UNIVERSITY OF CAPE TOWN

DEPARTMENT OF COMPUTER SCIENCE

CS/IT Honours Project
Final Paper 2022

Title: Archiving Archives

Author: Alex Olivier

Project Abbreviation: Arch2

Supervisor(s): Hussein Suleman

Category	*Min*	*Max*	Chosen
Requirement Analysis and Design	*0*	*20*	
Theoretical Analysis	*0*	*25*	
Experiment Design and Execution	*0*	*20*	15
System Development and Implementation	*0*	*20*	15
Results, Findings and Conclusions	*10*	*20*	20
Aim Formulation and Background Work	*10*	*15*	10
Quality of Paper Writing and Presentation	*10*		10
Quality of Deliverables	*10*		10
Overall General Project Evaluation (*this section allowed only with motivation letter from supervisor*)	*0*	*10*	
Total marks		**80**	

Snapshot Creation for an Archive of Archives

6 September 2022

Alex Olivier
University of Cape Town
Rondebosch, Cape Town,
South Africa
olvale005@myuct.ac.za

ABSTRACT

Digital archives form a critical component of research; however, they are complex structures, which usually fall on organizations to keep and maintain. However, this does not provide certainty of their persistence or a means to easily share archives, in low resource environments. A need therefore arises from these findings. A new breed of archive, an "Archive of Archives" was therefore constructed to try and fulfill this need. To build this archive, a means of extracting data from the original archives was needed. Data was extracted by creating snapshots of these archive websites, using a custom Web scraper, with focus on sites created using popular archiving toolkits. Results indicated that these snapshots could replicate the original archive sites to a good visual degree, however missing functionality severely affects the main use case and therefore usability for users.

KEYWORDS

digital libraries, archives, archiving archives, data preservation, Simple DL, metadata harvesting

1 INTRODUCTION

Digital archives are crucial in preserving digital items of the past and present for use in the future. Work on digital archives has become of great importance over the last two decades [1]. Any improper steps in the process of archiving, as archiving is an active state, could lead to a devastating loss to a complete archive [2]. Initially, work started out converting traditional library materials to a digital form for safekeeping, however recently the number of items 'born' digitally has increased [3]. Therefore, no second redundant hardcopy is available to regenerate an item if it were to be lost. This is in stark contrast to the main aim of digital libraries, which is to preserve significant digital data and make it persistently available across networks, for generations to come, for reasons such as education and research [4].

Digital archives are typically set up by well-funded organisations and institutions that can acquire the correct skills to create and maintain these collections. This however creates concerns around not only the underlying technology used in the creation of an archive, but also on the organisations responsible for maintaining them. These institutions therefore have great liberty in the methods and accuracy of the preservation applied and access provided [5]. This results in a dependence between the archive and the continued funding and commitment of its maintaining organisation. Some archives have failed to create trust in their ability to preserve information and have shut down due to a lack of funding to their managing organisations [6]. It could be argued that the extensive number of archiving tools created and research into digital libraries has solved the archiving problem [7]. Yet, we still face problems around failures in current systems, especially in under-resourced environments.

This research project forms part of a larger project, Archiving Archives, depicted in Figure 1.

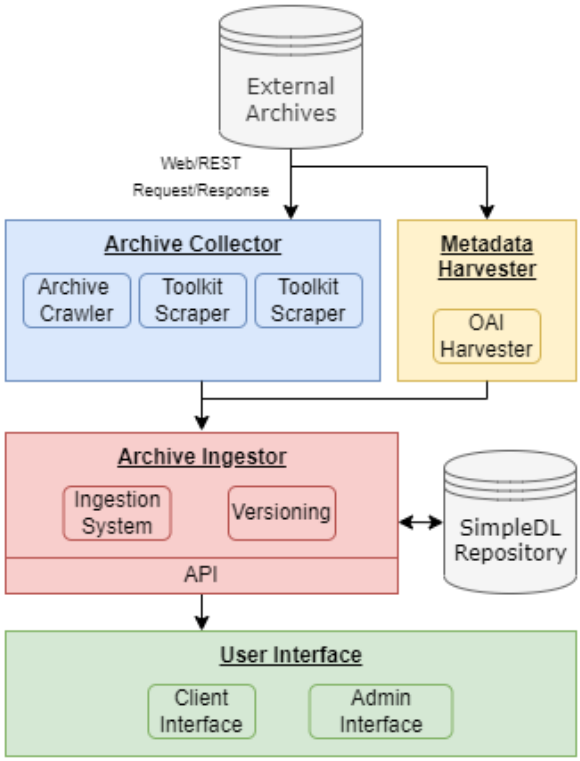

Figure 1: Archiving Archives Model

The overall research is focused on addressing some of the concerns around digital archiving, by conducting research around the creation of a new breed of digital archives, that preserve other digital archives to establish trust, versioning, and long-term persistence of data, through the public conservation of other archives.

This specific paper addresses the archive retrieval part for this Archive of Archives, depicted in blue in Figure 1. Other parts include: a metadata harvester; versioning functionality; and admin and user interfaces. In practice this research will involve the building and then experimentation on a system that creates "archive snapshots", that visually and functionally replicate archives, of already available archives, created using popular archiving toolkits, which can be included in the Archive of Archives. The goal of this paper is to investigate the degree to which this is possible.

2 PAPER LAYOUT

This paper first introduces relevant background to archives, their models, the popular toolkits we will be focusing on and Simple DL, the system which the overall project was built upon.

The research question, formulation and methodology are then covered. This is followed by how archive data was extracted by developing and implementing a system to create archive snapshots.

The experimental design and execution is then introduced, followed by the results we recorded, an analysis of these results and conclusions that we made. The paper ends with our thoughts on the direction of the project and future work to be done.

3 BACKGROUND AND RELATED WORK

3.1 Digital Library Models:

Firstly, repositories are responsible for managing their digital objects. This includes the insertion, deletion, and retrieval with restricted access to digital objects [8]. Most modern digital library's conform to the OAIS (Open Archival Information System) specification, with permanent access to the libraries content using a URI (Uniform Resource Identifier). To simplify matters, an interfacing abstraction is applied to the repository called an RAP (Repository Access Protocol) [9]. Concrete implementations can be found in OAI-PMH (Open Archives Initiative Protocol for Metadata Harvesting), Fedora and OpenDLib [10]. With the integration of the Internet however, RAP is now used for rich interactions between co-operating repositories, but HTTP-based (Hypertext Transfer Protocol) standards are embraced for Web usage [11].

Secondly, the system provides identifiers to where digital objects are stored. This is achieved by assigning general-purpose identifiers named handles, to digital objects in repositories. These handles are location independent and comprise of a repository identifier and local name [10]. Users request these when submitting digital objects to the repository [12]. The repositories are then able to return where the digital object is stored within the repository [9]. The handles are resolved by a handle server for the service provider [10]. Digital Object Identifiers (DOIs) are a well-known implementation of handles.

Thirdly, digital libraries can accrue large amounts of data. It is therefore likely that many indices and catalogs will be searched during the retrieval of information; these can be independently managed with multiple protocols [9].
Metadata is the core of any information retrieval system and impacts the ability of a digital library to deliver objects in a meaningful way, which greatly affects its long-term preservation ability [13]. Early implementations showed distinctive categories of information that should be captured: descriptive, administrative, technical, rights, digital and structural metadata. Confusion over category scope, specific requirements and the refinement of these categories means no single schema for metadata collection is prevalent. Libraries are rather implemented with an underlying metadata standard along with the institution's own metadata categories. This allows for a degree of interoperability while fulfilling their own operational requirements. Preservation metadata is also of concern to ensure the 'fixity' of information as control over the library needs to be maintained. This provides authenticity and validity to its data.

Finally, the user interface integrates the three other components, providing a two-part interface. The first part allows users to search and retrieve digital objects, while the second part allows system administrators to manage the collection [9]. The primary interface is usually Web browsers that connect to client services, an intermediary service between the browser and the rest of the system. It is common to find HTTP GET or SOAP requests for this interaction [1].

Knowledge of digital library models was key to our system development and implementation. Much underlying functionality is shared with digital repositories as a digital library is a form of digital repository. Online access to these repositories via HTTP requests meant that we could use a Web scraper to access these repositories and acquire their resources. Permanent access to repository items via a URI, that can be found in the metadata via an RAP such as OAI-PMH, meant that the archive ingestor, depicted in red in Figure 1, would have a means of finding and linking an archive item, from the archive collector, to its metadata, using a URI, from the metadata harvester, depicted in yellow in Figure 1. This linking of an archive item and its metadata is a requirement for the user interface of the Archive of Archives indexing.

3.2 Dublin Core

The Dublin Core metadata set, consisting of 15 broad elements, is one of the most common schemas for Web content and is widely used as it enables indexing by any metadata search engine [1]. It is recommended to use the schema for general data [8]. Dublin Core is also useful as it can be used on digital and physical resources. It has been found that other metadata sets produce many redundancies, especially in the structural element; this

produces unnecessary data and causes a storage and priority issue [13].

Dublin core was chosen as our supported metadata set as it was a commonly supported by the archives we tested on and had access to. This is mainly due to it being a well-known metadata set and because it is very similar to the DIM (DSPACE Intermediate Metadata) set used by DSPACE – an archiving toolkit we used during the project.

3.3 Archiving Tools

3.3.1 DSpace:

DSpace, jointly developed by MIT Libraries and Hewlett-Packard Labs, comprises of a three-layer architecture – the application, business logic and storage layers [14]. It was developed to encompass research functionality, while maintaining simplicity [15]. All three provide API interfaces for user customization and future enhancement [14]. The storage layer uses PostgreSQL database tables and offers two ways of storing data. It can either use the file system on the server or use SRB (Storage Resource Broker). The application layer supports OAI-PMH and the toolkit is used for many local collections. It was therefore a good toolkit to support.

3.3.2 EPRINTS

EPRINTS is a long-standing IR (institutional repository) software package created by the University of Southampton, more than 20 years ago [16]. It is open source and configurable. For this reason, smaller institutions prefer it. The initial setup, configuration and individual item addition is easy. It is recommended for first time repository creators and used by well-known institutions. However, it is difficult to move a preexisting repository to EPRINTS. The repository also supports the OAI protocol. Since these repositories support OAI-PMH and typically have fewer items, it was a perfect toolkit to support.

3.3.3 Simple DL

Simple DL takes an unconventional approach to building a digital library. The most successful implementations arise from organizations and universities that were well funded either by companies such as HP or the Mellon Foundation [17]. However, many poorer countries and unfunded universities and institutions lack the resources to build, implement or manage a digital library. While some implementations have been made, improper model use and a software failure (middleware) can result in the loss of a complete archive. Simple DL tries to address this gap by providing a practical toolkit enabling long term access to digital libraries even when active preservation is no longer applied. It is able to do this because Simple DL does not implement the traditional backend database and database management system; it stores unstructured data as flat files and structured data as XML [18]. The data is therefore easily distributed and viewed on many devices, however, still provides a basic Web application to display collections, which can be customized using CSS and XSLT. The collection is able to keep this static form due to Simple DL requiring data to be pre-processed.
Information retrieval is supported by a tf.idf (term frequency-inverse document frequency) search system in a Web application with JavaScript. Results indicated an adequate response time for less than one hundred thousand items [17].

Simple DL was expanded on in the archive ingestor and user interface parts of the project. It was therefore imperative to know that the archive collector needed to pass flat files to the ingestor as Simple DL does not use a traditional database.

4 RESEARCH QUESTION AND FORMULATION

4.1 Formulation

From our research and background work, we could infer that current digital libraries are complex structures. They are difficult to create, port, duplicate and maintain, especially in low resource environments. Preservation therefore falls on larger institutions and organizations, but this in itself raises questions of validity, security and long-term persistence. While Simple DL tries to address some of these concerns, it is difficult for current archive administrators to easily extract, duplicate and re-model their archive to create redundancy, and share archives where there is limited network access. We believe an Archive of Archives, that is able to extract archives created using popular archiving toolkits, with the background simplicity of Simple DL, could address these issues.

4.2 RESEARCH QUESTION

The research question is: “To what degree can the data of archives, created using popular archiving tools, be scraped to produce Archive Snapshots?”

4.3 Methodology

To test our research question, a system to extract data from online archives generated by specific archiving toolkits, more specifically, a custom Web scraper, was developed. A pilot questionnaire to gather results around the snapshot was used; followed by a second questionnaire. The questionnaires comprised participant background and task related questions to visual comparisons and functionality, conducted by the participants. More data was then extracted by comparing quantitative aspects of the extracted archive snapshots to the original archives. The data was then analyzed and split into suitable categories measuring specific degrees.

5 SYSTEM DEVELOPMENT AND IMPLEMENTATION

5.1 Development Methodology

An agile methodology was used while implementing the archive Web scraper. More specifically the FDD (Feature Driven Development) model was loosely used. This was best suited as FDD uses short iterations during development, that are highly

adaptable and further allows the development of frequent tangible results **[19]**. These frequent iterations coupled well with visual and functional inspection of the scraper's output - the archive snapshot - at every weekly meeting. Creating feedback for the next iteration of overall model development and feature list building, imperative to the FDD methodology.

It was found that a simpler, complete system, provided valuable feedback when trying to introduce more complex features into the scraper. For instance, it would be difficult to know if the scraper had detected all the images and image types on a page, until the page has been scraped and preserved, such that a snapshot could be opened and visually compared to the original archive page.

This was conducted on a weekly basis to align with the team and supervisor meeting. This allowed for feedback from the team and our supervisor around current issues and ensured output would integrate with the complete archiving system.

5.2 Python

We decided to use the Python programming language for implementing our system. It is suitable as Python is recognized as a useful text processer when crawling the Web [20]. String manipulation is supported and easy to work with using the Python Strings class. This was necessary and frequently used when working with URLs (Uniform Resource Locator), file, and folder names. Python also supports Object Oriented programming, a feature that was used in the design of the system. This made for code reuse between the Web scraper and metadata harvester, a separate system, and allows for system expansion in the future; for instance, supporting other archive types.

Python is also oriented towards smaller prototypes, often using fewer lines of code [20]. This perfectly complements an experimental project such as this. The language also includes many well documented libraries that we took advantage of. This ensured faster, more robust prototypes throughout development.

5.3 Libraries

The Beautiful Soup library was used as an HTML parser as it provided an idiomatic way of searching for specific tags and later modifying tags to conform to their new relative paths [21].

The Requests library was used to make HTTP requests to archive servers. A Response object is created by this library when a call is made. These objects contain the HTML or resource file, as well as details pertaining to the request.

The Python Standard Library, regular expression library, was used in identifying HTML, HTM or links pointing to resources such as PDF, image, CSS, and JavaScript files.

The Python Standard Library, "urllib.parse" library, was used to split URLs into their components. These components were then used for changing URLs from absolute to relative paths, checking if the URL has the same domain, removing fragments and limiting queries in the early stages of development.

The Python Standard Library, OS join, was used to concatenate local links, found in HTML pages, and their parents' domain. This was required so that an absolute link was formed, allowing a request to be made.

The Python Standard Library, threading library, was used to process the URLs relating to HTML files and their resources concurrently. This is relevant as these are reliant on many I/O operations, allowing there to be multiple requests to the archive server at once. Speedup is achieved as the processor does not have to wait for a server response, before making another request.

5.4 Handling static and dynamic sites:

Scraper development started on a static site. This reduced the complexity of the scrape and allowed us to focus on URL discovery. URLs were simplified to only include their scheme, netloc and path, as defined by the "urllib.parse" library, of the Python Standard Library. These parts are indicated in red in figure 2.

```
scheme://netloc/path;parameters?query#fragment
```

Figure 2: URL structure

When satisfactory snapshots were produced, visually and functionally, development changed focused on dynamically generated sites.

When running the scraper on archives generated by a DSPACE toolkit, which creates dynamically generated sites, very few archive items were initially fetched. We reintroduced all queries, represented by the green text in Figure 2, but discovered that this was not viable, even on smaller collections. This URL part, known as the query string, is used to pass parameters to the archive server when a user makes a request, which then dynamically generates a static HTML page using these parameters and returns the page.

Completely scraping and preserving these dynamically generated sites as snapshots, would require the scraper to consider every possible query string combination, as multiple query parameters can be passed to the server at a time. This was not viable for this project, due to the significant amount of server requests this would introduce. For example, some DSPACE archives use as many as four parameters to browse by a single subject.

Although it should theoretically be possible to do this, considering the archives tested on, bandwidth limitations and our research question. It was decided to focus on only including query strings that led to item discovery and more general browsing functionality.

5.5 Objects/Implementation

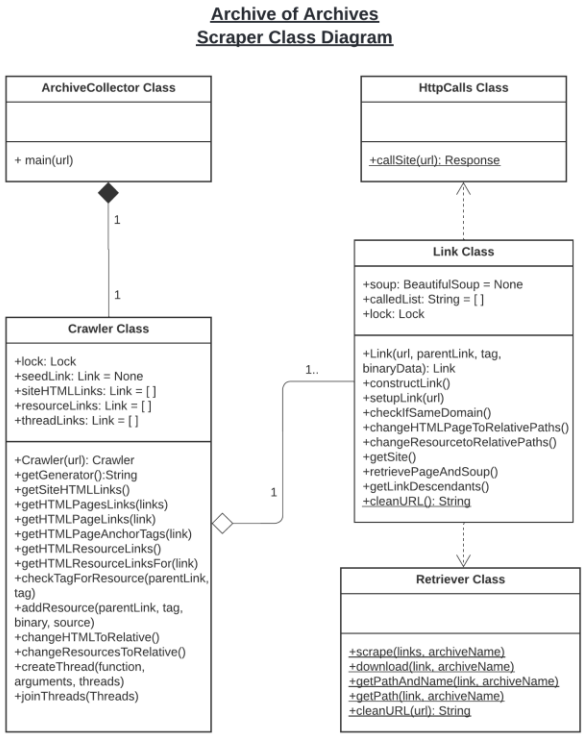

Figure 3: Class diagram

The scraper was implemented with five main classes: the ArchiveCollector, Crawler, Link, HTTPCalls and Retriever. These are represented in Figure 3.

The ArchiveCollector contains the main method and is the only class that needs to be used by the user. It should be invoked with an archives URL, as a parameter. The class ensures the correct sequential calls to the Crawler to scrape an archive website. These sequential calls can be seen in Figure 4.

The Crawler class handles the HTML and resource discovery of websites. The Crawler starts by making a seed Link object, with the URL provided by the user. The HTML header is checked for an EPRINTS or DSPACE generator tag. If this is found, the anchor tags pointing to HTML pages are extracted by a Beautiful Soup function call. Link objects are created for valid tags and saved in an array. These act as the seed links for the scraper.

Threads are spawned to further find HTML tags in the seed Link objects. Link objects are created for these tags and saved in a shared array. Once all threads have returned, the shared array is iteratively used as new seed links, until no new tags are discovered. Since the array size is known, the Links can be evenly distributed to a hardcoded maximum number of threads. A maximum was implemented after testing resulted in a server becoming unresponsive. Twenty to fifty threads seemed to yield the best results.

We chose this method as the scraped depth/iteration count, could be limited in the early stages of development, essentially completing smaller scrapes. This provided many benefits such as: catching bugs early, faster scrapes, low bandwidth usage and fewer requests to the archive server. We found that this provided instrumental value in our agile methodology, as inspection of the complete scrape or snapshot, was important in assessing which aspects needed focus next.

The next part is collecting all the resources on the site. The array of Link objects, pointing to HTML pages are split amongst threads. All HTML tags are searched for URLs pointing towards resources, such as JavaScript, CSS, PDFs, images, and videos. Link objects are created for each new resource and saved in an array.

The Crawler is then called to change Link objects, pointing towards web paths, to their new relative paths. This ensures that many visual characteristics are preserved, JavaScript functionality is present, hyperlinks are functional and resources such as PDFs and images are downloadable in the snapshot.

Finally, the Crawler is called to save the HTML and then the resources. It uses the static Retriever class, that takes an array of Link objects as input, to do this.

The Link class is responsible for representing a URL link and its surrounding functionality. Link objects are created from a parent Link object and Beautiful Soup Tag, which represents an HTML tag and its values. During Link creation, all URL links are transformed to absolute URLs. Links are also checked to see if their queries match the supported list. If this is not the case, all queries are stripped. It was necessary to include a subset of queries, to ensure all items on DSPACE toolkit generated archive sites were discovered.

During the creation of these Link objects, the class also holds the responsibility of ensuring that links relating to HTML pages, outside of the root domain, are not created and therefore not added. This ensures that the Crawler stays on the archive site. An exception is applied to resources as they might exist outside of the domain.

Once an absolute URL has been established in the Link object, a check is made to see if the URL matches a list of excluded URL's, such as CGI (Common Gateway Interface) calls - which are used for dynamic server responses and incur heavy overhead on servers as a new subprocess is created with each call - and if the URL has previously been called, before making a request to the archive server. If it has previously been called, a called index integer is returned to the Crawler; this ensures identical resources are not requested more than once. The Crawler handles this by recording that the URL occurs more than once. It adds the Beautiful Soup tag, where the URL originated from, to the Link Object representing this URL. Link objects support an array of tags. This enables a record to be kept of all the occurrences, and the ability to modify all the URLs at once, by modifying these tags. This functionality is used when the Crawler calls the change to relative URLs, so that the HTML can reference the newly formatted files that they will be saved under.

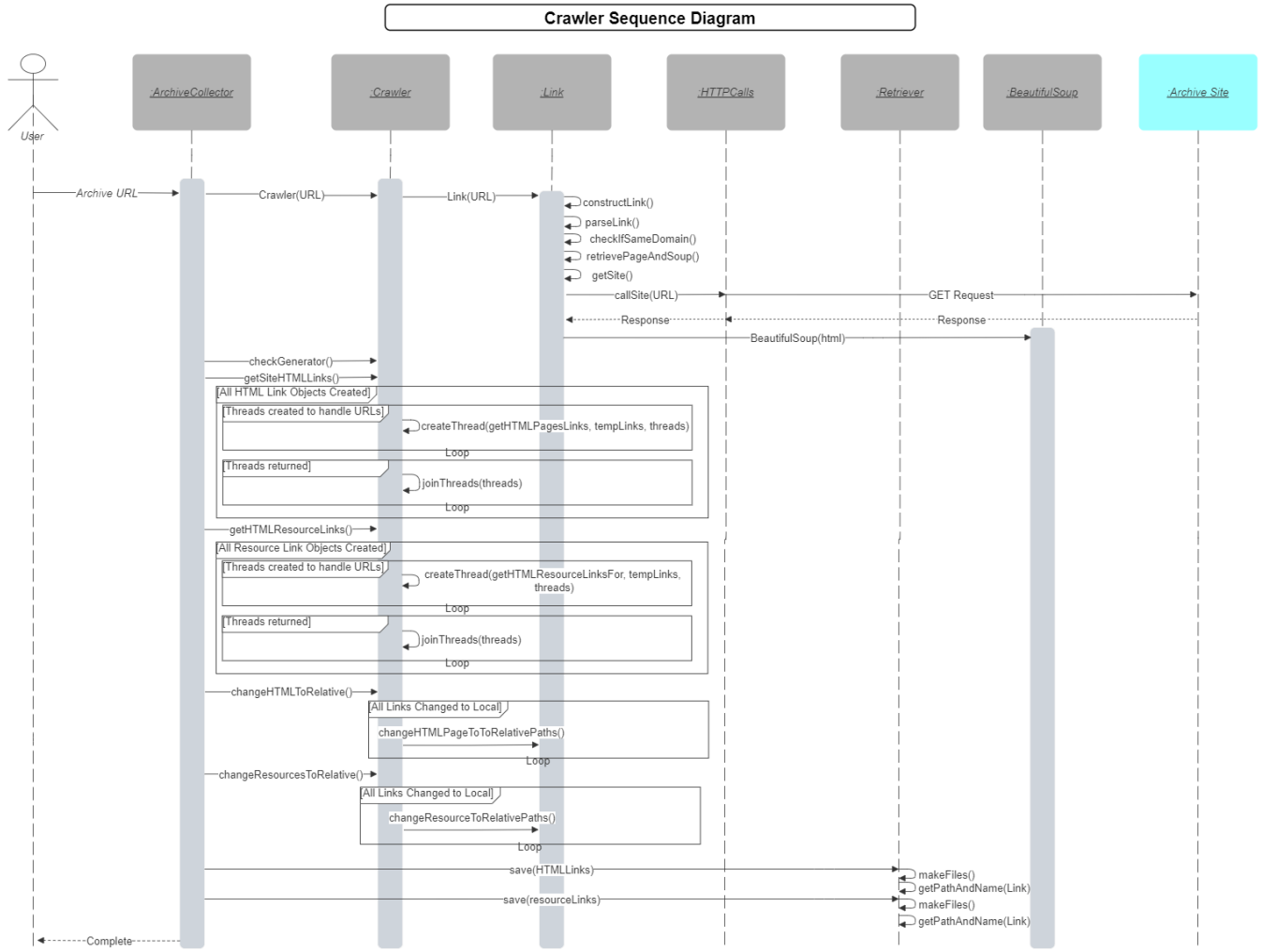

Figure 4: Sequence diagram

The last stage of the Link object creation is the HTTP GET request to the archive server. If the request is successful, a Response object, containing the HTML or resource, is added to the Link object. Further, if the object is HTML, a Beautiful Soup object is created for it, this object has methods to easily parse and modify the HTML.

Only once the complete site is crawled, will the ArchiveCollector invoke a call to save the files. This is handled by the Retriever class. The Retriever is a static class able to save an array of Link objects. The Retriever creates the relevant paths for saving the resources. The file path is created from the current working directory, a hardcoded structure and the original folder path and name. The name is also decoded to replace the percentage encoding, present in many URLs, which is necessary when saving and reopening files in a browser, to avoid incorrect path direction. The HTML files are separated in their own folder. This help asses scrape depth and number of objects found by the scraper. The paths and filenames are extracted from the Link object to ensure that the HTML path matches.

5.6 Performance

Archives often contain thousands of items. It is therefore expected that scraping these larger collections could take a considerable amount of time, especially if they contain items with large file sizes such as videos. We therefore structured the design of the application to support and use threading. Besides the initial URL provided by the user, which creates initial seed URLs, the request to HTML webpages and their resources is threaded.

This enabled a speedup as the server requests are not a CPU bound task. When using the threading library in Python, CPU bound tasks take are performance hit from the Python Global Interpreter Lock, that only allows one thread to control the interpreter at a time. It was determined that we would not need to use Python's Multiprocessing Library, which uses multiple interpreters to run threads in true parallel form, as the speedup would be negligible to server request times and problems with overloading the archive server with requests. This is suited to the main use case, where once an archive has been entered, archive scrapes will not be manually called, but happen automatically at set time intervals.

6 EXPERIMENT DESIGN AND EXECUTION

We set up an experiment to help answer our research question. “To what degree can the data of archives, created using popular archiving tools, be scraped to produce Archive Snapshots?” We decided to use a questionnaire to gather data that we could analyze and draw conclusions from.

6.1 Participants

Participants in our study included current University of Cape Town students, postgraduate students, past students, and regular Web users. Students were from different faculties and years. and job titles. This provided a wide range of users.

6.2 Research Measures

Two forms of data were recorded by our questionnaire: quantitative and qualitative data. To measure a participant’s level of agreement with a question/statement, a symmetric Likert Scale, with a scale of one to five was used. This provides participant’s the choice of neutrality in their response and does not force choice where there is no perceived value added **[22]**. The range was kept small. This provides for two choices on either side of agreement, allowing participants to note that they might not fully be in agreement, while choosing a side. For an initial experiment, finer granularity was not necessary.

Space was included for qualitative written responses from participants, to record feedback that could not be preempted or quantified. These responses were grouped and categorized where necessary in results and analysis.

The questionnaire was created around three archive snapshots. Two snapshots were created from the EPRINTS toolkit and one from the DSPACE toolkit. This allows us to have inter archive snapshot analysis between DSPACE and EPRINTS archives and intra archive analysis for EPRINTS archives.

6.3 Question Formulation

The questionnaire background questions were related to participants’ familiarity with websites and online archives. This was included so that correlations could be analyzed later if necessary.

Considering our research question, questions were formulated around three categories we decided would be of interest. To measure the degree of scraping and preserving an archive website, we used visual accuracy, functional accuracy, and an overarching category of usability.

Visual accuracy was tested by asking participants if the snapshot and original archive were visually identical. Three pages on each snapshot were chosen. The archive homepage was chosen because they provide a centre point from where users navigate the archive and often include many visual resources. An archive entry was chosen as these pages display important metadata to users and are used to access items, forming part of an archives main use case. Lastly, a dynamically generated page and a relevant static counterpart, on static sites, was used for the third comparison.

Functional accuracy was tested by asking participants if they are able to complete a set of tasks. The first task was to use the search feature of the snapshot to find papers by a specific author. This task was chosen as the snapshot does not have a server that it can make requests to. The second task is to browse the snapshot by a year or issue date. On DSPACE archives, these are dynamic requests. The third question is to open an archive snapshot entry and download/view the item. This was asked to test if the snapshot could reach the depth of entries.

Lastly, usability was tested. Since participants had visually inspected and completed tasks with the snapshot, it was possible to ask questions pertaining to their perceived experience. After a pilot questionnaire and interacting with participants, we felt that there was some ambiguity in these questions. We therefore replaced these with tasks. The tasks were formulated around finding specific papers on the snapshot, provided fewer details each time.

6.4 Test procedure

For participants to conduct comparisons between the archive snapshots and original archive website, a laptop with two browsers instances open simultaneously was used. This allowed all pages used in the comparison to be opened in tabs. One browser for all snapshot pages and all related original archive pages in the other, in order of tested comparisons. To help the participants identify the snapshot, the snapshots HTML title tag was renamed and included with the relative comparison question.

Pictures were first considered for comparisons; this would have allowed for the questionnaire to be conducted in a simpler fashion and possibly remotely. However, scaling the webpage to create an image resulted in the page being difficult to see.

The questionnaires were therefore conducted in person, with questionnaires in a printed format. Allowed participants to visually inspect and navigate the archives while recording their results.

6.5 Pilot Study

While conducting the questionnaire, which was done in person, as participants were helped navigating back and forth between snapshot and the original archives, problems were noticed with a few of our questions. Questions aimed at trying to capture if there was any importance in the way that the snapshot saved the original archive and displayed it were misinterpreted. It was instead interpreted that participants were rather analyzing the usability of the original archive interface, which is not part of our research. We therefore decided to use these initial responses as our pilot study, remove these questions, adapt the questionnaire, and continue our research.

6.6 Ethical Issues

Ethical clearance to conduct our research was applied for to the Faculty of Science Research Ethics Committee. Our approval code is: FSREC 056-2022. Thereafter research access to University of Cape Town students by application to the Executive Director of the Department of Student Affairs was acquired. All participants were first required to sign our voluntary consent form before being allowed to participate in the survey. This condition was also noted at the top of the questionnaire.

Verbal permission to use UCT's Computer Science Research Document Archive and written permission to use The Academy of Science of South Africa Archive throughout this project was also acquired. Archives built using EPRINTS and DSPACE toolkits respectively. A smaller dummy DSPACE repository was also built for demonstration purposes.

Throughout our results, no linking personal details of participants were used, and author and item details used in our questionnaire are omitted.

7 RESULTS AND ANALYSIS

Our questionnaire had a total of 21 responses. Eleven participants were a part of the pilot study. All results pertain to 21 participants unless indicated otherwise.

7.1 Participant Background

Our participants were 52% University of Cape Town students, presently or within the last five years. Ninety percent of participants indicated that they interacted with websites daily and 10% weekly. Further, 9% used a form of online archive daily, 29% weekly, 42% monthly and 20% indicated that they never use them.

7.2 Visual accuracy of snapshots

The visual accuracy, of toolkit specific archive snapshots, was tested by visual comparison conducted by participants. Three separate pages were tested: the homepage, an item entry page with its metadata and the advanced search page in EPRINTS snapshots and community list page in DSPACE snapshots.

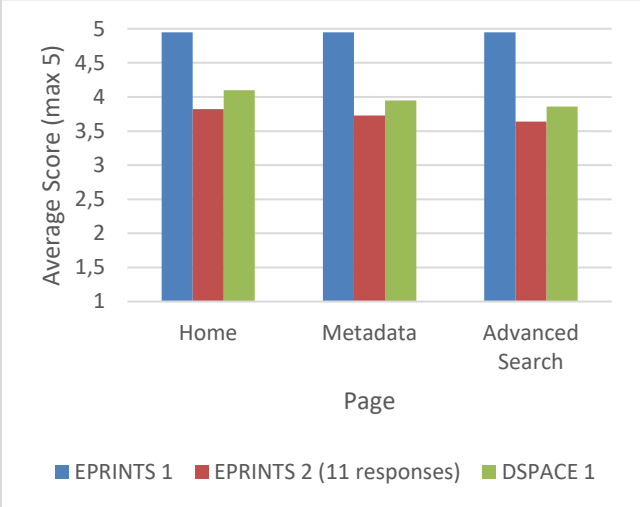

Figure 5: Visual accuracy graph of archive snapshots

Overall, we had very good results with visual accuracy. All three pages evaluated, as can be seen in Figure 5 above, scored above 3.5 on average (higher is better). "EPRINTS 1", had the best result of all three snapshots, averaging a higher score, on all three pages, followed by the "DSPACE 1" and "EPRINTS 2" snapshots. The question asked if the page in question was visually identical to the original archive page.

Participants noted the follow discrepancies between the pages, in Table 1.

Table 1: Percentage of participants that noted a discrepancy

	EPRINTS 1	EPRINTS 2 (11 responses)	DSPACE 1
Search bar placement	N/A	90,91%	N/A
Collections Expand Icon Missing	N/A	N/A	66,67%
Search Glass Icon Missing	N/A	N/A	71,43%
Home Icon Missing	N/A	N/A	85,71%
View /Dow n laod Icon Missing	N/A	N/A	4,76%

The most discrepancies occurred in the "DSPACE 1" snapshot. All four were related to icons. Upon further investigation, after testing, we discovered that these are a part of the Glyphicon Halflings component set. It appears these are generated from a class that extracts the icon from a server; this type of generation is most likely why they are not preserved. We did not consider this case during development. A trend also appears where more participants noticed discrepancies located towards the top of the page or that because these features are used more often, they are placed higher up and therefore their difference was noted more frequently.

"EPRINTS 2" was also noted to have a search bar discrepancy. Although the components are still visually identical, the placement of the search bar, search button and dropdown, to select whether searching between title, author, or date, had shifted. These components shifted underneath the "browse by" buttons which they are adjacent to in the original archive. In comparison to the "EPRINTS 1" snapshot, although both toolkits support similar components, a visual discrepancy was not recorded by any participant.

Despite these discrepancies, the "EPRINTS 2" and "DSPACE 1" pages averaged between 3.5 and 4 out of 5 on the Likert Scale, indicating that these discrepancies have minimal visual impact. This is strengthened by the fact that all discrepancies in the "DSPACE 1" snapshot were missing small component icons. Followed by the slightly lower averaging "EPRINTS 2" snapshot, where a shift in larger components, such as the search bar and button would be more visually striking, explaining the lower score around 3.7.

Another visually apparent result is seen in the consistency between the three page's results. This is highlighted by the apparent visual trend in Figure 5, where each snapshot's average score is consistent per page, indicating consistent or consistently weighted discrepancies per snapshot page. It can therefore be

reasoned that most pages in the snapshot will follow the same trend, as the tested pages were selected to give a fair and broad representation of the snapshot.

7.3 Functionality of snapshots

The preserved functionality was measured by asking participants to complete a set of tasks. The tasks were chosen to simulate the sequential actions that an archive user would take. These included searching for an author, browsing by a category, viewing an entry and its metadata, and finally downloading/viewing an item. Participants were helped if necessary to ensure that interface, visual preservation, or instruction clarity would not affect the results. The tasks were designed to test preserved functionality over the archives interface design. Functionality was tested on a yes/no ability.

Table 2: Percentage of tasks completed

Task	EPRINTS 1	EPRINTS 2 (11 responses)	DSPACE 1
Search	0%	0%	0%
Browse by year/collection	100%	100%	100%
Browse by author (DSPACE)	N/A	N/A	10%
Item entry	100%	100%	100%
Download item	100%	100%	100%

The functionality showed very strong results as indicated by Table 2 above. None of the snapshots' search functionality could be searched for a specific author. Either a "file not found" error or "no internet connection" appeared. This functionality could not be preserved as it relies on a request to the original archive server, which processes and returns results for the provided search terms. Since the snapshot encompasses only local files, this request is no longer possible.

Browsing the snapshots by year/collection and item entry was 100% successful. We were able to preserve this functionality, as the functionality stems from the user requesting a pre-generated HTML file on both DSPACE and EPRINTS sites. This request can be anticipated and accounted for by the custom crawler, when crawling the original archive site, as they are encompassed in the HTML files.

Browsing the DSPACE snapshot by a specific author was 10% successful. We expected 0%. This error can be explained by there being a visual change - loading of another page - when participants selected an author's name. However, care was not taken to ensure that they were indeed seeing the correct page. The reason that the hyperlink directed the participants to an incorrect link was due to the dynamic nature of the original archive website. The site uses URL queries to assign values as parameters when making a request to its server. During implementation, we only implemented a selective set of queries; this link was not one of them.

Downloading an item on all three snapshots was also 100% successful. This functionality was preserved for PDF and image items. During implementation, more items, such as 3GP, MOV and video formats were found. Although we detected and preserved these initially, this functionality was later removed for testing as most items appeared to be PDFs and therefore consumed unnecessary bandwidth. These papers were however all pre-selected. We therefore do not know the extent of items collected but are however sure that the scraper could reach as far down as to retrieve items.

7.4 Snapshot usability

To assess the usability of the snapshot, we included questions after the participants had a chance to view and interact with the snapshot.

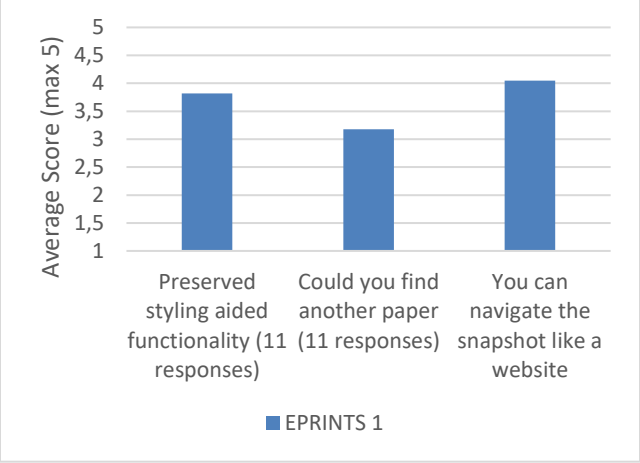

Figure 6: EPRINTS 1 usability scores

Participants were asked if they could navigate the snapshot like they would a website. This averaged a strong score over four, as seen in Figure 6, indicating that the combination of the visual and functional aspects produced a snapshot that replicated a website. Participants were also asked if they could find another paper if tasked to. Results indicated a mixed response, averaging a score close to three, indicating that participants were unsure if this was possible. This intermediate score therefore likely stems from the only missing functionality indicated - searching. Revealing that this functionality is very important when a participant interacts with these archives, which is relevant to our participants as 80% indicated that they interact with a form of repository at least monthly.

Participants were asked to find three papers, with fewer details provided per paper. The results can be seen in Figure 7. Papers one and two were found 100% of the time. Both papers were provided with either their entry date or subject. The snapshot could further be browsed by these categories. All participants found these two papers using this functionality. Ninety percent of the participants tried using the search functionality first, which

was not successful. Strengthening our earlier views around participants' dependency on this feature.

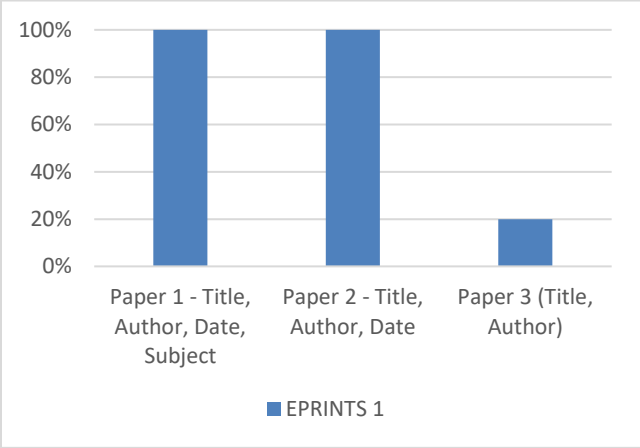

Figure 7: Percentage of papers found – EPRINTS 1

Paper three was found by 20% of the participants. We expected this question to have a 0% success rate. Participants were explicitly given the title and author of the paper, however in error, it was stated in the question that this was a conference paper. The participants who found the paper used this information to search the archive by paper "type" and since the archive was relatively small, they found the paper. The rest of the participants were stopped after three attempts at finding the paper.

It can therefore be seen that even if we know that a paper exists within the archive, it still cannot be found within an acceptable amount of time or given very specific details of the paper – such as the specific subject the paper was stored under. Although, being able to browse a snapshot by year and subject for example, which was preserved in EPRINTS snapshots and to an extent in DSPACE snapshots, could be a valid use case for some archive users.

8 CONCLUSIONS

This project aimed to see the degree that archives, built using popular archiving toolkits, could be preserved. The aim was to test this by building a custom Web scraper that specifically targeted EPRINTS and DSPACE generated sites. The scraper was built to crawl these archives and create a snapshot of the archive. Our questionnaire to evaluate these snapshots targeted three specific aspects: visual accuracy, functionality, and usability.

Our results indicated the ability to capture almost all visual components and their layout from the original archives. Discrepancies included missing or changed icons and a shift in a few components. These did not however affect any functionality that was tested, and results showed that users still perceived the snapshot as almost identical to the archive. The results are therefore pleasing, and I conclude that the snapshot can visually represent the archive.

All snapshots were unable to preserve the search functionality when creating a snapshot; we did not expect to be able to. Snapshots created for EPRINTS generated archives retained all other tested functionality. DSPACE snapshots were only able to preserve more general browse functionality. Trying to preserve more functionality exponentially increased the number of requests to the archive server, either causing long crawl times or overloading the archive server. DSPACE snapshots should theoretically be able to be preserved to the same extent as we were able to handle their dynamic aspects but were limited when testing with live archives. It can therefore be concluded that EPRINTS generated archives can be functionally preserved as far as possible, while DSPACE archives could only be functionally preserved to all static pages and a select set of simpler dynamic pages.

Snapshot usability is an imperative part of this project. It forms part of the overarching project of "Archiving Archives". It helps answer the degree of preservation and if an archive snapshot is an appropriate form of preserving an archive. Compared to something simpler, such as a metadata harvester, that also extracted archive items. Participants were in strong agreement that they could navigate the snapshot like a website and could find specific papers by browsing the snapshot, if provided with at least one category under which the item was saved. Results indicated that participants preferred to use the search feature as their first attempt and could browse for items by category. We therefore conclude that the preserved snapshots provided inadequate usability to our participants in finding specific papers, however, can support other use cases such as browse by category. Therefore, the snapshot usability is use case dependent and would differ between users.

9 FUTURE WORK

Future work lies in the outcome and further testing of the integrated project. This could be conducted with on a test scale only including well-functioning EPRINTS snapshots. The outcome of this would determine a better understanding of the need for a detailed snapshot, that is currently being created and preserved, provided the context of additional functionality the overall archive would have – such as a global archive indexer. From this point it can be determined if further work should be placed in preserving more functionality in dynamic archives and if more snapshot toolkits should be supported. This additional support can easily be implemented due to the development principles and modularity provided by the principles followed when developing the custom scraper.

10 REFERENCES

[1] A. Kumar, R. Saigal, R. Chavez and N. Schwertner, "Architecting an Extensible Digital Repository," Proceedings of the 2004 Joint ACM/IEEE Conference on Digital Libraries. DOI: 10.1109/JCDL.2004.239994, September 2004. Available: https://dca.lib.tufts.edu/dl/publications/pubs/jcdl2004-tufts.pdf.

[2] T. Owens, "The Theory and Craft of Digital Preservation," Johns Hopkins University Press. DOI: 10.5860/rbm.20.2.119, 2018. Available: https://osf.io/preprints/lissa/5cpjt/download.

[3] D. Gerrard, J. Mooney and D. Thompson, "Digital Preservation at Big Data Scales: Proposing a step-change in preservation system architectures," DOI: 10.1108/LHT-06-2017-0122, 2017. Available: https://doi.org/10.1108/LHT-06-2017-0122.

[4] V. Ambati, N. Balakrishnan, R. Reddy, L. Pratha and C. V. Jawahar, "The Digital Library of India Project," 2006. [Online]. Available: http://www.cs.cmu.edu/~vamshi/publications/vamshi_icdl2006.pdf.

[5] I. Carbajal and M. Caswell, "Critical Digital Archives: A Review from Archival Studies," The American Historical Review, Volume 126, pp 1101-1120. DOI: 10.24242/jclis.v1i2.50, 2021. Available: https://doi.org/10.1093/ahr/rhab359.

[6] R. Jantz and M. Giarlo, "Digital Preservation," Architecture and Technology for Trusted Digital Repositories. Volume 11 No. 6. DOI: 10.1515/MFIR.2005.135, 2005.

[7] H. Suleman, "Simple DL: A toolkit to create simple digital libraries," University of Cape Town, South Africa, 2021.

[8] D. Koutsomitropoulos, A. Tsakou, D. Tsolis and T. Papatheodorou, "TOWARDS THE DEVELOPMENT OF A GENERAL-PURPOSE DIGITAL REPOSITORY," High Performance Information Systems Laboratory, Department of Computer Engineering and Informatics, 2004. Available: https://www.scitepress.org/papers/2004/26374/26374.pdf.

[9] R. Pandey, "Digital Library Architecture," Indian Statistical Institute, March 2003. Available: http://dlissu.pbworks.com/w/file/fetch/44829234/B_architecture.pdf.

[10] H. Suleman, "Identifiers and Repositories (Slides)," 2006. Available: http://www.husseinsspace.com/teaching/uct/2006/csc4000w_2006_ii/csc4000w_2006_ii_notes/csc4000w_2006_ii_notes_5.pdf.

[11] M. Nelson and H. Van de Sompel, "A 25 Year Retrospective on D-Lib Magazine," 2020. Available: https://arxiv.org/pdf/2008.11680.pdf.

[12] R. Kahn and R. Wilensky, "A framework for distributed digital object services," DOI: 10.1007/s00799-005-0128-x, 2006. Available: http://doi.info/topics/2006_05_02_Kahn_Framework.pdf.

[13] R. Gartner, "Metadata for digital libraries: state of the art and future directions," JISC, 2008. Available: https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.183.3762&rep=rep1&type=pdf.

[14] H. Sastry and L. Reddy, "Digital Repository Software Packages: An extended architecture for image handling in open source packages," 2010. Available: https://www.researchgate.net/publication/279466681.

[15] M. Smith, M. Bass, G. McClellan, R. Tansley, M. Barton, M. Branschofsky, D. Stuve and J. Walker, "DSpace: An Open Source Dynamic Digital Repository," D-Lib Magazine. DOI: 10.1045/january2003-smith, 2003. Available: https://dspace.mit.edu/bitstream/handle/1721.1/29465/D-Lib%20article%20January%202003.htm?sequence=1.

[16] M. Beazley, "Eprints Institutional Repository Software: A Review," Partnership: The Canadian Journal of Library and Information Practice and Research. DOI: 10.21083/partnership.v5i2.1234, 2011. Available: DOI: doi:10.21083/partnership.v5i2.1234.

[17] H. Suleman, "Reflections on Design Principles for a Digital Repository in a Low Resource Environment," University of Cape Town, South Africa, 2019. Available: https://pubs.cs.uct.ac.za/id/eprint/1331/1/ho_2019_lowresource.pdf.

[18] H. Suleman, "Simple DL: A toolkit to create simple digital libraries," University of Cape Town, South Africa, 2021. Available: https://pubs.cs.uct.ac.za/id/eprint/1512/1/paper_88.pdf.

[19] F. Anwer, S. Aftab, U. Waheed and S. Muhammad, "Agile Software Development Models TDD, FDD, DSDM, and Crystal Methods: A Survey," INTERNATIONAL JOURNAL OF MULTIDISCIPLINARY SCIENCES AND ENGINEERING, March 2017. Available: http://www.ijmse.org/Volume8/Issue2/paper1.pdf.

[20] J. Thomas, "University of Arizona Department of Mathematics," Available: https://www.math.arizona.edu/~swig/documentation/python/handout.pdf.

[21] "Beautiful Soup Documentation," Available: https://beautiful-soup-4.readthedocs.io/en/latest/.

[22] A. Joshi, S. Kale, S. Chandel and D. Pal, "British Journal of Applied Science & Technology," Likert Scale: Explored and Explained. DOI: 10.9734/BJAST/2015/14975, Available: https://eclass.aspete.gr/modules/document/file.php/EPPAIK269/5a7cc366dd963113c6923ac4a73c3286ab22.pdf.

11 Appendix

Visual Comparisons			
EPRINTS Archive 1:			
Page	Participants	Total Score	Average
Homepage	21	104	4,95
Metadata	21	104	4,95
Advanced Search	21	104	4,95

Functionality			
EPRINTS Archive 1:			
Task	Participants	Yes	No
Search	21	0	21
Browse by year	21	21	0
View metadata	21	21	0
Download item	21	21	0

Usability			
EPRINTS Archive 1:			
Question	Participants	Total Score	Average
Styling aided functionality	11	42	3,82
Inherent link	11	36	3,27
Find another paper	11	35	3,18
Navigate like website	21	85	4,05

Visual Comparisons			
EPRINTS Archive 2:			
Page	Participants	Total Score	Average
Homepage	11	42	3,82
Metadata	11	41	3,73
Advanced Search	11	40	3,64

Functionality			
EPRINTS Archive 2:			
Task	Participants	Yes	No
Search	11	0	11
Browse by year	11	11	0
View metadata	11	11	0
View item	11	11	0

Usability			
EPRINTS Archive 2:			
Question	Participants	Total Score	Average
Styling aided functionality	11	43	4
Inherent link	11	38	3
Find another paper	11	34	3
Navigate like website	11	46	4

Visual Comparisons			
DSPACE Archive 1:			
Page	Participants	Total Score	Average
Homepage	21	86	4,10
Metadata	21	83	3,95
Advanced Search	21	81	3,86

Functionality DSPACE Archive 1:			
Task	**Participants**	**Yes**	**No**
Search	21	0	21
Browse by issue date	21	21	0
Browse by author	21	2	19
View metadata	21	21	0
View item	21	21	0

Usability DSPACE Archive 1:			
Question	**Participants**	**Total Score**	**Average**
Styling aided functionality	11	42	4
Inherent link	11	36	3
Find another paper	11	34	3
Navigate like website	11	45	4