



UNIVERSITY OF CAPE TOWN



DEPARTMENT OF COMPUTER SCIENCE

# CS/IT Honours Project Final Paper 2022

---

**Title:** Simple Archive Management

**Author:** Kamilah Louw

**Project Abbreviation:** ARCHMAN

**Supervisor(s):** Professor Hussein Suleman

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

# The Simple Archive Management Project

Kamilah Louw  
Department of Computer Science  
University of Cape Town  
Cape Town  
LWXKAM003@myuct.ac.za

## ABSTRACT

Simple DL is a digital archiving toolkit. It was built to enable users to build and maintain their own digital archives without prior technological expertise. The current administrative interface for file and data management is not usable. Since this is where all the resources of the repository are managed, it is important that the interface is intuitive to simplify tasks, speed up processes, and allow the management of these archives to be a smoother process. The solution is to develop a new usable interface to address the key usability challenges while expanding the capabilities of the current archival system. This project aims to improve the administrative system's file and data management. It includes essential tasks like uploading, deleting, creating, and downloading files and folders. The improved interface not only includes the core functionality but also additional features and improvements such as batch-uploading, downloading, deleting, renaming files and folders, accepting zipped files, and creating folders directly through the system. Through a cognitive walkthrough and using Nielsen's ten usability heuristics, the new interface proved to be more usable for users and intuitive.

## KEYWORDS

Digital libraries, Simple DL, User Testing, File and Data Management, Nielsen's Ten Usability Heuristics

## 1 Introduction

Digital archives or digital libraries (DL) are collections of data items. They are used to store digitized resources safely and conveniently [1]. The Simple DL toolkit was designed to be as simple as possible while allowing users to access the digital libraries even when there is a computer system or network failure. This software toolkit was designed to assist users in the creation of digital libraries [2]. The Simple Archive Management project, also known as the ARCHMAN project, involves the design and implementation of an improved, usable user interface for administrators of the Simple DL toolkit.

This project was proposed by the Emandulo staff. Emandulo is a project belonging to the Archives and Public Culture (APC) Center [2]. It provides a platform for the general public to browse through collections of artefacts that document the culture and

language of indigenous groups in South Africa. Simple DL was used in the creation of this repository. It enabled users to build and maintain the repository without prior technological expertise however, the administrative interface is not user-friendly. This project aims to design and build a usable interface with additional functionality to improve the usability of the system. The interface should be simple enough for anyone to use but include all the necessary functionality for the management of files and data.

The improved system was designed, prototyped, and improved through multiple iterations of user testing and development. Requirements gathering with Simple DL users and Emandulo staff highlighted the necessary improvements. Through a series of interviews and development cycles, an improved interface was developed. It is important that this interface is usable and simple. Having a system that could perform the basic functionality for the management of data and files in the repository is important as it improves productivity and simplifies the management of files and data which can be a very difficult task since these repositories can have multiple items. Improving the interface will simplify tasks, speed up processes, and allow the management of these archives to be a smoother process.

This project forms part of a larger project which was divided into three parts. The other two include the development of an interface for the configurations of the site and an interface for the editing of files. This project focuses on the development of a new, usable administrative interface for the file and data management of the system. The paper will be divided into sections including a short review of related work and some background information, the design and implementation of the new and improved system which will include the software development approach taken, the results and a discussion, and finally, conclusions and future work.

## 2 Related Work and Background

This section sheds light on some similar repository toolkits, highlights the importance and advantages of digital archives, discusses some familiar file and data management applications and their architecture, and provides an overview of Simple DL and the current system. Digital libraries not only allow resources to be accessed remotely but they also offer a safe place to store records that could have historical importance [1] [3]. Digital libraries can store various types of resources for long periods while reducing the chances of these electronic records being

stolen, lost, or possibly damaged. This can be advantageous for cases when the resources are rare, fragile, or treasured.

## 2.1 Repository Toolkits

Simple DL is not the first toolkit developed for creating repositories. Some other toolkits include DSpace, EPrints, and Greenstone. DSpace and EPrints are the most popular toolkits and are both web-based and have backend databases [2]. Greenstone requires the installation of software to access collections, but it was designed to work both offline and online.

*2.1.1 DSpace.* DSpace represents a collaborative initiative involving the Massachusetts Institute of Technology (MIT) Libraries and Hewlett-Packard (HP) labs [4]. Operating as an open-source platform, it serves as a digital repository designed for housing content generated by different organizations and institutions for research and educational objectives [5]. The communities within the system have a collection of related content that is grouped together. Members of the communities can upload content directly via a Web interface or perform batch imports through a batch importer. DSpace enables the capture of electronic material with corresponding metadata, allows for systematic listing of resources, and the preservation of digital data [6]. It also assists in searching with the use of keywords and metadata.

*2.1.2 EPrints.* EPrints was created by the University of Southampton in England [4]. It is a free software application that allows the academic community to submit preprints, post-prints, and other academic publications via a web interface. It uses conventional technology and runs entirely on Open-Source platforms. The administrator determines which metadata fields are stored for all resources and the administrative back end provides access to configuration options [7] [8]. EPrints makes use of MySQL, Oracle, and PostgreSQL for the database management system. EPrints does not support bulk uploading.

*2.1.3 Greenstone.* Greenstone is an open-source software that was distributed under the GNU (a free and open-source operating system) General Public License (GPL) that is often used to develop and publish repositories online [6]. Although Greenstone collections can be transferred on CDROM, accessing them requires the installation of software [2]. The University of Waikato's New Zealand Digital Library Project, UNESCO (United Nations Educational, Scientific and Cultural Organization), and the Human Info NGO worked together to build and distribute GSDL (Greenstone Digital Library) [9]. The software's purpose was to enable users to build their own digital libraries. The Greenstone software builds data structures for searching and browsing from the provided resources as opposed to relying on any manual work.

*2.1.4 Importance and Advantages.* Many advantages come with digital archives. First, they do not require physical space compared to a physical archive with physical files and documents,

and since all the resources are digitized, searching, browsing, organizing, and managing these resources can be done effortlessly [10]. Digital archives also allow for remote access at any time which means that resources can be accessed by multiple users since the distribution does not depend on the number of copies available [1]. Archives reduce the risk of wear and tear which consequently enables longer preservation and conservation since the original copy is never handled physically.

## 2.2 File and Data Management Applications

Management of resources in digital archives is important since these repositories could contain historically important documents. It is important that the structure of the archive is maintained, and that files and folders are organized correctly. File and data management of electronic resources has been made possible through various platforms. Some commonly known platforms include Google Drive [11], Dropbox [12], and OneDrive [13]. Each of these platforms includes different features but the basic functionality of storing and managing data is similar [3].

*2.2.1 Google Drive.* Google Drive allows users to store files in the cloud on Google's servers, share data, and sync files across devices [7]. It also enables users to sign into several accounts and switch between them with ease. Additionally, it offers shared disks for customers to collaborate.

*2.2.2 Dropbox.* Dropbox, like Google Drive, allows users to sync, share, and edit documents over the internet [7]. File sharing and updates are possible since users connect to secure servers. This also allows for ease of accessibility.

*2.2.3 OneDrive.* By downloading the application, users can access their own files offline [7]. Users can access their uploaded files from any device.

*2.2.4 File and Data Management Platform Architecture.* Each of the previously discussed applications are cloud storage systems. The basic services provided by each application are similar but differ in cost options for unlocking additional features and space [7]. The front end of these applications is responsible for user access to files. It provides the user with an interface to interact with. The back end oversees user information security and is utilized by service providers. It is also where the physical data storage is located and can either be an internal protocol that makes use of features or a traditional backend to physical disks [15]. The front end is connected to the backend via the internet. A cloud storage system requires a single data server that is connected to the internet [16]. A client transfers files to the data server through the internet, where the data is saved. When a user needs information, they connect to the data server via a web-based interface. The server either returns the files to the user or grants them access to manipulate the server's files.

## 2.3 Simple DL

Simple DL is a software toolkit that was designed to enable users to build their own digital archives without prior technological expertise. The toolkit was developed to be as simple as possible, to provide long-term access to digital libraries even in the absence of active preservation and in the event of a network or computer system failure [2]. Spreadsheets and XML (Extensible Markup Language) files are used to store metadata. All unstructured data is stored as flat files and structured data is stored as XML. There is no database and no database management system. Simple DL's primary operations are centered around ingesting a metadata collection and developing a website representation of the collection. This is done in three steps; import, index, and generate. Each time a new metadata sub-collection is added to the system, these three steps need to be invoked. Administrators manage files and data through the rudimentary Web interface shown in Figure 1.

to have a view of the file they are working with. Figure 1 displays the current interface of the Simple DL administration site.

## 3 System Design and Implementation

This section discusses the current functionality and user interface for the file and data management site of the Simple DL toolkit. It will also provide an overview of the software development approach taken and then dive into the finer details of each iteration of the implementation through a discussion of requirements gathering, user testing, and usability results.

### 3.1 Project Management

This project required the administrative site to be rebuilt. To separate tasks and complete the project, the system was divided

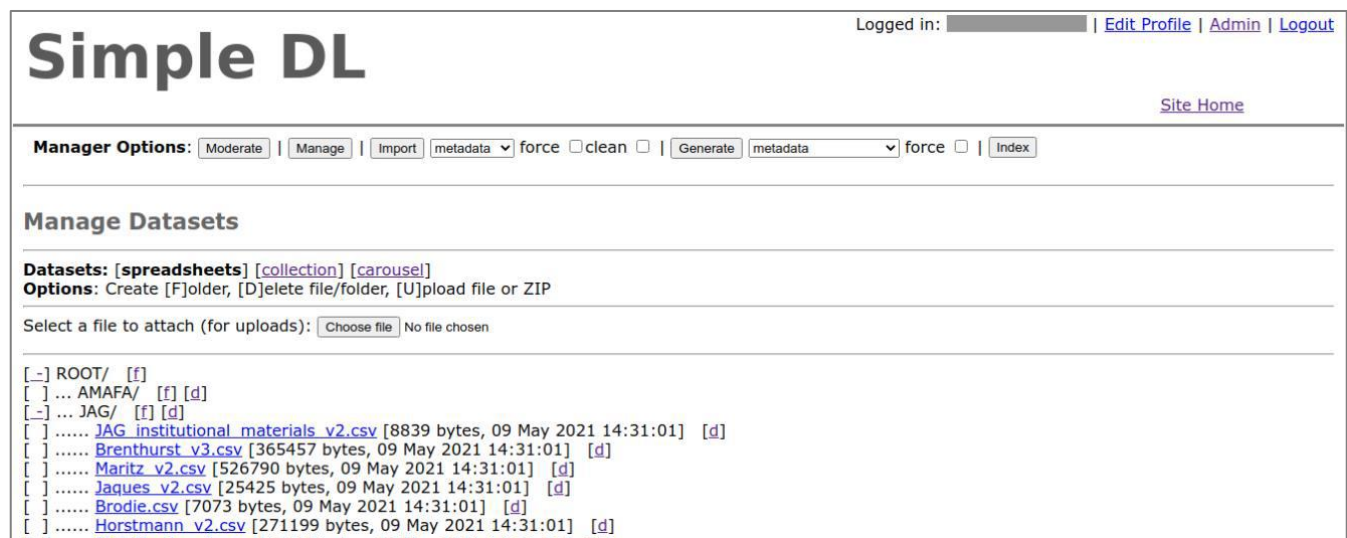


Figure 1: Current Simple DL administrative site

## 2.4 The Current System

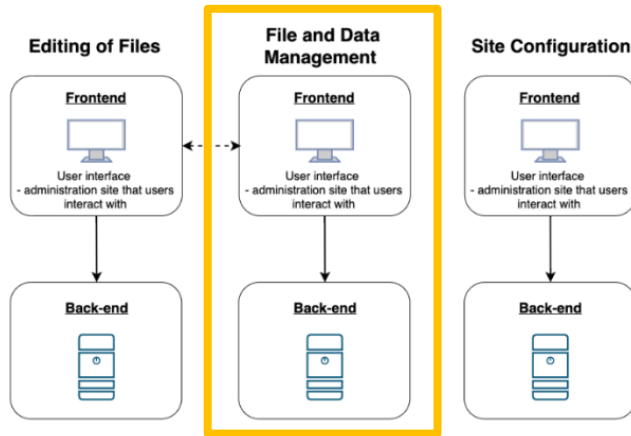
The current administrative site handles the files and data uploaded by administrators. The system can handle audio files, portable document format files (PDFs), documents, videos, comma-separated value (CSV) files, and many more. Although Simple DL can perform many actions such as uploading, deleting, creating, and downloading files and folders, the interface is not usable and requires many additional steps for the completion of tasks. For example, to upload a zipped folder, the user is required to zip the folder twice and then upload it. This is because the current system automatically unzips folders. To upload multiple files, the user is required to zip the files and then upload them since there is no functionality to select multiple files for downloading, uploading, or relocating. Some additional functionality that is completely missing from the system include a search bar, a sort-by function, and a preview box that allows users

into three separate projects. This division enabled the team to use an agile approach on a higher level since each member could work on a specific part of the system independently of one another. Kanban [17] was used to maximize the team's efficiency and communication. This approach allowed the project development to be divided into stages depending on progress. The use of a Kanban board made it easier to visualize and track workflow. Weekly meetings with our supervisor allowed for progress updates between iterations with users.

### 3.2 High-level Architecture

Before implementation, a model of the architecture for the new system had to be designed. A client-server model was agreed upon by the team which resulted in the architecture model displayed in Figure 2. The client side was developed using React.js and the server side was developed using Node.js. This

architecture allows tasks to be separated between the client side and the server side. The client side provides an interface for users to interact with while the server side manages all the requests in the backend and provides the result to the user. The editing of files is lightly connected to the file and data management system since each file in the system has metadata that should be editable. This project is focused on file and data management. The goal is to create a more usable administrative interface with the file and data management functionality and an interface like Google Drive.



**Figure 2: High-level system architecture**

### 3.3 Technologies

Before any coding or implementation could happen, the technologies of the project had to be understood. For this project, Node.js, React.js, JavaScript, and Express.js were utilized. The reason for this selection was to use a more recent technology that allows components to be reused, hence the choice to use React.js. Since JavaScript was going to be used in the frontend with React.js, and Node.js is one of the most popular platforms for hosting and running servers along with React.js, the team chose Node.js for the server side and React.js for the client side [18]. Another advantage to using Node.js was that it has a module with multiple functionalities for file system operations. The “fs-extra” module was used since it has additional functionality compared to the native “fs” module [19]. Node.js was used to build the server-side application while Express.js was used to handle the HTTP requests [17]. It is used on top of Node.js web server functionality and includes helpful utilities to Node.js HTTP objects. Express.js also allows users to define application routes using URLs which was necessary for this system. JSON (JavaScript Object Notation) is a text-based format used to represent structured data [18]. Since data had to be sent between the server and the client, this was also a prerequisite. It is based on the syntax of JavaScript and is mainly used for transmitting data in web applications. A careful review of the existing codebase was necessary to understand how the system currently functions. This review also highlighted the main functionalities and components of the system.

### 3.4 Requirements Gathering

Requirements gathering took place online since most of the Simple DL users were remote. Three users were recruited since they are current users of the Simple DL system and could provide expert input/opinion. The requirements were gathered through a series of interviews where the users were asked specific questions that were formulated beforehand based on what was missing from the current system and any additional functionality that could potentially make the system more usable. During the interviews, the interviewee could offer any additional suggestions or feedback. The questions were aimed at the usability of the system. These included questions such as:

1. What are the biggest challenges you face while using the system?
2. What works well or what do you like about the system?
3. What do you find most frustrating about the system?
4. What functionality do you think is missing from the system?
5. Do you prefer a tree-like structure or working in one directory at a time?

These questions aimed to encourage discussion to fully understand the issues and frustrations with the system. This semi-structured approach allowed for follow-up questions while following a set of pre-determined questions [17]. After the interviewee answered all the questions, the prototype was demonstrated. The goal of the demonstration was to give the user a basic idea of what the system would look like in terms of layout.

The requirement-gathering process was very insightful. The expert users had many suggestions for functionality and system improvements. The most common requirement was that the system display directories one at a time rather than in a tree-like structure. The second most common requirement was that the system could handle batch uploads without having to zip the files first. Many users also stated that the system required zipped folders to be zipped twice if the user wanted to upload a zipped folder. Many users found this to be frustrating. Additional requirements included some way to confirm that the file the user is working with is the correct file. For example, before a user deletes the file, they would want to confirm it is the correct file that they are going to delete. A preview window was suggested as a solution. Additionally, users wanted the option to select multiple folders and files before performing an action.

A use case diagram and a class diagram were created to understand what the system does and how it does it. Figure 3 displays the use case diagram and Figure 4 displays the activity diagram.

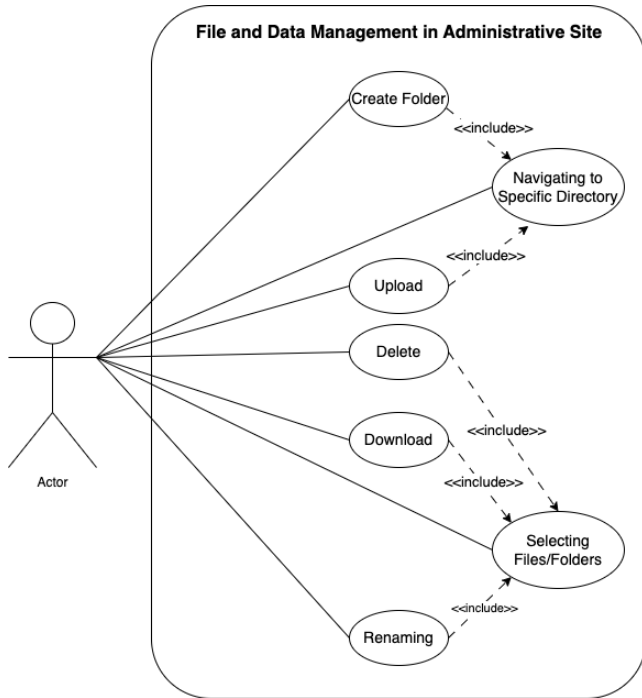


Figure 3: Use Case Diagram

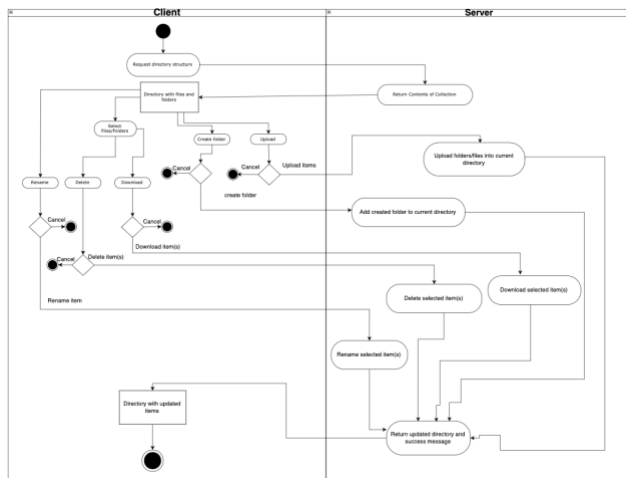


Figure 4: Activity diagram

### 3.5 Software Development Process

For the development of the new file and data management part of the system, an agile approach was used. This was to ensure a more user-centered design could be achieved with the user requirements and feedback being the driving force of development. This was achieved through an iterative process that began with a small subset of requirements and then iteratively improved the system with enhanced versions until the system was complete. Iteration one included the development of a prototype which was used to get some basic functionality working and give the users an idea of what the system should achieve. During the first iteration, the prototype was demonstrated to the project supervisor and a second evaluator. A second demonstration was done with Simple DL users (from the Emandulo project) for requirements gathering and to discuss some additional ideas. Iteration two included implementing the requirements of the users and developing a second, more functional system. This was used during user testing to evaluate the system and get any additional feedback from the users.

**3.5.1 Iteration One.** Before coding a paper-based design was drawn to visualize what the system would look like. Figure 5 displays the rough paper design used to visualize the first prototype.

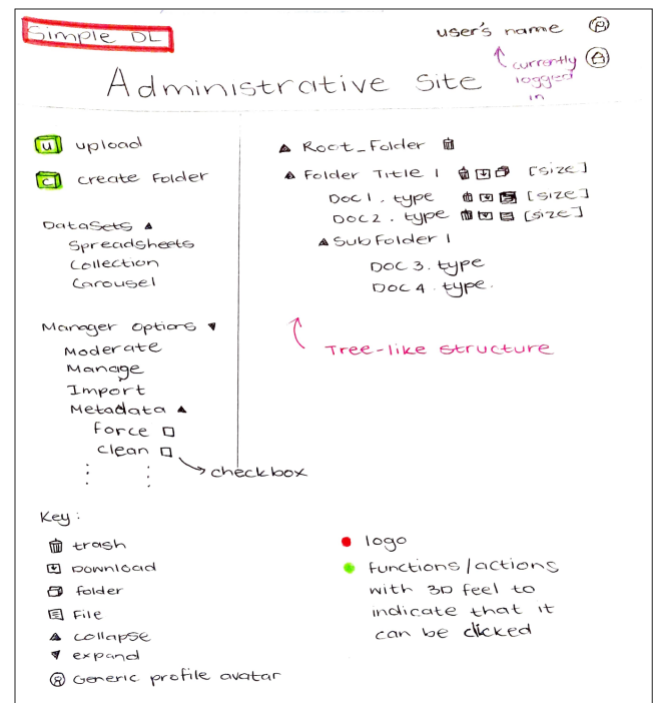


Figure 5: Rough sketch for prototype 1

Initially, the prototype built from this design was just a facelift of the previous system since user requirements were not complete yet. The system had the same tree-like structure but instead, the links used for the functionality were replaced with interactive

elements. This design aimed to get some basic functionality working and being able to display the directories and subfolders. After requirements gathering and some testing of the system, some changes had to be made. Initially, all the data of all the folders, subfolders, and files were returned to the front end through a JSON response. The system could display all the folders and their contents through plus and minus controls for expanding and collapsing the contents however, most of the users preferred working in one directory at a time rather than a tree-like structure. This meant that the system had to be redesigned to suit their preferences.

*3.5.2 Iteration Two.* After the redesign, the process of fetching the folder structure had to be changed since this was not proving to be very efficient. During the redesign, the system was developed to focus on one directory at a time, but only fetch the contents of a folder upon the user's request as opposed to the entire collection of folders, subfolders, and each of their contents. This required a GET request (an HTTP request used to request data from a specific resource) that could handle multiple subfolders. To handle the subfolders, a dynamic URL had to be used. The first attempt at using a dynamic URL included making use of a placeholder that was positioned at the end of the URL at the API endpoint. This failed since there were cases when no subfolder was chosen and therefore no value could be placed in the placeholder. To account for this, a question mark was used at the end of the placeholder to allow subfolders to be optional. Although this was more successful, it failed if a subfolder had a subfolder. This implied that the URL at the API endpoint had to be able to accept a dynamic path with any number of directories. After some research and experimentation with Express.js routing, a viable solution was created. This included the use of an asterisk in the route path to handle multiple subfolders appended at the end of the URL and a question mark to account for the case when no subfolders were selected. This system was used during iteration two where the testing was conducted with the same Simple DL users to collect expert feedback.

*3.5.3 Iteration Three.* Iteration three included the implementation of the additional functionality gathered during testing and fixing any bugs or issues highlighted. After the final improvements, the final set of usability tests was conducted with members of the digital libraries' laboratory at the University of Cape Town. These participants were recruited through email. These DL users are not specifically Simple DL users, to ensure the system is usable for any DL user and not only expert users. The goal of these tests was to ensure that potential users could use the system. The tests were used to confirm that the system was intuitive and performed as the user suspected.

## 3.6 User Testing

The functionalities discussed during requirements gathering were implemented to develop a more functional version of the first prototype. This version of the system was used for user testing with the Simple DL users during iteration two, to understand the

needs of the users and validate ideas that could be implemented. This was achieved by demonstrating the functionality through a series of task scenarios such as uploading multiple files and folders, deleting multiple files and folders, uploading zipped files without the system automatically unzipping the file, and navigating through the directory-focused system. After demonstrating the system to the users, users could also request certain tasks to be completed. Since users were remote, they would instruct the interviewer to perform a task after which the user would offer feedback or suggestions. The main goal of this iteration was to build the main structure of the directories, allow multiple uploads of files including zipped files and handle multiple deletions. Through the user testing, additional functionality and features were highlighted such as renaming folders, creating folders directly on the system, moving folders and files, and allowing users to copy files and paste them into different directories. These were features that were not included in the system used for testing.

*3.6.1 First round of user testing.* The feedback for the functionality built in this iteration was positive but through the evaluation of Nielsen's ten usability heuristics, additional functionality was highlighted. Through a questionnaire and a rating scale, the expert users rated each heuristic, these were recorded with the additional feedback and suggestions. The ratings were averaged and recorded in Table 1 to understand the trends and highlight the most necessary improvements. The main improvements suggested were including feedback or confirmation of when a task has been done successfully. For example, when deleting a file, alert the user that the file has been deleted successfully. One of the functionalities that the users requested to be included in the system was being able to create or rename a folder directly through the interface. With this, they also suggested that an exit or cancel button be included in case they change their mind. To improve the flexibility and efficiency of use, one of the users suggested implementing a hierarchical overview of the system on the left so that users could move files from one directory to the next without multiple steps involved. Most of the users were impressed and satisfied with the minimalist design of the system, however, there was a suggestion to include a path so that the user was aware of which directory they were currently in. Since no error messages were displayed during the testing, this was also highlighted as an area of improvement to ensure error messages are clear and expressed in plain language with a suggestion as a solution. The error prevention of the system also needed to be improved. During user testing one of the users asked what the consequence was if a file was uploaded to the system which already existed, this was one of the errors that had to be prevented in the system. One user suggested that the help and documentation of the system needed to be improved since it wasn't clear that the checkboxes were used for batch operations. Instead, the user suggested that an information block be displayed so that the user knows what the checkboxes are for.

*3.6.2 Refining the Requirements.* Since the focus of this project is to improve the interface by making it more usable, the focus would be placed on developing quality, and necessary functionality. After the requirements gathering and user testing, there was a long list of suggestions and additional functionality that could be implemented to improve the system which included:

- Upload, delete, and download in batches
- Directory-focused view of folders and files
- Creating folders
- Renaming folders
- Selecting multiple files and folders to perform a single action
- Allow zipped files to be uploaded without having to zip them twice
- Uploading and downloading of folders
- Exit buttons to cancel actions such as renaming, creating folders, uploading, etc.
- Confirmation boxes to confirm with the user before performing an action
- A file preview window
- View of current location or full directory
- Sort functionality
- Copying, pasting, and moving files and folders
- Error prevention and error messages

However, it was a priority to build a usable interface with reliable and necessary functionality. To refine the requirements two lists were written up. The non-negotiable list included all the functionality that was essential for managing files and data, including uploading, creating folders, downloading, and deleting. The remaining functionality was added to a separate list for additional functionality that could potentially improve the user experience but was not essential.

*3.6.3 Second round of user testing.* For the last iteration, after implementing the additional features and fixing some errors, the system was ready for usability testing. This was done in person with four DL users separately. The users were asked to perform a series of tasks while providing feedback. During the testing, a facilitator took notes of comments or important observations. Tasks included navigating to a subfolder, deleting items, creating a folder, uploading some resources into the folder, renaming a folder, navigating through the system and trying to determine which directory the user is in, navigating back to the home screen, downloading some resources from the system and sorting by any of the file properties. With each task, the user could provide feedback. After the usability testing, users were asked to rate the system based on Nielsen's ten usability heuristics and answer a set of questions. This was used to test the improvement of the system through quantitative and qualitative analysis. The set of questions asked included:

1. How would you describe your overall experience with the interface?
2. What did you like the least?
3. What did you find most frustrating?

4. What do you like the most about the system, and why?
5. How simple and clean was the interface?
6. Was the language in the system clear and straightforward?
7. Was there any task that you found difficult to complete?
8. What's your opinion on the way features and information are laid out?

All the users had a positive experience with the system, they all found the interface to be intuitive and simple. Three of the users said the interface felt like a Microsoft file manager and that it was very intuitive and straightforward. Through observations, it was clear that the folders and files had to be more distinguished from one another since many of the users found it difficult to differentiate between files and folders. Some users also struggle to distinguish between the file property names and the names of files. This layout can be seen in Figure 6.

<input type="checkbox"/>	Name	Type	Size	Last Modified
<input type="checkbox"/>	ackermann.pdf	.pdf	595.22 KB	Jun 26, 2021, 06:52 AM
<input type="checkbox"/>	dolan.pdf	.pdf	246.34 KB	Jun 26, 2021, 06:52 AM
<input type="checkbox"/>	greenberg.pdf	.pdf	87.99 KB	Jun 26, 2021, 06:52 AM
<input type="checkbox"/>	hinwade.pdf	.pdf	211.34 KB	Jun 26, 2021, 06:52 AM
<input type="checkbox"/>	macha.doc	.doc	274.50 KB	Jun 26, 2021, 06:52 AM
<input type="checkbox"/>	rama.pdf	.pdf	171.59 KB	Jun 26, 2021, 06:52 AM
<input type="checkbox"/>	srinivasan.doc	.doc	1.30 MB	Jun 26, 2021, 06:52 AM
<input type="checkbox"/>	srinivasan.pdf	.pdf	573.92 KB	Jun 26, 2021, 06:52 AM
Rows per page: 100 ▾ 1-8 of 8 < >				

**Figure 6: System layout with file properties**

The sorting functionality of the system was not intuitive since the arrow used to show the direction of the sort (ascending or descending) only appeared if the user hovered over the property that they wanted to sort by. Two users suggested that it would be clearer if the arrow was always present without the need for hovering.

## 3.7 Final Product

After iteration three, the final product was developed with the basic functionality to perform tasks such as uploading, creating folders, renaming folders, downloading and deleting files and folders. The new system displayed files and folders in a directory-focused view as opposed to the tree-like structure since this is what all the users preferred. This directory-focused approach also allowed for files to be sorted according to name, type, size, or the last modified date. Additionally, the following features and functionality were included:

1. A zipped folder could be uploaded without having to zip the contents twice
2. Bulk-uploading, downloading, and deleting were made possible
3. Folders can be renamed from the system directly



4. A sort-by functionality has been implemented for each file's properties such as name, type, size, and date last modified

To improve some of Nielsen's usability heuristics, exit and cancel buttons have been included for each action button (delete, rename, upload, create, download, etc.). Error messages have been included for clear communication to the user through pop-up messages, and confirmation messages are implemented to double-check with the user that they are sure they want to perform an action. This was included to reduce and diagnose errors. To offer a more comfortable and familiar feel, the system also includes a "current location" using a breadcrumb to clearly inform the user which directory they are currently working in. The system also enables users to drag-and-drop resources that they want to upload. The renaming of folders has error checks in place to ensure that the current directory does not already contain a folder with the same name and that naming conventions are followed. This is achieved through a regular expression check that prevents the user from starting folder names with special characters or whitespace. The system also sends success messages when actions have been completed successfully.

Apart from functionality, multiple visual aspects have been improved to contribute to the look and feel of the system. The biggest visible change was the layout of the folders. The folders and files are now displayed in rows for each directory. This eliminated the tree structure of the previous design. The system also includes physical buttons for functionalities such as uploading, creating folders, and deleting as opposed to the previous system which had linked letters that represented each function. Any possible functions that can be applied to folders or files have been placed in a button located toward the top of the page. This reduces the user's memory load by making actions and options visible. The system also greys out specific functionality to prevent the user from performing the wrong actions. For example, a user must select one folder to rename as opposed to selecting multiple folders and renaming all of them at once. The system also makes use of familiar phrases, terms, and symbols to match the system to the real world. Some smaller visual aspects that were implemented to improve the user experience and understanding included, changing the cursor depending on the actions that could be taken, highlighting functions and folders when the cursor hovers over it to indicate that they can be clicked to either action the function or enter the folder. Differentiating the column names in the table from the files and folders was inspired by the observations of the usability testing. Instead of the sort-by function only being visible when the cursor hovers over the file property, it is not always visible.

To achieve the goal of the system being something like Google Drive, the folders and files in a single directory were displayed as rows in a table that could be sorted by name, type, size, or date last modified. The drag-and-drop functionality for uploading was also inspired by Google Drive. The system also makes use of familiar terms and symbols for functionality to improve the user

experience and understanding. This also reduced the need for documentation and created a match between the system and the real world by following conventional layouts and making information appear in a natural and logical way. All the lines, colour placement of objects, and visual aspects of the system were kept simple to create a cleaner look and make the system more visually appealing.

## 4. Results and Discussion

This section discusses the aspects of the project that worked and those that didn't, the results, the limitations, recommendations to others trying to replicate the work, future work, and a reflection.

Overall, the system achieved the goal of being a more usable interface. The system is simple, and clean and provides all the necessary functionality for file and data management for Simple DL. The rating for Nielsen's ten usability heuristics decreased as seen in Table 1 indicating an improved, usable interface. Many of the users provided positive feedback about the usability of the system. Users mentioned that the system was "straightforward" and reminded them of the Microsoft File Explorer.

**Table 1: Comparison between initial and final ratings for Nielsen's ten usability heuristics**

Heuristic	Initial (Avg)	Final (Avg)	Standard Dev
1. Visibility of system status	2	1.25 ~ 1	1.09
2. Match between system and the real world	0	0.5~1	0.87
3. User control and freedom	1	0.33 ~ 0	0.43
4. Consistency and standards	0	0.33 ~ 0	0.43
5. Error prevention	2	0	0
6. Recognition rather than recall	0	0.33 ~ 0	0.43
7. Flexibility and efficiency of use	0.66 ~ 1	0.33 ~ 0	0.43
8. Aesthetic and minimalist design	1	0	0
9. Help users recognize, diagnose, and recover from errors	4	0	0
10. Help and documentation	1	0	0

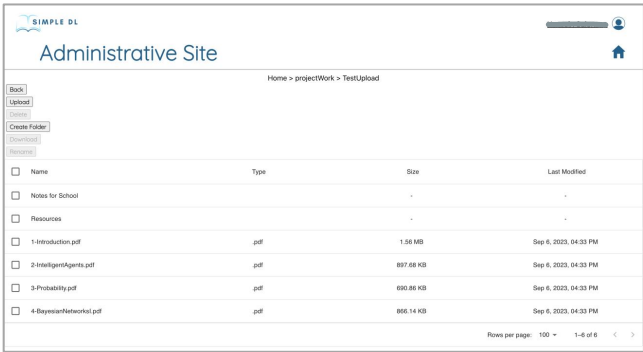
0 = No usability problem at all,

1 = Slight usability issue: need not be fixed unless extra time is available,

2 = Minor usability issue: low priority,

- 3 = Major usability issue: important to fix, high priority,
- 4 = Usability catastrophe: imperative to fix before the product can be released

The system used during the usability testing was more focused on functionality, to ensure that users could perform tasks, recover from errors, and understand the system without any prior expertise. The layout of the interactive elements used to perform the actions of the system was moved and displayed horizontally. The breadcrumb was placed above the table displaying the files and folders. The header column of the table was also changed to make it easier for users to differentiate between the header and the contents being displayed. The final system also included visual clues to emphasize functionalities such as clicking or selecting. Another round of testing would have been helpful to reflect the improvements of the system using Nielsen’s heuristics since the feedback and observations obtained during the usability testing influenced the final look and feel of the system.



**Figure 7: Interface used during usability testing**

Although most of the ratings decreased, with the largest improvement being helping users recognize, diagnose, and recover from errors, another round of testing could have been done to try and improve the visibility of the system status and the match between the system and the real world. Additionally, more participants could have been used as this may have produced better results since certain features were obvious to some participants but not to others. During testing, it was observed that a different testing laptop should have been used since some of the participants were not used to the operations of the testing laptop which could have influenced their experience with the system. Although this effect was not produced by the system functionality, it would have been better to let users test the system on a device were familiar with. Testing on multiple devices should be considered if the project were to be replicated.

It would have also been ideal to do a round of usability testing with the expert users to do a comparison with the initial ratings using the same group of individuals. If the project had to be done again, there would be two sets of testing, one with expert users and another without but both sets would have multiple rounds of testing. It would have also been ideal to have all the expert users

complete user testing in person so that they could get to experience the system for themselves however this was not possible in this case since the expert users were remote.

The functionality that was implemented from the list of requirements was chosen depending on their importance, such as the non-negotiable functionality, and the contribution it would make towards the usability and simplicity of the system. The goal was to create a simple, intuitive system that would not overload the user with too many functionalities and additional features. In future work, the system could include a preview window that allows users to get a view of the file they are working with. For documents, it would be a static view but for videos or audio, a different approach would be required. This could include a small media player window to play a few seconds of the video or the audio file. Since the files in the system can be any type, various preview windows will be required for each type. This was researched but React.js could only offer a preview window for specific types of files. The functionality to move, copy, and paste files and folders may also prove to be useful to users. During requirements gathering one user suggested including functionality to view the activity of users who are registered on the site. This would include viewing their contributions to the collection and viewing their comments. This was not discussed in this project since it was not within the scope of the project.

### 4. Conclusions

The project aimed to develop a new, usable interface for the administrative site of Simple DL for the management of data and files. Through requirements gathering with expert users and two rounds of testing with experts and non-experts, the final system proved to be more usable while having all the necessary functionality for file and data management. The users had positive feedback from usability testing and the ratings for Nielsen’s heuristics improved overall. Using multiple iterations of development and testing with users was a good approach to take as it allowed a more user-centered design and improvements could occur incrementally throughout the development process. The agile methodology encouraged this iterative approach and allowed development to be efficient and output-driven.

The system successfully addressed the needs of Simple DL users. This meant that the interface could assist digital library curators in managing the data and files of digital repositories without prior technological expertise. The intuitive design allows for the functionality of the site to be more efficient and easier to use. This can improve the productivity and efficiency of the site while simplifying the process involved with managing the files and data of the repository. The system also reduces the need for additional, unnecessary steps to complete tasks thus improving the user experience and simplifying processes. Although the system meets the aims of the project, additional functionality could be implemented in future work to provide users with additional functionality that could speed up processes.

## ACKNOWLEDGMENTS

Thank you to Professor Hussein Suleman for his invaluable support and guidance throughout the duration of this project. Thank you to the Simple DL users and DL users who participated in this project. Your crucial insights, feedback, and participation were valued and helpful.

## REFERENCES

- [1] D. Yadav, "Opportunities And Challenges In Creating Digital Archive And Preservation: An Overview," *International Journal of Digital Library Services*, vol. 6, no. 2, pp. 63-73, 2016.
- [2] H. Suleman, "Simple DL: A toolkit to create simple digital libraries," 1-3 December 2021. [Online]. Available: <https://pubs.cs.uct.ac.za/id/eprint/1512/>. [Accessed 23 August 2023].
- [3] A. Maxwell, "Digital archives and history research: feedback from an end-user," *Library Review*, vol. 59, no. 1, pp. 24-39, 2010.
- [4] G. Biswas and D. Paul, "An evaluative study on the open source digital library softwares for institutional repository: Special reference to Dspace and greenstone digital library," *African Journal of Library and Information Science*, vol. 5, no. 6, pp. 001-010, 2019.
- [5] R. Tansley, M. Bass, D. Stuve, M. Branschovsky, D. Chudnov, G. McClellan and M. Smith, "The DSpace Institutional Digital Repository System: Current Functionality," in *Proceedings of the 2003 Joint Conference on Digital Libraries (JCDL'03)*, Houston, 2003.
- [6] L. Verma, "Comparative Analysis of Open Source Digital Library Softwares: A Case Study," *DESIDOC Journal of Library & Information Technology*, vol. 38, no. 5, pp. 361-368, 2018.
- [7] K. K. Singh and M. Asif, "Emerging trends and technologies for digital transformation of libraries," *IP Indian Journal of Library Science and Information Technology*, vol. 4, no. 2, pp. 41-43, 2019.
- [8] M. Castagné, "Institutional repository software comparison: DSpace, EPrints, Digital Commons, Islandora and Hydra," 14 August 2013. [Online]. Available: <https://open.library.ubc.ca/soa/cIRcle/collections/graduateresearch/42591/items/1.0075768>. [Accessed 24 August 2023].
- [9] I. H. Witten, S. J. Boddie, D. Bainbridge and R. J. McNab, "Greenstone: a comprehensive open-source digital library software system," in *In Proceedings of the fifth ACM conference on Digital libraries (DL '00)*, New York, 2000.
- [10] S. J. Kumaravel, "Is Digital Archiving Simple?," in *INFLIBNET's Convention Proceedings*, Imphal, 2004.
- [11] A. Kochovski, "What Is Google Drive and How Does it Work? – A 2023 Guide," Cloudwards, 17 May 2023. [Online]. Available: <https://www.cloudwards.net/how-does-google-drive-work/>. [Accessed 10 September 2023].
- [12] D. Ginn, "A Beginner's Guide on How to Use Dropbox in 2023," Cloudwards, 12 April 2023. [Online]. Available: <https://www.cloudwards.net/how-to-use-dropbox/>. [Accessed 10 September 2023].
- [13] M. McLaughlin, "What Is OneDrive and How Does It Work?," Lifewire Tech for Humans, 12 September 2021. [Online]. Available: <https://www.lifewire.com/what-is-onedrive-and-how-does-it-work-5093633>. [Accessed 10 September 2023].
- [14] C. Ujjainkar, "Comparision Between Google Drive, One Drive And Dropbox," *International Research Journal of Modernization in Engineering Technology and Science*, vol. 4, no. 10, pp. 49-53, 2022.
- [15] G. Kulkarni, R. Waghmare, R. Palwe, V. Waykule, H. Bankar and K. Koli, "Cloud Storage Architecture," in *7th International Conference on Telecommunication Systems, Services, and Applications (TSSA)*, Denpasar, 2012.
- [16] D. KamalaKannan, D. K. Sharmila, M. C. Shanthi and M. R. Devi, "Study on Cloud Storage and its Issues in Cloud Computing," *International Journal of Management, Technology And Engineering*, vol. 9, no. 1, pp. 976-981, 2019.
- [17] C. Halton, "What Is the Kanban System?," Investopedia, 28 September 2022. [Online]. Available: <https://www.investopedia.com/terms/k/kanban.asp>. [Accessed 14 September 2023].
- [18] M. Gupta, "Best Backend for React in 2023," KnowledgeHut, 5 September 2023. [Online]. Available: <https://www.knowledgehut.com/blog/web-development/best-backend-for-react#best-10-backend-for-react.js-%C2%A0>. [Accessed 14 September 2023].
- [19] I. Shlueter, C. McConnel, J. Halliday and A. Kelley, "fs-extra," npm, 20 March 2023. [Online]. Available: <https://www.npmjs.com/package/fs-extra>. [Accessed 14 September 2023].
- [20] Geeks for Geeks, "Express.js," Geeks for Geeks, 19 April 2023. [Online]. Available: <https://www.geeksforgeeks.org/express-js/>. [Accessed 27 August 2023].
- [21] MDN contributors, "Working with JSON," Mozilla, 3 July 2023. [Online]. Available: <https://developer.mozilla.org/en-US/docs/Learn/JavaScript/Objects/JSON>. [Accessed 27 August 2023].
- [22] T. George, "Semi-Structured Interview | Definition, Guide & Examples," scribbr, 22 June 2023. [Online]. Available: <https://www.scribbr.com/methodology/semi-structured-interview/>. [Accessed 23 August 2023].
- [23] E. Ziemba, T. Papaj and D. Descours, "Critical Success Factors for Adopting Electronic Document Management Systems in Government Units," in *Proceedings of the of the*

## Supplementary Resources

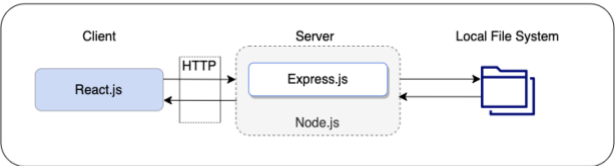


Figure 8: Technology used in the client-server architecture

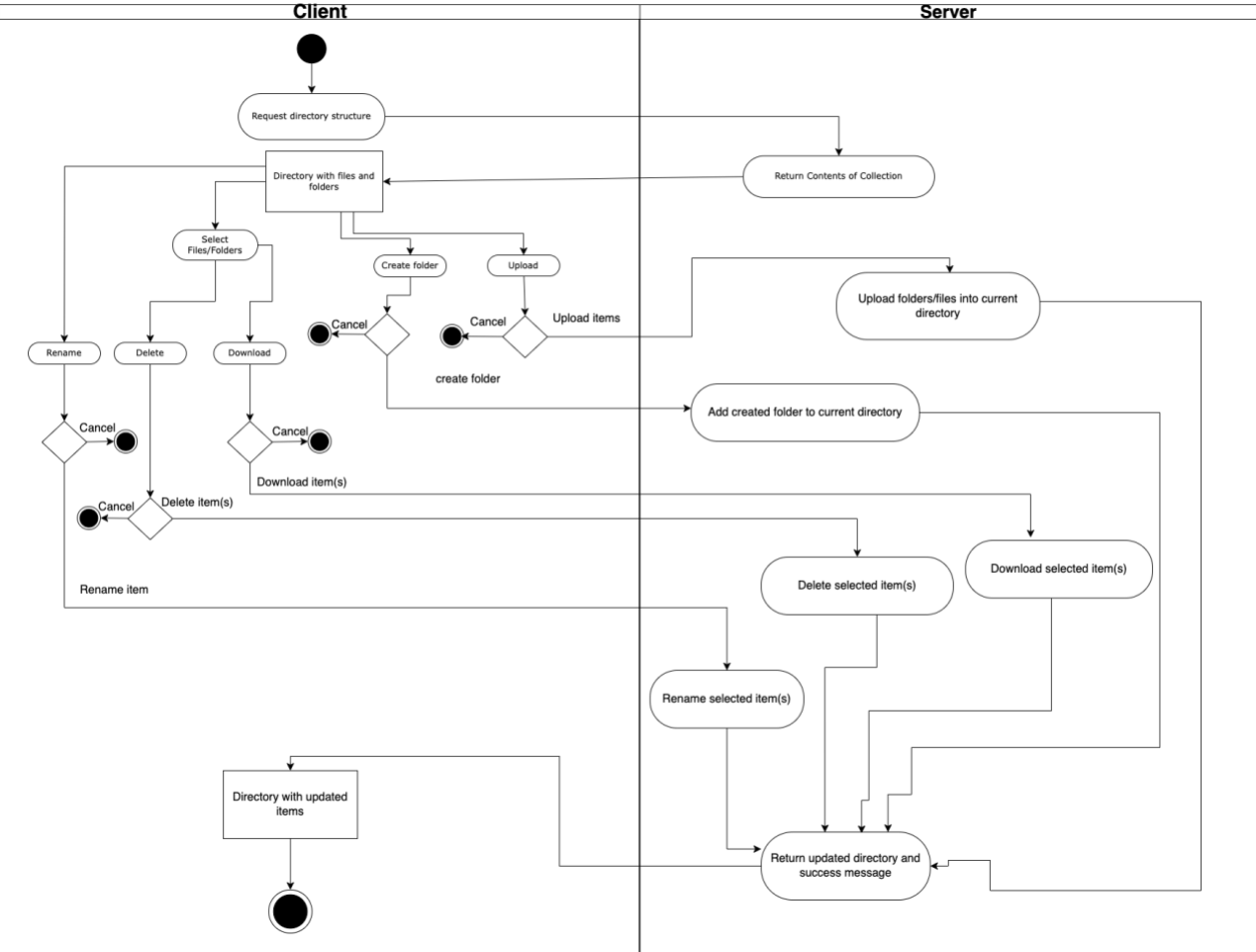




Figure 9: Enlarged version of Figure 4





Administrative Site

Home > projectWork > TestUpload

Back

Upload

Delete

Create Folder

Download

Rename

<input type="checkbox"/>	Name	Type	Size	Last Modified
<input type="checkbox"/>	Notes for School		-	-
<input type="checkbox"/>	Resources		-	-
<input type="checkbox"/>	1-Introduction.pdf	.pdf	1.56 MB	Sep 6, 2023, 04:33 PM
<input type="checkbox"/>	2-IntelligentAgents.pdf	.pdf	897.68 KB	Sep 6, 2023, 04:33 PM
<input type="checkbox"/>	3-Probability.pdf	.pdf	690.86 KB	Sep 6, 2023, 04:33 PM
<input type="checkbox"/>	4-BayesianNetworksI.pdf	.pdf	866.14 KB	Sep 6, 2023, 04:33 PM

Rows per page: 100 1-6 of 6 < >

Figure 10: Enlarged version of Figure 7