Honours Project Report

Zamani Data Archive: Public Portal and Archive

Author:
Jay Benson

Supervisor:
Prof. Hussein Suleman

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

Department of Computer Science
University of Cape Town
28 October 2014

Abstract

The Zamani Data Archive has been created to manage the partially structured spatial data collected by the Zamani Project. The Zamani Data Archive comprises of an Archive, Public Portal, Search Interface and Metadata Management Tool. This report outlines the development of the Archive and Public Portal components. These two components were developed over three iterations. Each iteration consisted of a planning, design and implementation, testing and evaluation phase. The archive needed to be able to create relational links between the records found in the archive to provide it with a good structure. Additionally, the Archive provides Archival Tools, allowing an administrator of the Zamani Data Archive to ingest and purge objects from the archive. The Public Portal allows users to create an account on the Zamani Data Archive. Registered users are then able to request data from the Zamani data set. The Archive also provides the administrator with the ability to manage these permissions through the Request Management functionality. The system was evaluated with a Usability Test that included a System Usability Scale (SUS). This was used to obtain a subjective label for each component based on an adjective rating scale. The mean SUS scores for both the Archival Tools and Public Portal fell into the bracket between good and excellent. The last evaluation done for the system was a User Acceptance Test with the Zamani Project team. The team responded positively to the Zamani Data Archive. This was further enforced when the team discussed future work on the system and the steps required to take the system to a production environment.

Acknowledgment

Firstly I would like to thank my project partner, Michael Ferguson for the guidance and support you provided me during the development process. Without your constant encouragement, I fear this project would never have been completed in time. I look forward to working on more projects with you in the future.

I would like to thank my project supervisor, Professor Hussein Suleman for the continued advice you provided during the various stages of the project. You served as a valuable source of feedback during our demonstrations. Additionally, your constant encouragement to meet deadlines helped me to complete the deliverable of the project on time.

I would also like to thank the Zamani Project team for their enthusiasm and engagement in the project. Thank you Professor Heinz Rüther, Roshan Bhurtha, Ralph Schroeder and Stephen Wessels for giving up your valuable time for consultations and demonstrations of the system. In particular I would like to thank Roshan Bhurtha for acting as the liaison between the Zamani Project team and the development team. Thank you for responding to emails quickly and providing all the required resources for the project.

Thank you to all the University of Cape Town students that participated in the Usability Test for the system. Your feedback provided the development team with valuable metric for evaluating the success of the system.

Finally I would like to thank my family for providing me with constant encouragement, reassurance and support during the high stress writing and development stages of the project.

Table of Contents

Table of Figures

Introduction

Project Description

The Zamani Project was initiated to create a permanent metrically accurate record of important heritage sites in Africa and the Middle East as well as increase the international awareness of these sites. The purpose of this preservation is fourfold in that the data has been collected for education, research, restoration and conservation purposes (Zamani Project, 2014) .

Thus far, the Zamani team has documented approximately 40 sites in 12 countries in Africa and the Middle East, amounting to close to one hundred three-dimensional (3D) models of individual structures. The dataset collected by the team contains approximately 44TB including backups and copies. The data comprises of different forms such as 3D models, plans, videos, panorama imagery and Geographic Information Systems (GIS) (Zamani Project, 2014).

The Zamani team requires an archive for their large geospatial data collection from cultural heritage sites. The problem consists of two major aspects, namely: archival storage management and end-user presentation.

The archival storage component of the system needs to consider the relationship between items in the Zamani data set. A Metadata Management Tool is needed to create, modify and transform metadata relating to the Zamani data set. Additionally, it needs to provide the administrator of the system with tools to ingest and purge content from the archive.

Researchers and the general public required access to the data as one of the primary objectives of the Zamani Project is creating awareness for the sites. The archive requires a Public Portal that will enable end-users to search, view, request and download data from the archive. The efficiency of file transmission needed to be considered due to the large file size found in Zamani's dataset. Additionally, access to the data collected by the Zamani Project must be controlled so that the data is used for its intended purpose and not exploited for commercial reasons. A diagram that outlines the user interactions mentioned above can be seen be seen in Figure 1.

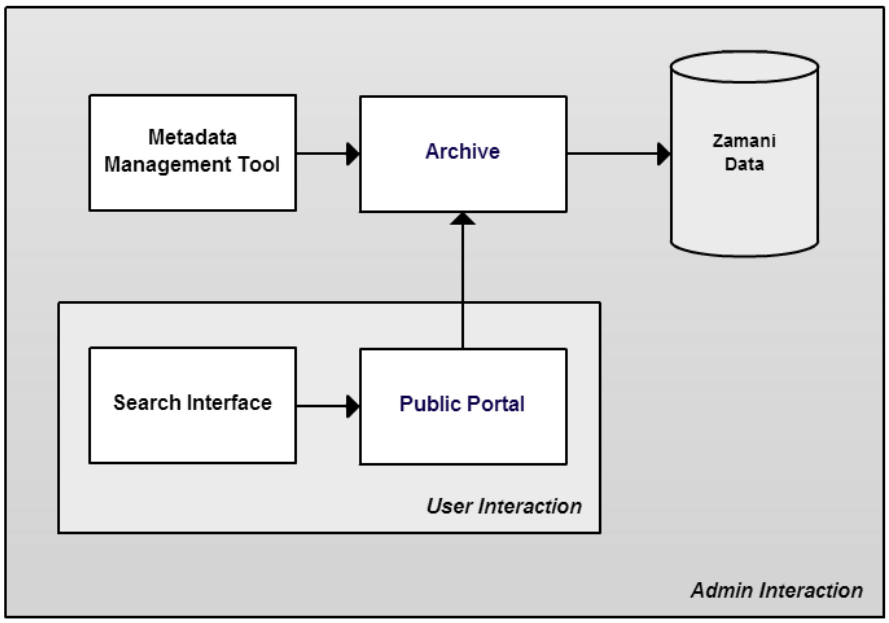


Figure 1: Overview of the Zamani Data Archive

Discussion of System Interaction

The proposed solution can be broken into user and administrator interaction with the Zamani Data Archive. Users are able to access the Zamani data set via the public portal. The viewing of records is done via the Search Interface which extends the public portal functionality. Additionally, the public portal provides functionality allowing the user to request and download data from the Zamani data set.

The administrative interaction with the system consists of creating, modifying and transforming metadata for the Zamani data set using the Metadata Management Tool. Additionally, the Archive provides Archival Tools for managing the content in the repository.

Motivation

Preservation of heritage sites throughout the world has been the goal of enthusiasts and researchers alike. Spatial data acquisition repositories such as Zamani have proved (Rüther, et al., 2009) to be invaluable in facilitating quantitative analysis and planning of the preservation of heritage sites. Aside from providing permanent records of cultural heritage for future generations, it has potential to aid in providing practical quantitative planning of the conservation and restoration of such sites. Finally, with regard to education and tourism, a project such as Zamani serves as a means to publicize African heritage.

Finding an effective way to archive the large collection of interlinked geospatial data held by the Zamani Project is a significant impact of the project, as the team has no tools to perform such tasks. The archive provides the Zamani Project with a means to structure their data and share their research with the public and collaborators alike. Additionally, this project served to provide future preservation projects with an efficient model to work from.

Proposed Solution

To solve this problem this project has created the Zamani Data Archive. The project was separated between Michael Ferguson and Jay Benson. Michael Ferguson was responsible for the Search Interface and the Metadata Management Tool. Jay Benson was responsible for the Data Request Interface, Archive, Archival Tools and Request Management. This document describes the components that Jay Benson dealt with, which can be seen in light grey in Figure 2.

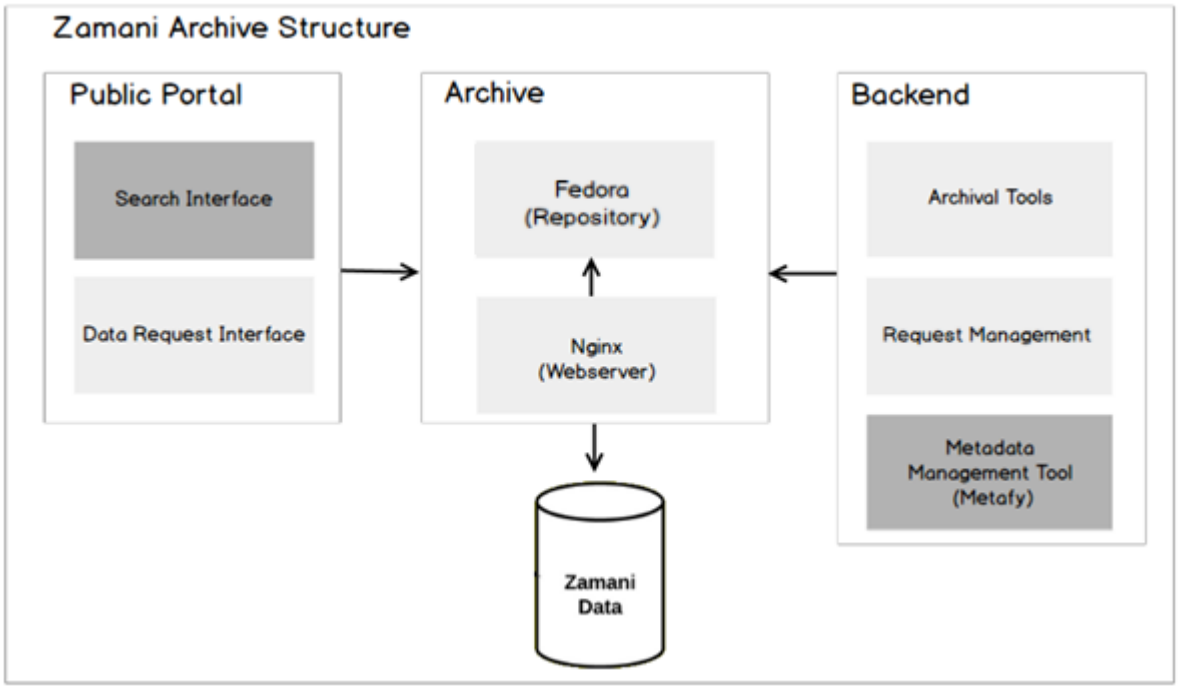


Figure 2: Zamani Data Archive structure (component breakdown)

Discussion of Proposed Solution

The Archive and Archival Tools of the Zamani Data Archive are the components responsible for archival storage management. Nginx served as both a Webserver and a proxy server to relay requests to Fedora. The Webserver was also responsible for serving static content relating to the Public Portal and Backend. Additionally, Nginx also served downloads to users. Fedora is responsible for storing Fedora Extensible Object Extensible Mark-up Language (FOXML) representations of the Zamani data set.

The Public Portal provides a presentation of the data to the end-users and can be broken up into the Search and Data Request interfaces. The Data Request Interface served as the Public Portal Component of the permissions system allowing users to request files Zamani Archival System. The Search Interface was responsible for allowing users to view and navigate through the Zamani data set. Additionally, the Data Request interface includes the Account page where users can see their current permissions for files they have requested and allows them to download requests that have been accepted.

The Backend comprised of all the administrative functionality of the Zamani Archival System. The Archival Tools allows ingesting and purging of records from the Fedora repository. The Request Management feature served as the administrative component of the permissions system. Additionally, the Backend includes the Metadata Management Tool (Meta-fy) for creating, updating and transforming metadata relating to the Zamani data set.

Both the Public Portal and Backend were created using a combination of PHP and JavaScript. PHP was chosen as the server side scripting language, while JavaScript was used to dynamically generate content on the client side.

Evaluation

During the development of the Zamani Archival System formative evaluations were conducted with the project team, the Zamani team and the supervisor. These evaluations gave feedback on functional requirements, implementation strategies and usability issues. These evaluations were used to refine the system and ensure that it could meet the requirements of the client.

The final evaluation of the Zamani Data Archive comprised of a system usability test and a user acceptance test. The usability test was conducted in a study with University of Cape Town students; this qualitative metric was used to access how well the requirements of the project were met. The user acceptance test was created to gauge the overall success of the project as determined by the client. Additionally, it provided the development team with an opportunity to demonstrate the product to the Zamani team. The final evaluations of the project were also used to identify any bugs in the system and opportunities for future work on the Zamani Archival System.

Background

Introduction

This chapter seeks to show the analysis that was performed after the final with regards to the repository software used for the Zamani Data Archive. It provides a comparison of Dspace and Fedora, the two alternatives identified for use in the Zamani Data Archive.

Dspace vs Fedora

Dspace

Dspace is a Java-based free open source repository package licenced under a BSD open source license. This means that the repository can be used by any organisation. Additionally, this licence permits the modification and integration of the source in any commercial product.

Dspace provides a "turn-key" installation, this means that it can be used directly after installation without the need for additional configuration. The Dspace repository comes with a Web-based interface that can be installed on Windows, Mac OSX and Linux. This allows for the quick creation of a testing environment (Dspace, 2014).

Dspace has a large community consisting of over 1000+ active repositories used for either production or project environments. The Registry of Open Repositories (ROAR) provides timely information about open repositories throughout the world. It currently has 1501 Dspace based repositories listed (ROAR, 2014). This community is far larger than of other the other open repositories found in the registry. This large community provides access to additional modules and configurations.

Dspace allows full customisation of the look and feel of its Web-based interfaces. There are currently two main alternatives: a standard Java Server Pages based interface and an XML interface based on the Apache Cocoon Framework (Dspace, 2008).

Dspace uses a Dublin Core metadata format that can be modified to the needs of the required project. Additionally, other metadata schemas such as MARCS and MODS can be mapped and loaded into the repository. Dspace allows you to choose which metadata fields you would like to index and offers full text-based searching on any of the desired Dublin Core fields (Dspace, 2014).

Dspace can be used in conjunction with either a PostgreSQL or an Oracle database. PostgreSQL is a free open-source database, while Oracle provides a paid for alternative (Dspace, 2014).

A clean Dspace installation with no additional configuration is able to handle any file format. File formats are categorised into three levels: Unknown, Known and Supported. A bitstream format registry is provided that allows the addition of more formats. This allows unrecognised formats to be identified in the future (Duraspace, 2014).

Fedora

Fedora is a java-based open source repository service licensed under the Apache Licence, Version 2.0. This means that the software is distributed on an as is basis without warranties or conditions of any kind, either express or implied (Fedora Commons, 2014). Fedora can be installed on Window, Linux and Mac OSX (Fedora, 2014).

Fedora manages all content as data objects, each composed of a PID and various datastreams that contain either content or metadata data relating to the object. Any number of datastreams can be defined for in a data object. The DC or Dublin Core and RELS_EXT datastreams are reserved in Fedora. The DC datastream can however be modified to the requirements of the problem. The RELS_EXT data stream is responsible for asserting RDF relationships between data objects in the repository (Fedora Project, 2007).

Fedora can be configured with a number of databases including Derby, MySQL, Oracle, PostgreSQL and Microsoft SQL Server (Fedora, 2014).

Fedora also comes packaged with the optional Generic Search Service (GSearch) which is able to index Fedora FOXML records through disseminator calls. The GSearch Service is also able to search these indexes. Additionally, the service can plugin with a number of search engines namely; Lucene, Solr and Zebra (Fedora Project, 2007).

Fedora does not come packaged with Web interface. Instead Fedora provides its core repository service through its extensive Web-based REST and SOAP APIs which allow you to create, read, modify and delete data objects in the repository. This provides the tools need for the creation of a custom Web-based interface (Fedora Project, 2010).

The Fedora community is relatively small with only 54 active repositories used in a production or project environment according to the ROAR (ROAR, 2014). Although the community is small there are a number of Fedora based alternative available that have extended the Fedora functionality to include a Web based interface and search engine integration. The two most popular derivatives are Islandora a Drupal and Solr implementation and Hydra which uses Ruby on Rails, Solr and Blacklight (Fedora, 2014).

Evaluation

In the project proposal it was specified that Dspace would be used as the repository architecture. However, after further analysis, it was determined that Fedora was more appropriate for the needs of the Zamani Data Archive. Dspace did not provide an out-the-box solution and would have required top-down development using the Dspace infrastructure and additional modules. Conversely Fedora Commons allows for bottom-up development with its extensive REST web API which could be used to access the contents of the repository.

Furthermore Fedora has the ability to create additional datastreams for data objects, which could be used to capture the additional file format specific information. This was required due to various file types found in the Zamani data set. Fedora's RDF support also provides a tool for providing structure to the Zamani Data Archive as the relationships between data objects can be define.

The main drawback of using Fedora over Dspace is the lack of an active community, which is evident from the number active repositories found on the ROAR website.

Development Lifecycle

Design Methodology

An iterative design methodology was chosen for this project as the requirements of the Zamani Data Archive could be broken up into sub-components namely, Archive, Public Portal and the Backend. Additionally, each of these components could be created separately.

The Development lifecycle used consists of the following stages in each iteration; planning, design and implementation, testing and evaluation. This design approach was taken to ensure robust software was created that could serve as a core for future improvements and requirements. This is of particular relevance as the Zamani Project plans to extend the Zamani Data Archive functionality in the future.

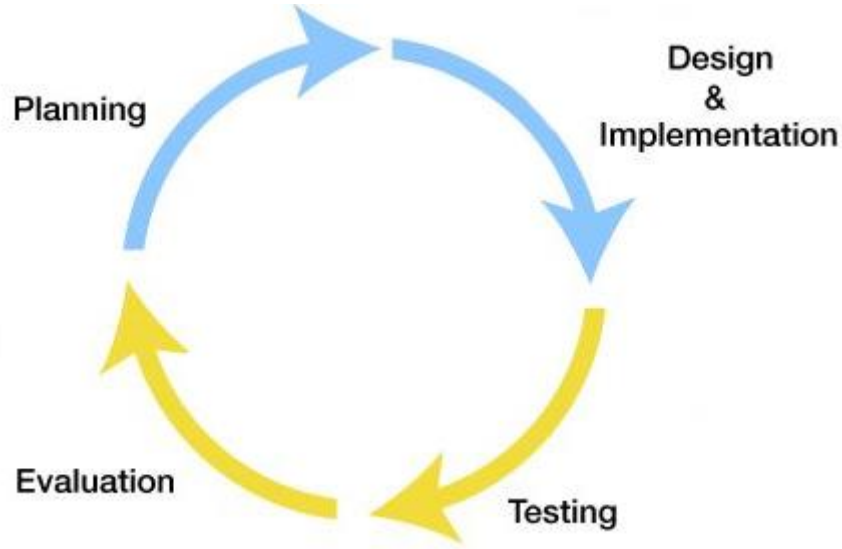


Figure 3: Development Cycle

Planning

Planning was used to gather and refine the requirements for the iteration. As there was a high level of client interaction in the design and development process the requirements or presentation of the project changed over time. The requirements to be addressed were selected at the beginning of each iteration.

In order to better understand each of the components and consider them in a more fine-grain manner, a series of sticky notes were used (seen in Figure 4). This technique was also used to consider the logic required for each component that needed to be developed. Once a task had been completed it was moved to the completed pile (seen in Figure 5). Additionally, flowcharts and use case diagrams were created to provide a high-level overview of components in the system.

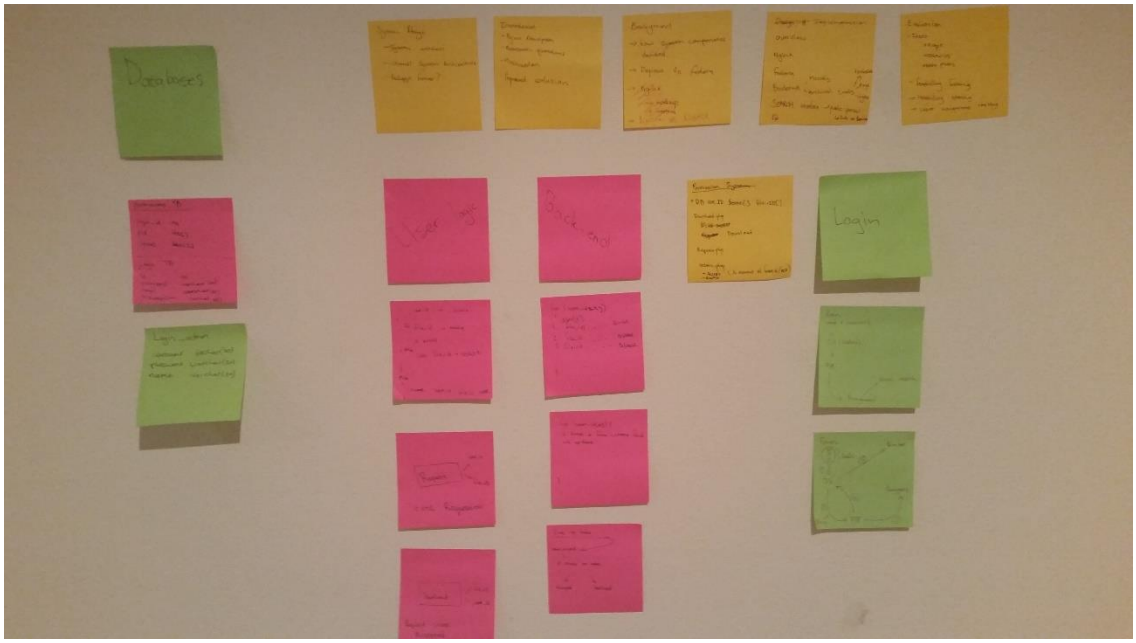


Figure 4: Information and functionality required broken down into smaller components

Figure 5: Completed functionality

Design and Implementation

In order to ensure that the requirements of the project were met and that the Zamani Data Archive offered a high level of usability to the end users, design and implementation of the project happening concurrently. Once initial analysis and design was complete the functionality for the iteration was implemented. The largest challenges were faced in this stage of the development lifecycle. In this document the implementation stage has been thoroughly described with all the configuration of the Zamani Data Archive. In order to better understand the requirements needed for each component use case diagrams and flowcharts were created specifying the different activities for both administrators and users.

Testing and Evaluation

To ensure that each component of the system met the requirements, feasibility testing was used along with formative assessment during each iteration. At the end of each iteration, the artefacts created were evaluated based on whether they fulfilled the requirements of the user and were error free. The evaluation stage also served to gather requirements for the following iterations.

First Iteration

Planning

Components such as the webserver and the repository were needed before the Archival Tools, Request Management and Public Portal could be developed. The webserver and repository were required in the early stages of the project so that live testing could be performed with the search interface. Additionally, the account management functionality was needed so that users could register and login to an account on the Zamani Data Archive. User accounts were also needed before the file permissions of users could be handled.

Dspace was the proposed repository architecture of choice, however after further evaluation (found in the background chapter) it was decided that Fedora would be better suited. Additionally, the Zamani data set was mounted the server and the account management functionality of the data request interface was added allowing users to register an account and login.

For the above reasons these components were prioritised for the first iteration of the development life cycle.

Design and Implementation

Fedora - Archive Architecture

In the project proposal it was specified that Dspace would be used as the repository architecture. However, after further analysis (found in the background chapter) it was determined that Fedora Commons was more appropriate for the needs of the Zamani Data Archive. Dspace did not provide an out-the-box solution and would have required top-down development using the Dspace infrastructure and additional modules. Conversely Fedora Commons allowed for bottom-up development with its extensive REST web API. Furthermore in Fedora all content is manage as data objects, each composed of datastreams that contain either content or metadata data. This allowed the creation of additional datastreams which could be used to store file format specific information. This was required due to various file types found in the Zamani data set.

Fedora was thus in line with the needs of the project as a custom software product was being created for the client. The Fedora configuration consisted of an Apache Tomcat application server in which Fedora is run, as well as a relational database management system (RDMS) to support some of the repository's functions.

PostgreSQL

PostgreSQL was chosen as the RDMS due to the extensibility it provides for handling GIS data which could be used in future work on the Zamani Archival System. PostgreSQL is designed for high volume environments which was needed for the large size of the Zamani data set.

PostgreSQL has the ability to create stored procedures (PostgreSQL, 2014). This functionality was used to create an array search function. The array search function takes a search value (needle) and an array (haystack) and returns the index of the needle in the haystack if it is found. This function was used extensively in the permissions system to look up file permissions for users.

The configuration for the PostgreSQL RDMS only required than a 'fedoraAdmin' role be created and a 'fedora3' database. The statements for creating these entities can be seen in Figure 6.

```
psql -d postgres
```

```
CREATE ROLE "fedoraAdmin" LOGIN PASSWORD 'fedoraAdmin';
CREATE DATABASE "fedora3" WITH ENCODING='UTF8' OWNER="fedoraAdmin";
```

Figure 6: PostgreSQL configuration

Fedora Object Extensible Mark-up Language (FOXML)

FOXML version 1.1 was the metadata format chosen to represent and ingest the Zamani data set in the archive. FOXML is a simple XML format that directly expresses the Fedora digital object model, additionally, it has been performance optimised for Fedora (Fedora, 2005). As FOXML provides a direct expression of the Fedora digital object model is able to handle multiple datastreams. Control groups are used to define where the bytestream content is stored. The 'Inline XML' control group was used so that XML pertaining to the digital object could be stored inline.

Each FOXML record created for the archive consists of four datastreams, namely, DC, RELS_EXT, POSITION and FILE. Tiff files additionally have a CALIBRATION datastream.

The DC or Dublin Core datastream represents the majority of the data relating to files with the other datastreams adding additional information.

```
▼<foxml:datastream CONTROL_GROUP="X" ID="DC" STATE="A">
  ▼<foxml:datastreamVersion FORMAT_URI="http://www.openarchives.org/OAI/2.0/oai_dc/" ID="DC.1" LABEL="Dublin Core Record for this object" MIMETYPE="text/xml">
    ▼<foxml:xmlContent>
      ▼<oai_dc:dc xmlns:dc="http://purl.org/dc/elements/1.1/" xmlns:oai_dc="http://www.openarchives.org/OAI/2.0/oai_dc/">
          <dc:description/>
          <dc:identifier>uctnew:0000009</dc:identifier>
          <dc:title/>
          <dc:coverage>Sudan</dc:coverage>
          <dc:coverage>Musawwarat es-Sufra</dc:coverage>
          <dc:coverage>Great Enclosure</dc:coverage>
          <dc:coverage/>
          <dc:coverage>100</dc:coverage>
          <dc:coverage/>
          <dc:creator>Ruther, Heinz</dc:creator>
          <dc:creator>Held, Christoph</dc:creator>
          <dc:creator>Schroeder, Ralph</dc:creator>
          <dc:creator>Bhurtha, Roshan</dc:creator>
          <dc:creator>Wessels, Stephen</dc:creator>
          <dc:date>2009-08-31</dc:date>
          <dc:format>image/jpeg</dc:format>
          <dc:source>University of Cape Town, Geomatics Department</dc:source>
          <dc:type>Photogrammetric Images</dc:type>
        </oai_dc:dc>
      </foxml:xmlContent>
    </foxml:datastreamVersion>
```

Figure 7: Dublin Core datastream example

The RELS_EXT datastream is a reserved datastream in Fedora used to assert relationships between digital objects in the repository. This datastream uses the RDF authoring style which specifies the relationship between a subject and the object it is bound to. Subjects and objects are referenced by their URIs which identify the digital objects in the repository. The 'isMemberOf' relation was used to preserve the structure of the Zamani data set. An example of such a relation can be seen below where the 'Sudan Collection' is being defined as a member of the Zamani Project Collection.

```
▼<foxml:datastream CONTROL_GROUP="X" ID="RELS-EXT" STATE="A">
 ▼<foxml:datastreamVersion FORMAT_URI="info:fedora/fedora-system:FedoraRELSExt-1.0" ID="RELS-EXT.0" LABEL="RDF Statements about this object" MIMETYPE="application/rdf+xml">
  ▼<foxml:xmlContent>
   ▼<rdf:RDF xmlns:fedora-model="info:fedora/fedora-system:def/model#" xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#" xmlns:rel="info:fedora/fedora-system:def/relations-
    external#">
    ▼<rdf:Description rdf:about="info:fedora/uctnew:0000009">
       <rel:isMemberOf rdf:resource="info:fedora/MusawwaratEsSufraCollection:1"/>
     </rdf:Description>
    </rdf:RDF>
   </foxml:xmlContent>
  </foxml:datastreamVersion>
 </foxml:datastream>
```

Figure 8: RELS_EXT relation example

The relational structure of the archive can be summarised as follows:

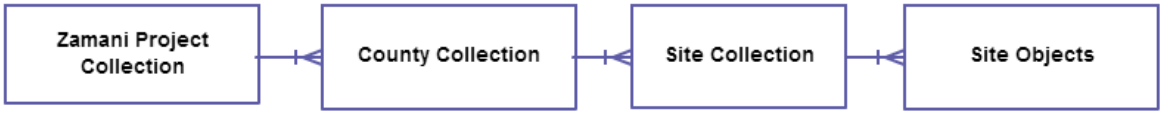


Figure 9: RELS_EXT entity relationships

The POSITION datastream contains the location related data for the record. This datastream was included to provide the needed information for the map based searching in the search interface. Additionally, this information would be displayed in the individual record view.

```
▼<foxml:datastream CONTROL_GROUP="X" ID="POSITION" STATE="A">
 ▼<foxml:datastreamVersion ID="POSITION1.0" LABEL="Position info about this object" MIMETYPE="text/xml">
  ▼<foxml:xmlContent>
   ▼<position>
      <latitude/>
      <longitude/>
      <ellipse/>
    </position>
   </foxml:xmlContent>
  </foxml:datastreamVersion>
 </foxml:datastream>
```

Figure 10: POSITION datastream example

The CALIBRATION datastream was required for TIFF files in the Zamani data set so that additional camera calibration setting could be stored.

```
▼<foxml:datastream CONTROL_GROUP="X" ID="CALIBRATION" STATE="A">
 ▼<foxml:datastreamVersion ID="CALIBRATION1.0" LABEL="Calibration info about this object" MIMETYPE="text/xml">
  ▼<foxml:xmlContent>
   ▼<calibration>
      <date>06/07/2009</date>
      <camera>NIKON D200</camera>
      <decenteringP1>-7.18639e-005</decenteringP1>
      <decenterinP2>-1.36250e-006</decenterinP2>
      <distortionAffineB1>-2.56338e-004</distortionAffineB1>
      <distortionAffineB2>1.65759e-004</distortionAffineB2>
      <distortionRadialK1>2.10732e-004</distortionRadialK1>
      <distortionRadialK2>-2.54109e-007</distortionRadialK2>
      <distortionRadialK3>1.42781e-010</distortionRadialK3>
      <lengthFocal>28.1331</lengthFocal>
      <pixelSizeX>0.007</pixelSizeX>
      <pixelSizeY>0.007</pixelSizeY>
      <principalPointX>0.0134</principalPointX>
      <principalPointY>-0.5442</principalPointY>
      <sizeX>3872</sizeX>
      <sizeY>2592</sizeY>
    </calibration>
   </foxml:xmlContent>
  </foxml:datastreamVersion>
 </foxml:datastream>
```

Figure 11: CALIBRATION datastream example

The FILE datastream provided information relating to the actual file that the digital object represented. This data stream included the filename, file size and file location. This datastream was created to provide a mechanism for accessing the files using the information in the repository. This was of particular importance when considering that the Zamani Archival System was required to allow users to download the files from the Zamani data set.

```
▼<foxml:datastream CONTROL_GROUP="X" ID="FILE" STATE="A">
  ▼<foxml:datastreamVersion ID="FILE1.0" LABEL="File info about this object" MIMETYPE="text/xml">
    ▼<foxml:xmlContent>
      ▼<file>
          <size>651.1 KB</size>
        ▼<client_path>
            C:\Zamani\Zamani-Core\Sudan\Musawwarat\Images\submitted\Photogrammetric Submit\SUD_7212_s.jpg
          </client_path>
        ▼<server_path>
            /mnt/shares/zamani/Zamani-Core/Sudan/Musawwarat/Images/submitted/Photogrammetric Submit/SUD_7212_s.jpg
          </server_path>
        </file>
      </foxml:xmlContent>
    </foxml:datastreamVersion>
  </foxml:datastream>
```

Figure 12: FILE datastream example

Fedora Installation

The Fedora installation first required that Java JDK 6 be installed and that and that the JAVA_HOME environment variable be set to the location of the Java JDK. Additionally, the FEDORA_HOME was set to the location of the Fedora installation in case any command line utilities were needed. The PATH environment variable required the addition of the Java and Fedora bin directories. In the case of Fedora, both the client and server bin directories were required.

GSearch was included in the installation which provides functionality for indexing Fedora FOXML records as well as searching through the created indexes. Additionally, it allows you to plugin a selected search engine. Solr was the chosen search engine for the Zamani Archival System. This provides the Fedora repository with more powerful mechanism to query the content of the repository.

Nginx – Webserver Configuration

Nginx, a HTTP and reverse proxy server, is an alternative to Apache and was chosen as the Webserver for the Zamani Archival System. Nginx handles threads differently to Apache (Nginx, 2014). A new process is not created for each Web request. Instead it comprises of worker processes, which are each handled by a single-thread. Each worker process is able to handle thousands of concurrent connections. Furthermore because of the way Nginx handles threads it uses less CPU and memory resources. This configuration was chosen to ensure that the Zamani Archival System was scalable and could handle a large number of requests if needed.

Nginx comprises of modules which are included at compile time (Nginx, 2014). The configuration used for the Research Data Archive can be seen in Figure 13. The following additional modules were used; GZIP module, Mail module and Mod_zip. The initial Nginx installation was done with version 1.1.5 as version 1.1.6 (the latest version) gave a compilation error in the ngx_http_zip_headers.c file stating that variables were set but unused.

Nginx is able to compile PHP natively. However it was decided that the PHP FastCGI Process Manager (PHP-FPM) daemon would be used to handle PHP requests instead, as this would reduce the load on the Webserver as this thread could generate the dynamic content for the Frontend and Backend. The

PHP-FPM package includes the ability to scale up the number of PHP processes that are running and handling requests based on load (PHP, 2014). This provides for better resource management on the server. A more detailed explanation of the modules and configuration used can be found in Appendix B1.

```
--prefix=/etc/nginx --conf-path=/etc/nginx/nginx.conf --error-log-path=/var/log/nginx/error.log --http-client-body-temp-path=/var/lib/nginx/body --http-fastcgi-temp-path=/var/lib/nginx/fastcgi --http-log-path=/var/log/nginx/access.log --http-proxy-temp-path=/var/lib/nginx/proxy --http-scgi-temp-path=/var/lib/nginx/scgi --http-uwsgi-temp-path=/var/lib/nginx/uwsgi --lock-path=/var/lock/nginx.lock --pid-path=/var/run/nginx.pid --with-debug --with-http_addition_module --with-http_dav_module --with-http_gzip_static_module --with-http_realip_module --with-http_stub_status_module --with-http_ssl_module --with-http_sub_module --with-ipv6 --with-sha1=/usr/include/openssl --with-md5=/usr/include/openssl --with-mail --with-mail_ssl_module --add-module=/zamani_BACKUP/mod_zip-1.1.5
```

Figure 13: Nginx compile configuration

Data Request Interface - Account Management

The first component of the data request interface that was developed was the account management functionality. The account management functionality would allow users to create an account on the Frontend of the Zamani Archival System. After meeting with the Zamani team was determined that a terms and conditions for the Zamani Archival System was needed for legal purposes. Users would be required to accept these conditions before creating their account. This would ensure that all users had agreed to any copyright restriction on the Zamani data set before they were able to request or download any content from the Archive. The account would be used to keep a record of the user's email address so that they could be informed of any updated permissions. Furthermore the account management functionality required validation of all fields to prevent any potential SQL injection attacks. A high-level overview detailing the use cases and processes of the account management functionality can be seen in Figure 14 and 15.

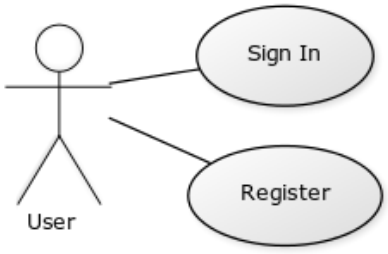


Figure 14: Use case diagram for account management

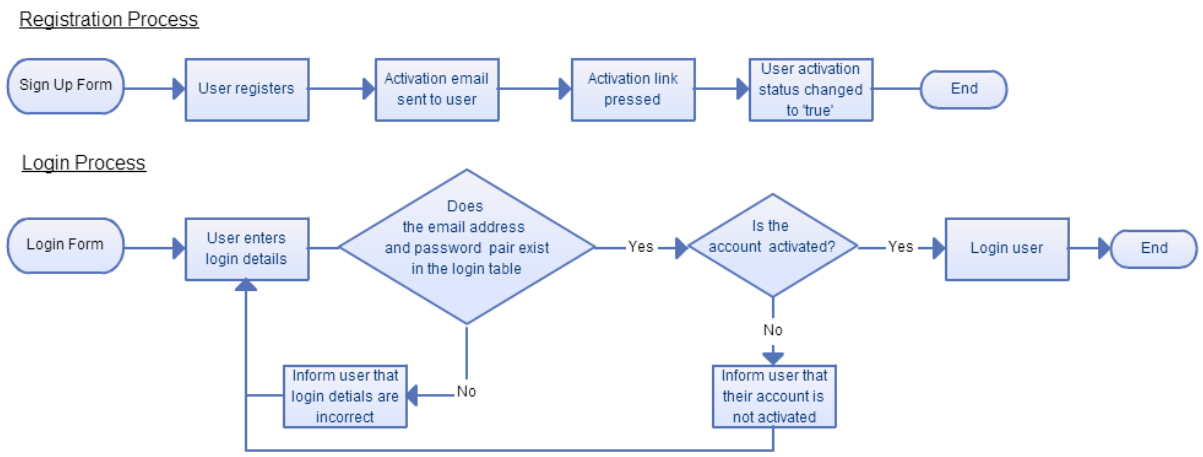


Figure 15: Flowcharts for registration and login processes

Database Design

This section details the tables found in the 'Admin' database that was created in the PostgreSQL RDMS. This database was created to house all the components of the permissions system. The database consists of three tables (seen in Figure 16).

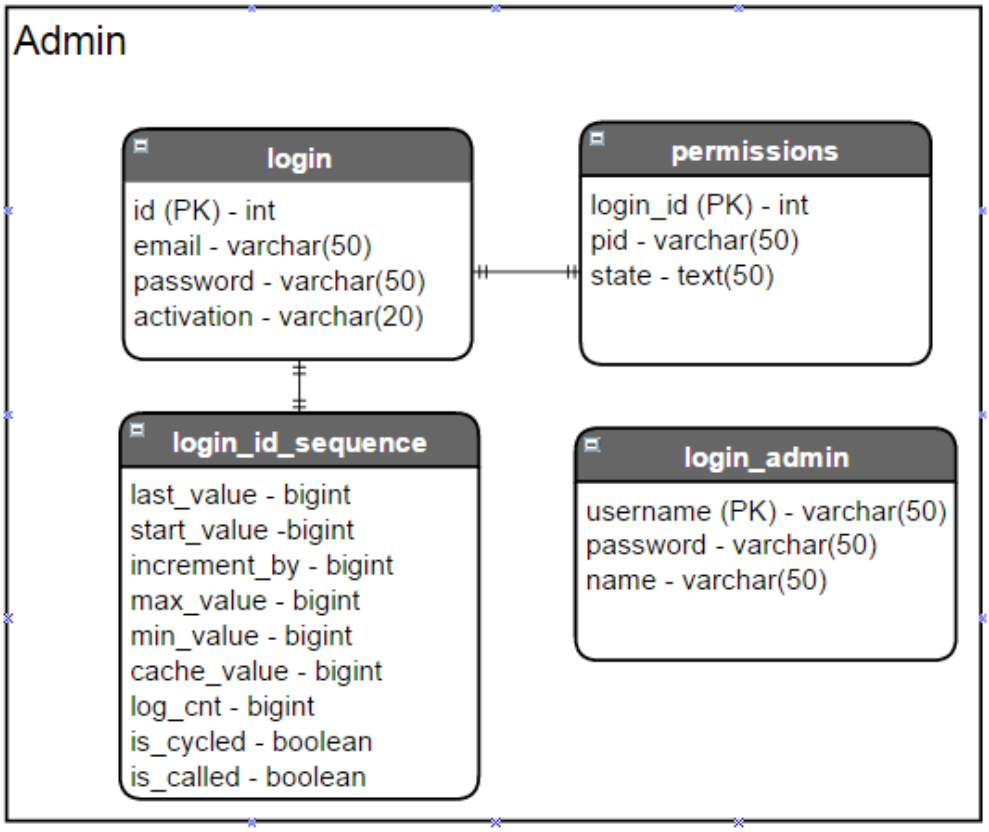


Figure 16: Admin database entity diagram

Login table

The login table was created to store the account details of a user. To ensure that each user has a unique id, the login_id_sequence was created. This sequence is responsible for generating unique identifier for the login table and provides the same functionality as MySQL's AUTOCREMENT feature.

The activation field was created to store the unique activation string that a user is given to activate their account. Once an account is activated, the value 'true' is stored in this field.

Permissions table

This table stores a permissions list for each user. The 'pid' and 'state' array fields store permissions for each file the user has requested. This is done by adding a corresponding personal identifier (PID) and state to the same index of the in both the 'pid' and 'state' array fields. The possible values of the state field are; 'requested', 'accepted' and 'declined'. These three values represent the states that a user request for a file can have. For example the first record requested by a user, has its PID placed in pid[0] with the 'requested' state in state[0].

The Data Request Interface is responsible for adding new requests into the permissions table with the 'requested' state. The Request Management functionality is then responsible for either changing the state to 'accepted' or 'declined' at the discretion of the administrator.

Login_admin table

This table was created to store the details of the administrators of the Zamani Archival System. It contains an additional 'name' for the administrator's given name. This is used for easy-to-read logging of which administrators has updated the file permissions for a user. These login details were kept separate from the end-user details to ensure only administrators could modify this table.

Account Management – Important Processes, Scripts and Functions

Index.php

This script contains the Register, Login and Terms and Conditions forms which can be found in Appendix A1. These forms pass requests to the scripts described below.

In order to display these forms a JQuery plugin called LeanModal was used. This plugin allows you to define a modal container which can contain any html objects. LeanModal also defines a modal trigger, which when clicked triggers the modal container to pop-up on the screen and makes the background opaque. The plugin is used by calling the 'leanModal' function, which takes three options. The first is the vertical position of the modal element in relation to the document body. The second options specifies the overlay opacity and the final option allows for an optional close button to be added to the modal. The call to the leanModal function in the index.php script can be seen below:

```
$("#requestModal_trigger").leanModal({top : 150, overlay : 0.6, closeButton: ".modal_close"
});
```

Figure 17: LeanModal function call used to create a pop-up container

The modal used for the account management allows the user to navigate between the login, registration form and the terms and conditions page. Screenshots of the account management functionality can be seen in the Appendix A2. All the forms in the modal contain validation to ensure that a valid email address has been added and additionally that all the required fields have been filled in.

Register.php

This script is sent an AJAX post request containing the user's email address and password from the registration form. The email is first validated by this script to ensure that a user has not already registered with the same email address. If the email address of the user is found to be unique, an activation code is generated for the user. This is used to create a unique activation link for the user, which is sent to their registration email address. The unique activation value is then inserted into the login table along with the activation code.

Activate.php

When an activation link is clicked on by a user this script uses the unique activation code at the end of the link to look up the user in the login table. If they are found, their activation field value is set to 'true'. Once this script has been run for a given user's activation code they are then able to login in to their account on the Zamani Data Archive.

checklogin.php

This script is sent an AJAX post request containing the user's email address and password from the login form. The script first checks if the email address password pair exists in the login table. If no match is made, the user is notified that their email address or password is wrong. If a match is found an additional test is done to see if the account has been activated. If it has not, the user is informed that they should activate their account. Otherwise, the user is logged in and PHP session and variables are created to store their id, email address and list of PIDs from their permissions table record if they have one.

When the 'id' and 'email' session values are set, the user is deemed 'logged in'. When these session variables expire, the user is required to login again.

Logout.php

This script allows users to logout of their account on the Zamani Archival System and does so by unsetting any session variables and then destroying the current session. These operations are performed by the session_unset and session_destroy function respectively.

Testing

Overview

The feasibility testing for this iteration required that Nginx and Fedora were accessible remotely and that a user was able to create an account on the Zamani Data Archive and login in. The results of these tests can be seen below.

Nginx

The screenshot below shows that the Nginx configuration was working correctly as this is the default index page served by Nginx. This indicated that the Nginx webserver was ready for live testing.

Figure 18: Nginx default index page

Fedora

The screenshot below shows that the Fedora repository was remotely accessible on Nala, as this is the default description page. This confirmed that the development team would be able to perform further testing in a 'live' environment.

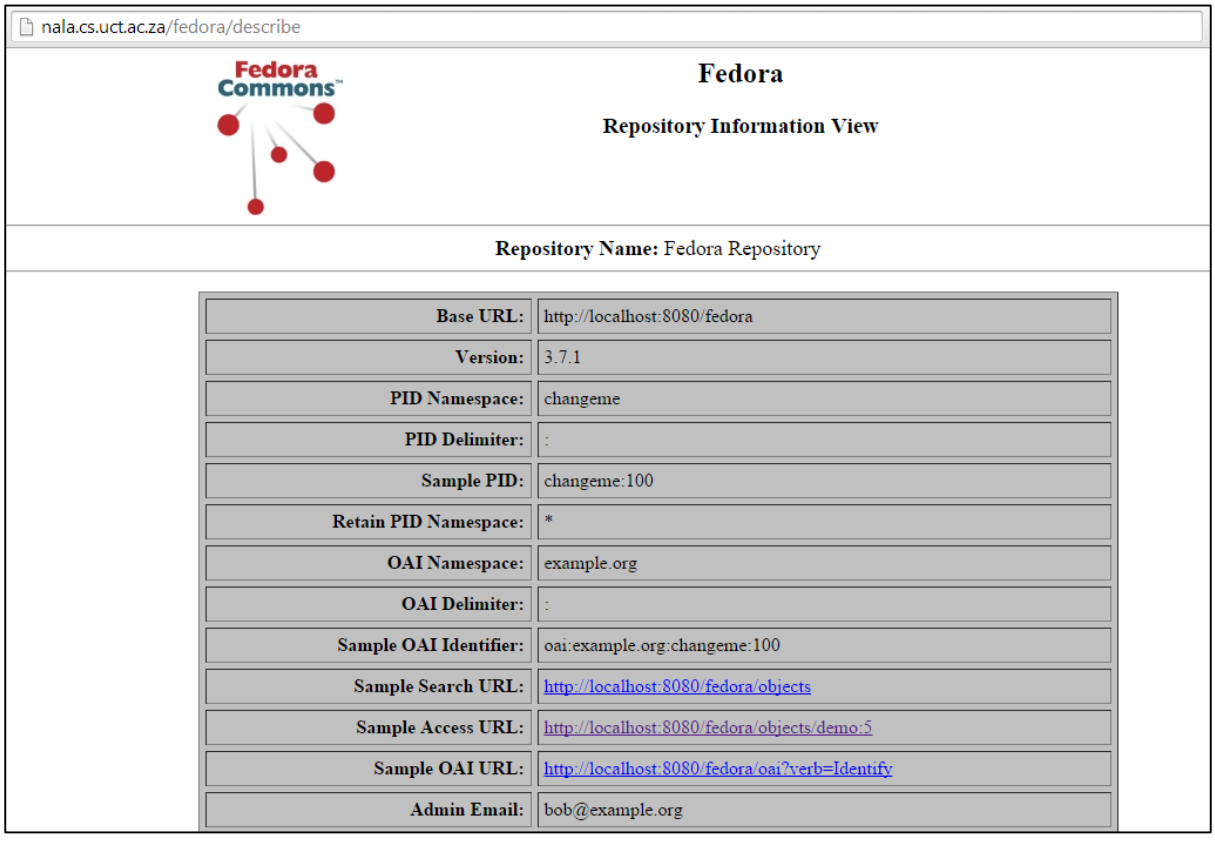


Figure 19: Fedora default description page

Account Management

The screenshots below show the process of a successful registration for a user as well as the ability to login to the Zamani Data Archive. The user first submits the registration form with their details. They are then sent an activation email to their registration account. Once they have clicked the activation link they are redirected to the index page and are able to login successfully.

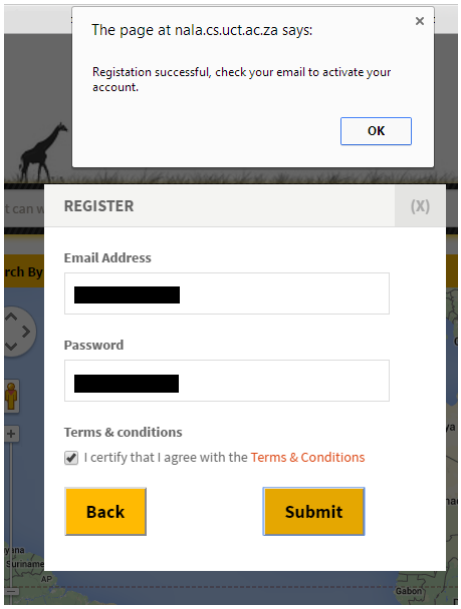


Figure 20: Successful User Registration with activation alert

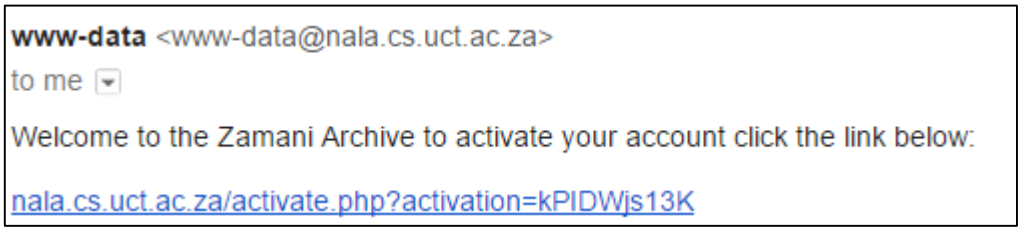


Figure 21: Email received containing an activation link

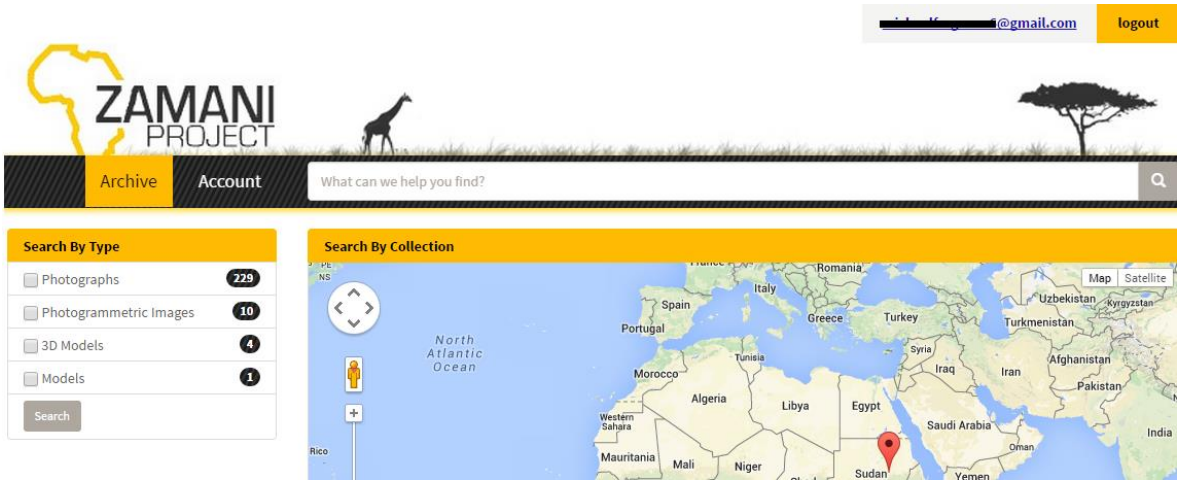


Figure 22: Successful user login

Evaluation

This iteration provided the Zamani Archival System with its core infrastructure: the Research Data Archive consisting of Nginx and Fedora. Additionally, the account management aspect of the permissions system was created.

The lack of ability to change your password if it has been forgotten was noted as a missing feature in account management functionality and was added to the backlog of the second iteration.

The software stack created by this iteration allowed for the much needed live testing with Fedora and provided a platform for the creation of the Search Interface component of the Public Portal.

Second Iteration

Planning

The primary goal of this iteration was to develop the account page and request components of the Data Request Interface as well as the Request Management functionality and interface. The Data Request Interface would allow a user to request files, view request states and download files.

The Request Management functionality would allow the admin to change file permissions for user requests. Additionally, a log was required stating which administrator had updated a user's file permissions for auditing purposes.

The information required for an administrator to identify a file was discussed with the Zamani team. The fields decided upon determined the content of the Request Management page detailed in this iteration.

Additionally, the ability for a user to change their password would be added to the existing account management functionality.

Design and Implementation

Data Request Interface – Account Management

During the evaluation of the first iteration it was found that the functionality for a user to change their password was required in case it was forgotten. In order to add this functionality a link was created on the login form to a change password form which allows a user to change their password by entering their email address. A high-level overview of the change password functionality can be seen in Figures 23 and 24.

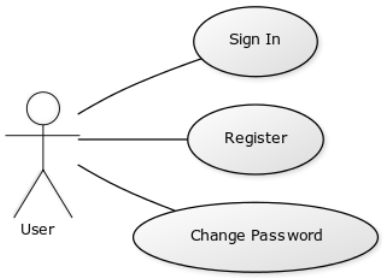


Figure 23: Updated use case diagram for account management

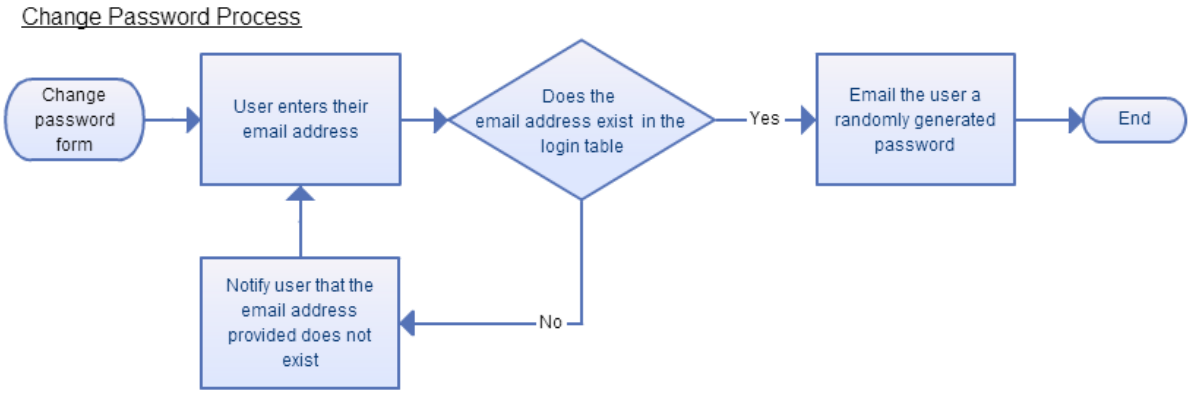


Figure 24: Flowchart of the change password process

Account Management – Important Scripts and Functions

changepass.php

This script first checks if the entered email address exists in the login table. If it does not the user is notified that the email address was not found. If the email address is found, a randomly generated password is created and used to replace the current password for the user. An email is then sent to the user's registration email containing their new password which they can now login with.

Data Request Interface – Account Page

The account page serves as the central location for a user to view their current file requests and download files that have been accepted. This page also includes additional information relating to the archive namely: a list of recommended software and the terms and conditions of the Zamani Data Archive. All these components of the account page are accessible through tabs in the control panel found on the left of this page. This was done to ensure that all the additional information relating to the Zamani Data Archive could be found in one place.

The terms and conditions tab was included for user to reference at a later point if they are unclear of the terms and conditions they initially agreed to when registering an account on the Zamani Data Archive. The recommended software was included after meeting with the Zamani team in order to provide users of the archive with links to software to interact with the data found on the archive. This includes software for dealing with VRML, PLY (normal detail 3D model), QSplat (high detail 3D model), Videos and Laser scans.

Users may download a single file or a batch of files from this interface. If multiple files are selected for download, a ZIP archive is dynamically generated by the Nginx Mod_zip module. If only a single file is requested it is served to the user directly.

This page is only displayed in the navigation bar when the user is logged in. This rule is enforced as users may not access the data in the archive directly until they have registered and agreed to the terms and condition of the archive. There are multiple paths to the account page allowing a user to more easily navigate to the page. The user can either select the account navigation bar item or they can click on their email address displayed at the top of the page. A high-level overview of the download process can be seen in Figure 24.

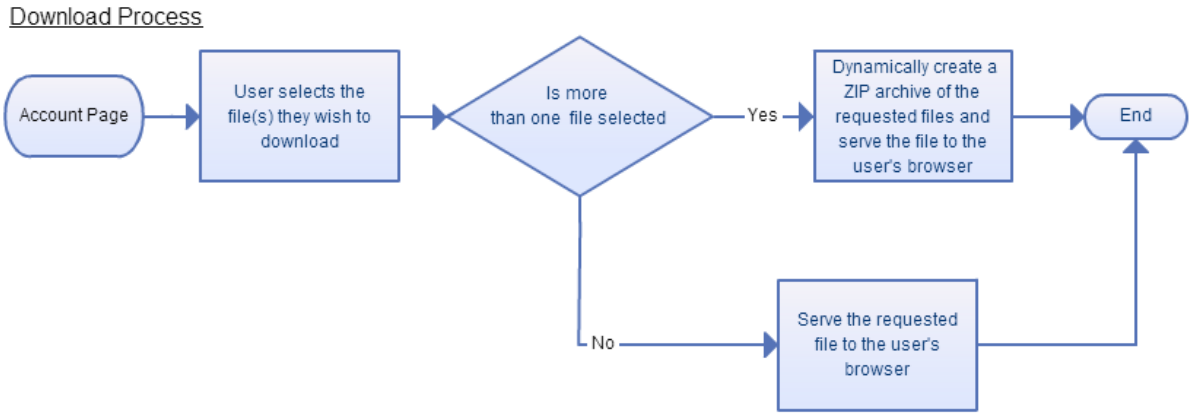


Figure 25: Flowchart of the download process

Account Page – Important Scripts and Functions

download_user.php

This script is responsible for dynamically generating a user's account page. This is done by querying the permissions table with the user's email address and processing the permissions list retrieved For each PID in the user's permissions list, two cURL requests are made, one to Solr to get the basic information for the file (file format, collection and county) and a request to the FILE datastream to get the file name.

The state of the file is evaluated and an appropriate tag containing the current state for the file is displayed. Files with the requested status are displayed with a pending tag. The accepted and declined tags are represented with a tag named the same. If the state of the request for a user is accepted a checkbox is displayed for the line item allowing the user to check it if they wish to download the file.

An additional request is made to the getScaledImage.php script which creates a scaled thumbnail of the record in the archive. Where images are not available (such is the case for some 3D models) an 'IMAGE NOT AVAILABLE' default image is displayed. An example call to the getScaledImage.php can be seen in Figure 24.

```
<img
src="http://nala.cs.uct.ac.za/zamaniProject_archive/inc/getScaledImage.php?w=100&amp;h=100&amp;mimeType='A'&amp;pid='B'" >
```

Figure 26: Request for a file with mime type A and a PID of B.

Once the user has selected the file/s they wish to download, they then press the Download File(S) button which sends a list containing the selected file PIDS to the download.php described in the next section.

download.php

This script takes a list of PIDs and determines the length of the list. If the list only contains one value the request is handled by the Nginx X-Accel-Redirect header which is passed the file location from a cURL request to the FILE datastream.

During the development of batch file downloads it was noted that file names in the Zamani data set contained spaces. This is not the naming convention in Ubuntu, the operation system running on Nala.

Additionally, it was noted that version 1.1.6 added the use of URL-encoded string for file names and paths, this allowed the Mod_zip module to work with file names containing spaces. It was thus decided that a working versions of Mod_zip 1.1.6 was needed to handle batch file requests for the archive. After looking at the Github repository (https://github.com/evanmiller/mod_zip/) for the module it was noted that updates had been done prior to the release of version 1.1.6 that fixed the compilation error. After the ngx_http_zip_headers.c file had been replaced with the most recent release from the Github repository the binary for Nginx was recompiled with the Mod_zip version 1.1.6. This did not require any additional reconfiguration as a recompilation of Nginx simply replaces the binary used by the server. This meant that all existing Nginx configurations could remain the same. To use the new binary the Nginx server was shutdown before compiling the new binary. Once the binary was replaced the server was started again.

If multiple files are requested, a string is built containing the CRC32 checksum of the file, file size, and url-encoded file path and name. Each of the requested files is then placed on new line with these parameters. Once the entire string has been build it is printed to the response body of the HTTP request. The X-Archive-files:zip header is then used to handle the request which dynamically creates a ZIP archive by scanning the response body of the HTTP request for a list of files with additional attributes which are retrieved and encoded in order. Once the ZIP archive has been created it is served to the user's browser by Nginx.

Data Request interface – Request Button

In order to allow a user to request a record, a request button has been integrated into the Search Interface metadata view. The button has been placed at the top of the view to be easily identified.

Before a user can use the request button, they first need to be logged in. If the user presses the request button and they are not currently logged in, the login window will appear indicating the need to be logged in. This was done to enforce that the user is logged in and has agreed to the terms and conditions of the Zamani Data Archive before requesting files.

Once the user is logged, a check is made to see if the user has already requested the file, if this is the case, a tag displaying the current state of the request is displayed instead of the request button. This was done to allow users to see which files they had requested by navigating to the metadata view of for a record. Additionally, it prevented users from requesting the same file multiple times. If the file has not been requested the request form is displayed which allows the user to specify the reason for their request. A high-level overview of the file request process can be seen in Figure 26.

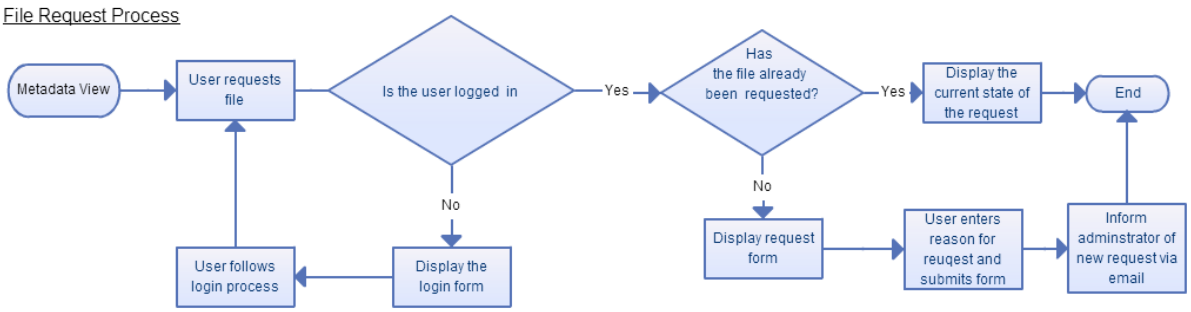


Figure 27: Flowchart of the file request process

Request Button – Important Scripts and Functions

login_test.php

This script has two functions. To determine if the user is logged in when they press the request button and whether they have already requested the file. If the user is not logged in, the login form is triggered when the request button is pressed. If the user has already requested the file then the current state of the file is displayed and the request button is hidden from view.

download_request.php

When a user first requests a file this script creates a permissions record for the user containing in the permissions table. The PID of the newly requested file is placed in the first index of the pid array and a 'requested' state in the first index of the state array. If the user has previously requested files, the newly requested file PID and state are appended to the pid and state array fields as described above.

Additionally, the user's email address, the reason for the request as specified in the request form and a time stamp are emailed to the primary administrator for the system to inform them of the new request. The primary administrator for the system is specified in this file.

Request Management

The Request Management functionality was created to provide the administrator of the Zamani Data Archive with a tool to manage file requests. This functionality is provided by the download management page. This interface separates requests by user and only displays the files that have the requested state for each user. Once the PIDs of the requested files have been obtained the download management page is dynamically generated in a similar manner to the account page mentioned above. The administrator is directed to the download management page once they have logged in. This has been done as request management is the most frequent functionality the administrator will use in backend of the system. A high-level overview of the request management process can be seen in Figure 28.

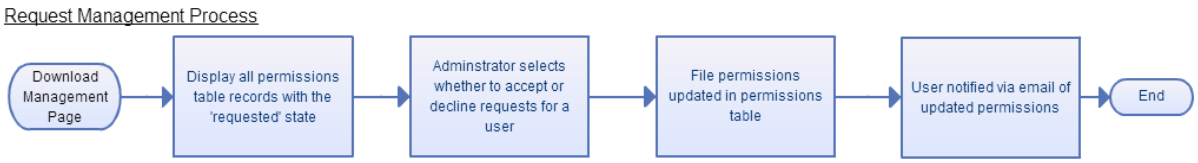


Figure 28: Flowchart of the request management process

download_management.php

This script is used to dynamically generate the download management page, which provides the request management interface for the administrator of the Zamani Data Archive. This is done by requesting the records from the permissions table for each user. Only requests that currently have a requested state are displayed on this page.

For each PID with a requested state in the user's file permissions list, two cURL requests are made, one to Solr to get the basic information for the file (file path, type, format, collection, country and date) and a request to the FILE datastream to get the file name.

A show/hide button was included so that the administrator would be able to see an overview of the current file requests on the Zamani Data Archive. This button expands a table containing all the file requests for an individual user. Additionally, a badge containing the number of requests for the user is displayed in the top left hand corner, to allow the admin to see the number of requests from each user without first expanding the table.

An accept all and decline all check box was included for each user to speed up the process of changing user permissions. Once one of these buttons is clicked the corresponding option is chosen for each request by the user. The admin can then modify the individual records as per their needs. A screenshot of the download management page can be seen in Appendix A2.

Once the administrator has selected their desired permission modifications for a user, they then press the update permissions button which calls the download_request.php script described in the next section. This script is used to update the permissions for the user.

update_permissions.php

This script is called by the download management page to update the permissions for a user. This is done by first looking up the index of the PID in the pid array field to find the corresponding state in the state array field that needs to be modified. This is done using the array_search function created in the PostgreSQL database described in the section relating to PostgreSQL above. Once the index of the PID has been found the corresponding index in the state array field is updated to either accepted or declined as specified by the administrator.

As each file permission is updated a string is built containing the file name, new state and a timestamp. Additionally, the number of accepted and declined requests are tallied as well as the total number of updated permissions. This string serves as the the body of an email which is sent to the user to notify them that their file permissions have been updated.

This script also creates a log of all permission updates. The log contains the name of the administrator who updated the file permissions for the user, the user's ID in the login table, the PID of the file and a timestamp. This feature was created to allow the Zamani Project team to determine which team member had update a user's permissions, should this be required for auditing reasons.

Testing

Overview

The feasibility testing for this iteration required the inclusion of functionality to allow a user to change their password. Additionally, that a user was able to request files, view request states and download either single files or batch downloads. The primary administrator was required to be notified via email of any new requests. The Request Management functionality was required to allow an administrator to update the file permissions. The user was also required to be notified via email of these changes in their permissions. The results of these tests can be seen below.

Change Password

The screenshots below show the process of a user entering their email address in order to change their password and the email they received which includes their new password.

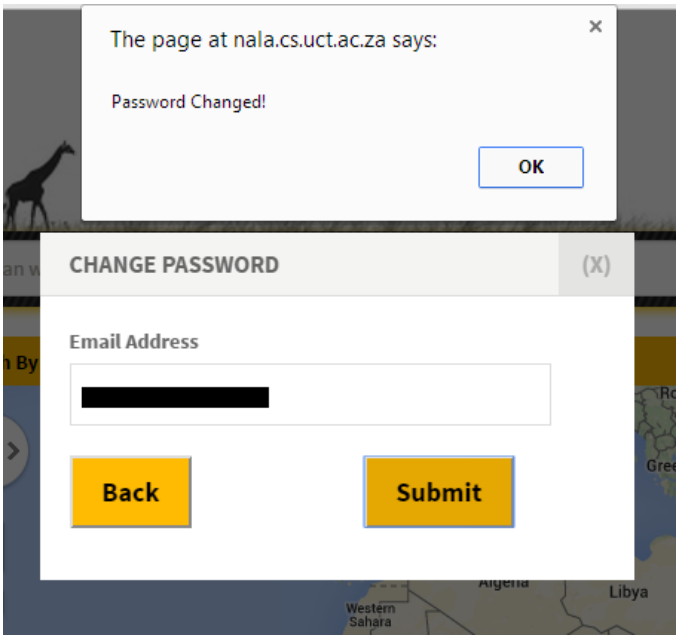


Figure 29: Successful password change

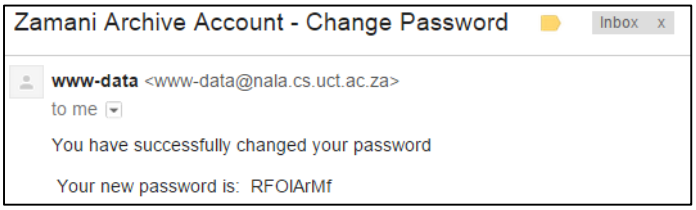


Figure 30: Email containing the user's new password

Data Request Interface

Request Button

The screenshots below show the request file form, which allows the user to request a file with a reason as well as the email sent to the primary administrator of the Zamani Data Archive when a request is made. The third screenshot shows a file that has already been requested.

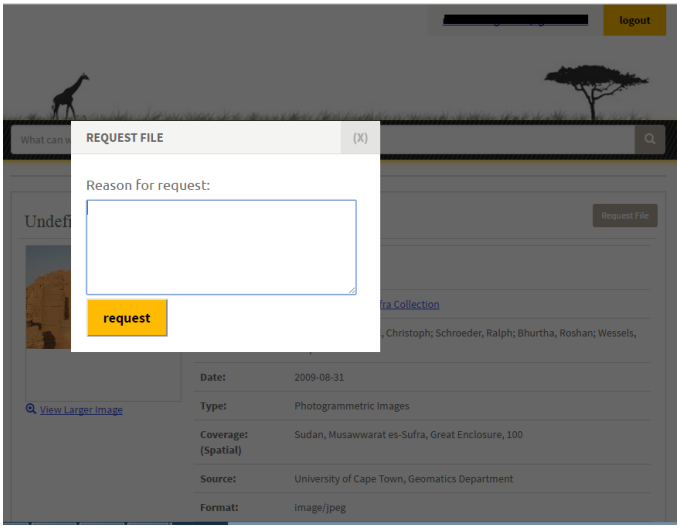


Figure 31: Request file form available after successful login

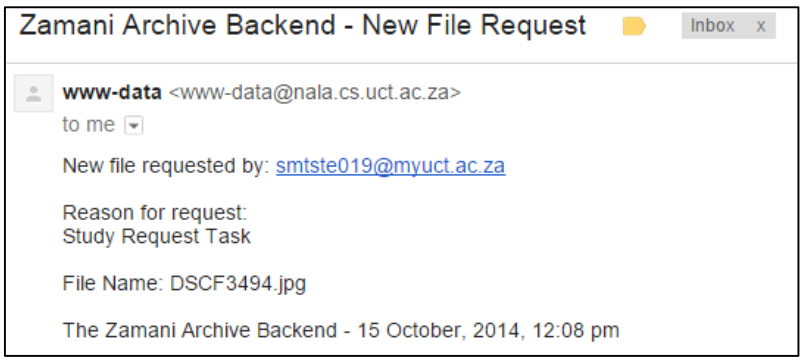


Figure 32: Email to notify the administrator that these is a new file request

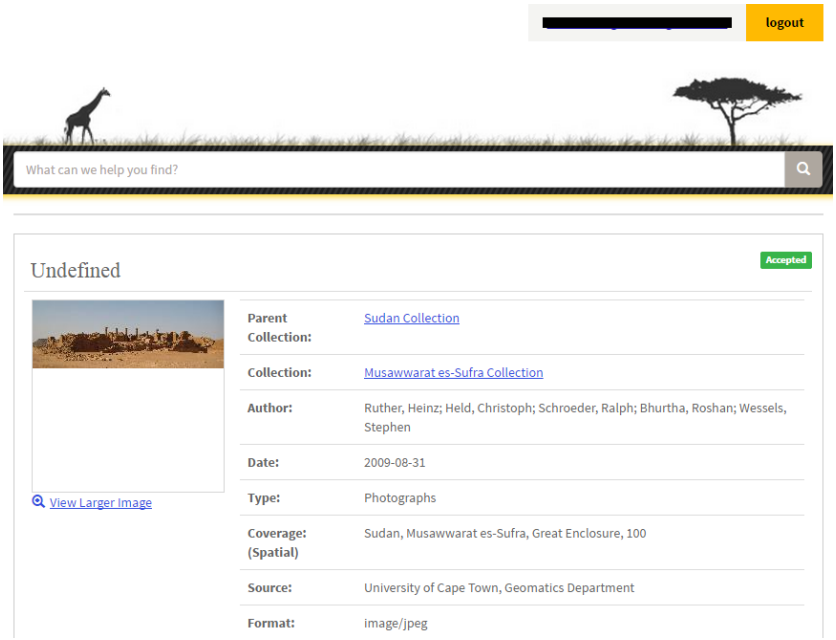


Figure 33: Metadata view of a file that has been accepted

Account Page

The screenshots below show the process of a user viewing their requests in the account page, downloading a single file as well as a batch downloads and the content of the ZIP file downloaded.

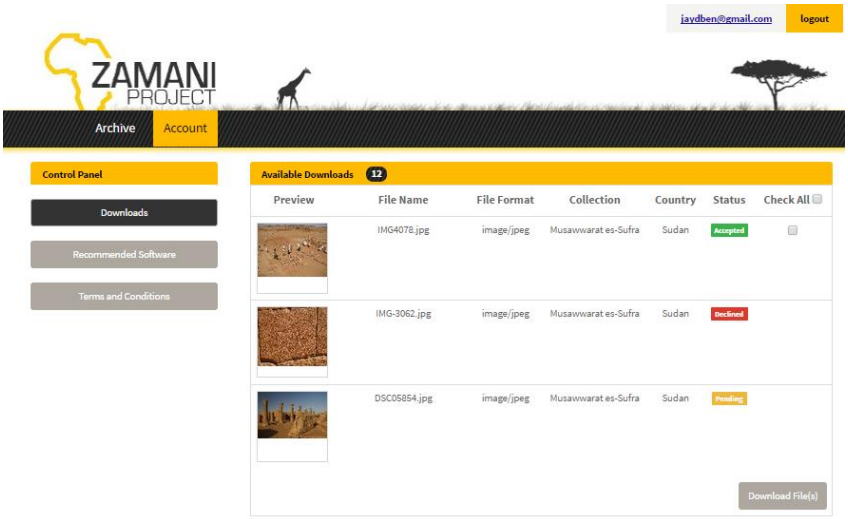


Figure 34: Account page showing the different request states a user can have

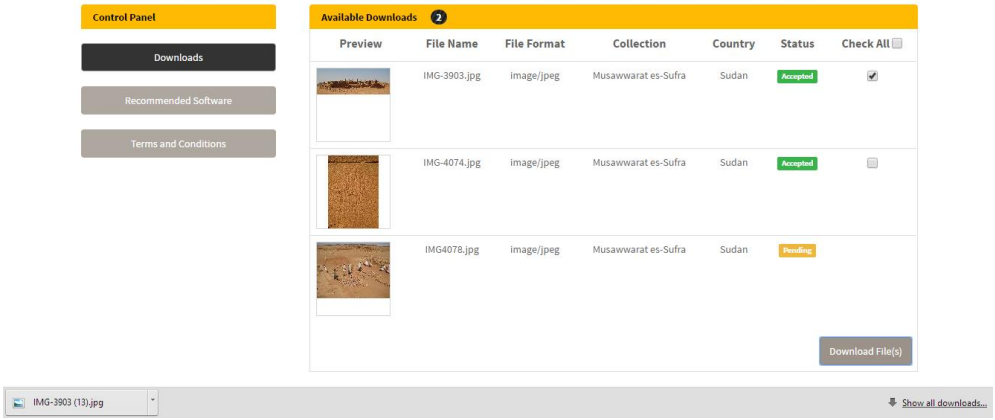


Figure 35: Single file download

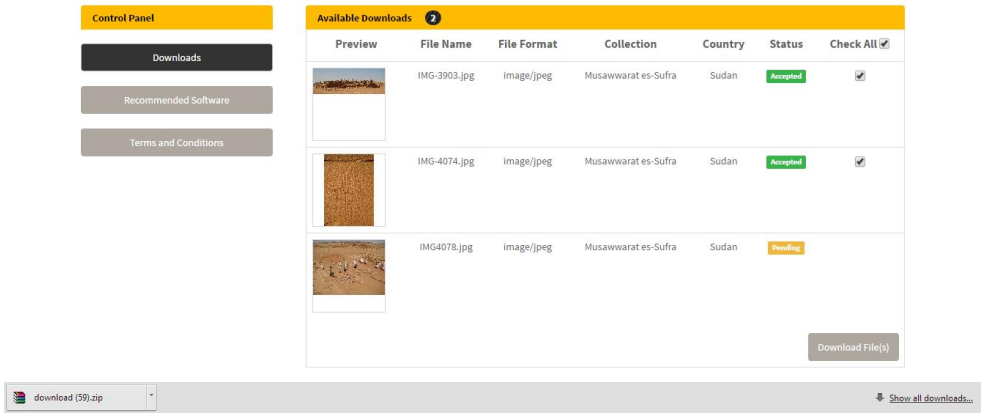


Figure 36: Batch file download

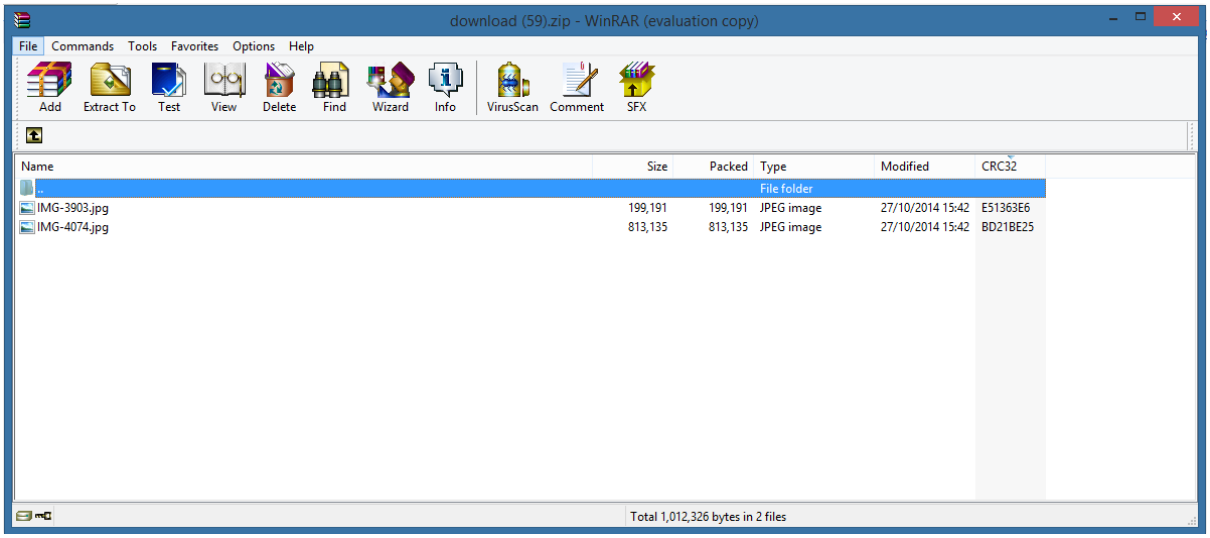


Figure 37: Contents of ZIP archive confirming the download was successful

Request Management – Download Management Page

The screenshots below show the download management page where an administrator can view the current file requests by users in the Zamani Data Archive. Both the expanded and minimized view of a user's requests can be seen. An extract of the permissions.log and the email sent to the user when their file permissions are updated has been included.

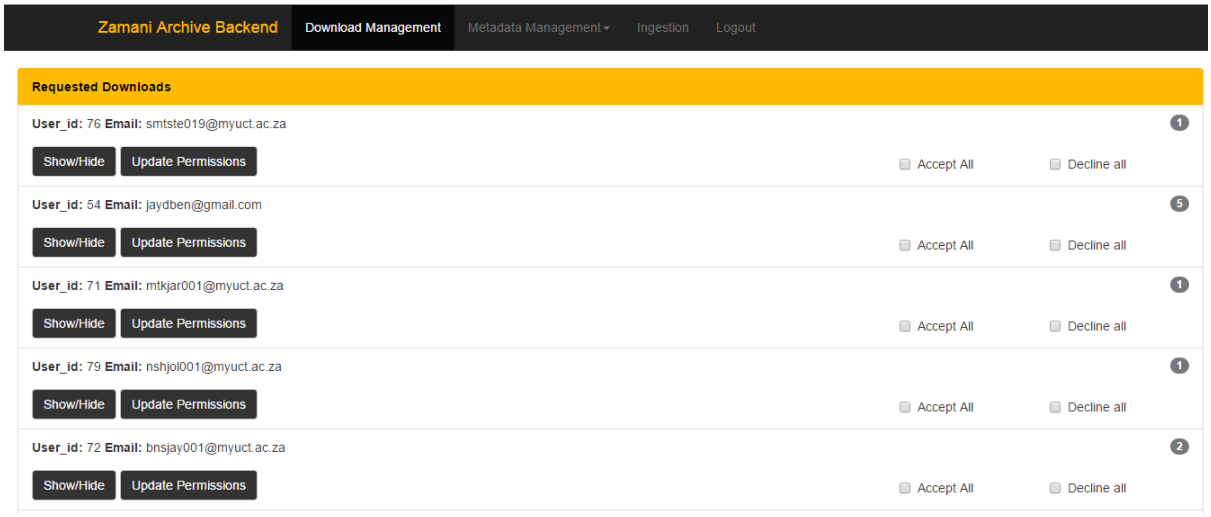


Figure 38: Download management page showing the currently requested files for various users from the Usability study

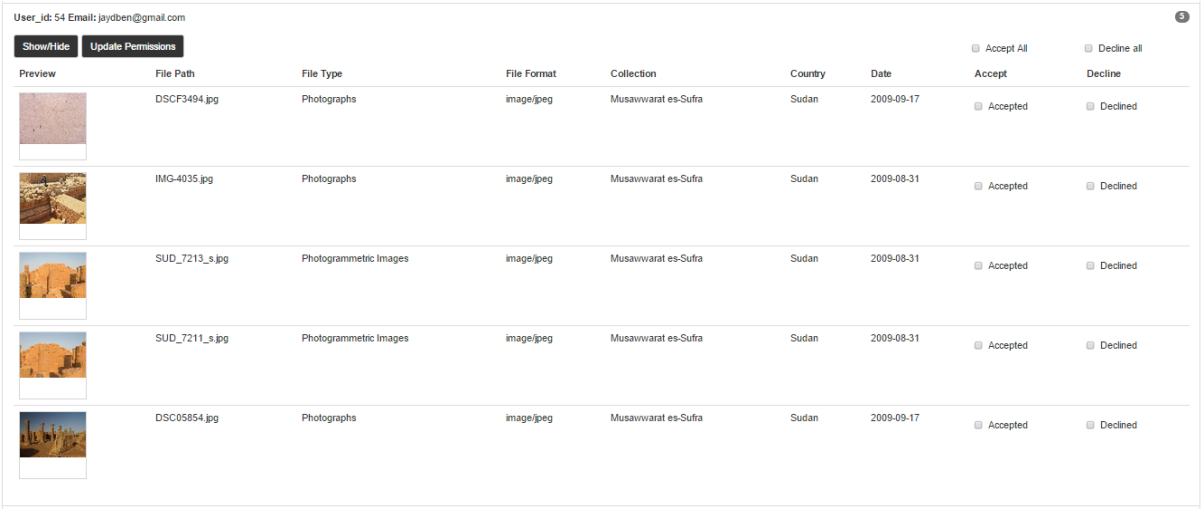


Figure 39: Expanded view of an individual user's requests

```
Admin: study2014 User_id: 71 Pid: uctnew:0000099 State: 'accepted' Time: 14 October, 2014, 2:15 pm
Admin: study2014 User_id: 71 Pid: uctnew:0000117 State: 'accepted' Time: 14 October, 2014, 2:15 pm
Admin: study2014 User_id: 71 Pid: uctnew:0000015 State: 'accepted' Time: 14 October, 2014, 2:15 pm
Admin: study2014 User_id: 71 Pid: uctnew:0000173 State: 'declined' Time: 14 October, 2014, 2:15 pm
Admin: study2014 User_id: 73 Pid: uctnew:0000224 State: 'accepted' Time: 14 October, 2014, 2:39 pm
Admin: study2014 User_id: 73 Pid: uctnew:0000015 State: 'accepted' Time: 14 October, 2014, 2:39 pm
Admin: study2014 User_id: 73 Pid: uctnew:0000005 State: 'accepted' Time: 14 October, 2014, 2:39 pm
Admin: study2014 User_id: 73 Pid: uctnew:0000161 State: 'declined' Time: 14 October, 2014, 2:39 pm
Admin: study2014 User_id: 74 Pid: uctnew:0000099 State: 'accepted' Time: 14 October, 2014, 3:58 pm
Admin: study2014 User_id: 74 Pid: uctnew:0000011 State: 'accepted' Time: 14 October, 2014, 3:58 pm
Admin: study2014 User_id: 74 Pid: uctnew:0000005 State: 'accepted' Time: 14 October, 2014, 3:58 pm
Admin: study2014 User_id: 74 Pid: uctnew:0000196 State: 'declined' Time: 14 October, 2014, 3:58 pm
```

Figure 40: Extract from the permissions.log file

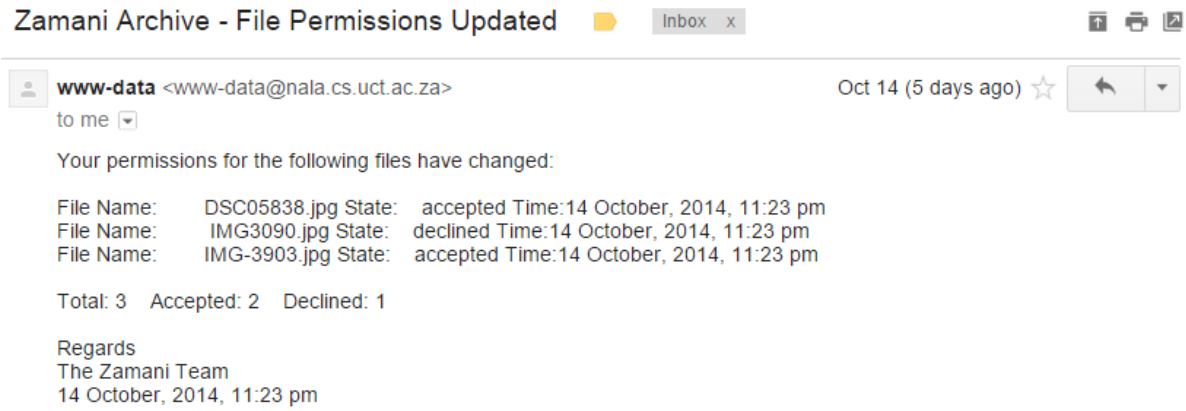


Figure 41: Email notification to use that their permissions have been updated

Evaluation

This iteration provided the Zamani Data Archivie with the remainder of the Data Request Interface allowing users to request files, view requests and download content. Additional functionality was created to allow a user to change their password.

It was noted that the ability to perform batch file requests would be useful and speed up the process of requesting multiple files for the user. This was not included in this iteration due to time constraints and will be further explored in future work on the Zamani Data Archive.

The functionality created in this iteration provided users with access to the Zamani data set, a primary goal of the project. Dynamically creating ZIP archives for batch downloads provided a mechanism to both lower the file download size and provide the user with an easy way of downloading multiple files.

Final Iteration

Planning

The final iteration of the development life cycle was used to create the Archival Tools which provide ingesting and purging functionality. This gives the administrator control over which sites are publically available in the Zamani Data Archive and what they contain. The administrator may also ingest and purge subsets of collections if updates on records are needed. Additionally, the ingestion process required files to be uploaded to the server, so that they could be locally accessed.

It was decided that the purge functionality would remove all the metadata records relating to the purge and that the files themselves would not be deleted in case of an accidental purge. Additionally, Archival Tools required functionality to handle ingestion and purging of a subset of a site.

Design and Implementation

Ingestion

The first part of the Archival Tools that was developed was the ingestion functionality. The file structure chosen for this was a parent collection (country) folder which contains a folder for each site. The site folder subsequently contains the metadata relating to the site that has been created by the Meta-fy tool. For example the Sudan parent collection contains the MusawwaratEsSufra site which subsequently contains all the metadata relating to that site. Both parent collection and sites consist of names with no spaces. This structure imitates the current structure of the Zamani data set allowing the administrator to easily understand the Zamani Data Archive structure should they need to access the metadata at a later point.

The metadata created by the Meta-fy tool is then selected by the user and uploaded. Once the upload is complete cURL is then used to make a POST request to the Fedora repository to ingest records in the site folder. The indexes of the repository are then updated and optimized using the Generic Search Service in conjunction with Solr.

Validation of the all the input fields for the ingestion functionality has been enforced to ensure that all the necessary fields are filled in and that the system does not behave in an unexpected manner due to invalid input. A high-level overview of the ingestion process can be seen in Figure 42.

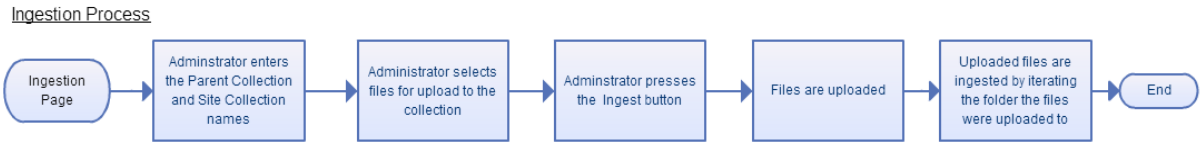


Figure 42: Flowchart of the ingestion process

Ingestion – Important Scripts and Functions

upload.php

This script is responsible for creating or updating the contents of a site in the Zamani Data Archive. Before any files are uploaded this script first checks if the parent collection and site folders exists in the uploads directory on the server. If they do not exists they are created. Once the folder structure has been created or identified, the script then uploads the files to the directory specified by the administrator.

Once the files have finished uploading the site directory containing the metadata is then processed using a directory iterator. Processing entails determining the PID of each file using its name. An example of a parent collection, site collection and object PID can be seen in Figure 43. Once the PID has been determined the ingestXMLFile function is called using the PID and file name as parameters. This function is found in the ingestor.php scripts which is described below.

```
collection1:Sudan
collection1:MusawwaratEsSufra
uctnew:0000001
```

Figure 43: Example PID of a parent collection, site collection and object

Once the entire directory has been iterated through the updateIndexFromXML.php and updateIndexOptimize scripts are called. The details of these scripts can be seen in the preceding sections.

This script also allows a user to update a subset of records by first purging the objects from the Fedora repository as described below and then uploading the files you wish to update.

ingestor.php

In order to ingest files this script contains the ingestXMLFile function which takes a PID and a file name as input. It then performs a cURL post request to the Fedora repository to ingest the data.

```
CURLOPT_RETURNTRANSFER => true,
CURLOPT_URL => $url,
CURLOPT_HTTPHEADER => array("Accept: text/xml", "Content-Type: text/xml"),
CURLOPT_USERPWD => $fedoraUserPwd,
CURLOPT_HTTPAUTH => CURLAUTH_BASIC,
CURLOPT_SSL_VERIFYPEER => false,
CURLOPT_POST => true,
CURLOPT_POSTFIELDS => $xml
```

Figure 44: cURL options used for ingestion of objects into the Fedora repository

updateIndexFromFOXML.php

This function is responsible for updating the Generic Search Interface and Solr indexing of the repository using the FOXML objects that have been ingested. This script is used after either ingestion or purging has been performed to ensure that the indexing of the archive contains the current set of objects in the archive.

```
$url = 'http://fedoraAdmin:fedoraAdmin@localhost:8080/fedoragsearch/rest';
$myvars = "operation=updateIndex&action=fromFoxmlFiles";

$ch = curl_init( $url );
curl_setopt( $ch, CURLOPT_POST, 1);
curl_setopt( $ch, CURLOPT_POSTFIELDS, $myvars);
curl_setopt( $ch, CURLOPT_FOLLOWLOCATION, 1);
curl_setopt( $ch, CURLOPT_HEADER, 0);
curl_setopt( $ch, CURLOPT_RETURNTRANSFER, 1);
$response = curl_exec( $ch );
```

Figure 45: cURL options used update indexes

updateIndexOptimize.php

In order to improve the performance of the Solr search functionality the indexes are optimised after they have either been ingested or purged. Optimising the indexes is similar to a defragmentation command. This operation reorganises the indexes into segments, increasing search speed (Fedora Project, 2007). Additionally, this function removes any indexes of objects that have been removed. The optimise operation is called via a cURL post request to the Fedora GSearch REST API.

```
$url = 'http://fedoraAdmin:fedoraAdmin@localhost:8080/fedoragsearch/rest';
$myvars = "operation=updateIndex&action=optimize";

$ch = curl_init( $url );
curl_setopt( $ch, CURLOPT_POST, 1);
curl_setopt( $ch, CURLOPT_POSTFIELDS, $myvars);
curl_setopt( $ch, CURLOPT_FOLLOWLOCATION, 1);
curl_setopt( $ch, CURLOPT_HEADER, 0);
curl_setopt( $ch, CURLOPT_RETURNTRANSFER, 1);
```

Figure 46: cURL options to optimise indexes

Purging

The next part of the Archival Tools that was developed was the purging functionality. Purging makes use of the file structure created in the ingestion process. A drop down list is provided to the administrator containing all site folders. In order to purge a site the administrator selects the site they would to purge from the Fedora repository and presses the purge collection button. The script then uses a directory iterator to determine the PIDs of the objects the administrator has selected for removal. Purging is done by using the purgeObject function in the purger.php script described in the following section.

Additionally, the administrator may also enter a comma separate list of PIDs if they wish to remove a sub-set of a collection. The list of PIDs is also passed to the purger.php script.

Purging – Important Scripts and Functions

purger.php

This script contains the purgeObject function which is used to purge objects from the Fedora repository by passing it the PID of the file the user wishes to remove. This script follows a similar structure to the ingestor.php script with the inclusion of a CURLOPT_CUSTOMREQUEST option which is set to DELETE. This tells the Fedora repository to remove the specified index from the repository.

```
CURLOPT_RETURNTRANSFER => true,
CURLOPT_URL =>$url ,
CURLOPT_HTTPHEADER => $headers,
CURLOPT_USERPWD => $fedoraUserPwd,
CURLOPT_HTTPAUTH => CURLAUTH_BASIC,
CURLOPT_SSL_VERIFYPEER => $fedoraVerifyPeer,
 CURLOPT_CUSTOMREQUEST => 'DELETE'
```

Figure 47: cURL options used for purging an object from the Fedora repository

Purging is done by sending a custom 'DELETE' request via cURL to the fedora repository. The fedora repository replies with a time stamp and PID in the console to inform the user that records have been successfully removed. The console also shows the administrator if the object being purged does not exists in the low level storage.

Testing

The feasibility testing for the final iteration required that an administrator be able to ingest and purge either a site or a subset of a site from the Zamani Data Archive. Additionally, these values needed to be indexed and optimised in order for Solr to power the Search Interface functionality of the archive. The results of these tests can be seen below.

Ingestion

The screenshot below shows the current facets available as well as collection marker on the map. This indicates that the objects were successfully ingested into the archive as the Search Interface utilising the indexes created during ingestion to generate these values.

Figure 48: Confirmation that records were successfully ingested

Purging

The screenshots below show a successful purge of both a collection and a subset of collection. This is confirmed by the timestamp and PID that can be seen in the console.

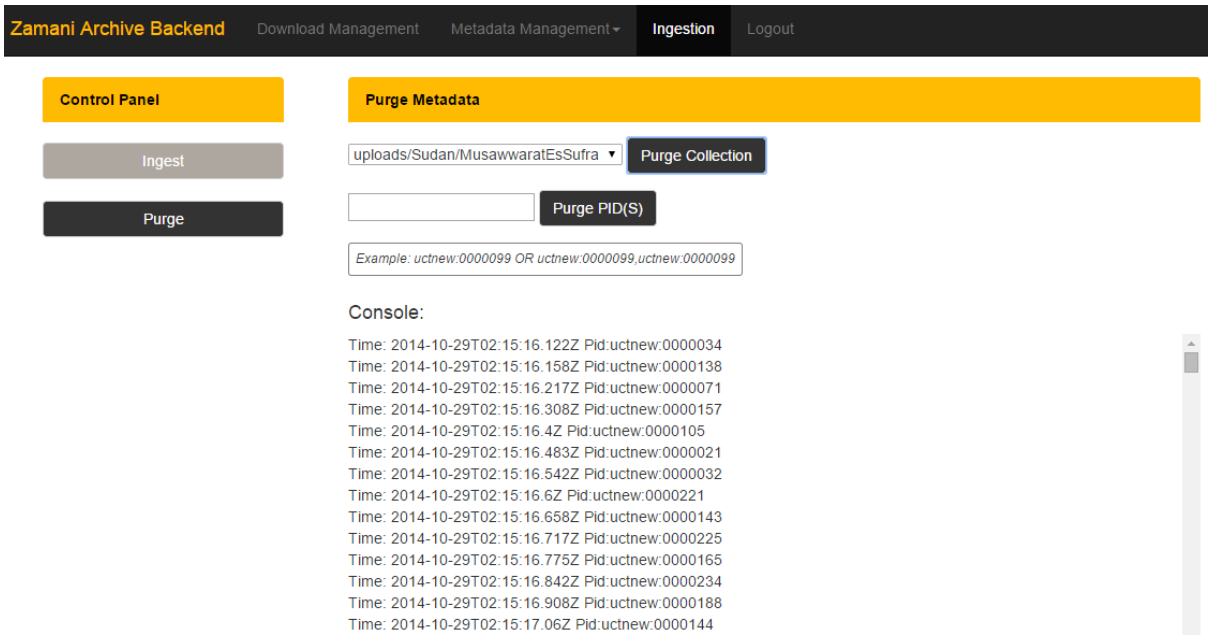


Figure 49: Successful purge of the Sudan Musawwarat Es Sufra collection

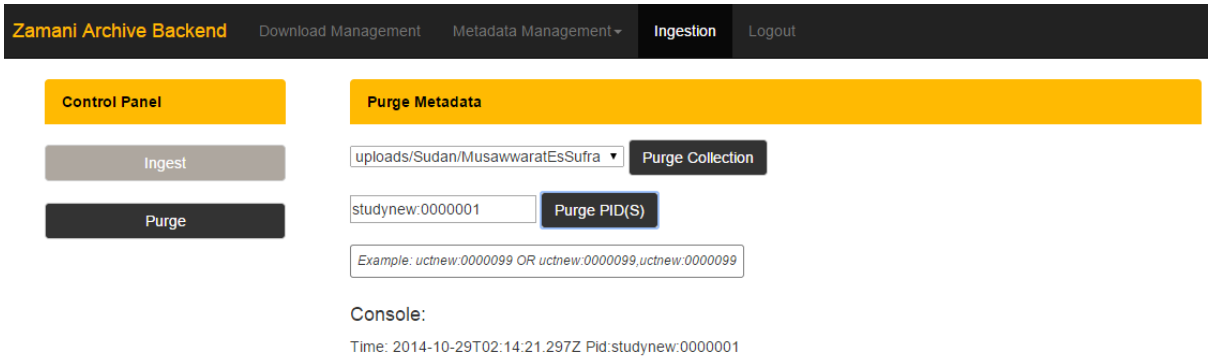


Figure 50: successful purging of a subset of a collection

Evaluation

This iteration provided the Zamani Data Archive with the final components it required to be fully functional, the ingestion and purging functionality.

The Zamani Team were pleased that a permissions.log had been included and understood the auditing value it provided.

During the evaluation of this iteration it was noted that the Ingestion page gives very little feedback that it is currently processing an ingestion request. Due to the development stage of the project been complete at this point no further work as was done on this interface. Better feedback for this interface could thus be considered an opportunity for future work on the Zamani Data Archive.

Evaluation

Introduction

This chapter describes the final evaluation of the systems. The proposal for this project required that the system be peer-reviewed and that customer feedback would be used to evaluate the overall success of the project. It was decided that system usability testing and user acceptance testing should be used to perform these evaluations.

Legal and Ethical Issues

All software used in the development of the Zamani Data Archive is open source, this was done to ensure the longevity of the system. Additionally, future development on the project would not be constrained by licences and fees. All the Zamani data used in the project was provided by the Zamani Project team with an understanding that it would only be used for the project and not distributed.

Before the Usability test was performed in the final evaluations of the system ethical clearance was obtained from the Department of Student Affairs granting access to University of Cape Town students (see Appendix C1). Additionally, clearance was obtained from the Science Faculty Research Ethics Committee (see Appendix C2).

Usability Testing

A primary objective of the project was to provide a way to present the Zamani Project data to the public and researchers alike. Additionally, the archive was intended to provide the Zamani Team with tools to manage their data and structure it. In order to determine if these objectives were achieved, a usability test was deemed appropriate as it measures the capacity of a product to meet its intended purpose (Brooke, 1996).

Hypothesis

The Zamani Archival System offers a high level of usability to both end-users and the system administrators.

Study Description

The usability test used a sample population consisting of 15 University of Cape Town students. Students were deemed an accurate representation of the target population as they possess similar levels of expertise. Additionally, this population was chosen as researchers involved in geospatial data are not easily accessible, which would have resulted in a smaller sample size.

The study was performed in the honours laboratory in the Computer Science building at the University of Cape Town. The duration of the study was approximately 45 minutes but this did vary between students. No constraints were placed on the time given to perform tasks as the chosen population varied in levels of expertise regarding; computers, web browsers, digital archives, permissions systems, metadata and search interfaces.

The same computer was used for all testing to minimize the variation in system performance and subsequently the interaction the participant had with the archive. In order to minimize the effects of the network instability, the computer used for the study was positioned in the same location in the honours laboratory.

Before participants were allowed to do the study, they were required to fill out a consent form (see Appendix C3). Participants were informed that their anonymity would be kept and that any information obtained would be kept confidential.

Participants were then asked to complete the Zamani Archival System Usability Test that can be found in Appendix C4, consisting of five sections. The first section was a background questionnaire which was used to gauge the participant's level of expertise. The remaining sections dealt with a number of tasks involving the main user activities of the Zamani Archival System. These tasks we broken up into permissions system and archival tools. After each task, the participant was required to answer a short questionnaire, relating to the task they had just completed and a System Usability Scale (SUS) for that component.

Results and Analysis

The study was performed with 15 participants ranging in age from 21 -24 years old. Participants were also spread across the Commerce, Engineering, Humanities and Science faculties. This spread ensured that the sample of participants used was a true representation of the population.

Every participant was able to complete all the tasks in the usability test for both the permissions system and archival tasks. This alone is a strong indication of the usability of the archive. Additionally, this reinforces the results of the SUS as feedback was given after the completion of all the tasks.

The mean SUS score for the permissions and archival tasks were 83.125 and 79.6875 respectively. With just these individual scores it is, however difficult to interpret what these scores mean. In order to provide a subject label to these values, the adjective rating scale seen in Figure 54 was added to the SUS as suggested by Bangor, et al (2009). Using this scale, we are to see that both the permissions and archival tasks fall between the 'Good' and 'Excellent' categories. It can thus been deduced that the system offers a high level of usability. However, the fact that these values could be higher indicates that there is room for improvements to the system.

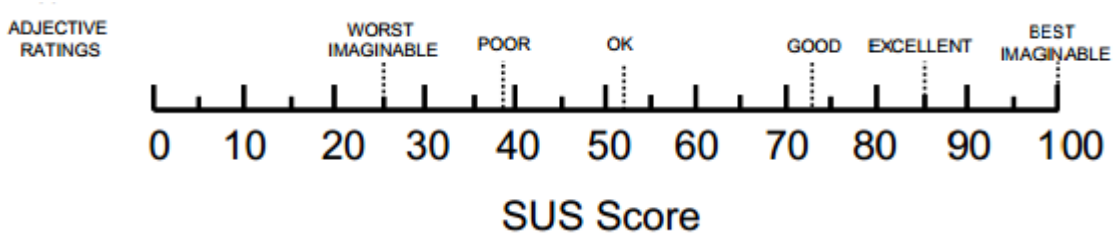


Figure 51: SUS adjective rating scale

To better understand these scores, a closer examination of the SUS results across the various faculties is needed. Students in the Commerce and Humanities faculties tended to give more neutral responses, whereas students in the Engineering and Science faculties gave higher scores to both SUS questionnaires. This indicates that the mean SUS scores had a central tendency, which can be explained by the composition of the population. As the Science and Engineering students more closely

represent the target population of the system this is not a cause for concern. However this still indicates that the usability of the system could be improved for non-expert users.

The lower score of the archival tasks can be further explained by the fact that some participants in the study were non-expert users. As the Archival tools were created for expert users, it was expected that the participants would find these tasks more difficult.

In answering the questions for the permissions tasks, it was noted that some participants responded 'No' to the question, "Was the download button easy to find on the accounts page?" This can be attributed to the fact that users of the system are required to scroll down to see the "Download File(s)" button. This indicates that the placement of the button could be revised in future work on this archive.

User Acceptance Testing

In order to evaluate whether the requirements of the project were met, a user acceptance test was deemed the most appropriate tool. This testing required the development team to demonstrate all the required functionality of the system to the clients. Additionally, the clients were asked to evaluate whether the requirements of each component of the archive were met. The user acceptance test used can be found in the Appendix C5.

Test Description

The user acceptance testing was done at the Geomatics Department at the University of Cape Town with the clients, the Zamani Project team. The clients were informed that the testing was being conducted to evaluate whether the requirements of the Zamani Data Archive had been met.

A fully functional testing environment was provided to the participants along with any required test data. Test scenarios were taken from the usability study conducted by the development team to demonstrate the core functionality of the system. The development team created secondary tasks on-the-fly to encourage client involvement. This was be used to demonstrate functionality and increase overall understanding of the system.

The Zamani Project Team was then asked if the initial requirements of the project had been met and if the system behaved in the expected manner. The Zamani Project Team was asked to assign a 'pass' or 'fail' status to each test case based on whether it has satisfied the specified initial requirements.

The scope of the User Acceptance Test included all the components of the permissions system and Archival Tools. The video streaming functionality was not implemented in the final system due to time constraints and this was deemed out of scope for this test.

The test performed can be seen in the Requirement-Based Test Criteria section.

Requirement-Based Test Criteria

ID	Criteria
1.1	Account Management
1.1.1	Register an account
1.1.2	Terms & Conditions of the Zamani Project Archive in registration
1.1.3	Activate registered account
1.1.4	Request a new password
1.2	File Requests
1.2.1	Request a file with a personalised message
1.2.2	Zamani Archive Admin notified by email
1.2.3	View file request status
1.3	File Downloads
1.3.1	Single file download
1.3.2	Batch file download
2.1	Permissions Management
2.1.1	View user file requests as an administrator
2.1.2	Accept user requests
2.1.3	Decline user requests
3.1	Ingesting
3.1.1	Ingest metadata produced by Meta-fy
3.1.2	Ingest a sub-set of a collection
3.2	Purging
3.2.1	Purge collection
3.2.2	Purge list of records

Results and Analysis

All test cases where passed in the user acceptance testing, which confirmed that the original requirements of the project had been met. This can be seen in the Results section below.

Results

ID	Test Cases	Pass/Fail	Tested By	Date Tested
1.1	Account Management	Pass	Zamani Team	20/10/2014
1.2	File Requests	Pass	Zamani Team	20/10/2014
1.3	File Downloads	Pass	Zamani Team	20/10/2014
2.1	Permissions Management	Pass	Zamani Team	20/10/2014
3.1	Ingesting	Pass	Zamani Team	20/10/2014
3.2	Purging	Pass	Zamani Team	20/10/2014

The overall reception from the client was excellent. This was confirmed when the client indicated that they felt the Zamani Archive was ready for production.

The clients indicated that they wished to do future work on the Zamani Archive, including improvements and new requirements. Notable new requirements included the inclusion of bulk file request functionality and download statistics.

Future Work

A number of opportunities for future work were identified during the development process of the Zamani Data Archive. This chapter aims to describe these potential sources of future work, which could be used to improve and extend the functionality of the Zamani Data Archive. Additionally, video streaming functionality has been included in this section as time constraints prevented the development of this feature.

Public Portal

Batch Requests

Currently the request functionality of the Public Portal only allows for single file requests. The possible inclusion of batch request for a result set could provide users of the Zamani Archive with a more efficient way of requesting multiple files.

Statistics

The inclusion of download statistics could provide the Zamani Project team. Additionally, statistics relating to the number of objects and sites in the repository could be added. This could provide the Zamani Project team with a metric to better gauge user interaction with the Zamani Data Archive. These statistics could be displayed in the Backend on a Statistics page.

Change Password

Currently the functionality to change a user's password in the Public Portal server is limited to sending the user a randomly generated password. The inclusion of functionality to change your password once you are logged in could be included. This will provide users that have forgotten their passwords and with a way of replacing their randomly generated password with something more familiar.

Video streaming

Due to time constraints, video streaming content was not developed for the system. The inclusion of this functionality could provide Zamani Data Archive with an additional method for presenting the Zamani data set to users.

Archival Tools

Ingestion interface Feedback

Currently the ingest interface in the Archival Tools does not provide any feedback after it has completed the uploading of files to the upload folder for ingestion. When the ingestion is complete the page refreshes clearing the input fields. This indicates that the ingestion is complete. This is due to use of a file input field which cannot be used in an AJAX request. It is thus difficult to display feedback in the same manner as the purge interface, which provides a console showing the PID and

timestamp of each record that is purged. Alternative methods of providing feedback in this interface could be further explored in future work on the Zamani Data Archive.

Conclusion

The proposed system for the Zamani Data Archive consisted of an Archive, Public Portal, Search Interface and Metadata Management tool.

The archive consisted of an Nginx Webserver and a Fedora repository which in conjunction provided the infrastructure to support the Zamani Data Archive. Nginx was used to serve content to users as well as proxy requests to a PHP-FPM daemon and the Fedora repository. Fedora provided an extensive REST Web API that could be used to access the Zamani Data Set in both the Search Interface and Public Portal. The Archival tools provided the Zamani Data Archive administrators with a way of managing the content available in the repository.

The Public Portal provided users of the Zamani Data Archive with a mechanism to access the Zamani data set. A permissions system was developed to allow users to request the data available on the archive. The public portal allows users to perform both single and batch file downloads. Batch downloads were handled by the Nginx Mod_Zip module which provided a mechanism to dynamically generate ZIP archives. The Request Management interface provided the administrator of the Zamani Archive with a tool to manage requests.

The Usability Test provided the development team with a subjective label for the overall usability of the system which was concluded to be in the range between good and excellent. As this score was not perfect this does indicate that there are still improvements which could be made to the system which was further explored in the Future Work chapter.

The positive response received from the Zamani Project team during the final evaluation of the system indicated that the system met its initial requirements. This was further enforced when the Zamani Project team indicated that they felt the system is ready for a production environment and wanted to explore possible future work on the archive.

Bibliography

Bangor, A., Kortum, P. & Miller, J. A., 2009. Determing what individual SUS Scores Mean: Adding an Adjective Rating Scale. *Journal of Usability Studies,* 4(3), pp. 114-123.

Brooke, J., 1996. SUS - A quick and dirty usability scale. In: *Usability evaluation in industry.* s.l.:s.n., pp. 189-194.

Dspace, 2008. *XMLUI Interface Customizations (Manakin).* [Online]
Available at:
http://dspace.org/sites/dspace.org/files/archive/1_5Documentation/configure.html#xmlui
[Accessed 24 October 2014].

Dspace, 2014. *Top Reasons To Use DSpace.* [Online]
Available at: Top Reasons To Use DSpace
[Accessed 20 October 2014].

Duraspace, 2014. *Add a new format to the bitstream registry.* [Online]
Available at:
https://wiki.duraspace.org/display/DSPACE/Add+a+new+format+to+the+bitstream+registry
[Accessed 24 October 2014].

Fedora Commons, 2014. *http://www.fedora-commons.org/licenses.* [Online]
Available at: http://www.fedora-commons.org/licenses
[Accessed 24 October 2014].

Fedora Project, 2007. *Fedora Generic Search Service.* [Online]
Available at: http://www.fedora.info/download/2.2/services/genericsearch/doc/
[Accessed 26 October 2014].

Fedora Project, 2007. *The Fedora Digital Object Model.* [Online]
Available at: http://fedora-commons.org/documentation/3.0/userdocs/digitalobjects/objectModel.html
[Accessed 24 October 2014].

Fedora Project, 2010. *Web Service Interfaces.* [Online]
Available at: https://wiki.duraspace.org/display/FEDORA34/Web+Service+Interfaces
[Accessed 25 October 2014].

Fedora, 2005. *Fedora - Introduction to Fedora Object XML.* [Online]
Available at: http://www.fedora.info/download/2.0/userdocs/digitalobjects/introFOXML.html#WHY
[Accessed 18 October 2014].

Fedora, 2014. *Fedora Integrations.* [Online]
Available at: http://www.fedora-commons.org/fedora-integrations
[Accessed 25 October 2014].

Fedora, 2014. *Installation and Configuration.* [Online]
Available at: https://wiki.duraspace.org/display/FEDORA37/Installation+and+Configuration
[Accessed 25 October 2014].

Miller, E., 2013. *Nginx - mod_zip.* [Online]
Available at: http://wiki.nginx.org/NginxNgxZip
[Accessed 18 October 2014].

Nginx, 2014. *Beginners Guide.* [Online]
Available at: http://nginx.org/en/docs/beginners_guide.html
[Accessed 17 October 2014].

PHP, 2014. *FastCGI Process Manager (FPM).* [Online]
Available at: http://php.net/manual/en/install.fpm.php
[Accessed 19 October 2014].

PostgreSQL, 2014. *Chapter 35. PL/pgSQL - SQL Procedural Language.* [Online]
Available at: http://www.postgresql.org/docs/8.0/static/plpgsql.html
[Accessed 24 October 2014].

ROAR, 2014. *Registry of Open Access Repositories.* [Online]
Available at: http://roar.eprints.org/
[Accessed 24 October 2014].

Rüther, H. et al., 2009. Laser scanning for conservation and research of African cultural heritage sites: the case study of Wonderwerk Cave, South Africa. *Journal of Archaeological Science,* 36(9), pp. 1847-1856.

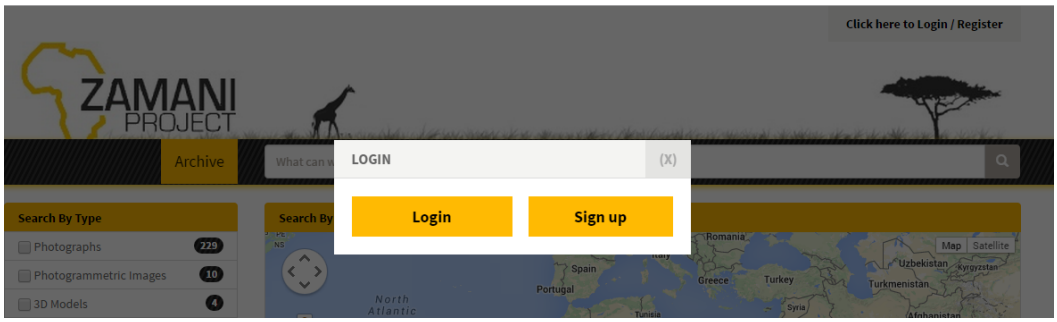
Zamani Project, 2014. *Zamani Project - Data Types.* [Online]
Available at: http://www.zamaniproject.org/index.php/data.html
[Accessed 22 May 2014].

Zamani Project, 2014. *Zamani Project - Project.* [Online]
Available at: http://www.zamaniproject.org/index.php/project.html
[Accessed 22 May 2014].

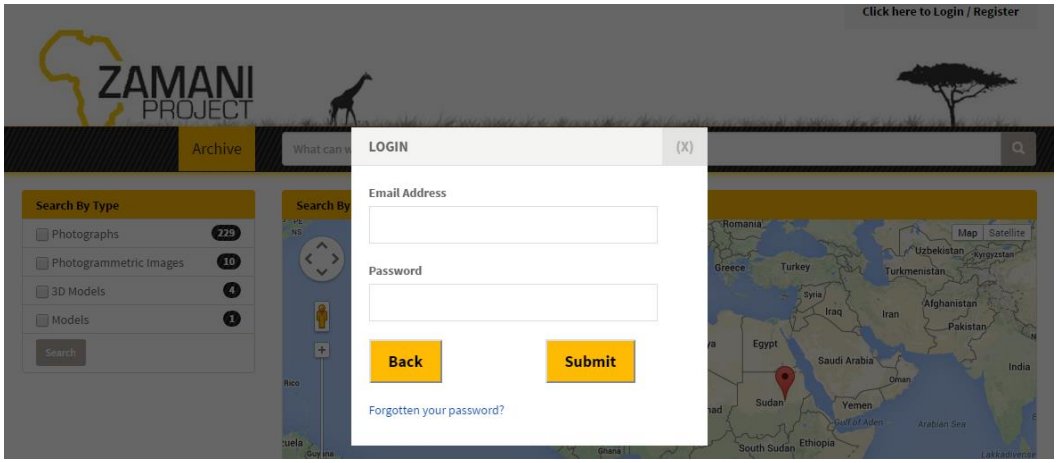
Appendix A

A1 - Account Management Screenshots

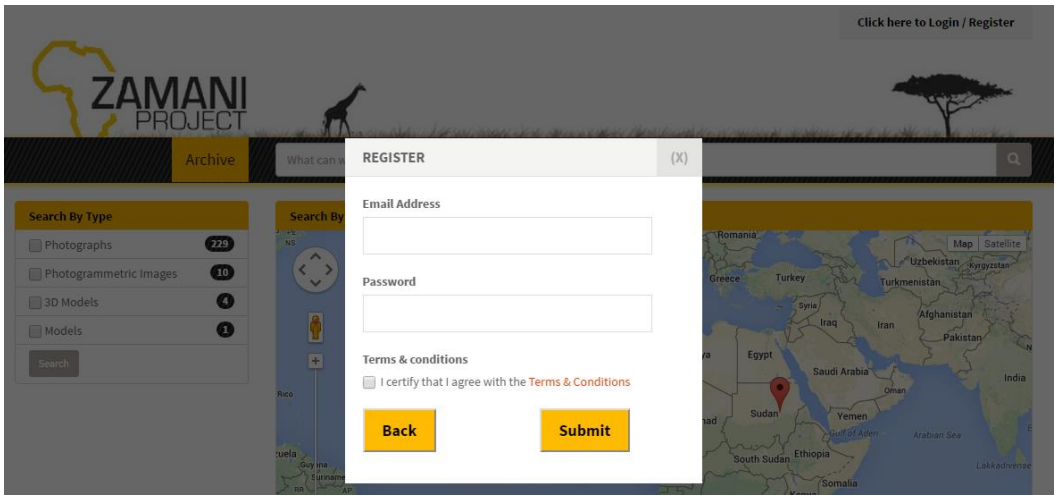
Login Home



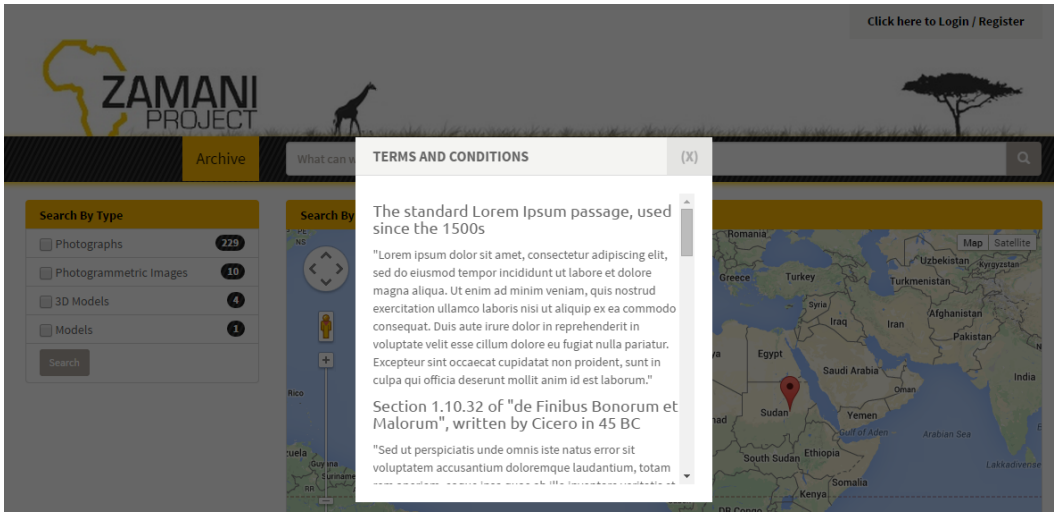
Login Form



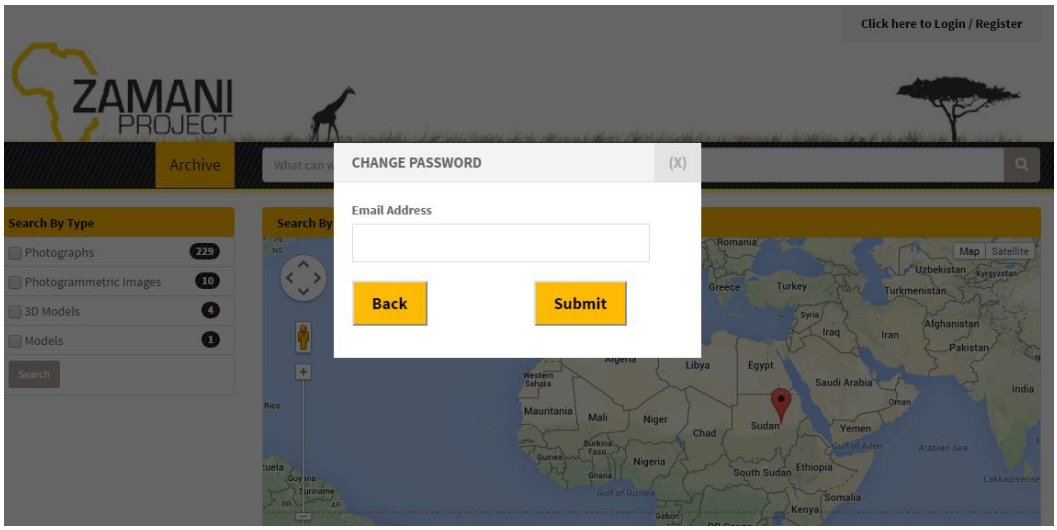
Registration Form



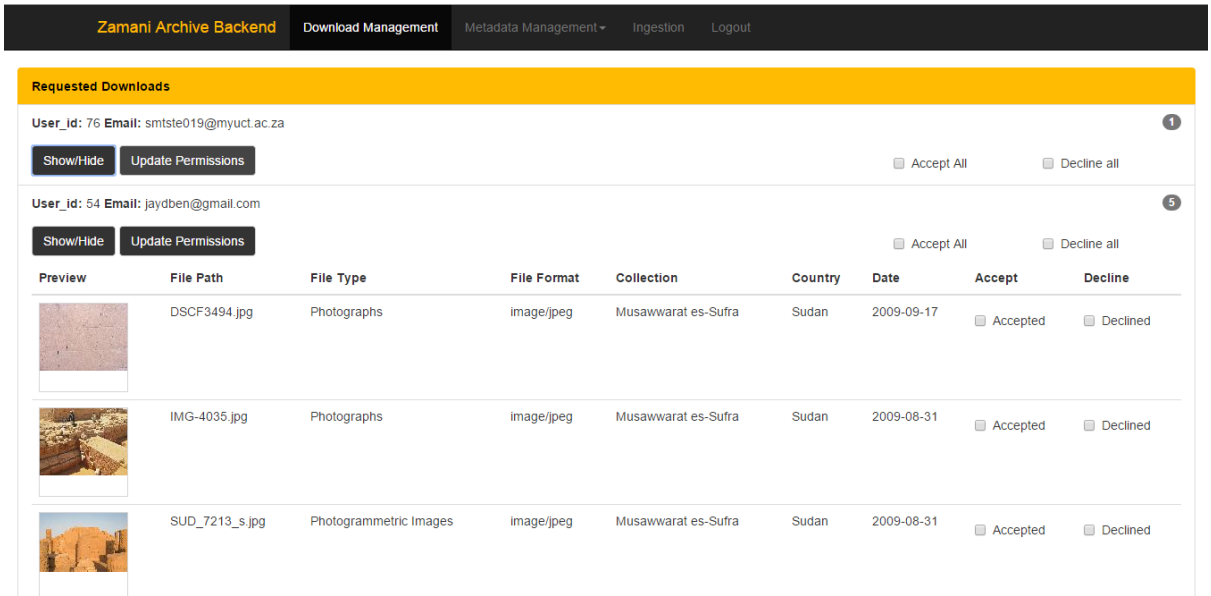
Terms and Conditions Page



Change Password Form



A2 - Download Management Page



Appendix B

B1 – Nginx Modules and Configuration

GZIP

The GZIP module was included to allow GZIP compression of all the static content served by the webserver relating to both the Frontend and the Backend. This was done in order to lower the bandwidth needed by the Zamani Archival system and to speed up page load times.

Mail

The mail module was included to provide email feedback about file permissions to both the users and the admin of the Zamani Archival System. The mail module linked to the existing mail server on Nala (the server used for development) which required no further configuration.

Internal Location

In order to ensure the security of the Zamani Data Archive it was mounted on a location outside the Nginx Web root. To allow Nginx to serve content in the mounted folder, a 'Location' was created. Nginx uses locations directives to define paths to locations on the server (Nginx, 2014). The true location of the data was aliased to a '/files' location which could then be referenced by other Nginx functionality. By using an alias the true location of the file was also hidden from potential attacks to the archive.

The 'internal' directive was used so that the location could only be referenced by Nginx. Any external requests to this location result in a '404 (Not Found)' response, further decreasing the chances of an attack on the Zamani data set. The configuration used can be seen below.

The GZIP module was disabled in this location to prevent the content being served in this directory from being compressed twice as this location was created to handle download which would include its own compression if needed.

```
location /files {
        autoindex off;
        gzip off;
        alias  /mnt/shares;
        internal;
  }
```

Figure 52: Nginx internal location (found in fedora.conf)

Nginx and PHP

In order to create the PHP-FPM daemon, the package was installed and configured to listen on port 9000. For Nginx to use the PHP-FPM daemon the 'php' upstream seen below in Figure 56 was created. This forwards PHP requests via the TCP/IP protocol to PHP-FPM daemon. Additionally, the location in

Figure 57 was created which specified the FastCGI parameters for PHP-FPM before sending the request to the 'php' upstream.

```
upstream php {
   server 127.0.0.1:9000;
}
```

Figure 53: Nginx PHP upstream (found in nginx.conf)

```
# pass the PHP scripts to FPM
location ~ \.php$ {
   try_files $uri =404;
   fastcgi_split_path_info ^(.+\.php)(/.+)$;
   include fastcgi_params;
   fastcgi_index index.php;
   fastcgi_param SCRIPT_FILENAME /usr/share/nginx/www$fastcgi_script_name;
   fastcgi_pass php;
   fastcgi_read_timeout 300;
}
```

Figure 54: Nginx worker PHP redirect to PHP-FPM (found in fedora.conf)

Proxy Application Server

The Nginx webserver also served as a proxy server, passing requests to the Fedora repository. This is due to the fact that Nala can only be accessed on port 80 due to security restrictions enforced at the University of Cape Town.

In order to configure Nginx to proxy requests to the Fedora repository running in the Apache Tomcat application server, the 'proxyPort' and 'proxyName' settings were required. These values were linked to the 'listen' port and 'server_name' directives in the Nginx configuration. The configuration used can be seen below.

```
listen 80;
server_name nala.cs.uct.ac.za;
```

Figure 55: Nginx proxy configuration (found in fedora.conf)

```
<Connector port="8080"  protocol="HTTP/1.1" proxyPort="80" proxyName="nala.cs.uct.ac.za"
connectionTimeout="20000" redirectPort="8443" enableLookups="true" acceptCount="100"
maxThreads="150" minSpareThreads="25" maxSpareThreads="75" URIEncoding="UTF-8" />

server.xml
```

Figure 56: Apache Tomcat proxy configuration (found in server.xml)

Mod_zip

This module was developed by Evan Miller (https://github.com/evanmiller/mod_zip) and can be used to dynamically assemble ZIP files. Mod_zip scans the response body of an HTTP request for a list of files with additional attributes, which are retrieved and encoded in order (Miller, 2013). The attributes needed by the module are a CRC32 checksum of the file, file size, file path and file name. This module was added for the compression of batch file download requests in order to reduce the size of the download for the user.

Appendix C

C1 – Science Faculty Research Ethical Clearance

Faculty of Science
University of Cape Town
RONDEBOSCH 7701
South Africa

E-mail: richard.hill@uct.ac.za
Telephone: + 27 21 650 2786
Fax: + 27 21 650 3456

22 October 2014

Mr Michael Ferguson & Mr Jay Benson
Department of Computer Science

ZAMANI DATA ARCHIVE

Dear Mr Ferguson & Mr Benson

I am pleased to inform you that the Faculty of Science Research Ethics Committee has approved the above-named application for research ethics clearance, subject to the conditions listed below. You are required to:

- implement the measures described in your application to ensure that the process of your research is ethically sound; and
- uphold ethical principles throughout all stages of the research, responding appropriately to unanticipated issues: please contact me if you need advice on ethical issues that arise.

Your approval code is: FSREC 080– 2014

I wish you success in your research.

Yours sincerely

RC Hill

Dr Richard Hill
Chair: Faculty of Science Research Ethics Committee

Cc: A/Prof Hussein Suleman, Supervisor

C2 – Research Access to Students

	RESEARCH ACCESS TO STUDENTS	DSA 100

NOTES

1. This form must be **FULLY** completed by the applicant/s who want to access UCT students for the purpose of research or surveys.
2. Return the fully completed **(a) DSA 100** application form **by email**, in the same word format, together with your: **(b) research proposal inclusive of your survey, (c) copy of your ethics approval letter / proof (d) informed consent letter to:** Moonira.Khan@uct.ac.za. You application will be attended to by the Executive Director, Department of Student Affairs (DSA), UCT.
3. The turnaround time for a reply is **approximately 10 working days.**
4. NB: It is the responsibility of the researcher/s to apply for and to obtain **ethics approval and to comply with amendments that may be requested; as well as to obtain** approval to access UCT staff and/or UCT students, from the following, respectively: (a) **Ethics:** Chairperson, Faculty Research Ethics Committee' (FREC) for ethics approval, (b) **Staff access:** Executive Director: HR for approval to access UCT staff, and (c) **Student access:** Executive Director: Student Affairs for approval to access UCT students.
5. **Note:** UCT Senate Research Protocols requires compliance to the above, even if prior approval has been obtained from any other institution/agency. UCT's research protocol requirements applies to *all* persons, institutions and agencies from UCT and external to UCT who want to conduct research for academic, marketing or service related reasons at UCT.

SECTION A: RESEARCH APPLICANT/S DETAILS

Position	Staff / Student No	Title and Name	Contact Details (Email / Cell / land line)
A.1 Student Number	FRGMIC005 BNSJAY001	Mr Michael Ferguson Mr Jay Benson	Email: frgmic005@myuct.ac.za / (072) 720 4871 Email: bnsjay001@myuct.ac.za / (073) 186 7141
A.2 Academic / PASS Staff No.			
A.3 Visitor/ Researcher ID No.			
A.4 University at which a student or employee	University of Cape Town	Address if *not* UCT:	
A.5 Faculty/ Department/School	Department of Computer Science		
A.6 APPLICANTS DETAILS if different from above	Title and Name	Tel.	Email

SECTION B: RESEARCHER/S SUPERVISOR/S DETAILS

Position	Title and Name	Tel.	Email
B.1 Supervisor	Hussein Suleman	+27 21 650 5106	hussein@cs.uct.ac.za
B.2 Co-Supervisor/s	Maria Keets	+27 21 650 2664	mkeet@cs.uct.ac.za

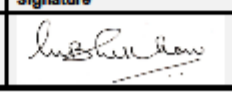
SECTION C: APPLICANT'S RESEARCH STUDY FIELD AND APPROVAL STATUS

C.1 Degree (if a student)	BSc Computer Science
C.2 Research Project Title	Zamani Data Archive
C.3 Research Proposal	Attached: Yes ■ No ☐
C.4 Target population	Students in different years from varying faculties
C.5 Lead Researcher details	If different from applicant:
C6. Will use research assistant/s	Yes ☐ No ■ If yes- provide a list of names, contact details and ID no.
C.7 Research Methodology and Informed consent:	Research methodology: Questionnaire and survey Informed consent: will be obtained prior to participation in the study. Confidentiality will be assured by not capturing personal information that can be traced back to a participant.
C.8 Ethics clearance status from UCT's Faculty Ethics Research Committee (FREC)	Approved by the FREC Yes ■ With amendments: Yes / No ☐ (a) Attach copy of your ethics approval. Attached: Yes (b) State date and reference no. of ethics approval: Date: 24 October 2014 Ref. No.: FSREC 080- 2014

SECTION D: APPLICANT/S APPROVAL STATUS FOR ACCESS TO STUDENTS FOR RESEARCH PURPOSE

(To be completed by the ED, DSA or Nominee)

	Approved / With Terms / Not	* Conditional approval with terms	Applicant/s Ref. No.:
D.1 APPROVAL STATUS	*Approved* ■	(a) Access to students for this research study must only be undertaken after written ethics approval has been obtained. (b) In event any ethics conditions are attached, these must be complied with before access to students.	FRGMIC005/ Mr Michael Ferguson BNSJAY001/ Mr Jay Benson

	Designation	Name	Signature	Date
D.2 APPROVED BY:	*Executive Director Department of Student Affairs*	*Dr Moonira Khan*		*24 October 2014*

Version. 2013 (6) Page 1 of 1 DSA 100

C3 -Consent Form

Consent to participate in a research study

Dear UCT student,

Invitation to participate in research investigating the usability of the Zamani Archival System.

Introductory Statement

The Zamani Archive is a system built for the Zamani Project (http://www.zamani-project.org/) to manage its large collection of geo-spatial data captured at various cultural heritage sites in Africa and the Middle-East. The systems sole purpose is the creation of an archive for the Zamani Project, in order to enable structure, preservation and distribution of the data. The system includes a metadata management tool, namely Meta-fy to ensure that the creation of metadata is easy and automated where possible. The system features a public portal that enables users to search through the archives metadata allowing them to view, request and download data from the archive. Additionally, the public portal includes a permission system whereby users need to register and accept the terms and conditions associated with the data before requesting and downloading data from the archive.

Study Purpose

You are being asked to participate in a research study being conducted by two computer science honours students at the University of Cape Town. The purpose of this study will be to analyse the Zamani Archival System.

Study Procedures

If you decide to participate in this study, you will be asked to complete a number of tasks as specified in the section to follow. After each task you will be asked to complete a short questionnaire. Please note that all information obtained from you will be kept strictly confidential.

Tasks

- Meta-fy related tasks
- Search interface related tasks
- Permission related tasks
- Archival related tasks

Environment & Duration

The study will take place in the honours laboratory in the Computer Science building at the University of Cape Town. The duration of the study is estimated to take 45 minutes but is user-dependent and will vary.

Voluntary Participation

Participation in this study is completely voluntary. You are free to refuse to answer any question. Your decision regarding participation in this study will not have any consequences for you. If you decide to participate, you are free to change your mind and discontinue participation at any time without any consequences.

Questions

Any study-related questions, problems or emergencies should be directed to the following researchers:

Michael Ferguson	frgmic005@myuct.ac.za
Jay Benson	bnsjay001@myuct.ac.za
Prof Hussein Suleman	hussein@cs.uct.ac.za

I have read the above information and am satisfied with my understanding of the study. I hereby voluntarily consent to participation in the research study as described.

_____ _____

Signature of participant **Date**

_____ **Michael Ferguson and Jay Benson**

Name of participant **Researchers**

C4 - System Usability Test

Questionnaire

There is a total of nine sections in this study, five of which consist of questions and the other four tasks.

1. Background Information

This section is essential in obtaining background information about you.

1.1 What is your age? _____

1.2 Rate your experience in the following:

- Using a computer (none, very little, basic, advanced, expert)
- Using a web browser (none, very little, basic, advanced, expert)

1.3 Rate your understanding of the following terms:

- Digital archive (none, very little, basic, advanced, expert)
- Permission system (none, very little, basic, advanced, expert)
- Metadata (none, very little, basic, advanced, expert)
- Search interface (none, very little, basic, advanced, expert)

1.4 What faculty are you enrolled in? _____

2. Permissions Tasks

The Zamani Archive index page is open in a tab in the browser. You will need to open another tab with your email address that you wish to register with. The Zamani Archive Backend index page is also in a tab in the browser.

2.1. Create an account on the Zamani Archive and activate it *(activation emails may take a short while)*

2.2. Request files in the archive

2.2.1. Search using any methods available on the search interface.

2.2.2. Navigate to a record view

2.2.3. Request the file using “Study Request Task” as the message for the request

2.2.4. Repeat steps 2.2.1 and 2.2.3 until you have requested 5 files

2.3. Accept the newly requested files

2.3.1. Navigate to the Zamani Archive Backend tab in the browser

2.3.2. Log in using the following details; Username: “study2014”, Password: “study2014”

2.3.3. Ensure you are now on the ‘Download Management’ tab

2.3.4. Look for your email address in the permissions table

2.3.5. Select “accept” for **3** files, “decline” for **1** file and leave **1** file unchecked

2.3.6. Update your permissions

2.3.7. Check your email address for a permissions update notification *(emails may take a short while)*

2.4. Download files you have accepted

2.4.1. Navigate to the Account page

2.4.2. Select one file and download it

2.4.3. Once your download is complete, Select all available files and download them

2.4.4. Verify that the downloaded zip contains the files requested

3. Permissions Questions

3.1. Did you successfully complete all of the 'Public Portal Tasks'?

Please choose one of the following (estimate where needed):

A) Yes
B) More than half the tasks successfully completed
C) More than 75 percent of the tasks successfully completed
D) Less than half the tasks successfully completed

3.2. Were you able to create an account? (YES , NO)
3.3. Were you able to request a file? (YES , NO)
3.4. Were you able to download a single file? (YES , NO)
3.5. Were you able to download a batch of files? (YES , NO)
3.6. Was the download button easy to find on the accounts page? (YES , NO)
3.7. Were you able to add a personal message with the file requests? (YES , NO)
3.8. Were you notified when you file was accepted? (YES , NO)
3.9. Did you receive a permissions update notification? (YES , NO)

Instructions: For each of the following statements, mark <u>one</u> box that best describes your reactions to the website *today*.

		Strongly Disagree				Strongly Agree
1.	I think that I would like to use this system frequently.	☐	☐	☐	☐	☐
2.	I found this system unnecessarily complex.	☐	☐	☐	☐	☐
3.	I thought this system was easy to use.	☐	☐	☐	☐	☐
4.	I think that I would need assistance to be able to use this system.	☐	☐	☐	☐	☐
5.	I found the various functions in this system were well integrated.	☐	☐	☐	☐	☐
6.	I thought there was too much inconsistency in this system.	☐	☐	☐	☐	☐
7.	I would imagine that most people would learn to use this system very quickly.	☐	☐	☐	☐	☐
8.	I found this system very cumbersome/awkward to use.	☐	☐	☐	☐	☐
9.	I felt very confident using this system.	☐	☐	☐	☐	☐
10.	I needed to learn a lot of things before I could get going with this system.	☐	☐	☐	☐	☐

4. Archival Tasks

The Zamani Archive index page is open in a tab in the browser. The Zamani Archive Backend index page is also in a tab in the browser.

4.1 Ingest new metadata into the archive

- 4.1.1 Navigate to the Zamani Archive Backend tab in the browser
- 4.1.2 Log in using the following details; Username: "study2014", Password: "study2014"
- 4.1.3 Navigate to the Ingestion page
- 4.1.4 Enter the name of the parent collection "Sudan" and site "MusawwaratEsSufra"
- 4.1.5 Select all of the files in the "Sudan_MusawwaratEsSufra_output" folder on your desktop and ingest them. *(The page will refresh when this is complete)*
- 4.1.6 Navigate to the Zamani Archive index page and verify your records have been ingested

4.2 Purge a collection of records in the archive

- 4.2.1 Navigate the to the Ingestion page
- 4.2.2 Press the "Purge" control panel option
- 4.2.3 Select the 'MusawwaratEsSufra' collection from the drop down list and purge it
- 4.2.4 Navigate to the Zamani Archive index page and verify your records have been purged

4.3 Purge a set of records in the archive

- 4.3.1 Repeat 3.1 so that you have records to remove
- 4.3.2 Ensure you are logged into the Zamani Archive Backend
- 4.3.3 Navigate the to the Ingestion page
- 4.3.4 Press the 'Purge' control panel option
- 4.3.5 Enter this list of comma separated PIDs (studynew:0000001, studynew:0000002, studynew:0000003) and purge them
- 4.3.6 Navigate to the Zamani Archive index page and verify your records have been purged

4.4 Ingest a sub-set of records into the archive

- 4.4.1 Enter the name of the parent collection "Sudan" and site "MusawwaratEsSufra"
- 4.4.2 Select the following files in the "Sudan_MusawwaratEsSufra_output" folder and ingest them.

  "studynew:0000001.xml", "studynew:0000002.xml", "studynew:0000003.xml"

  (The page will refresh when this is complete)
- 4.4.3 Navigate to the Zamani Archive index page and verify your records have been ingested

5. Archival Task Questions

5.1 Did you successfully complete all of the 'Ingestion Tasks'?

Please choose one of the following (estimate where needed):

A) Yes
B) More than half the tasks successfully completed
C) More than 75 percent of the tasks successfully completed
D) Less than half the tasks successfully completed

5.2 Were you able to accept or decline download requests? (YES , NO)
5.3 Were you able to ingest a collection? (YES , NO)
5.4 Were you able to purge a collection? (YES , NO)
5.5 Were you able to ingest a sub set of collection? (YES , NO)
5.6 Were you able to purge a subset of data? (YES , NO)

Instructions: For each of the following statements, mark <u>one</u> box that best describes your reactions to the website *today*.

		Strongly Disagree				Strongly Agree
1.	I think that I would like to use this system frequently.	☐	☐	☐	☐	☐
2.	I found this system unnecessarily complex.	☐	☐	☐	☐	☐
3.	I thought this system was easy to use.	☐	☐	☐	☐	☐
4.	I think that I would need assistance to be able to use this system.	☐	☐	☐	☐	☐
5.	I found the various functions in this system were well integrated.	☐	☐	☐	☐	☐
6.	I thought there was too much inconsistency in this system.	☐	☐	☐	☐	☐
7.	I would imagine that most people would learn to use this system very quickly.	☐	☐	☐	☐	☐
8.	I found this system very cumbersome/awkward to use.	☐	☐	☐	☐	☐
9.	I felt very confident using this system.	☐	☐	☐	☐	☐
10.	I needed to learn a lot of things before I could get going with this system.	☐	☐	☐	☐	☐

C5 – User Acceptance Test

Project name: Zamani Data Archive
Project Owner: Zamani Project – Geomatics Department UCT
Supervisor: Hussein Suleman
Test Date: 20/10/2014

In Scope

1. Permissions System
    1.1. Account Management
    1.2. File Requests
    1.3. File Downloads
2. Archival Tools
    2.1. Permissions Management
    2.2. Ingesting
    2.3. Purging

Out of Scope

1. Video Streaming
    1.1. Due to a lack of time this feature was not implemented in the final system and thus cannot be reviewed by the client.

Requirement-Based Test Criteria

ID	Criteria
1.1	Account Management
1.1.1	Register an account
1.1.2	Terms & Conditions of the Zamani Project Archive in registration
1.1.3	Activate registered account
1.1.4	Request a new password
1.2	File Requests
1.2.1	Request a file with a personalised message
1.2.2	Zamani Archive Admin notified by email
1.2.3	View file request status
1.3	File Downloads
1.3.1	Single file download
1.3.2	Batch file download
2.1	Permissions Management
2.1.1	View user file requests as an administrator
2.1.2	Accept user requests
2.1.3	Decline user requests
2.2	Ingesting
2.2.1	Ingest metadata produced by Meta-fy
2.2.2	Ingest a sub-set of a collection
2.3	Purging
2.3.1	Purge collection
2.3.2	Purge list of records

Results

ID	Test Cases	Pass/Fail	Tested By	Date Tested
1.1	Account Management	Pass	Zamani Team	20/10/2014
1.2	File Requests	Pass	Zamani Team	20/10/2014
1.3	File Downloads	Pass	Zamani Team	20/10/2014
2.1	Permissions Management	Pass	Zamani Team	20/10/2014
3.2	Ingesting	Pass	Zamani Team	20/10/2014
3.3	Purging	Pass	Zamani Team	20/10/2014