The following is excerpted from a Technology Platform Review and Project Scope Overview Prepared by Openflows for the Lloyd Sealy Library at John Jay College of Criminal Justice by Eric Goldhagen. **It is not for our class's eyes only.** Lightly edited by Jenna Freedman with emphases added.

Comparison of tools.

Collective Access and Omeka were installed and evaluated over the past 2 weeks.

Both Collective Access and Omeka have solid core development teams and healthy development communities working with and adding to the project's codebases.

One of the clear differences between the two projects is their approach to contributed code and modules. Omeka allows for the release of contributed code as distinct plugins; Collective Access accepts new features and contributed plugins, but carefully vets and reviews the code and once approved it becomes part of the core system. While both of these methods provide for community contributions, it seems that the **Collective Access model has been able to keep those features up to date for new versions** while many Omeka plugins have not been upgraded for the most recent version, and in some cases contributed code is 2 or 3 versions out of date. In some cases, functionality in one Omeka plugin is replaced by a different plugin for later versions. This could create difficulty when doing upgrades.

The user interface for entering records and adding metadata in Omeka is much more refined and user friendly than Collective Access. For example, when scrolling down the page to enter various metadata elements, the submit button in Omeka moves down the page with you so it is always in the upper right corner of the browser window. In Collective Access you need to scroll to the top or bottom of the page to find the submit button.

In Omeka, collections are essentially just another record, with all the same metadata fields as a standard record. **In Omeka, people associated with a record (author, contributor, interviewer, etc.) are simply fields that take input and provide no authority record, ability to de-dupe different spellings or aliases and no way of associating those people to other items in a clear way.**

In Collective Access, people and organizations (referred to as entities) are top level data objects. This allows for **clear authority records and more complex relationships to be drawn between objects, people and organizations**. Because of this data structure, it is simple to provide a user interface that shows all alias names related to a person, and all records where that person is connected to a record, where the relationship is simply a label. This way, you can have a record for Rudy Giuliani that lists his also known as names (Rudolph William Louis Giuliani, Mayor Giuliani) and lists out where he is tagged to various records (prosecutor listed in transcript x; Interview subject for audio file y). While Omeka does have a content type for People, those records are not by default used as authority records and are not used in the author/etc. fields by default.

Areas evaluated:

1: ability to use Dublin core metadata

Collective Access is entirely metadata centric. The installation process of CA requires you select which metadata standard you will be using and it builds its database around that schema. **Once installed, it is possible to hide certain fields or add new fields to the schema. This level of flexibility is not in Omeka.**

Omeka does support Dublin Core in the default install and has a Dublin Core extended field set available as a contributed plugin.

2: user interface for entering data

As detailed above, Omeka has a more user friendly interface.

3: user roles/permissions control

Both systems have the ability to assign user roles with certain permissions to an account. Collective Access however, provides a simple web admin interface that allows for the creation of new user roles and assigning various permissions to those roles.

4: associating records and entities (groups, organizations or people)

As mentioned above, because entities exist as top level data objects in Collective Access, it is much easier to build complex cross referencing interfaces based on a person or organization's roles/relationships to records in the system.

5: associating entities to entities

Core to Collective Access; somewhat similar but not as robust functionality is available in Omeka via a plugin.

6: how to create collections

Both systems allow for simple creation of new collections. Omeka has collections as a flag on a record that is assigned metadata as other records. Collective Access treats collections as a top level data type.

7: handling data of different types (different fields/names depending on type of object)

This is another area where Omeka's simplified data model and interface makes for a more user friendly experience. When adding an audio file for an interview, interviewer and interviewee are distinct fields. In Collective Access all the types of relationships between entities and records are added in the same

interface and get marked with the relationship type – which while enabling more complex possibilities for exploring data, creates a more confusing interface.

8: Search functionality built-in

Based on the very basic and simple tests done as part of this evaluation, the **search functionality in the base install of Collective Access is better than that in Omeka**.

9: Ability to enable field-level searching

Omeka enables field-level searching in an advanced search form by default. Collective Access provides functionality that allows for multiple complex search interfaces with an arbitrary number of fields exposed to the end user, but again the more advanced and flexible nature of Collective Access comes with an added configuration complexity.

10: Faceted searching or browsing?

Both tools state that they can enable faceted searching and browsing, although in the quick evaluation it was not immediately clear how to get this functionality working in either tool.

11: Ability to eventually integrate SOLR for search

Both tools have the ability to integrate with various SOLR search tools. For Omeka, there is a solr search plugin available, however it is a version behind and no information was available on when it would be upgraded for version 2 of Omeka. SOLR support for Collective Access is in the core codebase.

12: Ability to note storage location of physical items represented by digital content.

Storage locations, like entities, are top level data objects in Collective Access. This provides much more robust functionality that is available in Omeka.

13: Ability to track loans of objects or collections

This is another area where Collective Access provides functionality that is not in Omeka.

14: Available themes for public display of data

While both systems are customizable in their display of data to the end user. Omeka has a wide collection of prebuilt themes for download where Collective Access does not.

15: LCSH api access for autocomplete and accuracy of headings.

In Collective Access by default and Omeka via a separate plugin, both tools connect to the Library of Congress public API for subject headings and provide autocomplete and suggestions based on what

you type into the Subject Heading field.

16: Tagging and Controlled Vocabularies.

Both tools allow for unrestricted free tagging of items. The ability to create multiple controlled vocabularies and manage them via the web admin interface is far greater in Collective Access.

Conclusion on which tool to use:

Based on the requirements for this project and how those relate to the strengths and weakness of each codebase. It is our opinion that Collective Access is the right tool for this project.