

Data Archive

**CS 4624 Multimedia/Hypertext/Information
Access**

Virginia Tech, Blacksburg, VA 24061

4.28.2017

**Authors: John Sizemore, Gil Turner, Irtiza Delwar, Michael
Culhane**

Client: Seungwon Yang

Instructor: Edward Alan Fox

Table of Contents

Table of Tables.....	4
Table of Figures.....	5
Section I: Executive Summary.....	7
Section II: Introduction.....	8
Section III: User Manual.....	9
3.1 About Zenodo.....	9
3.2 Creating Account and Logging In.....	9
3.3 Manually Uploading Data.....	10
3.3.1 Basic Information.....	11
3.3.2 Other Required/Recommended Information.....	13
3.3.3 Optional Information.....	15
3.4 Search.....	16
3.5 Communities.....	17
3.5.1 Create a Community.....	17
3.5.2 Edit a Community.....	18
3.5.3 Delete a Community.....	19
3.5.4 Community URLs.....	19
3.5.5 Curate Communities.....	19
3.5.6 View Communities.....	20
3.6 Metadata: Dublin Core.....	21
Section IV: Developer Manual.....	22
4.1 Local Installation.....	22
4.2 Repository Maintenance.....	25
4.2.1 Obtaining Access Token.....	25
4.2.2 The Upload Script.....	27
4.2.2.1 Single Upload Guide.....	28
4.2.2.2 Bulk Upload Guide.....	30
Section V: Lessons Learned.....	33
5.1 Planning Effectively for a Large Technical Project.....	33
5.2 The Importance of Communication.....	33
5.3 Diligence.....	33

5.4 Future Work.....	34
Section VI: Acknowledgements.....	35
Section VII: References.....	36
Section VIII: Appendix.....	38
Appendix I. Requirements.....	38
Appendix 1.1 Application Overview.....	38
Appendix 1.2 Client Information.....	38
Appendix 1.3 Objectives.....	38
Appendix 1.4 Team Roles and Responsibilities.....	39
Appendix 1.5 Non-Functional Requirements.....	40
Appendix 1.6 Requirements Timeline.....	41
Appendix 1.6.1 Progress So Far.....	42
Appendix 1.6.2 Work In Progress.....	42
Appendix 1.6.3 Next Steps.....	42
Appendix II: Design.....	43
Appendix 2.1 Tools Used.....	43
Appendix 2.2 Docker vs. Developmental Zenodo Installation.....	44
Appendix 2.3 User Guide Documentation Design.....	45
Appendix 2.4 Data Flow Design.....	45
Appendix 2.5 Testing Approach.....	46
Appendix III: Implementation.....	48
Appendix 3.1 Acquiring Server Credentials.....	48
Appendix 3.2 Docker and Zenodo Installation.....	48
Appendix 3.3 DSpace Installation.....	49
Appendix 3.4 Implementation Timeline.....	50
Appendix 3.5 Metadata Specifications.....	50
Appendix 3.6 DSpace: Additional Information.....	50
Appendix 3.7 Metadata: Additional Information.....	51
Appendix IV: Prototype.....	54
Appendix 4.1 Local Installation.....	54
Appendix 4.2 Upload.....	54
Appendix 4.3 Search.....	58
Appendix 4.4 Metadata: Dublin Core.....	59

Appendix V: Testing.....	60
Appendix 5.1 Searching for a Dataset by Title.....	60
Appendix 5.2 Searching for a Dataset by Author.....	61
Appendix 5.3 Searching for a Dataset by Publication Date.....	62
Appendix 5.4 Searching for a Dataset by Relevant Terms (i.e. Tags).....	62
Appendix 5.5 Searching for an .mp3 File.....	63
Appendix 5.6 Searching for an .mp4 File.....	63
Appendix VI: Refinement.....	64
Appendix 6.1 DSpace Installation.....	64
Appendix 6.2 Zenodo Prototype.....	64
Appendix 6.3 User & Admin Manual.....	66

Table of Tables

1 Basic Information for File Upload.....	12
2 Timeline and Task Description.....	41
3 Docker and Developmental Installation Comparison.....	45

Table of Figures

1	Zenodo Homepage.....	9
2	Zenodo Login/Sign Up Button.....	10
3	Zenodo Upload Button.....	10
4	Zenodo Upload UI.....	11
5	Zenodo Upload Basic Information UI.....	13
6	Zenodo Upload Licensing UI.....	14
7	Zenodo Upload Funding/Communities UI.....	15
8	Zenodo Upload Option Information UI.....	16
9	Zenodo Searching.....	16
10	Zenodo Communities Button.....	17
11	Zenodo Communities New Button.....	17
12	Zenodo Communities Edit Button.....	18
13	Zenodo Communities Delete Button.....	19
14	Zenodo Communities URLs.....	19
15	Zenodo Communities Curate Button.....	20
16	Zenodo Communities View Button.....	20
17	Zenodo Dublin Core Export UI.....	21
18	Zenodo Metadata Export UI.....	21
19	Cloning Zenodo Source Code.....	22
20	Checking Out Zenodo Master Branch.....	23
21	Initial Docker Shell.....	23
22	Running Docker-Compose Build.....	24
23	Running Docker-Compose Up.....	24
24	Zenodo Applications Link.....	26
25	Zenodo Applications Page.....	26
26	Zenodo API Token Name.....	27
27	Zenodo API Token Share.....	27
28	Setting ACCESS_TOKEN in Script File.....	28
29	Starting Zenodo Upload Script in Command Line.....	28
30	Selecting Single Upload Type in Upload Script.....	28
31	Entering Filename in Upload Script.....	28
32	Enter Full Path for File in Upload Script.....	29
33	Enter Information About the File.....	29
34	Publishing the File.....	29
35	CSV File for Bulk Upload.....	30
36	Starting Zenodo Upload Script.....	30
37	Selecting Bulk Upload Script.....	30

38	Entering the Formatted CSV File.....	31
39	Publishing the File.....	31
40	Script Automatically Uploading Files.....	31
41	Message Once Script is Completed.....	31
42	Data Flow Design.....	46
43	Gantt Chart Representing Progress.....	50
44	Zenodo Homepage.....	54
45	Zenodo Upload.....	55
46	Zenodo Upload Basic Information.....	56
47	Zenodo Upload Licensing.....	57
48	Zenodo Upload Funding/Communities.....	57
49	Zenodo Upload Optional Information.....	58
50	Zenodo Searching for Files.....	58
51	Zenodo Dublin Core Export.....	59
52	Zenodo Searching by Title.....	60
53	Zenodo Searching by Author.....	61
54	Zenodo Searching by Publication Date.....	62
55	Zenodo Searching by Relevant Terms.....	62
56	Zenodo Searching for .mp3 File.....	63
57	Zenodo Searching for .mp4 File.....	63
58	Zenodo API Sample Code.....	65
59	Zenodo API Sample Request.....	66

Section I: Executive Summary

The goal of this project was for students of the Multimedia, Hypertext, and Information Access course in the Data Archive group to work with their client, Professor Seungwon Yang, to create a maintainable Zenodo repository to manage multimedia and datasets collected for scientific research at LSU. Zenodo is an open source data repository created for managing research data.

The students have worked with the client and the LSU IT Department to get Zenodo and potentially DSpace (another open source research data repository) installed and running on the LSU server. This process involved configurations and dependencies detailed below in this report. The LSU server has a firewall which blocks access to pages hosted by the server to users outside of the local network. Because of these obstructions, the students determined it would be more feasible to work through the above tasks locally and leave the rest of the installation process to the LSU IT Department which has more experience with the server's configurations and dependencies.

The students wrote a script to do both a single and a batch deposit of datasets through the Zenodo API. The datasets that will be included in the future batch deposits will be provided by the client and will be relevant to scientific research taking place at LSU. The script is generic and it accepts a separate, easily updated document that contains the datasets to be uploaded to facilitate the process for non-technical users to update the repository.

Finally, the students have created a guide for the future Zenodo administrators detailing how to install and manage the repository and add datasets to the script to be automatically deposited as well as a guide for the users who will be using the datasets in Zenodo for their work.

Section II: Introduction

The first step for this project was to choose a platform on which to build a data archive. Our client works for the Information Science Department at Louisiana State University. He teaches classes and manages projects that collect datasets, so he wanted a repository that makes sharing them with the rest of the scientific community simple and cheap. That's why he chose "Zenodo - Research Shared" as the appropriate platform for this project. CERN laboratory created Zenodo and made it open source. The maintenance and storage of Zenodo repositories are and will continue to be provided by CERN. In the remainder of this report we will describe the work we accomplished this semester, the issues we had to deal with, and the deliverables we are leaving our client with.

Section III: User Manual

The Zenodo server installation will be used by our client's Master's level class "Introduction to Digital Curation", taught at the School of Information Studies at Louisiana State University. The data archive will also be used by other project teams at LSU to collaborate data. Here is the user guide:

3.1 About Zenodo

Zenodo is a research data repository that was created and is maintained/hosted by OpenAIRE and CERN (The European Organization for Nuclear Research). Zenodo was launched in 2013 allowing researchers to upload files up to 50GB in size. With Zenodo, users will be able to upload data to an archive.^[1] Uploaded data is shared with others depending on access level. You will be able to search for various datasets on the data archive. Zenodo also features communities which will allow groups to collaborate and congregate data together.^[13] Figure 1 below shows the Zenodo homepage.

The screenshot displays the Zenodo homepage with a blue header containing the Zenodo logo, a search bar, and navigation links for 'Upload' and 'Communities'. A user profile 'jizess@vt.edu' is visible in the top right. The main content area is divided into two columns. The left column, titled 'Recent uploads', lists several items with their dates, titles, and authors. Each item includes a 'View' button and a 'Report' button. The right column contains promotional text and icons for 'Join Zenodo at Google Summer of Code 2017', 'Using GitHub?', and 'Zenodo in a nutshell'. The 'Zenodo in a nutshell' section lists key features: Research, Shared; Citeable, Discoverable; Communities; Funding; and Flexible Licensing. A footer at the bottom left shows a URL: 'https://192.168.99.102/record/7574'.

Figure 1: Zenodo Homepage

3.2 Creating Account and Logging In

To use the data archive you should first create an account with Zenodo. The Sign Up button is located in the top right corner of the page on the header. If you already have an account then

you should make sure that you are logged in. The Log In button is located next to the Sign Up button. Figure 2 below shows the location of the button.



Figure 2. Login/Sign up button

Note: You **must** have an account before you can upload data to the data archive and create communities. However, you can search and view data in the archive without logging in.

3.3 Manually Uploading Data

1. To begin the upload process find the upload button that is on the header of the web page. Figure 3 below shows the location of the upload button.



Figure 3. Upload button

2. Once directed to the new upload page choose a file to upload, either by clicking on the “choose files” button or by dragging files in from your computer’s file explorer.

3. After choosing the files that you want to upload, select the data type. The options available include: Publication, Poster, Presentation, Dataset, Image, Video/Audio, Software, and Lesson.

4. After choosing the format, use the drop down menu to select the subtype of your upload. This section is required so you should make sure to accurately select your data type. Figure 4 below shows the graphical user interface of the first part of the upload process.

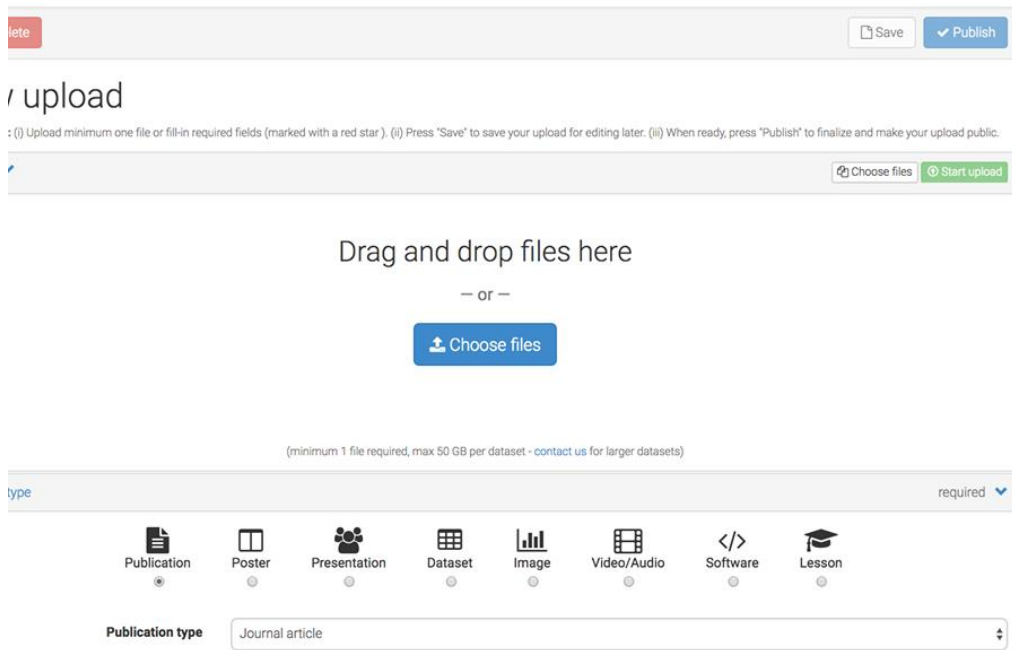


Figure 4. Zenodo Upload

3.3.1 Basic Information

The next step in the upload process is to supply basic information on the file upload. Figure 5 shows the Zenodo graphical user interface for inputting basic information about the file being uploaded. Table 1 describes the information that you should add to this section. Figure 5 below also shows the graphical user interface for the basic information needed for an upload.

Table 1: Basic Information for File Upload

Information Type	Description
Digital Object Identifier	Only input a value if you were told to by your publisher. If a DOI is not specified by the user, Zenodo will register one for you. Digital Object Identifiers are supplied to help others easily and unambiguously cite uploads ^[1] .
Publication Date*	Enter the date that the data was published in YYYY-MM-DD format. Note: Be sure to use the date the data was first published and not the current day's date.
Title*	Enter a title that accurately describes the data and allows other users to find your data easily.
Author(s)*	Enter the name of any authors who worked on collecting this data and their affiliation.
Description*	Enter an in-depth description for the data. A user reading the description should be able to understand what the data contains.
Keyword(s)	Optionally enter keywords to increase the amount of searchable terms that your data will appear for.
Additional Notes	Optionally add additional notes as needed.

*Denotes required information

Basic information required ▾

Digital Object Identifier

Optional. Did your publisher already assign a DOI to your upload? If not, leave the field empty and we will register a new DOI for you. A DOI allows others to easily and unambiguously cite your upload. Please note that it is NOT possible to edit a Zenodo DOI once it has been registered by us, while it is always possible to edit a custom DOI.

Publication date *

Required. Format: YYYY-MM-DD. In case your upload was already published elsewhere, please use the date of first publication.

Title *

Required.

Authors * ▾ ×

+ Add another author

Description *

Required.

Keywords ▾ ×

+ Add another keyword

Additional notes

Optional.

Figure 5. Zenodo Upload Basic Information

3.3.2 Other Required/Recommended Information

The next required input for uploading a file is the License section. While Zenodo does encourage all data to be shared, it also allows for varying levels of visibility, including open, embargo, restricted, and close access. Open access requires a license name, embargoed access requires a license name and an embargo date, restricted access requires conditions on which to grant other users access to the published data, and closed access doesn't require any additional information. Figure 6 shows the graphical user interface for the licensing information discussed above.^[13]

License
required ▼

Access right *

- Open Access
- Embargoed Access
- Restricted Access
- Closed Access

Required. Open access uploads have considerably higher visibility on Zenodo.

License *

Start typing a license name...

Required. The selected license applies to all of your files displayed in the top of the form. If you want to upload some files under a different license, please do so in two separate uploads. If you think a license is missing from the list, please inform us at info@zenodo.org

Figure 6. Zenodo Upload Licensing

In addition to all of the required information discussed above, there are other recommended sections that may be filled out. These sections include communities, funding, and alternate identifiers. Figure 7 below shows the graphical user interface for this section.

- Communities allow groups to have files uploaded and grouped together. Communities are explained in detail in Section 3.5. Communities allow for groups to have their own digital repository.^[13]
- The following section includes information on funding. If you have any grants that fund your research you can enter it in this section.
- Finally we have related/alternative identifiers. In this section you should include identifiers for any data that is related to the one your are uploading.

Communities
recommended ▼

Any user can create a community collection on Zenodo ([browse communities](#)). Specify communities which you wish your upload to appear in. The owner of the community will be notified, and can either accept or reject your request.

Communities

×

[+ Add another community](#)

Funding
recommended ▼

Zenodo is integrated into reporting lines for research funded by the European Commission via OpenAIRE (<http://www.openaire.eu>). Specify grants which have funded your research, and we will let your funding agency know!

Grants

×

Optional. European Commission FP7 and Horizon 2020 grants only. For general funding acknowledgements, please use the **Additional Notes** field.
 Note: a human Zenodo curator will need to validate your upload - you may experience a delay before it is available in OpenAIRE.

[+ Add another grant](#)

Related/alternate identifiers
recommended ▼

Specify identifiers of related publications and datasets. Supported identifiers include: DOI, Handle, ARK, PURL, ISSN, ISBN, PubMed ID, PubMed Central ID, ADS Bibliographic Code, arXiv, Life Science Identifiers (LSID), EAN-13, ISTC, URNs and URLs.

Related identifiers

×

[+ Add another related identifier](#)

Figure 7. Zenodo Upload Funding/Communities

3.3.3 Optional Information

The final part of the upload process includes various optional information that the user can input. Figure 8 shows the types of optional information that the user can add to the upload. For more information on each of these individual fields, simply click on them to expand their descriptions.

Contributors	optional >
References	optional >
Journal	optional >
Conference	optional >
Book/Report/Chapter	optional >
Thesis	optional >
Subjects	optional >

Figure 8. Zenodo Upload Optional Information

3.4 Search

The screenshot shows the Zenodo search interface. At the top, there is a search bar with the text 'Test Data' and a search icon. To the right of the search bar are links for 'Upload' and 'Communities', and a user profile dropdown for 'jsize8@vt.edu'. Below the search bar, the results are displayed. On the left side, there are three filter panels: 'Access Right' (Open (57), Closed (2), Restricted (1)), 'File Type' (Pdf (45), Pptx (7), Docx (6), Doc (1)), and 'Keywords' (Cern (47), Openlab (46), Student (40), Summer (40), Data (6), Big (3), Computing (3), Analytics (2), Cloud (2), Lhc (2)). The 'Type' panel at the bottom left shows 'Publication (52) +', 'Presentation (7)', and 'Dataset (1)'. The main search results area shows 'Found 60 results.' with pagination controls (1, 2, 3) and a 'Sort by:' dropdown set to 'Best match' with an 'asc.' option. Three results are visible, each with a date, title, author, and a 'View' button. The first result is dated 'April 6, 2017' and is a 'Dataset' with 'Closed Access', titled 'Test Data' by John Sizemore, uploaded on April 6, 2017. The second result is dated 'September 1, 2015' and is a 'Report' with 'Open Access', titled 'RAPIDIO USAGE IN A BIG DATA ENVIRONMENT' by Costa, Jorge; Barring, Olof, uploaded on February 16, 2016. The third result is dated 'September 1, 2013' and is a 'Report' with 'Open Access', titled 'Improved metrics collection and correlation for the CERN cloud storage test framework' by Lindqvist, Carolina; Zotes, Maitane; Heikkila, Seppo, uploaded on March 10, 2014. The fourth result is dated 'August 31, 2013' and is a 'Report' with 'Open Access', titled 'Evaluation of in-memory database TimesTen' by Andras Simon, Endre; Potocky, Miroslav, uploaded on March 10, 2014.

Figure 9. Zenodo Searching for Deposited Files

Another important feature in Zenodo is the search functionality. Figure 9 above shows the page that displays the results of a search query. With the search functionality, the user can search for a specific set of data within the data archive. The user can search for an archived file by searching for any specific metadata that was entered during the upload of that specific file.

Figure 9 shows a search done for the keyword “test data”. To correctly use the search feature, enter any specific keyword pertaining to the desired data to find it. The search results show the following information: date published, data type, access type, title, description, and upload date. You can also click the view button to view the data in that search box. On the left hand side there are various facets to use to narrow down the amount of items returned in the search. With this feature you will have a more refined search to work with.

3.5 Communities

Zenodo communities are a way for groups to collaborate. With communities groups can have files uploaded and grouped together, essentially allowing groups to have their own digital repository.^[13] The communities button is located on the header next to the upload button. Figure 10 below shows the location of the communities button.



Figure 10. Communities button

The communities web page can be found at the following link: <https://zenodo.org/communities/>. On the webpage there is a search bar that allows the users to search for communities. With communities users can find information that is part of the same project in an easy and efficient manner. To create a community you must be logged in.

3.5.1 Create a Community

1. Go to the communities web page on the Zenodo website
2. Click on the “New” button located on the right side of the screen in a gray box. Figure 11 shows an image for this step.

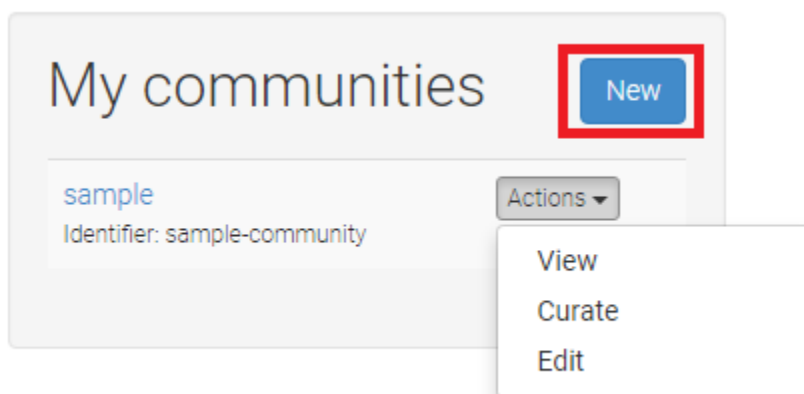


Figure 11. Communities “New” button

3. The web page will direct you to a form to fill out information
4. The following information is required:
 - a. Identifier - identifies the community and is included in the community url. This cannot be modified later, so wisely decide on a name for the community.
 - b. Title - a title for your community
5. The following information is optional:
 - a. Description - Use this box to accurately describe your community.
 - b. Curation - Describe the policy in which you will accept and reject new uploads to this community.
 - c. Page - A long description of the community that will be displayed on a separate page linked on the index.
 - d. Logo - An image to aid and promote public recognition.
6. Click Create

3.5.2 Edit a Community

Go to the edit communities web page on the zenodo website.

From the homepage:

1. Click on communities button
2. Find your community on the right hand panel
3. Click on the actions drop down and select edit as seen in Figure 12.

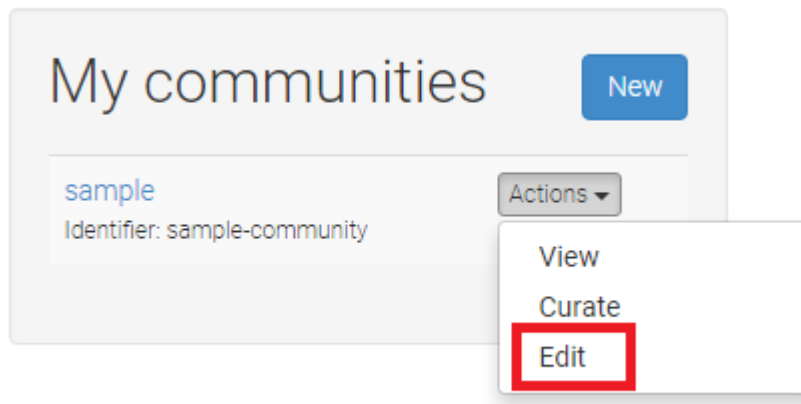


Figure 12. Communities “edit” dropdown button

4. Type the URL
 - a. <https://zenodo.org/communities/<your-community-name-here>/edit/>

On the Edit Communities page you can also edit the information entered when the community is created other than the identifier. This includes: Title, Description, Curation Policy, Page, and Logo. Make sure to save any information that you edit.

3.5.3 Delete a Community

You can also delete the community on this page. The Delete button is located at the bottom of the page. Figure 13 shows the button to delete a community.

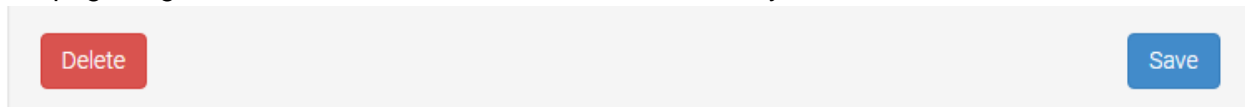


Figure 13: Communities “Delete” button

3.5.4 Community URLs

Find a list of community URLs on the Edit Communities page shown in Figure 14:

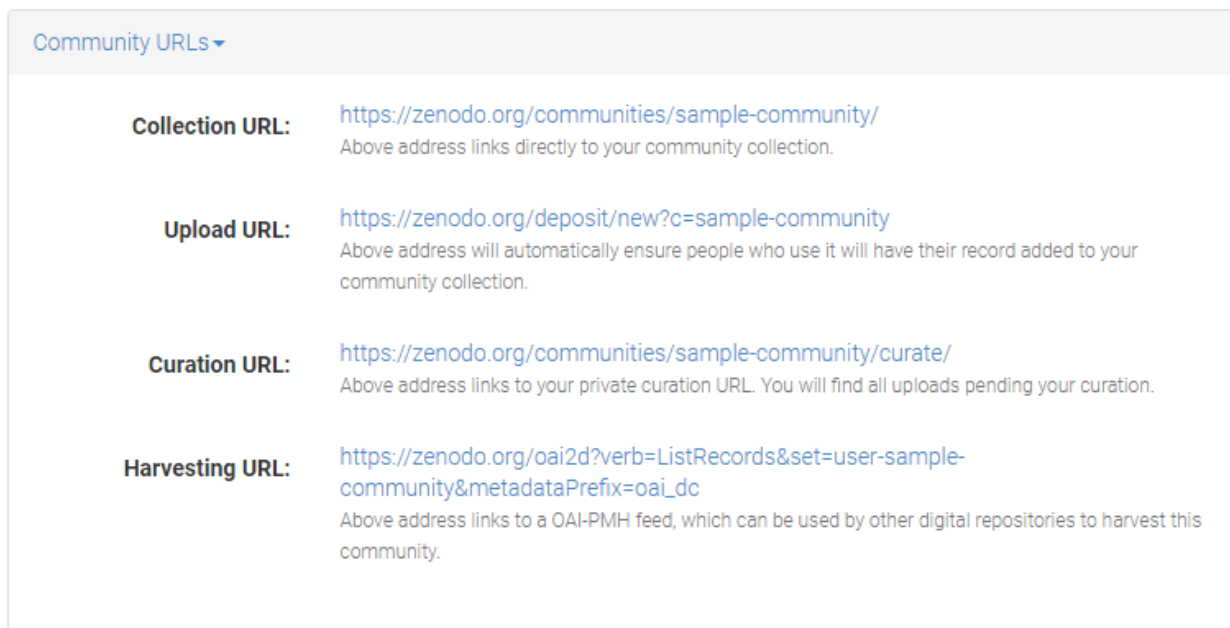


Figure 14. Community URLs

3.5.5 Curate Communities

To curate your community go to the curate communities webpage.

From the homepage:

1. Click on the Communities buttons
2. Find your community on the right hand panel
3. Click on the actions drop down and select curate as seen in Figure 15

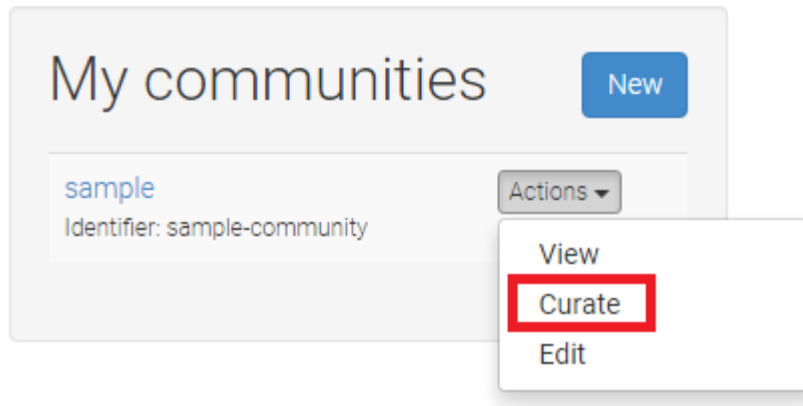


Figure 15. Communities “Curate” dropdown button

On this page you can select an uploaded file and either accept or reject the file to your community.

3.5.6 View Communities

To View your community go to the view communities webpage.

From the homepage:

1. Click on the Communities buttons
2. Find your community on the right hand panel
3. Click on the actions drop down and select View. Figure 16 shows an image for this step.

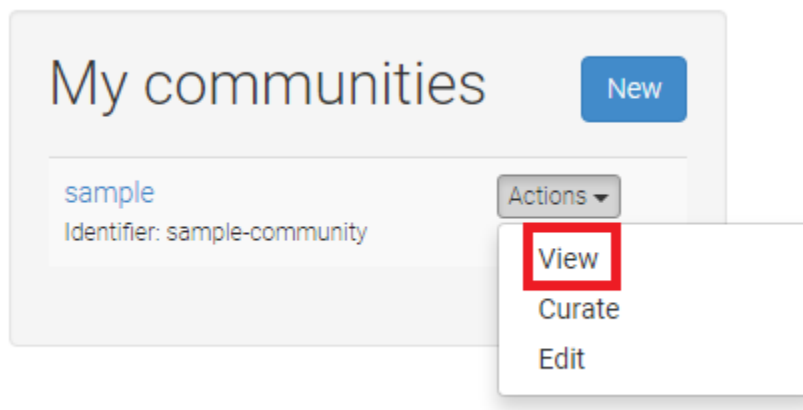


Figure 16. Communities “View” dropdown button

On this page you can view information about your community and any files that are uploaded to your community.

3.6 Metadata: Dublin Core

Zenodo has its own interface for providing metadata for a dataset during deposit, whether it is done manually through the GUI or through the API. Once a dataset has been deposited, Zenodo provides a tool to export its corresponding metadata in various formats. This is shown in Figure 17, where we attained a Dublin Core export for one of our test datasets. This Dublin Core export contains tags for fields such as creator, date, description, and subject, all of which are provided in .xml format.

April 6, 2017

Dataset Closed Access

Test Data

John Sizemore

Dublin Core Export

```
<?xml version='1.0' encoding='utf-8'?>
<oai_dc:dc xmlns:dc="http://purl.org/dc/elements/1.1/" xmlns:oai_dc="http://www.openarchives.org/OAI/2.0/oai_dc/" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://www.openarchives.org/OAI/2.0/oai_dc/ http://www.openarchives.org/OAI/2.0/oai_dc.xsd">
  <dc:creator>John Sizemore</dc:creator>
  <dc:date>2017-04-06</dc:date>
  <dc:description>&lt;p&gt;This is test data.&lt;/p&gt;</dc:description>
  <dc:identifier>10.1234/foo.bar</dc:identifier>
  <dc:identifier>https://zenodo.org/record/45450</dc:identifier>
  <dc:identifier>oai:zenodo.org:45450</dc:identifier>
  <dc:rights>info:eu-repo/semantics/closedAccess</dc:rights>
  <dc:subject>test</dc:subject>
  <dc:subject>data</dc:subject>
  <dc:title>Test Data</dc:title>
  <dc:type>info:eu-repo/semantics/other</dc:type>
</oai_dc:dc>
```

Figure 17. Zenodo Dublin Core (Metadata) Export

To get to the export section do the following:

1. Find a file that you want to view
2. Click on view to go the document's page
3. Find the export box shown in Figure 18
4. Click on Dublin Core

Export

[BibTeX](#) [CSL](#) [DataCite](#) [Dublin Core](#) [JSON](#)
[MARCXML](#) [Mendeley](#)

Figure 18. Export data

Section IV: Developer Manual

4.1 Local Installation

Below are the steps necessary to complete the Zenodo installation on a machine.

Note: *Docker must be installed prior to this process and root access to the machine is required.*

Step 1: Obtain access to the terminal and navigate to a desired path to put the Zenodo project.

Step 2: Clone the project from git by using the following command.

```
cd ~/src/
```

```
git clone https://github.com/zenodo/zenodo.git
```

The result should look like this:

```
michaels-MacBook-Pro-4:src mikeculhane$ sudo git clone https://github.com/zenodo/zenodo.git
Cloning into 'zenodo'...
remote: Counting objects: 13698, done.
remote: Compressing objects: 100% (194/194), done.
remote: Total 13698 (delta 117), reused 0 (delta 0), pack-reused 13495
Receiving objects: 100% (13698/13698), 8.15 MiB | 6.28 MiB/s, done.
Resolving deltas: 100% (9446/9446), done.
michaels-MacBook-Pro-4:src mikeculhane$ █
```

Figure 19. Cloning Zenodo source code.

Step 3: Checkout the master branch with the following commands.

```
cd ~/src/zenodo
```

```
git checkout master
```

```
[michaels-MacBook-Pro-4:src mikeculhane$ cd zenodo
[michaels-MacBook-Pro-4:zenodo mikeculhane$ sudo git checkout master
Already on 'master'
Your branch is up-to-date with 'origin/master'.
michaels-MacBook-Pro-4:zenodo mikeculhane$
```

Figure 20. Checking out Zenodo master branch

Step 4: Start up a Docker shell.

```

          ##          .
        ## ## ##    ==
       ## ## ## ## ## ===
  /"""""""""""""""""\  ===
 ~~~ { ~~~~ { ~~~~ } ~~~ } ~~~~
  \_____/  \_____/  \_____/

```

```
docker is configured to use the default machine with IP 192.168.99.100
For help getting started, check out the docs at https://docs.docker.com
```

```
michaels-MacBook-Pro-4:zenodo mikeculhane$
```

Figure 21. Initial Docker shell

Note: Store the default machine IP which is given when the initial Docker shell is started (as seen in Figure 21).

Step 5: Run docker-compose build.

```

michaels-MacBook-Pro-4:zenodo mikeculhane$ docker-compose build
mg uses an image, skipping
db uses an image, skipping
cache uses an image, skipping
Building static
Step 1/24 : FROM python:3.5
3.5: Pulling from library/python
6d827a3ef358: Downloading [=====>] 21.5MB/51.44MB
2726297beaf1: Download complete
7d27bd3d7fec: Downloading [=====>] 41.39MB/42.57MB
44ae682c18a3: Downloading [=>] 3.784MB/129.9MB
824bd01a76a3: Waiting
27387ba2d03b: Waiting
4b905e4f59a3: Waiting

```

Figure 22. Running docker-compose build

Step 6: Run docker-compose up

Note: For the remainder of this guide, we will assume that every command is executed in the `~/src/zenodo` directory.

```

Johns-MacBook-Pro-10:zenodo jsize85$ docker-compose up
zenodo_es_1 is up-to-date
zenodo_db_1 is up-to-date
zenodo_cache_1 is up-to-date
Starting zenodo_static_1
zenodo_mq_1 is up-to-date
zenodo_statsd_1 is up-to-date
zenodo_web_1 is up-to-date
zenodo_worker_1 is up-to-date
zenodo_frontend_1 is up-to-date
zenodo_lb_1 is up-to-date
Attaching to zenodo_es_1, zenodo_db_1, zenodo_cache_1, zenodo_mq_1, zenodo_statsd_1, zenodo_static_1, zenodo_web_1, zenodo_worker_1, zenodo_frontend_1, zenodo_lb_1
static_1  | ==> /var/log/alternatives.log <==
static_1  | update-alternatives 2017-03-21 19:12:04: run with --install /usr/bin/stream stream /usr/bin/stream-im6 100 --slave /usr/share/man/man1/stream.1.gz stream.1.gz /usr/sha
re/man/man1/stream-im6.1.gz
static_1  | update-alternatives 2017-03-21 19:12:04: link group stream updated to point to /usr/bin/stream-im6
static_1  | update-alternatives 2017-03-21 19:12:04: run with --install /usr/bin/display display /usr/bin/display-im6 100 --slave /usr/share/man/man1/display.1.gz display.1.gz /us
r/share/man/man1/display-im6.1.gz
static_1  | update-alternatives 2017-03-21 19:12:04: link group display updated to point to /usr/bin/display-im6
static_1  | update-alternatives 2017-03-21 19:12:04: run with --install /usr/bin/montage montage /usr/bin/montage-im6 100 --slave /usr/share/man/man1/montage.1.gz montage.1.gz /us
r/share/man/man1/montage-im6.1.gz
static_1  | update-alternatives 2017-03-21 19:12:04: link group montage updated to point to /usr/bin/montage-im6
static_1  | update-alternatives 2017-03-21 19:12:04: run with --install /usr/bin/mogrify mogrify /usr/bin/mogrify-im6 100 --slave /usr/share/man/man1/mogrify.1.gz mogrify.1.gz /us
r/share/man/man1/mogrify-im6.1.gz
static_1  | update-alternatives 2017-03-21 19:12:04: link group mogrify updated to point to /usr/bin/mogrify-im6
static_1  | update-alternatives 2017-04-06 22:01:11: run with --quiet --install /usr/bin/nodejs node /usr/bin/nodejs 50 --slave /usr/share/man/man1/node.1.gz node.1.gz /usr/share/ma
n/man1/nodejs.1.gz
static_1  | update-alternatives 2017-04-06 22:01:11: link group node updated to point to /usr/bin/nodejs
static_1  |
static_1  | ==> /var/log/apt <==
static_1  |
static_1  | ==> /var/log/bootstrap.log <==
static_1  | Setting up debian-archive-keyring (2014.3) ...
static_1  | Setting up libatcd+6:amd64 (4.9.2-10) ...
static_1  | Setting up libapt-pkg4.12:amd64 (1.0.9.8.4) ...
static_1  | Setting up libusb-0.1-4:amd64 (2:0.1.12-25) ...
static_1  | Setting up libreadline6:amd64 (6.3-8+b3) ...
static_1  | Setting up netbase (5.3) ...
static_1  | Setting up inetutils-ping (2:1.9.2-39.3a460-3) ...
static_1  | Setting up gmp3 (1.4.18-7+deb8u3) ...
static_1  | Setting up apt (1.0.9.8.4) ...
static_1  | Processing triggers for libc-bin (2.19-18+deb8u7) ...
static_1  | tail: error reading '/var/log/apt': Is a directory
static_1  | tail: cannot open '/var/log/btmp' for reading: Permission denied
static_1  | tail: cannot open '/var/log/dmesg' for reading: Permission denied
static_1  |
static_1  | ==> /var/log/dpkg.log <==
static_1  | 2017-04-06 22:01:09 status half-installed nodejs:amd64 6.10.2-1nodesource1-jessiel
static_1  | 2017-04-06 22:01:11 status unpacked nodejs:amd64 6.10.2-1nodesource1-jessiel
static_1  | 2017-04-06 22:01:11 status unpacked nodejs:amd64 6.10.2-1nodesource1-jessiel
static_1  | 2017-04-06 22:01:11 startup packages configure
static_1  | 2017-04-06 22:01:11 configure nodejs:amd64 6.10.2-1nodesource1-jessiel <none>
static_1  | 2017-04-06 22:01:11 status unpacked nodejs:amd64 6.10.2-1nodesource1-jessiel
static_1  | 2017-04-06 22:01:11 status unpacked nodejs:amd64 6.10.2-1nodesource1-jessiel
static_1  | 2017-04-06 22:01:11 status half-configured nodejs:amd64 6.10.2-1nodesource1-jessiel

```

Figure 23. Running docker-compose up

After running docker-compose up the terminal should be in a suspended state.

Step 7: While keeping the original Docker shell alive, start a new one. In the new shell run the following commands.

```
cd ~/src/zenodo  
  
docker-compose run --rm web bash /code/zenodo/scripts/init.sh  
  
docker-compose run --rm statsd bash /init.sh
```

Step 8: Load the demo records and index them using the following four commands.

```
docker-compose run --rm web zenodo fixtures loaddemorecords  
  
docker-compose run --rm web zenodo migration recordsrun  
  
docker-compose run --rm web zenodo migration reindex -t recid  
  
docker-compose run --rm web zenodo index run -d
```

Step 9: Visit the local Zenodo host running at the following URL.

```
https://<docker ip>
```

Note: “docker ip” is the IP address specified in step 4.

4.2 Repository Maintenance

4.2.1 Obtaining Access Token

In order for your account to be allowed to perform Zenodo API calls and use the upload script, you must first obtain an access token that is associated with your account. First go to the Zenodo page at: <https://zenodo.org/> and login. Once you’ve logged into your account, go to the top right of the page, click the drop down arrow next to your email and then click applications.

The screenshot shows the Zenodo website interface. At the top, there is a search bar and navigation links for 'Upload' and 'Communities'. The user's profile 'ttg195@vt.edu' is visible in the top right corner, with a dropdown menu containing options: Profile, Change password, Linked accounts, Applications, Shared links, GitHub, and Log out. The main content area displays 'Recent uploads' with three items:

- Transcription initiation peaks based on FANTOM5 CAGE data on hg38 and mm10** (April 12, 2017, Dataset, Open Access). Overview: Decomposition-based peak identification (DPI) is applied to re-processed FANTOM5 data.
- Replication Archive for "Earnings Inequality and Mobility Trends in the United States: Nationally Representative Estimates from Longitudinally Linked Employer-Employee Data"** (April 12, 2017, Software, Open Access). by John M. Abowd, Kevin L. McKinney, and Nellie L. Zhao.
- Sample calculation using protein APR** (April 11, 2017, Dataset, Open Access). Germano Heinzelmann, Niel Henriksen, Michael Gilson.

On the right side, there are promotional boxes for 'Join Zenodo at Google Summer of Code 2017', 'Using GitHub?', and 'Zenodo in a nutshell' which lists features like Research, Shared, Citeable, Discoverable, Communities, Funding, and Flexible licensing.

Figure 24. Zenodo Applications Link

Next, find the “Personal access token” box and click the “New token” button inside that box:

The screenshot shows the 'Applications' page on Zenodo. The breadcrumb trail is 'Home / Account / Applications'. On the left, a 'Settings' sidebar includes links for Profile, Change password, Linked accounts, Applications (highlighted), Shared links, and GitHub. The main content area is divided into three sections:

- Developer Applications:** A box with a '+ New application' button and the text: 'You have not yet registered any applications. Click the 'New application' button to create an application to access the API.'
- Personal access tokens:** A box with a '+ New token' button and the text: 'Following are personal tokens used to access the API:'. Below this, a table lists one token: 'Touch My Burrito - deposit:actions, deposit:write'.
- Authorized applications:** A box with the text: 'You have not yet granted any external applications access to your account.'

Figure 25. Zenodo Applications Page

Next, enter in a name for your access token and also check the permissions you want. We recommend selecting both the `deposit:actions` and `deposit:write` scopes so the script will be allowed to both upload and publish the data sets. Click create once you're done.

Home / Account / Applications / New

Settings

- Profile
- Change password
- Linked accounts
- Applications**
- Shared links
- GitHub

New personal access token

Name

Name of personal access token.

Scopes

- deposit:actions
Allow publishing of uploads.
- deposit:write
Allow upload (but not publishing).

Scopes assign permissions to your personal access token. A personal access token works just like a normal OAuth access token for authentication against the API.

Figure 26. Token Name

On the next screen, there will be a red box with your access token. Save the token **now**. Once you leave that page, there is **no way** to get that access token. Once you have the token saved somewhere, click save.

Home / Account / Applications / Edit

Settings

- Profile
- Change password
- Linked accounts
- Applications**
- Shared links
- GitHub

Personal access token / Testing Token

Access token

wf75zCkgrC1HmJ2Hkq5K3MjEC3Qd5YmeDk6NrhI.6ebg4 [REDACTED]

Do not share this personal access token. It gives full access to your account.

Name

Testing Token

Name of personal access token.

Scopes

- deposit:actions
Allow publishing of uploads.
- deposit:write
Allow upload (but not publishing).

Scopes assign permissions to your personal access token. A personal access token works just like a normal OAuth access token for authentication against the API.

Figure 27. Token Save

4.2.2 The Upload Script

There is a script called ZenodoUploadPython2.py (or ZenodoUploadPython3.py if you are using Python 3 instead of Python 2) which allows for either Single or Bulk upload of data files onto Zenodo. The script was written in Python and there are two versions of the script. One script supports Python 2, which is the Python version located on the LSU server, and the other version supports Python 3 which is the version commonly used on modern day computers. However, both of the scripts require the Python requests library in order for it to run. The easiest way to install this library if you do not have it is to do a pip install. In order to do this, open up your terminal and enter in: “pip install requests”, which will start the installation for the Python requests library.

For both scripts, you need to enter in your access token that you saved from Figure 27. In order to do so, open the script and go to line 8 which is under the imports. On that line you should see a variable called ACCESS_TOKEN. If this variable is not on line 8, find it near the top of the file. After finding the ACCESS_TOKEN variable, set it equal to the access token you saved earlier as shown in Figure 28. Doing this step is necessary to link the uploading done by the script to your personal Zenodo account.

```
8 ACCESS_TOKEN = "BqF6xslBYmmleTvpY4esm04B9Ct5gPSfuzo8gqgoHw..."
```

Figure 28. Setting the ACCESS_TOKEN so that the script will connect to your account.

4.2.2.1 Single Upload Guide:

To start running the script, go to the path the script is located in. Once you are in the correct directory, enter:

```
python ZenodoUploadPython2.py
```

Note: If you have the script on your local machine and have Python 3 installed, enter:

```
py ZenodoUploadPython3.py
```

```
C:\Users\Gil\Documents\CS 4624\Data Archive 4624 Project>python ZenodoUploadPython2.py
```

Figure 29. Starting the Zenodo upload script.

Once you run the Python script, as shown in Figure 29, the user will be prompted to enter the type of upload they will be performing. For single upload, enter in “Single” into the command line. If that step was successful, a 201 status message will be displayed in the command line as shown in Figure 30.

```
What type of upload are you performing? 'Single' or 'Bulk' : Single
201
```

Figure 30. Selecting single upload type.

Next, enter in the name of the file exactly as shown in Figure 31.

Note: “Exactly” means that spacing and case does matter when you enter the filename.

```
Enter in filename: typingData.CSV
```

Figure 31. Entering the filename.

After you enter in the filename, you will be prompted to enter the full path that the file is located in as shown in Figure 32. Once again, case and spacing do matter. If this step as successful, a 201 status message will be displayed in the command line. If an error has occurred during this

step, an error message saying that it cannot find the file or directory will be displayed to the user.

```
Enter in the full path for the file: /home/zenodo/video_csv/typingData.CSV
201
```

Figure 32. Entering the full path for the file.

Once you have specified the full path and received the successful status code, you will then be prompted to enter in some information about the file you are uploading onto Zenodo. The information includes the title for the file, a description for the file you are uploading, the name of the person who created the file, the affiliation for the author, and the upload type for your file as shown in Figure 33. Once you have entered in the information, you will receive a 200 status message if this step was successful.

Note: The different upload types you can enter include publication, poster, presentation, dataset, image, video, and software.^[16] When entering in the type for the file, it is case sensitive so please enter the type in lowercase. You can change the information as well as enter more detailed information about the file by logging in to the Zenodo website and then going to the Upload section and clicking on the file.

```
Enter in a title for the data: Typing Data
Enter in a description for the data: Info about typing
Enter in the name for the person who created the dataset: Seungwon Yang
Enter the affiliation of the author: LSU
Enter in the type for your file (image, video, dataset, etc.): dataset
200
```

Figure 33. Entering information about the file.

After successfully entering in the information about the file, you will be prompted to enter in whether or not you want to publish the file as shown in Figure 34. Enter 'y' if you do or enter 'n' if you do not. A message will be displayed stating whether or not your file was uploaded and published. If you did not publish your file, you can do so at anytime by logging into the Zenodo website, going to the Upload section, clicking on the file, and then clicking the publish button on that page.

Note: Once you publish a file on Zenodo, **removal or modification is not allowed**. The reason behind this is that once you publish the file, Zenodo will register a DOI for your file using Datacite.^[13]

```
Do you wish to publish the data set?: y/n n
Data set was uploaded but not published.
[zenodo@bagua src]$
```

Figure 34. Publishing the file.

4.2.2.2 Bulk Upload Guide:

The script also has the capability to perform a bulk upload so that you will not have to upload files one at a time. In order to perform a bulk uploading using the script you need to create a csv file that follows the format shown in Figure 35. The first row should contain information about what their corresponding columns are. Each of the rows after the first correspond to a file you are trying to upload onto Zenodo. The first column should include the name of the file, which is case sensitive. The second column should include the full path that the file is located in. The third column should include a title for the file you are trying to upload. The fourth column should include a description of the file you are trying to upload. The fifth column should include the author of the file you are uploading. The sixth column should include the author's affiliation. Finally, the seventh column should include the type for the file you are trying to upload. The different upload types you can enter include publication, poster, presentation, dataset, image, video, and software.^[16] When entering in the type for the file, it is case sensitive so please enter the type in lowercase.

	A	B	C	D	E	F	G
1	filename	path	title	des	author	aff	type
2	handwritingData.csv	/home/zenodo/video_csv/handwritingData.csv	Handwriting Data	Information about	Seungwon Yang	LSU	dataset
3	typingData.CSV	/home/zenodo/video_csv/typingData.CSV	Typing Data	Information about	Seungwon Yang	LSU	dataset
4	1.Concrete_Pouring.mp3	/home/zenodo/construction_sound_samples/1.C	Concrete Pouring Soun	An mp3 file of conc	Seungwon Yang	LSU	video
5	2.Drilling Sound.m4a	/home/zenodo/construction_sound_samples/2.C	Sounds of Drilling	An mp3 file of drilli	Seungwon Yang	LSU	video

Figure 35. CSV file format for bulk upload.

To start running the script, go to the path the script is located in. Once you are in the correct directory, enter:

```
python ZenodoUploadPython2.py
```

Note: If you have the script on your local machine and have Python 3 installed, enter:

```
py ZenodoUploadPython3.py
```

```
C:\Users\Gil\Documents\CS 4624\Data Archive 4624 Project>python ZenodoUploadPython2.py
```

Figure 36. Starting the Zenodo upload script

Once you run the Python script, as shown in Figure 36, the user will be prompted to enter the type of upload they will be performing. For single upload, enter in "Bulk" into the command line as shown in Figure 37.

```
What type of upload are you performing? 'Single' or 'Bulk' : Bulk
```

Figure 37. Selecting bulk upload type.

Next, enter in the name of the csv file whose format matches Figure 35's.

Note: The filename is case sensitive as shown in Figure 38.

```
Enter in the name of the csv file for bulk upload: ServerBulkUploadTest.csv
```

Figure 38. Entering the formatted csv file.

After entering in the name of the csv file, you will be asked whether or not you want to publish the files as shown in Figure 39. Enter “y” if you do or “n” if you do not wish to. If you did not publish your file, you can do so at anytime by logging in to the Zenodo website and then going to the Upload section and clicking on the file and then clicking the publish button on that page.

Note: Once you publish a file on Zenodo **removal or modification is not allowed**. The reason behind this is that once you publish the file, Zenodo will register a DOI for your file using Datacite.^[13]

```
Do you wish to publish the data set?: y/n n
```

Figure 39. Publishing the file.

Next, the script will automatically start uploading the files onto Zenodo and once done will prompt a message saying that the data set was uploaded and either published or not depending on what you answered in the previous step as shown in Figure 30 and 41.

```
201
{u'files': [], u'links': {u'files': u'https://zenodo.org/api/deposit/depositions/556446/files', u'edit': u'https://zenodo.org/api/deposit/depositions/556446/actions/edit', u's
elf': u'https://zenodo.org/api/deposit/depositions/556446', u'bucket': u'https://zenodo.org/api/files/5189fab1-c3c7-4d5e-94c0-e27213b2dbf6', u'publish': u'https://zenodo.org/a
pi/deposit/depositions/556446/actions/publish', u'html': u'https://zenodo.org/deposit/556446', u'discard': u'https://zenodo.org/api/deposit/depositions/556446/actions/discard'
}, u'created': u'2017-04-20T20:41:45.154950+00:00', u'title': u'', u'modified': u'2017-04-20T20:41:45.154957+00:00', u'submitted': False, u'record_id': 556446, u'state': u'uns
ubmitted', u'owner': 30234, u'id': 556446, u'metadata': {u'prereserve_doi': {u'doi': u'10.5281/zenodo.556446', u'recid': 556446}}}
201
<bound method Response.json of <Response [201]>>
200
201
{u'files': [], u'links': {u'files': u'https://zenodo.org/api/deposit/depositions/556447/files', u'edit': u'https://zenodo.org/api/deposit/depositions/556447/actions/edit', u's
elf': u'https://zenodo.org/api/deposit/depositions/556447', u'bucket': u'https://zenodo.org/api/files/fb606c32-d788-4b0e-b0cc-d59b3f23624f', u'publish': u'https://zenodo.org/a
pi/deposit/depositions/556447/actions/publish', u'html': u'https://zenodo.org/deposit/556447', u'discard': u'https://zenodo.org/api/deposit/depositions/556447/actions/discard'
}, u'created': u'2017-04-20T20:41:48.853848+00:00', u'title': u'', u'modified': u'2017-04-20T20:41:48.853856+00:00', u'submitted': False, u'record_id': 556447, u'state': u'uns
ubmitted', u'owner': 30234, u'id': 556447, u'metadata': {u'prereserve_doi': {u'doi': u'10.5281/zenodo.556447', u'recid': 556447}}}
201
<bound method Response.json of <Response [201]>>
200
201
{u'files': [], u'links': {u'files': u'https://zenodo.org/api/deposit/depositions/556448/files', u'edit': u'https://zenodo.org/api/deposit/depositions/556448/actions/edit', u's
elf': u'https://zenodo.org/api/deposit/depositions/556448', u'bucket': u'https://zenodo.org/api/files/14520119-12ff-4dcc-b3a5-263ab56da8eb', u'publish': u'https://zenodo.org/a
```

Figure 40. The script automatically uploading the files.

```
Data set was uploaded but not published.
[zenodo@bagua src]$
```

Figure 41. Message displayed once the script is completed.

You can verify that the script successfully uploaded and/or published the files by logging into the Zenodo website and then going to the Upload section.

Section V: Lessons Learned

5.1 Planning Effectively for a Large Technical Project

We have learned the importance of separating long term and short term planning and updating our plans as we run into issues along the way.

Initially we did not consider the fact that software installation would account for so much of the work during this project. We initially projected that we would have it done within the first week, perhaps within even one meeting session. It did not take too long, however, for us to realize how much of a hindrance this would be to the Data Archive, and we eventually found that we would be spending about half of our total time working on software installations and configurations for all the various components.

We realize that it was a lack of experience with many of the tasks we set out to complete this semester that led to such a slow start for us. We now understand the importance an expert or technical leader plays in software projects in general, and we consider this to be a valuable insight as we graduate and begin our journey in the workforce.

5.2 The Importance of Communication

When working on a project for a client it is very important to have a constant stream of communication with that client. In addition it is also very important to be receptive to the client's feedback and change the project and project goals accordingly. When we started this project we had explicit goals which we didn't expect to change but as the project progressed we altered our goals to be more in line with our client's needs which changed over time. Without the constant stream of communication with our client we would not have been able to have such a dynamic set of goals set around our client's changing needs such as getting DSpace and Tomcat installed on the LSU server.

Our client was located in Louisiana which is too far away to reasonably communicate in person, and as a result we relied heavily on Skype, Google Hangouts, and Gmail as our main sources of communication. Relying on remote communication highlighted the importance of client communication since we only had a limited number of Skype/Google Hangouts meetings.

5.3 Diligence

Many aspects of this project required us to use strong focus and determination. For example, there were quite a few subtle points with the configurations including ports, permissions, and reading technical descriptions of the technologies. Because there were so many minor issues to account for, it required us to narrow our focus and take a more precise approach to the work.

We also reached a few points in our timeline where we felt we were not making adequate progress towards our final goals and felt a little discouraged. We persisted, however, and finally made some breakthroughs which gave us the momentum continue to finish the project in time.

5.4 Future Work

Currently both the Zenodo developer and user guides have been created, a python script with various use cases (single deposit and batch deposit along with publishing options) have been written, and all of the supplied datasets have been uploaded to Zenodo. Going forward the main objective of this project will be migrating the Zenodo installation onto the LSU servers. This must be done by the LSU IT Department because they possess the proper permissions for and the knowledge of the LSU servers to alter various ports necessary to configure Zenodo. Once Zenodo is successfully migrated it will then need to be made accessible to students and faculty within LSU at <http://bagua.cct.lsu.edu/communities/>.^[19] Additionally, some scripts must be written to harvest resources from other data archives using OAI-PMH (Open Archive Initiative Protocol for Metadata Harvesting).

Section VI: Acknowledgements

Included in this section are individuals who have helped the Data Archive project progress to this point.

Professor Seungwon Yang

seungwonyang@lsu.edu

Professor Yang works as an assistant professor in both the School of Library and Information Science and the Center for Computation and Technology at Louisiana State University.^[5] He has been our primary client for the duration of the Data Archive project.

Professor Edward Alan Fox

fox@vt.edu

Professor Fox is the instructor of the Multimedia, Hypertext, and Information Access course at Virginia Polytechnic Institute and State University. He initiated the Data Archive project and helped to start communication between us and Professor Yang.

Section VII: References

1. "About Zenodo." Zenodo, CERN Data Centre & Invenio, 2017. Accessed 26 Apr. 2017. <<http://about.zenodo.org>>.
2. "CentOS Linux." About CentOS, The CentOS Project, 2017. Accessed 20 Feb. 2017. <<https://www.centos.org/about>>.
3. "Get Docker for CentOS." *Docker Documentation*, Docker Inc., 2017. Accessed 20 Feb. 2017. <<https://docs.docker.com/engine/installation/linux/centos>>.
4. "Zenodo - Research. Shared." *Zenodo 3.0.0.dev20150000 Documentation*, CERN, 2015. Accessed 20 Feb. 2017. <<http://zenodo.readthedocs.io/>>.
5. Yang, Seungwon. "Seungwon Yang." *LSU School of Library & Information Science*, Louisiana State University, 2017, Accessed 20 Feb. 2017. <http://www.lsu.edu/chse/slis/about_us/bios/yang.php>.
6. "PostgreSQL 9.6.2, 9.5.6, 9.4.11, 9.3.16 and 9.2.20 Released!" The World's Most Advanced Open Source Database, The PostgreSQL Global Development Group, 9 Feb. 2017, Accessed 20 Feb. 2017. <<https://www.postgresql.org/>>.
7. Krenz, Mark. "SSH Tutorial for Linux." Support Documentation, Suso Technology Services, Oct. 2008. Accessed 20 Feb. 2017. <https://support.suso.com/supki/SSH_Tutorial_for_Linux>.
8. "DSpace." Wikipedia, Wikimedia Foundation. Accessed 10 Mar. 2017. <<https://en.wikipedia.org/wiki/DSpace>>.
9. Good, Jeff. "A Gentle Introduction to Metadata." A Gentle Introduction to Metadata, University of California, Berkeley, 2002. Accessed 10 Mar. 2017 <<http://www.language-archives.org/documents/gentle-intro.html>>.
10. Rouse, Margaret. "What Is Dublin Core?" SearchMicroservices, Tech Target, Feb. 2006. Accessed 10 Mar. 2017. <<http://searchmicroservices.techtarget.com/definition/Dublin-Core>>.
11. Luyten, Bram. "NewDublinCore - DSpace." DuraSpace Wiki, Atlassian Confluence, 13 Dec. 2011. Accessed 10 Mar. 2017. <<https://wiki.duraspace.org/display/DSPACE/NewDublinCore>>.

12. "How to Create a CSV File." Computer Hope. Web. 10 Mar. 2017. Accessed 10 Mar. 2017.
<<http://www.computerhope.com/issues/ch001356.htm>>.
13. "Introducing Zenodo!" Zenodo, CERN Data Centre & Invenio, 2017. Accessed 26 Apr. 2017.
<<http://help.zenodo.org/features/>>.
14. "Virtual Private Network." Wikipedia, Wikimedia Foundation. Accessed 2 Apr. 2017.
<https://en.wikipedia.org/wiki/Virtual_private_network>.
15. "Apache Tomcat®." Welcome!, The Apache Software Foundation , 18 Apr. 2017. Accessed 26 Apr. 2017.
<<http://tomcat.apache.org/>>.
16. "REST API." Developers Zenodo. CERN Data Centre & Invenio, 2017, Web. 26 Apr. 2017.
<<http://developers.zenodo.org/>>.
17. Bass, Mick. "DSpace." Wikipedia. Wikimedia Foundation, 24 Apr. 2017. Web. 26 Apr. 2017.
<<https://en.wikipedia.org/wiki/DSpace>>
18. "Research. Shared." Zenodo. CERN Data Centre & Invenio, 2017, Web. 26 Apr. 2017.
<<http://about.zenodo.org/policies/>>
19. Yang, Seungwon. "Communities". Bagua CCT., 2017. Web. 25 Apr. 2017.
<<http://bagua.cct.lsu.edu/communities/>>
20. Hillmann, Diane. Using Dublin Core - The Elements. The Dublin Core Metadata Initiative, 7 Nov. 2005. Accessed 26 Apr. 2017.
<<http://dublincore.org/documents/usageguide/elements.shtml>>

Section VIII: Appendix

Appendix I. Requirements

Appendix 1.1 Application Overview

In this project we will be responsible for creating a data archive that can be used by students to upload textual and multimedia datasets. A data archive is used to provide long term storage for various forms of data. Data archiving is an important tool in the field of computer science especially for fields such as information analysis and data mining. The data archive that will be set up for this project will be using the data archiving application called Zenodo on a CentOS server provided to us by our client. For more information regarding Zenodo and CentOS, please see Section 2.1. We will work with both multimedia and text datasets throughout the project (including but not limited to social media data/metadata, CSV files, video data/metadata, and reports). Eventually, the data archive created for this project will be available at <http://bagua.cct.lsu.edu/communities/>^[19].

Appendix 1.2 Client Information

Our client for this project is Professor Seungwon Yang. Professor Yang is an assistant professor at the School of Library and Information Science, and the Center for Computation and Technology at Louisiana State University.^[5] His research is mainly in information archiving, analysis, visualization, and the use of data mining and natural language processing techniques within crisis situations and online communities.^[5] Our project should have a direct impact on his research as the data archive will be used by students in his classes to upload datasets. Specifically, our product will be used by one of Professor Yang's master's-level classes, Introduction to Digital Curation. We are required to hold biweekly meetings with Professor Yang to discuss our current progress, future goals, and any questions/concerns either party may have for the application.

Appendix 1.3 Objectives

The goal of our data archive is for it to manage data sets for project teams at LSU, including a master's-level class. The students taking the course as well as various other teams at LSU should be able to store a wide range of data types such as PDF, CSV, unstructured text, and PNG files onto the Zenodo application. We will have to test various different datasets provided by the client and create other datasets ourselves.

This is to make sure that once the application is used by students, they do not run into problems with uploading differing datasets. Once we have tested Zenodo and its capabilities on the LSU server, we will have to make a detailed administrator and general user guide. The admin guide will discuss configuration and administration of the data archive, whereas the general user guide should include information about how a user would actually use Zenodo.

Appendix 1.4 Team Roles and Responsibilities

We separated the various tasks involved with building this application across our four-person project team. The roles are as follows:

Michael Culhane is responsible for research and communication. In this role he is responsible for emailing the client and keeping him updated on the progress that is made regarding the data archive. He is also responsible for setting up meetings between the project members and the client, receiving server information from the client, and communicating any needs of the project team. He will also schedule meetings for the group to discuss the progress and future deliverables. In addition to communication, he is responsible for research into the tools used during our project. For the project, he must primarily research Zenodo as the project team members have not used this tool before. He will research the database we will need to use in addition to Zenodo (PostgreSQL) as well as information regarding the various file formats of text and multimedia datasets that will be uploaded to the data archive. This is to make sure our data archive is capable of handling the different file formats as well as archiving the data itself.

Gil Turner and John Sizemore are responsible for the data migration for our application. They will be responsible for installing the various tools we are using on the server as well as configuring Zenodo on the server. In addition, they will be responsible for formatting and uploading the datasets to be stored in the database. After that they will take feedback from the tester as well as the client in order to make changes to the application. After these changes are processed additional datasets will be uploaded to the application.

Irtiza Delwar will be responsible for documentation and testing for this project. As noted in Section 3.3, we will make an administrator guide as well as a general user guide for the application. In addition to documentation he will also be responsible for testing the application. Testing is important to make sure that users can store various types of data without any problems (i.e., loss of data). He will also notify the data migration team of any problems that occur.

Appendix 1.5 Non-Functional Requirements

Extensibility - The application should be extendable in the future to include new file formats that may be stored in the data archive.

Maintainability - This application should be maintainable after we complete the project. The administrator of the project should be able to maintain the product without too much difficulty via the admin user guide.

Simplicity / Usability - The application should be easy to use, accomplishing tasks in as few steps as possible.

Appendix 1.6 Requirements Timeline

Table 2. Timeline and Task Description

Date	Description
February 14	Develop a plan for installing Zenodo (Docker vs. development installation, pros & cons)
February 28	Install Zenodo and get it running on the server provided by our client. For this deliverable we must SSH into the LCS server. We should first install Docker and then follow the installation guidelines for the Docker installation.
March 14	Configure Zenodo for use through the LSU server, following the instruction manuals. Upload initial datasets provided by the client into Zenodo. The client shall provide multiple file formats to make sure the application is compatible. We should also test to make sure the uploads work properly and run a difference comparison on the files to make sure no data was lost.
March 28	Receive feedback from the client on the current state of the data archive. Based on this feedback we will make additional changes to the application. Create an admin user guide for the application. Make sure to include clear, concise instructions and diagrams to make it as easy as possible to follow the guide. Have the testing team verify the guide is accurate by checking that people who are not familiar with Zenodo can follow and understand it.
April 11	Meet with the client to discuss the current iteration of Zenodo on the server provided to us and how it can be improved. We will then take the feedback from our client and refine the data archive on the server to better fit our client's needs.
April 25	Review any additional feedback received from the client to make additional changes to the application. Deposit more datasets received from the client as well as create our own datasets to make sure the application can handle a wide variety of datasets. Create a general user guide for uploading data to the archive. Make sure to include clear and concise instructions as well as plenty of diagrams to make it as easy as possible to follow the guide. Have the testing team go through the guide to make sure that it is accurate and easy to understand. The testing team should also work with others to make sure that they can use it as well.

May 9	Make sure that all deliverables are completed as per the requirements. Make any final changes to the application. Create the final report for the project. Make sure to detail the experiences in creating this application as emphasized by the client.
-------	--

Appendix 1.6.1 Progress So Far

We have researched and gained basic knowledge involving Zenodo, Python, and PostgreSQL (database backend for Zenodo). We have installed DSpace onto the server at LSU as well as ANT, Maven, and Docker. We have also locally installed and configured Zenodo onto John Sizemore's machine to run various tests and to develop both the user and admin guides that our client has requested.

Appendix 1.6.2 Work In Progress

We are installing Apache Tomcat onto the LSU server as it is a requirement for DSpace to run properly. Secondly we are waiting for the LSU IT department to complete the developmental installation of Zenodo on the LSU server. We are also testing and familiarizing ourselves with the Zenodo upload process as well as their APIs so that we will be able to leverage them for the various scripts we will be writing. Lastly we have begun creating both the user and admin guides for Zenodo which our client has requested. We will produce these based on our experience with the locally installed version of Zenodo running on John's machine.

Appendix 1.6.3 Next Steps

After we finish most of the tasks we are currently working on we will then focus most of our efforts on developing the scripts that our client has requested: one for mass upload, one for mass deletion, and one for file identification-based retrieval. We will also need to refine both our user and admin guides per Professor Yang's feedback.

Appendix II: Design

There are many different requirements we are working on for this project, and for each of these requirements, we have multiple methods to consider. This makes having a well thought out design for each requirement crucial for the success of this project. The following subsections below go into more detail about some of the design decisions we have already had to make as well as decisions for the future.

Appendix 2.1 Tools Used

- Linux CentOS:
 - CentOS is a community supported Linux distribution which was created by Red Hat. The main goal of CentOS is to have and maintain a reliable, enterprise-class computing platform that is fully compatible with Red Hat Enterprise Linux (RHEL). ^[2]
- Zenodo (relies on PostgreSQL):
 - Zenodo is a research data repository that was created and is maintained/hosted by OpenAIRE and CERN (The European Organization for Nuclear Research). Zenodo was launched in 2013 allowing researchers to upload files up to 50GB in size. ^[1]
- DSpace:
 - DSpace is an open source repository software package that is usually used for creating open access repositories for scholarly digital content. DSpace's features are mainly focused on the long-term storage, access, and preservation of digital content. ^[17]
- PostgreSQL:
 - PostgreSQL is an open source, cross-platform, relational database management system that Zenodo uses as its data-storing infrastructure. PostgreSQL supports a variety of data types that should encompass the data that we are using in our project such as Arrays, Characters, Booleans, etc. For some data types, we may need to consider how to convert to similar / equivalent data types that Postgre supports. Tables in PostgreSQL use inheritance to allow for simplified partitioning of child tables. ^[6]
- SSH:
 - We are using SSH connections to connect to the server supplied by our client, which will be hosting the Zenodo client. SSH (Secure Shell) is a cryptographic network protocol for operating network services securely over an unsecured network. ^[7]

Appendix 2.2 Docker vs. Developmental Zenodo Installation

The first design decision we considered was the choice between the two types of Zenodo installations available: Docker vs. Developmental^[4]. This decision affects the rest of the project because we will be using that version of Zenodo for the entirety of the project. Some benefits with the Docker installation include easier setup, a development environment most similar to the production environment, and its directedness towards users who simply want to use Zenodo for its service. Some of the negatives regarding the Docker installation include: harder implementation, more add-on features, and the limitations of Docker containers.^[3]

For the developmental installation, further benefits are that it is easier for core developers to use, and it is easier for the user to add features to their Zenodo environment.

Some of the negatives surrounding the developmental installation include: more required installations and tools to use (e.g., NodeJS, Invenio, CleanCSS, Virtualenvwrapper) and features irrelevant to the average user.^[1]

Choosing between these approaches was our first requirement for the project; in the end, we decided on using Docker installation. We believe that it would be easier for the average user to use and requires less setup to get Zenodo running. The easier it is for the future users to use and manage our application, the better, because a wide range of people at LSU will be using Zenodo. Lastly, we would like to maximize the time we can spend focusing on migrating data and actually learning how to use Zenodo. Table 3 summarizes the comparison between Docker and Developmental Zenodo installations.

Table 3. Docker and Developmental Installation Comparison

Docker	Developmental
Fewer installations required. Docker only extra installation but has helpful installation tutorial.	Extra installations needed: Docker, Virtualenvwrapper, Invenio, NodeJS, SASS, CleanCSS, UglifyJS and RequireJS.
Meant more for an average users who just wants to use Zenodo service.	Mainly used for core developers which is meant for people actually committing and adding code to the Zenodo GitHub.
Docker provides the most similar environment to the production environment.	Can set up the local instance for easy code development.
<p><u>Both:</u> PostgreSQL (Red Hat version), Elasticsearch (basic Linux version), Redis (Stable version 3.2.8), RabbitMQ (CentOS version), and Python 2.7.</p>	

Appendix 2.3 User Guide Documentation Design

One of the main requirements for this project is to create both a user and an admin guide. These documents will contain detailed notes about how to use Zenodo from the perspectives of a general user and an administrator, respectively. Once we install and configure the Zenodo Docker installation, we will write step-by-step instructions on how we did it as well as take screenshots to show how we set up Zenodo. The admin and installation guides are due March 28; however, we will begin working on this guide the week of February 20, because during that week, we will create our administrative user. The general user guide is due April 25.

Appendix 2.4 Data Flow Design

Since our project relies heavily on the flow of data between the users at LSU and the Zenodo database, it is important for us to understand exactly how data will be flowing through our project. The main goals of Zenodo are to get data from the client side into the Zenodo database and to get data from that database back to the users. First, the user must structure the data into a Zenodo-accepting format. A list of some data types Zenodo accepts includes: PNG, PDF, and CSV. A full list can be found on the Zenodo website^[18]. Once the users prepare acceptable files, they will upload them to Zenodo via our deposit script (possibly linked to an upload button on the website Professor Yang

plans to make for the data archive website <http://bagua.cct.lsu.edu/communities/>). Zenodo will use PostgreSQL to store this data in the server.

If users wish to retrieve data from Zenodo, they will first send search requests to Zenodo with various queries and filters (such as date and author). Zenodo will use PostgreSQL to gather a relevant response which will be returned to Zenodo, which will then send it to the client.

Figure 42 shows our initial data flow design:

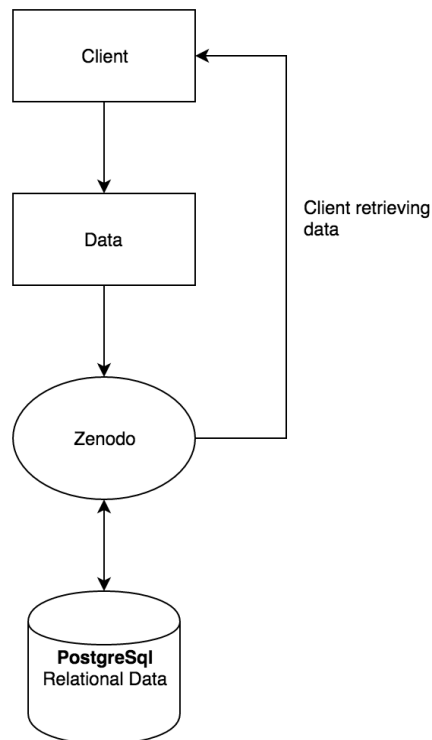


Figure 42. Data Flow Design

Appendix 2.5 Testing Approach

We will begin testing while depositing the initial datasets into Zenodo. Professor Yang will be providing us with various datasets to test different file formats. In addition to this, we will also be responsible for creating mock datasets to rigorously test our application.

The testing team will provide detailed feedback about the application to the data migration team. In addition to testing the functionality of uploading datasets to the archive, we will also test other functionalities of Zenodo such as: upload, search, edit, and delete. We will accomplish this by writing testing scripts that perform these operations on the mock datasets to automatically verify (via Zenodo API calls) whether

the upload, search, edit, and delete functions are working. We will also perform manual functional verification tests using the Zenodo UI to check the same functionalities.

After receiving feedback from the client, the data migration team will make any fixes necessary. During this time, the testing team will create more mock datasets to ensure the application is functioning correctly.

All team members will be responsible for proofreading the admin and user guides to make sure that they are correct. After that step is completed, we will then ask the client for additional feedback including feedback from students, who will be the real users for this application. Using this information, we will be able to make any additional changes necessary to complete our application. We want to focus on testing the robustness of our application as it is important for our application to be able to handle various file formats. We also want to focus on the usability of our product, by making the process of uploading datasets as simple as possible for the user. This should extend to the user guide which users should follow to upload datasets.

Appendix III: Implementation

During the implementation period, Michael was responsible for communicating any errors we encountered with our client, John and Gil were responsible for installing the software, and Irtiza was responsible for documenting errors that occurred.

Appendix 3.1 Acquiring Server Credentials

Initially, our client provided us with credentials for the server that will be hosting the application. Using this information we were able to SSH onto the servers remotely from our local machines.

Michael was responsible for receiving the necessary information to access the servers from the client while also connecting to the server to ensure we had adequate permissions (not including root access). During this time the rest of the team was reviewing the installation instructions for Docker as well as Zenodo.

Appendix 3.2 Docker and Zenodo Installation

Gil and John began the installation process, which started as planned, but after the first few steps we encountered various problems. The first issue was our lack of permissions on the server we were given access to. A lot of the installation commands for Docker and Zenodo required root access, which we did not have. Michael contacted Dr. Yang to deal with this issue. Our solution was to work with Dr. Yang over Skype since he did have root access on the server. We used Skype screen share to go through the Docker installation and setup together in order for Dr. Yang to perform the necessary sudo commands.

After successfully installing Docker, we moved on to the Zenodo installation. During the installation process, we encountered port issues with the Docker portion of the Zenodo installation. More specifically, Docker needed to use port 80, which was already in use by an Apache application. Dr. Yang consulted with the IT staff at LSU and was informed that we were not allowed to change the port number that the Apache application was using. After learning this, we decided to try alternative solutions ourselves. Our group had a four hour Skype session just before Spring Break with Professor Yang in which we tried to get the Zenodo installation to work using an alternative port number for

Docker. We were unable to get Zenodo installed with Docker during that time frame.

As a group, we decided to post our issues onto Zenodo's github forum to see if someone would be able to provide assistance while Professor Yang worked with the IT staff for the LSU server to see if they could get Zenodo installed. During the next few days, our group received responses on Zenodo's github forum, none of which resolved our issue. On March 9, a few days after our Skype session, Professor Yang was able to meet with an IT staff member to work through the Zenodo installation. After their five hour session, the IT staff member concluded that the software was fairly complex and difficult to set up due to multiple server components and their prerequisite components not being compatible with the Zenodo installation. The IT staff member told Professor Yang that they couldn't guarantee that they could get Zenodo running on our server but they would try again in a few weeks. Since the preconditions we were told about for the software and applications on the server were incorrect, and solving the issues we ran into were out of our control, we decided to use a new platform called DSpace in place of Zenodo. In addition to pivoting to DSpace, we also decided to install Zenodo locally and we will work on the local DSpace installation on the side.

Appendix 3.3 DSpace Installation

Before starting the DSpace installation, we acquired root access to the server to avoid permission errors similar to those which we encountered during the Zenodo installation. The following week during spring break, we spent time doing individual research on DSpace and the various applications that would be needed for the installation. When spring break ended, we began the installation of DSpace, however we ran into an issue with DSpace installation regarding the PostgreSQL that it uses as its database. We had a meeting on March 16 with Professor Yang regarding this issue and came to the conclusion that PostgreSQL was installed by the IT staff at LSU via Docker. As a result, the DSpace installation is having trouble locating PostgreSQL inside its Docker container. After doing some research, we were able to find a solution for this: when performing PostgreSQL commands on the command line, we must include two additional parameters. One of the parameters is the port number that PostgreSQL uses and the second is localhost. After solving this issue, however, we ran into an issue with connecting DSpace to a servlet engine. We attempted to use Jetty as our servlet engine because it was already on the server, but we were having issues getting that to work. After talking to the LSU IT staff, they recommended we download and install Tomcat for the servlet engine. We plan to perform this installation and will verify that Tomcat can work as the DSpace servlet engine.

Appendix 3.4 Implementation Timeline

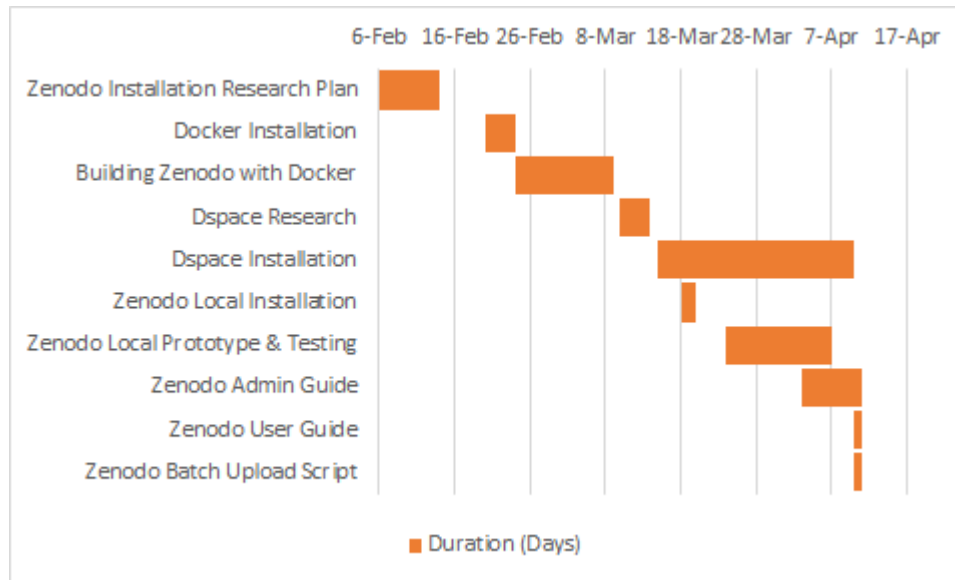


Figure 43. Gantt chart representing our current timeline progress.

Appendix 3.5 Metadata Specifications

Our client also asked us to look ahead in our project and to begin formatting the metadata for the datasets given to us by Professor Yang. The provided datasets include multimedia files, CSV files, and unstructured Text Documents. CSV, or “Comma Separated Values”, is a simple file format used to store tabular data sets such as spreadsheets and database entries. Unstructured Text Documents refer to information that does not have a pre-defined model. Unstructured data as a whole mainly has large blocks of text with a focus on numbers and dates.^[12]

Appendix 3.6 DSpace: Additional Information

According to Wikipedia, “DSpace is an open source repository software package typically used for creating open access repositories for scholarly and/or published digital content.”^[8]

This platform allows for repositories of digital content. The main focus of DSpace and why people use it over other digital libraries is its focus as a digital archive system. Since DSpace is designed for long term storage, storage, preservation, and efficient

access are important factors.^[8]

Digital libraries include a focused collection of digital objects, for example: text, video material, audio material, graphics, etc. Digital Libraries can fall along a spectrum when it comes to size and variety of content matter. The information spanning across the library can be stored on one server or hosted across a network.^[8]

It is common for data archive repositories such as this to be created as “Academic Repositories” within a digital library for storage and retrieval. Datasets stored within a digital library for academic projects are frequently opened to the public.^[8]

Some of the reasons why digital libraries are so significant include: easy and efficient access to data, scale of how much data/content can be stored in such a small space (relative to a physical library), multiple people accessing the same content simultaneously, and preservation of the content (i.e., no wear and tear like physical books/pamphlets/etc).^[8]

DSpace is built upon various programs including Java web applications developed to maintain an asset store which will go into the file system, and a metadata store. The metadata encompasses information supporting the access of specific data from the datasets and how everything is configured. The metadata is also stored in the database that DSpace uses which can be PostgreSQL or an Oracle Database.^[8] Since we already planned to use PostgreSQL with Zenodo, we will use it with DSpace as well.

Actions that we and the future users will be doing as developers on DSpace include:

- Data deposits
 - This entails loading the relevant datasets into the right spot in the archive.
- Searching/querying datasets to serve research, analysis, and applications
- Migrating/copying/backing up data

Appendix 3.7 Metadata: Additional Information

Metadata is essentially data that provides information about other data. A metadata catalog can be thought of as a “summary of the contents of a library or archive, like a card catalog.”^[9]

Metadata is especially important in locating resources^[9], so it is imperative that we include it in our project to assist with the access of data that future users of this data archive will use. When creating metadata, it is important to take into account what references will be listed for the datasets to facilitate the process of resource locating.

After discussing the subject with our client, we determined that the Dublin Core model is an appropriate format for this project and will thus create .xml files in the Dublin Core style for the datasets he gives us.

“Simple Dublin Core expresses elements as attribute-value pairs using just the 15 metadata elements from the Dublin Core Metadata Element Set.”^[10]

We will most likely stick to the Simple Dublin Core format to keep the project as simple as possible while still providing all necessary references. If we notice later in the project that we need the Qualified Dublin Core format to provide desired functionality, then we will change that later.

These 15 elements and some characteristics are listed as^[11]:

- Contributor - An entity responsible for making contributions to the content of the resource.^[20]
 - Range: Agent Class
- Coverage - The extent or scope of the content of the resource.^[20]
 - Range: LocationPeriodOrJurisdiction (recommended controlled vocabulary like Thesaurus of Geographic Names (TGN))
- Creator - An entity primarily responsible for making the content of the resource.^[20]
 - Range: Agent Class
- Date - A date associated with an event in the life cycle of the resource.^[20]
 - Range: Literal (recommended W3CDTF profile of ISO 8601)
- Description - An account of the content of the resource.^[20]
 - Range: none
- Format - The physical or digital manifestation of the resource.^[20]
 - Range: MediaTypeOrExtent (Recommended list of internet media types (MIME)).
- Identifier - An unambiguous reference to the resource within a given context.^[20]
 - Range: Literal
- Language - A language of the intellectual content of the resource.^[20]
 - Range: LinguisticSystem (Recommended RFC 4646 IETF standard)
- Publisher - The entity responsible for making the resource available.^[20]
 - Range: Agent
- Relation - A reference to a related resource.^[20]
 - Range: none (YET)
 - Note: This term is intended to be used with NON-LITERAL values. As of December 2007, the DCMI Usage board is seeking a way to express this intention with a formal range declaration.
- Rights - Information about rights held in and over the resource.^[20]
 - Range: RightsStatement
- Source - A Reference to a resource from which the present resource is derived.^[20]
 - Range: none (YET)
 - Note: This term is intended to be used with NON-LITERAL values. As of December 2007, the DCMI Usage board is seeking a way to express this intention with a formal

range declaration.

- Subject - The topic of the content of the resource.^[20]
 - Range: none (YET) For spatial or temporal topic of the resource, use the Coverage element instead.
 - Note: This term is intended to be used with NON-LITERAL values. As of December 2007, the DCMI Usage board is seeking a way to express this intention with a formal range declaration.
- Title - The name given to the resource.^[20]
 - Range: Literal
- Type - The nature or genre of the content of the resource.^[20]
 - Range: Any Class? (Recommended usage is the DCMI Type Vocabulary (DCMIType). For file format, physical medium or dimensions: use format!

Appendix IV: Prototype

Appendix 4.1 Local Installation

Our current prototype is a version of Zenodo installed locally on John Sizemore's machine. John was in charge of locally installing Zenodo on his machine and followed the same installation process for Zenodo that was done on the LSU server. The issues that were encountered when dealing with LSU's servers were nonexistent when installing Zenodo locally as there was no pre-installed software on John's computer using the ports necessary to run Docker and Zenodo. After the installation process, we were able to locally access the Zenodo Graphical User Interface. The home page for Zenodo is shown in Figure 44.

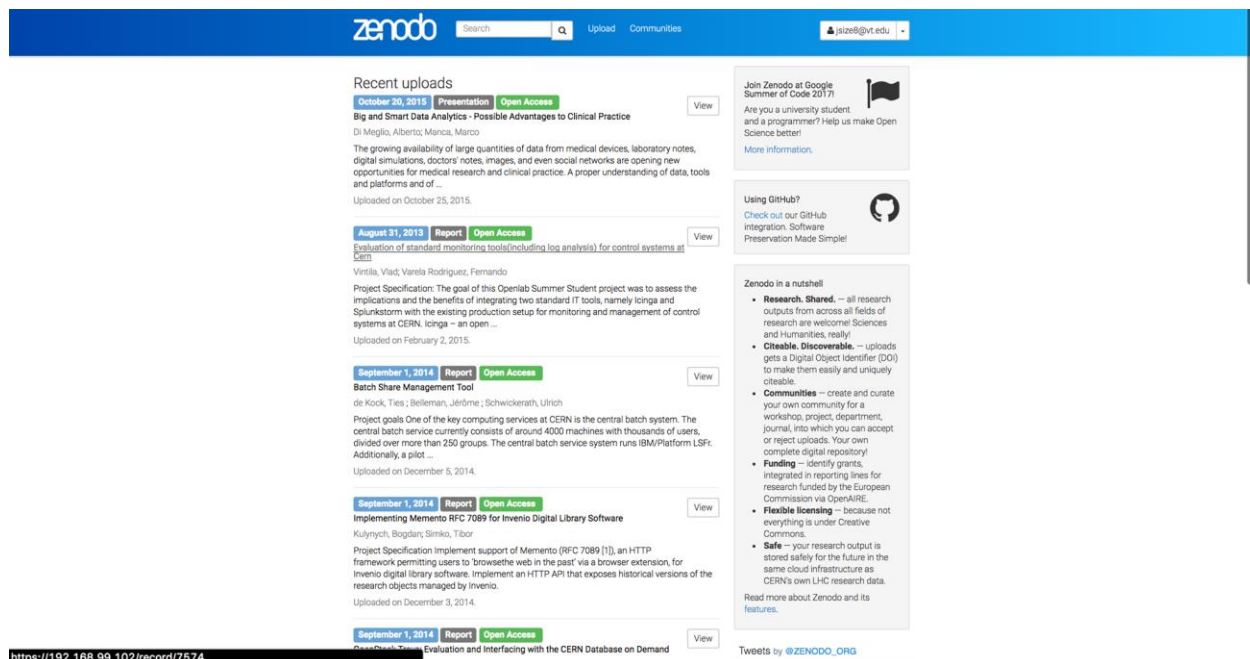


Figure 44. Zenodo Homepage

Appendix 4.2 Upload

After testing Zenodo's homepage, we began to examine its upload functionality. Zenodo's header contains an upload button which begins the upload process when clicked. Figure 45 shows the graphical user interface for file uploads that the user is redirected to in order to fill out the required information.

Once a file has been uploaded, the type of data that is being published can be specified. The built in data types include publication, poster, presentation, dataset, image, video/audio,

software, and lesson. We have been and will continue to test using the various datasets and images that were supplied by our client.

The screenshot shows the Zenodo upload interface. At the top, there are buttons for 'Delete', 'Save', and 'Publish'. Below this is the 'New upload' section. A note provides instructions: '(i) Upload minimum one file or fill-in required fields (marked with a red star). (ii) Press "Save" to save your upload for editing later. (iii) When ready, press "Publish" to finalize and make your upload public.' The main area is a large box with the text 'Drag and drop files here' and a 'Choose files' button. Below this, it says '(minimum 1 file required, max 50 GB per dataset - contact us for larger datasets)'. The 'Upload type' section is a horizontal bar with icons for 'Publication', 'Poster', 'Presentation', 'Dataset', 'Image', 'Video/Audio', 'Software', and 'Lesson'. The 'Publication type' dropdown menu is set to 'Journal article'.

Figure 45. Zenodo Upload

The next step in the upload process is to supply basic information on the file upload. Figure 46 shows Zenodo's graphical user interface for inputting the basic information about the file being uploaded. Some of the required basic information include upload type, publication date, title, author, and description. Optionally, a digital object identifier can be supplied, if your publisher has already assigned a DOI to your file, to help others easily and unambiguously cite uploads. If the user does not specify a DOI, then Zenodo will register a DOI for you. Keywords may also optionally be added to increase the searchability of uploads as well as additional notes to further document uploads.

Basic information required ▼

Digital Object Identifier 🔗
 Optional. Did your publisher already assign a DOI to your upload? If not, leave the field empty and we will register a new DOI for you. A DOI allows others to easily and unambiguously cite your upload. Please note that it is NOT possible to edit a Zenodo DOI once it has been registered by us, while it is always possible to edit a custom DOI.

Publication date *
 Required. Format: YYYY-MM-DD. In case your upload was already published elsewhere, please use the date of first publication.

Title *
 Required.

Authors * 📏 ✕
 + Add another author

Description *
 Required.

Keywords 📏 ✕
 + Add another keyword

Additional notes
 Optional.

Figure 46. Zenodo Upload Basic Information

The license section is the next required input for uploading a file after basic information. While Zenodo does encourage all data to be shared, it also allows for varying levels of visibility, including open, embargo, restricted, and closed access.^[13] Open access requires a license name, embargoed access requires a license name and an embargo date, restricted access requires conditions on which to grant other users access to the published data, and closed access doesn't require any additional information. Figure 47 shows the graphical user interface for the licensing information discussed above.

License
required ▼

Access right *

- Open Access
- Embargoed Access
- Restricted Access
- Closed Access

Required. Open access uploads have considerably higher visibility on Zenodo.

License *

Start typing a license name...

Required. The selected license applies to all of your files displayed in the top of the form. If you want to upload some files under a different license, please do so in two separate uploads. If you think a license is missing from the list, please inform us at info@zenodo.org

Figure 47. Zenodo Upload Licensing

In addition to all of the required information discussed above, there are other recommended sections that may be filled out. These sections include communities, funding, and alternate identifiers. Communities allow groups to have files uploaded and grouped together which is one of the features our client was most interested in as it allows student groups to create their own communities. Groups can then upload all of their data together in a well organized manner. Communities allow for groups to have their own digital repository.^[13]

Communities
recommended ▼

Any user can create a community collection on Zenodo ([browse communities](#)). Specify communities which you wish your upload to appear in. The owner of the community will be notified, and can either accept or reject your request.

Communities

Start typing a community name... ×

+ Add another community

Funding
recommended ▼

Zenodo is integrated into reporting lines for research funded by the European Commission via OpenAIRE (<http://www.openaire.eu>). Specify grants which have funded your research, and we will let your funding agency know!

Grants

Start typing a grant number, name or abbreviation... ×

Optional. European Commission FP7 and Horizon 2020 grants only. For general funding acknowledgements, please use the **Additional Notes** field.
Note: a human Zenodo curator will need to validate your upload - you may experience a delay before it is available in OpenAIRE.

+ Add another grant

Related/alternate identifiers
recommended ▼

Specify identifiers of related publications and datasets. Supported identifiers include: DOI, Handle, ARK, PURL, ISSN, ISBN, PubMed ID, PubMed Central ID, ADS Bibliographic Code, arXiv, Life Science Identifiers (LSID), EAN-13, ISTC, URNs and URLs.

Related identifiers

e.g. 10.1234/foobar.567890

▼

×

+ Add another related identifier

Figure 48. Zenodo Upload Funding/Communities

The final part of the upload process includes various optional information that the user can input. Figure 49 shows the types of optional information that the user can add to the upload.

Contributors	optional >
References	optional >
Journal	optional >
Conference	optional >
Book/Report/Chapter	optional >
Thesis	optional >
Subjects	optional >
<div style="display: flex; justify-content: space-between; align-items: center;"> Delete Save Publish </div>	

Figure 49. Zenodo Upload Optional Information

Appendix 4.3 Search

The screenshot shows the Zenodo search interface. At the top, there is a search bar with the text 'Test Data' and a search icon. To the right of the search bar are links for 'Upload' and 'Communities', and a user profile dropdown for 'jsize8@vt.edu'. Below the search bar, the results are displayed in a list format. On the left side, there are filters for 'Access Right', 'File Type', 'Keywords', and 'Type'. The main content area shows three search results, each with a date, a title, a description, and a 'View' button. The first result is 'Test Data' by John Sizemore, uploaded on April 6, 2017. The second result is 'RAPIDIO USAGE IN A BIG DATA ENVIRONMENT' by Costa, Jorge, Barrington, Olof, uploaded on September 1, 2015. The third result is 'Improved metrics collection and correlation for the CERN cloud storage test framework' by Lindqvist, Carolina, Zotes, Maitane, Heikkila, Seppo, uploaded on September 1, 2013. The fourth result is 'Evaluation of in-memory database TimesTen' by Andras Simon, Endre, Potocky, Miroslav, uploaded on August 31, 2013.

Figure 50. Zenodo Searching for Files

Another important feature in relation to our client's interests is the search functionality. The user

can search for a specific set of data within the data archive, e.g., for an archived file by searching for any specific metadata that was entered during the upload of that file. Figure 50 shows a search done for the keyword “test data”. Within the results obtained from this search are the files that contain either the word “test” or “data,” sorted by estimated relevance.

Appendix 4.4 Metadata: Dublin Core

April 6, 2017

Dataset Closed Access

Test Data

John Sizemore

Dublin Core Export

```
<?xml version='1.0' encoding='utf-8'?>
<oai_dc:dc xmlns:dc="http://purl.org/dc/elements/1.1/" xmlns:oai_dc="http://www.openarchives.org/OAI/2.0/oai_dc/" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://www.openarchives.org/OAI/2.0/oai_dc/ http://www.openarchives.org/OAI/2.0/oai_dc.xsd">
  <dc:creator>John Sizemore</dc:creator>
  <dc:date>2017-04-06</dc:date>
  <dc:description>&lt;p&gt;This is test data.&lt;/p&gt;</dc:description>
  <dc:identifier>10.1234/foo.bar</dc:identifier>
  <dc:identifier>https://zenodo.org/record/45450</dc:identifier>
  <dc:identifier>oai:zenodo.org:45450</dc:identifier>
  <dc:rights>info:eu-repo/semantics/closedAccess</dc:rights>
  <dc:subject>test</dc:subject>
  <dc:subject>data</dc:subject>
  <dc:title>Test Data</dc:title>
  <dc:type>info:eu-repo/semantics/other</dc:type>
</oai_dc:dc>
```

Figure 51. Zenodo Dublin Core (Metadata) Export

We discussed what metadata and Dublin Core are in Section 5.7. Zenodo has its own interface for providing metadata for a dataset during deposit, whether it is done manually through the GUI or through the API. Once a dataset has been deposited, Zenodo provides a tool to export its corresponding metadata in various formats, including Dublin Core. This is shown in Figure 51, where we obtained a Dublin Core export for one of our test datasets. This export contained tags for fields such as creator, date, description, and subject, all of which were provided in .xml format.

Appendix V: Testing

We are currently testing through the use of manual uploads in Zenodo. We do this by uploading a dataset and providing Zenodo with all the relevant metadata to each set. Then, we use Zenodo's built-in ElasticSearch to retrieve the proper dataset that a user might be using to find it. For example, for a dataset containing data on Hurricane Sandy tweets collected by LSU, a user could find the dataset in Zenodo by searching for various labels such as "Twitter Data", "tweets", "Hurricanes", "LSU", and any other keywords that were supplied when the dataset was uploaded. This is possible because the uploader or admin of this repository (in this case it is us for the time being) specifies what metadata to include with each upload and a section of that is "keywords" tags.

Once we have created automated scripts, testing will be done on them to ensure they work as intended. Ideally, a project like this will have automated functionality testing for the batch deposit, however, we may not have time to set up a testing environment, write test scripts, and undertake other aspects of automating the testing process. So we will start by performing manual functionality verification testing. We will essentially be running the same manual testing process described above, except for all the individual datasets in our deposit script.

There are also ways to ensure that the Zenodo REST API is properly being called in the automated scripts. Individual components of the scripts can be tested with unit tests as Zenodo provides a sandbox environment which can be used to test the API calls. The sandbox environment will be helpful for testing because it can be cleared and reset at any time. This will help speed up the development process for the automated upload script since there will be no need to manually delete the datasets each time the script is ran.

Various test cases that we have chosen to perform on the datasets include:

Appendix 5.1 Searching for a Dataset by Title

The screenshot shows the Zenodo search interface. At the top, the Zenodo logo is on the left, a search bar contains 'Prototype Dataset', and navigation links for 'Upload' and 'Communities' are in the center. On the right, the user 'jsize8@vt.edu' is logged in. Below the search bar, there are filters for 'Access Right' (Open (5), Closed (2)) and 'File Type' (Pdf (5)). The search results show 'Found 7 results.' with a pagination control showing '1'. The first result is 'Prototype Dataset' by Michael Culhane, uploaded on April 6, 2017. The description reads: 'This is a dataset for the prototype of our Capstone Project'. The result is marked as a 'Dataset' and has 'Closed Access'. A 'View' button is visible next to the result.

Figure 52. Zenodo searching by title

It is important that users can find datasets by title, as in Figure 52. For example, some of the data will be tweets collected that have to do with Hurricane Sandy. So if the dataset is titled “Hurricane Sandy Tweets” then any user doing research on Twitter, hurricanes, or on this specific hurricane can easily be matched with this dataset.

Appendix 5.2 Searching for a Dataset by Author

The screenshot shows the Zenodo search interface. At the top, the Zenodo logo is on the left, followed by a search bar containing "Michael Culhane" and a search icon. To the right of the search bar are links for "Upload" and "Communities", and a user profile dropdown for "jsize@vt.edu". Below the search bar, the page displays "Found 1 result." with a pagination control showing page 1 of 1. On the left side, there are two filter panels: "Access Right" with a checkbox for "Closed (1)" and "Keywords" with checkboxes for "Capstone (1)", "Cs4624 (1)", "Prototype (1)", and "Vt (1)". On the right side, there are sorting options: "Sort by:" with a dropdown menu set to "Best match" and a "View" button. The search results section shows a single result: "Prototype Dataset" by Michael Culhane, with a date tag "April 6, 2017", a "Dataset" tag, and a "Closed Access" tag. The description reads: "This is a dataset for the prototype of our Capstone Project" and "Uploaded on April 6, 2017". A second pagination control is located at the bottom of the results section, also showing page 1 of 1.

Figure 53. Zenodo Searching by Author

Similarly, we will also be testing to ensure the search engine Zenodo is running will retrieve based on queries for the author of a dataset, as in Figure 53.

Appendix 5.3 Searching for a Dataset by Publication Date

The screenshot shows the Zenodo search interface. The search bar contains 'April 6, 2017'. The results show 92 items. The first result is 'Upgrading the Huawei Cloud Storage Benchmark Framework for ROOT6 Compatibility' by Seetharaman, Surya; Arsuaga Rios, Maria; Heikkila, Seppo S., uploaded on October 22, 2015. The second result is 'Prototype Dataset' by Michael Culhane, uploaded on April 6, 2017. The interface includes filters for Access Right (Open, Closed, Embargoed, Restricted), File Type (Pdf, Docx, Pptx, Doc), and Keywords.

Figure 54. Zenodo Searching by Publication Date

Publication Date is also encompassed by the metadata that we include during deposit, therefore Zenodo search retrieves queries for the date. We are also ensuring uploads can be retrieved this way, as in Figure 54.

Appendix 5.4 Searching for a Dataset by Relevant Terms (i.e. Tags)

The screenshot shows the Zenodo search interface. The search bar contains 'lotype capstone cs4624'. The results show 5 items. The first result is 'Prototype Dataset' by Michael Culhane, uploaded on April 6, 2017. The interface includes filters for Access Right (Open, Closed) and File Type (Pdf).

Figure 55. Zenodo searching by relevant terms

The terms we include in our metadata during deposit are an important set of information for our datasets to be retrieved by. Testing this search after the deposit, as in Figure 55, ensures that they were included properly in the upload.

Appendix 5.5 Searching for an .mp3 File

The screenshot shows the Zenodo search interface. The search bar contains 'Dead Combo'. The user is logged in as 'jsize8@vt.edu'. The search results show 2 results. The first result is 'Prototype Audio File' by John Sizemore, uploaded on April 6, 2017. The file is categorized as 'Video/Audio' and has 'Closed Access'. The description states: 'Audio file we are using for the prototype testing. Uploaded on April 6, 2017'. The interface includes filters for 'Access Right' (Closed (1), Open (1)) and 'File Type' (Docx (1)).

Figure 56. Zenodo Searching for Audio File (.mp3)

.csv files for datasets are not the only type of file that will be uploaded to this repository, so we are testing other multimedia file formats to ensure they all can be uploaded and retrieved. See Figures 56 and 57.

Appendix 5.6 Searching for an .mp4 File

The screenshot shows the Zenodo search interface. The search bar contains 'Prototype Video'. The user is logged in as 'jsize8@vt.edu'. The search results show 7 results. The first result is 'Prototype Video' by Irtiza Delwar, uploaded on April 6, 2017. The file is categorized as 'Journal article' and has 'Closed Access'. The description states: 'Video we are uploading for testing our prototype. Uploaded on April 6, 2017'. The interface includes filters for 'Access Right' (Open (4), Closed (3)) and 'File Type' (Pdf (4)).

Figure 57. Zenodo Searching for Video File (.mp4)

Regarding video resources, .mp4 is another type of multimedia file format that we are testing.

Appendix VI: Refinement

There is still a fair amount of work to complete for our client. We would like to work with Professor Yang and the LSU IT department to get Zenodo installed on the LSU server. Furthermore, we would like to update the user and admin guides to reflect the process of working with Zenodo as hosted on the LSU server. For the DSpace application, the main goal is to get it successfully installed on the server and get an operational URL hosted at LSU and protected from within the firewall.

Appendix 6.1 DSpace Installation

The PostgreSQL database that DSpace will use has already been correctly installed and configured. Because the Jetty servlet engine was not successfully running DSpace, the IT staff at LSU suggested we install Tomcat onto the server. Tomcat is an open-source java servlet container developed by the Apache Software Foundation.^[15] Once we have Tomcat installed on the server, we will have to edit the DSpace configuration files so that it uses the Tomcat web applications instead of Jetty. Once DSpace is rebuilt with the updated configuration files, we will get a new URL to access DSpace on our browser. An example URL that would be used would be `http://bagua.cct.lsu.edu:9090/xmlui`. One problem with accessing the service is that its URL would be behind an LSU firewall. This means in order to see if the rebuild is successful, we either need to contact Professor Yang so that he can test the URL, or get a VPN and test the URL ourselves. VPNs or Virtual Private Networks are used to add security and privacy to networks such as the internet.^[14] We will have to contact Professor Yang and the LSU IT staff as there may be sensitive information on the server.

Since there were a few issues with the DSpace installation, we may not have time to make the user and admin guides for DSpace. As stated before, our client wants us to focus on getting DSpace installed with the guides for DSpace being a secondary goal if time permits.

Appendix 6.2 Zenodo Prototype

Our other main goal is to refine our local Zenodo prototype. The current iteration of our Zenodo prototype currently involves us manually uploading data sets to the Zenodo repository. This has worked well in establishing how the process of retrieving a dataset in Zenodo works, however we would like to extend this by writing a Python script to upload all of the datasets relevant to this application as a batch process. We have created a Zenodo account and have received an API token in order to use Zenodo's RESTful API. We are currently working on using the Zenodo API to run the commands that will perform this batch deposit.^[16]

The programming language we plan to use for the Zenodo API is Python. One reason for this decision is that on the Zenodo API page, the examples they provide are in Python.^[16] Another

reason is that we are all experienced with Python's requests library that is used to perform the API calls.

We plan to create a script that will perform a mass deletion from the repository for all the datasets under this project in a separate script in case the LSU team would ever like to migrate away from Zenodo in the future or switch out its datasets. Another script we plan to create is single Zenodo file retrieval. The user will be able to enter in a file ID and the script will return the metadata information for that file. If time permits, we will also modify that retrieval script so that the user can enter in multiple file IDs or a text file with file IDs and the metadata for all the files would be returned in a csv file.

There are some special cases regarding data that Professor Yang gave us to deposit into Zenodo. The first case is for video files with associated .csv datafiles. When users upload videos, their associated .csv datafile should be stored on the same page as the video. An example of this case is a video that shows human hand movement while typing which is paired with a .csv file containing accelerometer data about those movements. Another special case we were given is that some data files are related to each other and those files should be uploaded and stored on a single page in Zenodo.

```
>>> data = {
...     'metadata': {
...         'title': 'My first upload',
...         'upload_type': 'poster',
...         'description': 'This is my first upload',
...         'creators': [{ 'name': 'Doe, John',
...                         'affiliation': 'Zenodo' }]
...     }
... }
>>> r = requests.put('https://zenodo.org/api/deposit/depositions/%
...                  params={'access_token': ACCESS_TOKEN}, data=j
...                  headers=headers)
>>> r.status_code
200
```

Figure 58. Zenodo RESTful API for Uploading Metadata. This sample code has been taken from Zenodo's Developers website which shows how to upload metadata information for deposition files onto Zenodo using their API.^[16]

```
response = requests.get('/api/deposit/depositions',  
                        params={'q': 'my title',  
                              'access_token': ACCESS_TOKEN})  
  
print(response.json())
```

Figure 59. Zenodo RESTful API for Deposition File Retrieval. Sample API call request from Zenodo API that returns files based on the title “my title”.^[16]

We plan to create scripts using some of the API calls shown in Figures 58 and 59 and test these scripts using the datasets that Professor Yang has provided to us. We will also aim to make the deposit script as generic as possible to simplify the insertion process for the future admins of this project.

Appendix 6.3 User & Admin Manual

In addition to the tasks above we are also required to write two manuals for our client, the first being the admin manual. The admin manual will focus on how to install and manage the Zenodo data archive. The admin guide should include instructions for configuration, bulk upload, and harvesting resources from other data archives using OAI-PMH^[18] and other basic management tasks. Our client has also tasked us with keeping a record of difficulties and solutions that we experience while installing the data archive. This information will also be added to the admin manual. Information about the Zenodo Restful API will also be added to the admin manual.

A user manual will also be created. The intended audience for this manual will be researchers, students, and faculty. Users will be shown how to search for and deposit resources including published papers and datasets in various formats such as .csv, multimedia, and text documents.

It is very important that the information in the user and admin manual be clear and concise to minimize the issues that the users and administrators experience. Many pictures and figures will also be added to the manuals to clearly identify the correct steps in the various processes in both manuals.