System/Subsystem Design Description

Casey Blair, Northwest Knowledge Network

June 30, 2017

1 SCOPE

1.1 Identification

This SSDD applies to the Metadata Editor developed for the Northwest Knowledge Network. The Metadata Editor is also know as mdedit.

System Overview The Metadata Editor was developed to allow authenticated users to input metadata about a data file in to a database, and upload that metadata along with the file to a server. The data will be converted in to ISO 19115 or Dublin Core XML format, whichever the user specifies while creating the metadata. That metadata and accompanying file is then inserted in to a web based search system hosted at https://northwestknowledge.net/data-search. The search results are displayed in a web page: the metadata generated when a record is published is used to form the search results, and displayed when a user queries the search system. And the user can download the file the metadata describes by clicking on a download link. They can also download the accompanying metadata XML files.

1.2 Document Overview

This document comprises of $_{sections:}$

This section

Another section

2 REFERENCED DOCUMENTS

2.1 Project Documents

3 SYSTEM-WIDE DESIGN DECISIONS

3.1 Form Wizard

The system uses a web site that the user interfaces with. This website is accessable to any device that has internet connectivity, and is designed to work both on mobile devices and laptop/desktops. This website uses a "form wizard" design that breaks the entire form down in to different sections.

The "form wizard" design was decided upon because of the number of form inputs in the system were overwhelming to have all on one page. An earlier design did have all the form inputs on one page, and while they were broken up in to different sections on the same page, it was not very organized or asthetically pleasing. Plus, there was not way to force the user to save their progress on the form while they were filling out the form: there was just a "Save" button in the top right that the user had to know to press to save their inputted data to the database. However, with the "form wizard" format, we could break down the sections to appear one at a time on the page, and every time the form section was changed, the system would automatically save the data to the database.

3.2 Progress Bar (Breadcrumb Nav Bar)

The majority of the form input elements are required, but not all of them. To help the user know which element are required, we developed a custom progress bar that alerts the user if they have filled out all the required form inputs in that section or not. This system alerts the user which form section they still have to finish instead of waiting until the user tries to submit the form and having an alert that there is a form element missing.

This progress bar shows a checkmark and turns green if the user has completed all required form inputs in that section. The progress bar turns red and shows an "X" if the user has not finished all form inputs

in that section. And shows a circle and turns dark green if the section has been selected by the user. This system gives the user visual queues to what parts of the form still need to be finished before the system will allow the user to submit the record for review by a data manager for publication.

The user can review all the data on a "Review" page before they submit the metadata for publishing. And the user can only submit the metadata record if they have filled out all the required form inputs.

3.3 Admin Panel

An admin panel was developed to replace the search system that a Geoportal system was providing before. This admin panel allows the admin to see a list of all records in the system in their three states in the publishing process:

1. not submitted yet
2. submitted for review
3. published records

The admin can delete records in the first two states, but not the "published" state. Published records should not be changed and should exist in the system unaltered after they are published. If a published record needs to be updated, a new record must be published with the updated information, and the published record must be "unpublished" which is not currently possible in the current sytem, but will be added in the future.

The admin can also assign DOI/ARK values for records in the "submitted for review" state. The system will, in the future, automatically assign DOI's and allow the user to enter in their own ARK values.

4 SYSTEM ARCHITECTURAL DESIGN

4.1 Overview

The system uses AngularJS 1.5 for the frontend, Python Flask for the backend middleware, and MongoDB for the database. The system,

when installed on one of the NKN servers, also interfaces with a user authentication script that checks session ID's against a LDAP authentication database, an Elasticsearch server, and the host machine's file system. admin$_p ublishing_d iagram$

4.2 Frontend

The system uses AngularJS 1.5 which allows the site to be a one page web app. The HTML files are managed by the AngularJS framework which manages swapping out the HTML pages all client side (in the browser) without having to re-query the web server for a new web page. This allows for the system to be extremely fast as every time a new page is requested by the user the website does not need to send another request to the web server.

4.2.1 Dependencies

The AngularJS system uses the following dependencies:

1. angular-mocks: 1.5.0
2. jquery: 2.1.1
3. bootstrap: 3.1.1
4. angular-ui-date: 0.0.8
5. octicons: 2.4.1
6. underscore: 1.8.3
7. angular-route: 1.4.7
8. angular-animate: 1.5.0
9. ngmap: 1.15.4
10. angular-environment:1.0.3

4.3 Backend

The backend middleware is built using Python Flask. Flask was chosen because of its clean syntax, date formatting capabilities, and MongoDB connection support. The clean syntax (no curly brackets) would help code maintenance.

4.3.1 Dependencies

The Python Flask backend uses the following dependencies:

1. appnope==0.1.0
2. click==5.1
3. decorator==4.0.2
4. dicttoxml==1.6.4
5. docutils==0.12
6. Flask==0.10.1
7. Flask-Cors==2.0.0
8. Flask-Moment==0.4.0
9. flask-mongoengine==0.7.1
10. Flask-Script==2.0.5
11. Flask-WTF==0.11
12. Flask-Uploads==0.2.0
13. Flask-Elasticsearch==0.2.5
14. functools32==3.2.3.post2
15. futures==3.0.3
16. geocoder==1.6.0
17. gnureadline==6.3.3
18. ipdb==0.8.1

19. ipython==4.0.0
20. ipython-genutils==0.1.0
21. iso8601==0.1.4
22. itsdangerous==0.24
23. Jinja2==2.7.3
24. jsonschema==2.5.1
25. lxml==3.4.4
26. MarkupSafe==0.23
27. mongoengine==0.9.0
28. nose==1.3.7
29. path.py==8.1.2
30. pexpect==4.0.1
31. pickleshare==0.5
32. protobuf==3.0.0a3
33. psycopg2==2.6.1
34. ptyprocess==0.5
35. pymongo==2.7.2
36. pymssql==2.1.3
37. python-dateutil==2.4.2
38. pytz==2015.2
39. ratelim==0.1.6
40. requests==2.7.0
41. simplegeneric==0.8.1
42. six==1.9.0
43. traitlets==4.0.0

44. virtualenv==13.1.0
45. Werkzeug==0.10.4
46. wheel==0.24.0
47. xmltodict==0.9.2