University of Cape Town

Masters Thesis

Designing an interface to allow users to sort, filter and search web-based annotations.

Author:
Nicola du Toit

Supervisor:
Professor Gary Marsden

A thesis submitted in partial fulfilment of the requirements for the degree of Master of Philosophy in Information Technology

in the

Department of Computer Science

February 2014

Declaration of Authorship

I, Nicola du Toit, declare that this thesis titled, 'Designing an interface to allow users to sort, filter and search web-based annotations.' and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed: _____

Date: _____

"All of Computer Science is a subset of Human-Computer Interaction."

Gary Marsden

Abstract

Systems to annotate online content are becoming increasingly common on the World Wide Web. While much research and development has been done for interfaces that allow users to make and view annotations, few annotation systems provide functionality that extends beyond this and allows users to also manage and process collections of existing annotations.

Siyavula Education is a social enterprise that publishes high school Maths and Science textbooks online. The company uses annotations to collate collaborator and volunteer feedback (corrections, opinions, suggestions) about its books at various phases in the book-writing life cycle.

Currently the company captures annotations on PDF versions of their books. The web-based software they use allows for some filtering and sorting of existing annotations, but the system is limited and not ideal for their rather specialised requirements.

In an attempt to move away from a proprietary, PDF-based system Siyavula implemented Annotator (http://okfnlabs.org/annotator/), software which allowed for the annotation of HTML pages. However, this software was not coupled with with a back-end interface that would allow users to interact with a database of saved annotations.

To enable this kind of interaction, a prototype interface was designed and is presented here. The purpose of the interface was to allow users to easily filter, search for and sort through a web-based collection of annotations about Siyavulas online content.

Usability tests demonstrated that the interface was successful at giving users this new and necessary functionality to process annotations.

Once integrated with front-end software (such as Annotator) and issue tracking software (such as GitHub) the interface could form part of a powerful new tool for the making and management of annotations on the Web.

Acknowledgements

The completion of this dissertation would not have been possible without the endless love, support and patience of my partner Ewald, my parents Jill and Pierre and my extended family and friends who are too numerous to name individually.

Thanks are also due to my employers and colleagues at Siyavula for giving me the time and space to do this research.

Contents

List of Figures

List of Tables

Abbreviations

LAH **L**ist **A**bbreviations **H**ere

This dissertation is dedicated to the memory of Gary Marsden: the finest supervisor and mentor a student could wish to have.

Part I

Introduction

Chapter 1

Introduction

1.1 Introduction

Siyavula Education is a Cape Town-based social enterprise that creates open source high school maths and science textbooks. The books are written collaboratively by volunteers and then edited and refined in-house. The volunteer community plays a central role in the book making process. Volunteers contribute towards authoring new content, editing existing content, proofreading and translation. The final versions of the books are available as printed hard copies, PDF downloads and as web books, which can be read online using a variety of devices.

One mechanism by which volunteers, or any member of the public, can give Siyavula feedback is via annotations. This is not only a tool for flagging small errata in the books; Siyavula also uses annotations to get feedback from volunteers during the authoring process, and during the editing and proofreading phases.

1.1.1 Current annotation system

Currently, Siyavula uses the web-based software A.nnotate.com[1]. To use this system the company must upload draft PDFs of a particular chapter of a book, and external users (who have been given the necessary permissions) can then view, highlight and make comments on the content. Users can select a particular type for their comment (error, comment or suggestion) and they can also add custom tags to comments. Any user who has access to a particular PDF can view and reply to all annotations made on the document.

[1] http://a.nnotate.com/

A.nnotate.com is not very easy to use for volunteers who need to make annotations: pages are often slow to load (especially for large documents, and a single chapter of a book may be anywhere between 20 and 90 pages long) and the interface is not intuitive, particularly for less advanced computer users. Similarly, it is not user-friendly for the Siyavula team members who have to process annotations made.

'Processing' an annotation involves an employee locating an annotation in the context of a book (or subject or grade), assessing its validity (e.g. "Is there really an error in the text as flagged by a volunteer?"), making changes to the book content's source code if necessary and somehow marking that annotation as resolved.

To do this currently, employees have to trawl through the uploaded PDF documents one page at a time to view annotations in the context in which they were made. Whilst A.nnotate offers basic filtering, searching and sorting of notes, this functionality is predetermined (and limited) and not customised to Siyavula's workflow. For example, there is no efficient way to mark annotations as resolved or to lock down a particular annotation and its replies (one can only prevent access to the entire document). It is not possible to view annotations with a preview of the the text to which they relate and it is not possible to group annotations by subject (e.g. Maths), type or username, or to cross-reference annotations between different PDFs. There have also been problems with old annotations simply being deleted from the a.nnotate.com database.

1.1.2 New annotation system

Due to the limited functionality of a.nnotate.com (software arguably not designed for the kind of functionality that Siyavula requires from it); its proprietary nature (one has to pay to upload documents over a certain file size); and the fact that it can only handle PDF documents, it was decided that Siyavula would implement its own annotation software on the company's websites. This would allow Siyavula to develop the software according to its own rather specialised needs and to capture annotations on (HTML) web versions of the books, not merely PDFs.

For external users, the beta version of Siyavula's new annotator behaved in much the same way as a.nnotate.com, albeit with a simpler, cleaner interface. The software allowed users to highlight text in a static webpage and make an annotation about that text, in one of three categories: "errata", "comment" and "suggestion".

These annotations were then stored in a database, and could be viewed by employees in a table with the most recent annotations listed first.

1.2 The problem

The very limited back-end interface (a single table) provided by the new annotation software did not include any functionality for Siyavula team members to filter, search, sort or process annotations that have been made. Users could scroll through the contents of the table, but had no tools whatsoever to manipulate or interact with the information provided.

Questions that needed to be answered before a new interface could be designed included:

- How do different internal users need to interact with existing annotations?
- What new functionality do users require?
- How would users want and expect the interface to look and behave?

1.3 The solution

The solution to the above problem was to develop a new interface customised to Siyavula's requirements, that would make it easy for team members to filter, search for and sort existing annotations.

This would enable users to locate sets or subsets of annotations, to find individual annotations, or to view particular details about a single annotation or user, all previously impossible tasks. Such functionality could streamline the ways in which team members process and resolve annotations made by volunteers and external users.

A user-centred approach was adopted to identify and answer specific questions about user requirements and to include user feedback in as many stages of the design process as possible. Once the high-fidelity prototype was complete, user-centred evaluation was also undertaken in order to determine whether or not the final prototype properly met user requirements and expectations and adequately provided them with new and desired functionality for processing annotations [1, p. 327].

1.4 The structure of this dissertation

Chapter 2 provides more technical details about Siyavula's annotation software and its functionality. Chapter 3 provides an overview of existing research in this field. Chapter 4 outlines the user-centred methodology, tools and design guidelines used in the development of this interface. Chapter 5 deals with the design process while Chapter 6 covers

the technical details about development of the high-fidelity prototype. Chapter 7 deals in detail with the process of user-centred evaluation and iterative improvements to the interface. The success of the final interface and future work is discussed in Chapter 8.

Chapter 2

Background

2.1 Overview of Annotator

The beta version of Siyavula's Annotator software allowed people to highlight HTML text in a webbook and make an annotated comment about their selection. In order to use the annotator, users had to be be logged in so that their annotations could be correctly attributed to them.

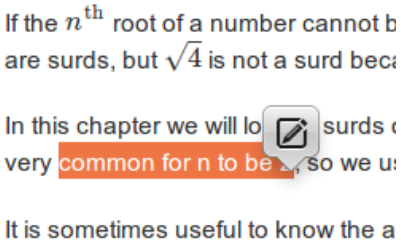


FIGURE 2.1: Highlighted text with the "Annotate" icon

Annotations could be made in one of three categories: errata, suggestion and comment.

They were then stored in a database for future reference. All users could view existing annotations on a webpage when they logged in to the annotated version of the webbook. Users could also reply to existing annotations made by themselves, or other users.

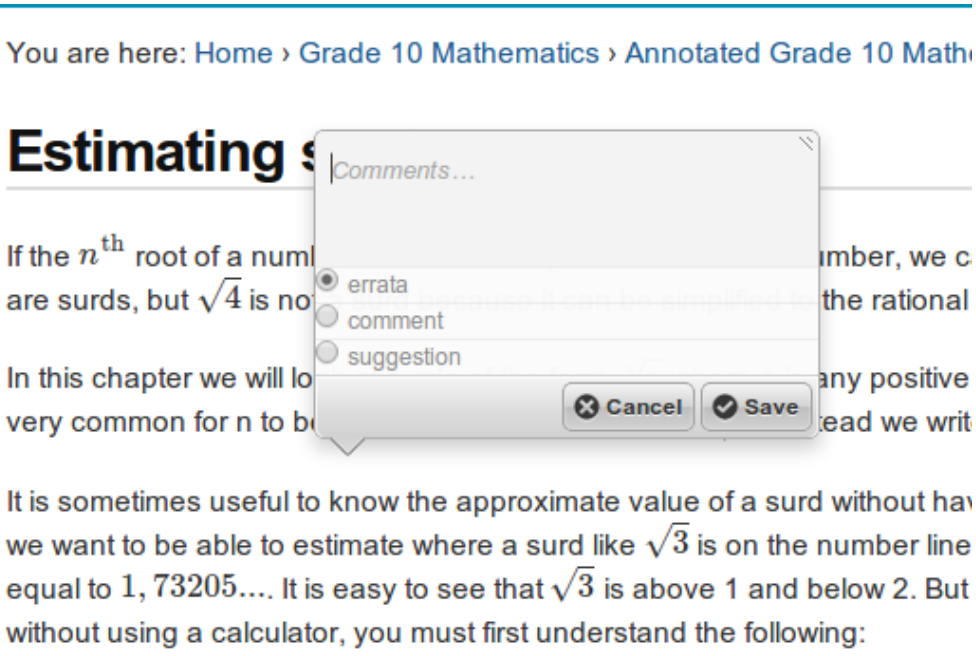


FIGURE 2.2: Three categories of annotation

2.2 Technical details of Annotator

The annotation software Siyavula decided to utilise (and modify for the company's own purposes) is based on an open source application called "Annotator" [1] (forked by Siyavulas at https://github.com/ezietsman/annotator). Annotator is a Javascript application that is written in Coffeescript. It uses a Couch database and has a Python server back-end. It does not come packaged with any functional user interface for the viewing of saved annotations outside of the webpage where they were made.

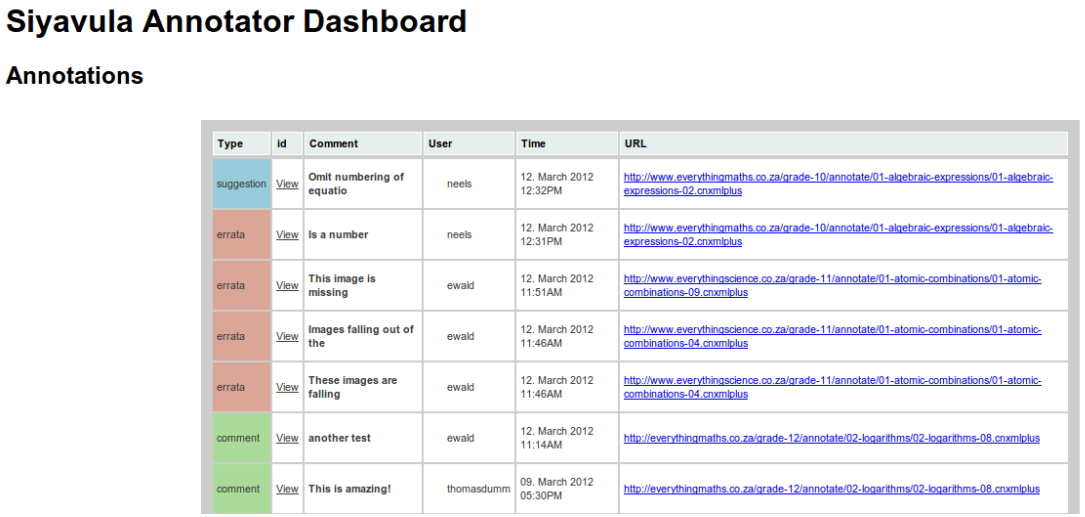
Siyavulas version of Annotator ran on static versions of the companys webbooks, not on the latest live versions.

2.3 Existing Interface

The backend interface for the annotator consisted of a simple table list of annotations.

Table columns included: "Type", "id" (the database primary key), "Comment" (the text typed by the user), "User", "Time" (date and time) and "URL" (the webpage on which the annotation was made.) Most recent annotations were listed first. The type of annotation was also indicated by one of three colours in the left hand column.

[1] http://okfnlabs.org/annotator/



Siyavula Annotator Dashboard

Annotations

Type	id	Comment	User	Time	URL
suggestion	View	Omit numbering of equatio	neels	12. March 2012 12:32PM	http://www.everythingmaths.co.za/grade-10/annotate/01-algebraic-expressions/01-algebraic-expressions-02.cnxmlplus
errata	View	Is a number	neels	12. March 2012 12:31PM	http://www.everythingmaths.co.za/grade-10/annotate/01-algebraic-expressions/01-algebraic-expressions-02.cnxmlplus
errata	View	This image is missing	ewald	12. March 2012 11:51AM	http://www.everythingscience.co.za/grade-11/annotate/01-atomic-combinations/01-atomic-combinations-09.cnxmlplus
errata	View	Images falling out of the	ewald	12. March 2012 11:46AM	http://www.everythingscience.co.za/grade-11/annotate/01-atomic-combinations/01-atomic-combinations-04.cnxmlplus
errata	View	These images are falling	ewald	12. March 2012 11:46AM	http://www.everythingscience.co.za/grade-11/annotate/01-atomic-combinations/01-atomic-combinations-04.cnxmlplus
comment	View	another test	ewald	12. March 2012 11:14AM	http://everythingmaths.co.za/grade-12/annotate/02-logarithms/02-logarithms-08.cnxmlplus
comment	View	This is amazing!	thomasdumm	09. March 2012 05:30PM	http://everythingmaths.co.za/grade-12/annotate/02-logarithms/02-logarithms-08.cnxmlplus

FIGURE 2.3: The existing back-end interface: a simple table of stored annotation data

Columns were not sortable, and there was no search or filter mechanism. The back-end merely provided a list of annotations, with some basic information about each entry. For in-house processing of small numbers of annotations (< 20), this basic back-end was adequate (albeit very rough) for advanced members of the team. However, this interface was not scalable (processing > 100 annotations listed like this would be extremely difficult) and not user-friendly at all, particularly for less technologically advanced employees. Indeed, it had not been designed with any user requirements in mind at all - it had merely been cobbled together as a temporary way to view annotations that had been made.

Given the simplicity of the existing interface and the potential for a functional annotator to be an integral part of many stages of Siyavula's book-writing and -maintaining process, it was decided that there would be immense value in designing a unique back-end interface. This interface could be tailored to accommodate different team members' specific user requirements and therefore help to maximise the use of the annotator as a functional tool for making, storing, processing and resolving annotations containing volunteer and general feedback.

Chapter 3

Related Research

3.1 Overview of Annotator

The Concise Oxford English Dictionary describes an annotation as an explanatory note added to a book or document [2]. Haslhofer et al [3] extend this and state that an annotation can be seen as "a remark, explanation or interpretation added to the original document". According to Ovsiannikov et al [4] annotations may take the form of written notes, a symbol, a drawing or, in the digital space, a multimedia clip.

Analogue annotations and marginalia in books and other hardcopy documents have a long tradition [3] and come in a variety of formats (some more formal than others) including handwritten notes in margins, printed margin notes in textbooks and Post-it notes stuck on to content. More recently, with the inevitable shift towards digital reading, annotations have become possible in desktop software suites, on e-reading devices such as Amazon's Kindle, and on the World Wide Web.

Much research has been done into the different types of annotations that exist [5] [6], the workflows by which they are created, and their purpose and usage particularly in the digital realm [7] [4]. Agosti et al. [7] name three major uses for annotations: to create new information resources, to interpret existing ones, to access resources in new ways, and to support the effective use of resources. Arko et al. [8] state that annotations also allow for community engagement and the capturing of "ephemeral information" that would otherwise be "lost in transient media like conversation" or emails. This is applicable to analogue annotations, but is particularly true of digital annotations, which can so easily be shared, viewed and processed online.

While desktop software to make and read annotations (or "notes" or "comments") has been around for years (e.g. MSOffice [9] for Microsoft documents, and Adobe Reader

[10] for PDFs), a multitude of new technologies are becoming available due to increasing interconnectivity and new storage options given to us by the Web. Examples of online annotation software include Google Drive [11] (for documents and spreadsheets), A.nnotate.com [12] and AnnotateIt [13].

As online annotation possibilities have expanded, a number of frameworks have been developed to try and standardise the ways in which annotations can me made, stored and manipulated, particularly on the Semantic Web [14] [15]. Notable examples of such frameworks include Annotea [16], CREAM [17] and LEMO [3].

A number of online annotation systems have emerged out of these frameworks. The vast majority are concerned with creating and viewing annotations. Only a handful go beyond this and deal with annotation management. Many of these front-end systems have been analysed and compared extensively by Kahan et al [16] and Haslhofer et al [3]. To avoid exhaustive repetition, only those systems that are web-based and that include some functionality to manage or process annotations that have already been made, will be discussed here.

Amaya [18] is W3C's test-bed web editor/based that includes an implementation of Annotea, a collaborative annotation system. Annotea/Amaya allows users to make and view annotations in a webpage. Some extra functionality is provided via a dropdown menu which allows users to reply to existing annotations, or to delete them [19]. Mozilla's instance of Annotea, Annozilla [20], provides the same options. Beyond merely creating and viewing annotations, users can also reply to them and delete them.

This is very similar to the Google Drive "comments" system [11]. It is standard today for systems that integrate annotations to allow users to write (and edit) view, reply to and delete annotsations. Google Drive adds one more piece of functionality to this which is to mark annotations as "Resolved". This hides the annotations, but they can still be viewed in the document history, and restored if need be.

A.nnotate [12] (the software that Siyavula currently uses to annotate PDF books) offers users some processing of annotations or "notes". Apart from browsing annotations one page at a time, in the context of the PDF, users can also view all notes made on a PDF. They can then sort the annotations displayed by date, subject, tag and document. It is possible to search for text in annotations, and filter by tag, and a few predefined options such as "Include all notes/notes on text/notes on images". In addition, users can export annotations as CSV files.

The Open Knowledge Foundation's Annotator (upon which Siyavula's annotation software is based) [21] offers a simple and user-friendly front-end system which allows users

create annotations on any website. Although it comes packaged with a hosted web service for storing annotations (AnnotateIt [13]) or a customisable storage API [22], neither of these options provides functionality to process existing annotations.

The Debora (Digital access to Books of the Renaissance) interface [23] is unfortunately no longer functional online [24]. The original interface did go beyond merely displaying annotations: it allowed users to "chain together" paths of annotations, and then group those in "virtual chapters" and "virtual books" [23]. They did so to help users navigate between different content and annotations. This is surely one of the earliest online examples of an annotation interface giving users additional power to manipulate existing annotations.

Mojiti [25] is also no longer available online but can be accessed via the Internet Archive [26]. It allowed users to annotate videos online. Beyond this, it also allowed users to share their annotated videos by sending a link to the data or embedding it. This service has arguable been replaced by Google's YouTube Video Annotations [27], which provides the same functionality today. Users can make and view annotations in video content, but YouTube provides no annotation-specific functionality beyond this.

Vannotea [28] is annotation software for audiovisual content, built on Annotea. It provides users with basic search and filter functionality of existing annotations. Users can filter annotations based on their associated metadata (e.g. author or date), and also perform a basic search for keywords in one or more metadata fields [29]. Additionally, Vannotea provides a timeline that corresponds to the length of the audiovisual material, and Annotea annotations are surfaced (as icons) on this - allowing for easy visual browsing.

Like Annotator, the Mozilla Firefox plugin WebAnnotator [30] [31] also allows for easy annotating of the web. The only extra functionality it provides though is the option to save and export annotations.

The current version of Yawas [32] allows users to highlight content in a webpage, which is then stored as Google Bookmarks [33]. Users can add tags to saved bookmarks and can search for them as they can for any saved Google bookmark. The original version of Yawas [34] also allowed users to search for existing annotations and to import and export them.

Whilst there are many interfaces that enable users to make and manage annotations, it is evident that only a handful provide extended functionality beyond simply making, viewing and deleting annotations. Parts of the web annotation problem have been resolved in different projects, but there is not one outstanding system that presents a comprehensive solution. For example, the interface to make, view and edit annotations

in Google Drive is hugely successful, but it is not coupled with comment filtering functionality. A.nnotate provides better filtering and sorting capabilities than most, but it is limited to PDF documents and can not handle HTML webpages.

One noteworthy project that will enter (and possibly dominate) the online annotation space soon is Hypothes.is [35]. The beta version of the Hypothesis annotation system is very promising, and in the near future this software may well present a solution to the problems involved with annotating the web. However it is not yet clear whether it will offer back-end processing functionality, or merely a very elegant solution to creating and viewing annotations online.

It is apparent that further research needs to be done into emerging (and increasingly complex) workflows for managing existing annotations. It is possible that Siyavula's "processing" workflow involves new use cases for annotations, and that there are others like it that are undocumented. It is also apparent that a research opportunity exists for the investigation of functionality and design of an interface specifically tailored for annotation management. It is time to starting thinking (and developing) beyond how annotations are made, and to start exploring what can be done with annotations that already exist.

Part II

Design

Chapter 4

Methodology

4.1 Introduction

In order to design a prototype interface that would meet users' needs and solve the problem of being able to sort, filter and search for annotations, a user-centred design process was undertaken. This included establishing user requirements; analysing those requirements to understand job roles and to build a conceptual model of the system; and researching design principles and guidelines to inform the development process. Participatory design sessions were then held in order to create a paper prototype of the interface. This was then converted into a high-fidelity prototype, which was formatively evaluated via usability tests. Improvements were made to the interface based on this evaluation, and then summative evaluation took place to determine whether or not the finished product met the original requirements and could be deemed a success.

4.2 User-centered design

User-centered design (UCD) is a design philosophy and framework that aims to include end-users in as many stages of the design process as possible [36]. The purpose of this inclusion is to use real user tasks and goals to drive development in order to design systems that are relevant to users and that adequately address their needs [1, p. 327].

Gould and Lewis [37] outline three principles which are now standard tenets of the user-centered approach, namely: early focus on users and tasks, empirical measurement and iterative design [1, p. 327].

The UCD framework includes several well-established methodologies designed to capture the abovementioned principles in the development process. These include establishing

and analysing user requirements (e.g. by conducting user interviews, observing existing workflows, holding focus groups), prototyping (which could take the form of conceptual and/or participatory design) and iterative evaluation (and improvements to the design), during which users are asked to test and provide feedback about the ongoing product [1, p. 330 - 331].

Team participation (in planning, brainstorming and development) is a core part of Siyavula's corporate culture, and because the system was being designed for a niche group of users, a heavily user-centered process was deemed particularly appropriate in this context.

The design process began with establishing user requirements by holding stakeholder interviews and having a focus group to determine use cases and requirements for the new interface. Once this was complete, a participatory design process took place, during which users were actively involved in brainstorming conceptual and actual physical designs on paper. The outcome of this process was then developed into a high-fidelity prototype, which was iteratively tested (formatively and summatively) and improved according to user feedback and assessment.

4.3 Establishing requirements

4.3.1 Stakeholder identification

The stakeholders were divided into three broad categories of job role: "Development", "Production" and "Sales". These three categories corresponded to three different internal teams of employees, but were also generally correlated to computer experience levels. Whilst all employees were fully computer literate, those in the "Development" category were the most advanced users (professional programmers and command line experts). "Production team" users had above-average (for the team) computer skills (basic programming, familiarity with markup languages like XML and LaTeX) and "Sales" users had average computer skills, being fully proficient with computers and the Internet, but unfamiliar with command line operations and programming or markup languages.

4.3.2 Stakeholder interviews

Traditional contextual interviews (including observations) were dismissed as a viable option because the annotation software was still being developed and there was no existing workflow to observe [38, p. 38]. Alternative annotation systems used by employees

(such as a.nnotate.com) were markedly different to the system being developed (they also had no back-end mechanism by which users could process annotations) so there was no analogous, precedent workflow that could be used in a contextual interview.

Instead, stakeholder identification and one-on-one stakeholder interviews were held in order to identify user requirements. Because of the small group of users involved, interviews were a practical means of getting individual feedback from every possible user of the system. A fairly open-ended interview process allowed for individual discussion and discovery about the varied ways in which different team members would interact with the system.

4.3.3 Focus group

Once the interviews were complete, an informal focus group was held with all users to clarify points raised in the interviews and finalise the user requirements identified in the requirements analysis process [1, p. 365].

The focus group enabled users to exchange and clarify ideas and issues arising from the interviews and also collectively reach consensus on their requirements. As well as being a mechanism to obtain detailed user input, interviews and brainstorming allowed users to feel that they were participating actively in the design process, to voice any concerns and express their wishes for the future annotation system [1, p. 365].

4.3.4 User requirements analysis

During stakeholder interviews and in the focus group, individuals were asked the general question "Why annotate?". Several answers emerged, including:

- To improve existing products (post-publication)
- To collaboratively build new products (pre-publication)
- To involve community at all stages of product development

Stakeholders agreed that annotations must have a type and associated priority (by date e.g. old errata are most important); and status (e.g. new, open or resolved). Discussion and replies on a resolved annotation should not be possible - either these annotations must be hidden, or locked down.

Annotations should be assignable to team members (e.g. certain chapters could be pre-assigned to certain team members) and re-assignable (e.g. for specific questions, or to

get a second team member to double check a processed annotation before it is marked as resolved).

Stakeholders agreed that annotations should be sortable (by date, type, location, status, priority, user) and browsable (by subject, book, chapter etc.) and that it should be possible to build a query based on the latter categories.

Some stakeholders suggested that push notifications via email would be useful. Internally, these could notify team members if there was a new annotation in one of their pre-assigned chapters, for example. Externally these could notify an external user if there was a reply to their annotation e.g. requesting clarification; or to notify them that it has been resolved.

In terms of viewing existing annotations, stakeholders said that they would like to be able to view all information associated with an annotation (including location in book, highlighted text, user, user comment etc). They also agreed it would be useful to view an annotation in the context of the webbook in which it was made (to see the highlighted text); to be able to move easily from one annotation to the next; and to be able to easily view all annotations made in a given section of a webbook.

4.3.5 Use cases

The three job role categories mentioned above also generally correlate to three different types of use case, which emerged from the stakeholder interviews. "Development" users are seldom involved in the "processing" of annotations. They tend to work with the actual development and maintenance of the annotator software. "Production" users are typically involved in the bulk of annotation processing. They would be the most frequent and intense users of the new system, needing to do batch processing (viewing, searching, filtering etc.) per subject or per chapter. "Sales" users are less intensive users who typically need to locate a particular annotation made by a specific volunteer, in order to provide feedback to that volunteer on the status of an annotation (e.g. "*has an error been fixed or a suggestion implemented?*"). These use cases overlap in a complementary manner. "Sales" requirements are a subset of "Production" requirements, and "Developer" users could slot into either of these use cases if and when the need arises.

4.4 Conceptual model

From the the stakeholder interviews and focus group a conceptual model of the interface emerged. Johnson and Henderson [39, p. 27] describe a conceptual model as a "high-level description of how a system is organised or operates". Conceptual models provide metaphors and analogies that convey understanding about what a system is for and how to use it. They also include the concepts that users will encounter when using the system, the relationships between said concepts, and information about the mappings between the concepts and the task domain the system is supposed to support [1, p. 40-41]. The benefits of constructing a conceptual model before design begins include a simpler, more coherent end-product, and a better match between user expectations and design intentions[39, p. 26]. Conceptual models can also be used to guide and inform the design process, keeping it as close to the original task requirements and user domain space as possible[1, p. 40].

Using Johnson and Henderson's task-based conceptual model[39, p. 30], it was possible to view each annotation as an object with attributes. Attributes for each would include subject, grade, chapter numer, highlighted text, username, user comment and a timestamp. Tasks would include grouping objects by attributes (e.g. annotations can be grouped per grade or subject) and combinations thereof. Other tasks or actions to be performed on annotation objects would include filtering by attribute, sorting by attribute and searching by username.

In terms of metaphors and real-world analogies, searching for a particular item via a process of refinement occurs in many places in reality and on the internet. The binary sort performed to locate a word in a hard copy dictionary or telephone directory is a good analogue example of this. Online, all users were familiar with searching via refinement from Gmail (e.g. filter by label, or search inbox for a particular sender), GitHub (e.g. filter issues by label), Google, Flickr and Wikipedia, to name but a few. Users with programming skills were also familiar with the design of various kinds of sorting algorithms.

Several annotation attributes (by which annotations could also categorised) map directly to print book schema: subject, grade and chapter are a common hierarchy used to map high school textbook publishing. Annotations themselves could be viewed as being post-it notes, made (and stuck) in a particular book, about a particular section or page of text. Those post-it notes could then be categorised, depending on where and when they were made - much like a paper library indexing system.

4.5 Design guidelines

In addition to the conceptual model, standard design principles, guidelines and rules were also used to inform the design process, as well as standard interface design and behavioural patterns. Design principles represent a mixture of theoretical knowledge, practical experience and common sense[1, p. 26]. Although design principles are usually fairly high-level and abstract they provide well-established suggestions to designers about best practices to follow and include in their work. The principles and rules used to guide the current design process are documented in depth in the literature; however a brief summary is provided below.

Rogers, Sharp and Preece[1, p. 26-29] suggest five design principles to follow, namely:

- **visibility**: the more visible functionality is, the more likely it is that users will know what to do next.
- **feedback**: giving users feedback about what they have done or what has happened allows them to continue with the activity.
- **constraints**: restricting the kinds of interaction that can take place at any point in time.
- **consistency**: design interfaces to have similar operations and use similar elements for achieving similar tasks.
- **affordance**: give objects attributes in such a way that users will know how to use the object.

Similarly Dix and Finlay[40, p. 260] outline:

- **learnability**: the ease with which new users can start interacting with a system and know how to use it (including predictability, synthesizability, familiarity, generalisability and consistency).
- **flexibility**: how many ways in which a user and a system can effectively exchange information (including dialog initiative, multi-threading, task migratibility, substitutivity and customizability).
- **robustness**: the level of support provided to the user in order for them to successfully assess and achieve goals (including observability, recoverability, responsiveness and task conformance).

No design work can be undertaken without acknowledgement of Shneiderman's *Eight Golden Rules of Interface Design*[41]:

1. Strive for consistency.
2. Cater to universal usability.
3. Offer informative feedback.
4. Design dialogs to yield closure.
5. Prevent errors.
6. Permit easy reversal of actions.
7. Support internal locus of control.
8. Reduce short-term memory load.

Similarly, Nielsen and Molich's *10 Usability Heuristics for User Interface Design*[42, p. 249] (discussed at length in Chapter 7) were included in the process.

4.6 Participatory Design

Once the requirements gathering and analysis process was complete and a conceptual framework and design guidelines were in place, participatory prototyping sessions could be undertaken, to integrate as much user input as early on in the design process as possible.

The participatory design model provides a sramework in which designers and users (and any other stakeholders) can collaborate during the design process. It allows for ongoing user input and the inclusion of user expertise and knowledge. Prototyping is a participatory design method by which designers and users can research, explore and start iteratively designing together [43].

Paper prototyping was chosen because it allowed for creative, ongoing user involvement and an opportunity to build on the research obtained through the stakeholder interviews. Prototyping (and the dialogue that it involved) gave participants the opportunity to refine their initial ideas and to move from hypothetical discussion and conceptual design about their future use of the system to concrete functionality and paper mockups. It was also selected because it allows for easy visualisation and exploration of the interface features, using tools (papers and coloured pens) that all participants are familiar with[44, p. 380].

Ongoing user input was deemed particularly valuable given that there existed no precedent for the back-end interface in question, and that future users had rather specialised user requirements for the interface. Additionally, all stakeholders (even those in the "Sales" category) were computer literate and familiar with a variety of web-based software, and therefore able to offer informed opinions about their work processes and needs.

Two 1.5 hour paper prototyping sessions were held with a representative sample of team members, who had different job roles and work experience, and could therefore bring a variety of input and opinions to the design. The first session focused only on how users would expect to be able to search or filter annotations (not on how those annotations or search results would be displayed). The second session focused on the displaying of annotations once a user had implemented a search or set of filters.

The aim of these sessions was to produce a paper prototype of the interface design that included search/filter functionality annotations, and an interface for displaying filtered results.

Once a paper-prototype existed, this could then be converted into a high-fidelity prototype which could be iteratively evaluated as mentioned above. The design and evaluation process is described and discussed at length in the chapters that follow.

Chapter 5

The Participatory Design Process

Four participants (from different job roles and with varying experience) were selected for the prototyping sessions. Two participants had seen the existing annotator back-end (the basic table) before. These same two participants were familiar with the process of making and processing annotations. The other two users knew in theory how that process worked, but had never taken part in it.

This group of participants was selected from a team of 12 members, to be as representative a sample as possible, from the team's (very small) population of users, in order to obtain as varied and balanced user input as possible. Participants were representative of differing job roles, computer literacy levels and in-house experience, particularly in terms of previous experience with the annotation process. Several other users were earmarked early on as being a similarly representative sample who would take part in usability testing, further into the design process.

Consent was given by all participants for the filming of the two sessions. Participants were provided with large pieces of paper, coloured pens, highlighters and coloured paper, with which to sketch out their ideas.

It was made clear to the participants that the interface in question would involve the back-end processing of annotations only, and would only be used by internal team members, once annotations have already been made and stored in the database. At this point the only web-enabled device considered was a browser running on a desktop or laptop computer as it is highly unlikely that team members would need to process annotations on a mobile device, for example. It was assumed that all external users (those who make the annotations and those in-house users who process them) would have to be logged in, due to existing functionality on the book websites.

The first session focused only on how users would expect to be able to search or filter annotations (not on how those annotations or search results would be displayed). In other words, "*what do you expect to see when you log in and what kind of search functionality do you want?*"

The second session focused on the displaying of annotations (or query results) once a user had implemented a search or set of filters, i.e. "*how do you sort and refine results?*". Discussion and ideas about this latter concept refined the ideas that emerged in the first session. As a result, part of the second session also involved consolidating and finalising ideas for a final paper prototype.

5.1 Session 1

5.1.1 Filter parameters

In the first session, users initially listed all possible parameters by which they may need to search or filter annotations. Suggestions included:

- subject
- grade
- alignment [curriculum (CAPS/NCS)]
- user (who made the annotation)
- chapter
- category (errata/comment/suggestion)
- keywords (in user comments AND in highlighted text)
- date (before a date/after a date/chronologically)
- resolved (fixed/not - status)

These filters were divided into two groups by users: high-level (subject, grade, alignment and chapter), and low-level. The high-level filters are those that users believed would be most frequently used.

Users agreed upon two possible workflows for processing annotations: one would be to go through annotations chronologically, in the order in which they were made (e.g. "*today I'm going to look at annotations, from most recently made to oldest*"). Another would

be to process annotations made in a particular book and/or chapter (e.g. "*today I'm going to look at all the annotations in Grade 10 Maths, Chapter 1*").

5.1.2 First interface

Initially, a basic drop down menu structure was suggested:

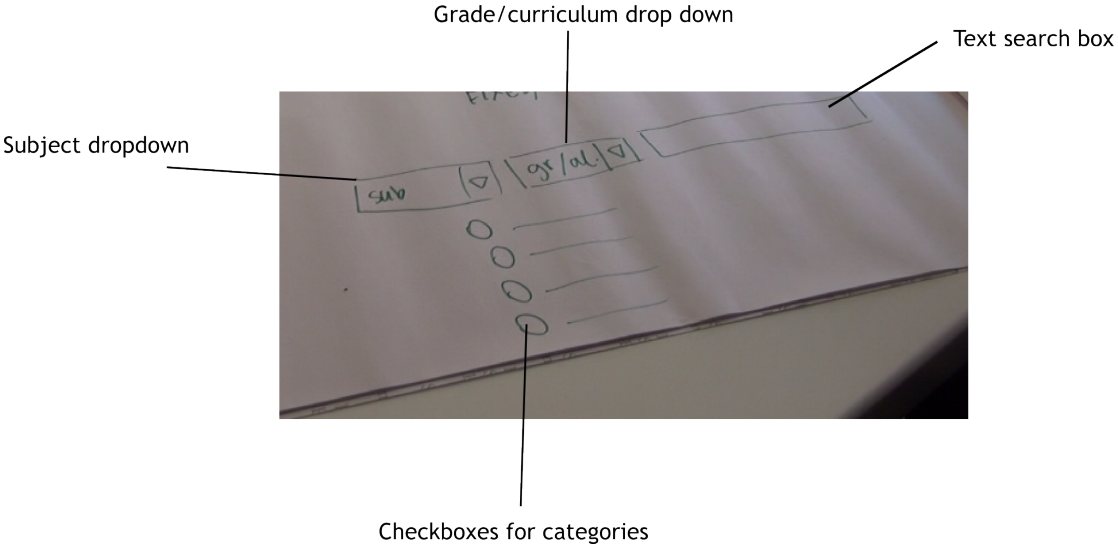


FIGURE 5.1: Users' initial drop-down interface idea

Users suggested drop-down lists for high-level fields (subject, grade, curriculum) with a text search box. A checkbox list for the type of annotation (suggestion, errata, comment) was also suggested.

It was agreed that there would be value in starting with a broad overview of annotations and then refining the list of annotations based on filtering. This lead to the suggestion (from a "Development" user) that all annotations should be loaded into the browser as a default view (on first load).

5.1.3 Second interface

The second interface proposed by participants included a default view of all annotations (and any replies to them) listed in a table, in sortable columns including "date", "comment", "location" (URL) and so on. Users would then be able to refine their query using a combination of filters on the left hand side of the page. These filters would include subject, grade and category of annotation. Each set of filters would initially be visible but easily collapsible, so as not to take up too much screen real-estate. Parameters

within a filter would be selectable by checkbox lists (one or many could be selected simultaneously). Each set of filters would also have a "select all/none" checkbox, to avoid multiple clicking within a set of filters.

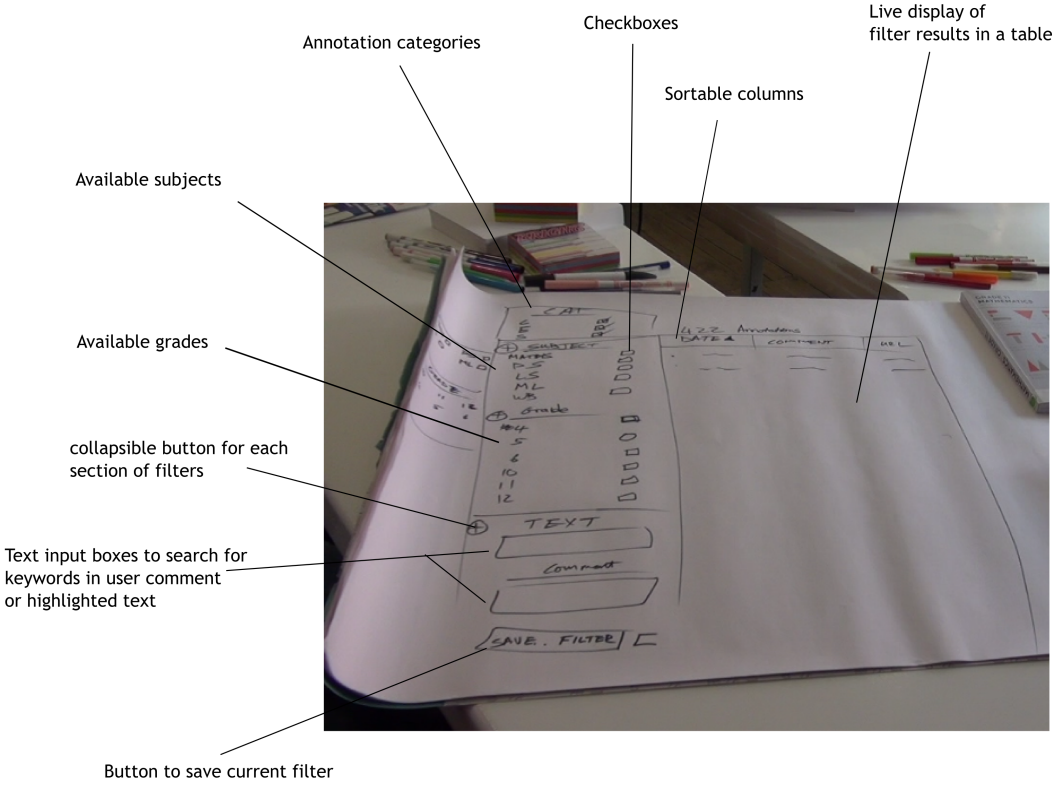


FIGURE 5.2: Users' second interface idea involving filter/search options on the left and results on the right.

The inclusion of a text search box was suggested, to search for keywords in the user comment part of the annotation, or in the text being annotated. Whilst two text searches were initially suggested, users later narrowed this down to one, that could search for text in either part of the annotation. Being able to search by username (based on a pre-populated list) in this search box was also mentioned as being desirable functionality.

5.1.4 Behaviour

In terms of behaviour, the default view would include all annotations, and the list would be refined as users clicked on different checkboxes and narrowed their results (e.g. if no specific grade was selected, annotations in all grades would be displayed). In the left hand menu, the high-level filters (e.g. subject and grade) would initially be expanded, whilst the other lower-level sets of filters would be collapsed. It was suggested that the default view included greyed out, selected checkboxes to indicate that all options were

automatically selected. Clicking on one box would then select that box, and deselect the others.

As filtering options were selected, the list of annotations displayed in the table of results (by date, with the most recent first) would then refresh in real time so that users would have a continuous feedback loop between their filter selection and the search results. This would enable them to quickly refine their search based on the immediate displaying on results.

The live updating of results would eliminate the need for a "Search" or "Submit" button. However, it was agreed that if the database were to get very large (admittedly an unlikely scenario for the company at present, given that each annotation is just a few bytes of text data) it would be more efficient to have a static search, plus a "submit" button, which then returned results, otherwise the website would get very slow.

It was agreed that it would be useful to be able to select how many results will be displayed per page (e.g. Wikipedia's "Next | 20 | 50 | 100") to prevent unlimited scrolling through results. "Previous | Next" breadcrumbs would also be useful for navigation between pages of tabulated results.

5.1.5 Saving filters and a user's last view

Users initially agreed that it would be useful to have an option to save specific filters, which could be selected from a drop-down list. The saving of a specific combination of filters should include the option to give that filter combination a name. The drop-down list of displayed filters should include the filter combination's name, the date on which it was saved, and the option to delete the save combination.

If a pre-saved filter was selected, the associated check-boxes in the filter menu should automatically be populated. Users would then be able to select or deselect items to further refine their results.

It was agreed that the browser should remember a user's last selection of filters. Whether a user logged out and in again, or simply moved to another page and hit the "Back" button on their browser, they should immediately view the last list of results (and associated filters/selections).

Users suggested a "Reset" or "Clear all" button to return to a default overview of all annotations.

5.1.6 Issue tracking

Users also decided that it would be useful to have some mechanism by which annotations could be marked as "new" or "resolved", and be assigned to different team members for processing. This lead to a lengthy conversation about how the interface could possibly function as a skin for issue tracking software like GitHub or BitBucket. It was agreed that it would be useful to be able to set the status of an annotation, and to view said status in the results table. Resolved annotations should be listed last, or one should be able to easily exclude them from results, although they should not be deleted from the database.

5.2 Session 2

The second session focused on how users would want their filtered results displayed, and how they would like to be able to sort those results. Many ideas were discussed, and some ideas from the first session were refined, based on decisions and opinions about the table of displayed results.

Following on from the issue tracking discussion in the first session, users focused on being able to assign an annotation a "status" and a "person responsible". These fields were initially included in the table of displayed results. A tabbed table view was suggested, with "unassigned" issues as one tab, "busy-being-dealt-with" issues as a second tab and "resolved" issues as a third tab (much like the GitHub web interface).

At some point in the second session however, one of the participants who is also a software developer pointed out that the group was no longer thinking about annotations - instead they were thinking about issues (in the bug tracking sense of the word). Issue tracking was then discarded by the users as being too complex because it would mean a rework of the entire system, to design an interface between the annotator software and an issue tracker like RoundUp or GitHub. They then narrowed the paper prototype down to a more simple design, excluding the tabbed view and table columns that would indicate issue status and the responsible user.

5.2.1 Displaying results

After further discussion, users agreed that results should be displayed in a table with the following sortable columns:
| Category | Info (URL, comment, highlighted text preview) | Number of replies | Date |

The default sort would be by date, with the most recent annotations listed first. The category column would include a visual representation of the type of annotation (comment, suggestion, errata), using coloured symbols to indicate type. This column would have a repetitive, circular sort, so one click would push one type of annotation to the top of the table, a second click would push the next type to the top and so on.

The "Info" column would be sortable by URL (which means that annotations would essentially be sortable by section in the book, given the nature of the URLs). This column would display (vertically) a preview of the URL, the user comment made in the annotation and the text that the user highlighted. The URL would be hyperlinked to the original page on which the annotation was made.

It was decided that "Number of replies" was a useful category, because an annotation with many replies is likely to need more urgent attention and processing than an annotation with no replies (e.g. if many users agree upon an erratum).

5.2.2 Detailed results

Users discussed how the system should behave when a user clicks on a specific result listing. It was agreed that clicking on a listing should open a detailed view of that annotation that included all of the information about that annotation, the full URL, comment and highlighted text. This detailed view would open in the same window or frame. It would feature a back button, to return to the search results. Clicking on the URL would take the user to the original page in which the annotation was made for a contextualised view.

5.2.3 Refined filters

Based on the results columns, the left-hand menu was then refined to contain only the following filters:

- Subject
- Grade/alignment
- Chapter
- Category (Type of annotation)
- Username search box

It was agreed that a top-down filter selection would need to be done. For example, the grades available depend on the subject selected, and similarly, the number of chapters available depend on the subject and grade selection. Users decided that the dependent filters should populate automatically as the selection is refined. So, once a user selects a particular subject, the list of available grades changes automatically to correspond with what books are available.

Users decided to eliminate the keyword text search box from the first session (they believed it would not be that useful), and instead have a username search, that worked like an autocomplete search (so that internal users do not have to remember external usernames or worry about misspelling them).

The option to save filters was discarded based on the new, simplified list of left-hand menu filters. Users agreed that as long as their previous set of filters was saved (using cookies) when they returned to the site, that should be sufficient, because the list of possible selected filters was now much shorter.

5.3 Consolidated prototype

At the end of the second session, participants consolidated their ideas into the following interface:

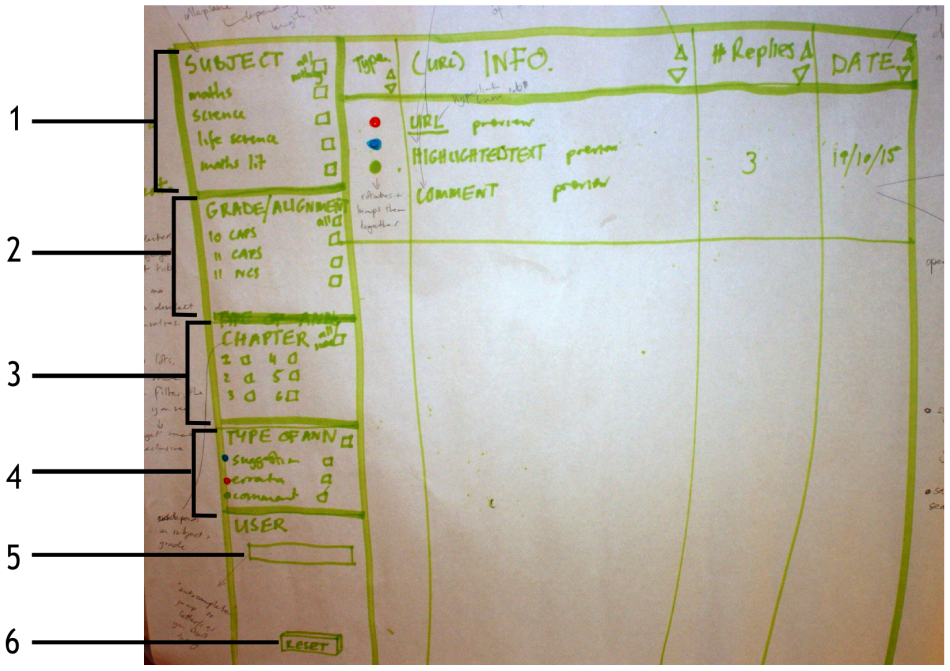


FIGURE 5.3: The final paper prototype.

5.3.1 Description of interface

The interface would be divided into two main segments: a list of filters on the left hand side of the page, and a table of results taking up the rest of the page. At first login, the user will see the filter options (see Figure 5.4):

1. Subject
2. Grade/alignment
3. Chapter
4. Type (of annotation - errata, suggestions etc.)
5. Username search box
6. Reset button (to return to default filter state)

Everything would be expanded, and everything would be selected (so all annotations would be displayed initially). The fact that everything is automatically selected would be indicated by greyed out, ticked, checkboxes. Each set of filters would be collapsible, and each would have a "select all/none" button. Clicking on one filter option (e.g. Subject = "Maths") would automatically deselect the other options.

Lower level filter options would automatically change based on higher level selections. (E.g. currently there only exists a Grade 10 book for Maths Literacy, so if a user selects "Maths Lit" as the "Subject" filter, they should only see "Gr 10 CAPS" as an option under the "Grade" filter).

The basic behaviour is that the user would initially see all the available annotations, and that "the more they click, the less they see". So selecting filter options would refine the results displayed (live) in the table on the right.

As mentioned above, the username search box would function like an autocomplete search, querying the database of known usernames as a user starts typing. A "Reset" or "Clear" button (Label 6 in Figure 5.3) would be placed at the bottom of the left-hand options to return the displayed results to the default view.

The table of displayed results would have the following sortable columns (see Figure 5.5):
| Type | Info (URL, comment & highlighted text previews) | Number of replies | Date |

The default sort would be by date, with the most recent annotations listed first. The

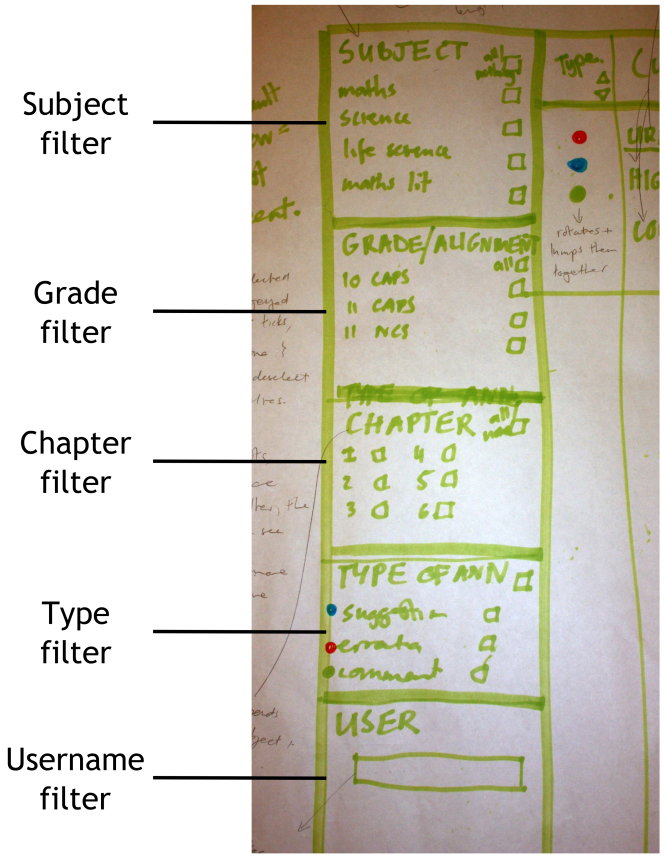


FIGURE 5.4: Filter boxes in the final paper prototype.

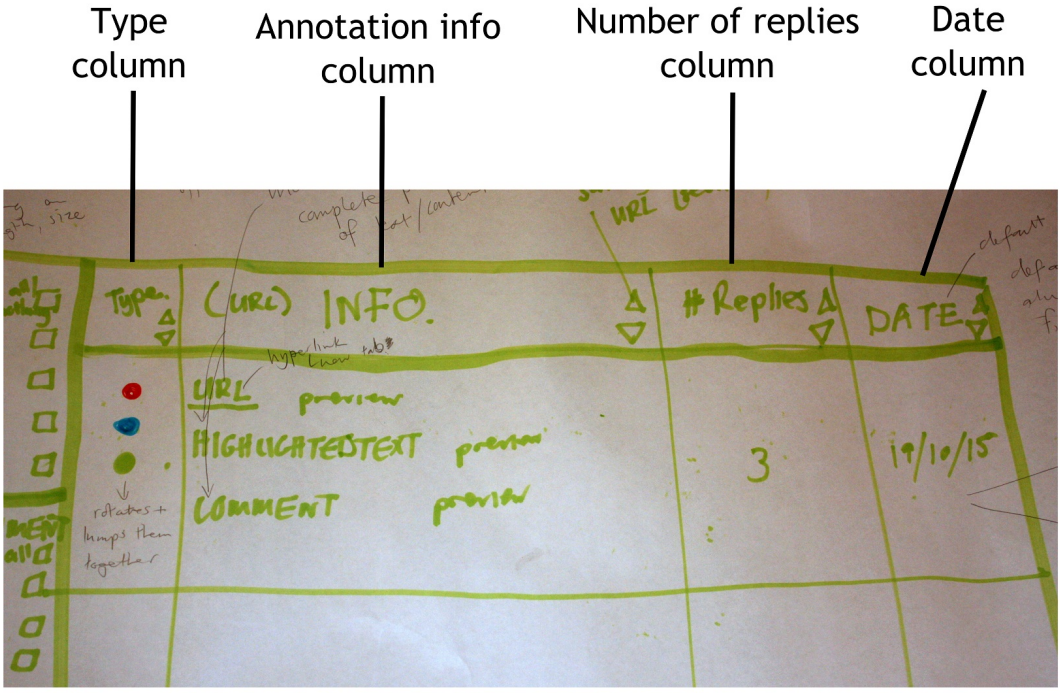


FIGURE 5.5: Table heading columns in the final paper prototype.

URL would hyperlink to the actual annotation in the book and the "Info" column would be sortable by URL (which corresponds to books and their chapters and sections). Breadcrumbs like | Previous | Next | 20 | 50 | 100 | would occur at the bottom of the table of results.

The type of annotation would be indicated in the leftmost column by a small coloured icon. This column would sort in a rotational manner.

Clicking on a particular annotation would open a detailed view of said annotation in a window (or frame) in the same page. Right-clicking a particular annotation would allow the user to open it in a new tab. This detailed view would contain all the available information about the annotation (full URL, comment, highlighted text, username, date, replies), and the hyperlinked URL to take one quickly to the original annotation in the context of a book. Clicking "Back" from the detailed view would take users back to

FIGURE 5.6: Detailed view overlay in the final paper prototype, showing more information for a specific annotation.

their previous set of filters and results. Likewise, if a user logged out and in again, or navigated back to the back-end interface from another page, they would see their last search results.

5.4 Discussion

Overall, the participatory design process was very constructive and informative. It was extremely valuable to get user input, and having users work in a group together meant that they could discuss and refine ideas with each other and reach a unified design concept which satisfied everyone.

Drawing several iterative sketches of the interface enabled users to explore and experiment with different possibilities together. These sketches also seemed to help users visualise concepts and explain ideas to each other. Many ideas were brainstormed and then included or discarded as the users' design evolved. Whilst some users were more comfortable talking than drawing, the group struck a balance between thinking out loud and visualizing their ideas on paper for others to see.

Having two separate sessions spaced one week apart also gave participants an opportunity to mull over ideas that had been discussed before the second session.

Sometimes the brainstorming became very creative and extended beyond the scope of the interface in question. In these instances it was challenging to try and keep users focused on the task at hand, without dampening their enthusiasm.

Ultimately, any proposed functionality that required issue tracking integration (e.g. GitHub) or server-side processing was excluded, in order to limit the scope of the project slightly and focus on the interface functionality itself, instead of the design of a database system that would have to include user authentication, server integration and so on.

Chapter 6

Building a High Fidelity Prototype

Once the paper prototype was complete, it was converted into a Balsamiq[1] mockup (see 6.1, to be as tidy and legible as possible, as well as digitally portable.

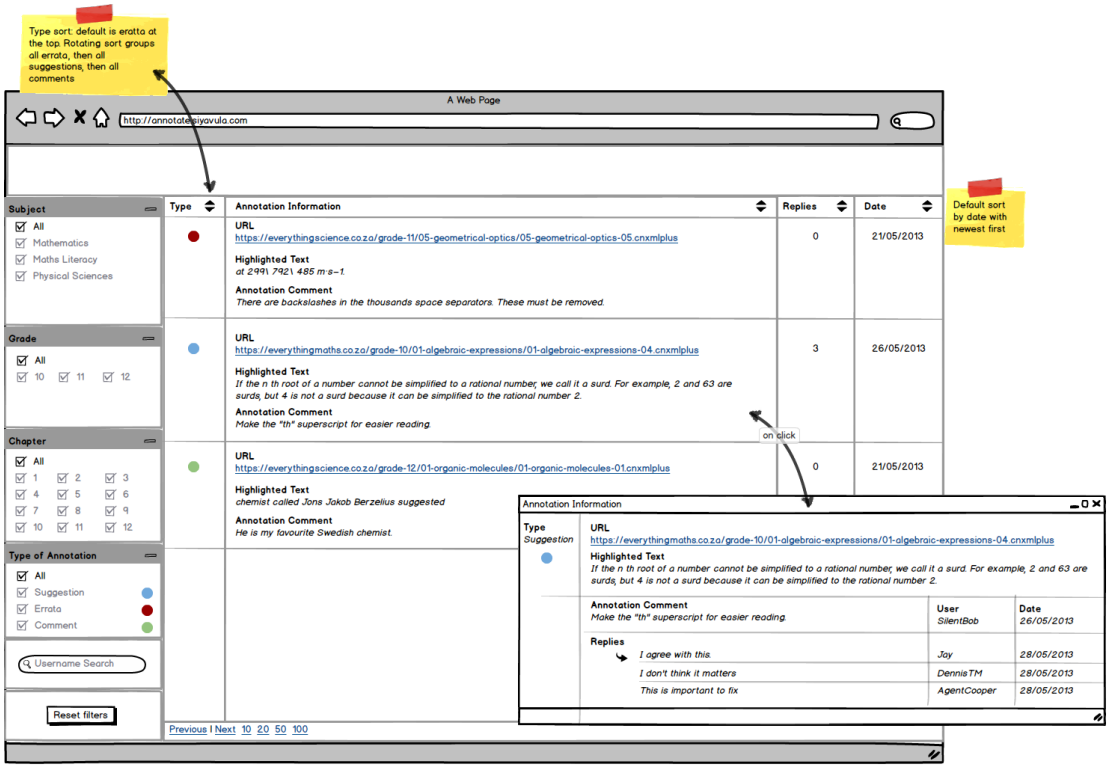


FIGURE 6.1: Balsamiq mockup of the final paper prototype.

[1] http://balsamiq.com/

This mockup was then converted into a functional, high-fidelity prototype website using HTML, CSS and XML, as is detailed below. Scripted functionality was built using Javascript and the JQuery library in particular.

The site was hosted using GitHub Pages[2] at http://nicoladt.github.io/MIT-Thesis/ which offers simple, free hosting, and seamless integration with GitHub version control software. Assistance was given by Ewald Zietsman with the Javascript and JQuery coding.

6.1 Hi-fi prototype version 1

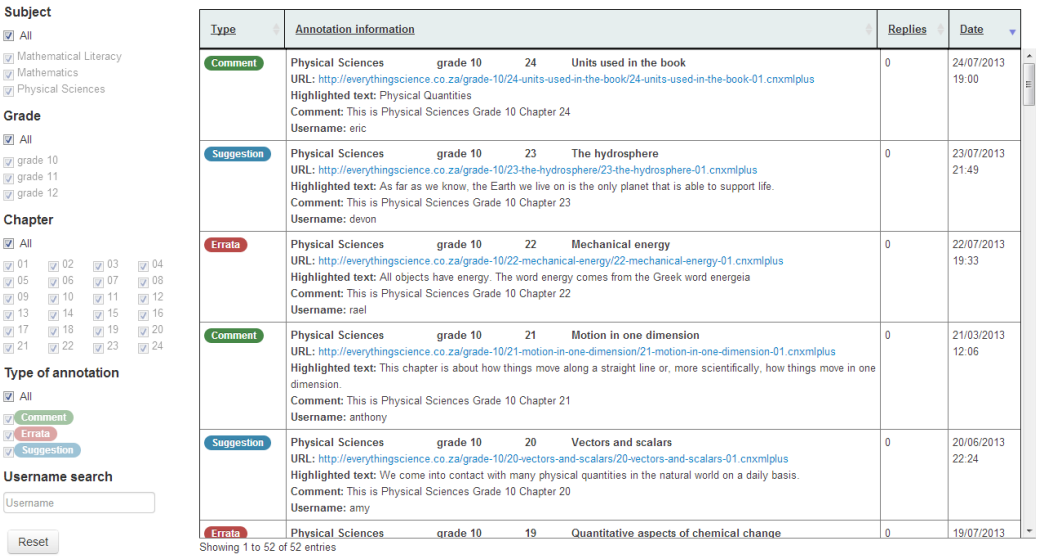


FIGURE 6.2: Screenshot of the high fidelity prototype.

6.1.1 Technical details

The interface was comprised of two main areas. The column on the left contained all the filter options, and the table making up the rest of the page displays the annotations in rows.

Placeholder annotations were made in an XML file. Each annotation was assigned a type and had a simple data structure including a timestamp, a URL (pointing to a particular webpage on the live webbooks sites), a username, highlighted text (from the book), and a user comment. In addition each annotation could have one or more replies. Replies in turn also had a timestamp, username and user comment:

[2] http://pages.github.com/

```
<annotation type="">
  <url></url>
  <username></username>
  <datetime></datetime>
  <highlighted></highlighted>
  <comment></comment>
  <replies>
    <reply>
      <username></username>
      <comment></comment>
      <datetime></datetime>
    </reply>
  </replies>
</annotation>}
```

For the prototype, annotations were not given unique ID's or primary keys. Whilst this would be necessary for real annotations (due the the possible complexities of many users making many annotations simultaneously), for the prototype a combination of the URL and timestamp was used as a unique identifying characteristic.

Annotations were constructed carefully with possible future evaluation tasks in mind. They were created so as to represent at least every possible combination of subject and grade. Annotations were assigned to the three types (comment, errata, suggestion) and random usernames were created. Replies (in number from 1 to 3) were added to some annotations.

The interface was built upon the basic components of the Bootstrap 2.3.2 framework[3]. Bootstrap was selected not only because of its built-in cross-browser compatibility but also because it comes packaged with clean and simple CSS and a number of default elements (like buttons) that are easily customizable.

The DataTables[4] Javascript plug-in was implemented for easy control over the table and its functionality and behaviour.

A fixed header for the the table was deemed important, so that users always knew what each column represented. However, DataTables fixed header functionality does not work correctly with pagination (a known DT bug), so a compromise was reached and vertical scrolling was enabled instead. Pagination breadcrumbs (arguably not necessary for sets of annotations < 100 entries anyway) beneath the table, were substituted with the

[3] http://getbootstrap.com/2.3.2/
[4] https://datatables.net/

automatically-updated line of text saying "*Showing xx number of yy entries*", where xx was the number of rows in the table, and yy the total number of annotations.

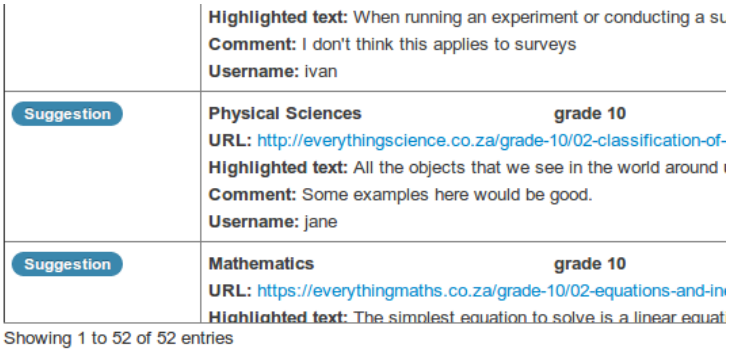


FIGURE 6.3: Text beneath the table indicated how many rows (annotations) are surfaced

DataTables also contains an unfortunate (and well-documented) CSS bug, in which the fixed table header `<th>` cells do not always align correctly with the table data `<td>` cells beneath them. However, this is a minor visual issue concerning a few pixels.

DataTables can only handle an alphabetical sort of columns, which meant that a three-way (rotational) sort of the "type" column was not possible - i.e. it was not possible to sort so that "errata" was listed at the top, "e" being between "c" (comment) and "s" (suggestion) in the alphabet. Additionally, it is only possible to perform a default sort on one column initially, so a sort based on Date and then Type was not possible.

Whilst the loss of these possibilities is noteworthy, it was decided to proceed with the implementation of the DataTables plugin, because the missing functionality could easily be substituted using other techniques (e.g. it was very easy to filter all the annotations by "type" to get to the third sorting category of "errata"). For this reason it was decided that the pros of using the DataTables plugin outweighed these few cons.

Javascript and the JQuery library were chosen to do the information processing required by the interface. JQuery allows for client-side processing. It is simple to implement, and runs as a light-weight instance in any browser[5]. Practically, client-side data processing (as opposed to HTTP calls to a server) allows for rapid updates to a webpage, and therefore the interface in question.

GitHub Pages was selected as the simplest hosting solution for the prototype, because it is secure and reliable, and most-importantly integrates seamlessly with GitHub version control software. However GitHub Pages can only serve static HTML and it cannot handle server calls. Again, the pros of using GitHub Pages outweighed the single loss

[5] http://jquery.com/ and http://en.wikipedia.org/wiki/JavaScript

of functionality it presented, which was the inability to use cookies to store user session information.

In terms of scripting behaviour, the script written parses the list of XML annotations and builds then builds the filters on the left of the interface based on what it finds in the XML. Each URL from everythingmaths.co.za and everythingscience.co.za[6] contains information about the subject, grade, chapter and section applicable to that annotation. This information is taken from the URL and presented in a human-readable text format, both as filters on the left hand side, and as inline text for each annotation.

6.1.2 Behaviour

Once the filters have been constructed from the XML, the table is then populated based on the selection filters on the left of the page. The default is that all the annotations are selected and therefore all the annotations are displayed in the table. The basic behavioural pattern is that interacting with the left-hand filters automatically changes what is visible in the table. Any changes made to the left-hand selection cause the table to display only the annotations relevant to the new parameters selected on the left.

The checkboxes are grouped according to the field[45, p. 439]: i.e. by subject, grade, chapter number and type.

Initially the 'All' checkboxes were ticked, and the sub-checkboxes were also ticked but greyed-out, as per the paper prototype. Clicking on a sub-check selected that particular checkbox, deselected the "All" checkboxes and deselected all other sub-checkboxes. It also made the sub-checkboxes opaque.

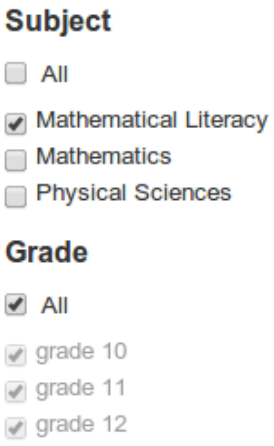


FIGURE 6.4: Single selection of sub-checkbox, compared to selection of "All" checkbox.

[6] http://everythingmaths.co.za/ and http://everythingscience.co.za/

Clicking on an checked "All" checkbox deselected the "All", selected all of the sub-checkboxes in that filter category, and made them opaque. In other words, the "All" checkbox behaved as an all or subset toggle, not an all or nothing toggle (which would just result an empty table).

Whilst this deviated from the standard checkbox behaviour (*all or nothing*)[45, p. 435] this modification allowed for the possibility that a user would want invert a selection: to select most (but not all) of the sub-checkboxes in a category. For instance, a user might want to select 23 of of 24 chapter checkboxes. Clicking 23 times would be tedious indeed. With the "All" behaviour described above, a user could simply deselect the "All" checkbox and deselect the chapters they did not want included in the results.

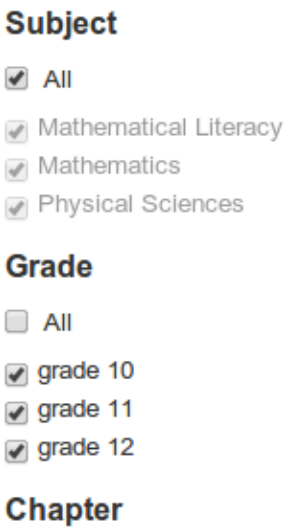


FIGURE 6.5: Subject, with "All" checked vs. Grade, with "All" **unchecked** and all sub-checkboxes activated.

The username search box was a simple, case-sensitive text search. It did not serve up an auto-completing drop-down list. Instead, the table updated automatically (with each new character typed) as the user began typing into the input box. This alternative behaviour to that specified in the paper prototype was selected because it was consistent [40, p. 261] with the behaviour of the table with all other filter fields, whilst still giving users the immediate feedback they required (in the form of instantly updated results with every keystroke) when performing a potentially uncertain search for a username.

The button labelled "Reset" simply resets the page to the default view.

The table was sorted by default by descending date. As mentioned above, it was not sorted with errata at the top of the type column due to DataTables' strictly alpha-numeric sorting capabilities. That being said, an alternative method for viewing "errata"

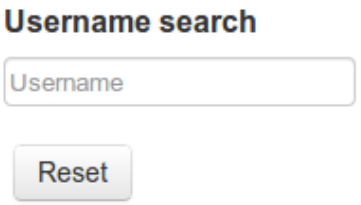


FIGURE 6.6: The username search box and Reset button.

type annotations at the top of the table was provided by the type filter on the left so this was deemed an acceptable alternative.

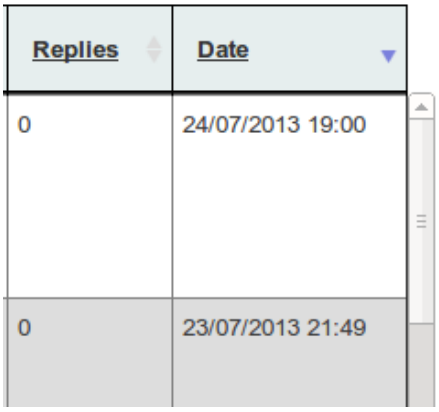


FIGURE 6.7: "Sortable" vs "Sorted: descending" icons in the table header

On mouseover of the table rows, the row changed colour (from white to light grey) and the cursor changed to be a "zoom in" cursor to indicate that action is possible: that the row is clickable and there are more details to be viewed or expanded.

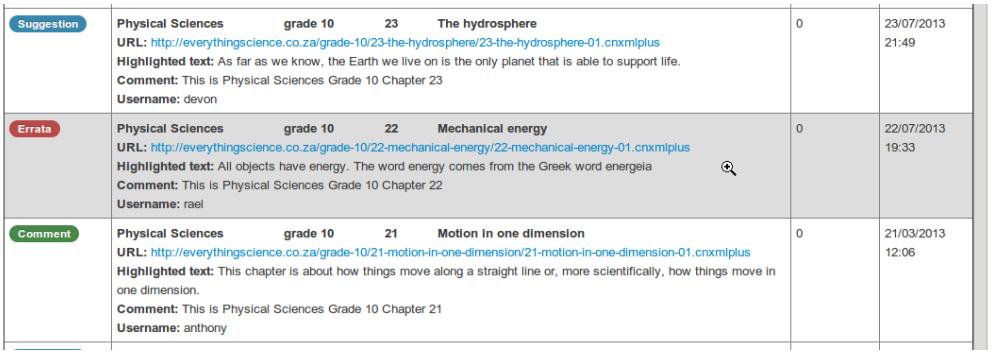


FIGURE 6.8: Grey row colour and zoom icon on hover.

Clicking on a particular annotation brought up the detailed view window as an overlay on the table. As well as all of the annotation information visible in the table, this view also includes all replies to an annotation and the reply details. To close the detailed window, users could click on the [x] in the top right corner of the window or they can press the [ESC] key.

Because the detailed view was an overlay of the interface and not a separate browser window, a "Back" button (as in the paper prototype) was not included. Users were given two standard alternatives to close the overlay instead.

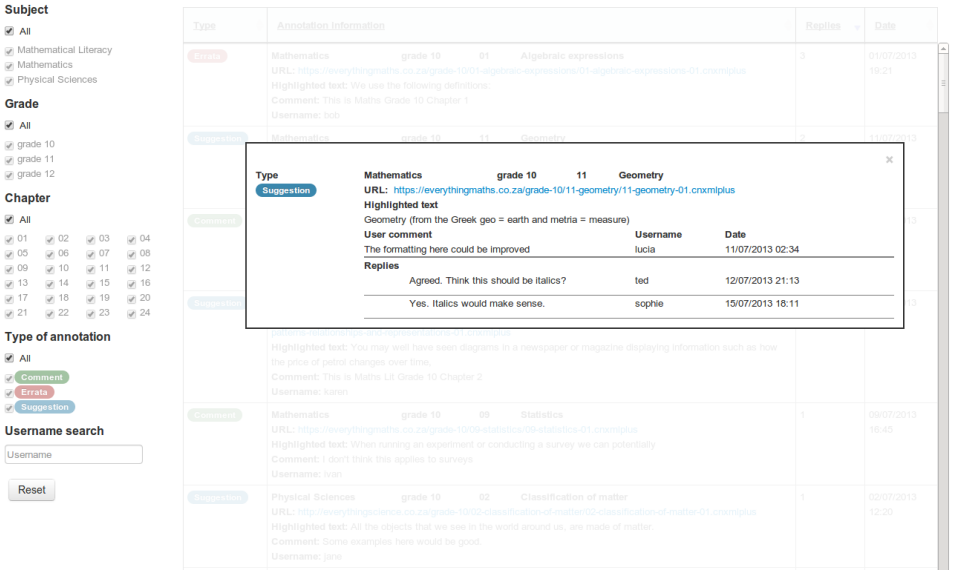


FIGURE 6.9: The detailed view overlay.

6.2 How and why the high fidelity prototype differs from the paper prototype

As mentioned in Chapter 5, certain functionality that was discussed in the paper prototyping process was excluded because it required the integration of server-side processing and issue tracking software, which would significantly extend the scope of the system. Differences between the paper and high fidelity prototypes are detailed as follows.

1. **No cookies**
   For simplicity of development and in order to use of GitHub Pages for hosting, the interface only used client-side processing, without any server integration. This meant that cookies storing information about a user's session could not be saved, and that if a user navigated away from and back to the webpage, it would be reset to its default view: the user's last filter selection would not be saved.

   The advantages gain by using GitHub were significant, as mentioned already. Additionally, it could be argued that the selection of filters and quick updating of the resulting table was simple and quick enough that redoing a task would be relatively trivial.

2. **No user authentication**
   Again, because of the lack of a server, user authentication was not implemented in this prototype. Authentication is arguably not needed to test the interface functionality itself, however: it would be an independent process that would take place in its entirety before a user came to view the interface in question.

3. **No collapsible filters**
   The groups of filters and checkboxes were not collapsible, because they all fitted onto one page. The collapsibility was suggested by users during the paper prototyping sessions because they were concerned that there would be too much information to fit onto one webpage. In implementation however, this was not the case, so this functionality was set aside. Because the filters offer a graphical representation of the system status (what is visibly selected on the left maps directly to what information is in the table) displaying all filters on one page also improved observability [40, p. 270] for users.

4. **No username autocomplete suggestions**
   The username searchbox did not offer a dropdown list of autocomplete suggestions when users started typing. Instead, it updated the table of results automatically with each keystroke. This behaviour was the same as for all the other filter categories, and so it replicated a pattern found throughout the interface (consistency and predictability), whilst still giving users immediate feedback (instantaneous responsiveness) and the easy ability to undo actions (recoverability)[40, p. 272].

5. **Extra information added to each annotation cell**
   At the top of each "Annotation Info" cell, a line of bolded text containing the subject, grade, chapter number and chapter name was added, to give users more, easily-readable information about where each annotation was from, so they did not have to decipher a long URL.

6. **No back button in detailed view**
   Because the detailed view was an overlay and not a new webpage a "Back" button was not included (users making the prototype specified that the detailed view should open in the same page). Instead, users could close the overlay by clicking on the [x], in the top right corner, or by pressing the [Esc] key - both standard interface patterns [45, p. 345]. Using an overlay instead of a new webpage or window also prevented users from leaving the webpage unnecessarily, and therefore reduced the risk of users getting lost in a navigation process.

7. **No Life Sciences subject**
   Due to external factors, the Life Sciences Grade 10 book was not yet available

on the Everything Science website, so no annotations were made for that subject, because there were no live URLs available.

8. **No curriculum alignment filter**
   By the time the high fidelity prototype was being built, all available textbooks on the Everything Maths and Science websites were aligned to the current CAPS curriculum, and no old NCS content was available. Hence, the curriculum alignment filter was excluded.

9. **Lower-level filters do not update**
   Users in the PD sessions suggested that lower level filters (like Chapter) should update, based on the higher selection. So, the listed grade filters would update depending on the subject choice, and the number of chapters would change to reflect what was available in a specific combination of grade and subject.

   If users performed a unidirectional filter operation this functionality would be useful. However excluding it and displaying all possible subjects, grades and chapters meant that users could easily change their selection, and filter both forwards and backwards. This made it far simpler to correct mistakes, undo actions or browse the data and discover new information, all of which conforms to good design principles [40, p. 272].

   Additionally, displaying all of the possible filters at once meant that users always have visual feedback as to what they have selected, in the context of what they can select. This makes current and future states more apparent and discoverable (which ties into the design principle of observability [40, p. 270]).

10. **No coloured dots next to "Type"**
    The coloured circle icons suggested for "Type" categories were replaced by replaced by coloured labels beneath the "Type" text. The coloured labels are packaged as part of the default Bootstrap styles, and were therefore simplest to implement, the visual effect arguably being the same. Three colours were selected based on users' paper prototype suggestions, available Bootstrap label colours and common meanings for colour [45, p. 635] (e.g. red, which often signifies a warning was used for errata).

11. **No right-clicking of rows**
    Right-clicking an annotation row does not open it in new tab. This is simply because the detailed view was constructed as a visual overlay, and not as a new webpage.

12. **DataTables-related changes**

(a) **No double default sort**
As discussed above, using DataTables meant that it was not possible to default sort by date and type. Because of the ease with which users could filter by type on the left hand side of the interface, the default sort was simply by date, descending.

(b) **No breadcrumbs or pagination**
Due to DataTables' fixed table header not working with the built in pagination functionality, pagination was substituted with vertical scrolling. Instead of "Previous | Next | 10 | 20 | 50 |" breadcrumbs at the bottom a message was surfaced indicating how many annotations (rows) were displayed in the current table.

(c) **Strict alphanumeric sorts**
Because DataTables' sorting functionality is strictly alphabetical, a three-way sort for the "type" column (to surface "errata" at the top) was not possible. Similarly, the "annotation info" column could not be sorted by URL.Iinstead it was sorted by the subject, grade and chapter number text at the top of each cell (the same information that the URL provided anyway).

6.3 Discussion

Whilst the high fidelity prototype did diverge from the paper prototype slightly, wherever possible, behaviour or design specified in the paper prototype that could not be included was substituted with an equivalent solution. Trade-offs (e.g. using DataTables) were carefully evaluated so that whenever possible, more functionality was gained than was lost, or had to be altered.

Any divergence could (and would) also be thoroughly evaluated by users in the next part of the process, involving usability testing and formative evaluation.

In some cases, the implementation of the prototype in an actual web browser opened up new possibilities - such as styling the filters to all fit on one standard 1366 x 768 resolution screen. The visual effects of an overlay are also easier to envision and explore with HTML and CSS than on paper.

Several aspects of the current prototype are not highly scalable. For example, client-side processing of thousands (not dozens) of annotations will get extremely slow. Similarly, while reloading the table currently takes milliseconds, with a large database of annotations, and a more complex set of filters (e.g. more subjects, more grades) the load time will be significantly increased. There is no doubt a performance threshold at which it

would become practical to implement server-side processing instead, which would also mean that user authentication and use of cookies would become possible. Similarly, with a larger database, pagination of the table (with breadcrumbs) as opposed to vertical scrolling would be necessary to reduce load and response time of the interface.

Whilst DataTables provided a lot of pre-packaged functionality, it is currently not very robust, and it appears that new releases and improvements, while pending, are slow to occur. Ideally it would be better to reinvent the wheel, and write customised code to handle sorting (e.g. three way sort, implementing more than one default sort parameter), pagination, header behaviour and so on.

Once the high fidelity prototype of the interface was complete the next step was to formatively evaluate the high-fidelity prototype with a new group of users [1, p. 329], to ensure that the design thus far still conformed to user requirements and expectations. Once this was done, the interface could then be iteratively improved based on the first round of usability tests and user feedback. This process of evaluation and iteration will be discussed at length in the next chapter.

Chapter 7

Formative Evaluation

7.1 Evaluation

According to Rogers, Sharpe and Preece [1, p. 433] evaluation is a fundamental component of the design process. It allows designers to collect information about what users experience when interacting with a prototype. Evaluation focuses both on the usability and user experience of a prototype, and its purpose is to improve a prototype design.

One form of evaluation is usability testing. Usability tests involve collecting data using a variety of methods including observations, interviews and questionnaires. Rogers et al [1, p. 438] state that the fundamental goal of usability tests "is to determine whether an interface is usable by the intended user population to carry out the tasks for which it as designed. This involves investigating how typical users perform on typical tasks." Similarly, Shneiderman and Plaisant [41, p. 144] state that "usability tests are designed to find flaws in user interfaces". According to Beyer et al [38, p. 373] they "tune an interface at the tail end of design, to clean up any rough edges or unnecessary difficulty in understanding or interacting with the interface."

While usability tests can be performed in controlled laboratory settings (e.g. if the performance of a prototype needs to be measured), it is also possible to perform them in more casual settings familiar to the user [1, p. 438]. This latter option was selected because the testing related to functionality and behaviour (not computing performance) and because of the simple practicalities involved in testing with users in their own workplace. Physical and system variables were kept consistent wherever possible and all tests used the same physical equipment and software.

To evaluate the high fidelity prototype interface in question, two rounds of usability tests were undertaken. The first set of tests was intended as formative evaluation, i.e. they

were performed during the design process to ensure that the design was still conforming to user expectations and requirements [1, p. 437].

The results from these tests were then analysed and used to improve the design of the prototype. A second round of summative usability testing was then undertaken, to assess the interface as a finished product and to determine whether it did in fact allow for simple and easy filtering, finding and searching of annotations.

The usability tests were designed to include interview-style questions about what users thought aspects of the interface represented, and how they expected them to behave (without interacting with them). Additionally users were asked to perform a number of simple tasks, carefully selected and designed to evaluate various behavioural aspects of the interface. For the summative set of tests, users were also asked to complete a satisfaction questionnaire.

Both sets of usability tests are discussed in detail in the following sections. To summarise the process briefly: five new users were chosen for each set of tests and under the same conditions (in the boardroom at their office) users were asked the pre-determined questions and to complete the list of tasks. Users were tested in groups of five because there were ten new users available in the team, who had not seen the interface before.

Testing with these ten users (chosen to be a representative sample of the three user groups identified who would use the final interface [1, p. 461]) as well as designing with the original four users meant that the entire group of users available to this process would have participated by the time summative evaluation was finished. Small sample sizes are common in usability testing, and it is well documented that a high number of usability issues can be determined from tests with just a few users [44, p. 119].

Each session was recorded using screencast software that recorded the active desktop and the conversation. Additionally the evaluator took notes detailing user comments and actions.

The data collected from each session was analysed for misconceptions, misunderstandings and misclicks. The aim of the analysis was to identify any interface elements or behaviours that confused users, or behaved differently to their expectations, and to "identify user interaction components or features that both support and detract from user task performance" [46].

The results of the analysis were then summarised and used to suggest design changes and improvements to the interface, to bring it more closely in line with user expectation and requirements.

7.2 Formative evaluation

Formative evaluation is evaluation that takes place during the design process and is used to improve a design citep[p. 149]Hartson. It focuses on identifying usability problems in a prototype that should and can be addressed during an iterative design process. It also ensures that users continue to be included in development, because their feedback is central to improving the design [46].

The goal of the formative evaluation process was firstly to determine that the high-fidelity prototype was an accurate representation of the paper prototype, according to the users' conceptual models, and secondly to establish that the interface met the high-level user requirements determined earlier in the design process.

Following the process outlined by Gabbard et al [46], user questions and tasks were developed to test all functionality and behaviour of the interface. In particular, the tasks were designed to test the user requirements that emerged from the requirements analysis process and the paper prototyping sessions. The test was practised informally (with a willing family member) before users were involved, to ensure that the wording of questions and tasks made sense and that the evaluator was comfortable with the test material.

Users were first asked interview-type questions to assess their conceptual interpretation of the interface. Without interacting with the system (they could move the mouse and hover over elements, but not click on anything), users were asked what they thought various visual elements represented, and how they would expect them to behave were they to interact with them. These questions were designed to determine initial user expectations and assumptions, and were phrased such as "*what do you think xx represents?*" and "*what would you expect to happen if you clicked on yy?*".

Users were then asked to complete the tasks. The tasks were contextualised with reality-based scenarios, for example: "*Let's say a user calls the office to ask if his annotation has been captured in the system. His username is "bob". You need to find all the annotations made by him. How would you go about doing this?*"

7.2.1 Results

User A thought that the greyed out sub-checkboxes did not look clickable. He also thought that the username search box looked greyed out. He was uncertain as to how the username search box would behave: while he noticed there was no "Search" button to submit a user search he did not know if the search box would "*filter on the fly*" or

offer an autocomplete dropdown list of matching usernames. He said it was not always obvious that the table rows had actually updated when he had interacted with a filter.

He realised that the zoom icon (when hovering on a table row) indicated expansion, but was not sure what information an expanded view would include. He commented that the cursor did not change if he hovered over the table header sort icons, so it did not seem clear that the sort arrows were clickable because: "*When you can click on things you get the finger*" ( ). He also noticed that when hovering over the table header text (e.g. "Type") the cursor changed to be a text input cursor ( ). This was a default DataTables/Browser behaviour which the evaluator had not noticed in testing before.

In addition, User A noticed the DataTables CSS bug already mentioned, where in some instances the fixed table header cells do not align correctly with the table data cells in the rows beneath them.

To close the detailed view overlay, he tried clicking off the box, which did not work - he felt it "*probably should*". He also mentioned that he thought the Reset button was "*a bit obscured*" at the bottom of the page, and suggested renaming it to "Reset Filters" and moving it to the top of the page.

User B also did not think that the greyed out sub-checkboxes were clickable - she assumed they would become clickable if she unchecked "All". She also expected the "All" checkbox to be an all/none toggle, so that if "All" was unchecked, then nothing would be selected. When she realised this was not the case (unchecking an "All" selected all the sub-checkboxes instead), she deselected the sub-checkboxes to get to the required filter set. For example, to select the "chapter 5" filter - she unchecked the "All", which selected the 24 sub-checkboxes for "Chapter" and proceeded to deselect 23 checkboxes until just "5" was selected. Even though she realised this was inefficient and unlikely to be the only solution, she did not experiment with the checkbox behaviour to see if an alternative option was available.

The purpose of the username search box was not initially clear to her as she did not know what it would be used to search for. She also was not sure how it would behave and whether she would have to press "Enter" to submit a search, for example. Once she interacted with the search box however, she realised that it filtered the table automatically.

User B also missed the visual indication of the default date sort ( ). She realised the table was sorted by date, but she deduced this from the date column entries.

User C did assume that the greyed out sub-checkboxes were clickable. However, when she interacted with the system, she first tried to deselect one, and then discovered that

clicking on one actually selected it. However, she did not think to deselect all subject checkboxes to get no results in the table.

She did not expect anything to happen if she started typing in the username search box, and she mistook the "Reset" button for a username "Search" (submit) button: she assumed she would have to type in the box and then press "Reset" to submit her user search.

When she started interacting with the interface it was apparent that her confusion about the "Reset" button extended beyond this: after she selected the relevant group of checkboxes to a particular task she clicked the "Reset" button to submit her entire filter query (i.e. she did not notice the table updating at all) and then was perplexed when she realised her filter choice had just been cleared. According to her: "*I would want to choose something and then click something else. It would make sense if you had some instructions. I think it's important to have a Reset button because there's quite a lot of options But my initial thing is that you choose and click Enter [to] search. That's how you do it on Google*".

User C thought that the zoom icon on row mouseover meant that the rows were clickable, but was not sure what clicking on a row would do ("*either it would bring up more information, or single out that particular comment but I don't know how...*"). Eventually we established that she expected the zoom icon to indicate a literal, visual zooming in, i.e. making the text bigger ("*why would you want to see it bigger, when you can read it?*"). She did not think that clicking on a row would bring up more detailed information and she only noticed that the URL in each row was clickable.

She was uncertain as to how the "Annotation Information" column would sort itself, and was not sure why one would want to sort the table by "Type", when "*you can do that with the filters on the left*". When she initially clicked on a sort icon, the table update was not obvious (the sorting change to the visible results was minimal), and this confused her. However she persevered and clicked again, and then realised that the table had updated.

She thought that the coloured type labels looked like clickable buttons. She also noticed a bug where, even when there were no entries in the table, the counter text at the bottom still displayed "Showing 1 of 1 entries".

User D did not think that the greyed out sub-checkboxes were clickable in their initial state but thought they probably would be if the "All" checkbox was deselected (i.e. an all/some toggle). That being said, initially she did not try to click on a greyed out sub-checkbox - she deselected "All" first and then clicked on a (already checked) sub-checkbox. She realised that deselected the sub-checkbox. She experimented and

discovered by accident that she could just select one greyed out sub-checkbox directly. (For the chapter selection task where she discovered this alternative behaviour she said: "*I wasn't going to uncheck them* [23 chapter numbers] *all!*"). Interestingly, even after she realised she could just directly select a greyed out sub-checkbox, for the next tasks she went to uncheck "All" again, first. Like User C, she did not think to deselect all subjects to get no results in the table.

User D thought that the username search box was for "*your username*" (i.e. that one would input your own username into the box, and that it was not a search). She did not expect typing in it to have any effect on the table. Then, in one task, she typed the text "Comment" in the box, to search for it. Based on this experiment, she then realised that the search box was specific, not general (i.e. not a site search), and then understood that it was to search for usernames in the table. In a later task, when she searched for "s" in the username search box, the table updated so fast that she initially did not notice the results had changed.

She did not know what the sorting arrows in the table header indicated, and did not notice the Date default sort icon. She did not think that it was possible to sort by a column header until she tried clicking on the "Replies" header. When asked to sort by the highest number of replies she said: "*I'd try and click here somewhere [clicked] "Oh and that's what it does!*". She also noticed that the date sort, which at first glance looked correct, was in fact not (caused by a bug in the code that converted a long timestamp into human-readable text).

User D did also not believe that the zoom icon indicated that the row was clickable. For her, it was not obvious that she could click on a row, or that doing so would surface additional details about that row.

User E initially thought that deselecting an "All" checkbox would result in one of the sub-checkboxes being selected. She did not think that the greyed out sub-checkboxes were clickable, but said that the changing mouseover icon suggested that they were. She expected to have to uncheck the "All" checkbox first. When she actually interacted with the interface however, she tried clicking directly on a greyed out sub-checkbox (without deselecting the "All") and realised that was possible. Despite this, she said "*my instinct is still to uncheck All' first and then click on the others*".

With respect to the username search box, User E said that she hoped the table would not update until she had finish typing a username into the search box, because she was concerned that it would be very slow. She thought the search box might offer a dropdown list of autocomplete options, and said that that "*would be good*". She initially assumed that she would need to hit "Enter" to submit her search, but then noticed that there was

no "Search" button, so she assumed the search box must filter the table automatically. In a later task when she actually interacted with the search box, she noticed that it updated automatically as she typed.

Like User A, User E said that she expected the cursor to change when she moused over the sort icons in the table header. She noticed that a three way sort was not possible and that she could not sort to have the "Errata" type at the top of the table. "*It only has two...Then again I could just click on Errata [in the type filter on the left] if I just wanted them*". She suggested the text "asc" and "desc" instead of just the sorting arrows - she felt that the arrows alone were too visually subtle. She also was not sure how the "Annotation Information" column would be sorted.

For the detailed view overlay, like User A, she tried to click off the detailed view box to close it, which did not work. She also suggested using a "zoom out" icon when in the detailed view, to "*match the zoom in icon*" which opened the detailed view in the first place.

The above feedback can be summarised as follows:

- It was apparent that the behaviour of the checkboxes was not immediately obvious to users and that this needed to be reevaluated.
- Additionally, the table often updated too quickly for users to notice that it had changed.
- The cursor icons on hover needed to be changed, to make it very clear where and when users could click on things (e.g. on the sort arrows but not on the header text).
- The sort arrows were not visually obvious enough.
- The purpose of the Reset button was not immediately apparent, and users got confused as to whether or not it was related to the username search box.
- The purpose of the username search box was not clear enough, and users did not expect the automatic updating behaviour (of the table) associated with the search box.
- Users expected to be able to click off the detailed view to close the overlay.

In addition, the evaluator also noticed a number of bugs and issues with the interface during the usability tests:

- A possible fix for the DataTables CSS `<th>` bug needed to be investigated: at least one user noticed it and commented on the fact the the table header did not always line up.
- It was observed that clicking on the URL in a row triggered the OnClick event to open the detailed view. So, if a user clicked on the URL, the detailed view would open, and only then would the user be taken to the relevant Everything Maths/Science webpage. This needed to be fixed.
- It was also noticed that if a user had searched for a username, navigated away from the web page (e.g. to the Everything Maths/Science URL) and hit "Back" in the browser to return to the interface, the filters would all be reset on the page but the last string the user had searched for would still be visible in the username search box, which was misleading.
- It was noted that the numeric date sort did not work correctly because of the order in which the date components (year, month etc) were parsed by the JavaScript.
- It was also noted that not one user pressed the [Esc] key to exit the detailed view.

Bibliography

[1] Y. Rogers, H. Sharp, and J. Preece, *Interaction Design: beyond human-computer interaction*, 3rd ed. Chichester: John Wiley & Sons Ltd, 2011.

[2] D. Thompson, Ed., *The Concise Oxford Dictionary of Current English*, 9th ed. New York: Oxford University Press, 1995.

[3] B. Haslhofer, J. Wolfgang, R. King, C. Sadilek, and K. Schellner, "The LEMO annotation framework: weaving multimedia annotations with the web," *International Journal on Digital Libraries*, vol. 10, no. 1, pp. 15–32, 2009.

[4] I. A. Ovsiannikov, M. A. Arbib, and T. H. McNeill, "Annotation technology," *International journal of human-computer studies*, vol. 50, no. 4, pp. 329–362, 1999.

[5] C. C. Marshall, "The future of annotation in a digital (paper) world," in *Proceedings of the 35th Annual Clinic on Library Applications of Data Processing: Successes and Failures of Digital Libraries*, 2000, pp. 43–53.

[6] C. C. Marshall and A. B. Brush, "Exploring the relationship between personal and public annotations," in *Digital Libraries, 2004. Proceedings of the 2004 Joint ACM/IEEE Conference on.* IEEE, 2004, pp. 349–357.

[7] M. Agosti, N. Ferro, I. Frommholz, and U. Thiel, "Annotations in digital libraries and collaboratories-facets, models and usage," in *Research and Advanced Technology for Digital Libraries.* Springer, 2004, pp. 244–255.

[8] R. A. Arko, K. M. Ginger, K. A. Kastens, and J. Weatherley, "Using annotations to add value to a digital library for education," *D-Lib Magazine*, vol. 12, no. 5, p. 2, 2006.

[9] "Microsoft Office." [Online]. Available: http://office.microsoft.com

[10] "Adobe Reader." [Online]. Available: http://get.adobe.com/reader/

[11] "Google Drive." [Online]. Available: https://drive.google.com/

[12] "A.nnotate.com." [Online]. Available: http://a.nnotate.com/

[13] "AnnotateIt." [Online]. Available: http://annotateit.org/

[14] T. Berners-Lee, J. Hendler, O. Lassila *et al.*, "The semantic web," *Scientific American*, vol. 284, no. 5, pp. 28–37, 2001.

[15] V. Uren, P. Cimiano, J. Iria, S. Handschuh, M. Vargas-Vera, E. Motta, and F. Ciravegna, "Semantic annotation for knowledge management: Requirements and a survey of the state of the art," *Web Semantics: science, services and agents on the World Wide Web*, vol. 4, no. 1, pp. 14–28, 2006.

[16] J. Kahan, M.-R. Koivunen, E. Prud'Hommeaux, and R. R. Swick, "Annotea: an open RDF infrastructure for shared web annotations," *Computer Networks*, vol. 39, no. 5, pp. 589–608, 2002.

[17] S. Handschuh and S. Staab, "Authoring and annotation of web pages in CREAM," in *Proceedings of the 11th international conference on World Wide Web*. ACM, 2002, pp. 462–473.

[18] "Amaya." [Online]. Available: http://www.w3.org/Amaya/

[19] "Annotea." [Online]. Available: http://www.w3.org/2001/Annotea/User/Tutorial/quicktutorial.html

[20] "Annozilla." [Online]. Available: http://annozilla.mozdev.org/

[21] "Annotator." [Online]. Available: http://okfnlabs.org/annotator/

[22] "Annotator API Documentation." [Online]. Available: https://github.com/okfn/annotator/wiki/Storage

[23] D. Nichols, D. Pemberton, S. Dalhoumi, O. Larouk, C. Belisle, and M. Twidale, "DEBORA: Developing an interface to support collaboration in a digital library," in *Research and Advanced Technology for Digital Libraries*. Springer, 2000, pp. 239–248.

[24] "DEBORA." [Online]. Available: http://www.comp.lancs.ac.uk/computing/research/cseg/projects/debora/set.html

[25] "Mojiti." [Online]. Available: http://web.archive.org/web/20071019205040/http://mojiti.com/

[26] "The Internet Archive." [Online]. Available: http://archive.org/web/

[27] "YouTube Annotations." [Online]. Available: https://www.youtube.com/t/annotations_about

[28] "Vannotea." [Online]. Available: http://www.itee.uq.edu.au/eresearch/projects/vannotea

[29] "Annotea sidebar documentation." [Online]. Available: http://www.itee.uq.edu.au/eresearch/filething/files/get/projects/vannotea/Annotea%20Sidebar%20-%20User%20Documentation.pdf

[30] "Mozilla WebAnnotator addon." [Online]. Available: https://addons.mozilla.org/en-US/firefox/addon/webannotator/

[31] "WebAnnotator." [Online]. Available: http://perso.limsi.fr/xtannier/en/WebAnnotator/

[32] "Yawas." [Online]. Available: http://www.keeness.net/yawas/index.htm

[33] "Google Bookmarks." [Online]. Available: https://www.google.co.za/bookmarks/

[34] L. Denoue and L. Vignollet, "An annotation tool for Web browsers and its applications to information retrieval." in *RIAO*. Citeseer, 2000, pp. 180–195.

[35] "Hypothes.is." [Online]. Available: http://hypothes.is/

[36] C. Abras, D. Maloney-Krichmar, and J. Preece, "User-centered design," in *Encyclopedia of Human-Computer Interaction.*, W. Bainbridge, Ed. Great Barrington, Massachussetts: Berkshire Publishing Group LLC, 2004.

[37] J. D. Gould and C. Lewis, "Designing for usability: key principles and what designers think," *Communications of the ACM*, vol. 28, no. 3, pp. 300–311, 1985.

[38] H. Beyer and K. Holtzblatt, *Contextual design: defining customer-centered systems*. San Francisco: Morgan Kaufmann Publishers, 1998.

[39] J. Johnson and A. Henderson, "Conceptual models: begin by designing what to design," *interactions*, vol. 9, no. 1, pp. 25–32, January/February 2002.

[40] A. Dix, J. Finlay, G. Abowd, and R. Beale, *Human-computer Interaction*, 3rd ed. Harlow, Essex: Pearson Education Limited, 2004.

[41] B. Shneiderman and C. Plaisant, *Designing the user interface: Strategies for effective Human-Computer Interaction*, 4th ed. Boston, MA: Addison Weasley, 2004.

[42] J. Nielsen and R. Molich, "Heuristic evaluation of user interfaces," in *Proceedings of the SIGCHI conference on Human factors in computing systems*. ACM, 1990, pp. 249–256.

[43] C. Spinuzzi, "The methodology of participatory design," *Technical Communication*, vol. 52, no. 2, pp. 163–174, 2005.

[44] J. T. Hackos and J. Redish, *User and task analysis for interface design.* New York: John Wiley and Sons, Inc., 1998.

[45] W. O. Galitz, *The essential guide to user interface design*, 2nd ed. New York: John Wiley & Sons, Inc, 2002.

[46] J. L. Gabbard, D. Hix, and J. E. Swan, "User-centered design and evaluation of virtual environments," *Computer Graphics and Applications, IEEE*, vol. 19, no. 6, pp. 51–59, 1999.