the form of ratings (which is an explicit sign of approval), purchases, or just views and a longer focus on an item (which are implicit signs of approval with no need for the user to do more than just using the web application, whereby such strong activities as purchasing are heavier indicators for approval than simple viewing of an item).

A famous example of this technique can be found on Amazon.com (https://www.amazon.com/) and is known as Item-to-Item Collaborative Filtering: “Rather than matching the user to similar customers, item-to-item collaborative filtering matches each of the user’s purchased and rated items to similar items, then combines those similar items into a recommendation list” [20]. These similar items are found by filtering items that are often purchased together. This recommendation feature can be seen on Amazon.com under the slogan "Customers who bought this item also bought".

**Details page in IROM**

In the IROM application two kinds of recommendations are shown to the user (on the detail page): the first one is similar to Amazon’s “also bought” recommendations, which offers the user courses that were also watched by other users (in this case regardless of the similarity between the courses). The data for this recommendation type comes from the users’ visit logs. The second type of recommendations comes from the content-based recommendation engine developed by Joseph Birkner in the course of his bachelor’s thesis [3]. This way IROM offers both types of recommendations and can provide the user with the best suggestions.

The usefulness of recommendations in general was analyzed by Lin et al. on a web platform for medical publications [19]. They studied the query log data of a week and found out that in over 18% of sessions users followed links to suggested articles and having done that once, continued to do so in more than 40% of the cases. Together with the fact that many prominent web sites such as Google.com, Amazon.com and Youtube.com show recommendations of some kind, this is a strong evidence for their usefulness and acceptance.

Another benefit of an extra page for a result item in the IROM application is the ability to track users’ actions such as clicks on recommendations and time spent on the details page. This indirect user feedback helps to measure users’ satisfaction with the application and to offer good collective recommendations. Through the evaluation of the users’ click-through data the system can gather valuable information to further improve the recommendation algorithm, not only based on item similarity but also on users’ preferences.

— **Search vs. Browsing** —

Often search is not as simple as submitting a query to the search engine and promptly receiving the right answer. In many cases the searcher needs to learn about a specific topic, having only a vague knowledge of it. This is especially the case with searching for MOOCs. To support the user in his knowledge seeking process the search application can offer an overview of existing items categorized by their general domains, without the user having to come up with an (uncertain and probably way too general) initial query [28, p. 38].
Such a segmented listing helps the user to explore the field and get an idea about how widespread a domain is, thus finding unexpected content and understanding, which components are especially important for her [13, p. 174].

A browsing option also supports the earlier mentioned psychological principle of recognition over recall: In the same way as autosuggestions help the searcher to remember a specific keyword for her query, the browsing feature shows her possible labels and keywords, so that she doesn’t have to come up with the matching terms by herself [13, p. 74].

In the IROM application the browsing feature is offered as a hidden filtering menu on every page that can be revealed by clicking a filter button in the upper left corner. The opened sidepanel lists the main and sub-categories of MOOCs that are currently available. Clicking on such a category leads to a results page showing all related courses.

— **User history** —

Another important aspect for user support is saving the individual search history, because quite frequently searchers don’t look for new information but rather try to re-find some source found in an earlier search process [13, p. 87]. Teevan et al. conducted a study, in which they observed re-finding behavior through queries submitted to the Yahoo! search engine over the course of a year [36]. They reported that in 40 % of cases users clicked on earlier viewed results. As stated earlier, recognition is easier for humans than recall [13, p. 19], therefore every application, which tries to offer an interface with good usability, should follow this principle and provide recognition cues wherever appropriate.

Samples of this guideline can be found in every modern user application. For example Amazon.com shows a user her recently viewed items directly on the homepage in case she wants to review them [28, p. 79]. The online retailer also provides, like most other e-commerce platforms, a user’s shopping history, showing her purchased items, so that she can easily rebuy an item if needed or recommend it to a friend. Another well known example is the browser itself: As soon as you type something in the address bar, a drop-down with matches is displayed directly beneath the field, coming from the user’s history of visited web pages.

In search applications the easiest way to retrieve earlier searches is to record the search history and provide the user access to these records [13, p. 162]. In IROM the searcher is being supported by saving all her visited courses and showing them in a separate area of her IROM account labeled as "Your viewed courses". This view contains all previously viewed MOOCs sorted by the date of last visit.

— **General design guidelines** —

This section describes some minor design guidelines, specifically for search applications as well as for web pages in general. The aspects discussed further are a prominent design of the main function of the application, the importance of a fully responsive layout of the user interface, as well as common Gestalt principles and the value of aesthetics.
1.3. RELATED WORK

Prominent design of the main function

First of all, each user interface should bring the user’s focus directly to that element, which is the most important in the interface. Therefore, if the main functionality of an application is to provide a search interface, then the search bar should be designed most prominently. The most extreme example for this rule gives the Google search site: The homepage of the search engine is held strongly minimalistic showing only the search bar in the center of the page and two buttons for submitting the search query (one of them a shortcut to get directly to the first result skipping Google’s results page).

Following this guideline the IROM application preserves almost the half of the viewport (i.e., the area above the fold) on the homepage for the search bar, positioning it in the center of an eye-catching background image, like it is done on many modern (vertical) search websites. Furthermore the search bar is placed at the top of every other page and stays there in a fixed position, so that when the user scrolls down e.g. to view more results on the results page and decides to reformulate her query, she doesn’t have to scroll all the way up, but has only to look at the top of the viewport and place the cursor into the input field.

Another important feature to enhance usability is to activate the focus on the search bar on page load [28, p. 101], so that the searcher can directly type in her query without the need to click in the input field first.

Responsiveness

A functionality that is expected of every web page nowadays is a responsive interface. This means that the content of a web page scales to the size of the viewport on all possible devices, be it a large desktop monitor or a rather small smartphone display.

Optimizing for mobile users is extremely important because their market share continually rises. According to the research sector GlobalStats of the independent web analytics company StatCounter, “Internet usage by mobile and tablet devices exceeded desktop worldwide for the first time in October [2016]”. Mobile and tablet devices then accounted for 51.3 % of Internet usage worldwide, compared to 48.7 % by desktop. Since then the gap between smartphone and desktop web usage widens steadily in favor of smartphones with 52.64 % alone (tablet usage remaining rather steadily by 4.62 %), whereas desktop usage decreased to 42.75 % as of August 2017.

“With the ubiquitous presence of mobile devices, anytime and anywhere access to the information world has become a reality” [26], therefore it is essential for every modern website to meet the trend and offer an optimized mobile view of their application. The IROM search platform was tested on the most common devices for large, medium and small screen sizes, to guarantee the responsiveness of the different interfaces.

---

6 For more details on the I’m Feeling Lucky functionality please refer to https://www.lifewire.com/im-feeling-lucky-button-1616813
CHAPTER 1. INTRODUCTION

Gestalt principles

Like in design of all things web interfaces should be organized following the Gestalt principles, which describe the natural perceptional patterns of humans. The most important principles are those of similarity and proximity [13, p. 237]. They state that elements, which look similar in terms of color, shape, size and orientation, and which are placed in smaller distance to each other, are perceived as belonging to the same group.

Most obvious aspects that follow these principles are e.g. a consistent use of a color scheme and similar shapes throughout the design of a website: All pages of a website usually use the same base colors for important elements such as action buttons or headlines, the same font family and other styles, generally speaking the same theme.

This is realized across the entire IROM application. Four base colors have been chosen for the theme of the application (compare to figure 1.2): a light Blue as the "corporal identity" color, which is the same as the color of the IROM logo, a light Green for small action buttons and highlights of other elements, as well as a prominent Red for elements signaling importance, used for error messages, the feedback button and the "Go to course" button, and a dark Brown for the footer and the info button on a course result. Furthermore some (not fifty) shades of Grey were included, e.g. a light one for the header bar and course descriptions background and a darker one for the IROM logo.

Figure 1.2: Main colors of the IROM search application (screenshot of Adobe Color CC)

These theme colors as well as the same font family is used everywhere in the application to ensure a consistent layout and improve the learnability of the interface. Another example for the similarity principle is the same styling of result items and the same layout for displaying many items on one page, e.g. on the SERPs, on category pages, on the "Your saved courses" page and on the "Your viewed courses" page.

Common examples for the proximity principle are text labels for input fields which are placed in direct neighborhood and are therefore perceived as belonging to each other, or the arrangement of pictures near the corresponding text paragraphs. Beside these obvious styles this principle is applied on the arrangement of recommended items on the
1.3. RELATED WORK

details page, where two different recommendation algorithms are being used: The items of each recommendation type are clearly distinguishable by positioning them in two separate groups, divided by an explicit label for the respective type.

Aesthetics

Last but not least an application should look highly appealing by choosing the right arrangement of elements as well as harmonizing colors and shapes. Hearst emphasizes the importance of aesthetics in design, referring to several studies conducted by different researchers, which confirmed that a well designed interface improves the task efficiency of an application and leads to higher user acceptance regarding subjective quality and usability judgments [13, pp. 26-28].

The IROM theme colors (compare to figure 1.2) have been chosen carefully to ensure their harmonization by using the Adobe Color CC (https://color.adobe.com/) tool (formerly known as Adobe Kuler). The entire design of the application has been kept rather simple and modern, applying the guidelines of Material Design (https://material.io/) provided by Google in 2014 and (beside the Google products itself) successfully used by such prominent websites as WhatsApp Web (https://web.whatsapp.com/) and Airbnb (https://www.airbnb.com/). The look and feel of the interface is further being made more vivid by integration of images and subtle animation effects.
2 CONCEPTION

2.1 OVERVIEW OF THE BACKEND STRUCTURE

This section illustrates the functionalities and the operating processes of the IROM backend, before discussing the frontend side of the application in section 2.2. First the individual components (such as the Search engine and the Backend Server) and their connections among each other are shown, to explain the data origin and data flow. Then the structure of the database scheme is discussed in more detail to clarify the relations and dependencies of the various object types such as User and PageView.

2.1.1 SEARCH ENGINE COMPONENTS

Figure 2.1 shows all the components of the IROM search application that are responsible for the data flow. These components have the following functionalities:

1) Web App Client The web app client is the user interface of the IROM search application. Through this rich client the user can interact with the system: search for MOOCs, get useful recommendations and bookmark or save specific courses for later. The web platform is described in more detail in section 2.2. It is written with the Angular 2 frontend framework which is generally used with TypeScript, a programming language based on the backward compatible ECMAScript 6 Standard (the successor of JavaScript).

2) Database The MongoDB database, a document-based NoSQL type of database, contains all the relevant application data – regarding the courses, the user and the tracking (for later analysis and evaluation). The database schema with the different collections is described in section 2.1.2 in more detail.

3) Search and Recommendation engines The third important component of the system are the different search and recommendation engines. These were mostly developed in the course of several theses of the IROM project and are a major part of the project idea. The task distribution is planned as follows:
   - Search: The search functionality is handled by the Fulltext search engine, based on Apache Solr, a search platform that uses the Boolean retrieval method.
Recommendation: The recommendations are provided by the other three engines, namely the Rank 1 Recommendation engine based on the average Word2Vec method, the Rank 2 Recommendation based on a sequence-to-sequence autoencoder and the third engine that works with cluster-based similarity. Which of the three engines is being used shall be determined randomly, to test and evaluate their functionality and efficiency.

The different engines can be reached over a REST API, a standardized communication method for distributed systems. Therefore the selection of an engine is as simple as calling its specific URL, so that the different recommendation solutions can be easily tested.

4) Backend Server The center of the system is the Backend Server. It is built with the Play Framework, which is based on the programming languages Scala and Java. This server is responsible for the communication with the web client, the database and the different search and recommendation engines. The latter two endpoints deliver the data requested by the client and transferred by the server. Functioning as a terminal for the three endpoints the server shall parse incoming information and adapt it to a version that is readable by the individual endpoints. Therefore it is a transmitter and translator of data for the different nodes.
2.1. OVERVIEW OF THE BACKEND STRUCTURE

Having this overview of the four main components, the work flows in which the individual nodes are involved can be described further.

**Searching for MOOCs**

Through the client a search query is sent to the backend server along with certain metadata (discussed in section 2.3). The server parses the client’s request and sends a further request to the search engine. The search engine responds with a set of course IDs to the server, which then gets all course details for these IDs from the database and sends the complete data set to the web client. The web client renders the received search results and the user can view the individual courses fitting to her search query together with detailed information.

**Bookmarking MOOCs and finding previously viewed courses**

The application provides the ability to bookmark interesting courses, so they can be easily found later. The client sends a request with the ID of the pinned course along with the user ID to the server, which then saves the course pin into the database. When the user requests all her saved courses later on, the server retrieves them from the database and sends the information to the client.

A further feature of the application is to access one’s previously visited courses. For this the client requests all viewed courses of a user, the server retrieves them again from the database and sends them back to the client.

For these two features the application needs to identify a user. This is achieved by letting a user log in to the application, thus saving her as a unique person. The saved or viewed courses are then mapped to the respective user by her ID.

**Getting recommended MOOCs**

Recommended courses are shown on the detail page of a course. Below the course description and further infos two types of recommendations are presented to the user: First, which courses other users also viewed, and second, which courses are similar to the one being currently viewed.

The first type is based on the previously viewed courses of users, who also visited this particular course. So the procedure is again to retrieve relevant data from the database over the server communication.

For the second type the recommendation engines are used. Similar to the search functionality the client requests recommended MOOCs based on the certain course and the server sends the request in a RESTful service compliant way to the recommendation engine. The engine responds with matching course IDs, based on which the server retrieves course details from the database and sends them back to the client.

**Tracking interactions**

As mentioned earlier, the database also contains tracking information gathered throughout the interaction with the application. Certain actions of a user are being registered by the client and sent to the server, which then saves the received metadata in the database. Details on this tracking functionality are outlined in section 2.3.
2.1.2 Data scheme

The models represented in figure 2.2 are defined as classes on the server as well as in the client, and as collections in the database. The relations of the different models of this structure are built with the respective IDs, such as the CourseID field of the CourseView pointing to the related ID field of the specific Course. These relations are visualized as arrows between the models.

The User model consists of a universally unique ID (also called UUID), an Email field and a GoogleID. The latter comes from the login functionality, which is currently solved through Google Login. A repeatedly logged in user can be matched with the respective database entry by her GoogleID. The login mechanism can be enhanced by other methods, e.g. through Keycloak. For this a new field for the Keycloak identifier would have to be added to the User model to recognize returning users.

The PageView model represents a visit of the application, similar to a session. This means that browsing the subpages of the web app does not generate a set of PageViews but only one, for the entire session of the current user, starting with the visit of the first app page and ending when the user leaves the application. A PageView has also a unique ID, along with the UserID if the visitor is logged in to the application, an IPAddress field for tracking purposes and a Timestamp field to be able to find the latest visit of a certain user.

The Course model contains all detailed information about a specific course. The fields are the same as defined by the crawler and represent a complete course object as retrieved from the source. This information is used to present a course in the application, on the overview pages like the search results as well as on the course detail page. Some of the shown fields are not filled yet because there is currently no updated course data, therefore such information as the category cannot be used yet.
The **CourseView** model is similar to the **PageView**. A new entry is created every time a course detail page is visited. The object contains a unique identifier (created by the database, hence it has the type `ObjectId`), the current `PageViewID`, the respective `CourseID`, a `Timestamp` and optionally a `SearchQueryID` (defined by the Query model). The latter field is being set, when the user comes from a `SERP`. This information also belongs to the tracking mechanism and is intended for a later analysis of a search query and the clicked results. If the user comes to a course detail page from the recommendation section of another course, the fields `ReferredCourseID` and `RecommendationType` are filled with the respective values. Furthermore it is being tracked through the field `OriginRequested`, when the user clicks out to the external course provider.

The **Query** model is used to save all search queries made by the users of the application. The purpose of these data in the first place is to analyze the interactions of the users and relate queries to the clicked search results. But the information has great potential to do much more: The autocomplete functionality of the search bar could be enhanced by similar queries of other users, by most popular queries or simply by those already made by the current user, thus showing her past queries. Such history-driven suggestions are used for example by common browsers in the url input mask or by smartphones within the typing keyboard. Each query is being saved uniquely, so that the same query string written at different times or by different users is saved individually. The reason for this is to be able to distinguish between users as well as between different instances of time, so that there is room for further processing of the data.

The **CoursePin** model is created for providing the ability for a user to save a specific course for later. Beside the standard identifier it consists of a `CourseID`, a `UserID` and a `Timestamp`. Like the CourseView, this extra model is needed because a Course can be saved by multiple Users, as well as a User can save multiple Courses. Therefore the `Timestamp` field applies only to an individual relation of one course to one user. The necessity of this field is justified by the fact, that the bookmarked courses should be presented to the user sorted from latest to the earliest pin. Furthermore the model has two additional fields called `ReferrerUrl` and `Specifier`. These indicate, on which page type the user saved a course, e.g. whether it happened on a `SERP` or on a course detail page. Hence the URL type of the current page is being saved as `ReferrerUrl`. The `Specifier` is added when there are multiple areas on the page, from which the course could have been saved. All possible values of these two fields are shown in figure 2.3.

<table>
<thead>
<tr>
<th>ReferrerUrl</th>
<th>Specifier</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>/home</td>
<td>mostViewed</td>
<td>„Most viewed courses“ section</td>
</tr>
<tr>
<td>/home</td>
<td>newCourses</td>
<td>„New courses“ section</td>
</tr>
<tr>
<td>/category/categoryId</td>
<td>query</td>
<td>(no special info on this page type)</td>
</tr>
<tr>
<td>/search</td>
<td></td>
<td>the searched query</td>
</tr>
<tr>
<td>/detail/courseID</td>
<td>viewedCourse</td>
<td>main course of the detail page was saved</td>
</tr>
<tr>
<td>/detail/courseID</td>
<td>alsoLike</td>
<td>“You might also like“ recommendation section</td>
</tr>
<tr>
<td>/detail/courseID</td>
<td>alsoWatched</td>
<td>“Other users also watched“ recomm. section</td>
</tr>
<tr>
<td>/history</td>
<td></td>
<td>(no special info on this page type)</td>
</tr>
<tr>
<td>/saved-courses</td>
<td></td>
<td>(no special info on this page type)</td>
</tr>
</tbody>
</table>

Figure 2.3: All possible values of the referrer fields of a CoursePin
2.2 **SPECIFICATION OF THE USER INTERFACE**

The frontend of the web application, namely the complete UI, is discussed in this section by providing a detailed specification of all possible page types and the individual components that occur all across the application. After that the connections between the pages of the system are covered in detail, illustrated by a comprehensive use case example of a new user who discovers the application features the first time.

### 2.2.1 PAGE LAYOUT TYPES

The web client uses three different page layout types that are the basis for all kinds of pages. The first type, **Home**, is used only for the homepage (compare figure 2.4a). It contains an eye catching background image, on top of which the search bar is displayed prominently. After this top section follows a listing of the most viewed courses and another one of new courses. Lastly there is a teaser area that describes what the IROM search platform does and why a login to the system benefits the user.

The next type of page layout is the **Category overview** (compare figure 2.4b). This layout is also used for one page type only: the categories. The web client has two types of categories: main categories and sub-categories. The discussed page layout is used to display the main categories, clicking on which leads to the list of the respective sub-categories, which are represented using the same layout.

Clicking on a sub-category leads to the view of course results, which uses the third layout type **Course results** (compare figure 2.4c). This view is a list of all relevant courses arranged in a grid layout. Beside the category courses this view is used for SERPs as well as for the representation of the user's viewed and saved courses.

The last page layout type, **Course details**, is also used for only one page type, namely the course details page (compare figure 2.4d). This page can be reached by clicking on a course e.g. from a results page or the homepage course listing. In the top area of this layout the details of the course are displayed, including title, picture, description, source and a prominent “Go to course” button leading to the website of the course provider.

Below this information section two types of recommendations are offered. The first recommendation section suggests similar courses to the original course. These courses are provided by the recommendation engines mentioned in section 2.1.1. The second shows courses that other users also watched (similar to the famous Amazon recommendation feature described in section 1.3.2). These courses are found as follows: First gather all CourseViews of this course, then extract the PageViewIDs from those CourseViews and get all other CourseViews with the extracted PageViewIDs. Lastly count the occurrences of equal CourseIDs in the found CourseViews and sort them by the highest number. The result is a sorted list of courses by the highest number of views, which were visited along the current course.

Both recommendation sections are arranged in a horizontally scrollable container, so that fewer (vertical) space is used for each section and the user sees both sections instantly without having to scroll over all courses of one section to view the other. This way the chance that the user misses the second recommendation section is minimized.
2.2. SPECIFICATION OF THE USER INTERFACE

Figure 2.4: The different page types

(a) Home

(b) Category overview

(c) Course results

(d) Course details
Throughout the discussed page layout types certain common components are used: the header, the course card and the feedback button. These are described in more detail in this section.

**The header**

The header appears on all pages except the homepage. The reason for this is that the main element of the header, namely the search bar, is placed outside of the header on the homepage to promote the search feature, as it is the main functionality of the web client. The other header elements, namely the category and the login menu buttons, are kept in place to be still accessible. The search bar is equipped with an autocomplete functionality that primarily suggests existing course titles based on the typed query string. As soon as the web client is used regularly by different people this feature can be enhanced by suggestions of common queries of other users, similar to the Google search engine.

The second element of the header is the category menu, which contains links to the category pages. The menu button opens a side navigation menu with all the main categories and their sub-categories listed underneath each other. To save vertical space, so that the user does not have to scroll far below to see all existing categories, and to prevent the user from being overwhelmed by the number of different categories, the sub-categories are hidden per default and can be expanded by clicking on the entry of the respective main category. To bring the existence of categories to the user’s attention, the homepage also contains links to the main categories just below the search bar in their respective colors. Such a colorful representation of the categories on the homepage helps the users to learn the color guidance system from the start, so that later they relate a certain color to the respective category at a glance without the need to read the label.

The last header element is the login button. As long as the user is not logged in to the web client, the button is only a link to the Google login form. After having logged in, clicking the button triggers a login menu to open, which contains links to the user’s history (all viewed courses so far), her saved courses and a logout action.

When the header is displayed, its position is fixed at the top of the page to always stay in the viewport of the browser. This way the search bar (and the other components) can be easily accessed regardless of how far the page is scrolled. Furthermore the search input field is always focused on every page load to allow instant search query input without having to click inside the field first.

**The course card**

The course card component occurs on almost all pages. It is used to show a preview of the course, by clicking on which the user gets to the respective course detail page. This component is used to list course results on a results page layout as well as in other course containers such as the most viewed courses section on the homepage.

This card consists primarily of a preview image and the course title. Furthermore it has a save button to save the respective course (if the user is logged in) and an info button, which triggers a preview of the course description, so the user can get more information about a course without having to visit its detail page for that. When the category of a course
is known (currently only on the category results pages), the category label is shown on the course card. With new course data all course cards shall have this information, which can help the user to scan a listing of such cards faster and thus to find relevant items easier.

The feedback button

The feedback button reveals a form through which the user can easily send feedback to the system administrators by providing a message and optionally her email address, if she wants to receive an answer. The button is placed fixed at the bottom left corner of the viewport on all pages of the web client to be always present and hence most accessible.

The idea behind this kind of user feedback is to offer a way for the user to directly communicate her opinion on the client’s functions, to make feature requests or to ask for help regarding the search platform. Such a form of direct feedback is invaluable for understanding the users’ interactions and learn about their wishes for new functionalities.

2.2.3 Connections between pages

To illustrate possible workflows of a user, figure 2.5 shows all connections between the different page types. These connections are represented as arrows between the pages and are basically page links. Following these links creates a chain of user interactions, a so-called workflow. A probable workflow of a new user could be as follows:

The user Kate enters the web client through the homepage. After a short scan of the existing page elements she decides to search for some MOOCs by typing a search query in the search input field. She submits the query and lands on a SERP with an overview of all courses matching her query. Kate evaluates the relevance of the proposed results by analyzing the course title and the description preview. Then she decides to get more information about a certain course and clicks on the specific search result. This way she enters the course detail page, where she sees more information about that MOOC. The course sounds interesting and Kate decides to enroll in it by clicking the “Go to course” button.

Having found such an interesting MOOC she is curious to explore the IROM web client deeper, so she goes back to the course detail page. In the lower section of the page she discovers other courses that are recommended for her. Several of these courses seem quite interesting but our user does not have the time to enroll in all of them simultaneously, so she decides to save them for later by clicking the save button of the respective item. But this action requires her to log in to the web client, therefore she clicks the login button in the header area of the page, completes the external Google login process and is redirected back to the previous course detail page. Now Kate can save the interesting MOOCs and access them later by going to the “Your saved courses” page.

The found courses have inspired her so much that she wants to find more interesting things without a concrete idea what she is actually looking for. So she browses the categories of the search platform that sound promising. After viewing several courses in detail Kate leaves the web client. The next day she has a conversation about psychology with her friend and remembers that she has actually come across an interesting MOOC regarding exactly that topic, but she can’t remember the course title. Luckily her browsing history was saved in the IROM search application, so she just has to log in to the client and go to the “Your viewed courses” page to find the certain MOOC.
Figure 2.5: Connections between the different pages
2.3 GATHERING METADATA

As stated in the objective, this master thesis aims to create a platform for the IROM search and recommendation engines, which tracks all important user interactions to analyze and evaluate the effectiveness and efficiency of the proposed engines. Through such an extensive tracking of the web client the evaluation results can be highly useful for improving the algorithms of the recommendation engines. Therefore the client must contain logic to gather as much valuable metadata about user interactions as possible.

Evaluation of the engines used

To find out how good the employed engines work it is important to know, how valuable the proposed results are for the users. In the case of the search engine the efficiency can be measured for example by how often a query is reformulated before the user finds a suitable course for her interest or if she finds one at all. It is also helpful to know, which query produced the desired results and if it is an intuitive match or simple luck. For this the system must provide the ability to associate a course view with the respective search query. Furthermore it can be analyzed, how many queries did not result in any course views, thus being unproductive. A stronger signal of approval is when the user saves a MOOC or even follows the link to the course provider, probably meaning that she wants to enroll in this course. Such insightful information can help the developer to assess the performance of the engine and to find any flaws in the algorithm, resulting in concrete ideas for further improvement.

To assess the quality of the recommendation engines it is important to know, how often, if at all, the recommended courses are viewed by the users. The client must track all interactions with the recommendations, whether the user only viewed the recommended items, saved one of them or actually clicked on one of the proposed courses, together with information about, on which specific course the recommended item was found, which type of recommendation it was and so on. Having this information shall help the developer of the individual recommendation engine to check, whether the provided suggestions are helpful for the users, and to improve the quality of the algorithm.

Evaluation of the application design

In order to test the engines many users are required to obtain meaningful evaluation results. This is why the web application needs to have a user friendly and attractive interface, so that the platform is enjoyable to interact with and more people are willing to use it. To ensure the usability of the designed interfaces their quality also needs to be evaluated. Therefore the client must further gather metadata that can help to assess the user acceptance of the web client. As a first approach of analyzing the application it must be tracked, if all important elements of the interface are used to a desirable extent. For example, minor interaction with an element usually means that it is not designed or placed properly. Such indirect signals help the developer to discover flaws in the interface design and to make specific adaptations.
In the test phases, which are part of this work, some major design choices shall be evaluated regarding their functionality and quality. One such choice is the grid layout of the SERPs. The main purpose of the grid layout is to encourage users to consider multiple search results as potentially relevant and this way to explore more content, without being driven by a hypothetical ranking of the search engine. To test the effectiveness of this design method the application must track the position of the clicked results in order to determine, if users still prefer the top results or if the clicked results are more distributed on the SERPs, which would approve the intention behind this design choice. Further design choices that shall be tested are described in the last part of section 4.2.

**Tracking methods**

The necessary metadata shall be collected in two ways: with Piwik on the one side and with custom client code on the other. The reason for such a comprehensive approach is that the analytics platform Piwik does not track user interactions deeply enough by default. Of course one can enhance the tracking functionality by defining custom user actions that should be tracked. But in several cases it is easier and much more meaningful to track those actions in the main database of the web client by custom code, because the metadata can be included into the existing data structure of the database scheme. Nevertheless the analytics platform shall be used extensively to collect as much relevant metadata as possible, even if this approach results in redundant information. Nowadays redundancy of data is desired, rather than avoided, to ensure information completeness if one system should fail. Furthermore the two systems can be used to enhance the information gain by matching such redundant data and combine individual additional information into a bigger picture.
3 IMPLEMENTATION

3.1 Developing the application

The purpose of this first version of the IROM search platform is to be a proof of concept. Therefore the main objective was to develop a fully functioning prototype with focus on the newest versions of the most common browsers. With such an approach new technologies can be used including recently released web frameworks and plugins. Furthermore the platform shall consist of a rich client that handles the complete logic of the user interface, and a backend server that is responsible for communication and data flow between the client and the data providers, namely the database and the search and recommendation engines. This section gives an overview of how the application has been implemented, which technologies have been used and what the resulting interfaces look like.

3.1.1 Structure of the rich client

In the IROM application the rich client comprises the complete user interface with all its components and functionalities. It is built with the Angular 2 framework (https://angular.io/) including the use of the Angular Material package for modern interface elements, which is based on Google’s Material design, and of the Angular Webpack package, an efficient build-tool for development and production purposes.

An application developed with Angular is a single page app: this means all sub-pages of the platform are not standalone web pages but rather components which are being loaded into the main application container on user request. These components represent either entire individual pages or common reusable elements, both types have been illustrated in sections 2.2.1 and 2.2.2. They are mainly JavaScript classes that contain all the logic for each functionality of the page or element and consist further of an HTML template with Angularized markup and CSS (or as in this case, Sass, a preprocessor for cascading style sheets) to represent the content in the browser. As far as needed such a component includes additionally a service that is responsible for data sharing between individual components or between the client and the server. Furthermore common models are defined in the client, if needed to represent the data. These correspond with the models defined on the backend side of the application, e.g. User or Course.
When the client needs certain data, e.g. the most viewed courses for the homepage, the respective component asks the responsible service to fetch these courses from the backend. The service makes then an HTTP POST request to the backend server, which in turn retrieves the requested data from the database, or gets it from the engines in other cases. After preprocessing this data on the server side it is passed back to the client service as a JSON object, from which the component receives its information, decodes the JSON response using a defined model (in this case the Course model) and displays the data in the respective user interface. Such calls to the backend server are made asynchronously to enable loading of other page elements before all needed information has been retrieved (this is the principle of lazy-loading), so that the user can see the client's reaction to her request through the loading process.

### 3.1.2 Backend Communication

The backend server, which is implemented in Java with the Play Framework (https://www.playframework.com/), retrieves requested data in two different ways: either using the Java module Jongo to communicate with the database, or making requests to the search and recommendation engines through a REST API. Each way is now explained in detail by taking the example of two representative methods.

*Jongo-based communication with the database*

![Sequence diagram of retrieving course details from the database](image)
Jongo (http://jongo.org/) is a Java module which offers the developer an easy way to make queries to a MongoDB database and unmarshall the results into Java objects. The typical communication process can be illustrated using the example of retrieving details of a specific course for the course detail page (refer to figure 3.1).

The first step of this controller method is to check whether the request is authorized and thus comes from the client. If the authentication is successful, the next step is to make a request to the DBService which uses a PlayJongo instance to communicate with the database. The DBService retrieves the requested course details by the provided course ID and hands the result over to its own ResultHandler method. This ResultHandler processes the course result object into a new Course Java object acquiring all needed information, which is then returned to the initial controller method and is sent as a JSON object to the client.

RESTful communication with the engines

To get course data from the search or recommendation engines, the backend server has to make a request through a REST API. Such an interface describes how distributed systems can communicate with each other using standardized structures like HTTP/S and JSON. Through the use of standard methods almost every web service meets the criteria to be RESTful. Such a service is often called by its URL through an HTTP method like GET or POST that tells the service which action to take. Data sharing is commonly realized through descriptive languages such as JSON (JavaScript Object Notation) and XML (Extensible Markup Language).
Figure 3.2 shows the steps for retrieving similar courses to a specific course. These courses are shown in the “Similar courses you might also like” recommendation section on a course detail page at the frontend side. First the request from the client is authenticated. Then it is verified that the API URL with the provided query parameters, in this case the course ID and the requested number of similar courses, is valid and after that the API call is made. The response of this call contains the \( n \) requested most similar courses specified by their ID along with some additional information such as the title and the course provider. From here on this data is processed and returned in the same way as in the Jongo-based example described previously.

### 3.1.3 Design of the user interface

Since the IROM project is still in progress, the course data is not complete yet. Therefore not every functionality of a modern vertical search application could be implemented within the scope of this master thesis. However, some features have been implemented using mock data, to show how the application can look like when the features are fully realized. In the following, the design of the most important parts of the search platform is being presented.

---

**Search bar**

The header contains the most important part of the application: the search input bar. The developed prototype uses an adaptation of the recommendation engine for the search functionality, since the planned Solr search engine is not ready yet. This recommendation engine works by retrieving courses that are similar to the provided search query. Therefore search methods such as partial matching could not be implemented yet. To show how auto-completion of a search query might work the application uses all course titles as possible suggestions. This feature can be enhanced as soon as the Solr search engine is ready for use, e.g. by providing suggestions for similar queries and not only course titles that start with the typed in query string. As planned in the conception phase, the search bar meets all criteria to offer a satisfying user experience (compare with figure 3.3):
– The input field is *focused* on page load to enable immediate typing

– The *label* of the search input field describes what the user can search for

– The search bar stays *fixed* at the top of the viewport on all pages (except the homepage) to eliminate the need of scrolling up when the user has previously scrolled down

– The *autocompletion* feature immediately updates its suggestions on every change of the input value and

– The query is *preserved* in the input field after submitting the search form, to facilitate reformulating it and to show it to the user, so that she does not have to recall it.

---

**Figure 3.4: Placement of the search bar on the homepage**

As mentioned in the conception (refer to sections 2.2.1 and 2.2.2), the search bar is displayed more prominently on the *homepage*. Figure 3.4 shows the implementation of the idea. Maintaining space around the search bar and highlighting it with an eye-catching background image, which consumes a major part of the screen, increases the focus on the element. Furthermore this place is used to promote the categories by showing a label for each of the main categories directly beneath the search bar. The intention of this design is to communicate the browsing feature of the platform. Different colors are used for the individual labels to make the user familiar with the color guidance system of the categories.

---

**Course cards in listings**

A key component of the interface are the cards for individual courses. These are used throughout the application in every type of a course listing, e.g. for SERPs, recommendation sections and category listings. As described in the conception, any course listing is presented either in a *grid layout* (see figure 3.5a), when this listing is the main content of a
page, or as a carousel that can be scrolled horizontally (see figure 3.5b), when the listing is offered additionally to the main content or multiple listings are present. Therefore the grid layout is used for SERPs, category courses, user’s saved courses and user’s viewed courses, and the most viewed and new courses (pages behind the “view more »” links on the homepage), whereas the carousel is applied for the two homepage sections (most viewed and new courses) and the two recommendation sections on the course detail page.

The advantage of such carousels is that they consume far less vertical space and therefore the user can see which other content the page has to offer without having to scroll too far down. But such carousels should not contain too many items to avoid annoying the user with much scrolling. Therefore the number of items is limited to ten and if there is more content available, then a “view more »” link leads to a separate course result page.

To enable comfortable horizontal scrolling of the carousels across different devices they can be scrolled by using the scrollbar below the listing, as shown in the upper carousel of figure 3.5b. The scrollbar appears as soon as the user hovers with the mouse over the carousel or touches it. On touch devices the container can be scrolled by swiping the content in the desired direction. Furthermore paddles are shown on each side of the carousel to indicate that there is more content in the direction of the arrow. These paddles can also be used to scroll the container item after item by clicking or tapping on them.

![Grid layout](image)

![Carousel](image)

(a) Grid layout

(b) Carousel

**Figure 3.5: Course listing types**

![A typical course card](image)

![Further course information](image)

(a) A typical course card

(b) Further course information

**Figure 3.6: Course card design**
Course card design

As mentioned before, the listings consist of cards for each course. Such a card contains a picture, the course title, the logo of the provider, the category label and two action buttons (compare to figure 3.6a). There are some constraints in the current representation of a course card. One constraint is that the images used for a course are just placeholders for the real course images, because the data that is used for this prototype is not complete yet. To show how the interface could look like with complete data, random pictures were assigned to each item. The pictures are blurred to lessen the distracting effect of the fact that they have little to do with the course subject.

Another constraint are the missing category labels on all cards except those on category pages. The reason for this is that the course data provides no information about which category a certain course belongs to. The category pages offer the opportunity to show how this information can be displayed on a card in the future, even though it is the one page type where this information is actually redundant. But in all other listings, be it the user’s saved courses or recommendations on a course detail page, such category labels with an explicit color can help the user to classify the items instantly without having to read each course title.

The third constraint concerns the course information snippet. When a course is provided as a search result or a recommended item, it is helpful to know, why this certain course is being offered. Regarding search results the explanation is often a snippet of the item description with keywords related to the search query (KWIC) highlighted in some way. In recommendations such a snippet could have highlighted keywords related to the course, for which the item has been suggested. Such helpful snippets are not being provided yet by the engines used in this application. Therefore in order to offer the user some additional information the cards contain a preview of the actual course description. This preview can be revealed by hovering (or tapping) the info action button of the card. The text appears as an overlay over the course image (see figure 3.6b). The design choice of hiding the description has been made with the intention to avoid overwhelming users with too much text and to save valuable screen space, so that more results can be shown "above the fold", meaning the screen space that is visible without scrolling down the page.

The last elements of a course card are the second action button shown as a bookmark icon, clicking on which adds the respective course to the user’s list of saved courses, and the provider logo on top of the course image. This provider information is useful for instantly seeing which provider offers the MOOC, because some users don’t want to register on every MOOC platform but concentrate only on a few chosen ones. This way they don’t have to visit the course detail page to learn, whether a certain course is being offered by their desired provider.

--- COURSE DETAIL PAGE ---

The design of the course detail page is shown in figure 3.7. The top area of the page is reserved for detailed information about the course and the "Go to course" button. As this action button is the most important element of the page, it is designed most prominently to attract the user’s attention immediately. This is achieved by a large size of the button and an eye-catching color. To ensure that the color seems outstanding and remarkable it is used
sparingly across the platform. Furthermore the new tab icon after the button text implies that it is a link to an external source, meaning that the user leaves the IROM platform and lands on the external page of the course provider by clicking this button. Of course the external page is opened in a new tab of the browser, so that the IROM application does not get lost when navigating away.

Figure 3.7: Example of a course detail page
3.1. DEVELOPING THE APPLICATION

The details section at the top further contains the course picture, the source of the course (mostly a university), the complete description and additional information such as the provider and the price model. On the right side of the course title there is also a "save" button, as known from the course cards, to let the user save a course also on its detail page. After the details section follows the recommendation section with similar courses as well as courses that other users also watched. Both recommendation listings are arranged as carousels, as mentioned before.

— Categories —

Figure 3.8b shows the page with the main categories. Each of these are links to an overview of the respective sub-categories arranged in the same way. The category titles take up again the colors used for the category labels shown on the homepage. Clicking on the "Browse" button in the top left corner of the header reveals the category menu as demonstrated in figure 3.8a. The list of sub-categories can be expanded by clicking the respective main category, and each of these sub-categories link to the respective course results page. The first link in the menu, "All categories »", leads to the page displayed in figure 3.8b.

![Category menu](image1)

(a) Category menu

![Main categories](image2)

(b) Main categories

Figure 3.8: Preliminary categories

The individual categories are made up since the necessary data is not present yet. They have been defined in order to implement a basic template for future categories and to provide a browsing feature for users. Nevertheless the classification has been chosen carefully based on common practice, and license-free images have been picked to match the topic as best as possible. Although actual course data for each category is missing, related courses that can be browsed are found by asking the search engine for the individual category title. These search results are then displayed on the respective sub-category pages as illustrated in figure 3.5a of section 3.1.3.
CHAPTER 3. IMPLEMENTATION

--- RESPONSIVE LAYOUT ---

The complete platform is designed to be responsive, meaning that the arrangement of the elements is adapted for every screen resolution. Figure 3.9 demonstrates how a course detail page looks like on the most common screen sizes. For large monitors the maximum width of the content is limited to a size that is still comfortable to look at, so that the content can be comprised completely at a glance rather than having to scan it from side to side.

The application uses two different breakpoints to distinguish between small, medium and large screen sizes. These breakpoints are used for the placement of certain elements. For example, the details section of the page (refer to figure 3.9) is reordered on smaller screens (from medium down) by letting the content on the right side of the image float beneath it. This way the description text is not squeezed into the narrow space between picture and edge of the screen but is extended to the entire width of the space beneath. Furthermore the “Go to course” button is extended to the entire width on small devices, because then it is perceived better and its size is most comfortable for tapping it with the finger.

![Figure 3.9: Responsive design on the example of the course detail page](http://ami.responsivedesign.is/)

Some other general adaptations include unfixing the header bar on smaller screens, so that it no longer sticks to the top of the viewport but is scrolled together with the main content. The reason for this behavior is to avoid clustering the viewport with unmovable elements, which would reduce the available space for the main content and could annoy the user. Also the search input field is no longer focused automatically on page load on touch devices because otherwise the virtual keyboard of the device would be opened automatically, thus reducing the already limited viewport size further.

--- Image generated with http://ami.responsivedesign.is/ ---
Another important adaptation is the diminution of the general font size on smaller screens. Such downscaling is allowed because devices with smaller screens are usually held in a smaller distance to the eyes. Since almost all size-related styles of the interface are defined relative to the general font size, all respective elements and their margins shrink together with the font on small viewports.

— Material design —

Since the application shall look modern and follow usability guidelines also in the design aspect, the application has been styled following the Material design. The core idea of Material design is the placement of content on material, meaning planes in various dimensions. To imitate dimension and elevation of elements different shadows are used which can change on user interaction, thus simulating elevation change of the respective element under pressure. Material design stands further for simplicity, clarity and closeness to natural behavior of the various elements of the interface.

To implement these design aspects a Material plugin has been developed specifically for Angular². With this service Material components can be realized by simply using predefined Angularized HTML syntax. Common elements like buttons, cards and dialogs with their appearance and behavior are thus a product of this plugin. Other components and their behavior, which are not provided by this plugin, have been designed following Material design guidelines as far as possible.

Furthermore the platform is designed using a clear and easy-to-read Google font, Material icons for several buttons that are simple and intuitive, and colors picked from the professionally created Material color palette based on the colors which were chosen earlier with Adobe Color CC.

²https://material.angular.io/
CHAPTER 3. IMPLEMENTATION

3.2 METADATA COLLECTION

In order to properly analyze how the platform is interacted with, a web analytics tool is needed. Web analytics covers the measurement, acquisition, analysis and interpretation of web data to optimize a web application [15]. With this knowledge the stakeholders of the application gain insight into the actual user interaction and can adapt the interface to improve user experience. This is important because web applications are eventually made for users, therefore it is their purpose to provide maximum usability.

For the IROM search platform a web analytics tool named Piwik\(^3\) is integrated into the application. Piwik is an open source service similar to Google Analytics\(^4\), but it is installed on your own servers thus providing full access to and ensuring ownership of the collected data. The software is built on PHP and MySQL, and it starts tracking all website visits once the respective JavaScript code has been implemented into the application. Beside the number of visits, the service further tracks which device has been used, where the user comes from and how long she has stayed on the platform among other useful metadata. The method such analytics tools are using is also known as Page Tagging, meaning that each page of an application is being tracked, thus giving a full overview of the website traffic [12, p. 29]. The gathered metadata is then represented in a number of interactive graphs on the dashboard of the tool.

To enable tracking of specific events such as clicks or other interactions, one line of JavaScript code must be added to the code base of the platform. The interactive Piwik platform lets you further define certain goals, which are marked as achieved based on these specific events. An IROM related example of such a goal is when the user clicks the “Go to course” button. It probably means that she has found an interesting course on IROM and discovered how to get there. To add such an action to an Angular application, a Piwik wrapper is necessary to integrate its logic into the Angular code. For the IROM search platform Angular2Piwik has been used\(^5\). With this lightweight wrapper the aforementioned page tagging actions can be defined using native Angular 2 code.

In the IROM application almost every button is being tracked using custom events. Furthermore interactions other than clicks are being monitored such as scrolls to certain areas or submitted search queries. This extensive monitoring allows a precise analysis of the user interactions and their interpretation. The next chapter describes in detail which conclusions for the IROM search platform can be drawn from the gathered metadata and other analysis methods.

---
\(^3\)https://piwik.org/
\(^4\)https://www.google.com/intl/de/analytics/
\(^5\)https://github.com/awronka/Angular2Piwik
4.1 The Basics of Web Analytics

— Why Evaluate —

Implementing an application designed for user interaction is not the final step in development but rather the initial. An interface created with the best intentions is nothing worth without testing the result and verifying its usability, quality and users’ approval. Experience has shown that users are very different and have various individual expectations and levels of expertise. Therefore it is not enough to design an application based solely on thoughts and assumptions of the developers, but should rather be analyzed extensively how the implementation is perceived and operated with by actual users.

With the diverse possibilities of today’s technologies web applications are predestined to be thoroughly analyzed and evaluated, because they are the only medium the interaction with which can be tracked so extensively. Every click, every scroll or hover, the precise time spent on every page and even areas that the users looked at are a form of indirect user feedback and can be monitored with modern tracking methods.

Through such detailed information about the user’s behavior the stakeholders can comprehend how the application is interacted with to a much greater extent, thus discovering design flaws and getting the opportunity to eliminate them. Therefore every modern web platform uses web analytics to track its use and interaction. Web interfaces are designed for users and their happiness and satisfaction defines the success and profit of the platform. Therefore it is essential to provide the most pleasing experience with the application.

— How to Start —

The interpretation of the gathered tracking data is not always self-explanatory. It often comes down to experience and intuition of the evaluators. To verify one’s assumptions new tests and evaluations must be conducted on the redesigned product, and the success of the redesign is then measured based on some predefined metrics. Particularly the common steps of such an evaluation process is as the following [12, p.38]:

— Evaluation —
1) Measure use of the platform – once
   Choose and apply appropriate analytics methods, start gathering meta-
   data

2) Evaluate and interpret – weekly
   Monitor and interpret changes in important metrics, draw conclusions

3) Optimize application – monthly
   Implement minor improvements based on previously made conclusions

4) Use gained insight for redesign – yearly
   Conduct a redesign of the application based on all previously gained in-
   sight and experience

It is a continuous process that is never ending. Each new feature of the application has to be analyzed regarding its use and efficiency. Furthermore its impact on other elements and functionalities must be checked to ensure a smooth integration into the platform. As soon as web analytics is introduced, more and more insights are gained from the gathered data that reveal weak spots offering opportunities for improvement.

— VARIOUS ANALYSIS METHODS —

Tracking user interactions on web platforms can be done applying various strategies and such forms of indirect feedback are not the only way to conduct web analytics. Direct user feedback in form of surveys and interviews also belongs to common analysis methods. Hassler names the following as the most important web analytics methods [12, p. 28]:

– Page tagging (most common method used by tools such as Google Analytics)
– Logfile analysis
– A/B testing and multivariate testing
– Online surveys
– Personal interviews and user observations

Direct user feedback which can be gained through surveys and interviews has a huge advantage over other analysis forms: Through these methods concrete questions can be answered and the user’s opinion can be heard directly without having to make assumptions on her behavior. But on the other side a lot more time has to be invested in such analysis forms compared to gathering data automatically. Furthermore direct user feedback requires the user to invest valuable time without gaining any benefits. This means small amounts of very detailed and concrete feedbacks that cost a lot of time stand against far more tracking data in shorter time that are very abstract and ambiguous. Therefore, to gain valuable insights, the best strategy is to apply both types of analysis to an appropriate extent.
4.2 Testing Approach

The purpose of project IROM is to continuously improve the recommendations, therefore many users are needed to gather enough metadata. But only a functioning and enjoyable interface encourages people to use it. Interaction with the IROM search platform has been extensively analyzed and evaluated to learn whether the concept is working as intended or if certain things can be improved. A two-phase approach has been applied, the first was carried out as a lab study consisting of direct user monitoring and interviews, and the second was a broader empirical field study with tracking click-through data and other relevant metadata.

Both testing approaches aim to analyze the usability of the design interfaces, whether buttons are visible, the placement of elements is understandable, and so on. The field study should additionally give insights in the quality and user acceptance of the implemented engines, as far as such evaluations are possible given by the rather small amount of users and limited study duration of only ten days. Since the IROM application implemented by this thesis is newly created and almost unknown, only few people visit the platform. Therefore not every aspect of a standard web application can be measured and evaluated appropriately. The purpose of testing is predominantly to get a prove of concept and to examine whether the kind and amount of gathered metadata is sufficient for a modern web analytics evaluation.

Lab study

Five people have participated in the lab study carried out by this thesis for the evaluation: Two full-stack developers, one scientific employee with a master’s degree in Renewable Resources, one Law student and one employee in the Logistics field. Knowledge about MOOCs as well as experience with web applications ranged from expert to layman. Each session took ca. 30 minutes to an hour. Each participant sat in front of a desktop monitor with the application being open. Everyone agreed on being filmed during the session for the only purpose of making a transcript afterwards, so that no simultaneous manual writing was needed. This way not only the participant’s answers can be easily retrieved afterwards but also his or her behavior and feelings during the session, such as annoyance or excitement while interacting with the application.

The testing procedure was as follows: After a short description of what the application is about, the task was to get familiar with the platform by exploring the interface. The participants were free to interact with the application any way they wanted. As soon as they thought they were done exploring the interface, the participants were asked to search for courses related to nutrition. After questions about how and why they chose their result, the participants had to analyze the course detail page further and tell which information they find helpful and what is missing in their opinion. Having dealt with this part, the next task was to explore the content of the platform further by browsing the categories and give some feedback about the respective interfaces. The last part of the experiment was to find out, whether the personalized features “Saved courses” and “Viewed courses” were understandable and useful. At the end the participants could express further thoughts about the platform, if something came to their mind that was not mentioned yet.
For the empirical field study carried out by this thesis, students and scientific employees have been recruited. The participants had to fill out a short form with their field of study and their email address to get details on the study later. The purpose of such a survey approach was to estimate how many people (and with which background knowledge) participated in this study. After a registration period of five days the subscribers received an email with detailed information on how they should proceed to participate. By the time there were only twelve people who filled out the form correctly. But according to the tracking data a lot more people visited the platform during the testing period of ten days.

The idea behind the field study was to let users explore the application on their own, so that the click-through data was as realistic as possible. But to give participants a starting point (applies only to those who had registered), they were asked to visit the platform, explore the content and search for some interesting topic, e.g. "python programming". The task was held short and ambiguous deliberately to let the participants freely decide, what they want to do and how they interact with the application. The participants were also invited to leave some feedback about their experience with the platform, if they wished to.

1) Explorative grid layout of course results
   Analyze the distribution of the position of clicked courses – High distribution indicates better exploration

2) Presentation of course information on course cards
   Find out how often information text is viewed – High number of views indicates that the presentation concept is working

3) Design of the „Go to course“ button
   Detect how many course views lead to a visit of the course provider – The proportion indicates whether the button encourages users to click out

4) Usefulness of recommendations
   Check how often they are viewed of all course views and how many of the recommended courses are clicked – A high interaction with recommendations indicates a high interest in them

5) Usefulness of features for logged in users
   Inspect the number of logins after failed attempts to save a course; check how many logged in users viewed "Your saved courses" and "Your viewed courses" pages – High ratios indicate a rather strong usefulness

The core of the evaluation process concentrates on these aspects, but during the lab study some other things have been found that were confusing the participants. A detailed review of these things is given in the next section.
4.3 Evaluation Results

4.3.1 Insights from the Lab Study

The lab study sessions were the most enlightening regarding important user feedback. Every participant was very engaged in the process and expressed his or her thoughts about the platform freely. Thus the most important ideas for improvement resulted from this test phase and some of them were realized before starting the second test phase. Of course there were also very individual opinions, because every person percepts things from her own point of view and wants the application to provide answers that are very specific for a certain interest, which is not always possible when dealing with many other topics as is the case with MOOCs. An example was the demand for a filter on specific programming languages which requires very detailed information about the courses and cannot be applied to courses from other domains e.g. medicine. A possible solution might be to define different filters for each category of courses, as soon as the course data contains more specific information.

Nevertheless the participants provided feedback regarding each aspect that had to be analyzed as listed in the last part of the previous section 4.2. The following section presents the gathered evaluation results and provides possible solutions for found issues.

— Grid layout of course results —

During search all participants had scrolled the SERPs quite far down but lastly chose one of the first few results because they hadn’t found any interesting courses further down the page and assumed that the search results are ordered by relevance. Altogether they were not satisfied with the results, because none seemed really relevant. The first reason for the rather poor search quality is that the used search engine is not optimal for this kind of result retrieval. A Solr driven search engine should provide better results and shall therefore be used for the application as soon as it is implemented. The second reason is probably the small amount of courses in the database due to outdated data. As soon as recent course data from more providers is collected, the search engine could reveal better results.

Although an order by relevance was assumed, which is not very surprising due to modern every-day usage of all kinds of search engines, the results were scanned rather randomly without a strictly systematic approach such as top to bottom or left to right. One participant explained this behavior by the fact that the nature of this type of search is not as precise as of a search for certain facts but is rather informative, thus the order by relevance cannot be that strict.

The amount of search results, which currently defaults to 50 items, had been found very overwhelming and three of the five participants wished for a pagination of the results. One of them mentioned that by scrolling too far down she had lost track of which items she had already scanned and which she hadn’t. Another showed her annoyance with the question, if the results presentation is going to end anytime. On the question, whether he
would go to the second SERP when a pagination would be provided, the third participant answered that he would do it because the nature of the search results impedes a strict order by relevance. From these observations it can be derived that the representation of course results should be paginated and that each page should contain a rather small amount of items. As soon as the search engine has been changed, pagination functionality should be provided and it should also be tested how many results are most appropriate for this kind of SERPs.

— DESIGN OF THE COURSE CARDS —

Feedback on existing information

When the application was tested with the first two participants all course cards had the same placeholder image which contributed to a dreary appearance of the interface. After having received such strong feedback on this matter, in order to bring more life and variety into the platform random images have been added to the individual courses as a temporary solution until new course data is available. The last three participants then admitted that they first looked at the images to assess what a certain course is about, even though they were told that the images are not real but just temporary placeholders. This fact suggests that images on the course cards are important and help the users perceive the different courses faster and easier by automatically drawing their attention.

The next important information for the testers was the course title. One participant said that in her opinion a course result is not that relevant when the searched keyword is not present in the title, particularly if the keyword is a special term and not a general topic. The third source of further information, namely the description snippet when hovering the course info button on the card, had been found by every participant without problems. In the first two sessions this text snippet had been rather long which discouraged the participants to read the information. Subsequently this snippet has been shortened to reduce the overwhelming effect. After that the functionality received positive feedback. It was found good that the information is shown only when the user needs it and thus information overload is being prevented, but the access to it is still straightforward upon need. Only one participant said that he would rather see the description instantly without the necessity of user interaction. Thus a further study is required to evaluate which kind of presentation works better for users.

Additional helpful information

One participant said that it would be helpful to see which provider offers the course, without having to visit the detail page first, because a user might not want to register on all the different platforms only for one course that sounds interesting. Thereupon this information has also been added to the cards in the form of a logo of the respective provider.

A more important additional information suggested by the same participant was the respective category of the course. Upon developing the idea further the color guidance system for the categories has been created and the category label in its respective color has been added to the course card where this was possible (currently only on the category pages to show how it could look like). Explaining and showing the feature to the following
study participants received very positive feedback because this information facilitates the scanning of course results a lot, since the user can concentrate on those items, which belong to the desired category. One of these participants mentioned that as soon as the category information is available for all courses it should also be displayed on the course detail page, linked to a course results page with more courses from this category.

**Presentation on small screens**

The last feedback regarding the course card design relates to the responsive design of the application. Three participants had found the cards too bulky for small smartphone screens and would appreciate a representation of courses in a list layout.

An example of how such a listing could look like can be seen in figure 4.1. Reducing the size of the image and placing the course title beside it instead of below leads to a smaller size of a course item, thus providing space for more items inside the viewport of the screen.

But this design has not been implemented yet due to shortage of time to test, if it will be approved by users. Therefore this redesign could be a subject of the future work on this application.

![Design suggestion for SERPs on a small screen](image)

**Figure 4.1: Design suggestion for SERPs on a small screen**

"Go to course" button

The purpose and functionality of this important button was instantly clear for all participants in the lab study. Everyone had also perceived this element at first sight of the course detail page. But one participant mentioned that the very prominent design of the button reminds him of advertisements rather than a serious link to an external site. He said that he would look twice before clicking on such a button. Therefore it should be analyzed further whether more users perceive this design the same way and if a more discreet design would be appropriate. For such design changes A/B testing would make sense as soon as more users visit and interact with the application.
— Acceptance of recommendations —

The design of the recommendations listings as carousels has found the participants’ approval. The handling had been easy and intuitive. Several recommended courses had been viewed across both types of recommendations. Both types of recommendations had been found interesting and encouraged further exploration of the contents. On the question of whether the difference between the two types (similar courses and courses other users also watched) is clear, every participant agreed. Regarding the second type (courses also watched by others) the participants had no problem with the fact that the suggestions were not necessarily similar to the main course and had found it rather useful for thinking outside the box.

— Features for logged in users —

The features discussed in this part are saving courses and accessing one’s history of viewed courses. Basically the participants understood the purpose of these functionalities and found them helpful, but their real usefulness can only be determined after analyzing their actual usage across many people in the long run.

**Saving courses**

For two participants the purpose of the save button on course cards and the course detail page was instantly clear and they understood that the feature is working only for logged in users. Other two participants clicked the save button, read the error message and logged themselves in to use the feature. But they had difficulties to find the saved courses afterwards. This indicates that the place, where the user can find her saved courses (namely behind the account menu item “Your saved courses”), should be pointed out by some kind of hint or not being hidden behind the account menu.

The last participant clicked the save button on a card and assumed that the course had been marked somehow without being logged in, until it was pointed out to her that there was an error. She had not seen the error message appearing as a small snack bar at the bottom of the page and after clicking the save button again she didn’t manage to read the error message because it was disappearing too quickly. Therefore the duration of the snack bar appearance has been extended to ensure that every user gets enough time to read the message. This participant made further a suggestion to enable the saving feature also for not logged in users like it is the case with some online shops where you can mark items during exploration. Such unlimited functionality of this feature is also a possible task for the future work of this project, and should then be made clear for the user through explicit hints.

All five testers got the idea behind this feature and found it useful. Everyone had a similar situation in mind, when the feature might be helpful, for example if one has currently no time to explore a certain course in detail but wants to remember it for later. One participant compared the feature with the principle of a shopping list.
History

Two participants found the naming of the list as "Your viewed courses" irritating because they thought of courses that they had actually visited and completed. One of them actually wondered how the courses landed on that list without him taking any action. Upon scanning the items he realized that those were the ones he viewed in detail. Subsequently he suggested to offer users a list to which they can actively add courses that they had already visited, so those are not recommended to them anymore on the platform.

The other three participants had no problem understanding the functionality, once they had discovered it. All in all the feature had also been found useful and all testers could imagine a situation in which it might come in handy. The functionality had further been compared to the browser history.

Two of the participants wanted the list to be structured somehow. The suggestion of segmenting the course results by time periods such as current week, last week, current month etc. had found their approval. This segmentation could be implemented in future versions of the platform, and it should then be tested, which time periods fit the user’s need best.

— Further feedback —

Beside the earlier defined criteria for analysis the participants gave further feedback to almost every part of the application and offered ideas for improvement of the platform. The gained insights have been very valuable and some further improvements had been made in the course of evaluation. Following is a short overview of those ideas.

Homepage specific aspects

The version, which had been tested in the lab study, used the grid layout for the listing of most viewed and new courses on the homepage, and each list consisted of eight courses. One participant criticized the long expansion of the lists on smaller screens, which had the effect that the user couldn’t differentiate between both lists because it just seemed to be a series of course cards. Another participant had found it strange that the listings contained only eight items and not the Top 10 as it is usually seen in other cases. The third participant had found the separation between the two listings also too unremarkable even on a standard-sized desktop screen.

Following these comments the representation of both lists has been changed to two course card carousels as known from the recommendations listings on the course detail page. This way the separation between the two lists is more straightforward, the user does not have to scroll far down to see the next list and the number of items could be extended to ten courses, now that there was no layout problem anymore.

Another interesting fact which could be observed during the lab study is that all participants tried to click on the headline of these listings to view more courses of this type (new or most viewed). Thereupon additional pages were created which contained more of these courses and were linked to from the headlines, which also got a "view more" hint.
One participant had further the idea to personalize this course representation for logged-in users. For example at the top there could be a list of new similar courses to the ones the user had viewed lately, either based on direct similarity or on the same category. This idea has also been implemented (as far as possible with the outdated and incomplete course data). All in all the idea of these listings on the homepage had been found interesting and encouraged the participants to explore the course offer of the platform further.

An important information which the participants had wished for was a full specification of the sources being searched. The reason mentioned the most was to see instantly whether the platform covers the desired providers or whether the user has to search further elsewhere. One participant said that upon this information he would decide whether it was worth for him to invest time into this platform. This information has now also been added on the homepage, after the teaser area.

The teaser area with information about the platform had been viewed by all participants. Only one of them ignored the contents completely despite scrolling them into view multiple times. The other four participants read the information carefully and found it useful for understanding the purpose and functionality of the platform. But all of them mentioned that one of the teaser tiles was difficult to read because of the light green color of the text, therefore the color was adapted to a better readable color.

Filter and search options

The participants were asked which filter options they would like to have to filter course results. Different answers had been given by the individual testers. Some wanted a filter option on course providers, others wanted to filter results by the language of the courses, when the platform covers providers with several languages. A common desire was to filter courses by their authors or sources such as a specific university and by courses offered for free.

In this context the wish emerged to be able to search for a specific author or university directly within the search input box. For this functionality the Solr search engine needs to be implemented first. Then it should be made clear to the users that such a search functionality exists, e.g. by pointing it out in the label text of the search input field and with additional hints somewhere on the homepage.

All these features could not be implemented yet, but as soon as new and complete course data is available and the Solr search engine is ready for use, the IROM application can be enhanced by these additional search and filter features, providing overvalue to the platform and its users.

Missing information on the course detail page

To analyze which additional information users could need on the course detail page, the participants were asked to mention every missing aspect that came to their mind. This interrogation brought a lot of new ideas for enhancing the crawlers and thus the value of presented course information.

The first aspect that was named by all participants was more detailed information about the price model and certification details. In particular they were interested in the requirements and possible costs for getting a certificate for absolving a course, if the possibility
to be certified existed at all (which was also mentioned as a missing detail). It was also often asked, if there were any conditions of participation, like a limited number of people, required additional material and what was demanded to successfully complete the course, whether it was only watching the videos or certain tasks had to be fulfilled.

Another important information that almost all participants demanded was the **duration** of a course, to know how much time one would have to invest to finish the course. Some of them also asked whether there was a certain start date or time slot during which the course was available. If there is no **time limit**, this also should be explicitly stated so that the user can be sure of being able to enroll in the course.

Two of the participants wanted to know who was the **specific author** of a certain course. The source specification in form of the name of the university had not been found sufficient because in certain topics they were mostly interested in one famous person or professor and would like to attend specifically his or her lectures. One of them mentioned at this point that the source could be linked to an overview of all courses from this source, which would be very helpful in her opinion. Indeed such links would represent a short path to the desired course overview instead of filtering or searching for the certain source. Such shortcuts are known to be efficient and user friendly and should be implemented wherever possible throughout the platform to provide a satisfying user experience.

To rather individual wishes belonged information about the target audience and level of **difficulty**, the **language** of the course, a **demo** of the course contents (similar to movie trailers) and whether one has to **register** on the course platform to enroll. These things were described as rather secondary information that would be nice to have and would make the presentation more interesting.

The **most desired aspects** mentioned so far are represented in figure 4.2 ordered by the frequency of their naming. Such a representation serves the purpose to show at a glance, which information should be gathered additionally by the crawlers to enhance the value of the displayed course details. This process is the most important for the entire application and has thus the first priority when improving the platform.

![Most important missing course details](image)

Figure 4.2: Most desired missing course information
General impression of the platform

Three participants found the platform **useful and interesting**. The other two testers showed no concrete interest, perhaps because they had rather little to do with MOOCs until then. Probably due to the fact that the first two participants of the study used the application without course images they found the interface rather boring. After adding random course images the impression of the further participants became more positive. Overall, interaction with the platform had been classified as **easy and intuitive**. One participant found the design tidy and not overloaded, whereas another wished for some more interesting content to engage the user in spending more time on the platform.

The **category pages** proved to be quite **popular**, the segmentation was clear to all participants and all of them explored the categories right after scanning the homepage during the first task of the study, which was to get familiar with the platform. This behavior confirms that a browsing feature is essential for a **vertical search application**, especially as a starting point for a new user.

### 4.3.2 Field study results

--- Evaluation of general numbers ---

As mentioned in the testing approach description of the empirical field study in section 4.2 only a small part of testers registered as a participant. Therefore it is quite difficult to specify how many people used the IROM application during the study period. Two different data sources exist which contain detailed information about the platform visits: the custom application database and the additional Piwik database. Both sources share common metadata and also contain individual information. Therefore some evaluation results can be compared between the two metadata collectors and others can be complemented.

The first interesting thing to analyze is how many visitors came to the platform during the study period. According to the IROM database **198 page views** (a.k.a. visits) took place from 90 unique IP addresses. But it does not mean that 90 different people visited the website, because an IP address is unique for a network, therefore several devices can have the same (public) IP as well as one user can visit the platform from different networks and thus IPs. But the number of unique IP addresses is the closest thing to individual users.

The metadata gathered with Piwik is not that complete. Only **107 visits** have been tracked, this is just 54% of all recorded visits in the IROM database. A reason for this could be that users prevent websites from tracking. Since the Piwik data is more detailed and provides a more complete picture of the users’ actions than data from the IROM database, its data is being primarily used further. From the 107 visits 36 lasted for 0 seconds. Such data has no value for the evaluation and can be therefore eliminated. This results in only **71 true application visits**, which were made by **56 unique visitors**. The Piwik tracker differentiates visitors based on the cookie information stored in the browser of a device. Thus if a user visits the platform from two different devices or browsers, she is marked as a new visitor each time. But this constraint is similar to the IP mapping described above and shall be accepted further.
4.3. EVALUATION RESULTS

A number of 56 visitors is a rather small amount for evaluating a major user interface, but considering the study length of only ten days the number is greater than expected. A further analysis of the visitors revealed that 17.9 % had returned to the application at least once and 19.6 % left the website without visiting any subpages after viewing the homepage (refer to fig. 4.3). The remaining 62.5 % used the application only once and visited at least one subpage.

A bounce rate of nearly 20 % after viewing the homepage appears to be quite high. If this rate won’t drop with further usage of the application, it must be assumed that the homepage does not arouse enough interest to explore the contents further. Then the presentation should be enhanced by showing more intriguing material. Nevertheless over 15 % of visitors came back to the application which indicates that this service has been considered useful by a group of users. The major part of the visitors probably just fulfilled the study task of exploring the platform and had no further interest in its functionality.
Figure 4.4 shows in greater detail how the 71 visits have looked like. The first section of the graph with 0 visits of subpages correlates to the jumped off visitors who viewed only the homepage of the application. But they spent about 4 to 5 minutes on average on the platform, which shows that they invested some time to explore the content. A rather small part of the made visits consisted of only 1 subpage view. In the majority of visits 2 to 6 subpages had been viewed which seems to be quite normal. These users spent 1 to 4 minutes on average, mostly proportional to the number of viewed subpages. In 20 visits the users viewed 7 to 16 subpages with corresponding visit duration of about 10 minutes and in 6 visits a sum of 19 to 29 subpages had been viewed, thus spending the most time on the platform averaging by 28 minutes.

--- GRID LAYOUT OF COURSE RESULTS ---

In order to assess how far the grid layout of SERPs influenced the users’ choice of a search result, the position of the clicked items can be analyzed. As a reminder: the grid interface consists horizontally of maximum four items, and this number decreases with narrower viewports such as on small devices. On most desktop devices two rows of results can be viewed above the fold of the viewport, meaning the visible area of a page without scrolling down. It can be therefore assumed that for most users 8 course results were completely visible without scrolling. This assumption helps to interpret the evaluation results illustrated in figure 4.5.

Of 48 search result clicks only 4 were made from mobile devices. Since the presentation of results on small screens is limited to one course card per row, it can be seen as a list rather than a grid layout. Therefore these clicks are not considered further. The data shows that the first search result was obviously of particular interest. Following with a distance of 5 clicks each are the second, third and fourth result positions. Positions 5 to 8 have gained only little interest averaging by 2 clicks. After position 8 the viewport limit comes
4.3. EVALUATION RESULTS

into play. The remaining results were probably partly (or completely) hidden due to the viewport size and the users did not want to scroll down to see further results. This can be clearly seen in the diagram where almost no clicks are present for further positions. Only three times a result from rather far down had been clicked. Of course the possibility exists that users did scroll down but haven’t found any relevant results. To analyze this more deeply the tracking of interactions with course cards on SERPs could be enhanced by their position, e.g. when tracking the opening of a course card description (through a hover of the information symbol) the gathered data could be extended by the position of the card.

The evaluation results are not that surprising, after all the SERP courses are ordered by their relevance to the user’s query. But the distribution of clicked positions does not heavily concentrate on the first or second results as it is known from linear layouts used by the Google search for example. This rather small testing data indicates that the intention behind the grid layout, namely encouraging users to explore the results more freely, has worked to a certain degree. More tracking must be done further to prove the functionality of this concept.

--- DESIGN OF THE COURSE CARD INFORMATION FEATURE ---

Empirical evaluation can give only hints rather than direct answers regarding the quality and user acceptance of certain design choices. In the case of the course cards, the gathered metadata can only indicate whether the placement of further information about a course had been discovered by the testers and if it had been considered useful. From now on, for the sake of simplicity, the action of hovering over the information icon on course cards, to reveal the description snippet, is referred to as an information view. Extensive analysis of these information views can provide more insights into their use across the application. Figure 4.6 shows several approaches for how this aspect can be examined more deeply.

Figure 4.6: Analysis of course information views
The course information has been primarily added to the course cards in order to provide more details for a search result. Indeed a major part of information views happened on SERPs, lying by 36 % of all information views and by 52 % of those that were actually followed by a course view (meaning a click on the course card to visit the course detail page), as illustrated in figure 4.6 on the left. Following with 31 % and 22 % are information views on the homepage, respectively. A smaller amount of information views happened on the course detail page from the recommendation sections and on category result pages. The scarcest sources of information view actions were users’ history pages (“Your viewed courses”), the most viewed courses page and pages of users’ saved courses. Information views on these last three places did not invoke a course view.

The diagram further shows that of a total of 174 information views only 23 resulted in a course view. This indicates that the testers used the feature rather frequently to make a decision whether the course is worth visiting the detail page and in most cases have saved their time not having visited the detail page to get further information.

The next thing to analyze is how many visits contained information view actions at all. The upper right diagram of figure 4.6 depicts the proportion of visits with and without information views. In 38 of the 71 visits people did not activate the information icon, these are 53 % of all visits of the platform. Such a high rate could mean that many users did not find the information feature, because they didn’t considered the information icon being interactive. Perhaps a more button-like design of the icon makes it more perceivable. Such an adaptation is also a good opportunity for A/B testing in the future. Furthermore some of the users of these visits might not have found any interesting courses based on the titles and thus didn’t look for further information. If the title of an item does not sound promising, there is just no need for the user to seek more details about it.

The remaining 33 visits which contain information view actions also contain 53 course views. But again less than a half of these had been preceded by an information view (compare with the lower right diagram of figure 4.6). The reason might be that in the other cases the course title had been sufficiently convincing to visit the detail page without referring to the information snippet first. Another reason could be that earlier information views had not satisfied the user and she therefore stopped using the feature. This ratio should be monitored further as soon as the information snippets have been improved (currently the description snippet is only a placeholder for future information that shall be implemented as soon as it is available).

"Go to course" button

The main purpose of tracking clicks on the “Go to course” button was to analyze, how often users request the provider page of a course. But the gathered metadata can indicate several other things. Figure 4.7 depicts the amount of these outclicks, as they are called from here on, from various points of view. The diagram on the left shows the number of platform visits in respect to the outclicks. In 15 of the 37 visits, which contained at least one course view (meaning visit of a course detail page), outclicks had been issued by the testers. These numbers suggest that in 59 % of the visits (which contained a course view) the users either did not understand the function of the “Go to course” button and therefore ignored it, or the viewed course had not aroused the user’s interest to visit its provider. It is further possible that the testers just wanted to interact with the IROM application for the testing purpose and did not want to distract themselves by following links to external pages.
4.3. EVALUATION RESULTS

To elaborate further on this data, the **32 course views from the 15 visits**, which contain outclicks, are explored in more detail in the middle diagram of figure 4.7. The **high ratio of course views with an outclick** also leaves room for different interpretations. On the one hand it could be a sign of missing important information on the course detail page. If the testers were not satisfied with the offered information they could have repeatedly tried to find it on the provider page of the course. On the other hand the users could have found so many courses that they wanted to enroll in. A third possibility is that some testers did not expect the button to lead to an external source and just wanted to explore the platform contents further.

When comparing this data with **all 65 course views** of the study (regardless of the visit type), the ratio between course views with and without outclicks radically changes (compare to diagram on the right of figure 4.7). Such a rather **low frequency of outclicks** in proportion to all course views indicates that the application fulfilled its purpose, helping the user find only relevant courses without the need of viewing numerous external course provider pages, but rather discovering the contents at one place. However, this interpretation contrasts with the assumptions made before.

Analyzing the outclicks in greater detail can provide more insights. In figure 4.8 the **15 visits that contain outclicks** are listed individually, broken down into course views with and without an outclick. 4 of these visits contain 2 course views, where one of the views
resulted in an outclick. This behavior corresponds to the desired effect of the application to help the user find the relevant course. In 7 visits only one course view had been triggered and was further followed by an outclick. Due to this small amount of course views no significant explanation can be given about the intentions of the respective outclicks, because every reasoning mentioned above could be applied to this case. The remaining 4 visits show inconsistent behavior. Whereas one of these visits contains only 1 outclick of 3 course views, which conforms to the expected usage of the platform, in 2 visits the use of this button was extremely frequent, namely 6 outclicks in 7 course views and 3 outclicks in all 3 course views. These two visits strongly indicate that the information provided on the course detail page was not sufficient for the users. The last visit to mention contains 4 course views, in 2 of which an outclick has happened. This is also difficult to interpret reliably when considering all the possible explanations.

All in all it can be said that too few visits contained an outclick, but in these the number of outclicks was probably too high considering the number of respective course views. This means that only a smaller portion of the users used the “Go to course” button, but then more extensively. More metadata must be gathered to elaborate on the assumptions regarding this behavior. But based on the information of this study it can be assumed firstly that the inhibition level to click this button was too high, and secondly that the high number of outclicks in the other cases had been caused by incomplete course information provided on the detail page.

— USEFULNESS OF RECOMMENDATIONS —

Since course recommendations are the most important component of the application after the search functionality, their quality is also being examined based on users’ interactions with their content. But the gathered click-through data is not enough for extensive evaluations, therefore only general statements can be made. Similar to the analysis of interactions with the “Go to course” button, the number of recommendation interactions, further referred to as RIs, is evaluated in relation to different aspects.

Figure 4.9: Interaction with recommendations from different points of view (RV = recommendation view, RC = recommendation click)

Figure 4.9 illustrates the proportions of RIs related to three different aspects. The left diagram shows the number of visits that contain RIs regarding all visits with course views. The RIs are further broken down into recommendation views (RVs), which mean that the user focused on the contents of the recommendation listings, and recommendation clicks (RCs), namely clicks on a course card from the recommendation listings in order to visit its detail.
4.3. EVALUATION RESULTS

In nearly half of the considered visits users interacted with the offered recommendations, whereas half of these RIs further resulted in a RC. This shows that in 43 % of the visits (16 out of 37) the recommended courses aroused the users’ interest, and the amount of following RCs proves that the suggested items had been interesting and useful for the users in 50 % of the cases (8 out of 16).

In 16 of these visits 36 course views had happened, but not in all of them the users viewed the recommendations. A total of 21 RVs had been tracked in all the 36 course views, 11 of which had led to a RC (refer to the middle diagram of figure 4.9). This means that in 58 % of course views (that were made in a visit, which contains RIs at all) the recommendation listings aroused users’ interest and 52 % of the RVs resulted in RCs. In other words, users, who made use of the offered recommendations during their application visit, did that in more than half of their course views and even followed a recommendation (clicked on a course card) in a third of these course views. Such amounts of RIs indicate that the listings were often considered useful, probably every time a viewed course seemed to be promising for the users. And roughly every second listing offered useful recommendations.

For the sake of completeness, the last aspect is the entire RI distribution over all 65 course views (compare to the right diagram of figure 4.9). Here also course views are taken into account that happened in application visits without RIs at all. In these cases the users either did not see the recommendation listings on course detail pages or were not interested in the additional material. Altogether only 32 % of course views (21 out of 65, or a third of the views) contained interactions with the listings.

Taken as a whole it can be said that the recommendations attracted users’ interest quite often regarding the visits with RIs, but only in a third of all course views. Still the rather high portion of clicked recommendations indicates their high quality. But one aspect has not been mentioned yet: the type of recommendations that were interacted with. It turns out that of all RIs, all except one happened on the listing of similar courses generated by the recommendation engine, and the one exception was a RV on the listing of courses other users also watched. There are probably two reasons for this extremely small interest in the second recommendation type. Firstly, the engine generated recommendation listing is placed above the second listing, thus the first gained naturally more views and often resulted in leaves of the page because of an RC on the listing. A second presumable reason is the rather small amount of the suggestions in the second recommendation listing (if existing at all), since often courses other users also watched just didn’t exist due to the freshness of the platform, which was released and announced a couple of days before. As soon as the platform gets more frequent visits, the second recommendation type can also be evaluated.

— FEATURES FOR LOGGED IN USERS —

Unfortunately nobody of the (newcoming) testers logged in to the platform. Only one user tried to save a course, or perhaps just wanted to check out the button functionality. Thus the usefulness of the course view history can not be analyzed based on the field study data. As to the “save course” feature, it is not clear whether the testers did not understand the functionality, did not find the login button, or just refused to log in to the platform only for the sake of trying out the feature. Since the platform is quite new and in a beta phase, the visitors might have not wanted to invest much time and effort in the use of the application because they were not sure, if it is a stable product that has a reliable future.
If this issue persists in future visits of the platform, it could be elaborated on how login functionalities can be offered and presented better. One idea is to show the user a pop-up dialog, as soon as she wants to leave the platform, with some teaser text that suggests her to log in in order to preserve her history of viewed courses. Another approach could be enabling the functionality of saving courses for not logged in users as well, so that they can play with the feature without greater effort, and again showing them some kind of a hint, that they can permanently save their courses if they log in to the platform. To reduce the inhibition level of logging in, the user could first be led to an information page about the advantages of logging in when clicking the login button, instead of directly being redirected to the Google login mask. And lastly, in future the application should offer native login for users, who don’t want to disseminate their data across multiple applications.

— Direct user feedback —

The platform offers a feedback form where the users can share their opinion about the application. In the course of recruiting testers they were encouraged to leave some feedback after testing the application, if they wanted. Two visitors indeed provided some important feedback. The feedback contained positive and negative statements regarding functionality and layout. Here are the respective snippets of the feedback messages:

- “when the search results appear and you click on one of the lectures and see the specific page, it would be good idea to have a button to go back to the search results on that page.”

- “Excellent website, I specifically like the menu on the left!”

- “the search results are not sorted by best match. If I type in ‘Scala’ the only Scala course is in the 4th place. Additional tagging might enhance that.”

- “I personally missed a button to go back to the search results when I’m on a detail page. Thus one could move between the course details and the search results much quicker.”

The first and last quotes originate from the two different users who had provided feedback. It is quite interesting, that individuals mention the same thing, and is a strong sign for an actually important aspect that concerns many users. Looking back to the lab study, this feedback is consistent with the behavior of its participants. Two of them indeed struggled to go back to the SERP from the course detail page: One eventually used the back button of the browser after hesitating for a couple of seconds, and the other asked straightly how to get back to the SERP. When evaluating the lab study sessions, it has not been considered an issue, but after receiving direct requests for the feature, its importance has got obvious. If the feature is going to be implemented into the interface, it should further be able to lead the user not only back to the SERP but particularly to the scroll position from which she came from. This way she does not have to scroll to her previous position and does not lose track of already viewed results.

The comment about the wrong order of the search results indicates that the currently used search engine is not optimal for the search task and should be replaced by the Solr search engine, as soon as the latter is ready for use. Further, the approving comment about the quality of the application and design of the category menu indicates that the user experience with the interface was satisfying and the user enjoyed interacting with it.
4.4 LESSONS LEARNED

The value of direct user feedback

Evaluating user interfaces directly in form of lab studies results in invaluable feedback about the usability of the respective application. Individuals, especially outsiders of the project, interact with the platform following their intuition. These users have no prior knowledge about the application that might influence their perception. Furthermore they don’t know about the intentions behind certain design choices and thus are not "prejudiced".

Nielsen and Molich examined in four experiments the effectiveness of heuristic evaluation, which is an informal method of analysis where usability laymen judge an interface and try to find usability issues in it [25]. A huge amount of usability guidelines exist that could be used as a basis for evaluation, but as the authors point out, such extensive analysis is highly time-consuming, therefore most heuristic evaluations are based on “intuition and common sense”. Although the experiment participants found mostly not even half of the existing usability problems on their own, aggregating all findings resulted in a greater coverage of them. The authors conclude that such evaluations are profitable even if only three to five people participate in the evaluation. To further increase the number of found problems they recommend extending heuristic evaluations with additional methods such as thinking aloud.

Short summary of the results

Following these recommendations the IROM application has been analyzed using two evaluation techniques, heuristic (actually together with the thinking aloud method) with a manageable amount of participants, and empirical with a greater number of testers. The results revealed important insights into the use of the application. The rather short time period spent on the platform by a major part of the visits could be extended by offering more interesting material. The time data can further be used as an indicator for improvement, if the average time spent on the platform increases after making adaptations to the application, as aptly pointed out by Marschall in his own recently published review of applying web analytics [23].

The evaluation also shows that the categories were quite popular among the users. In both studies the users viewed many category pages during their visits. This can mean that the layout design and structure of the contents had been appreciated by the users, but it also shows that new visitors first like to browse the contents before searching for specific topics. Furthermore, for a freshly announced platform the application gained a surprisingly high ratio of returning visitors, but unfortunately the number of shortly leaving users and visits with very few interactions was as high. The development of these numbers must be monitored further to get more solid evaluations regarding this aspect.

Regarding the specifically analyzed aspects it can be summarized that the known weak spots of the application, which are mostly caused by incomplete backend data, have been discovered by the users. As soon as these have been eliminated, a fresh and perhaps more
detailed evaluation is needed to completely analyze the concept of the application. The metadata gathered so far indicates further that the grid layout concept worked out as intended and the quality of recommendations had been approved. The main weak spot based on the evaluation seems to be the login. This topic must be elaborated on further and new testing data is needed thereafter to evaluate its value for the users. As Marschall explains, not used features are not automatically useless but could indicate poor design choices [23] regarding e.g. their placement or minor relevance and value for the user.

**Importance of web analytics**

Generally speaking, the duration of the field study as well as the number of users are not sufficient for an extensive evaluation. The aim was to conduct a first analysis of the user interface, prove that the concept works and gain valuable feedback for improvements. Thus this master thesis covers only the first of the three steps in the analytics process: First, gather metadata to reveal weak spots, then make improvements on the platform and lastly conduct evaluations again to find out, if the adaptations were successful. Therefore a major part of the future work on this project should be the implementation of suggested improvements. And after that the platform should be analyzed again, this time also considering additional aspects such as the impacts of changes on the amount of visitors, and preferably monitored over a longer time period of several months. This way the gathered metadata provides more reliable and insightful results. Furthermore, the repetition of such advanced analytics in appropriate intervals is crucial for the success of the continuously changing and extending project. The tracking basis is already implemented and a lot of metadata also for future evaluations is being gathered. In addition, adaptations and extensions of the data can be carried out easily.

As put by usability professionals during an interview mentioned in [15, p. 264], web analytics reveals details on user behavior so that the platform use is no longer a "black box" for the operators and they can detect and react on occurring issues quickly. But still, as emphasized by Hassler, any evaluation cannot be absolutely straightforward and automatically right [12, pp. 33-36]. In spite of the high number of clues provided by web analytics the evaluation remains only an interpretation of a great amount of data. Such empirically gathered metadata normally does not contain direct answers but just describes the behavior of users. Furthermore, often special cases occur such as visits that were ended abruptly just because the user had to close the browser window due to an external interference. This noisy data is always present to a certain degree and can influence or even distort the evaluation results. Therefore it should be kept in mind, that empirical evaluations reveal only trends and are not an exact science or statistics.
CONCLUSION AND FUTURE WORK

5.1 ACHIEVEMENTS

With this master thesis a new vertical search application for MOOCs has been created. The aim was to design, implement and evaluate an interaction system that conforms to modern usability standards and offers all features of a state-of-the-art search platform. Furthermore, the application shall function as a source of valuable usage data for analyzing and improving the search and recommendation engines of the application.

After extensive research on the search domain a detailed concept for the platform has been developed. The application should offer a satisfying user experience in terms of design and functionality to attract many visitors. With a high number of people interacting with the platform, its service can be optimized based on gathered usage data, thus offering the users an outstanding quality. Moreover the application should be equipped with a thorough tracking functionality in order to collect the necessary click-through data, relying on a two-system approach: On the one hand, extending "in-house" application data (which is necessary for the individual functionalities of the platform) by additional information about user interactions, such as the SERP position of a clicked course result; and on the other hand, tracking all interactions separately using web analytics software to gain a greater and more detailed coverage of the user actions with minimal programming effort.

The concept ideas have been implemented using innovative web development techniques and a modern design approach. The focus lay on an appealing, user-friendly interface that was understandable and easy to use. The application is built in a way that lets the employed engines to be easily interchanged. Thus different algorithms can be tested for their quality and efficiency. The structure of the backend data was designed to be most effective as well as easily maintainable and extendable. This ensures a smooth enhancement of the application in future. For web analytics purposes the open-source software Piwik has been implemented to track the interactions of platform visitors.

To prove that the designed concept works, the application had been tested and its quality evaluated. In order to ensure high coverage and get comprehensive valuable results, a two-phase approach had been taken: First, a heuristic lab study had been conducted with a total of five participants and a focus on qualitative analysis. Then followed a field study with (an estimated) total of 56 participants whose interactions had been evaluated using a summative empirical methodology. For this quantitative analysis Piwik data had been primarily used, due to the fact that several tracked visits did not belong to the field study.
data and could not be filtered out from the application database as easily as with Piwik, thus it could have distorted the evaluation results. As soon as the platform gains more true visits, such irrelevant data won’t have a significant impact and the metadata collected in the application database can be used more thoroughly.

The evaluation results of both studies confirm the concept functionality. Moreover, the idea behind the application has found approval and received positive feedback from the study participants. Their engagement has further motivated interesting ideas for improvement and enhancement of the platform. The click-through data gathered during the field study has also revealed many insights into the usage and provides hints on further optimization of both, the frontend interface and the backend engines.

Thus a fully functioning search application for MOOCs has been created and is ready to support students in their search for appropriate online courses. Users can also benefit from the already implemented recommendation engine and this way discover related content that they would not think of otherwise. Through the extensive tracking functionality applied within the application, valuable click-through data is being collected and can further be used to improve the quality of the employed search and recommendation engines, which is the main purpose of project IROM.
Several features have been mentioned throughout the report that have not been implemented yet, mostly due to the lack of complete course information. Other features have been demanded by the study participants for a better user experience. The following summarizes these aspects, as a reference for future enhancements.

**Getting started**

The target audience of the IROM application are students who want to strengthen their knowledge in a domain of their interest. But not every student knows more about MOOCs than the meaning of the acronym, if at all. Thus they are missing out on the great opportunity to gain free professional knowledge. To give these people an initial understanding of how such online courses work and what one has to do to enroll in a MOOC, a short introduction page could be implemented that covers the key aspects of this subject. Thus the users’ interest could be aroused and at the same time information could be provided that is necessary for understanding the entire concept of online courses in general as well as the idea and functionality of the IROM platform.

Here are a few of the often asked questions during interviews in the lab study:

- "What do I have to do to visit a course and do I get certified?"
- "Which languages are provided?"
- "Can I search for a specific university or author?"

Answering these and similar questions will help users to get started and successfully use the search service.

**Search functionalities**

Starting at the search input bar, the first opportunity for improvement is the autocompletion functionality. As soon as more user data has been collected, the feature should be extended to provide query suggestions based on the most frequent queries submitted to the application. At this point it should also be analyzed how reasonable an autocorrection functionality is. Research has shown that automatically correcting user input is not the optimal approach to retrieve relevant results, but that the correction should rather be suggested as a hint above the search results, especially if no results have been found. A famous implementation of the idea is the "Did you mean ..." feature used by the Google search engine.

To further avoid empty SERPs, the search engine could instead retrieve partial matches for the search query, so that the user does not get stuck but rather gets a way out of the dead end. If still no results can be found, the SERP could suggest similar queries or offer some popular courses instead of presenting a blank page. But the shown courses should be clearly marked as such to avoid irritating the user with seemingly irrelevant results.
The most important feature missing in this first version is the ability to filter search results. All kinds of filters are possible: General options such as free courses or a certain provider, as well as more specific ones like the author of the course or the programming language (when looking through the category of courses for programming). But such a scoped search is particularly interesting with the use of categories. The advantage of this functionality is to offer the user an ability to filter out unnecessary results for an ambiguous search query such as "python" which could refer to the programming language as well as to the reptile. Moreover this way category pages could be searched through.

**Presentation of course results**

As soon as real categories are available, they should further be used for differentiating and structuring unfiltered search results. First, the course cards should be extended by the respective category label as suggested in the design of cards on category pages. This way the search results, as well as e.g. recommended courses on the course detail pages, can be scanned easier and faster. Furthermore SERPs can appear more structured when grouping search results by their respective category. Therefore such a grouped presentation should be implemented and its efficiency and acceptance evaluated.

The presentation of course results on mobile could further be enhanced by changing their size and layout as suggested in section 4.3.1. This way more results can be viewed on the limited viewport and the user gets a better overview of the listing. The same applies to the course carousels: Here the layout of the cards should be probably maintained but their size decreased, so that at least two items can be viewed at the same time.

The last important element of the results is the information snippet of a card. Currently the displayed text is just a snippet of the course description. But the engines can provide more relevant information in the near future. This information is also a short snippet of the description, but it is chosen regarding either the search query (in the case of the search engine) or the course based on which the result has been recommended. The relevant words shall then be emphasized by using e.g. bold font face as it is often done by other high quality search services. Thus the user can better understand why a certain course is presented to her.

Speaking of course information, it should also be mentioned at this point again, which details users expect to see on the course detail page. Beside standard information about the price model, possible certification option, as well as existing time limits and duration of the course, users also wanted to see further details on the author(s) of the course, the difficulty level and preferably insights into the contents of the course. These are aspects mentioned the most frequently by study participants. Such feedback helps to adapt the crawler of the search engine in future, so that it can retrieve all desired information, as far as possible.

**Personalized features**

Since the login functionality has not been used by testers of the field study, it indicates that its presentation should be optimized. It is not clear whether the current login button is too inconspicuous or the users just did not want to use the feature. To encourage visitors to log in, several options can be pursued. One possibility is to show the user a hint with the information that she can access her history of viewed courses later on if she logs in to the application. This hint can be displayed when she views a course detail page the first time or when she wants to leave the platform.
Spinning the idea further, the “save course” feature can be used in a similar manner. If this functionality is also enabled for not logged in users, acting as a temporary track list, a hint could be displayed as soon as the feature has been used, that the list can be saved permanently, if the user logs in. Furthermore the benefits of a login could be promoted on an information page (such as the one suggested earlier) telling the visitor which additional functionalities the application has to offer.

A future version of the application should also offer native login beside the Google option. Participants of the lab study admitted that they have mixed feelings using their Google account on an unfamiliar platform. In the meantime an explanation of using this login option could be provided on the information page suggested before, where the visitor could be assured further that her Google data won’t be used or passed to third parties.

Based on the results of the lab study it was also found that users want the personal lists (history and saved courses) to be grouped by certain time periods, to get more structure into the presentation and find desired items faster. The choice of appropriate periods on which to group items is a further aspect that should be evaluated if the sectioning of these listings will be implemented.
5.3 ADDITIONAL FEATURES

Enhancing users’ lists

To follow up on the topic of the login features, in the following some more ideas on this subject are proposed. The user might benefit from the possibility of tracking her actually visited courses. By marking a course as already completed it would not be suggested to her anymore somewhere on the platform and a dedicated list of her finished courses would help her to keep an overview and not lose track. Moreover, the information about which courses have been absolved by users can be used in several ways. For example, thus the most popular courses could be determined and the list could be displayed on the homepage for interested visitors. Furthermore the individual users could get their own “wall of fame” in their personal account, where their accomplishments such as finished courses or even earned certifications are presented, as a strong motivation to achieve more.

Marking courses as finished could be further enhanced by the ability to update the status of a course the user is currently enrolled in, e.g. by recording reached milestones such as attended lectures, completed assignments or finished weeks. A structured overview of the user’s current courses with their status and perhaps even notifications via email for upcoming schedules and deadlines would be helpful for time management and would minimize the risk of forgetting about a course.

The list of saved courses could also be enhanced and personalized by providing users the ability to annotate their marked courses with personal notes. Moreover additional functionality could be offered by letting users sort their saved courses as they like and search through the list based either on the course title or the made annotations. Thus the application becomes a personal organizer which tracks courses from different sources and providers at one place.

Social tagging and rating

The application and eventually the users could greatly benefit from social tagging activities. For example the platform could let users add tags to courses in order to get a more profound knowledge about their content. The advantage of such community-based tags is the possible improvement of the search engine. People can describe the same thing in many different ways (also known as the vocabulary problem [13, pp. 24-25]), therefore the search engine could “learn” from the user-provided knowledge and thus offer advanced (rather than basic full-text) search results.

Another way of social tagging is to let users judge the difficulty level of courses. This information had been desired by the lab study participants, therefore it can be assumed that such a social rating would help users to assess a course better. But users could further get the opportunity to rate the quality of a course. Thus the obscure amount of MOOCs could be better differentiated based on their quality and difficulty. Spinning the idea even further, users could comment on the course contents and their experiences.
Of course social tagging and rating activities are not always perfectly legitimate. There are always people who deliberately provide false feedback, e.g., rate or comment on things that they actually had no experience with. To minimize the risk of such false information it could be checked whether the user at least visited the provider page of the course, or the user could be asked explicitly on that matter, before accepting any rating or tagging. But this gives absolutely no guarantee on the fact that the user actually enrolled in the course, to be capable of honestly rating its content. Since the IROM application acts only as a mediator between the user and the course provider, and has no access to the course contents or the user's data on the provider's platform, there is no way to prove whether the user visited the course or not. Thus it can never be ensured that the given feedback is justified.

But such false feedback should not occur very often to impact the overall valuation. The domain of MOOCs is characterized by mostly freely available content, thus competition does not play a major role. Students who might give unjustified feedback should also be very scarce because there is no real benefit in falsifying the rating of a course. Thus the interest in spreading dishonest feedback should not be that high to result in a meaningful impact on the entire community input.

A mobile app

Well, this idea for future work on this project is not very surprising. In our digital age owning a smartphone is probably as natural as having a tooth brush. As stated by Dave Chaffey from Smart Insights, “it’s no longer a case of asking whether mobile marketing is important, we know it is”; moreover, as of January 2017 (native) app usage dominates browsers in mobile usage in 90% of time (although this should be interpreted carefully, since social media, games and utility apps share naturally the greatest portion of the spent time)\(^1\).

A native mobile app is much more comfortable to use than a web page, even if the web page is fully responsive. Interacting with a native app is often smoother, easier and faster, not least because it is "close at hand" if residing on the home screen. Furthermore it is almost considered a status symbol for web applications to additionally provide native apps for Android and iOS. Offering native apps for both major operating systems can increase people’s trust in the platform and indicate its integrity.

In the specific case of the IROM platform, using an app can provide additional benefit for the user. For example the earlier mentioned notification feature for upcoming schedules could be realized in form of push notifications rather than emails. Furthermore such schedules could be added to the user’s native calendar app automatically, thus better integrating the courses in her daily life.

\(^1\)Mobile Marketing Statistics compilation by Dave Chaffey from Smart Insights, https://www.smartinsights.com/mobile-marketing/mobile-marketing-analytics/mobile-marketing-statistics/
“Now this is not the end. It is not even the beginning of the end.
   But it is, perhaps, the end of the beginning.”

– Sir Winston Churchill
This project has required a great deal of effort. All the more magnificent was the journey that I could experience. During the project I have obtained invaluable knowledge and have polished my professional skills – in technical, linguistic and social manners. Every obstacle on the way to the finishing line was a chance to reflect on the already covered ground and provided an opportunity to see the big picture.

The domain of web search, or specifically vertical search, is very exciting, especially in the current age of seemingly unlimited information overload. Saving users’ time by quickly finding the desired content is the main driver of innovative searching methods. People are focused on operating most efficiently and thus get quickly impatient and frustrated, if a search application does not offer relevant information. Modern search technologies have further contributed to this behavior by facilitating information retrieval through such features as intelligent autocompletion of queries and instant search results. Thus users’ expectations rise and motivate researchers and developers to come up with even better technologies. This is the natural process of evolution.

The self-learning sector is also strongly on the rise. Every day new opportunities for further education appear. Access to high quality educational material has never been as easy as in recent years. Thus it is important to share this knowledge, and providing support in finding relevant content makes the spreading of the information even easier.

The extensive evaluation of the created platform was highly enlightening. Having observed and analyzed users’ behavior has made it clear just how differently people percept certain things. The gained insights will further help me to genuinely understand users, thus creating better user interfaces and ensuring an enjoyable user experience.

The created platform provides a solid base for enhancement of the application. There probably exist countless possibilities for optimization and personalization of the platform. Developing user interfaces is a never-ending process and new features can be constantly added to provide a superior service. Therefore I am very keen on seeing further developments of the application, and also proud to be a part of the process.


5.3 Glossary

A/B testing
A/B testing is an analysis method for comparing two versions of an application feature by delivering the different versions to certain groups of users and evaluating the resulting performance 19, 52, 57, 66

Click-through data
Click-through data contains information about users' clicks throughout an interface during their visit of an application 4, 5, 21, 53, 54, 68, 73, 74

Collaborative Filtering
Collaborative (or social) Filtering refers to a method for providing automated recommendations based on taste or preferences of like-minded people 20, 21

Gestalt principles
Gestalt principles describe how parts of a visual field are perceived as connected or unitary forms, separated from the rest of the field 22, 24

Invisible Web
The Invisible Web consists of web pages that cannot be retrieved by search engine crawlers due to various constraints such as the absence of incoming links or password protection 6, 8

PageRank
PageRank is a ranking algorithm for web pages based on the analysis of number and quality of origin of the incoming links 12, 13

Rich client
A client is referred to as rich if it is responsible not only for the presentation of content but also for the functionality and logic of the application 3, 5, 27, 39

Vertical search
Vertical search describes the domain-specific search over a limited subset of sources, thus providing specialized, profound search results 1, 3, 6, 13, 14, 15, 20, 42, 62, 73

Web analytics
Web analytics refers to the collection and evaluation of web usage data to find reasons for certain user behavior and improve a web application 23, 50, 51, 52, 53, 71, 72, 73
5.3 Acronyms

Hyperlink-Induced Topic Search (HITS)
HITS (also known as hubs and authorities) is a ranking algorithm for websites based on link analysis and categorization of pages as hubs (pointing to many authorities) or authorities (providing valuable content) 12

Intelligent Recommender of MOOCs (IROM)
Project IROM was initiated by Yingding Wang to support students in their search of appropriate MOOCs on a cross-provider basis through an intelligent search and recommendation platform 3, 4, 5, 14, 15, 17, 18, 19, 20, 21, 22, 23, 24, 25, 27, 32, 35, 37, 39, 42, 45, 50, 53, 55, 60, 62, 66, 71, 74, 75, 78, 79

Keyword-in-context (KWIC)
KWIC is a type of query-oriented description snippets for search results that show the query keywords in their document context, often highlighted by a bold font-face or different color 17, 45

Massive Open Online Course (MOOC)
MOOCs are (mostly) free and open online courses, provided by universities and institutions, with an unlimited number of participants and access from anywhere at any time over the web 2, 3, 17, 21, 22, 27, 29, 35, 37, 45, 53, 55, 61, 73, 74, 75, 78, 79, 88

Representational State Transfer (REST)
REST is a way of enabling communication between distributed computer systems over the web using standardized techniques such as HTTP requests and a common exchange format for data such as JSON and XML 5, 28, 29, 40, 41

Search engine result page (SERP)
SERP is a common abbreviation in the search engine domain used to describe pages that contain primarily results for a submitted search query 16, 17, 18, 19, 20, 24, 30, 31, 32, 35, 38, 43, 55, 57, 64, 65, 66, 70, 73, 75, 76