Honours Project Report

Bonolo: A Web-Based Digital Repository System

Stuart Hammar
shammar@cs.uct.ac.za

Supervised By: Dr Hussein Suleman
hussein@cs.uct.ac.za

	Category	Minimum	Maximum	Chosen
1	Requirement Analysis and Design	0	20	15
2	Theoretical Analysis	0	25	0
3	Experiment Design and Execution	0	20	15
4	System Development and Implementation	0	15	10
5	Results, Findings and Conclusion	10	20	10
6	Aim Formulation and Background Work	10	15	10
7	Quality of Report Writing and Presentation	10		10
8	Adherence to Project Proposal and Quality of Deliverables	10		10
9	Overall General Project Evaluation	0	10	0
Total Marks		**80**		**80**

Department of Computer Science

University of Cape Town

2011

Abstract

Most digital repository systems (DRSes) manage digital collections of information that are storied in predefined complex file structures, such as databases. Data stored in complex file structures are not easily distributable, nor are they easy to preserve or establish, but exploring their information is robust and fast. Few architectures make use of simple file structures to store digital collections. Simple data structures can make preservation and dissemination of digital objects easier. But it was uncertain whether performance and usability would be affected by interacting with a simple data store. Therefore, an opportunity existed to explore whether a Web-based DRS that interacts with a simple file-based collection is feasible.

A Web-based end-user interface was built to allow users to explore and interact with an XML-centric digital collection. The project studied whether or not the underlying data store affected the performance and usability of the system. It was shown that the overall end-user experience when using the system was not hindered by the collection, and that the interface was comparable to other DRSes. Furthermore, when the collection size was scaled up, the performance remained acceptable. Consequently, the system shows positive signs and potential toward the development of a complete Web-based DRS based on a simple hierarchical file store.

Keywords

Digital libraries, end-user interface, information retrieval, hierarchical file store

Acknowledgements

First and foremost, I would like to thank my project supervisor and mentor, Dr Hussein Suleman. His organised attitude and passion about the topic really helped to ensure that a high-quality project was produced. His patience was really tested with the multitude of questions I asked him, and for that I am grateful.

A thank you is deserved by Lighton Phiri, a Masters student who gave us regular updates of the repository that served as the core in the Bonolo project.

I also have to thank Miles Robinson – my friend, my project partner and developer of the curator interface. We spent many stressful and joyful hours packed with laughs and fun in the creation and completion of our project.

The Honours class of 2011 really is a class to remember. It was so diverse and filled with students from all walks of life, each with a story to tell and a sense of humour to explore and exploit. I am really lucky to have been a part of such a great crowd of students, all of whom are going to go far in life and that I would love to work with again in the future.

To all of the anonymous users – you know who you are – who evaluated my system during the second and final iterations, thank you!

A final big thank you goes to my family and friends. All of you had to deal with my stresses and long working hours and managed to keep me sane. Thank you to my parents for providing me with the opportunity to pursue a degree in Business Science and Computer Science at UCT – without you, who knows where I would be today!

--ttfn

Contents

List of Figures

List of Tables

1. Introduction

1.1. The Bonolo Project

Custom Digital Repository Systems (DRSes) manage collections of data stored in predefined file stores. These file stores can range from complex databases to a simple file hierarchy containing only files and folders. However, few DRSes make use of simple file stores to store collections.

The goal of Bonolo is to develop a Web-based interface to explore and interact with a predefined XML-centric digital collection. These interfaces will provide services such as search and browse to end-users. Furthermore, administrators will be provided with methods to manipulate digital collections and their metadata. These tools will be made accessible via two Web-based user interfaces, namely the end-user and curator interfaces.

Additionally, research will be concluded to determine whether the use of a hierarchical file-based data store creates any significant limitations in the development of a DRS.

1.1.1. Motivation

Current DRSes provide tools that use central databases and other data stores. These data stores are often considered complex and the information they store is not easily distributable. This exposes a research opportunity to investigate whether building a DRS that interacts with a simple file-based data store is feasible.

The CALJAX project attempted to create an experimental offline DRS using Web 2.0 technologies [1]. This system requires only the use of a Web browser and does not require any pre-installed software. CALJAX was built around providing services for and manipulation of a predefined hierarchical data store.

The idea of CALJAX paves the way forward for the development of a DRS that takes advantage of the user's Web browser for accessing a simple data store. However, the focus of CALJAX was to develop a DRS that was offline and easily distributable.

The opportunity exists for an online DRS built on a simple file store to be explored. Thus, the aim of Bonolo is to develop a general Web-based DRS solution that provides powerful end-user and collection management tools to explore and manipulate a predefined simple file-based data store.

1.1.2. The Framework

The framework of Bonolo comprises three distinct sections: the predefined digital collection; the curator interface; and the end-user interface. The end-user interface and the curator interface are both independent of each other. However, they both require access to the digital collection to provide a set of services that allows the user to explore and manage the repository. Figure 1 provides a high-level overview of how these components interact.

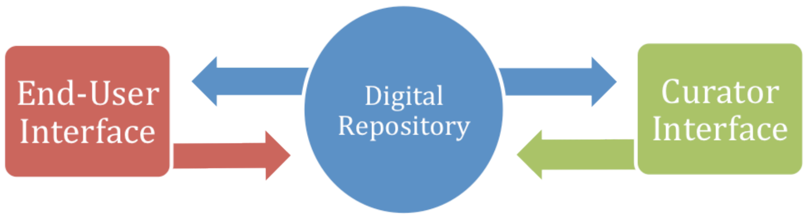
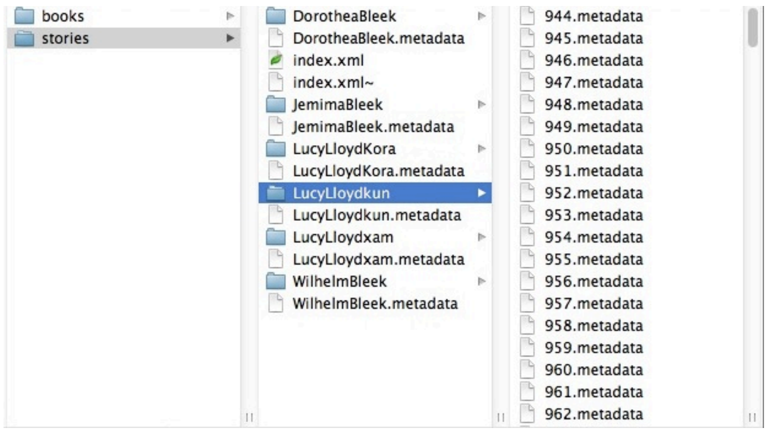


Figure 1 High-level overview of the component interaction

1.1.3. The Predefined Digital Collection

The predefined repository that was provided for Bonolo was a generalised version of a subset of the Bleek and Lloyd collection that Masters student Lighton Phiri had been working on. It presents itself at the core of the end-user and curator interfaces. The Bleek and Lloyd collection is based on an XML-centric solution[1], as opposed to the traditional database approach [2]. This means that the collection is made up of a number of XML files - which serve as metadata - and high-resolution images that are the preserved notebooks. All of these files are organised into corresponding folders (see Figure 2).



```
books                DorotheaBleek              944.metadata
stories              DorotheaBleek.metadata     945.metadata
                     index.xml                  946.metadata
                     index.xml~                 947.metadata
                     JemimaBleek                948.metadata
                     JemimaBleek.metadata       949.metadata
                     LucyLloydKora              950.metadata
                     LucyLloydKora.metadata     951.metadata
                     LucyLloydkun               952.metadata
                     LucyLloydkun.metadata      953.metadata
                     LucyLloydxam               954.metadata
                     LucyLloydxam.metadata      955.metadata
                     WilhelmBleek               956.metadata
                     WilhelmBleek.metadata      957.metadata
                                                958.metadata
                                                959.metadata
                                                960.metadata
                                                961.metadata
                                                962.metadata
```

Figure 2 A screenshot of the Bleek and Lloyd XML-centric repository

The generalised repository became the cornerstone for the structure of the information that would be made searchable by the search infrastructure. In terms of this collection there are two primary sections that are interconnected, namely books and stories, which each require their own separate search index. The structure of the collection is straightforward:

- There is a root directory.
- Within the root directory there are two folders, one called books and another called stories (this is the case for the Bleek and Lloyd collection, but it may differ for other collections).
- Within these two folders there are more sets of files and folders. For each file and folder there exists a ".metadata" file to describe it and its contents.

This structure is the generalised structure that the Bonolo end-user interface has been based on. Therefore, any collection stored in this manner can, in theory, be displayed using the

[1] It is also referred to as a simple hierarchical data store, collection or repository throughout the report.

Bonolo end-user interface. Due to the focus on the simple hierarchical file store, the system has been donned *Bonolo*, which means “easy” or “simple” in Sotho.

The Curator Interface

The curator interface is comparable to a collection management system. This allows the user to add, edit and delete digital objects from the collection. Miles Robinson focused on the development of the curator interface by making use of JavaScript, JavaServer Pages (JSPs) and Ajax. The interface interacts directly with the repository, in its natural file structure, without adding it to a database of any kind. The features supported in this interface are:

- Log in and registration functionality.
- Browse through the file structure of the collections.
- Add new resources to the collection:
    - Support for single file upload.
    - Support for batch uploads (as a zipped file).
- Delete resources from the collection.
- Download resources from the collection.
- Edit the metadata of the digital objects in the collections.
- Can view a log of all the actions conducted on the interface.
- Access control on the collections:
    - Users can request access to collections, which is then approved or denied by the administrator.

With the aforementioned features, this component of the project aims to meet the requirements of providing a set of curatorship tools that perform with minimal hindrance even though they are interacting directly with the simple file store.

The End-User Interface

The end-user interface provides a set of tools that allows the users to search and browse the information stored within the repository. The aim of this interface is to provide users with a way to access the metadata and images stored within the repository without compromising the overall user experience. Succeeding in meeting this aim will provide the end-user with a system that is transparent and responsive, despite the simple hierarchical file store used. This report focuses on the development and implementation of the end-user interface as a component of the Bonolo framework.

1.2. The Bonolo End-User Interface

1.2.1. Problem Statement

Databases and other complex data structures can make the discovery of digitised information robust and fast for the end-user. However, this poses a number of problems when it comes to distributing and preserving digitised information. Databases present problems of not being easy to set-up, their data is not easily exportable and transferrable, and often database software is not platform-independent. This leads to confusion as to how to disseminate information and also how to make this information preservable in the future. These problems raise the question of whether one needs to use databases in constructing a DRS or whether using a simpler data structure is possible.

1.2.2. Motivation

The motivation behind the Bonolo end-user interface is to try to address the problems identified in the problem statement. The development of a generalised end-user interface built on a simple file store exposes the possibilities of allowing for the simple and rapid creation of new digital repositories.

Developing an open-source end-user interface, coupled with its generic feature set, has the potential to be used by all kinds of people in making data accessible and discoverable to its users. In addition to this, the open-source nature of the Web application would allow developers to take advantage of its extensibility and componentisation. Thus, helping to create a more robust DRS.

Using a hierarchical file store has expected disadvantages when it comes to response times and robustness. However, removing the need for databases makes the digital collection more easily preservable, more readily accessible by end-users, more extensible for developers and more easily distributable by curators. This report will explore whether a non-database solution is, in fact, viable. And if it is viable, how will its consequences affect the Digital Library community?

1.2.3. Research Questions

Given a generalised XML-centric digital collection – the Bleek and Lloyd notebooks – can an end-user interface be developed that provides an experience that is beneficial and responsive to the end-user, while still maintaining the simple file structure of the digital collection?

The Bonolo project addresses three research questions that will meet this abovementioned aim. The three questions that apply to the end-user interface are as follows:

1. Will using a simple file hierarchy as a file store affect the overall end-user experience when using the end-user interface?
2. What is the impact of using a hierarchical file-based data store on the performance of the end-user interface?
3. Is it possible to create an end-user interface, built on a simple file store that is comparable to other DRSes?

To answer these questions, the experience of end-users, coupled with the performance metrics of the system, will be analysed.

1.2.4. The Framework

The Bonolo end-user interface is a Web application that allows for users to explore and interact with the underlying repository. Furthermore, it emulates the standard features of digital repository systems, such as searching and browsing the collection and viewing individual objects. In addition, it has a numerous supplementary features for added value for the end-user. Because the system is a Web application, it relies heavily on user input, so the Web pages being viewed by the user are dynamically created. An overview of the framework of the Bonolo end-user interface is shown in Figure 3.

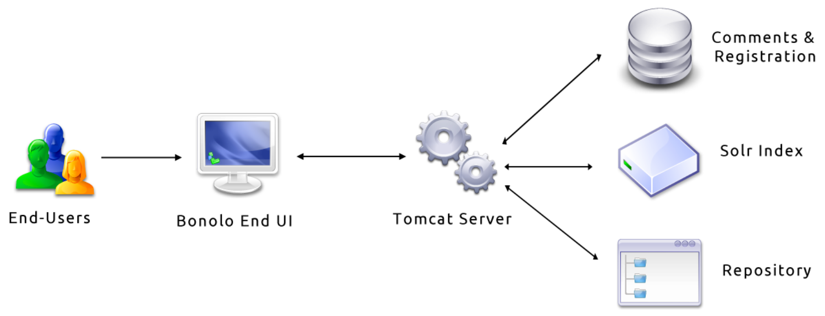


Figure 3 Bonolo end-user interface components and framework

Bonolo End-User Interface

The Bonolo end-user interface allows the users to explore and navigate the digital collection:

- Offers core discovery and exploration services to the end-user, namely search and browse.
- The user can browse the individual books (from cover-to-cover).
- The user is able to view stories from beginning to end, as well as the metadata associated with it.
- Additional features are offered to the end-user:
    - Commenting on stories.
    - Downloading the book and story images for offline viewing.
    - Receiving email notifications.
    - Account registration.

Tomcat Server

The server houses the various applications that return information to the end UI. It allows these applications to communicate with the comments and registration database, Solr index and the repository itself. The applications then return any results to the end-user interface for the benefit of the user.

Comments and Registration Database

The comments and registration database is the only database used in the Bonolo end-user interface. It contains the information of the registered users, as well as any comments they have made. The tables and fields used in this database are illustrated in Figure 8.

Solr Index

The Solr index contains all of the metadata in the digital repository in an inverted index. This allows for search results to be returned in a fair amount of time to the end-user.

- Allows for a full-text search of the repository.
- Returns a MoreLikeThis search so that similar documents are returned.
- Allows for faceted searching to be conducted by the user.

- The user is able to filter their searches.
- Returns the search results, for all of the above, to the end-user.

Repository

The digital repository is stored on the server. It contains all of the metadata files, high-quality images, as well as thumbnails of the images. These images are returned to the user when they are viewing either a book or story, or when conducting a search.

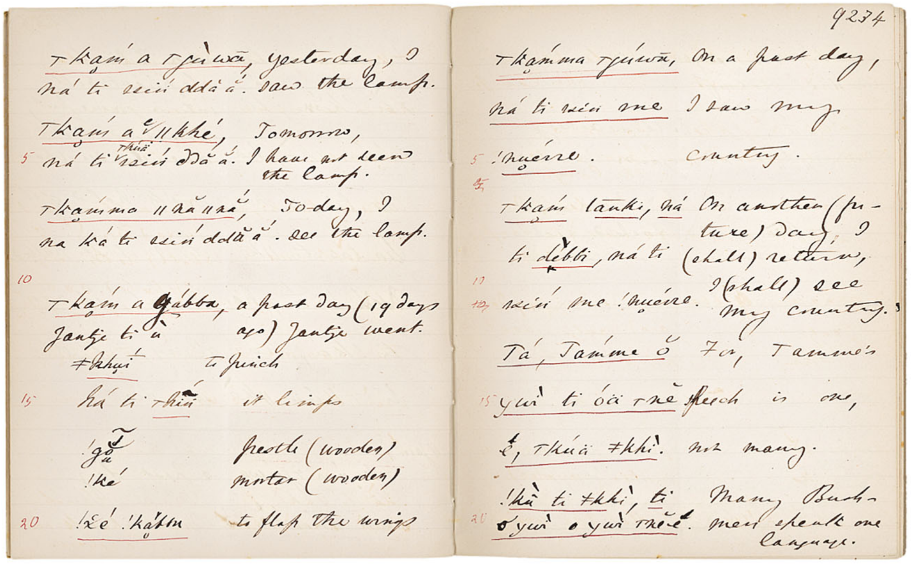
1.2.5. Description of Test Bed Data

The Bonolo Web application relies on two data sources for its services to function properly, both of which are housed in the predefined digital repository. Within the repository there are ".metadata" files, which contain information about the files and folders they are related to. These XML files are a necessity in setting up the search engine. In addition to these files is a number of image files. The image files are stored as high-quality JPEGs. These images are of the original notebooks that were created over four hundred years ago. Each and every cover, spine and page from these notebooks has been photographed on a white background (see Figure 4).

1.2.6. Ethical, Professional and Legal Issues

All of the software used in the Bonolo project is open source so that, if the project were to be continued, there would not be any licensing or legal issues. Apache Solr, Apache Tomcat and ImageMagick are all made available under the Apache License 2.0 [3-5]. The jQuery JavaScript library, as well as the qTip library are made available under the MIT license [6], [7]. The use of SQLite is completely free, and no licenses are necessary for its use [8].

To conduct ethical user testing, ethical clearance was granted from the University of Cape Town, as well as clearance for allowing the use of the university's students in testing. The Faculty of Science Research Ethics Committee and the Department of Student Affairs granted these two forms of ethical clearance respectively. To further ensure that the ethics of user testing were upheld, each user that performed an evaluation had to read and sign an informed consent form, which explained the evaluation and ensured that their feedback remains anonymous in this report.



ŧkąm a ŧgúwã,	yesterday, I
ná ti ssiń ddâ ã.	saw the camp.
ŧkąm a ‖khé,	Tomorrow,
ná ti thúã ssiń ddâ ã.	I have not seen the camp.
ŧkąmma ‖nã‖nã,	To-day, I
na ká ti ssiń ddâ ã.	see the camp.
ŧkąm a gábba,	a past day (19 days
Jantje ti u	ago) Jantje went.
ŧkhųi	to pinch
ná ti ŧhiń	it limps
!gũ	pestle (wooden)
!ká	mortar (wooden)
!ẓé !kábbu	to flap the wings

9234

ŧkąmma ŧgúwã,	On a past day,
ná ti ssiń me	I saw my
!ŋuérre.	country.
ŧkąm ĺauki, ná	On another (fu-
	ture) day, I
ti débbi, ná ti	(shall) return,
ssiń me !ŋuérre.	I (shall) see my country.
Tá, Tamme ǒ	For, Tamme's
yui ti óá ŧne	speech is one,
ẽ, ŧkúä ŧkhi.	not many.
!kũ ti ŧkhi, ti	Many Bush-
ǒ yui o yui ŧne.	men speak one language.

Figure 4 A page from the Lucy Lloyd !kun notebooks

1.3. Summary of Report

The report main begins by discussing the background of Digital Repository Systems and exploring similar work that has been conducted in the field. Following this, the design and implementation of the Bonolo end-user interface is discussed. Subsequently, the final system is evaluated in terms of system performance, as well as through user experience testing. In addition, results are drawn from the evaluations. Thereafter, future work that could be conducted is explored. Finally, the document closes with conclusions being drawn from the results of the project.

2. Background

2.1. Introduction

Digital repository systems allow for digitised copies of physical objects to be stored in them. Furthermore, these systems allow users to manipulate, explore and interact with these digital objects through various services. However, a perfect solution to storing mankind's heritage, in a digital form, does not exist. A universal, reconfigurable, scalable, easily searchable and preservable digital repository system is required. To develop such a system, the challenges, needs, successes and failures of current and previous solutions need to be uncovered.

In this section, various Digital Library (DL) architectures and their components have been discussed, analysed and compared. The DL architectures that are discussed include the Flexible and Extensible Digital Object and Repository Architecture (Fedora) [9]; DSpace [10]; the National Science Digital Library (NSDL) architecture [11]; the Tufts Digital Library (TDL) architecture [12]; and CALJAX [1]. Due to the lack of standardisation of digital library architectures, the comparison and evaluation of systems is challenging.

The topic of digital library architectures is a broad one. Only a few specific areas will be discussed in this report. The focus is on the challenges faced by digital library architectures.

2.2. Challenges Facing Digital Library Architectures

Kuny and Cleveland [13] highlight that the Internet may change the fundamental concept of a library in the 21st century; they were right. The importance of a DL model – or more generally known as a digital repository model – is essential in going toward the future and for disseminating information.

The creation of a successful digital repository system (DRS) poses many technological challenges. Information of all formats – video, audio, image and text – needs to be stored in collections [13]. These collections need to be made accessible and discoverable by multiple users simultaneously. They also need to be easily copied for backup, preservation and dissemination purposes.

The main challenges of digital library architectures include: data and metadata storage; searching and exploring the digital collections; creating digital collections; curatorship and access control; and digital preservation.

2.2.1. Data Storage

The way digital repository systems store their data – or digital objects – can influence various choices on preservation, interoperability and dissemination. Three different storage techniques are discussed below.

DSpace offers two methods for storing digital content [14]. The first is in the file system on the server. The second is using the Storage Resource Broker (SRB). Both methods can be achieved using an API. SRB is suggested as an optional file storage system or to be used in conjunction with the server file storage system. The data objects are stored in and retrieved from a file system via the bitstream storage manager API [15]. The relationships between the data, the bitstream information and metadata are stored in a relational database on the server.

On the other hand, the CALJAX DRS is database-free and stores its data in an XML-centric repository [16]. The central repository contains a collection of digital objects and metadata. These are stored as files in hierarchical directories, where each file is associated with a metadata file. This allows digital collections to be easily distributed. To distribute the repository, the central repository's contents are simply copied on to a removable media device. This is a very simple and lightweight system in comparison to DSpace's heavyweight infrastructure.

The NSDL system is a Networked Digital Library (NDL) system. Its actual digital objects are stored on various servers. Using a metadata repository (discussed in *Section 2.2.2*), the metadata information is made accessible to its services [11]. The digital content is made accessible through HTTP or FTP linked via the object identifier in the metadata.

DSpace and CALJAX support almost every file type available [1], [15]. However, only text-based file formats (e.g. PostScript, PDF, ASCII text, HTML and Word documents) are supported by NSDL [11].

2.2.2. Metadata Storage

Metadata is ultimately data about data. However, from an architectural perspective, there appears to be no significant difference between metadata and data (i.e. metadata is data in its own right). Librarians make use of metadata to catalogue printed information [13], [17], [18]. Additionally, they make use of a fixed vocabulary for describing the data in the collections they keep [18]. This paradigm can be transposed to the digital arena – curators make use of metadata to catalogue digital objects. Conducting searches on indexed metadata information is more efficient than searching full-text documents. To aid with interoperability and to create standardisation, many digital toolkits and architectures (e.g. DSpace, Fedora and NSDL) make use of standard metadata formats such as Dublin Core (DC).

The Open Archives Initiative (OAI) has developed a Protocol for Metadata Harvesting (OAI-PMH) as a promising method for connecting data providers to service providers [15], [19]. The OAI-PMH is a client-server protocol. Providers use the OAI-PMH to expose metadata in different ways, and the service providers use the protocol to gather or harvest the metadata. The providers can then process the data and add value to it in the form of services [20], such as a search or citation service. Shearer [20] highlights that using the OAI-PMH helps with the interoperability of repositories so that they can contribute to a larger global system. Following from the OAI's standard, Suleman [21] supposes that standardised components can then be designed for DLs. He goes on to state that these components could include search engines and browsing services, both of which are important for resource discovery and dissemination.

DSpace and Fedora are free and publicly available digital library frameworks. Both DSpace and Fedora support multiple, if not all, digital file formats in their repositories. Fedora makes use of various APIs to allow applications to interact with its data repository [15], [22]. It makes use of metadata and data objects, which are stored in XML and databases. However, Fedora does have software prerequisites and does not allow for distributable copies of its collections to be made. Furthermore, Fedora is only a framework and is not a complete system.

DSpace architectures maintain a DC metadata record for each data object that is stored. Three DC fields are made compulsory per item, namely the title, language and submission date [10]. All other metadata fields are made optional. Each metadata record is stored in a relational database on the server [14]. DSpace supports the OAI-PMH as a data provider. Only the basic unqualified DC metadata set export is enabled by default; this is easy for DSpace since it stores DC metadata for each item. Therefore, the inclusion of the DC record simplifies the sharing of metadata and interoperability.

The NSDL manages its metadata differently to the previously mentioned architectures. This is because it is a networked digital library. The metadata that it gathers from its various independent collections are not all in the same format – consequently it supports eight standard formats of metadata [11]. The NSDL gathers metadata and stores it in a central metadata repository. To populate the metadata repository, the NSDL harvests metadata in five ways: via OAI; via FTP, e-mail or Web-upload; through direct entry; and by using a Web crawler. Constructing this metadata repository allows for a faster searching service than if all the metadata were searched on the distributed network [23].

There is no single standard for metadata or data in DL systems. Therefore, deciding which metadata to store and how to store it is a challenge and each DL system handles this differently. The OAI-PMH has provided a way for repositories to become interoperable and is a step towards a standardised component.

2.2.3. Searching and Exploring Digital Collections

Information on the Internet exists in many formats [13]. Search engines provide a means for users to search the Internet for information. A search service is an integral part of DRSes. However, searching the full-text of each document is an inefficient and expensive process [1].

As discussed in *Section 2.2.*1, CALJAX is a lightweight architecture for creating digital collections. CALJAX is a database-free DRS. The system is OS independent and can be run directly from a portable media device [16]. No preliminary software needs to be installed; the user only needs a Web browser to search and manage the digital collections. CALJAX also supports almost every file type as long as a metadata file is associated with it [1].

CALJAX provides users with a completely offline digital repository system built on Ajax [1]. CALJAX makes use of Java to pre-process and generate an inverted index of its metadata. The inverted index is created to facilitate an indexed search. The indexed search is then performed using JavaScript and the results are displayed to the user via the Web browser.

CALJAX removes the necessity for a heavyweight back-end to do processing and instead uses the power of the Web browser.

As discussed earlier, the NSDL makes use of a centralised metadata repository. The reason for this is that using a distributed style of searching – such as in the Dienst system – does not scale well [23]. As the number of independent servers grows, the services become less reliable and less responsive. Moreover, in the NSDL, it is not expected that each system understands the same query formats, making a distributed search difficult.

The search engines in both TDL and DSpace make use of the Lucene library [12], [15]. TDL makes use of an indexing and search function to make its data available to users [12]. TDL uses methods to present the content of a digital object to the search engine. This is indexed and returned as DC metadata. Using this type of indexing allows full-text and advanced searching to be conducted, in addition to metadata searching.

2.2.4. Creating Digital Libraries

The creation of digital libraries poses the question as to whether institutions should use standalone architectures (e.g. DSpace or Fedora) or whether networked DL architectures (e.g. NSDL) should be focused on.

NDLs are meant for sharing resources among DLs of similar interests and content [24]. The NSDL has integrated many smaller digital libraries to provide information to people all over the world. Historically, the largest NDL on the Internet was the Networked Computer Science Technical Reports Library (NCSTRL) [25]. NCSTRL uses Dienst as its framework. Dienst is a system for organising a set of separate services running on networked servers to cooperate in providing the services of a digital library.

Of the systems discussed, CALJAX is the easiest to set up. It is OS independent and does not need any pre-installed software. It is a proof-of-concept system and its experimental results indicate that it is a feasible system [16]. However, when scalability and collection size are noted, DSpace, Fedora and the NDL architectures seem to become more feasible system choices.

2.2.5. Curatorship and Access Control

Kuny and Cleveland [13] point out that librarians are an authority and they, therefore, provide a trusted service. DL systems need to be carefully audited to ensure that the information and collections that they store are trusted. To guarantee that digital collections are managed correctly, an authority needs to have access to adding, removing, editing and monitoring functions. These curators or administrators will be able to edit the metadata of the digital information to ensure that it remains useful and relevant.

Document discovery and retrieval can be used anonymously in DSpace. However, for a user to use submission, subscription or administration features they must be authenticated. Therefore, for a user to perform a task on an object or a collection, the user must have permission to perform that task [15]. DSpace's user interface is Web-based. End-users and curators each have a different interface where they can log in.

DSpace and Fedora are set up in such a way that a user may be granted access rights to certain collections but not others. Furthermore, their rights may only be to read information, or to edit and manage it too. In order to maintain a digital repository, various users need to have different access rights to different collections in the repository.

Administration of NDLs is more complicated than that of standalone DL systems. In the case of NCSTRL, Dienst is used to provide a means for definition and management of distributed collections [25]. For NDLs, administrators need to ensure that their repository is kept valid and up to date to certify that the information is always accessible on the NDL.

There is no standardised manner for handling access control in digital repository systems, and the architectures offer their own solutions to this. Yet, they all tend to have similarities in their access control models.

2.2.6. Preserving Digital Objects

DRSes should be able to store large amounts of information that can be accessible many years into the future.

A key use of DSpace is preservation. To support preservation of resources, DSpace associates each bitstream (the actual file) with a bitstream format [15]. Fundamentally, a bitstream format is a distinctive way to refer to a particular file format. Each file format, which a user submits, is captured in the DSpace repository. To facilitate preservation, each bitstream format has a support level, which indicates how likely it is that the content will be preserved in the future by the institution.

Similarly, Fedora makes use of DataStreams [9]. DataStreams preserve the internal format and encoding of the file type. However, Fedora stores the DataStreams inside a DigitalObject so that mixed forms of data can be treated in a uniform manner. The DigitalObjects are stored in a repository for later access.

The CALJAX DRS handles preservation differently to DSpace and Fedora. Due to its digital collection being stored simply in files and folders, its content can be transferred onto a secondary storage medium. The simplicity of the collection allows for it to be easily distributed and thus, easily duplicated so that many replicas can exist.

Hitchcock et al. [26] claim that digital repository software is not sufficient to ensure preservation of data. It is suggested that repository support teams be employed for preservation management.

2.3. Discussion

The ideas surrounding what digital repositories are have been discussed. The techniques used by current architectures have been noted and these can illustrate the way forward for new digital repository architectures.

There are similarities in all of the architectures regarding user interfaces for curators and public users. The general trend is to make use of a Web-based user interface. The architectures also make use of back-end systems, which conduct pre-processing and indexing of data, as well as searching and returning the data to the user. Only CALJAX makes use of the client's browser to perform back-end operations through the use of Ajax. Future digital libraries could potentially make use of powerful client-side scripting languages, such as JavaScript, to reduce the need for complex back-end systems.

More similarities arise with metadata management. Many of the architectures appear to have adopted the OAI-PMH for harvesting and disseminating metadata. This assists the systems with interoperability and harvesting metadata from independent systems. Making use of good metadata assists the architectures with dissemination, interoperability, as well as for making resource discovery easy. Many of the architectures employ the Dublin Core metadata standard for each of their digital items.

The browse and search functions in the architectures use metadata that has been inversely indexed as the searchable information. From the handle information in the metadata, the actual digital objects can be retrieved for the user. The reason the architectures use metadata search – as opposed to full-text search – is to aid with scalability and to improve search performance.

Currently, a perfect general solution does not exist. Each of the architectures seems to answer a specific set of user needs. The systems that answer a general set of needs appear too complicated and heavyweight. The power of the client's Web browser is only being harvested by CALJAX. The amount of processing happening at the back-end of the system could be shifted on to the browser. Thus, the back-end could be simplified greatly and made lightweight. This presents an opportunity for a simpler, more robust, preservable, scalable and standardised solution to be created.

3. Design and Implementation

3.1. Overview

The purpose of the Bonolo end-user interface is to develop a Web-based solution for exploring and discovering digital content that is stored in flat files, as opposed to a database. The Web application should offer the core services of a digital repository system to aid in accessing the digital objects, as well as additional features that may add value to the system. These additional features were decided upon by the end-users themselves by conducting a focus group.

The reason behind the lack of databases and other complex file stores is to facilitate the preservation and dissemination of digitised information. It also ensures that the interface can be easily set up for any digital collection without the necessity of creating and updating a database. The only database that was used in the end-user interface was an SQLite database for storing user information.

The outset of the project is to determine whether a system built on accessing a simple file store can offer typical DRS functionality without negatively impacting on system performance, which would detract from the overall end-user experience.

This section of the report explains the design decisions and implementation plan of the Bonolo project. It begins by detailing the technology and tools used in creating the final Bonolo interface. Subsequently, a high-level overview of the system is provided, which is then followed by a more detailed and low-level description of the interface. Finally, the iterative design process is described and is coupled with the results from each of the design iterations.

3.2. Technology and Tools

The technology and tools used in developing the Bonolo end-user interface include server-side applications, client-side applications, Web development languages, search engine libraries and server software to host the culmination of these technologies in a Web application. Each technology choice is described in this section along with the rationale for choosing it.

3.2.1. Server-Side Applications

The functionality of the end-user interface has primarily been written in Java by making use of JavaServer Pages (JSPs), Java Servlets and JavaBeans. JSPs create modularity by separating the end-user interface from content generation [27]. This allows Web designers to alter the overall appearance and layout, without affecting any of the dynamic Web content. Java Servlets also evoke modularity by delivering a component-based, platform-independent way of developing Web-based applications, without incurring the performance constraints of CGI programmes [28]. JavaBeans are classes written in Java that are reusable and can be composed together in applications [29]. Moreover, any Java class can be a JavaBean. These Java technologies have been chosen due to the extensibility and componentisation offered by them [27-29]. Java is platform-independent, which makes the Web application transportable regardless of the operating system. This will be beneficial if the project is carried forward. Thus, simplifying the ability of making digital collections accessible.

3.2.2. Client-Side Applications

JavaScript, jQuery and various jQuery libraries[2] have been used to aid in the functionality of the interface and to provide the user with a modern and interactive application. The notion behind jQuery is that it simplifies JavaScript and allows for an easier way of conducting Ajax interactions [6]. Galleria [30] is a free JavaScript library that has been used to effectively display the pages stored in the repository. Initially, a JavaScript image gallery was negated, as it would not allow the user to bookmark the page they are viewing. The reason behind this is because JavaScript conducts actions on the Web page without altering the URL. However, Galleria was chosen because it possesses a history plugin, which provides the user with the fluidity of a jQuery interface, as well as the ability to bookmark and keep browser history of the images being displayed. The Java files are all used in processing information on the server-side of the system, whereas JavaScript processes it on the client-side, within the Web browser.

3.2.3. Web Development Languages

Hypertext Markup Language (HTML) has widely been used in the creation of the website to present the output from the server-side Java applications. In addition, Cascading Stylesheets (CSS) have been used throughout the interface to ensure that the layout of the website is presentable, neat and eye-catching.

3.2.4. Search Engine Library

Apache Solr was used to make all of the metadata within the repository searchable and browsable to the end-user. Solr is a stand-alone, full-text search server [3]. It is written entirely in Java and runs within a servlet container, such as Apache Tomcat. Solr forms part of Apache Lucene, which is an open-source search engine built in Java [3]. It can handle and create indices for XML files and other rich text documents, such as PDFs. Also, it has the ability to scale well when the indices increase in size [3]. Again, the use of Solr is beneficial as it is platform-independent.

3.2.5. Miscellaneous Software

The only database that is used in Bonolo is an SQLite database, which has been implemented to store the annotation and registration information of the users of the Bonolo Web application. Other databases make use of a separate server process, but SQLite is server-less [8]. This makes it easily implemented and a lightweight database tool that is cross-platform. Therefore, it aids in making the Bonolo end-user interface platform-independent and easily distributable.

[2] qTips2 and Galleria are libraries that have been used that have been built on jQuery.

ImageMagick [5] was used in the creation of thumbnails for all of the images stored within the repository. Creating these thumbnails helped to reduce the amount of data being transferred from the server to the Web browser. In an end-user interface, the reduction in data transfer is integral in ensuring that the Web application delivers optimal performance.

3.2.6. Server Software

All of the aforementioned technology choices are hosted from an Apache Tomcat server. Apache Tomcat is an open-source server that allows for the execution of Java Servlets and JavaServer Pages [4]. This makes it ideal for this project for two reasons. The first reason is that the primary development language for Bonolo is Java. Secondly, Tomcat is also open-source and platform-independent, thus promoting dissemination and transferability of the Web application.

3.3. System Features and Functionality

3.3.1. Creating the Solr Indices

Before any of the features of the end-user interface could be developed, the information within the repository had to be added to Solr. Solr creates an inverted index of the documents that have been added to it [31]. This indexing process allows Solr to conduct full-text searches that return results in an amount of time that would be faster than searching through each individual metadata file in the repository.

To configure these indices, the *schema.xml* and *data-config.xml* files need to be created. These two files dictate which XML tags (from within the ".metadata" files) are to be indexed, which faceted searches to allow and how these fields are to be indexed. All of the metadata tags are stored as fields within the Solr index. For example, the metadata tag "`<dc:title>`" may be stored in the field named "`title`", within the *schema* file.

The *schema.xml* and *data-confix.xml* file are customised for each index that is created. In this case, the two indices that are made are called "bonolo" and "bonoloBooks". The indices are stored within the Solr home directory, which is inside the Bonolo Web application. The *web.xml* file within the Bonolo Web application is configured to direct Bonolo to the Solr search indices. Adding the following XML to the file does this:

```
<env-entry>
   <env-entry-name>solr/home</env-entry-name>
   <env-entry-value>../../../indices/solr</env-entry-value>
   <env-entry-type>java.lang.String</env-entry-type>
</env-entry>
```

Navigating to the following URLs creates the two indices:

- Stories: `<ServerIP:Port>/<webapp>`[3]`/bonolo/dataimport?command=full-import`

[3] The webapp's name is also "bonolo" in this case.

- Books: `<ServerIP:Port>/<webapp>/bonoloBooks/dataimport?command=full-import`

Once the indices have been created, the collection's metadata are searchable without needing to traverse the file structure of the repository and without the need for a database. A configuration application was developed to assist in this being general, which is discussed in *Section 3.3.6*.

3.3.2. Search and Browse

The ability to search and browse is a core feature of the end-user interface and it comprises two major parts. Firstly, processing the search requests and generating the responses needs to be conducted on the server. Following this, the search results then need to be returned and displayed to the end-user.

In terms of making the system browsable, faceted searches are implemented. Faceted searching or browsing is defined by Keith Instone as the "interaction style where users filter a set of items by progressively selecting from only valid values of a faceted classification system" [32]. The search and browse functionality of the interface is described further in this section.

3.3.2.1. Processing the Search Query

An end-user may conduct two types of search through the interface. They can either conduct a new search, or they may conduct a filtered search. A new search means that every search term entered, or search facet that is clicked, returns a fresh set of results independent of their last search.

Creating a SolrQuery

A new `SolrQuery` is created each time a new search is made. The search facets are specified when creating the `SolrQuery`, and the facets are ordered by their name, instead of by the number of search results that the facet will return. This query is transmitted and, from the `QueryResponse`, a `SolrDocumentList` is returned containing all of the matching documents (`QueryResponse.getResults()`). An example of how this works is illustrated in Figure 5. The default search uses an OR operator. This means that each search term entered, within the search string, is searched to see if there is a match. To conduct an AND search, which looks for an exact match of the entire search string, the search terms may be placed within double quotation marks.

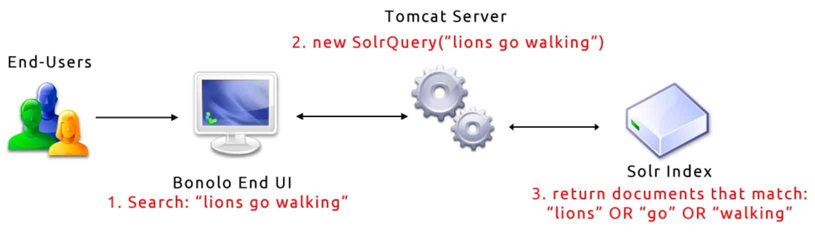

Tomcat Server
2. new SolrQuery("lions go walking")
End-Users
Bonolo End UI
1. Search: "lions go walking"
Solr Index
3. return documents that match:
"lions" OR "go" OR "walking"

Figure 5 Illustration of how a new search is conducted

Filtering a SolrQuery

A filtered search allows the user to further refine their original search – their initial `SolrQuery` – by entering a search or by clicking search facets. When a filtered search is made, a filter query is added to the `SolrQuery` using `SolrQuery.addFilterQuery(String)`. Once the filter query has been added, the `SolrQuery` is sent to the server. The server then returns a `SolrDocumentList` containing the documents within the original result set that match the filter query (see Figure 6). Filter queries can be added, as previously described, but can also be removed from a `SolrQuery` (`SolrQuery.addFilterQuery(String)`). This allows the `SolrQuery` to become unfiltered again.

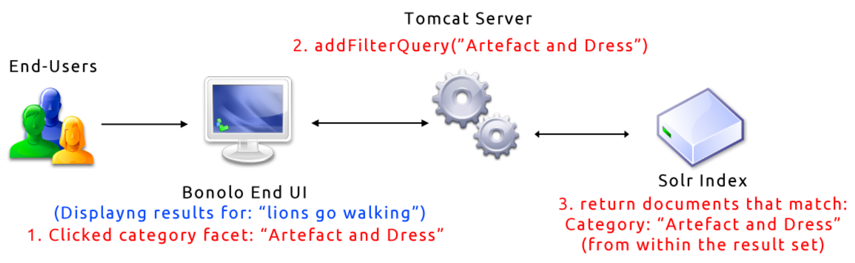

Tomcat Server
2. addFilterQuery("Artefact and Dress")
End-Users
Bonolo End UI
(Displayng results for: "lions go walking")
1. Clicked category facet: "Artefact and Dress"
Solr Index
3. return documents that match:
Category: "Artefact and Dress"
(from within the result set)

Figure 6 Illustration of how a filtered search is conducted

The JSPs communicate with the Solr index through the use of JavaBeans as opposed to a Servlet. The JavaBean that contains this functionality is run as an instance of the `Search` class. This allows for the class's accessor and mutator methods to be called so that the output from the `SolrQueries` made can be displayed to the end-user.

3.3.2.2. Displaying the Results

Once a user has conducted a search or browse, by either entering a search term or by selecting a search facet, the matching results need to be displayed in an intuitive manner. In addition to this, the search results need to be returned in a timely matter. Once a search has been submitted, the browse facets update to show the items that apply to the current search that has been made. Therefore, the page needs to be updated to reflect this.

Returning the Results

The aforementioned Search class coordinates retrieving the search results. It makes use of the SolrJ Java library to communicate with the Solr index. Each `SolrQuery` returns a `SolrDocumentList` containing the set of matching `SolrDocuments`. Each `SolrDocument` is of type `Map<String, Object>` object. As a result, each of the document's map entries is added to a `HashMap <String, Object>`. If there are any duplicate values in the `HashMap` they are removed. Finally, the `HashMap` is added to an `ArrayList` that contains all of the `SolrDocuments` that were returned in the `SolrDocumentList`.

From here the documents may be processed and returned to the interface by iterating through the `ArrayList` and returning the values corresponding to the `HashMap`'s keys, as illustrated in Figure 7. These results are formatted with HTML within the Search class and returned to the JSP to be output.

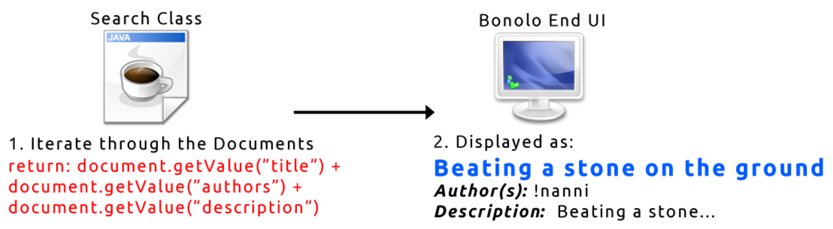

Search Class
JAVA
1. Iterate through the Documents
return: document.getValue("title") +
document.getValue("authors") +
document.getValue("description")
Bonolo End UI
2. Displayed as:
Beating a stone on the ground
Author(s): !nanni
Description: Beating a stone...

Figure 7 Illustration of how the results are generated and displayed

In addition to some metadata text being output on the search page, a thumbnail of the first story page is also output. Initially the absolute path of the image is generated from the metadata stored about the story – it is assumed that the image path is stored in the metadata. From this, the relative path to the thumbnail folder is generated and this is used to output the thumbnail image.

The default number of documents returned is 10 per page. Therefore, there is pagination of the search results. The page numbers are also formatted in the Search class and are returned to the JSP to be displayed to the user.

Updating the Search Facets

After each search is conducted, the search facets need to be updated. The facets need to be retrieved from the `QueryResponse` to output the updated facets. `QueryResponse.getFacetField("FacetName").getValues()` returns a `List` of the facet entries. These entries are then traversed and their outputs are returned for the user to browse.

3.3.3. Displaying Stories and Books

The end-user needs to be able to browse the contents of a story or book. From the search and browse page the user is able to click the story's title and is taken to a Web page that displays the story's metadata and images. Also on this page is a list of suggested stories that are closely related to the story. Another feature when displaying the story information is that a user can conduct a browse for stories that have similar topics to the viewed story. Users are also able to comment on the viewable story, as well as being able to download the story images, these features are described in *Section 3.3.4*.

Browsing books can be conducted from the home page by clicking one of the book facets. This will take the user to the respective Web page where they can view the book's pages from cover-to-cover. Again, the user has the ability to download all of the book images from this page.

3.3.3.1. Viewing Stories

Displaying the Story Metadata

The process of loading the story information contains various steps. The first step is to load the metadata associated with the story. Each story is assumed to have a unique identifier or key. Therefore, an instance of the `Search` class is instantiated as a JavaBean. A `SolrQuery` is conducted for the unique key associated to the story. This will return only one `SolrDocument`. Following the process described in *Section 3.3.2.2*, the document is iterated through and the specified values relating to the metadata tags are returned. Only the tags that are specified within the *storyconfig.xml* file (see *Section 3.3.6*) are displayed to the end-user.

Returning a List of Similar Stories

A similarity search is also computed on the story. This returns a list of 10 stories that are similar to the story being viewed, based on a number of metadata fields. Solr offers a MoreLikeThis search that performs this calculation. The MoreLikeThis search returns a `SolrDocumentList` of similar stories ordered by their similarity calculation, where 1 is the most similar story and 10 is the least similar story. This document list is traversed, and the story titles are returned to the end-user. This feature aids in providing a method of serendipitous discovery of information for the end-user.

Browse Similar Topics

Some of the metadata fields may be of interest to conduct a new search or browse directly from the story page. Therefore, some of the metadata fields are returned as a set of terms that will allow the user to browse other stories that have these terms in common. If the user clicks any of these terms, they are taken to the search page where they can find the set of results that match the selected term.

Displaying the Story Pages

Finally, the story images are displayed in a JavaScript image gallery using Galleria. Each story is assumed to contain metadata fields containing the paths to where their respective images are stored in the collection. Using Solr, the absolute paths to these files are calculated along with the relative paths so that they can be displayed to the end-user. The paths to the image thumbnails are also computed.

Galleria is set to preload two full-size images at a time. The reason behind this is so that when the end-user navigates to the next page, the image will be ready to be viewed. If a user wishes to view the page in its full size, they can click on the image and it will open in a new window or a new tab, this is Web browser dependent.

3.3.3.2. Viewing Books

Viewing the books contained in the collection is more straightforward than viewing the stories. The books do not contain metadata about where their pages are stored, only the page names are stored. A brute force algorithm is employed that searches through the digital collection for the folder that matches the book's name, as this folder is assumed to house the book's images. From here, the absolute and relative paths can be constructed so that the book's images and thumbnails are made displayable to the end-user. Galleria is then utilised to display the images in a container to the end-user. Navigating the book images is identical to navigating the story images.

On this page, the user is provided with faceted browse options that contain all of the book names in the collection. The user can then navigate between books and view their pages. These facets are generated in a similar manner to the facets described in *Section 3.3.2.2*.

3.3.4. Annotations and Registration

Registration and annotations are two of the additional features offered by the Bonolo end-user interface. These two features are the only features that make use of a database. The database system used for these features is SQLite. The tables and fields used for these two features are shown in Figure 8.

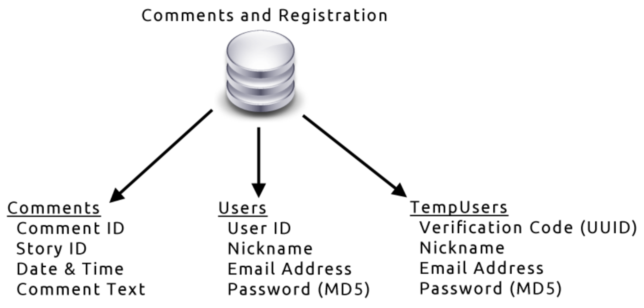


Figure 8 Tables and fields in the comments and registration database

3.3.4.1. Registration

A user has to first register an account before they are able to annotate, or comment, on a story. The reason behind needing to register before annotating is due to the fact that it will prevent spammers and flamers from abusing the commenting capabilities of the system. Also, by registering an account, a user can receive email notifications when other users comment on the same story as them (discussed in *Section 3.3.4.2*). It was important that the user does not have to log in to explore Bonolo and that logging in only provides access to additional features.

The registration process for Bonolo is kept as short and simple as possible, requiring only a valid email address, a nickname (which is suggested to the user using jQuery) and a password. The purpose of this is to reduce any user frustration, which is normally encountered when faced with a tedious registration process.

When the user is filling out the registration form, Ajax is used to validate the information on the server and to inform the user about its validity prior to the form being submitted. The validation checks are conducted by sending information to the `RegistrationServlet`, which calculates the field's validity and sends back a response. The form validation will only allow the user to complete the registration once all of their information is valid. Each text input block receives a green or red notification, to show whether the input is valid or invalid, respectively. The email and nickname inputs have an additional yellow notification – this tells the user that the email address or nickname has already been registered on the system. To further assist the user, the nickname field takes the substring of the first part of the user's email address (up to the "@" sign) and suggests it as a nickname. If the nickname, email and password fields are all valid, the user can submit the registration form.

Once the form is submitted, a verification code is generated for the user using a Universally Unique Identifier (UUID). The user's nickname, email address, UUID and an MD5 hash of their password are then stored in the TempUsers table in the SQLite database. The user will then receive an email at their registered email address with a validation link. Once the user clicks the link, their account will be validated – they are added to the Users table in the database. Now they may log in and post comments on the website.

3.3.4.2. Annotations

A user must have a registered Bonolo account before they can write comments on a story. Once a user has logged in with their account details, they are able to post comments on the story by entering information into the comment box and pressing the "Post Comment" button. From here, the user's unique ID, the story's unique ID, the comment text and the time at which the comment was made are all stored in the Comments table. When commenting on a story, a `JOIN` function is used to compute the email addresses of the users who have also commented on the same story. These users are then sent an email notification to inform them that someone else has commented on the same story as them.

If a user wishes to remove a comment that they have written, they may do so. However, they may not remove a comment that someone else has posted.

The six most recent comments (ordered by date using an SQLite `SELECT` query) are displayed on the home page along with a link to the story it belongs to.

Any SQLite queries that are conducted by either the `Comments` class or the `RegistrationServlet` are all conducted using `PreparedStatements`. This reduces the opportunity for unruly users to try to use an SQL injection attack on the system.

3.3.5. Exporting Images

Exporting the story and book images was a suggestion gathered from the focus group conducted for the second design iteration (see *Section 3.4.2*). This feature dynamically creates a compressed zip archive of the story or book's images and sends them to the user to download. The archive receives either the book title or the story, depending on which page it was downloaded from. Once the download is complete, the zip file is deleted from the server.

The steps in dynamically creating a compressed zip archive of the images (see Figure 9) involves getting the absolute path to the images, adding the files to the zip archive, sending the zipped file to the user and, finally, deleting the zip file upon transfer completion.

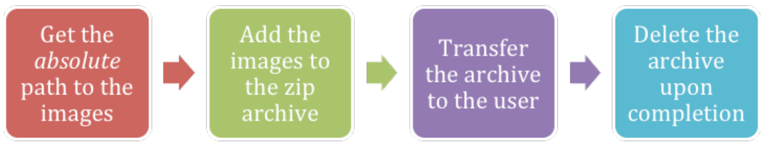

Get the *absolute* path to the images
Add the images to the zip archive
Transfer the archive to the user
Delete the archive upon completion

Figure 9 Illustration of the export process

3.3.6. System Generality and Configurability

To ensure that the entire system is kept as general as possible, an initial configuration application was written in Java. This configuration application is a command-line tool and can be found within the Web application.

Before making the system general, a couple of assumptions were made. The first assumption is that the structure of the generalised Bleek and Lloyd collection is how other general collections will be structured. Extending from this, each general collection is assumed to have a "books" folder and a "stories" folder – one for the images and their metadata, and one for the only metadata. These two folders do not have to be called "books" and "stories". Another assumption is that only images are to be viewed, and that the file extension for the metadata files is ".metadata". Lastly, the entire collection is to be stored in a root folder called "archive". This general structure can be seen in Figure 10.

The purpose of the configuration application is to allow the user to set up the configuration files for Solr and for the Bonolo end-user interface without being required to know XML or about Solr. As a result, the application generates six output files: two *data-config.xml* files, two *schema.xml* files, *storyconfig.xml* and *bookconfig.xml*. The *data-config* and *schema* files are essential in making sure that the information within the ".metadata" files are correctly indexed. The other two *config* files (*story-* and *bookconfig*) are necessary for customising what information to display on the Bonolo end-user interface.

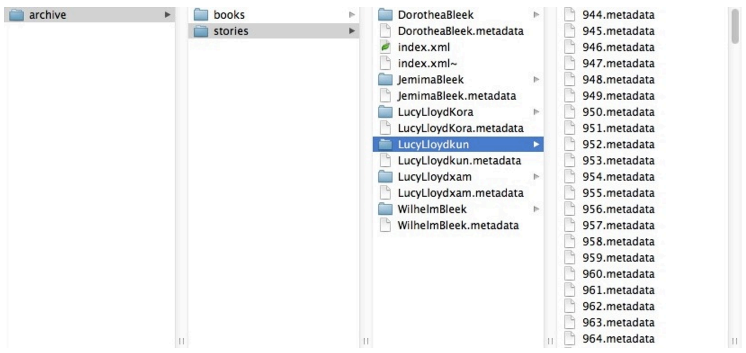


Figure 10 Structure of a generalised hierarchical file-based store

Firstly, the Solr indexing procedure needed to be customised. The tags within the metadata files and the paths to the metadata files are all different depending on the collection. Therefore, a *data-config.xml* and *schema.xml* file need to be generated for the corresponding "story" and "book" metadata of the collection. The user specifies the path to the "story" and "books" folders and the application recurses through the folders and subdirectories and finds all of the ".metadata" files and reads in their tags. These tags are then kept in an `ArrayList` maintaining only the unique tags. This is a brute force algorithm to ensure that no tags are missed.

Once these tags are collected, they are passed to a method that calculates the XPath String necessary for the *data-config.xml* files to successfully add them to the Solr index. The user is presented with a list of these tags and chooses human-readable names for them – as these will be displayed on the Bonolo end-user interface. In addition, the user chooses which tag is the primary key, and which tags contain the image name, or the path to the image name.

Following this, the *data-config.xml* file is generated. Successively, the *schema.xml* file has to be created. This file is generated from a template *schema* file. The *schema* file tells the *data-config.xml* file where to store the information that it is trying to index. In addition, the *schema* file also contains the faceted search information. A *schema* and *data-config* file are written for the "books" and "stories" in the collection.

Following this, two more configuration files – *storyconfig.xml* and *bookconfig.xml* – are written. These two configuration files dictate the "stories" and "books" are displayed on the website. These two configuration files dictate the following:

Table 1 The contents of the *story-* and *bookconfig* files

Storyconfig.xml	***Bookconfig.xml***
The list of tags stored in the index	
Which facet queries to display on the home and search pages	
The headings of the facet queries (e.g. "Browse By [custom name]")	
Which tag is considered the primary key or identifier	
Which metadata to display on the "view story" page	Which metadata to display on the "browse book" page
Which metadata tag contains the path to the image files	Which metadata tag contains the name of the image file
Which metadata tags to use when calculating the similarity of a story	
The metadata to be displayed for the search results on the search page (e.g. the title, author and summary)	
The similar topics to display on the "viewing story" page	

Once a change is made to the configuration file, the user refreshes the Web page, not the server, and the new information will be displayed. Unfortunately, due to time constraints, there is currently no way to make amendments to these files, unless one has knowledge of XML.

3.4. Design Iterations

After conducting background research into DRSes, it was decided that there be a set of core features in the Bonolo end-user interface. The core feature needed is the ability to explore the information within the collection through searching and browsing.

However, there was room for additional features to be added to the system. To ensure that the features of the interface be successfully implemented, a User-Centred Design (UCD) approach was adopted. UCD means that the developers and the end-users work together in creating a final solution.

Each of the development iterations involved either a focus group or user testing being conducted. This aided in the identification of what users expected from a DRS, the kinds of additional features they would expect to find in the interface, along with the current issues that were present in the system. The iterations are discussed below and describe the steps taken in identifying which features were implemented as the system evolved from a prototype to an almost-complete system.

It is to be noted that the Bonolo project is not an attempt at making a usable interface by implementing various heuristic evaluations, usability testing and other HCI techniques. Instead, the user testing involved was to gather insight into what users expected to see from the system and whether there were any suggestions for improvements. The focus here is to identify and deliver a set of services that would perform as the user expected.

3.4.1. Iteration 1: Feasibility Prototype

The aim of the feasibility prototype was to decide on the technologies that were going to be used in the final system and to have a primary function of the interface working.

Requirements Gathering

The feasibility prototype did not incorporate any end-user requirements gathering. Instead, a set of core features was decided on through conducting research on the main functionality of DRSes. Moreover, it was decided that any additional features of the system be suggested by the end-users, as they are whom the system would be developed for

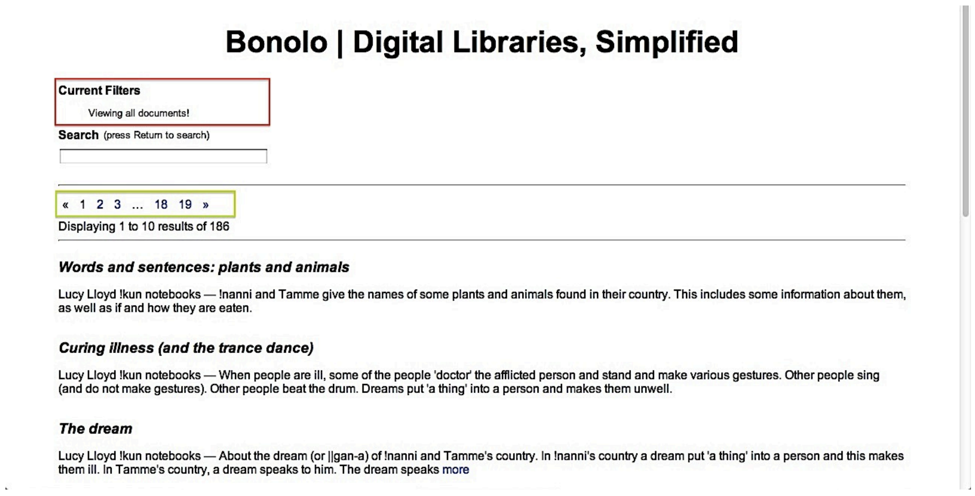
The following features were outlined as goals for the prototype:

- Allow the user to successfully search the digital collection.
- Create a low-fidelity website to show that this core feature is feasible.

Implementation

The initial prototype was developed using only JavaScript, jQuery, Ajax and HTML, which all interacted with the server. The result of the prototype can be seen in Figure 11 and Figure 12. A separate home page had not yet been developed.

The functionality of the website allowed the user to search the story objects that were in the Solr index using only the search box. The search results were paginated (see the *green boxes*) so that ten results were displayed per page. The Web page would load displaying all of the results in the collection. From here, the user was able to narrow down their search with every search conducted, as illustrated in the *red box* in Figure 12. The user could then broaden their search again by removing any search filters applied.


Bonolo | Digital Libraries, Simplified

Current Filters

Viewing all documents!

Search (press Return to search)

« 1 2 3 ... 18 19 »

Displaying 1 to 10 results of 186

Words and sentences: plants and animals

Lucy Lloyd !kun notebooks — !nanni and Tamme give the names of some plants and animals found in their country. This includes some information about them, as well as if and how they are eaten.

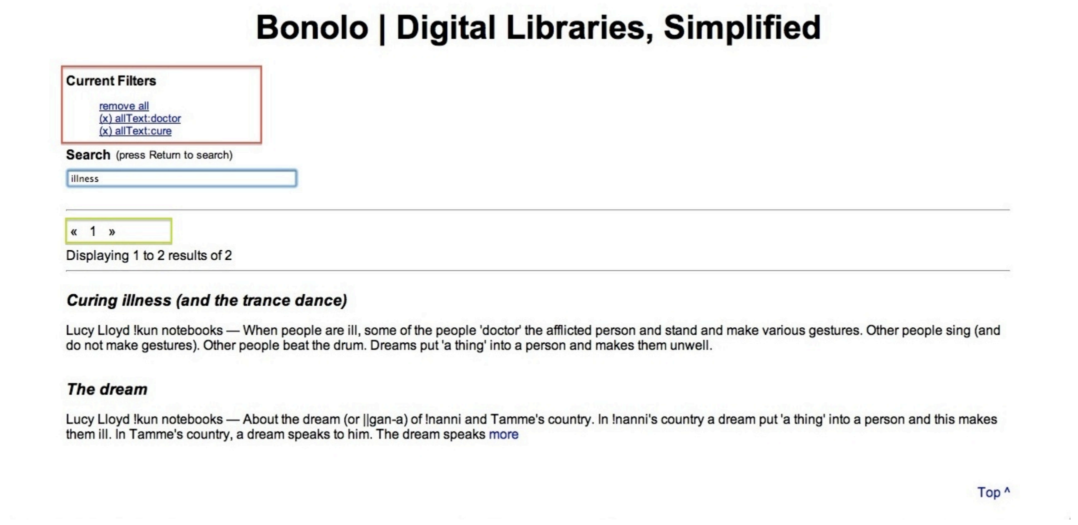
Curing illness (and the trance dance)

Lucy Lloyd !kun notebooks — When people are ill, some of the people 'doctor' the afflicted person and stand and make various gestures. Other people sing (and do not make gestures). Other people beat the drum. Dreams put 'a thing' into a person and makes them unwell.

The dream

Lucy Lloyd !kun notebooks — About the dream (or ||gan-a) of !nanni and Tamme's country. In !nanni's country a dream put 'a thing' into a person and this makes them ill. In Tamme's country, a dream speaks to him. The dream speaks more

Figure 11 Initial view of the prototype, before conducting a search


Bonolo | Digital Libraries, Simplified

Current Filters

remove all
(x) allText:doctor
(x) allText:cure

Search (press Return to search)

illness

« 1 »

Displaying 1 to 2 results of 2

Curing illness (and the trance dance)

Lucy Lloyd !kun notebooks — When people are ill, some of the people 'doctor' the afflicted person and stand and make various gestures. Other people sing (and do not make gestures). Other people beat the drum. Dreams put 'a thing' into a person and makes them unwell.

The dream

Lucy Lloyd !kun notebooks — About the dream (or ||gan-a) of !nanni and Tamme's country. In !nanni's country a dream put 'a thing' into a person and this makes them ill. In Tamme's country, a dream speaks to him. The dream speaks more

Top ^

Figure 12 View of the prototype after a search has been conducted

Evaluation

This was the shortest system evaluation because the first iteration was a feasibility prototype and no user evaluations were conducted. Instead, Dr Suleman and the project's second-reader, Dr Anne Kayem, evaluated the system. It was concluded that the system is feasible and that many additions need to be made to the system. However, these additions were expected.

3.4.2. Iteration 2: First Implementation Prototype

The goal of the second developmental iteration was to have a functioning prototype that had all of the core features present (search and browse), as well as some additional features. However, to determine the kinds of additional features to be added needed requirements gathering to be conducted.

Requirements Gathering

It is important to identify the expectations of the end-user in developing the end-user interface. Therefore, a focus group was conducted with the purpose of identifying the standard features in the system, as well as any additional features that the users would like to see present.

The focus group consisted of six people. Four of the participants were Computer Science Masters Students working in the area of Digital Libraries. The additional two members were Computer Science Honours students. The reason for these two groups of students is so that a range of suggestions could be explored from both experts and novices in the area of DRSes. The focus group was recorded using a laptop with a microphone, as well as it being documented with pen and paper.

Following this, a report was generated from the audio recording and the notes taken and the following feature list was created and categorised:

- **Searching**
  - Keyword search.
  - Ability to search by category.
  - Ability to apply search filters.
  - Possibly an Ajax live search.
- **Annotations**
  - Ability to comment on stories.
  - Make it unnecessary for a user to log in to post comments.
- **User profiles**
  - Ability to create their own collections.
  - Download archives.
  - Bookmark stories for later viewing.
  - Ability to suggest stories to their “friends”.
- **Use information relationships**
  - The system should suggest similar stories to the end-user.
- **Export the stories**
  - Ability to export story images in different formats.
  - Batch download the images.
- **Homepage**
  - Display recent comments.
  - Display recently added stories.
- **Interoperability**
  - Post stories to Facebook and Twitter.
  - The ability to subscribe to RSS updates.
- **Curator assistance**
  - Report offensive comments.
- **Administrator functions**
  - The ability to live-edit the layout of the website.
- **Navigation**
  - Make use of breadcrumbs.

This report was discussed, analysed and prioritised. As a result, the feature set was narrowed down to remove the focus on interoperability, user profiles, curator assistance and the administrator functions, as these appeared to be out of the project's scope and could be suggested for future work instead.

From the narrowed down feature set, the following functionality was to be implemented in the second iteration:

- **Home Page**
  - Let the user see what has recently been commented on.
  - Allow the user to start searching and browsing from the home page.
- **Search and Browse Page**
  - Allow the user to search by entering search terms into the search box.
  - Allow the user to click on faceted searches that have been created for them.
  - Allow the user to filter their searches.
- **Viewing Stories**
  - Allow the user to click on a story from the search and browse page to view more information about it.
  - Display the comments on the story.
  - Add comments to the story (need to be logged in to comment).
  - The ability to browse stories that are similar to the currently viewed story.
- **Log in and Registration**
  - Allow the user to register an account.

Implementation

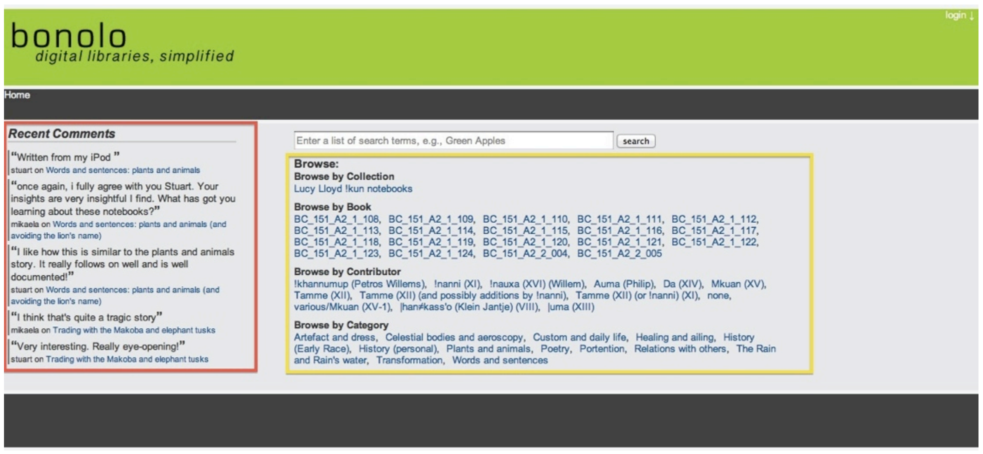
At this point in the project, a few problems were encountered when trying to expand on the prototype from iteration one. It proved challenging trying to extend the JavaScript prototype to include a subset of the aforementioned feature list. As a result, a change in technology was decided and the JavaScript solution was changed into a Java solution using JavaServer Pages, JavaBeans and Java Servlets. JavaScript and various jQuery libraries were still used, but for display purposes.

Adopting the previous JavaScript solution to a Java solution proved beneficial. It became more seamless when adding new features to the interface, as new Java classes could be created and instantiated as either a Servlet or by making use of JavaBeans within the JSPs.

By the end of the development phase of iteration two, all of the features that were specified in the design phase were successfully implemented.

Home Page

Figure 13 illustrates the home page created. As can be seen in the *red box* is the recent comments by users on various stories. The *yellow box* displays some search facets that the user can click to begin searching and browsing the stories. The user also has the ability to enter a search directly from the search box – above the *yellow box.*


bonolo
digital libraries, simplified

login ↓

Home

Recent Comments

"Written from my iPod "
stuart on Words and sentences: plants and animals

"once again, i fully agree with you Stuart. Your insights are very insightful I find. What has got you learning about these notebooks?"
mikaela on Words and sentences: plants and animals (and avoiding the lion's name)

"I like how this is similar to the plants and animals story. It really follows on well and is well documented!"
stuart on Words and sentences: plants and animals (and avoiding the lion's name)

"I think that's quite a tragic story"
mikaela on Trading with the Makoba and elephant tusks

"Very interesting. Really eye-opening!"
stuart on Trading with the Makoba and elephant tusks

Enter a list of search terms, e.g., Green Apples [search]

Browse:

Browse by Collection
Lucy Lloyd !kun notebooks

Browse by Book
BC_151_A2_1_108, BC_151_A2_1_109, BC_151_A2_1_110, BC_151_A2_1_111, BC_151_A2_1_112, BC_151_A2_1_113, BC_151_A2_1_114, BC_151_A2_1_115, BC_151_A2_1_116, BC_151_A2_1_117, BC_151_A2_1_118, BC_151_A2_1_119, BC_151_A2_1_120, BC_151_A2_1_121, BC_151_A2_1_122, BC_151_A2_1_123, BC_151_A2_1_124, BC_151_A2_2_004, BC_151_A2_2_005

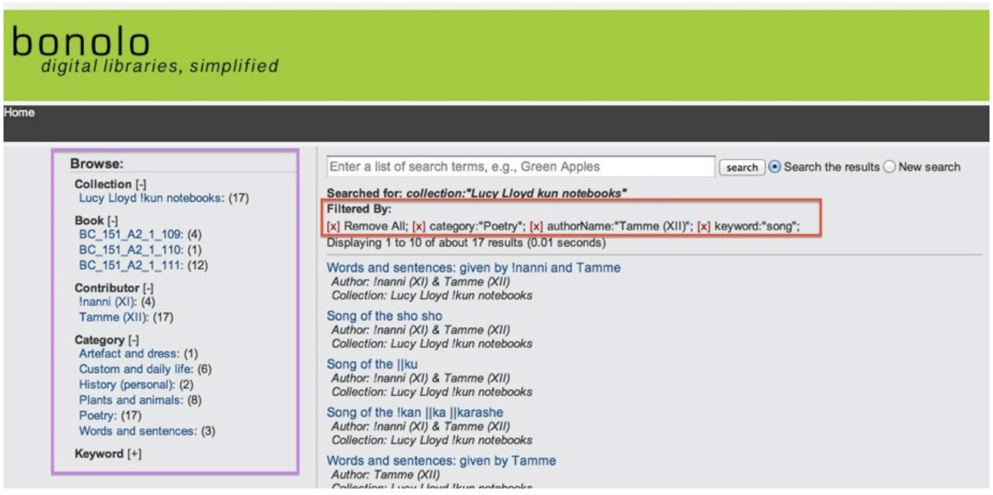
Browse by Contributor
!khannumup (Petros Willems), !nanni (XI), !nauxa (XVI) (Willem), Auma (Philip), Da (XIV), Mkuan (XV), Tamme (XII), Tamme (XII) (and possibly additions by !nanni), Tamme (XII) (or !nanni) (XI), none, various/Mkuan (XV-1), |han≠kass'o (Klein Jantje) (VIII), |uma (XIII)

Browse by Category
Artefact and dress, Celestial bodies and aeroscopy, Custom and daily life, Healing and ailing, History (Early Race), History (personal), Plants and animals, Poetry, Portention, Relations with others, The Rain and Rain's water, Transformation, Words and sentences

Figure 13 The home page created in Iteration 2

Search and Browse Page

Figure 14 displays a screenshot of the search and browse interface. From this page the user may conduct broad searches or filtered searches, by selected a radio button beside the search box. Once one of these options is selected, it affects how the search results are generated. In Figure 14, it can be seen that the initial search was for "Lucy Lloyd kun notebooks"; "Poetry", "Tamme (XII)" and "songs" have then been applied to filter the results from the initial search. These filters that have been applied are displayed in the *red box*. By clicking the "[x]" the user may remove a search filter, and thus broaden their search again.


bonolo
digital libraries, simplified

Home

Browse:

Collection [-]
Lucy Lloyd !kun notebooks: (17)

Book [-]
BC_151_A2_1_109: (4)
BC_151_A2_1_110: (1)
BC_151_A2_1_111: (12)

Contributor [-]
!nanni (XI): (4)
Tamme (XII): (17)

Category [-]
Artefact and dress: (1)
Custom and daily life: (6)
History (personal): (2)
Plants and animals: (8)
Poetry: (17)
Words and sentences: (3)

Keyword [+]

Enter a list of search terms, e.g., Green Apples [search] [✓ Search the results] New search

Searched for: *collection:"Lucy Lloyd kun notebooks"*
Filtered By:
[x] Remove All; [x] category:"Poetry"; [x] authorName:"Tamme (XII)"; [x] keyword:"song";
Displaying 1 to 10 of about 17 results (0.01 seconds)

Words and sentences: given by !nanni and Tamme
Author: !nanni (XI) & Tamme (XII)
Collection: Lucy Lloyd !kun notebooks

Song of the sho sho
Author: !nanni (XI) & Tamme (XII)
Collection: Lucy Lloyd !kun notebooks

Song of the ||ku
Author: !nanni (XI) & Tamme (XII)
Collection: Lucy Lloyd !kun notebooks

Song of the !kan ||ka ||karashe
Author: !nanni (XI) & Tamme (XII)
Collection: Lucy Lloyd !kun notebooks

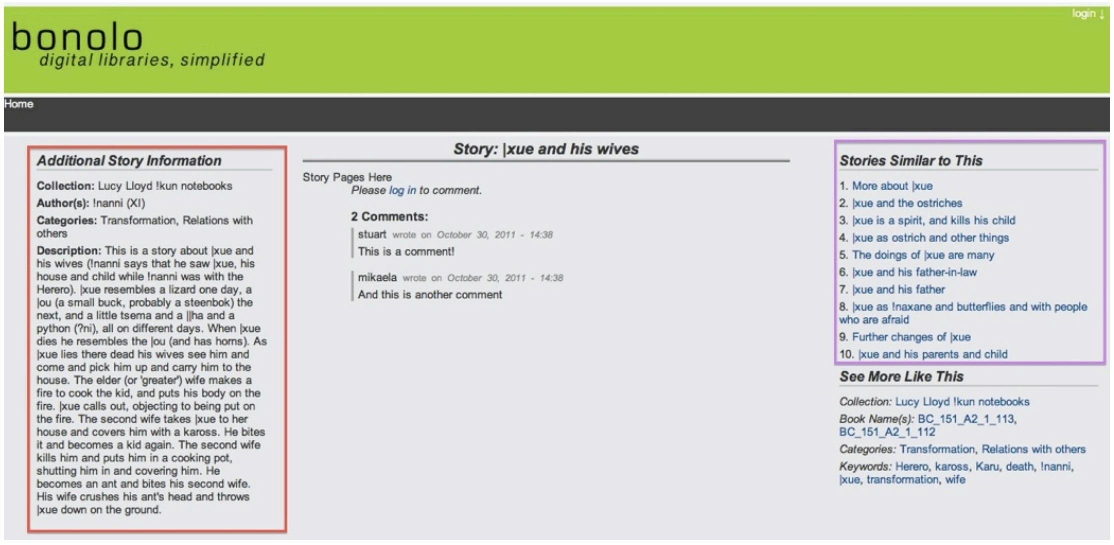
Words and sentences: given by Tamme
Author: Tamme (XII)

Figure 14 Search and browse interface in Iteration 2

On the left of the Web page (see *purple box)*, a list of faceted search options is displayed to the user, grouped by category (e.g. Collection, Book and Contributor). Next to each of the facet options is a number displayed in brackets – this illustrates the number of results that will be returned if that facet is selected. The user may also click on the "[–]" to collapse the list of facets, illustrated by "Browse By Keyword [+]". This helps to remove the clutter from the browse options.

Viewing a Story

Once a user clicks on the title of the story, displayed as a blue hyperlink on the search and browse page, they are taken to the story page, as seen in Figure 15. The "View Story" page displays the story's metadata in the *red box*. On the right of the page, a list of similar stories to the current story is suggested to the user (see *purple box*). A number of fields stored in the Solr search index are used to calculate the similarities of the stories. These stories are ordered by most similar to least similar. Beneath this, some of the story's information has been used to suggest new searches that can be made. For example, clicking on the "wife" keyword will take the user to the search page displaying the results, which have the keyword "wife" in it.



Story: |xue and his wives

Story Pages Here
Please log in to comment.

2 Comments:

stuart wrote on *October 30, 2011 - 14:38*
This is a comment!

mikaela wrote on *October 30, 2011 - 14:38*
And this is another comment

Additional Story Information

Collection: Lucy Lloyd !kun notebooks
Author(s): !nanni (XI)
Categories: Transformation, Relations with others
Description: This is a story about |xue and his wives (!nanni says that he saw |xue, his house and child while !nanni was with the Herero). |xue resembles a lizard one day, a |ou (a small buck, probably a steenbok) the next, and a little tsema and a ||ha and a python (?ni), all on different days. When |xue dies he resembles the |ou (and has horns). As |xue lies there dead his wives see him and come and pick him up and carry him to the house. The elder (or 'greater') wife makes a fire to cook the kid, and puts his body on the fire. |xue calls out, objecting to being put on the fire. The second wife takes |xue to her house and covers him with a kaross. He bites it and becomes a kid again. The second wife kills him and puts him in a cooking pot, shutting him in and covering him. He becomes an ant and bites his second wife. His wife crushes his ant's head and throws |xue down on the ground.

Stories Similar to This

1. More about |xue
2. |xue and the ostriches
3. |xue is a spirit, and kills his child
4. |xue as ostrich and other things
5. The doings of |xue are many
6. |xue and his father-in-law
7. |xue and his father
8. |xue as !naxane and butterflies and with people who are afraid
9. Further changes of |xue
10. |xue and his parents and child

See More Like This

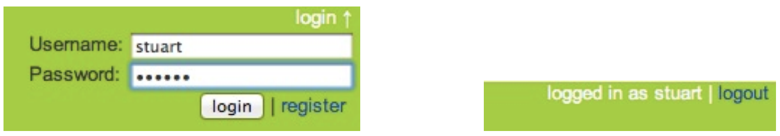
Collection: Lucy Lloyd !kun notebooks
Book Name(s): BC_151_A2_1_113, BC_151_A2_1_112
Categories: Transformation, Relations with others
Keywords: Herero, kaross, Karu, death, !nanni, |xue, transformation, wife

Figure 15 Viewing a story in Iteration 2

The end-user can log in to post comments on the story. Clicking the "log in" hyperlink causes the page to focus on the log in form on the top right of every Web page. The user can then enter their log in details, or may click "register". Clicking the latter will direct them to the registration page. Once the user logs in, they are able to post comments on the story that they are viewing. They also have the ability to delete the comments that they have posted.

Log in and Registration

Every Web page used in the end-user interface has a log in label on the top right hand corner of it. If the user clicks this text, the log in form appears with the option to log in or to register a new account. If the user logs in successfully, their nickname is displayed with the option to logout next to it (see Figure 16).

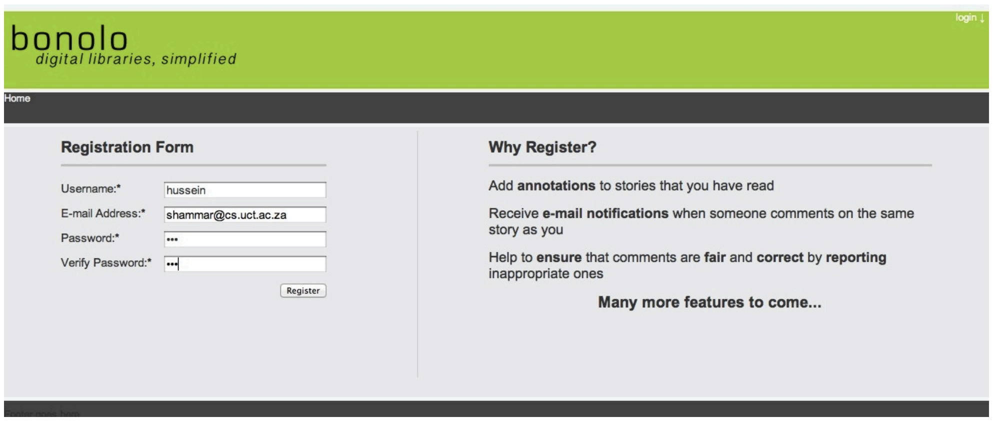


Username: stuart
Password: ••••••

logged in as stuart | logout

Figure 16 The log in information at the top of every page

If the user clicks the registration hyperlink, they are taken to the registration page (see Figure 17) where they can then register a new account. In this iteration there is no client-side form validation; the validation is done on the server and the user is informed whether it was a successful registration or not. At this stage, the email address validation process is not present either. Therefore, once the user completes the registration, they are able to log in and post comments.



Registration Form

Username:*	hussein
E-mail Address:*	shammar@cs.uct.ac.za
Password:*	•••
Verify Password:*	•••

Why Register?

Add **annotations** to stories that you have read

Receive **e-mail notifications** when someone comments on the same story as you

Help to **ensure** that comments are **fair** and **correct** by **reporting** inappropriate ones

Many more features to come...

Figure 17 Registration page in Iteration 2

System Configuration

At this point in the design process, much of the programming had been hard-coded. This was not ideal in a system that was meant to be general. However, the core functionality of the system was of a higher priority, as they would provide answers to the research questions. Therefore, it was only considered for the final iteration.

Evaluation

To further aid in the implementation of the additional features, end-user testing was conducted after the second iteration to gather feedback for changes for the final iteration. A total of six user evaluations were conducted for this iteration. The users that performed the test aged between 19 and 65 years old. Four of the users were university students, each of which belonging to a different faculty. The additional two users were employed outside of the university. Despite there only being six user tests, there was a range of feedback was received due to their various backgrounds and expertise. Furthermore, slight trends were noticed in their feedback.

The evaluation that was carried out was based on the constructive interaction method described in [33]. This form of evaluation involves the users thinking aloud and explaining what they are doing to the observer. The evaluation commenced with an overview of the project being given to the user to read over. This explained what the evaluation entailed and what the project was about. From here, the users were informed that their information would be kept anonymous and their verbal consent was given. The evaluation consisted of a set of 12 tasks that the user would complete. The tasks would be observed and if any unexpected outcomes occurred, they were discussed in the post-evaluation interview. In addition to this, the user was asked if they had any other difficulties with the system and whether they had any further suggestions to improve the experience of using the system. From the six user tests conducted, the following results were obtained:

- **Home page**
    - 1 of the 6 users found it difficult to decide which comment was the most recent. It was suggested that a numbering system, or the date and time be displayed to assist the user.
- **Search and Browse page**
    - 1 user suggested that the search facets be collapsed on default.

    - 4 of the 6 users found that the search filters were not catching their attention. They suggested that the search filters be moved above the "Browse:" heading on the left of the page.
    - These 4 users suggested that the applied filters be listed vertically instead of horizontally in an indented list, similar to the browse facets.
    - 1 user commented that the search filters be neatened up and made more transparent to the user. Therefore, the semicolon, quotation marks and the category being searched should be removed from the output. This same user also suggested that the "Remove All" label be made its own hyperlink.
    - 5 out of the 6 users did not notice the radio buttons that let them choose what kind of search to make. Two of the users suggested that they be moved beneath the search box and that the filter option should be changed to "search within results".
    - 2 users suggested that the search should be set to "search within results" by default, as this will keep the interface consistent.
- **Viewing Stories**
    - There was some confusion about the "Similar Stories" and "See more like this" headings. 2 users suggested that the latter heading be changed to "Browse similar topics" to better highlight the difference.
    - All of the users expected there to be images for them to view.
    - 1 user asked whether email notifications would be received if someone replies to a comment on the thread.
    - Another user suggested that the headings of the information be kept consistent and made larger.
- **Registration and Log in**
    - 2 of the users found that the "log in" label at the top of the screen was too small and unnoticeable. They suggested that it be made larger and bolder.
- **General**
    - 1 user suggested that the site navigation is unclear. It was further suggested that breadcrumbs and a static navigation be added to the website to improve this.
    - This same user also complained that the headings were not large enough or noticeable enough.

The users appeared to be confused by the large amount of information displayed to them. The way that some of the information was laid out and the headings used appeared to create confusion and hindered the user experience. As a result, these changes were then carried forward as requirements for implementation of the final system.

3.4.3. Iteration 3: Final System Implementation

The final system implementation was be expected to have the complete feature set discussed and agreed upon at the beginning of the second iteration, along with fully functioning search and browse features. This final system would then be used in conducting a final user experience evaluation and performance evaluation.

Requirements Gathering

The suggestions received from the evaluations conducted in iteration two were reviewed and prioritised. On review, other considerations and suggestions were made and would have to be completed for the final implementation:

- **Home page**
    - The users would need to be able to view an entire book from cover-to-cover.
- **Search and Browse**
    - The results that are returned should contain a small image thumbnail showing the first page of the story.
- **Viewing a Story**
    - The story images need to be displayed.
    - Users should be able to bookmark whichever story they are viewing so that they can send it to a friend if they wish.
    - Thumbnails of the images in the collection should be created so as to minimise the amount of information downloaded by the client.
- **Viewing a Book**
    - The users should be able to view a book from cover-to-cover.
    - The users should be able to browse through the books in the collection.
- **Log in and Registration**
    - Change "username" to "nickname".
    - Allow the user to log in with their email address instead of their nickname.
    - Remove password length restrictions on registration.
- **Annotations**
    - The users should receive notifications when someone comments in the same thread.
- **Export Images**
    - Allow the user to download the images of the book or story as a compressed zip archive.
- **Generality**
    - The system needs to be general so that it can be applied to any digital collection.

Final Implementation

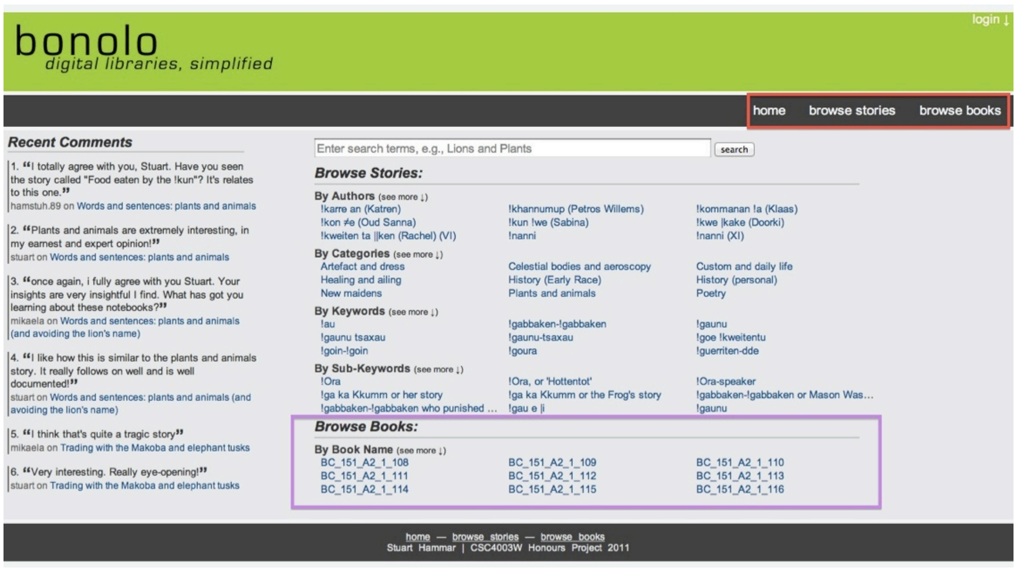
By the final iteration, all of the changes identified in the evaluation of the second iteration were prioritised and rectified. In addition, the other suggestions and clarifications were also added to Bonolo. There were a number of minor adjustments made to the CSS throughout this iteration. The final system features are described in this subsection.

Home Page

The home page of the final system received a few updates since the second iteration. Figure 18 demonstrates that there is now a static navigation menu on the Web page (see *red box*). In addition to this, there is faceted browsing so that users can explore the books in the collection, from cover-to-cover (see *purple box*). A "see more" label so as to remove the clutter and information overload when first viewing the home page accompanies each faceted browse heading. In addition, the facets are organised more neatly. The final notable change present on the home page is that the recent comments now have a numbering system associated with them – one is most recent, six is least recent. This is present as end-users were unsure about the ordering of the recent comments.

Browse Stories Page

The browse stories page is now directly accessible from the home page without needing to enter a search. When the user does this, every record in the search index is returned to the user.


login ↓

bonolo
digital libraries, simplified

home browse stories browse books

Recent Comments

1. “I totally agree with you, Stuart. Have you seen the story called "Food eaten by the !kun"? It's relates to this one.”
hamstuh.89 on Words and sentences: plants and animals

2. “Plants and animals are extremely interesting, in my earnest and expert opinion!”
stuart on Words and sentences: plants and animals

3. “once again, i fully agree with you Stuart. Your insights are very insightful I find. What has got you learning about these notebooks?”
mikaela on Words and sentences: plants and animals (and avoiding the lion's name)

4. “I like how this is similar to the plants and animals story. It really follows on well and is well documented!”
stuart on Words and sentences: plants and animals (and avoiding the lion's name)

5. “I think that's quite a tragic story”
mikaela on Trading with the Makoba and elephant tusks

6. “Very interesting. Really eye-opening!”
stuart on Trading with the Makoba and elephant tusks

Enter search terms, e.g., Lions and Plants [search]

Browse Stories:

By Authors (see more ↓)

!karre an (Katren)	!khannumup (Petros Willems)	!kommanan !a (Klaas)
!kon ≠e (Oud Sanna)	!kun !we (Sabina)	!kwe |kake (Doorki)
!kweiten ta ||ken (Rachel) (VI)	!nanni	!nanni (XI)

By Categories (see more ↓)

Artefact and dress	Celestial bodies and aeroscopy	Custom and daily life
Healing and ailing	History (Early Race)	History (personal)
New maidens	Plants and animals	Poetry

By Keywords (see more ↓)

!au	!gabbaken-!gabbaken	!gaunu
!gaunu tsaxau	!gaunu-tsaxau	!goe !kweitentu
!goin-!goin	!goura	!guerriten-dde

By Sub-Keywords (see more ↓)

!Ora	!Ora, or 'Hottentot'	!Ora-speaker
!ga ka Kkumm or her story	!ga ka Kkumm or the Frog's story	!gabbaken-!gabbaken or Mason Was…
!gabbaken-!gabbaken who punished …	!gau e |i	!gaunu

Browse Books:

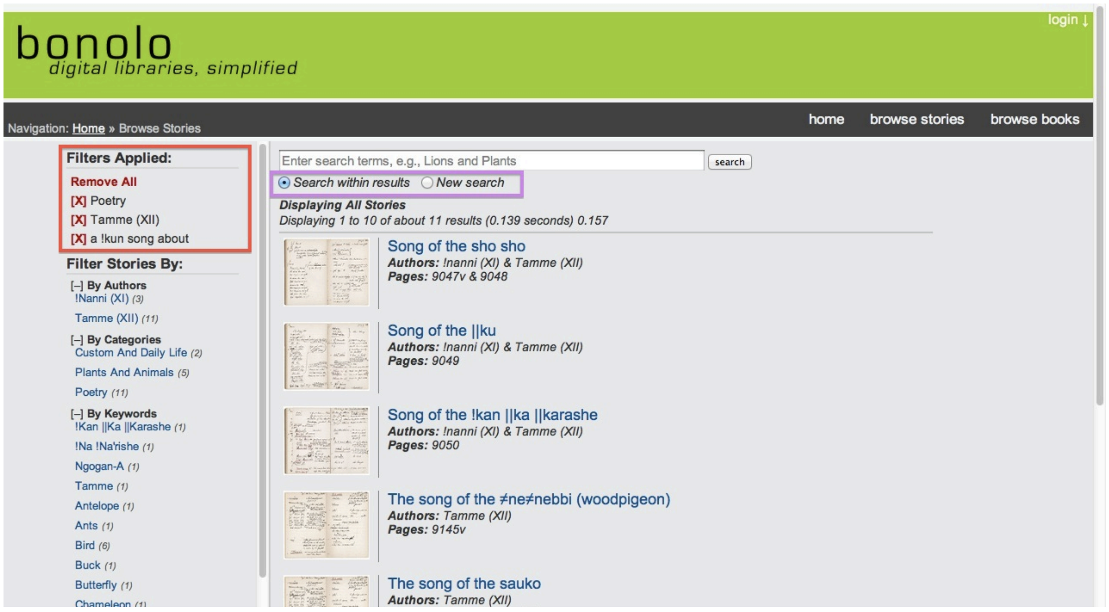
By Book Name (see more ↓)

BC_151_A2_1_108	BC_151_A2_1_109	BC_151_A2_1_110
BC_151_A2_1_111	BC_151_A2_1_112	BC_151_A2_1_113
BC_151_A2_1_114	BC_151_A2_1_115	BC_151_A2_1_116

home — browse_stories — browse_books
Stuart Hammar | CSC4003W Honours Project 2011

Figure 18 Home page after Iteration 3

The major notable changes that were made to this page are due to the suggestions received from the user evaluation in the second iteration (see Figure 19). These changes include the search options being renamed and relocated to beneath the search bar (see *purple box*). The goal here is to make these search options more visible to the end-user. The search filters that were previously displayed horizontally beneath the search bar have now been relocated to the left of the page and are displayed vertically (see *red box*). Moreover, the text that was previously displayed (e.g. category:“Poetry”) has been formatted so as to make it more transparent and less confusing to the end-user (e.g. it only says “Poetry”).


login ↓

bonolo
digital libraries, simplified

Navigation: Home » Browse Stories

home browse stories browse books

Filters Applied:
Remove All
[X] Poetry
[X] Tamme (XII)
[X] a !kun song about

Filter Stories By:

[–] **By Authors**
!Nanni (XI) *(3)*
Tamme (XII) *(11)*

[–] **By Categories**
Custom And Daily Life *(2)*
Plants And Animals *(5)*
Poetry *(11)*

[–] **By Keywords**
!Kan ||Ka ||Karashe *(1)*
!Na !Na'rishe *(1)*
Ngogan-A *(1)*
Tamme *(1)*
Antelope *(1)*
Ants *(1)*
Bird *(6)*
Buck *(1)*
Butterfly *(1)*
Chameleon *(1)*

Enter search terms, e.g., Lions and Plants [search]
[✓ Search within results] New search

Displaying All Stories
Displaying 1 to 10 of about 11 results (0.139 seconds) 0.157

Song of the sho sho
Authors: *!nanni (XI) & Tamme (XII)*
Pages: *9047v & 9048*

Song of the ||ku
Authors: *!nanni (XI) & Tamme (XII)*
Pages: *9049*

Song of the !kan ||ka ||karashe
Authors: *!nanni (XI) & Tamme (XII)*
Pages: *9050*

The song of the ≠ne≠nebbi (woodpigeon)
Authors: *Tamme (XII)*
Pages: *9145v*

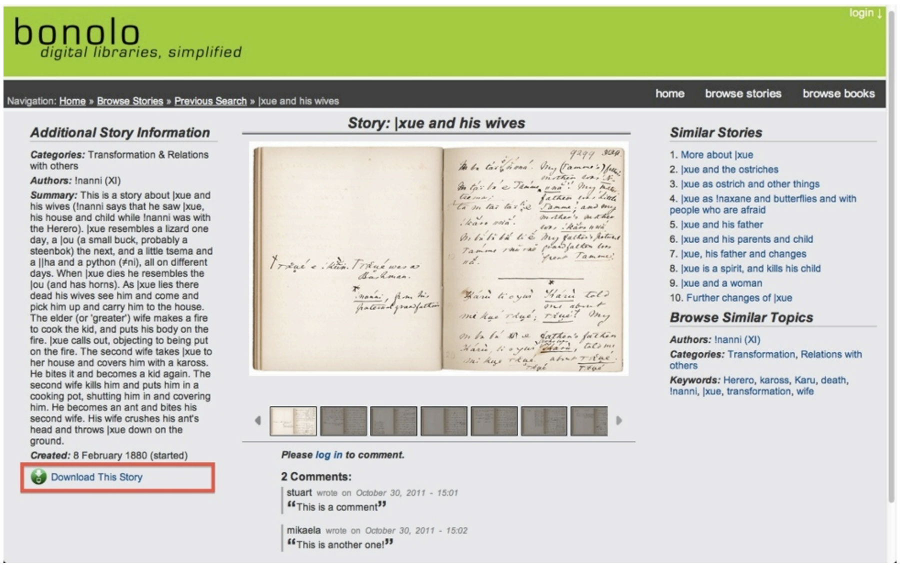
The song of the sauko
Authors: *Tamme (XII)*

Figure 19 Searching and browsing stories in Iteration 3

When the search results are loaded, a thumbnail of the first page of that story is displayed to the end-user alongside the metadata. Also, the page number that the user is on is underlined. It was decided that the search facets remain expanded as this removes the number of clicks needed by the user to choose a facet. In addition, when observing sites that use faceted searching, their facets are expanded by default (e.g. Amazon.com and Take2.co.za).

Viewing a Story

Viewing the pages of a story, in the correct order, was an important factor for this interface, as these were the preserved content. As a result, the story pages are located in the centre of the page (see Figure 20). The images are displayed in a JavaScript gallery that is familiar to the end-user. The image gallery was fully implemented, with thumbnails and image preloading in this iteration. Despite using JavaScript to display the images, they are still "bookmarkable". The heading "See More Like This" was changed to "Browse Similar Topics", as suggested by the users. Also, the ability to export the story's pages was added to the interface (see *red box*). These were the only updates seen to the "Viewing Story" page.



Navigation: Home » Browse Stories » Previous Search » |xue and his wives

Additional Story Information

Categories: Transformation & Relations with others

Authors: !nanni (XI)

Summary: This is a story about |xue and his wives (!nanni says that he saw |xue, his house and child while !nanni was with the Herero). |xue resembles a lizard one day, a |ou (a small buck, probably a steenbok) the next, and a little tsema and a ||ha and a python (≠ni), all on different days. When |xue dies he resembles the |ou (and has horns). As |xue lies there dead his wives see him and come and pick him up and carry him to the house. The elder (or 'greater') wife makes a fire to cook the kid, and puts his body on the fire. |xue calls out, objecting to being put on the fire. The second wife takes |xue to her house and covers him with a kaross. He bites it and becomes a kid again. The second wife kills him and puts him in a cooking pot, shutting him in and covering him. He becomes an ant and bites his second wife. His wife crushes his ant's head and throws |xue down on the ground.

Created: 8 February 1880 (started)

Download This Story

Story: |xue and his wives

Please log in to comment.

2 Comments:

stuart wrote on October 30, 2011 - 15:01
"This is a comment"

mikaela wrote on October 30, 2011 - 15:02
"This is another one!"

Similar Stories

1. More about |xue
2. |xue and the ostriches
3. |xue as ostrich and other things
4. |xue as !naxane and butterflies and with people who are afraid
5. |xue and his father
6. |xue and his parents and child
7. |xue, his father and changes
8. |xue is a spirit, and kills his child
9. |xue and a woman
10. Further changes of |xue

Browse Similar Topics

Authors: !nanni (XI)

Categories: Transformation, Relations with others

Keywords: Herero, kaross, Karu, death, !nanni, |xue, transformation, wife

Figure 20 Viewing a story in the final iteration

Viewing a Book

Viewing a book from cover-to-cover is a completely new feature in this iteration (see Figure 21). From the home page, or using the static navigation, the user may browse all of the books in the collection. The user may also download the book's pages if they wish (see *red box*). The functionality surrounding viewing the book's pages is identical to that on the "Viewing Story" page.

Log in and Registration

The log in form was updated so that users log in with their registered email address as opposed to their nicknames. The reason behind this is that users tend to remember their email addresses better than their usernames. In addition to this, if a user logs in with an invalidated account or a non-existent account, they receive a notification to tell them.

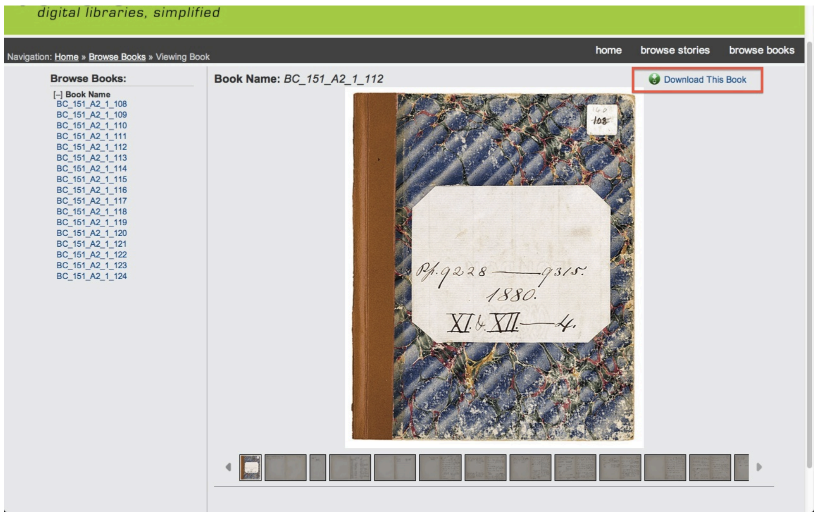
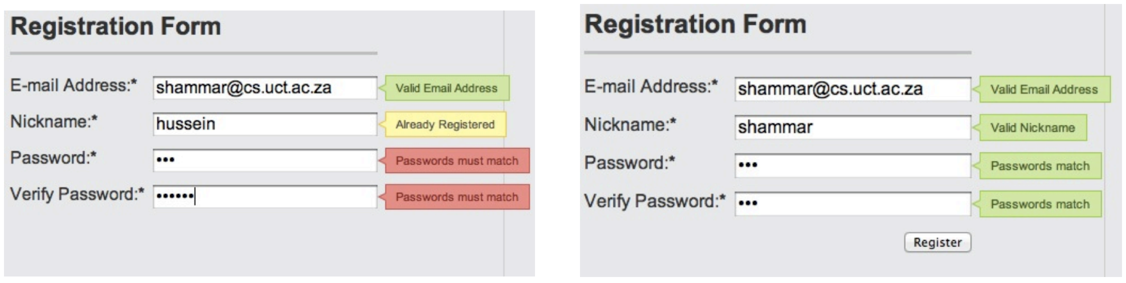


Figure 21 Browsing books from cover-to-cover in the final iteration

On the registration page, client-side validation was implemented (see Figure 22). This provides the user with live feedback while completing the registration form. The password length restriction (e.g. a password had to be more than five characters long) was removed, as this would only lengthen and complicate the registration process. Once the user clicks "Register", an overlay pop up informs them that the registration was successful or not. It also tells the user to validate their email account. The user can only log in with their account once they have followed the verification link that was emailed to them post-registration.



Registration Form

E-mail Address:*	shammar@cs.uct.ac.za	Valid Email Address
Nickname:*	hussein	Already Registered
Password:*	•••	Passwords must match
Verify Password:*	••••••	Passwords must match

Registration Form

E-mail Address:*	shammar@cs.uct.ac.za	Valid Email Address
Nickname:*	shammar	Valid Nickname
Password:*	•••	Passwords match
Verify Password:*	•••	Passwords match

Figure 22 Live validation of the registration form in the final iteration

Annotations and Exporting Images

Over and above the user being able to comment on stories, in this iteration they will receive email updates whenever someone comments on the same thread as them. This was felt to be beneficial so that users can follow the comments on the stories as they happen.

In this iteration, users are able to download the all of the pages of the story or book that they are viewing (see *red box* in Figure 20 and Figure 21). The pages are exported to the user in a compressed zip file.

System Configuration

The hard coding within the system had been removed by the final iteration. This made the system more configurable and customisable. However, due to time constraints and scope limitations, there is no end-user interface for the user to dynamically update the information contained in the configuration files. Currently, the user can only configure the system if they have knowledge of XML, otherwise they can run the command-line configuration application that was written. The command-line application generates the custom files necessary. The user has to manually place these files into the right directories within the Web application.

Final Evaluation

The final evaluation of the system was conducted in two parts. The first part was by conducting user experience testing and the second part is by conducting a system performance evaluation. These evaluation procedures, and their respective results, would decide the success of the project. More specifically, the results from these evaluations would decide whether the research questions have, in fact, been answered. The final evaluations and the results are discussed in *Section 4*.

3.5. Portability and Maintainability

The Bonolo end-user interface was written primarily in Java and makes use of various Java libraries to provide the necessary functionality to the interface. All of the software used in creating the Bonolo Web application is open-source and is written in Java. The reason for this is so that the software is free, as well as being platform independent. This aids in the portability of the Web application so that it can be compiled and run on any operating system.

The end-user interface is divided into various Web interfaces. Furthermore, the server-side applications are stored as JavaServlets or are written as classes and implemented as JavaBeans within the JSPs. As a result, the Web application is modular. This puts a strong emphasis on the componentisation of the Web application. Moreover, it makes the Web application extensible, as other developers can create their own modules for the interface. This has the potential to lead to a number of new and robust modules being added to the interface.

3.6. Issues

Due to the amount of human input received by the server, this can lead to various issues that were initially unforeseen. Searching with binary operators (e.g. + or −) resulted in the search not returning an “AND” search as the user would have expected. In addition, some arbitrary search characters (e.g. ?, !, : and *) would result in all searches being returned by the search engine.

Another issue encountered was that the application to configure the Solr indices and the end-user interface was not user friendly or intuitive to use. As a result, the user would need to have knowledge of XML to customise the end-user interface.

When working with the Solr search engine, it was uncovered that new metadata files cannot simply be added to the indices by using a delta-import [34]. Instead, all of the metadata has to be re-indexed. This is an issue that is known to the Apache Solr developers.

The ability to view the story and book images relies heavily on the fact that the paths and image names are stored within the metadata. This is due to the fact that the Web application does not traverse the file structure and relies on the search engine to locate the images.

The project was always tested within Google Chrome and Safari. Therefore, the layout and functionality were all suited to these two browsers. However, some layout issues arose when testing in other browsers, such as Mozilla Firefox and Internet Explorer. In addition, when a user logged out, the other browsers would only update it when the page reloaded.

3.7. Discussion

This section of the report has explored the iterative UCD design and implementation process of the Bonolo end-user interface. The functionality of the interface, as well as the design decisions, has been explained so that these concepts can be adapted for future solutions. The iterative design process describes how the system evolved and how the evaluations from the previous iteration had an influence on the next iteration. Finally, the flaws in the design and implementation were explored.

4. System Evaluation

The system evaluation discussed in this section relates only to that of the final design iteration. The other evaluations that were conducted can be found in *Section 3.4*. Before the evaluation of the final implementation, the design and system functionality were finalised.

The success of the Bonolo project relies on three key factors:

1. The end-user's experience when using the system is not hindered due to the XML-centric repository.
2. The efficiency of the system is acceptable for large amounts of data in the repository, i.e., the hierarchical file store does not have a negative impact on system performance.
3. The tools and features offered by the system are comparable to those offered by other digital repository systems.

To determine whether the system answers the three research questions and, in turn, meets the success factors, two sets of evaluations were carried out. Firstly, user experience evaluations were conducted to determine whether the first and third success factors were met. A performance evaluation was carried out on the system to determine how well it meets the second success factor. These two evaluation techniques are discussed in more detail in this section.

4.1. User Experience Evaluation

The first set of evaluations conducted on the system was a user experience (UX) evaluation. A UX evaluation is different from a typical HCI usability evaluation. The International Standards Organisation defines UX as being a measure of satisfaction in usability [35]. Therefore, it helps to ascertain how the user feels when they are using the system. Whereas, a usability evaluation attempts to improve human performance through measuring how effective, usable and efficient a system is.

UX is subjective because each user may feel different emotions (e.g. confusion, satisfaction or frustration) when using the Web application. The results of the UX evaluation would point out any areas in the interface where users felt uncomfortable, frustrated, dissatisfied or confused. Moreover, it will also illustrate whether the output and interactions with the system were what the user expected and whether the system responded to user interaction within a reasonable amount of time.

All user testing was conducted on the following computer with the following configuration:

- **Apple Macbook – always hosted the Web application**
  - 2.2 GHz Intel Core 2 Duo
  - 4 GB RAM
  - 120 GB Hard Drive
  - Mac OS X Lion 10.7.2
  - Apache Tomcat 7.0.19
  - Solr Version 3.3.0

        - 1 183 digital objects were in the index
    - Java Standard Edition Version 6 Update 26
    - Google Chrome 15.0.874.106

4.1.1. Users

17 users completed the final UX evaluation of Bonolo. Each of the users was required to have basic Web experience. The types of users that completed the test are as follows:

- 6 Computer Science students
- 8 students from various faculties:
    - 3 Health Science students
    - 3 Business Science students
    - 1 Bachelor of Arts students
    - 1 Engineering student
- 2 users external to UCT
- 1 user with professional experience in UX

The ages of the users ranged between 18 and 30.

There were 11 males and 6 females that completed the UX evaluation.

4.1.2. Aim

The aim of the UX evaluations is to answer two research questions:

1. Will using a simple file hierarchy as a file store affect the overall end-user experience when using the interface?
2. Is it possible to create an end-user interface, built on a simple file store that is comparable to other DRSes?

4.1.3. Methodology

The overall end-user experience when using the Web application was evaluated by conducting a practical, hands-on test. Each user test was conducted individually and was supervised by the researcher. Prior to the test beginning, the user was given an informed consent form to read and sign. The form explained what the project is about, what kind of information is stored in the repository, the evaluation procedure and how the evaluation results were to be used. The user was reminded that they were not being tested, and that the system was being tested. Also, they were reassured that if they did get confused, lost or discouraged they could ask the researcher for a clarification. This provided the user with a sense of comfort and relaxation before beginning the test.

The user was provided with a test consisting of 5 sets of tasks. Following each set of tasks was a qualitative evaluation questionnaire relating to the tasks that were just completed. The test concluded with a number of general questions about the interface. The user was observed as he/she completed the tasks specified so that the researcher could take note of any obscure or unexpected actions. Any unexpected actions were later discussed with the user to gain insight as to why they occurred. The informed consent form and evaluation form can be found in *Appendix A* and *Appendix B* respectively.

The Web application was hosted on the Local Area Network. This removes the volatility of a broadband Internet connection when the users performed the UX evaluation. Furthermore, two pilot evaluations were conducted to ensure that the evaluation would run smoothly – naturally, the results from these are not included in *Section 4.1.4*.

4.1.4. Results

The results obtained from the user evaluation are grouped by and described in their respective sections. The results recorded in these sections are all measured on a five-point Likert scale [33]. Due to the UX nature of the test, the words on either end of the scale were not identical for every question. However, the numbering scale itself was always consistent – 1 and 5 represent a bad and good experience respectively. Some questions also asked for feedback if they evaluated the question with a 1.

Search and Browse

In this section, the user was given seven tasks to complete. This was then followed by 11 evaluation questions.

Figure 23 illustrates the users' responses to the 11 evaluation questions. Question 1 shows that 13 of the users (76%) found conducting a search to be quite or very natural and intuitive. Only 24% of users found the process to be quite or very unnatural and counter-intuitive. The users that found it less natural stated that the interface initially felt foreign and unfamiliar, but then instinct took over and helped them figure out what to do. The user that felt it was completely unnatural commented that it would have been beneficial to be given a few minutes to use the interface before beginning the test.

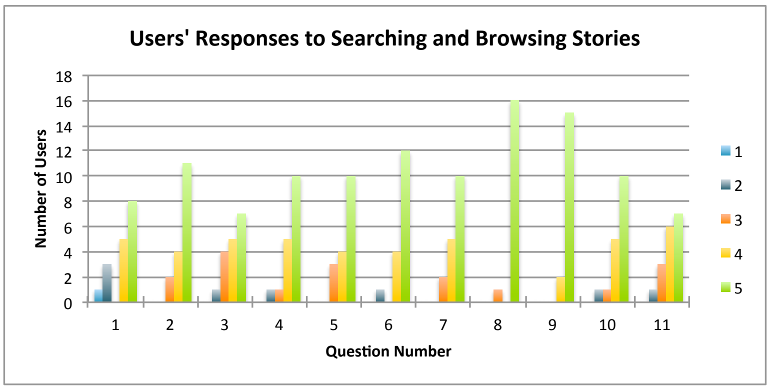


Figure 23 Users' responses to searching and browsing stories

Question 3 displays that there was some confusion surrounding the ease of filtering a search, however, 65% of users found filtering to be beneficial (Question 2). Those users who found filtering confusing suggested that they be given more time to interact with the system prior to using it, echoing the comment made for Question 1. Other users misunderstood the task, which resulted in initial confusion, but once they understood what was expected, they found filtering to be straightforward. Other users said that originally they did not notice the ability to filter, as it was seen as "information overload", but after that it appeared intuitive and easy to use.

Questions 6, 7 and 8 related to displaying results, search expectations and the response of the system. 94% of users found that the results were displayed exactly as or almost as expected. Only one user found that the results were displayed quite unexpectedly. A further 10 users found that the system conducted searching and browsing exactly as they would have expected, with no users saying that it was unexpected. 16 users (94%) felt that the system responded to their search and browse requests within a reasonable amount of time.

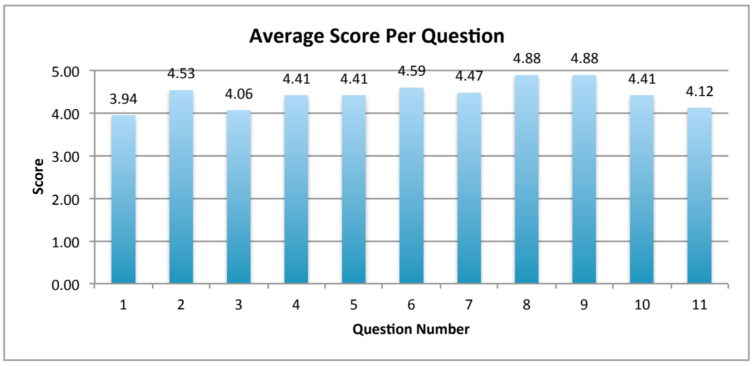


Figure 24 Average score per question when searching and browsing stories

The final 3 questions summarise how the user felt when they interacted with the system. 100% of users felt that the system responded either exactly or quite as they would have expected; no users felt that the responses were unexpected (Question 9). On average the users were fairly comfortable and satisfied with the search and browse interface (Question 10), scoring an average of 4.41 out of 5 (see Figure 24). Question 11 asked whether users were frustrated with the system, the average score was 4.12. Only one user found the interface fairly frustrating. Some of the things that frustrated the users were the inability to locate the filtering functions and the initial confusion from all the information on the pages. Once the user got used to the system, they commented that it was easy to use.

Viewing a Story

The user was given seven tasks to complete, which were then followed by 10 evaluation questions.

76% of all users found that viewing more story information from the search page was fairly straightforward (Question 1, see Figure 25). Only one user found this to be slightly complex. Identifying the story confused some users, purely because of the strange Khoi-Khoi punctuation.

100% of the users found that the system responded fast and within a reasonable amount of time when loading this Web page (Question 2). Question 3 and 4 addressed the way that images were displayed. 14 of the users (82%) found that navigating stories was very straightforward and that viewing the story images worked as was expected. Only two users felt that the latter did not perform entirely as expected. Comments from the users included that they did not expect the full-size story to open in a new tab; instead they thought it would pop up in an overlay. Another common comment was that one should be able to navigate the stories while viewing the pages at full size.

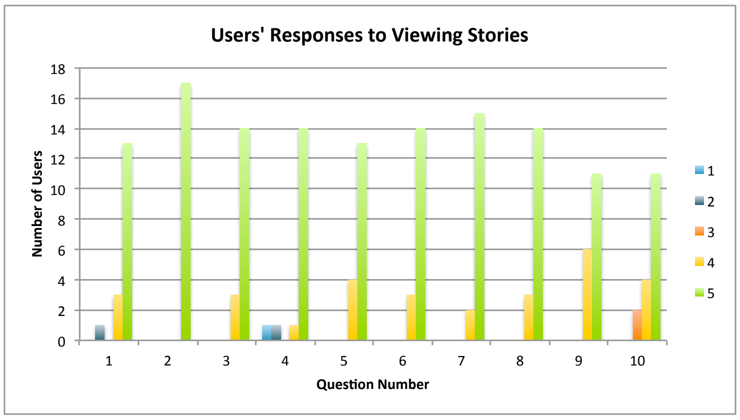


Figure 25 Users' responses to viewing stories

76% of users found the list of suggested stories to be beneficial (Question 5); another 82% of users found the ability to browse similar topics to be effective too. No users found either of these features to be non-beneficial. Generally, users found the ability to download the story pages in a zipped archive to be beneficial – Question 7 had an average score of 4.88, which illustrates this (see Figure 26). However, one user felt that the download link was not noticeable enough and should have been bigger or relocated.

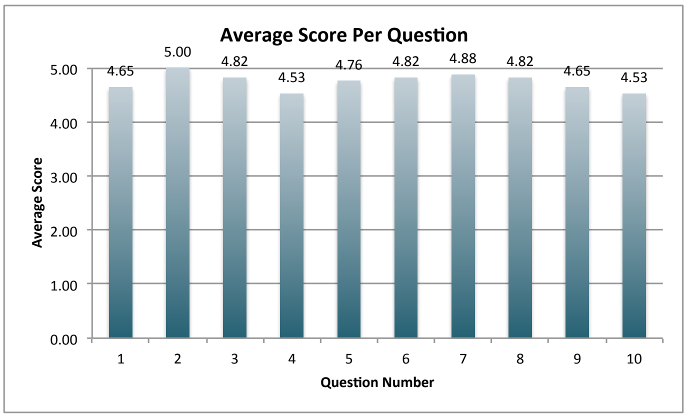


Figure 26 Average score per question for viewing stories

The final three questions summed up how the users felt when using the interface. Question 8 illustrated that on average the users felt that the features and actions performed on the story page were as expected, with an average score of 4.82 out of 5. Question 9 and 10 display that the users felt fairly comfortable and were not frustrated when using the interface – this is justified by 65% of users feeling this way.

Annotations and Registration

Again, the user had seven tasks to complete. An evaluation of 10 questions came after this.

Question 1 displayed that on average, users felt that identifying the most recently commented on story was straightforward; the average score was 4.59 out of 5 (see Figure 28). However, one user was confused because they were not sure which comment was the most recent, despite the numbering system. This user suggested that the comments be dated on the home page.

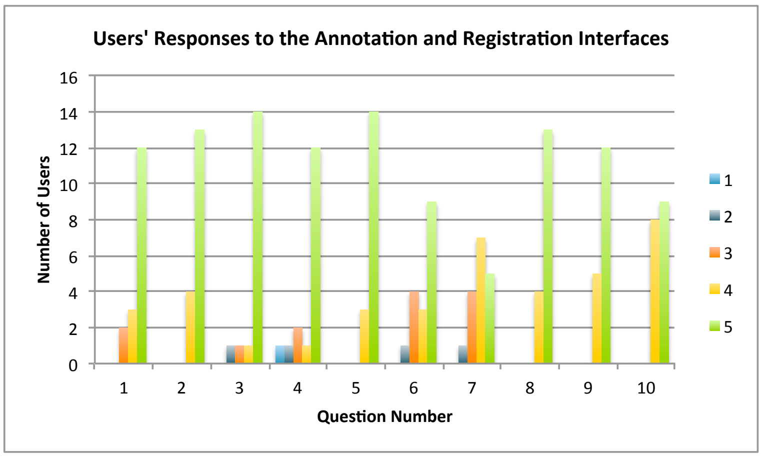


Figure 27 Users' responses to the annotating and registering

Question 2 and 3 highlighted that 13 users (76%) found the registration process very intuitive; another 14 users found the process to be short and simple (see Figure 27). One user felt that the process was long and tedious. A general comment from the users was that there was no direct way to get back to the story they wanted to comment on. Instead, they had to return to the home page and find it again. This caused some irritation.

Questions 4 and 5 showed that on average posting comments was fairly intuitive (an average score of 4.29) and that the full annotation functionality performed as the user expected (an average of 4.82). The users that had problems with posting a comment noted that it took a fair amount of time to post the comment, which led to them double- or triple-posting the comment. Some of the users decided to comment on the ordering of the comments, despite this not being a question. The overall feeling was ambivalent as some felt that the most recent comment should be at the top of the list, while other users felt the ordering should remain as it was.

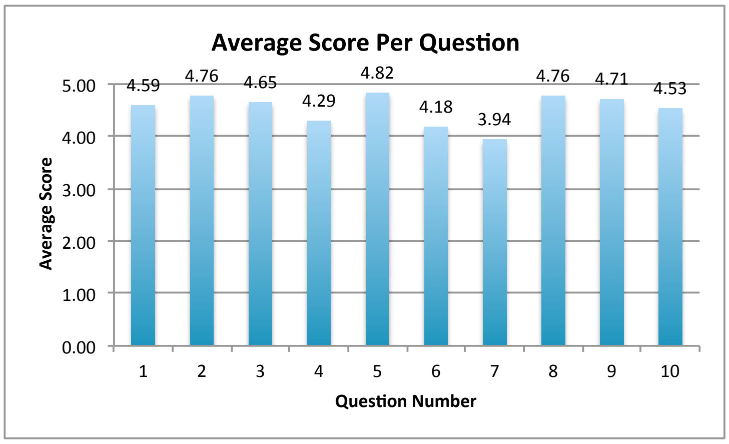


Figure 28 Average score per question for annotations and registration

Question 6 received mixed responses from the end-users. 29% of users felt that the system either responded in an average or below average amount of time. Users had mixed reactions on whether comment notifications were beneficial. The average score was 3.94, showing that users were leaning toward this feature being quite beneficial, but not very beneficial (Question 7). This result is echoed by 12 of the users (71%) feeling it was either quite beneficial, indifferent about it, or that it is fairly non-beneficial. Furthermore, users justified their response by saying that email notifications, as compared to Facebook notifications, tend to become annoying. One user summed it up by saying that at one level it is beneficial, but at another level, one does not want to be bothered by these emails.

All in all, questions 8, 9 and 10 displayed that the users were satisfied and comfortable with the functionality of the interface. No users feeling frustrated by the annotation or registration interfaces echoed this.

Browsing the Books

Five tasks were completed by the users, after which six evaluation questions ensued.

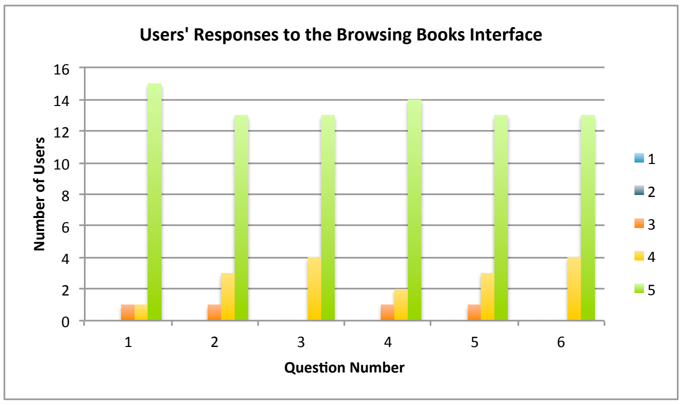


Figure 29 Users' responses to browsing books

On the whole, Figure 29 illustrates that the end-users found browsing books to be beneficial, with only one user feeling indifferent about this feature (Question 1). Regarding Question 2, 76% of users felt that the interface was intuitive. However, some users commented that the link to download the book was not where it was expected, as it was placed in a different location to the download link on the story page. Another suggestion was, again, for a full-screen browse and navigation option.

In this set of tasks, only four responses were indifferent. Thus illustrating that in general, the end-users found this interface and its features to be beneficial and made them feel comfortable and satisfied when using the interface. Furthermore, all of the users felt that the interface responded to their requests within a reasonable amount of time (Question 3).

Navigation

The user was required to complete seven tasks, before conducting an evaluation of seven questions.

Initially Question 1 demonstrated that users felt that the site was fairly straightforward (82% of users scored 4 or greater) to navigate with only two users feeling unsure about this (see

Figure 30). 11 of the users found the breadcrumb[4] navigation to be very effective, while one user found it to be slightly ineffective (Question 2). This user found the breadcrumbs to be ineffective because he did not even notice them. Another user found that the "previous search" breadcrumb did not always perform as expected. This issue arose due to the dynamic creation of the breadcrumbs. A similar expectation result was noted when using the static navigation to navigate the site (Question 3). However, there were three users (18%) who found that getting back to where they came from was neither easy nor difficult (Question 4).

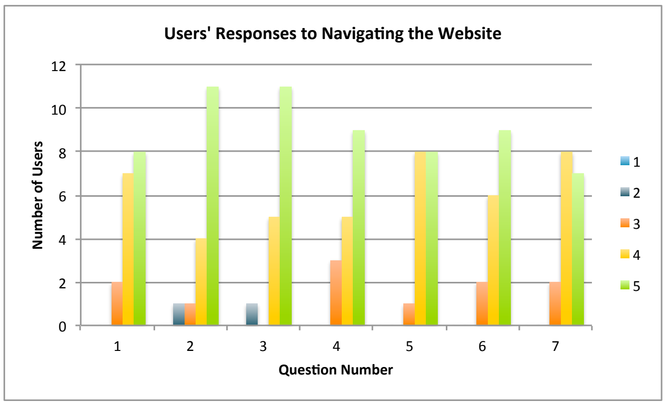


Figure 30 Users' responses to navigating the interfaces

Overall, the end-users were satisfied with the visibility of the information on the Web page. On average (see Figure 31) the users were 88% satisfied with the navigability of the site (Question 6) and only two users felt that the navigability was a little bit frustrating (Question 7). Users commented that there should be a registration button that is visible to the user, instead of having to click "log in" to see it.

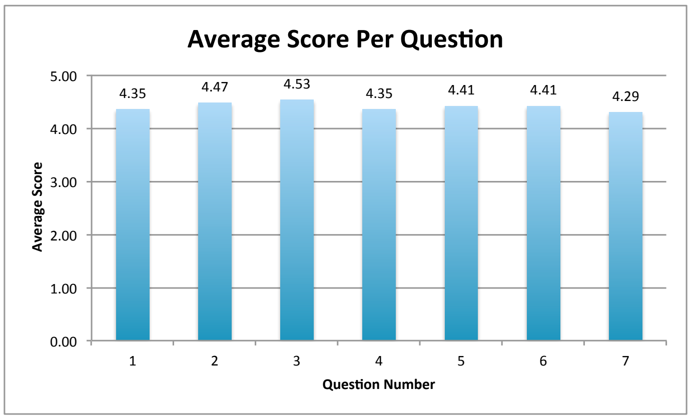


Figure 31 Average score per question on navigation

Similarity Evaluation

No tasks were to be conducted in this section; only an evaluation questionnaire of five questions was to be completed.

[4] Breadcrumb navigation is the form of navigation that shows the user the pages that they have visited to get to where they currently are.

Of the 17 users evaluated, only 2 of them (12%) had never used a DRS before. Of those who had used one before, 12 of them (80%) felt that the Bonolo end-user interface was comparable to other DRSes in terms of its feature set (Question 2, see Figure 32). 13 users (87%) felt that the end-user interface of Bonolo was as intuitive and easy-to-use as other DRSes (Question 3). Question 4 showed that only one user, of the 15, felt that the interface responded neither as expected, nor unexpectedly. The other users felt that it did respond to user interaction as expected. Question 5 indicated that the 100% of users felt that Bonolo responded to their page requests in a reasonable amount of time, comparable to other DRSes.

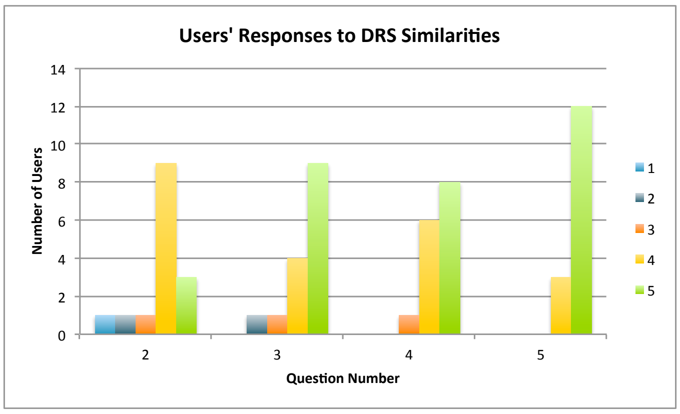


Figure 32 Users' responses to DRS similarities

General Evaluation

No tasks were completed by the end-user in this section. They answered an evaluation containing seven questions instead.

In terms of the general questions asked, the following results were noted. Question 1 asked the user whether they felt that Bonolo was general enough to cater for other digital collections. The response to this question was mixed, with 11 users feeling that Bonolo is general enough. The other six users felt that the solution was specific to displaying collections that stored typed of books or printed media. One of these users felt that currently it is specific, but it has the potential to be expanded to support other collections.

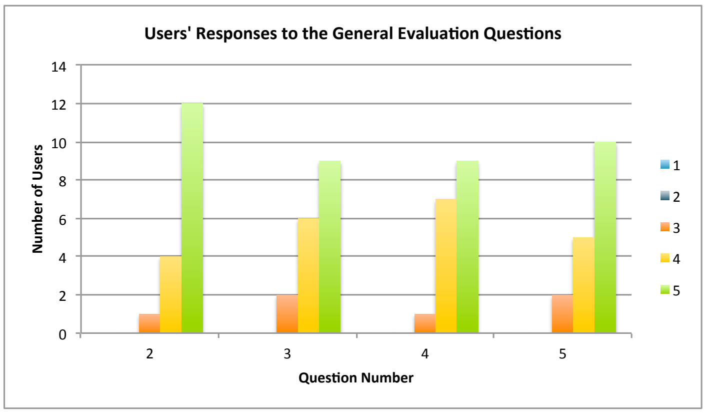


Figure 33 Users' responses to the general evaluation questions

After having used the system and performed all of the tasks at hand, 16 of the 17 users felt that the system was fairly intuitive to use (Question 2, see Figure 33). Furthermore, Question 3 resulted in only two users feeling that the system was neither complex nor straightforward to interact with. The other users felt that it was fairly straightforward to interact with. Only two users felt indifferent about using the interface, whereas the other 15 users felt fairly satisfied and comfortable with it (Questions 4 and 5).

Question 6 then asked the user if they could identify how they thought the digital collection was stored on the server. Three users (18%) thought that the collection was stored as files and folders. The other 14 users thought that the collection was stored in a database. Of these 14, only four users thought that there were files and folders used for storage too.

This question was followed up by asking the user if they could feel a performance difference due to the way the collection was stored. All 17 candidates answered that they could not feel a hindrance on performance, and that the way the data was stored was transparent to them.

Unexpected Outcomes

The results that were unexpected are as important as the results that were expected from the evaluation. The unexpected results help to identify aspects of the interface that detracted from the experience of using the Web application. The majority of the unexpected outcomes pertained to the search and browse functionality interface.

It was noted that when conducting the first search and browse from the home page, very few of the users used the faceted search options. On examination, the users felt overwhelmed by the amount of information on the home page, and went directly for the search bar, as it seemed to be the object that was most natural to them. Furthermore, users stated that they were used to a search paradigm as opposed to a browsing paradigm – hence the immediate instinct to search using the search bar.

Despite moving and adjusting the “New search” radio button on the search and browse page, numerous end-users still did not see it. However, when they did figure out how it worked, they felt it was intuitive and natural. One user suggested that this could be improved by changing the radio buttons to buttons with pictures, as this would catch the user’s eye. It was also surprising that users preferred typing in the search terms given to them, as opposed to looking for them in the search filters first. However, the filters were initially avoided due to the abundance of information on the screen, which deterred the user’s eye. Some users even suggested that the filter categories be collapsed by default.

It was surprising that there were not too many complaints from the users about the time taken to post a comment. Generally, comments took between 4 and 10 seconds to post to the Web page – due to programming error. According to [36], users only have an attention span of 4 seconds before they lose concentration and provoke an on-screen reaction.

Lastly, despite conducting two pilot evaluations, some of the users were baffled by the questions, which hindered their experience on the system.

4.1.5. Discussion

The results that were derived from the user evaluation can be assessed and the two research questions can be answered. Initially, the end-users did feel confused by the system due to the factors discussed in the *Section 4.1.4*. Although, once they got over this learning curve and progressed through the list of tasks, their confidence increased and they felt the system was intuitive, responsive and made them feel comfortable and satisfied. Therefore, the answer to first research question - *will using a simple file hierarchy as a file store affect the overall end-user experience when using the interface?* – is no. On average, users evaluated each task and feeling with 4 out of 5. This illustrates that the users felt satisfied and comfortable with all of the interfaces. This was echoed by none of the users stating that they felt a negative effect on the interface due to the file store (*General Evaluation – Question 7*).

The results from the similarity evaluation illustrated that the users felt that there were similarities between the Bonolo end-user interface and other DRSes that they had used (e.g. Gumtree, JStor, ACM, Google Books and Wikipedia). The users felt, when comparing Bonolo to these other DRSes that Bonolo was as intuitive to use and responded as expected within a reasonable amount of time. In addition, the users felt that the services offered by Bonolo are comparable to those offered by other DRSes. It is important to recall that the tests were conducted over a Local Area Network, therefore, the response times from Bonolo were always expected to be fast; data transfer over a Local Area Network is significantly faster than a broadband connection. Despite this, the second research question – *is it possible to create an end-user interface, built on a simple file store that is comparable to other DRSes?* – can be answered as yes, for the previously mentioned reasons.

4.2. Performance Evaluation

The second evaluation performed on the system is a performance evaluation. This involves testing the response times of the system as the digital collection increases in size. As a result, standard tests were conducted to determine how well the system scales and how robust it is. All of the performance evaluations were conducted on the Apple Macbook with the specifications described in *Section 4.1*.

4.2.1. Aim

The aim of the performance evaluation is to answer the remaining research question:

1. What is the impact of using a hierarchical file-based data store on the performance of the end-user interface?

4.2.2. Methodology

The performance evaluation involved a logarithmic increase in the size of the collection – testing collections with 1 000, 2 000, 4 000, 8 000, 16 000, 32 000 and 64 000 digital objects. The actual structure of the collection remained the same in these tests. With each increase in the number of stories, sets of tasks were carried out, in order, six times each until a mean response time was recorded. Of these six repetitions, three were from cold starts, and three were warm starts[5]. In addition to this, the amount of data transferred from the server to the client was recorded.

This evaluation was conducted over the Local Area Network, as this removes the unreliability of an Internet connection. Google Chrome's "Inspect Element" feature was used to record the page load and response times, as well as the amount of data that was transferred. In addition, caching was turned off for the evaluation to ensure that page load times were not skewed. Finally, before each task set was conducted, the server was restarted and the collections were re-indexed. To record the indexing times, the Solr administration Web page was used (`<Server IP:Port>/<webapp>/bonolo/dataimport`), as it records the time in seconds, to the nearest thousandth of a second. The amount of time taken to conduct a Solr search was recorded using the `QueryResponse.getQTime()` and `QueryResponse.getElapsedTime()` methods. The former method calculates the time taken to execute a query; the latter displays the total time to execute the query and to transmit the results.

4.2.3. Results

The performance tests were conducted and the results were documented. The average scores were then calculated and analysed and were grouped by their test type for analysis.

Test 1: Indexing Times

The indexing times of the seven collection sizes are illustrated in Figure 34. From this graph it can be seen that there is an exponential increase in the indexing time as the collection increases in size. The change in indexing time from a 1 000 story collection to a 64 000 story collection is a 4 698% time increase. The collection would have increased by 6 300%, therefore displaying the non-linear relationship between the index time and the size of the collection. The average time taken to index a collection of 64 000 stories is approximately 248 seconds (4 minutes and 8 seconds).

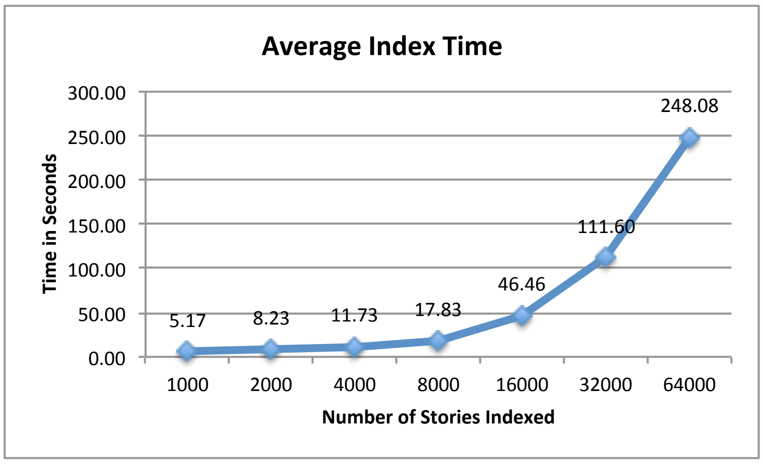


Figure 34 Average indexing times for the various collection sizes

[5] A cold start means that the server is first restarted and then the task is conducted. A warm start is when the server is not restarted before the task is conducted.

The times taken to index the collections in Figure 34 can be manipulated to calculate that the average number of objects indexed per second is 302.17 objects. Therefore, using this figure, one can estimate that the time to index one million objects would take approximately 3309.45 seconds (55 minutes and 9 seconds).

When conducting the performance evaluation, it was interesting comparing the indexing times from a cold and warm start respectively. These average index times can be seen in Figure 35.

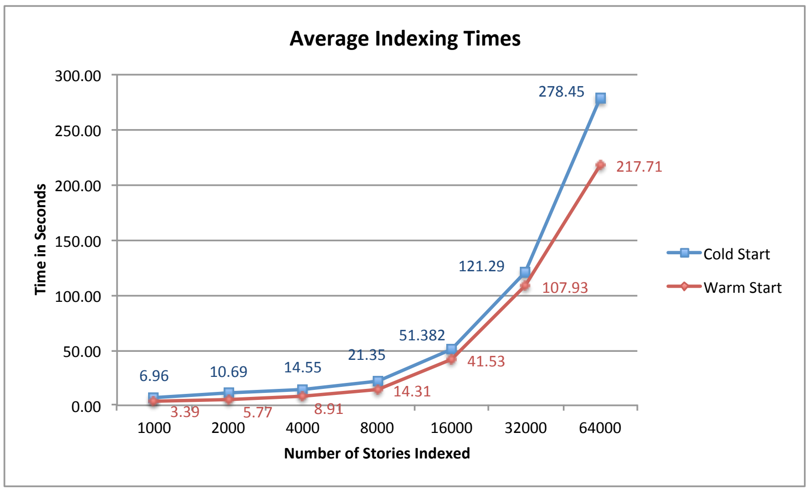


Figure 35 Average indexing times from a cold and warm start

From the figure above, the difference in index times becomes more significant as the collections grow in size. On average, there is a 32% decrease in the indexing time when the index is created from a warm start. This is noteworthy as it illustrates the benefit of not restarting the server before doing an index. An important thing to recognise is that while the indices are being built, the Web interfaces still function. This is due to the fact that the changes to the index are only committed once the indexing process has been completed. In addition, the server does not have to be restarted before the new index is used by the Web application.

Because of these index times and the fact that a delta-import cannot be used to index XML files, it is recommended that the indices are not created whenever a new metadata file as added to the collection. Instead, the collection could be re-indexed twice daily or nightly, depending on the number of items in the collection.

Test 2: Solr Query Times

This test was conducted to identify the average amount of time taken to conduct a query on the Solr index. The query was always to return the full collection to the user. Two measures were used: the `QTime()` method and the `getElapsedTime()` method, as mentioned earlier.

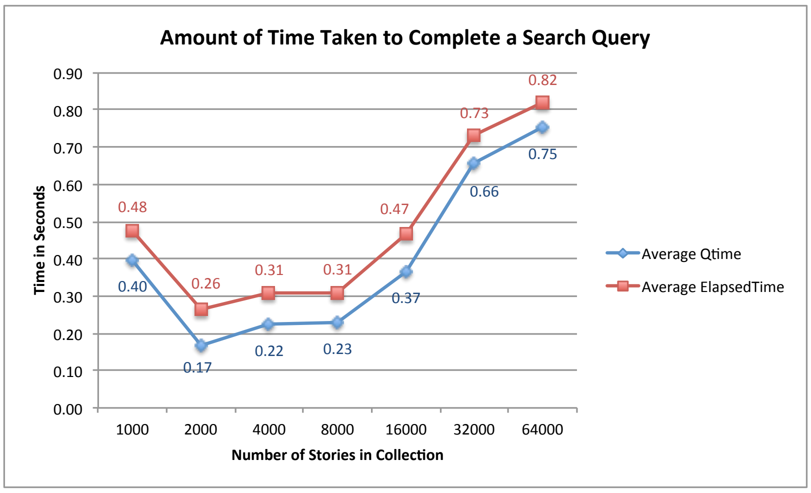


Figure 36 Average times taken to conduct a search query

The results indicated in Figure 36 show that the query time does, in fact, increase as the collection increases – there is a clear upward trend in the average query time. It is not clear as to why the search time was comparably longer for the collection of size 1 000. This could be as a result of inconsistent testing. As can be expected, the `QTime` is always shorter than the `ElapsedTime`. This is due to the fact that the `QTime` only takes into account the time taken to execute a query. `ElapsedTime` measures the total time to execute the query and to transmit the results. Therefore, it can be noted that even as the collection increased in size, the increase in search time was not in proportion to this increase.

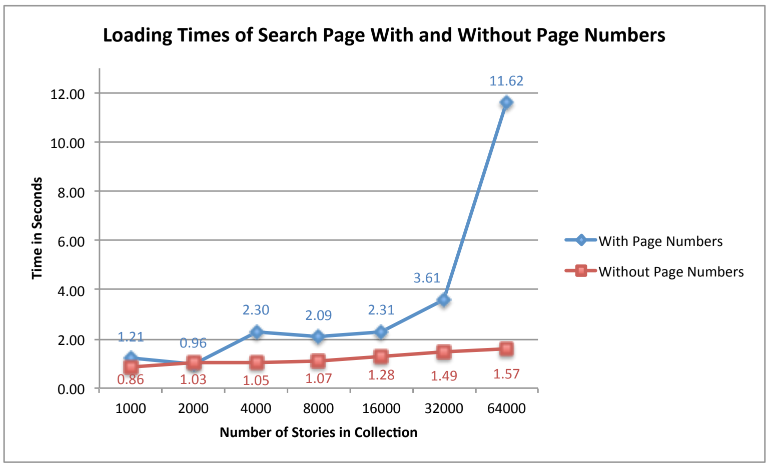


Figure 37 Loading the search page with and without page numbering

The time taken to load the search page so that the user could browse them was also explored. The purpose here is to identify whether the page can load all of the results before a user's attention span causes a negative effect on system usability. Figure 37 shows a graph detailing the page load times with and without page numbers of the search page. The first thing that stands out is that there is a significant increase in the page load time when page numbers are included. This is due to every page being output to the search page. This is a lengthy and processor intensive action. For example, with the collection of size 64 000, 6 400 page numbers were output – 10 stories per page. This poor performance issue is solely due to the programmer not generating the page numbers correctly, and this is the reason for the comparison. Generally, only 10 or 15 page numbers are shown on the page regardless of the number of results, not in excess of 200 as was the case here.

When removing the page numbers from the search page, it is evident that the page load time does increase as the number of digital objects increases. However, the page load time does not increase as drastically as it did when the page was loaded with the faulty page numbering algorithm. The *red* line – showing a linear relationship between the time taken and the number of stories – offers a better estimate of the page load times, because it would be inefficient to show in excess of 200 page numbers on a Web page. From the above results, it is shown that the search page will load all of the stories in less than two seconds.

Test 3: Viewing a Story's Web Page

In this test, two types of story were loaded: a story with 21 pages[6] and a story with no pages. The idea here was to gain a benchmark page-load time for these two cases.

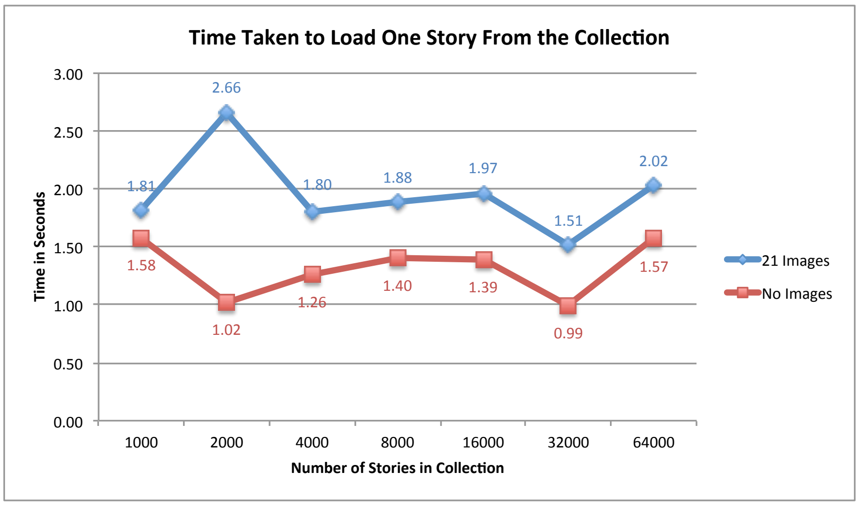


Figure 38 Time taken to view a single story in the collection

Figure 38 displays the average times taken to load the two story pages in the collection. It can be noted that the story with 21 images takes between 500 milliseconds and one second longer to load than the story with no images. This result is to be expected, due to the fact that the images had to be loaded onto the page. What was unexpected was the spike at 2 000 stories. This result may have been recorded due to poor performance testing on the researcher's part. However, the general trend that can be identified from the graph is that despite the increase in collection size, the story page load times appear to remain constant: approximately 1.95 seconds and 1.32 seconds for the 21-page story and no-page story respectively.

Test 4: Data Transferral

The amounts of data transferred will be consistent regardless of the size of the collection. As a result, 6 different books, stories and searches were conducted to assess the average amount of data that is transferred. Also, the user evaluation that was conducted in *Section 4.1*, was

[6] 21 pages was chosen as a mid-range number of story pages. Some stories can have in excess of 40 pages.

then carried out by the researcher and the total data transferred was recorded – this emulates a user browsing the website for 10 minutes. These tests were repeated twice each: once with thumbnails and once without thumbnails. This will display the benefit of thumbnails in reducing the amount of data being transferred.

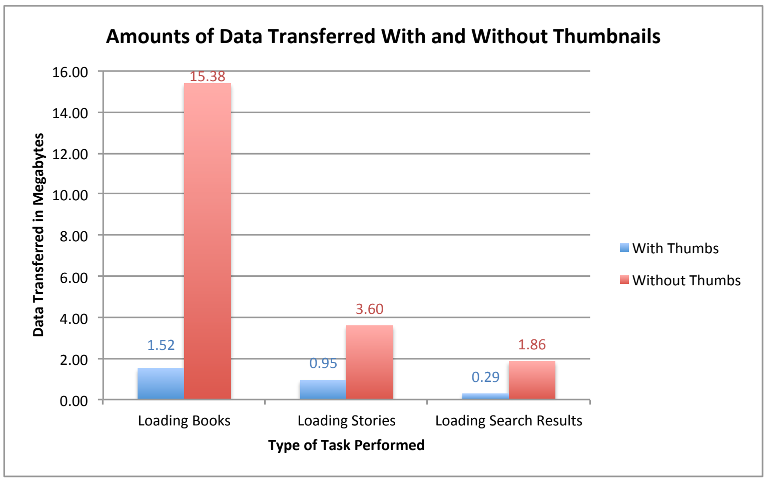


Figure 39 Average data transferral with and without thumbnails

Figure 39 highlights the disparity in the amount of data that was transferred between the server the Web browser. Without thumbnails, the amount of data transferred ranged between 1.86 and 15.38 megabytes depending on the task. However, the data transferred with thumbnails in place was between 0.29 and 1.52 megabytes. This is a large decrease in data transfer, which can have a very positive effect on the performance of the Web application. When viewing a book, there is a 90.11% decrease in data being transferred when thumbnails are used. The trend remains consistent for loading a story page and search results with a 73.49% and 84.55% decrease in data transferred, respectively.

The mock-user interaction that was conducted displayed similar results to those mentioned previously. Figure 40 displays the amount of data transferred with and without thumbnails when doing the user emulation. There was a 52.96% decrease in data being transferred – from 100.47 to 47.26 megabytes – when adding thumbnails to the Web application.

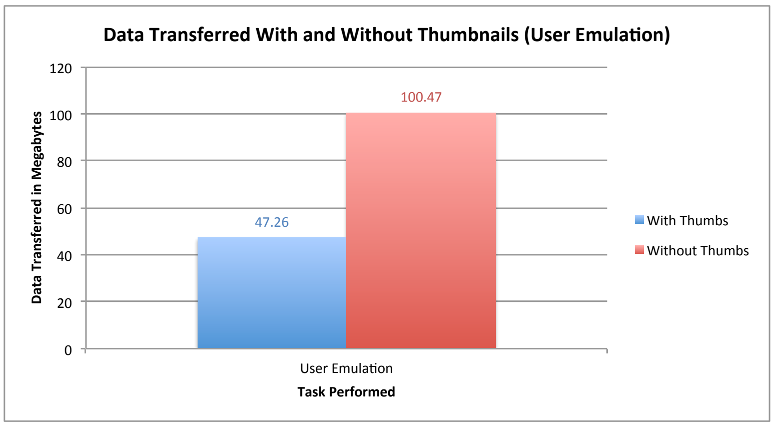


Figure 40 Data transferred when emulating an end-user

Unexpected Results
An unexpected result was noticed early on in the performance evaluation. When assessing and evaluating the page load time for displaying all the search results, the majority of the page load time was spent on generating the page numbers – up to 11 seconds. However, the search results were flushed to the display as soon as they were loaded, so this would allow the end-user to browse while the page numbers loaded. This is not ideal, as the page numbers should not take up so much processing time.

To compensate for this, and remove the hindrance of the poor page numbering algorithm, the evaluation was repeated without page numbers being displayed. This displayed a more distinct and realistic relationship between the increased index and the page load times.

4.3. Discussion

The results of the evaluation suggest that the answer to the final research question – *what is the impact of using a hierarchical file-based data store on the performance of the end-user interface?* – is that the effects of this are felt, but only in certain areas of the Web application.

On average, a user will wait as long as four seconds for a Web page to load, before deciding to desert it [36]. When Web pages take in excess of four seconds to load, the end-users start to feel dissatisfied with their interaction with that site. With the test bed collection that was used in the user-evaluation, the page load times occurred within the four-second range. However, this performance evaluation examined whether those response times would be replicated as the collection scaled up by 6 300%.

Solr indexes an estimated average of 302.17 objects, per second. Thus, as the digital collection increases in size, the collection will take significantly longer to index. This coupled with the fact that the index cannot simply be updated, means that large collections should be re-indexed over night, or at the end of a week. However, performance improvements can be noted when the re-index was conducted from a warm start. It is estimated that there is a 32% decrease in time taken when the collections are indexed from a warm start, as opposed to a cold start. Therefore, this is a hindrance on performance, as the end-users may be interacting with an out-of-date collection of information. Databases are able to perform delta-updates, which simply adds the updated fields to the search index, instead of re-indexing the entire collection.

When assessing the response times for page loads and search queries the results were positive. For all of the collection sizes tested, the average time taken to complete a query was less than one second. Unfortunately, due to a poor pagination algorithm, these query times were not reflected in the time taken to display the results of these queries to the user. As the collection increased, the time taken to display the results increased significantly; it would take up to 11.62 seconds to load the Web page completely for a collection with 64 000 objects. To compensate for this, the same queries were conducted on the various collection sizes, but without conducting the pagination algorithm. As a result, the time taken to display the query's results took 1.57 seconds for a collection of 64 000 objects. This shows that if the pagination algorithm were improved, page load times will decrease to within the four-second range.

When loading the “view story” pages with and without images, the page load times averaged at 1.95 and 1.32 seconds respectively. This falls within the user’s attention span, showing that there was not an adverse effect on performance.

The final test displayed that there was a significant decrease in bandwidth required when image thumbnails were used, as opposed to the original images only. This is essential in ensuring that the page can load within a reasonable amount of time if it were to be hosted on the Internet. To further reduce bandwidth consumption, preloading the book and story images can be disabled.

5. Future Work

While developing the Bonolo end-user interface, there were suggestions and improvements that were received, but were out of scope for the project. Instead, they could be implemented in future research projects. These recommendations and enhancements are discussed in this section.

5.1. Continuous Improvements

User and performance testing revealed that there are numerous improvements that can be made to the Bonolo Web application. Applying the knowledge obtained from the evaluations, continuous improvements could be made to the interface and layout of information.

This could lead to a more user-friendly and intuitive way to navigate the digital collections and to improve the experience of interacting with the data. Improvements could be made to the current annotations system. This could help to streamline the commenting system so that the amount of time required to post a comment is decreased. Also, the comments could be displayed in a more intuitive manner, possibly adding in other social media tools, e.g., “liking” a post or story.

The page numbering system on the search page could be improved so that it does not negatively affect the page load times as was evident in the final performance evaluation.

5.2. Exporting Images

Due to time constraints, the user was only able to download story and book images in a compressed zip folder. Future work involves allowing the images to be exported in different file formats, e.g., PDFs, compressed tarballs or Microsoft Word documents.

5.3. Account Settings

The scope of the project did not include an account management system. The user could only register an account and could not customise it further. Associated with this could be the ability for users to bookmark collections that they are viewing; customising their nickname; resetting their password; adding information about themselves; uploading an avatar; and disabling email notifications.

5.4. Viewing Other Digital Objects

Currently, the Web application can only display image files to the end-user. However, for this solution to be more dynamic and general, support for other digital media could be included. These digital media could include: sound files; video files; and other digital documents, such as PDFs.

5.5. System Configurability

Due to time constraints, the configurability of the system was not made very user friendly. In the future, the configurability of the Web application could be transposed to a Web page that an administrator has access to. This Web page would allow the administrator to index the digital collections and adjust the layout of the information displayed to the user. In addition, the administrators should be able to configure the number of search indices to make accessible to the end-user (currently, there are two: books and stories). This future work would allow the administrators to update the Web pages remotely, as opposed to having to be on the server.

5.6. Interoperability

Interoperability is the ability for one application to interact with another application by sharing and exchanging information. Future work could allow for users to subscribe to RSS feeds that inform them about recent activity on a collection they are monitoring. Otherwise, they could have the option to share the information they are viewing on Facebook, Twitter or other social networking sites.

5.7. Additional Testing

Additional user evaluations could have been done with the Web application being hosted on the Internet. This would allow for more realistic data response times to be recorded and would make some of the evaluation questions more relevant.

The performance testing conducted on the Bonolo end-user interface could be more rigorous. Other performance evaluations – such as altering the file structure – could be conducted so as to further investigate the scalability and robustness of the end-user interface. Also, more tests should be conducted so that more data can be captured. This will help to derive more accurate estimations and relationships from the results.

6. Conclusions

Background information suggests that DRSes manage collections of data stored in predefined file stores. However, few DRSes make use of simple file stores to house their collections; they tend to use complex structures, such as databases. The aim of the project is is to develop a set of Web-based tools around a predefined XML-centric digital collection. These tools will provide core DRS services such as search and browse to end-users. Additional features were suggested from end-users to add value to the Bonolo end-user interface.

The framework of the system was outlined and it illustrated that each component in it contributes to making an end-user interface that is effective, beneficial and intuitive to use for the end-user.

The iterative design and implementation of the system proved beneficial. Each of the iterations utilised the evaluation results of the previous iteration and built on it.

The main functionality of the components of the interface were all completed on schedule. However, due to time constraints, some small issues are still present in the components. But, given additional time, these issues can be removed to have a well-functioning Web application.

A number of possibilities for future work on the research topic have briefly been discussed and identified.

Despite the Bonolo end-user interface being experimental, it is believed that with a few minor adjustments to some of the functionality, it could be implemented in reality. However, further performance testing would need to be conducted.

Three research questions were to be answered when completing this project. In answering the first question – w*ill using a simple file hierarchy as a file store affect the overall end-user experience when using the interface?* – it was found that despite the simple file store, this did not detract from the end-user experience. Users generally felt comfortable and satisfied with the way that the interface behaved and made them feel.

The second question – *what is the impact of using a hierarchical file-based data store on the performance of the end-user interface?* – was answered by noting that there are some performance compromises when scaling the collection size up. The tests showed that page load times remained consistent at approximately 2 seconds – thus showing that there is no negative effect on page display times. It was found that the algorithm for paginating the search results was detrimental in maintaining reasonable page load times. It is suspected that if this algorithm were improved, then page load times on the search page will become reasonable. The amount of data transferred from the server to the Web browser showed significant improvements when thumbnails were introduced. Thus reducing up to 90% of data transferred, making all of the Web pages under 2 megabytes to load.

When answering the final research question – *is it possible to create an end-user interface, built on a simple file store that is comparable to other DRSes?* – it was found that users felt that the Bonolo end-user interface was comparable to the interfaces offered by other DRSes. End-users felt that the feature set and usability of the system displayed this. Therefore, this suggests that it is, in fact, possible to create an end-user interface that is comparable to other DRSes, despite it being built on the simple file store. However, for this to be more conclusive, a more comprehensive user evaluation would have to be carried out.

7. Bibliography

[1] M. Bowes, "CALJAX : An In-Browser Digital Repository System," Department of Computer Science, University of Cape Town, Cape Town, Western Cape, 2009.

[2] H. Suleman, "Digital Libraries Without Databases: The Bleek and Lloyd Collection," in *Research and Advanced Technology for Digital Libraries*, L. Kovács, N. Fuhr, and C. Meghini, Eds. Heidelberg, Berlin: Springer-Verlag, 2007, pp. 392–403.

[3] The Apache Software Foundation, "Welcome to Solr," 2011. [Online]. Available: http://lucene.apache.org/solr/. [Accessed: 12-Oct-2011].

[4] The Apache Software Foundation, "Apache Tomcat - Welcome!," 2011. [Online]. Available: http://tomcat.apache.org/. [Accessed: 10-Oct-2011].

[5] ImageMagick Studio LLC, "ImageMagick: Convert, Edit, Or Compose Bitmap Images," 2011. [Online]. Available: http://www.imagemagick.org/script/index.php. [Accessed: 15-Oct-2011].

[6] The jQuery Project, "jQuery: The Write Less, Do More, JavaScript Library," 2010. [Online]. Available: http://jquery.com/. [Accessed: 12-Oct-2011].

[7] C. Thompson, "qTip2 - Pretty powerful tooltips," 2011. [Online]. Available: http://craigsworks.com/projects/qtip2/. [Accessed: 18-Oct-2011].

[8] Hipp Wyrick & Company Inc., "SQLite Home Page," 2011. [Online]. Available: http://www.sqlite.org/index.html. [Accessed: 09-Oct-2011].

[9] S. Payette and C. Lagoze, "Flexible and extensible digital object and repository architecture (FEDORA)," *Research and Advanced Technology for Digital Libraries*, vol. 1513, pp. 517–517, 1998.

[10] M. K. Smith et al., "DSpace: An Open Source Dynamic Digital Repository," *D-Lib Magazine*, vol. 9, no. 1, Jan. 2003.

[11] C. Lagoze et al., "Core Services in the Architecture of the National Science Digital Library (NSDL)," in *Proceedings of the 2nd ACM/IEEE-CS Joint Conference on Digital libraries*, New York, New York, USA: ACM, 2002, pp. 201–209.

[12] A. Kumar, R. Saigal, R. Chavez, and N. Schwertner, "Architecting an extensible digital repository," in *Proceedings of the 2004 joint ACM/IEEE conference on Digital libraries - JCDL '04*, New York, New York, USA: ACM Press, 2004, p. 9.

[13] T. Kuny and G. Cleveland, "The Digital Library: Myths and Challenges," *IFLA journal*, vol. 24, pp. 107–113, 1998.

[14] "DSpace Documentation," 2010. [Online]. Available: http://www.dspace.org/1_7_0Documentation/.

[15] R. Tansley et al., “The DSpace Institutional Digital Repository System: Current Functionality,” in *Proceedings of the 3rd ACM/IEEE-CS joint conference on Digital Libraries*, Washington, DC: IEEE Computer Society, 2003, pp. 87–97.

[16] H. Suleman, M. Bowes, M. Hirst, and S. Subrun, “Hybrid online-offline digital collections,” in *Proceedings of the 2010 Annual Research Conference of the South African Institute of Computer Scientists and Information Technologists*, New York, New York, USA: ACM, 2010, pp. 421–425.

[17] M. Baldonado, C. C. K. Chang, L. Gravano, and A. Paepcke, “The Stanford Digital Library metadata architecture,” *International Journal on Digital Libraries*, vol. 1, no. 2, pp. 108–121, 1997.

[18] R. Daniel Jr, C. Lagoze, and S. D. Payette, “A Metadata Architecture for Digital Libraries,” in *Proceedings of the Advances in Digital Libraries Conference*, Santa Barbara, California: IEEE, 1998, pp. 276–288.

[19] H. Suleman, E. A. Fox, R. Kelapure, A. Krowne, and M. Luo, “Building digital libraries from simple building blocks,” *Online Information Review*, vol. 27, no. 5, pp. 301-310, 2003.

[20] K. Shearer, “Institutional repositories: Towards the identification of critical success factors,” *Canadian journal of information and library science*, vol. 27, no. 3, pp. 89–108, 2003.

[21] H. Suleman and Edward A. Fox, “Designing Protocols in Support of Digital Library Componentization,” in *Research and Advanced Technology for Digital Libraries*, M. Agosti and C. Thanos, Eds. Heidelberg, Berlin: Springer-Verlag, 2002, pp. 75–84.

[22] T. Staples, R. Wayland, and S. Payette, “The Fedora Project: An Open-source Digital Object Repository Management System,” *D-Lib Magazine*, vol. 9, no. 4, 2003.

[23] W. Arms et al., “A spectrum of interoperability: the site for science prototype for the NSDL,” *D-Lib Magazine*, vol. 8, no. 1, p. 9, 2002.

[24] H. Suleman, E. A. Fox, and D. Madalli, “Design and Implementation of Networked Digital Libraries: Best Practices,” in *DRTC Workshop on Digital Libraries: Theory and Practice*, Bangalore, India: DRTC, 2003.

[25] R. Pandey, “Digital Library Architecture,” *Theory and Practice*, 2003.

[26] S. Hitchcock, T. Brody, J. Hey, and L. Carr, “Digital preservation service provider models for institutional repositories: towards distributed services,” *DLib Magazine*, vol. 13, no. 5/6, 2007.

[27] Oracle, “JavaServer Pages Overview.” [Online]. Available: http://www.oracle.com/technetwork/java/overview-138580.html. [Accessed: 20-Oct-2011].

[28] Oracle, “Java Servlet Technology Overview.” [Online]. Available: http://www.oracle.com/technetwork/java/overview-137084.html. [Accessed: 20-Oct-2011].

[29] Sun Microsystems, “JavaBeans Components in JSP Pages,” *The J2EE Tutorial for the Sun ONE Platform*, 2003. [Online]. Available: http://download.oracle.com/javaee/1.3/tutorial/doc/JSPIntro11.html. [Accessed: 20-Oct-2011].

[30] A. Aktiebolag, “Galleria – The JavaScript Image Gallery,” 2011. [Online]. Available: http://galleria.aino.se/. [Accessed: 10-Sep-2011].

[31] RAD Lab, “Full Text Search Using SOLR,” *Framework*. Berkeley, University of California, California, pp. 1-16, 2008.

[32] C. Hostetter, “Faceted Searching With Apache Solr.” ApacheCon, Austin, Texas, pp. 1-35, 2006.

[33] M. Jones and G. Marsden, *Mobile Interaction Design*. West Sussex, England: John Wiley & Sons Ltd, 2006, p. 369.

[34] The Apache Software Foundation, “DataImportHandler - Solr Wiki,” 2011. [Online]. Available: http://wiki.apache.org/solr/DataImportHandler. [Accessed: 20-Aug-2011].

[35] N. Bevan, “What is the difference between the purpose of usability and user experience evaluation methods?,” *Proceedings of the Workshop UXEM’09*. Uppsala, Sweden, 2009.

[36] Akamai Technologies Inc., “Akamai and JupiterResearch Identify ‘4 Seconds’ as the New Threshold of Acceptability for Retail Web Page Response Times,” 2006. [Online]. Available: http://www.akamai.com/html/about/press/releases/2006/press_110606.html. [Accessed: 25-Oct-2011].

8. Appendices

A: Consent Form for the Final UX Evaluation

Informed Consent Form for Students

This informed consent form is for students who we are inviting to take part in Computer Science Honours research, entitled "Bonolo: A Web-Based Digital Repository System".

Principle Investigator:	Stuart Hammar
Organisation:	Computer Science Honours (CSC4003W) University of Cape Town
Project and Version:	Bonolo: A Web-Based Digital Repository System (Final Evaluation)

Part I: Information Sheet

Introduction
I am Stuart Hammar, a Computer Science Honours student. I am doing research on the development of the end-user interface of a Web-based digital repository system. I am going to give you information and invite you to be a part of this research. If there is anything that is unclear or that you do not understand, please ask me to stop and I will explain it more clearly. If you have questions later, you can ask them at any time.

Purpose of the Research
A Digital Repository System (DRS) is software that takes physical pictures and documents and stores them digitally for preservation and accessibility. Such software provides users with an end-user interface so that they are able to access the digitised information by searching and browsing. I am exploring a way to create an end-user interface for a DRS that is based on a different way of storing the information. I believe that you can provide me with insight and information into helping me evaluate the success of my system. I want to learn about the user's experience when using my system and whether there are any changes, suggestions, praises and difficulties you may find when using it, as this will be beneficial towards future changes in the system.

Participant Selection
You have been invited to participate in this research because I feel that your experience as a regular Internet user, and your regular exposure to search engines and certain social media techniques can contribute much to the evaluation of my system.

Voluntary Participation
Your participation in this evaluation is entirely voluntary. It is your choice whether you participate or not, and you are able to opt out at any point during the evaluation. It is important to remember that it is the system being evaluated and not you. Any responses or input that you give during the evaluation will be beneficial.

Evaluation Procedure
I am going to be asking you to help me with an evaluation of the end-user interface I have developed for my Web-based DRS. The current set of information in the repository pertains to the Bleek and Lloyd Notebooks. These are the notebooks of researchers who documented the Khoi-Khoi, and have been digitally preserved.

If you decide to participate, you will be given a task list and will be expected to complete each task on it. Following each set of tasks on the list, there is an evaluation to be completed by you. This evaluation contains questions about your experience when completing the tasks and is measured on a scale of one to five. I will also be documenting your actions by writing down any responses or activities that were interesting or problematic.

The tasks are set out and worded in such a way that there are no direct hints as to what you should click or do. However, if you need assistance in completing a task, I am available to assist you. It will be helpful if you can talk aloud as you perform the tasks, but if you do not, it is not an issue.

Note: *If at any point the system reacts unexpectedly or freezes, or you get stuck and would like to restart the task, you may return to the Bonolo homepage. From there, I will assist you in returning to the task you were on.*

Duration
This evaluation will last approximately forty-five minutes to an hour.

Confidentiality
The research I am conducting may be made publicly available through the University of Cape Town. As a result, all personal information relating to you will not be used, written or published in my final research document; it will be kept private. Only the results of this evaluation will be published.

Part II: Certificate of Consent

I have been invited to take part in Stuart Hammar's user evaluation. This is research that will be used towards his Computer Science Honours project. The research topic is about the development of an end-user interface of a Web-based digital repository system.

Participant's Statement

I have read the foregoing information, or it has been read to me. I have had the opportunity to ask questions about it and any questions I have asked have been answered to my satisfaction. I consent voluntarily to be a participant in this study.

Print Full Name of Participant: ________________________________

Signature of Participant: ________________________________

Date: ________________________________

Researcher's Statement

I have accurately read out the information sheet to the potential participant, and made sure, to the best of my ability, that the participant understands what will be done.I confirm that the participant was given an opportunity to ask questions about the study, and all the questions asked by the participant have been answered correctly and to the best of my ability. I confirm that the individual has not been coerced into giving consent, and the consent has been given freely and voluntarily.

Print Name of Researcher Taking Consent: ______________________________

Signature of Researcher Taking Consent: ______________________________

Date: ______________________________

B: Final User Experience Evaluation

Note: *You may suggest a change, an opinion or an idea that is not on the evaluation sheet at any time during this evaluation. I will happily jot it down and take your suggestion into account.*

Searching and Browsing Stories:
Go to the Bonolo home page.

Tasks:

1. Conduct a search/browse for the category "Artefact and Dress".
2. Refine your search/browse by the following:
   a. "Dia!Kwain (David Hoesar) (V)" as the author
   b. With the keyword, "Quagga"
   c. And the sub-keyword, "Quagga makes flour"
3. You accidentally chose the keyword "Quagga" and sub-keyword "Quagga makes flour". Remove them from the search/browse
4. Refine your search/browse again by searching for "cloud"
5. Clear all of the terms you searched for
6. Conduct a broad search, or new search, for "Lucy Lloyd"
7. View the results on the 87th page

Evaluation:
1. When you first conducted a search/browse it felt:

Unnatural & Counter-Intuitive----(1)----(2)----(3)----(4) ----(5)----Natural & Intuitive

If it felt unnatural and counter-intuitive, please state why?
____
____

2. How effective was the ability to refine a search (i.e. the entire filtering process: adding filters, removing, setting it to search within results):

Ineffective----(1)----(2)----(3)----(4) ----(5)----Effective

3. How intuitive was the ability to refine a search (i.e. the entire filtering process: adding filters, removing, setting it to search within results):

Counter-Intuitive----(1)----(2)----(3)----(4) ----(5)----Intuitive

If you felt it was counter-intuitive, please explain why and how it could be improved?
____
____

4. The complexity of conducting a new search, or broad search, was:

Complex----(1)----(2)----(3)----(4) ----(5)----Straight-Forward

5. Once you had completed your first search/browse with the software, how satisfied were you with this feature (i.e. did things happen as you would have expected):

Dissatisfied----(1)----(2)----(3)----(4) ----(5)----Satisfied

6. Did you feel that the way the system displayed the search results was as you would expect search results to be displayed:

Unexpected----(1)----(2)----(3)----(4) ----(5)----Expected

7. Did the system conduct searching and browsing as you would expect it to:

Unexpected----(1)----(2)----(3)----(4) ----(5)----Expected

8. Did the system respond to the search/browse requests in a reasonable amount of time (i.e. did the search/browse results load in a reasonable amount of time):

Unreasonable (Slow)----(1)----(2)----(3)----(4) ----(5)----Reasonable (Fast)

9. When you did an action, did the system respond as you would have expected it to (i.e. when you clicked on certain buttons, did the interface react/change as you thought it would):

Unexpected----(1)----(2)----(3)----(4) ----(5)----Expected

10. Overall, using the search and browse interface and its features made you feel:

Uncomfortable & Dissatisfied----(1)----(2)----(3)----(4) ----(5)----Comfortable & Satisfied

11. Overall, how frustrated were you when using this feature:

Frustrated----(1)----(2)----(3)----(4) ----(5)----Not Frustrated
If you were frustrated, what made you frustrated?
____
____

Viewing Stories:
Go to the Bonolo home page.

Tasks:

1. Browse the stories written by “!nanni”
2. Go to page 7 and view the story entitled “|xue and his father-in-law”

(You are now viewing the story page)

3. View all of the pages of this story (i.e. just go through the pages to get a feel for the interface)
4. View a page in its full size
5. Find a story that has been suggested to you that is entitled “|xue and his father” and view it
6. Download the pages of this story
7. Find other stories that are about “resurrection”

Evaluation:
1. The complexity of viewing more information about a story (clicking the name on the search page) was:
Complex----(1)----(2)----(3)----(4) ----(5)----Straight-Forward

If it was complex, please explain why? ____

2. When you viewed a story, did the system respond in a reasonable amount of time when loading the web page and the images:

Unreasonable (Slow)----(1)----(2)----(3)----(4) ----(5)----Reasonable (Fast)

3. How complicated was it to navigate and view the story images (i.e. view the next, previous and full-size images):

Complex----(1)----(2)----(3)----(4) ----(5)----Straight-Forward

If it was complex, please explain why? ____

4. When viewing the story images, did the function work as expected:

Unexpected----(1)----(2)----(3)----(4) ----(5)----Expected

If it was unexpected, please explain why? ____

5. How beneficial is the ability to browse the similar stories to the current story:

Not Beneficial----(1)----(2)----(3)----(4) ----(5)----Beneficial

6. How effective is the ability to browse stories that have similar topics to the story being viewed:

Ineffective----(1)----(2)----(3)----(4) ----(5)----Effective

7. How beneficial is it to be able to download the story pages:

Not Beneficial----(1)----(2)----(3)----(4) ----(5)----Beneficial

8. Overall, the way the features and actions on the story page performed were:

Unexpected----(1)----(2)----(3)----(4) ----(5)----Expected

If you the interface did not perform as expected, please explain why this was so? ____

9. What was your overall feeling of using this interface:

Uncomfortable & Dissatisfied----(1)----(2)----(3)----(4) ----(5)----Comfortable & Satisfied

If you were uncomfortable and dissatisfied, please explain why this was so? ____

10. Overall, how frustrated were you when using this interface and its features:

Frustrated----(1)----(2)----(3)----(4) ----(5)----Not Frustrated

If you were frustrated, please explain why this was so? ____

Annotations and Registration:
Go to the Bonolo home page.

Tasks:

1. Display the story that has been annotated on most recently
2. Add a comment to this story
3. You do not have an account, follow the procedure for creating an account with the following details:

a. email address: **hamstuh89@gmail.com**
   b. username: leave it as is (although you can choose your own)
   c. password: **password**

4. Now, assume you have created an account, log in with the following details:

. email address: **test@test.com**
   a. password: **password**

5. Now write any comment you like on this story
6. Once the page has re-loaded, find your comment and delete it
7. Log out and return to the home page

Evaluation:
1. How complex/confusing was it identifying the most recently commented story:

Complex----(1)----(2)----(3)----(4) ----(5)----Straight-Forward

If you found this complex, please explain why?
____
____

2. When registering, did you find the registration page to be intuitive:

Counter-Intuitive----(1)----(2)----(3)----(4) ----(5)----Intuitive

3. Did you find the registration process to be tedious and long-winded:

Long & Tedious----(1)----(2)----(3)----(4) ----(5)----Short & Simple

4. How intuitive was it to post a comment:

Confusing----(1)----(2)----(3)----(4) ----(5)----Intuitive

If you found this confusing, please explain why?
____
____

5. Did commenting on a story perform/happen as you had expected (i.e. did the entire commenting process - including deleting a comment - perform as you expected):

Unexpected----(1)----(2)----(3)----(4) ----(5)----Expected

6. Did the system respond in a reasonable amount of time when using the commenting features:

Unreasonable (Slow)----(1)----(2)----(3)----(4) ----(5)----Reasonable (Fast)

7. Did you feel that being notified on comments is beneficial:

Not Beneficial----(1)----(2)----(3)----(4) ----(5)----Beneficial

8. The overall level of satisfaction when using the comment functionality is:

Dissatisfied----(1)----(2)----(3)----(4) ----(5)----Satisfied

9. Overall, using the annotations interface and its features made you feel:

Uncomfortable & Dissatisfied----(1)----(2)----(3)----(4) ----(5)----Comfortable & Satisfied

10. Overall, how frustrated were you when using this interface:

Frustrated----(1)----(2)----(3)----(4) ----(5)----Not Frustrated
If you were frustrated, please explain why this was so?

Browsing Books:
Go to the Bonolo home page.

Tasks:

1. View the book entitled "BC_151_A2_1_122"
2. Go through the various pages of this book
3. Load a different book (your choice!)
4. Show a page in its full size
5. Download this book

Evaluation:
1. How beneficial is it to browse an entire book (cover-to-cover):
Not Beneficial----(1)----(2)----(3)----(4) ----(5)----Beneficial

2. How do you feel the interface for browsing books is:

Complicated----(1)----(2)----(3)----(4) ----(5)----Intuitive
If it was complicated, please explain why this was so?

3. Does the system respond in a reasonable amount of time when performing tasks on this interface:

Unreasonable (Slow)----(1)----(2)----(3)----(4) ----(5)----Reasonable (Fast)

4. How beneficial is the ability to download an entire book:

Not Beneficial----(1)----(2)----(3)----(4) ----(5)----Beneficial

5. Overall, using the browse interface for books and its features made you feel:

Uncomfortable & Dissatisfied----(1)----(2)----(3)----(4) ----(5)----Comfortable & Satisfied
If you chose uncomfortable and dissatisfied, please explain why this was so?

6. Overall, how frustrated were you when browsing the books:

Frustrated----(1)----(2)----(3)----(4) ----(5)----Not Frustrated

If you were frustrated, please explain why this was so?
____
____

Navigation:
Go to the Bonolo home page.

<u>Tasks:</u>

1. Navigate to the page that will allow you to view all of the stories in the collection
2. View the first story that is on page 10 of the search page
3. Return to the search page you just came from. Now go back to the home page
4. Navigate to the story that has been commented on third most recently
5. Navigate to the least similar story to the story you are viewing
6. From this page, navigate to the page that will allow you to view all of the books in the collection
7. Finally, go to the page where you can create a new Bonolo account

<u>Evaluation:</u>
1. On first use, how straight-forward did you find navigating the site:

Confusing----(1)----(2)----(3)----(4) ----(5)----Straight-Forward

If you found it confusing, please explain why?
____
____

2. How effective did you find the use of breadcrumbs in the system (breadcrumbs are the form of navigation that show you where you have come from - these are present in the top left of the web page):

Ineffective----(1)----(2)----(3)----(4) ----(5)----Effective

If you found them ineffective, please explain why?
____
____

3. How effective did you find using the main navigation:

Ineffective----(1)----(2)----(3)----(4) ----(5)----Effective

4. How easily did you find you could back-track to a familiar page:

With Difficulty----(1)----(2)----(3)----(4) ----(5)----Easily

5. Was it straight-forward to find what you were looking for on the page (i.e. are the headings and other information easy to identify):

Complex----(1)----(2)----(3)----(4) ----(5)----Straight-Forward

6. Overall, how satisfied were you with the navigability of the site:

Dissatisfied----(1)----(2)----(3)----(4) ----(5)----Satisfied

7. Overall, how frustrating was the navigability of the site:

Frustrating----(1)----(2)----(3)----(4) ----(5)----Not Frustrating

If you found it frustrating, please explain why?
____
____

General System Evaluation
System Similarities

1. Have you used any other Digital Repository Systems (DRSes) before? E.g. ACM, JStor, Google Books, Wikipedia?

Yes/No

2. To what degree are the services offered in Bonolo similar to other DRSes (i.e. search/browse, annotations, notifications etc.):

Different----(1)----(2)----(3)----(4) ----(5)----Similar

3. The end-user interface is as intuitive as other DRSes (i.e. is Bonolo as easy-to-use as other DRSes):

Complex----(1)----(2)----(3)----(4) ----(5)----Intuitive

4. The way in which Bonolo's interface responds is as you would expect from other DRSes:

Unexpected----(1)----(2)----(3)----(4) ----(5)----Expected

5. Bonolo responds to page requests in an amount of time that is comparable to other DRSes (i.e. Bonolo loads its pages in a reasonable amount of time):

Unreasonable (Slow)----(1)----(2)----(3)----(4) ----(5)----Reasonable (Fast)

General Evaluations

1. Do you think that this system is general enough to cater for various kinds of digital collections, or is it only tailored for a specific solution?
____

2. After having used the end-user interface, it felt:

Complex/Counter-Intuitive----(1)----(2)----(3)----(4) ----(5)----Intuitive

3. Overall, interacting with the interface was:

Complex/Counter-Intuitive----(1)----(2)----(3)----(4) ----(5)----Straight-Forward/Intuitive

4. Overall, using the interface made you feel:

Dissatisfied & Uncomfortable----(1)----(2)----(3)----(4) ----(5)----Satisfied & Comfortable

5. How comfortable did you feel when interacting with Bonolo's interface:

Uncomfortable----(1)----(2)----(3)----(4) ----(5)----Comfortable

If you felt uncomfortable, please explain why?

____

____

6. How do you think that the digitised information is stored in this digital repository? (E.g. In a database, in files and folders, XML documents etc.)

____

7. In Bonolo, the digitised information is stored in simple files and folders. Can you feel an effect on performance because of the way the information is stored (i.e. when searching/browsing, viewing stories/books and their images)?
Note: *Comments and Registration are stored in a Database, not simple files*

____

8. Are there any other comments or suggestions that you may have regarding the system?

____

____

____

____

Thank you for your time, patience and participation!