6-8-2018

Design, Development and Testing an Academic Repository

Kevin M. Handeli
Montclair State University

Stefan Robila
Montclair State University, robilas@mail.montclair.edu

Design, Development and Testing an Academic Repository

Kevin M. Handeli
Department of Computer Science
Montclair State University
Montclair, USA

Stefan A. Robila
Department of Computer Science
Montclair State University
Montclair, USA
robilas@mail.montclair.edu

***Abstract*— Technological advances, established best web development practices, and modern design of the 21st century have provided students, professors, researchers, etc., quasi-effortless access to academic documents encompassed within cleaner user interface and a higher level of usability. Access to and rigorous management of research publication and products is also mandated by funding agencies. Given these characteristics, we have developed an academic repository to maintain the student projects in a typical academic department. A repository of this nature is comprised of two major components: the database and the user interface. The key focus is to provide secure access to research materials (e.g. Master's projects and the like), similar to that of a community or university library's online catalogs, while sustaining a keen eye for contemporary front-end design. In addition to providing access to documentation, this repository uses collected data to present various citation styles, assisting users with easy composition of reference pages. Through a familiar search engine paradigm, information on projects, as well as the projects themselves, are discovered and queried using keywords and will be as if performing a Google search. The repository, while focused on single academic unit, can be expended as a campus wide tool, and provides an in house, easily expandable solution to off-the shelf repository solutions.**

Keywords-academic repository; user experience; user interface

I. INTRODUCTION *(HEADING 1)*

Each year, a typical university advances hundreds of students through its many master's and doctoral programs. Lining the walls of the buildings are artifacts presenting student research and projects. Many of these projects are so interesting and useful, that they can be presented as references for future research and projects. Unless materialized in a professional publication, the documentation that coincide with these artifacts are not easily accessible. This proves problematic to those who wish to expand upon the work of others. Some of the documentation (such as doctoral or masters theses) are digitized and stored in institutional repositories [1]. Previous research showed that digitization and archiving of student projects resulted in increased exposure and enhanced access [2]. While requiring financial investments, institutional repositories have shown to be cost effective approaches to increased data access across a variety of institutions and disciplines [3]–[5]. Moreover, institutional repositories provide the solution to the data access mandate instituted by many publishers and funding agencies (such as the National Science Foundation, National Institutes of Health, NASA) [6].

In many cases, however, deployment and maintenance of such repositories may prove cumbersome or beyond the needs presented by the documents, and data they are expected to service. Such is the case for student reports and projects, where broad accessibility beyond the campus community is not a requirement and where submission of entries and maintenance of the system does not benefit from central support (such as a library).

The benefit of such repository for student projects constituted the driving force behind our work. The main purpose of this paper is to describe the design and development of an online database for graduate projects and theses, with an emphasis on security, efficiency and user friendliness. Though this project was intended for a single department (Computer Science) at our institution (Montclair State University) it can be expanded to meet the needs of other departments and, further, other universities. The second purpose of this work was to engage students in meaningful tool development shaped as a service learning project.

This paper focuses on the development process. It is organized as follows: Section 2 details some of the existing off the shelf repositories, Section 3 presents the requirements established for this project, together with desired outcomes. Section 4 describes the design and implementation for the repository. Section 5 provides a summary of the testing methodology including use cases, security and User Interface (UI) testing. The paper ends with Conclusions and Future Work.

II. INSTITUTIONAL REPOSITORIES

A repository is not a unique concept. One of the most well-known repositories is GitHub, a collection of projects' code to be disseminated to the public, used extensively within the software development community [7]. An institutional repository is defined as a "digital collection capturing and preserving the intellectual output of a single or multi-university community" [8]. Several comparative studies have been completed since the institutional repository concept was introduced decades ago. Such studies are valuable not only through their ability to summarize the current state of the art but also as a mechanism that identifies the requirements and

Identify applicable sponsor/s here. *(sponsors)*

specifications for repositories. As examples, work done by Fay and colleagues and Bankier and Gleason include guides to building institutional repository software. This encompasses information on infrastructure, necessities, design, controls, content discovery, etc [9], [10]. Reading through this document provided many paths to a variety of ideas to integrate into this repository due to its academic skew.

Next, research of existing academic repositories assisted in understanding what should go into one. Examples of known institutional repositories include DSpace, Fedora, and SimpleDL. DSpace, authored by the Massachusetts Institute of Technology (MIT) and Hewlett-Packard (HP) is an open source institutional repository. Its purpose is to distribute research and educational materials complied by researchers at universities [11]. Stated in a January 2003 article in D-Lib Magazine, "It [DSpace] is an attempt to address a problem that MIT faculty have been expressing to the Libraries for the past few years…there is a need to collect, preserve, index and distribute them [research material]" [11]. DSpace's organizational structure is based on "Communities" describing storage and management by schools, departments, labs, etc [11].

Fedora is an open source repository platform created at Cornell University for management and dissemination of digital content. It is capable of housing large, complex collections of works [12]. It is not specialized for institutions, but can maintain academic works for universities. Additionally, it is implemented for libraries, government agencies, and other organizations [12]. Examples of organizations using Fedora are New York University and the Chemical Heritage Foundation.

SimpleDL is a licensed digital content manager and display platform. It is capable of storing a large number of digital formats including images, text, videos, and audio files [13]. SimpleDL differs from other repositories through its sleek viewing platform on top of storage. It is also customizable; its users can modify its look to fit their needs. The downside to this system is its price.

III. Requirements

The project requirements include system requirements as well as specifications regarding format, usage, database design, and interface design. A repository of this nature consists of two main components: the database management system (DBMS) and the graphical user interface (GUI). These represent two major components because they are necessary features that must be developed to design, program, and test a working repository. Fig. 1 depicts these elements and their interactions. In addition to the DBMS itself, initial data must be entered. Although not necessary for functionality to exists, data should be rendered to initiate a proof of concept as well as test properly. These data will be a collection of graduate reports and possibly other related documents. When deployed in a production environment, the application may expand into a repository covering more than a single department, or even other universities.

At the completion of this project, professors, researchers, and students alike, will be able to more easily participate in academic research regarding obtaining good data from very recent sources. Since this repository is to have good upkeep, familiar interfaces should be provided. As example, any user performing research through a search engine such as Google Scholar should be able to adapt their knowledge of searching to this new repository.

Developing the repository in a similar fashion to Google brings with it some merits. Access to the information becomes self-explanatory and familiar. A user will be able to approach this repository and know exactly how to use it without having to search for a tutorial or a guide. Furthermore, users will able to navigate this repository without second guessing themselves as to which page they are being directed. The Google parallel will resolve each of these potential issues. In general, the repository is structured in a way to allow easy access to academic works on a highly usable interface while retaining a high-level of sophistication. The database will be designed to efficiently query data and be kept readable by the administrator. The structure of project storage will be designed to query projects using several different fields. Fields include identifying information, a copy of the abstract, and a file name. Multiple fields will also be used for citation purposes.

The above listed users have specific tasks which they should be able to complete to use the repository to its fullest potential. For example, the lead functional requirement is to be able to complete queries to the database to obtain information on projects. Keywords used to query the database should be able to return information using the title and abstract of the paper as well as a few identifiers such as supervising professor and/or year of publication. Additional functional requirements are to download PDFs of projects, render citations, and for administrative users, to add and remove projects and update user information.

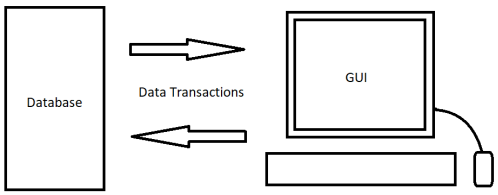

Figure 1. Database and GUI interactions

IV. Design and Implementation

Users	
Fields	
1. Id	2. Username
3. Password	4. Created
5. First Name	6. Last Name
7. Email	8. User type

Supervisor	
Fields	
1. ID	2. Name

University	
Fields	
1. ID	2. Name
3. City	4. State

Upload	
Fields	
1. ID	2. Name

Publications	
Fields	
1. ID	2. Author
3. Title	4. School
5.Semester	6. Year
7. Department	8. Supervisor
9. URL (Optional)	10. Project Type (e.g. Master's Project, Thesis, etc.)
11. Abstract	12. Data file

Figure 2. Database and GUI interactions

After requirements analysis, the next stage is to create the database schema for the repository. The database schema is the description of the database. It is used to map out the appropriate structure of the data as well as explicitly state the data types and constraints for each column of the tables in the database [14]. As a repository involving academic pieces, there are a number as characteristics to consider. Fig. 2 lists these structures.

Designing a simplistic and enjoyable user interface is a challenge faced by many web designers. An important aspect to GUI design is ensuring that all end users are able to use the software/module without confusion and without having to take superfluous steps in order to accomplish a single task. There are multiple steps to preparing a highly usable user interface. Print Magazine highlights an 8-step process to good web design, steps of which were followed when designing the MSU Project Repository interface [15]. Following the refining of the project requirements and scope, user interface application design recommends the establishment of the most important features and creating wireframes or mockups of potential designs. By synthesizing mockups, each page can be laid out and prepared for writing the code [16]. Next, the appropriate web technologies should be decided upon. By choosing relevant web frameworks, programming languages, and web tools, producing the desired look and functionality of the website will become more intuitive. After making these decisions, writing the markup and scripts begins to bring the application to life. Upon creating the structure and functionality, the style can be applied. If a user enjoys the look of a site, user retention is more likely to improve. Finally, the site can be tested, launched, and continues to be maintained by the site administrator [16]. A similar approach was used in our project. Fig. 3 display one of the mock-ups.

Figure 3. Design mockup for the document administration page.

The next major step to simple and accessible design is to choose appropriate web technologies. For the basic layout of this, and most other websites for that matter, Hypertext Markup Language (HTML) has been chosen. HTML is simplistic, allowing developers to easily and quickly design websites. To make website creation even simpler, and standardized, the HTML/CSS/JS framework Bootstrap has been implemented. Bootstrap is open source and used for responsive web design. For this repository, Bootstrap is being used primarily for its grid system [17]. Grid systems allow for easy organization, formatting HTML into rows and columns. Its responsive design allows changes to the layout when moving from desktop sized screens to mobile size. In addition, other Bootstrap classes have been implemented to set a framework for features such as dropdowns, buttons, a navigation bar, etc.

With respect to the client-side scripting language, JavaScript and JQuery have been chosen. JQuery is a JavaScript library used to develop in a faster manner. It also makes HTML traversable and manipulatable through an API which works cross-browser [18]. It also allows for less code to be written and is highly readable, for when future developers wish to continue development. For the back end processing a phpMyAdmin database was used, as provided by XAMPP. XAMPP is a backend web server providing a LAMP stack (Linux, Apache, MySQL, PHP) for development [19]. PHP was used as server-side language. PHP facilitates practical and understandable database integration and manipulation. To perform transactions with the database, Structured Query Language (SQL) is implemented. More specifically, MySQL has been chosen as the relational database management system (RDBMS). MySQL is open source and provides quick processing, is reliable, simple, and flexible [20]. As this repository requires frequent relational database access, MySQL is preferred. Using a relational database specifically is ideal for the repository as there are many relationships connected among fields. An example includes IDs of supervisors related to the supervisor field in publications.

The approach described below resulted in the implementation of a repository with the functionality in line with the requirements. Figs. 4 through 10 provide an overview of the tool. When a user enters the MSU Project Repository website, they are presented with the login form. In Fig. 4 the completed landing login page is presented. After successful login, the user is presented with the main search engine page. Here, they may enter a search query or move on to the advanced search page for more specific search needs. The main page is shown in Figure 5.

If the user chooses to move on to the advanced search, they can select the “Advanced Search” button and will be presented with multiple filtering options. This page can be seen in Fig. 6. After finding and selecting a project, either through advanced search or general search, they have the option to have the document cited for them for paper writing purposes. Fig. 7 presents an example of a citation display modal. If the user does not need the citation, they can move on to information directly related to the project or thesis itself. After selecting the desired project, a page showing identification info, the abstract, and a download of the project is shown. As example, see Fig. 8.

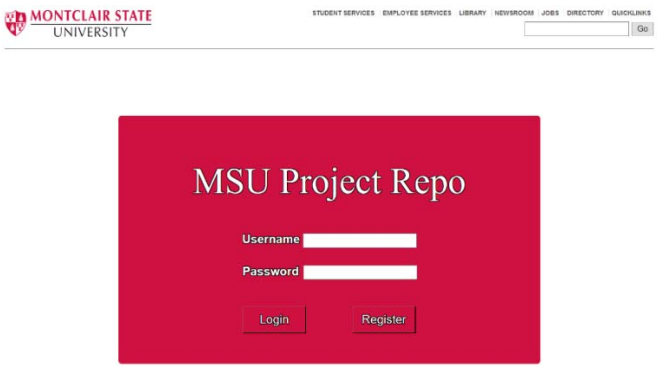

Figure 4. Repository screenshot:Search Engine

Figure 5. Repository screenshot: User Login

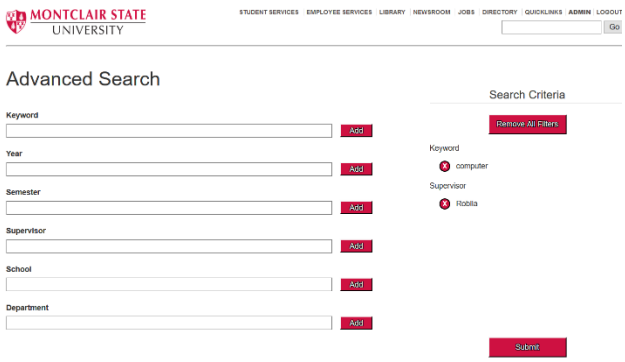

Figure 6. Repository screenshot: Advanced Search

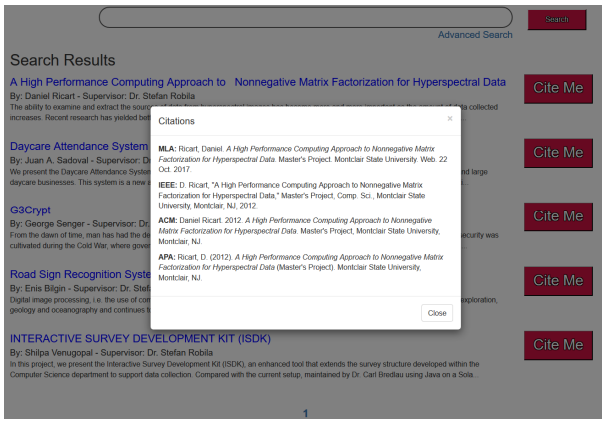

Figure 7. Repository screenshot: Citations

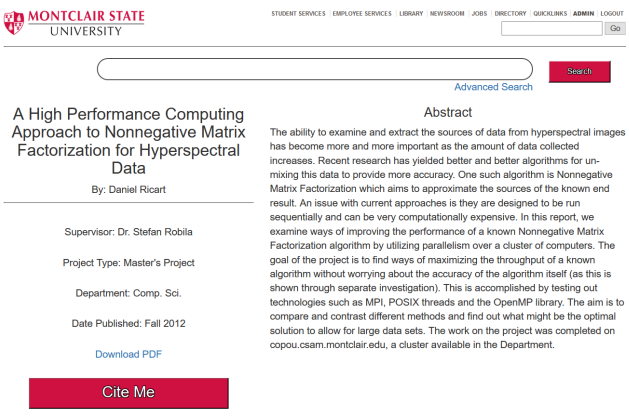

Figure 8. Repository screenshot: Project Information

If the administrator is logged in, they can access the administration page. They have a choice to add a new project, remove a project, or edit user information. Fig. 9 displays the administration page. One completed example of an administration subpage is the "Add New Project" page. The admin can fill out a form prompting for identification information about a project and a copy of the abstract. The "Add New Project" page is shown in Fig. 10.

V. Evaluation

A. General Testing

Following the initial implementation of the application, testing functionality is the next major step in the software engineering lifecycle. Testing is one of the most important steps as it ensures a quality product free many bugs. Of course, the user should ideally encounter no bugs.. The process of testing the various features of the program will be discussed in this section starting with planning the test and then identification of use cases. These use cases were tested and debugged. By the end of testing, a high-quality product was ready for deployment.

With respect to the testing approach. Testing should be completed systematically. They should be prioritized to determine which are the most important functions to work properly at the soonest time and which can be dealt with in the final stages or through patches post-release. Each function will be tested for security holes, expected functionality, and usability. This is also the last stage of software development before usability testing (beta testing) and deployment.

A variety of use cases exist among the functions provided by the repository. Use cases exist in the main search engine, the search engine for removing projects, the insertion of new projects into the database, the registration and login functions, the edit user section, and the results and citation displays. Note that use cases for functionality mostly exists with user input. This is because most user interactions will occur with user input rather point and click. Each case is given a priority from high to low, has an objective, contains lists of inputs/ interactions and invalid data, follows a procedure, and has some output. Two examples of use cases are provided in Fig 11. After identifying the most prominent use cases and testing them steps were taken to remove all discovered bugs.

B. Security Testing

Identifying and patching security vulnerabilities within the system is the next important step. Given the wide variety of users (students, researchers, general public) the risk of security exploitation is high. In the design of the repository we needed to ensure that the users of this repository are not prone to a variety of security risks. Additionally, as the repository requires user management, password reuse is also a risk. As such, these passwords must be protected. Finally, the data in the database must be protected from loss. Without data, the application loses its purpose.

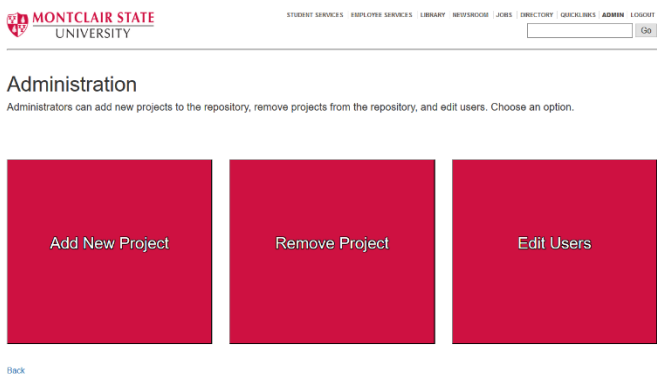

Figure 9. Repository screenshot: Administration Page

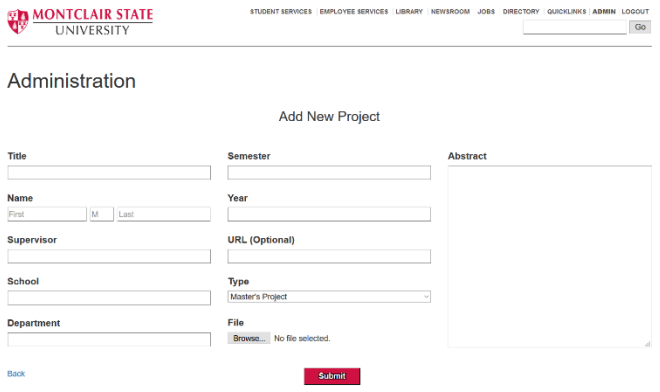

Figure 10. Repository screenshot: Adding a New Project.

Within the large web applications threat landscape Cross-Site Scripting (XSS) attacks are one of the most prevalent [21]. WhiteHat security defines XSS as "a type of injection, in which malicious scripts are injected into otherwise benign and trusted web sites. [They] occur when an attacker uses a web application to send malicious code, generally in the form of a browser side script, to a different end user" [22]. WhiteHat also claims that approximately 75% of websites they have studied contained this vulnerability, making it the least detected and patched threat.

To mitigate this vulnerability, a variety of validation and sanitation techniques have been implemented. For validation using JavaScript, certain input boxes have been constrained to specific patterns. For example, when a user is creating an account, their email address and password must follow a regular expression (regex) pattern otherwise the form cannot be submitted. In addition, all user provided input strings are escaped and have all tags stripped prior to entering into the database. By filtering input through these sanitation functions, any JavaScript or HTML tags sent into the database are removed, rendering their functions into safe plaintext, thus preventing users from unknowingly executing them. By using the "mysqli_real_escape_string()" function, any special characters (e.g. quotation marks, semicolons, etc.) will be escaped into text, again, preventing malicious scripts from being executable when entered into the database. When the data is later queried by the user, several XSS attacks will have been prevented through these methods [23].

Another form of security risk being prevented by a variety of security measures is MySQL injection. MySQL injection is one of the most common threats to a database and is done so through malicious code being entered through user input and executed on submission [23]. Additionally, according to studies run by WhiteHat, approximately 50% of websites have active MySQL injection vulnerabilities [22]. Examples of damage caused by MySQL injection are the dropping of database tables, disclosure of sensitive user information, and changes to field data. To prevent MySQL injections, similar measures to that of XSS attacks have been taken. User input is sanitized and special characters are escaped. Escaping special characters is especially important in the prevention of batched MySQL statements because semicolons are used to terminate queries. By rendering them into plaintext they cannot be used as a terminator, thus fending off malicious queries which may reveal, change, or delete data.

Finally, since it would be very difficult to patch every single database vulnerability, it is important to protect sensitive data, provided that an attack has performed a successful query. In the case of this repository, sensitive data comes in the form of a user's password. Passwords are protected in the database by passing them into a hashing algorithm. The hash will take a combination of the username and password and convert it into a long string of seemingly random characters. Given a situation where an attacker has queried this information, their returned data will be much more difficult to use since they do not know which hashing algorithm is being implemented. To crack the hash, they could attempt to brute force it, but it may not be worth their time and can be very expensive.

C. *UI Testing*

Following the completion of the programming component a brief usability study was also completed. A usability study is essential to good software engineering because it indicates if the repository will be easily usable and appealing to the average user [16].

Test Name: Input a value to query the database
Priority: High
Objective: Verify that the admin can query projects within the database
Possible Input: Sample Test Cases: Integer / decimal values, Characters Aa – Zz, Strings of characters, Punctuation, A mix of all four types.
Invalid Data: None
Procedures: Admin directs self to admin page, selects "Remove Project" button, inputs a value into the search box and selects the go button.
Output: The interface returns a selection of results along with checkboxes for each project.

Test Name: Select projects for removal
Priority: High
Objective: Verify that the admin can select one or more projects to remove from the database
Possible Input: Point and click on checkboxes.
Invalid Data: None
Procedures: From the "Remove Project page" the admin queries projects and selects projects to remove. Then the admin selects the remove button.
Output: The interface returns an alert asking if they are sure they wish to continue.

Figure 11. Use Case examples employed for testing

A variety of users completed several tasks which a regular user would, at some point, complete while using the repository. A pre-survey was first administered to each subject, the subjects then completed the tasks with no scaffolding, and a post-survey was administered to gauge their opinion of the application. Additionally, the usability study further assisted in finding bugs which may not have been found during the testing phase. The tasks to be completed by the subjects are detailed in Fig. 12.

Task 1

1. Go to the MSU Project Repository homepage.
2. Select the "Registration" button.
3. Fill out all fields.
4. Select "Submit".
5. You have successfully registered!

Task 2

1. Log in using your newly created username and password.
2. Perform a search.
3. If results appear, select a project. If no results appear, perform a new search. Repeat as necessary.
4. Download the PDF.
5. Log out.

Task 3

1. Log in to your account.
2. Perform a search.
3. If results appear, select a project. If no results appear, perform a new search. Repeat as necessary.
4. Select a "Cite Me" button.
5. Log out.

Task 4

1. Log in to your account.
2. Select advanced search.
3. Enter two search filters.
4. If results appear, select a project. If no results appear, perform a new advanced search. Repeat as necessary.
5. Log out.

Figure 12. Usability studies tasks

The usability study included users of a variety of backgrounds, ages, and levels of technical knowledge. Consenting subjects were briefed on the goals of the usability study and were required to complete a several tasks that a regular user would come across while using the MSU Project repository. Each subject was first presented with a pre-survey to assess demographic information, background about themselves, information about academic research completion, and computer usage. After collecting this information, subjects were asked to complete each task, with some scaffolding as necessary. Scaffolding included suggestions of possible queries as the database currently has a limited number entries. After each task was completed subjects were asked a variety of questions about their opinion on the usability of the repository. Questions included their opinion on the ease of use, their likes and dislikes, and suggestions on improvements. Completing this study also acted as a beta test as recommendations for changes were requested. At the time of writing, only the small number of users engaged in the study (three) and may require additional testing for validation.

VI. Concuisons and Future Work

A fully functional project repository was designed, implemented and tested using a variety of software engineering approaches focused on application reliability, security and user friendliness. Further improvements include Google or institution specific authentication as replacement of the current system. Google authentication would streamline the login process as most users already have a Gmail account. The repository would use the Google APIs Client Library as a tool for linking user accounts to the authentication system.

Another feature would be to expand the number of types of publications and field characteristics to be added to the database. In addition, one major feature which could be added in the future is the ability to export the project information as an XML (or another format) document. This would be useful for easy transfer of data to other applications through an export function as well is simpler insertion of data through an import function. By adding this feature, the application administrator would take significantly less time working on data entry as it would all be completed through a file upload and the press of a button similar to the approach used in citation applications commercially available (such as Endnote or Zotero).

Finally, some improvements suggested during the usability study could be implemented if they are found to be good refinements. One valuable suggestion idea was the addition of recommended categories. As it is a new system, users may not know exactly where to begin their search, so suggested categories can provide a good starting point. Furthermore, categories can propose popular search terms, for example the words "computer" or "AI". Another improvement suggested was a "favorites" functionality which also inspired a "most popular" filter as well.

The project progress not only contributes to the increased visibility of the institution that may deploy the tool in the future but also has a significant educational value. It also served as an excellent mechanism for learning and applying sound software engineering approaches to the research, design, development and testing of a ready to use web based application to the first author, as part of his culminating experience for the MS degree in Computer Science.

References

[1] K. Yiotis, "Electronic theses and dissertation (ETD) repositories: what are they? Where do they come from? How do they work?," *OCLC Syst. Serv. Int. Digit. Libr. Perspect.*, vol. 24, no. 2, pp. 101–115, 2008.

[2] M. Piorun and L. A. Palmer, "Digitizing dissertations for an institutional repository: a process and cost analysis," *J. Med. Libr. Assoc. JMLA*, vol. 96, no. 3, p. 223, 2008.

[3] C. S. Burns, A. Lana, and J. M. Budd, "Institutional Repositories: Costs and Benefits," in *Proceedings of the Annual Conference of CAIS/Actes du congrès annuel de l'ACSI*, 2013.

[4] Y. Li, S. H. Theimer, and S. M. Preate, "Campus partnerships advance both ETD implementation and IR development: A win-win strategy at Syracuse University," *Libr. Manag.*, vol. 35, no. 4/5, pp. 398–404, 2014.

[5] J. Giesecke, "Institutional repositories: Keys to success," *J. Libr. Adm.*, vol. 51, no. 5–6, pp. 529–542, 2011.

[6] C. M. Buys and P. L. Shaw, "Data Management Practices Across an Institution: Survey and Report.," *J. Librariansh. Sch. Commun.*, vol. 3, no. 2, 2015.
[7] F. Lanubile, C. Ebert, R. Prikladnicki, and A. Vizcaíno, "Collaboration tools for global software engineering," *IEEE Softw.*, vol. 27, no. 2, 2010.
[8] R. Crow, "The case for institutional repositories: a SPARC position paper," 2002.
[9] J. G. Bankier and K. Gleason, *Institutional repository software comparison*. Unesco, 2014.
[10] E. Fay, "Repository software comparison: building digital library infrastructure at LSE," *Ariadne*, no. 64, 2010.
[11] M. Smith *et al.*, "DSpace: An open source dynamic digital repository," 2003.
[12] T. Staples, R. Wayland, and S. Payette, "The Fedora Project," *-Lib Mag.*, vol. 9, no. 4, pp. 1082–9873, 2003.
[13] J. A. Bartlett, "Internet Reviews: Open Access Institutional Repositories," 2015.
[14] M. O'Kane, "A Web-Based Introduction to Programming: Essential Algorithms, Syntax, and Control Structures Using PHP, HTML, and MariaDB/MySQL," 2017.
[15] G. Contributor, "8 Phases of the Web Design Process," *Print Magazine*, 28-Jul-2014.
[16] B. Shneiderman, C. Plaisant, M. Cohen, S. Jacobs, N. Elmqvist, and N. Diakopoulos, *Designing the user interface: strategies for effective human-computer interaction*. Pearson, 2016.
[17] K. A. Rani, *Mastering Web Development with AngularJS and Bootstrap*. Packt Publishing, 2016.
[18] J. Duckett, G. Ruppert, and J. Moore, *JavaScript & jQuery: interactive front-end web development*. Wiley, 2014.
[19] A. Friends, "XAMPP Installers and Downloads for Apache Friends," *Apachefriends Org Www Apachefriends Orgesindex Html 040914*, 2014.
[20] A. B. MySQL, *MySQL reference manual*. 2001.
[21] D. Wichers, "Owasp top-10 2013," *OWASP Found. Febr.*, 2013.
[22] WhiteHat Security, "Web Applications Security Statistics Report 2016," 2016.
[23] Z. S. Alwan and M. F. Younis, "Detection and Prevention of SQL Injection Attack: A Survey," 2017.

Subject 2

Pre-survey

- Age: **31**
- Gender: **Male**
- Ethnicity: **White**
- Occupation: **Software Engineer**
- Education Level (HS diploma, Some college credits, Bachelor's Degree, etc.): **Bachelor's Degree**
- How often do perform searches on the internet: 0 - 1 time per week, 2-3 times per week, 4-5 times per week, **more than 5 times per week**
- On a scale of 1 to 7, where 1 is very poor and 7 is very good, rate your tech savviness: **7**
- On a scale of 1 to 7, where 1 is never and 7 is very often, rate your frequency of completing academic research: **1**
- Do you have any disabilities that would hinder your ability to use the repository, you may decline to answer? **Decline**

Post-survey

- On a scale from 1 to 7, where 1 is very difficult and 7 is very easy, how easy was the repository to use? **5**
- On a scale from 1 to 7, where 1 is very difficult and 7 is very easy, how easy was the repository to learn the functions? **6**
- On a scale from 1 to 7, where 1 is very difficult and 7 is very easy, how easy was the repository to remember the functions between uses? **5**
- On a scale from 1 to 7, where 1 is very difficult and 7 is very easy, how easy is the repository to navigate? **6**
- What was your favorite part of using the MSU Project Repository? **The citation functionality**
- What was your least favorite part of using MSU Project Repository? **Weren't recommended searches. No suggested categories**
- On a scale of 1 to 7, where 1 is very dissatisfied, and 7 is very satisfied rate your overall satisfaction with the MSU Project Repository. **6**
- Please suggest 1 - 3 improvements you would like to see in the MSU Project Repository: **Suggestions when empty. ("You could do"). Same with advanced filters. More information to make it clearer. Possible departments available, Cite me button. Use "Show Citation". Adding projects to favorites or history.**

Subject 2 is a 31-year-old male who is very tech savvy as he has worked in field of software engineering for approximately 10 years. He had no problems navigating the repository and was even able to make good suggestions for enhancements to the design. The usability was found to be very high, as well as the ease of use. The only additionally prompted was suggesting search terms as the database does not have a large amount of data, though he did provide some decent guess work. He found the citation function button to be a little confusing as compared to selecting the project's title when moving to the project information page; a suggestion to also consider for future work.

Figure 13. Usability studies – User Evaluation Example