UNIVERSITY OF THE PHILIPPINES MANILA

COLLEGE OF ARTS AND SCIENCES

DEPARTMENT OF PHYSICAL SCIENCES AND MATHEMATICS

A Special Problem in Partial Fulfillment

of the Requirements for the Degree of

Bachelor of Science in Computer Science

Submitted by:

Ahmad A. Arceo

October 2012

University of the Philippines Manila
College of Arts and Sciences
Department of Physical Sciences and Mathematics

University of the Philippines Manila

Institutional Repository

A Special Problem in Partial Fulfillment

of the Requirements for the Degree of

Bachelor of Science in Computer Science

Submitted by:

Ahmad A. Arceo

2006-11685

October 2012

Acceptance Sheet

The Special Problem entitled "University of the Philippines Manila Institutional Repository" prepared and submitted by Ahmad A. Arceo in partial fulfillment of the requirements for the degree of Bachelor of Science in Computer Science has been examined and is recommended for acceptance.

_____

Aldrich Colin K. Co, M.S. (candidate)
Adviser

EXAMINERS:

		Approved	**Disapproved**
1.	Gregorio B. Baes, Ph.D. (candidate)	_____	_____
2.	Avegail D. Carpio, M.S.	_____	_____
3.	Richard Bryann L. Chua, M.S.	_____	_____
4.	Perlita E. Gasmen, M.S. (candidate)	_____	_____
5.	Vincent Peter C. Magboo, M.D., M.S.	_____	_____
6.	Ma. Sheila A. Magboo, M.S.	_____	_____
7.	Geoffrey A. Solano, M.S.	_____	_____
8.	Bernie B. Terrado, M.S. (candidate)	_____	_____

Accepted and approved as partial fulfillment of the requirements for the degree of Bachelor of Science in Computer Science.

_____

Avegail D. Carpio, M.S.
Unit Head
Mathematical and Computing Sciences Unit
Department of Physical Sciences and Mathematics

_____

Marcelina B. Lirazan, Ph.D.
Chair
Department of Physical Sciences and Mathematics

_____

Reynaldo H. Imperial, Ph.D.
Dean
College of Arts and Sciences

Abstract

With the creation of numerous digital materials every day, there lies an inherent problem on where and how to store them in the context of long-term archiving. This institutional problem for the University of the Philippines Manila is further aggravated with the utilization of multiple repositories – creating information islands of digital institutional assets in the process. This greatly hinders data gathering and dissemination of digital materials within the institution.

As a solution to the said problem, an online centralized repository system for the University of the Philippines Manila is proposed. The repository system allows users to be able to view, store, and share digital materials while providing functionalities such as material commenting and annotation.

The development of the University of the Philippines Manila Institutional Repository made it possible for users to be able to easily search and contribute digital materials with the use of the Internet. On the administrative side, System Administrators are fully capable of moderating content and can even assign moderators to each community to help in the moderation process.

Keywords: *digital library, institutional repository*

Table of Contents

I. Introduction

A. Background of the Study

Education in the Philippines has come a long way. To say the least, the amazing crowd that makes up the local education sector is constantly refining how information is taught and acquired in the academe. It is known that since early 1950s, educators in the Philippines were already utilizing different communication tools such as radio, print, and even video technologies in disseminating information [1]. For the past decade, with the advent of low-cost computers and the Internet, we've witnessed a new age of learning where computers are used as educational tools and as a consequence, a staggering rise in electronic academic materials was observed [2]. This age of learning, also known as digital learning, is responsible for the digitization of traditionally printed academic texts and documents, thanks to readily available and user-friendly word processors and presentation managers. The process of digitization is a game-changer to information dissemination, affecting both how information is delivered and managed, without necessarily compromising content [3].

An institutional repository is a repository of materials created by and for the community of an academic institution [4]. Research on advancements in digital library gradually took off as open archives were being used in storing valuable academic materials. Open archives, as part of the open-access initiative, became quite popular because of the cheap cost (free, in most cases) and ease of delivery of different kinds of materials such as peer-reviewed journals [5]. Result of the research effort came in 2001, where establishing open source institutional repositories (unlike

open archives, institutional repositories are more institution-focused) became a trend among universities in the United States with the release of the EPrints open source repository package followed by the DSpace repository package released one year after the former [6]. These packages both make use of data collecting and archiving tools freely available through the internet.

Today, select universities across the Philippines such as the Ateneo de Manila University, De La Salle University and the University of the Philippines established programs which encourage mass production of well-researched academic materials ranging from theses to university publications, in addition to electronic course materials, annually [7]. However, a system for urgent preservation of these said materials (which would be beneficial for tomorrow's scholars) is yet to be defined for some of the exemplary universities in the Philippines. For example, the University of the Philippines Manila, a premier university in the Philippines which specializes in health medicine, boasts a number of catalogs for preservation, with the help of digitization and open archives, such as the University of the Philippines Manila Research Database [8], the iLib [9], and in some cases smaller archives created by professors to be used in storing relevant course materials [10]. For example, professors create their own course sites or conveniently take advantage of social networking sites such as Yahoo! Groups and Facebook to communicate with students and distribute their own materials. Strangely enough, valuable academic materials are dispersed.

B. Statement of the Problem

Being a premier university in the Philippines, the University of the Philippines Manila and its constituents were tasked by the government to use modern methods of scholarly communication necessary to provide world-class education. In this modern age, computers, with the help of the Internet, address communication gaps between members of the university. The problem lies on how to make use of the Internet to help professors and students find institutional information with ease.

The emergence of computer-assisted learning introduced us to electronic documents and multimedia where information is now stored in languages computers can solely understand. Digital materials come in different formats such as .txt, .doc, .ppt, .jpg and .pdf. The computer then processes it afterwards so that the stored information can now be understood and extracted by us. However, repositories for such files are scarce. Without proper file handling, there is a risk of information loss involved and preservation of such files should be scrutinized immediately.

Moreover, while the presence of separate archiving systems employed by the University of the Philippines Manila could be beneficial to the institution, it could also be a limitation to data gathering. A dissemination problem here is that there is no centralized repository of information; instead we tread in data that is separated in multiple information islands and this method ultimately is time-consuming. Also, multiple repositories, if not organized and well thought-out, would only be redundant and would be just a waste of university resources.

Lastly, we need to utilize a fully-customizable, powerful and flexible repository tool that can serve various institutional needs such as protecting access to data. The popular packages DSpace and EPrints come pre-packaged with archiving functionalities and only offer limited extensibility via plugins. Fedora is an emerging alternative (to the two aforementioned repository packages) that provides the repositorial backend (e.g., content manager) to a system that can be accessed and controlled through a user-created interface [11].

With the problems enumerated, a control would be needed in order to provide a functional repository. A proposed solution would be the implementation of a unified institutional repository with Fedora as its backend to be used by the University of the Philippines Manila and its constituents.

C. Objectives of the Study

The objective of this study is to develop an online institutional repository for the whole University of the Philippines Manila community where users can view, store, and share digital materials while proving functionalities such as material commenting and annotation.

The application establishes different system roles for users with corresponding access rights and permissions. There are four types of system roles:

1. *System Administrator* – The faculty of the University Library are the ones in-charge of the management of the whole repositorial system. System Administrator accounts are

either created by the system's web developer or by another System Administrator. They are also responsible of assigning users to certain roles listed below.

2. *Community Moderator* – The University of the Philippines Manila is comprised of several units namely: University Library, College of Allied Medical Professions, Arts & Sciences, Dentistry, Medicine, Nursing, Pharmacy, Public Health, National Teachers Training Center for the Health Professions, School of Health Sciences (Palo, Leyte), Learning Resource Center (LRC), Office of the Pahinungod and Continuing Education, Philippine General Hospital (PGH), National Institutes of Health (NIH) and the UP Interactive Learning Center (ILC). Each UP unit can have their own *community* within the repository. A community is a collection of digital materials within the repository. Each is managed by Community Moderators who represent their respective units.

3. *University Users* – University Users are comprised of university professors and research experts. However the list is not exclusive to these two entities since the approval of University Users will greatly depend on the contributions of an individual to university research and activities, and university policies.

4. *Registered Users* – Registered Users are comprised of current UP Manila students and alumni, and people that have a direct connection to the university (e.g., honorary members). Additionally, government researchers and former faculty members might fall under this category.

Specific Objectives:

1. allow Registered Users to:
    a. view and download digital materials
    b. search and browse digital materials
    c. create and view personal material annotations
    d. view and add comments to digital materials
    e. view repository news
    f. mark a digital material as a “favorite”
    g. send and receive messages to and from other registered system users
        i. send and receive personal messages
        ii. send feedback to the uploader regarding a material submission
        iii. report a violating user to the System Administrators
        iv. report inappropriate content to System Administrators and Community Moderators
    h. update account details

2. allow University Users to:
    a. view and download digital materials
    b. search and browse digital materials
    c. deposit digital materials (subject to the approval of System Administrators and/or Community Moderators)
    d. manage digital materials one has submitted

i. update the metadata of said digital material

ii. remove digital materials

e. view, add, and manage annotations to digital materials (subject to the approval of System Administrators and/or Community Moderators)

f. view and add comments to digital materials

g. view repository news

h. mark a digital material as a "favorite"

i. send and receive messages to and from other registered system users

j. update account details

3. allow Community Moderators to:

a. view, download, search, browse, and deposit digital materials

b. manage digital materials filed under their respective community

i. approve, add, and delete digital materials

ii. approve, add, edit, and delete digital material annotations

iii. add and delete material comments

c. view repository news posted by System Administrators

d. mark a digital material as a "favorite"

e. send and receive messages to and from other registered system users

f. manage community description

g. update account details

4. allow System Administrator to:
    a. view, download, search, and browse digital materials
    b. manage digital materials
        i. approve, add, and delete digital materials
        ii. approve and delete digital material annotations
        iii. add and delete material comments
    c. manage user accounts and privileges
        i. add and delete system users
        ii. change user roles
        iii. change user account details
        iv. apply account expiration
    d. add and edit communities
    e. view the institutional repository's *system log* (record of site operations)
    f. favorite a digital material
    g. send and receive messages
        i. send and receive personal messages
        ii. send and receive feedback, content flags, and user violation reports
    h. add, edit, and delete site news
    i. update account details

Summary:

Function	System Administrator	Community Moderator	University User	Registered User
View digital material information	●	●	●	●
Download digital materials	●	●	●	●
Favorite material	●	●	●	●
Add material comments	●	●	●	●
Author material annotations	●	●	●	
Send and receive messages	●	●	●	●
Deposit digital materials	●	●	●	
Manage own digital materials	●	●	●	
Approve and delete digital materials	●	●		
Approve, edit, and delete digital material annotations	●	●		
Manage communities	●	●		
Manage news	●			
Manage users	●			
View system log	●			
Update account details	●	●	●	●

D. Significance of the Study

The institutional repository for the University of the Philippines Manila helps the whole academic community in publishing information with ease and practicality. The system allows materials, published or not, be available to everyone especially to people outside the institution. One aim of the study is to reach out to a wider set of audience (perhaps globally), providing information that is the product of the members of the respected institution of the University of the Philippines. The system, being web-based, is focused on a key aspect, which is efficiency, in delivering academic materials.

In addition, the system would be a crucial revolutionary tool in defining the teaching and learning dynamic of scholarship. The system and its data, being community-based, are subject to revision with the help of the annotating and commenting system. These revisions would be important in molding information to its best quality.

The system would also be a great tool in strategic asset management. The system discourages putting to waste information created by the community. This holds true for authentic data, created by professors and students alike that would be a valuable asset to the university. The assets of the university would be great references for future academic works.

E. Scope and Limitations

1. Requests for repository access are subject to the approval of the System Administrators (University Librarian). The System Administrator also determines the system roles for the users in the system, as well as potentially suspending or revoking access rights of a user. This process is not considered to be within the scope of this software.
2. The application for an account is subject to the administrator's approval.
3. The approval of uploaded digital materials can only be done by the System Administrator, of the moderators of the community that the uploaded materials are classified under.

4. Metadata, such as the deposited data's category or subject, are manually defined by the depositor. Due to the variety of digital materials, it would be impossible to automatically generate metadata for all types of content.
5. Digital materials and their metadata submitted by users cannot be changed by System Administrators and Community Moderators. However, they can simply reject the submission and let the submission be revised by the uploader.
6. There is no way to automatically determine the changes that were made between versions of a deposited material.
7. Indexing of file contents is limited to text-based digital materials (PDF uploads are included as well as text-based files). Digital materials such as pictures and videos will rely on user-defined descriptions.
8. The size of data files should not exceed by 100 MB. The 100 MB limit is reasonable enough to be able to archive most text-based digital materials. Data that exceed the said limit will not be ingested in the system.
9. The system doesn't automatically access data that web browsers could not natively handle. Users should make use of required software applications necessary to open specific file formats.
10. The system doesn't utilize web spiders that automatically track inappropriate content. Also, the system would not be able to detect plagiarized files. One way in which this could be done would be that users would report inappropriate content to the moderators.

11. Response to tickets (flagging of inappropriate data, reporting a violating user) is subject to the System Administrators and/or Community Moderators. There is no auto-response ticketing system available.

F. Assumptions

1. It is assumed that the use of the repository is subject to institution policies, and that the assigned System Administrators are responsible for carrying them out. These policies define what materials are accepted, or deemed appropriate, and what sanctions are applied to violating users.
2. The community as a whole is responsible in safeguarding the credibility of the institutional repository.
3. The depositor of the digital material is assumed to be its owner, and that the owner is capable of providing all required metadata during its submission.
4. It is assumed that this system would undergo further improvements to comply with university policies as well as to complement with new communication paradigms in the future.
5. The system assumes that users are knowledgeable enough to be able to navigate the web interface. Users are also assumed to have been assigned one of the four system roles that have access to the system.

II. Review of Related Literature

Every time we came across the term repository, we think of storages such as libraries and storerooms where books, papers and other physical objects are being kept for archiving purposes. Today, the usage of computers brought a rise in the production of digital materials. People learn to exploit the technology and the convenience from utilizing such materials are loved by people. In fact, in a Digital Library Federation (DLF) study conducted in 2002, survey shows that the use of the internet and electronic materials is unanimously favored and appreciated by teachers and students of collegiate-level institutions. About 87% of the respondents prefer going online to look for research materials as opposed to using library catalogs and terminals for physical books [12]. The increase in the production of digital materials produced by the scholarly community ushered the revolutionary development of web-based repositories.

Repository systems are always attributed to systems that solve the problem of digital material storage. The internet became a platform for many popular data storage services such as Google Docs [13] and Dropbox [14]. Depending on the service, a user can deposit any type of digital files (e.g., a .doc file) to the data storage service. Likewise, a user can also pull data out of the service, and if permitted, can also access other people's storage. It can only provide the basic storage needs, however, although Google Docs have editing functionalities for file formats such as .doc because it also acts as a word processor. Usually, these services have limited storage capacities and in the case of the two aforementioned services, hold the user under some terms for usage. Google Docs for example, holds the right to modify and delete uploaded materials from

any of their services [15]. Generally, using these data storage services presents a risk to long-term data storage and collection.

The Protein Data Bank, arguably one of the most prolific example of data collection in modern history, went several changes to address data preservation over time since its launch in 1971. From a meager dozen files worth of research, the Protein Data Bank today boasts a large amount of electronic data and is currently the definite source of information in the field of structural biology [16].

The same year the Protein Data Bank was launched also produced yet another repository system, the Project Gutenberg [17], which became the pioneer in the field of digitization and storage of electronic texts. Through volunteer efforts, the project aims to deliver electronic versions of books and music sheets for free across the internet [18]. What started as a spontaneous idea to its colleagues by its founder, Michael S. Hart, became storage for over 20,000 e-books ready for immediate and free download. The non-profit organization's main website, hosted in the United States, counts more than two million downloads each month [19].

Besides books, media files such as images have a special place in the internet. Wikimedia Commons [20] is an online repository for deposited free-use images, sound and other media files. The site is currently home to over 11 million media files. These files are freely available for download provided it is used for non-profit purposes. Wikimedia Commons was established to act as a central repository for images to be used by other projects of the Wikimedia Foundation such Wikipedia (a popular online encyclopedia) [21].

There are also digital archives that are based on commercial electronic document management systems (EDMS). The Public Record Office Victoria (PROV), a repository of records for state government agencies in Australia, makes use of Documentum EDMS in implementing digital archiving functionalities [22]. Documentum is a popular EDMS that includes versioning, workflow, and an interface that allows extensive customization in Java [23]. Documentum greatly stresses archiving final versions of documents and it is not recommended to store information that is considered as work-in-progress such as theses drafts [24]. The use of commercial electronic document management system is profound in non-academic institutions which aim to store information privately while being able to organize them in a unified environment. Additionally, the low adoption rate to commercial systems by academic institutions is due to the significantly high costs of such systems, at least from an academic standpoint [25].

Repository systems seemed to have evolved over the past years. Transitions from passive data storage systems to interactive systems became a trend nowadays. An example of such implementation is the Casepedia, a repository platform that allows medical professionals to publish, comment on, and classify authentic medical cases. The platform's design integrates users into the system by providing the tools necessary for interaction and collaboration while providing repositorial functions. The commenting system can attract users work with each other while at the same time, can be used to increase the quality of information available on the repository [26].

Discipline-based repositories established by research groups, such as the PubMed Central began to surface last decade – addressing new challenges in long-term content curation. The

PubMed Central freely hosts digital journals in the field of biomedical and life sciences and it features automatic indexing in Medline [27]. Another noteworthy discipline-based repository that made contributions to advancements in scholarly communication is the physics-based arXiv repository. While it does not provide certification to publications, it conveniently eased information dissemination and as a consequence, the repository currently holds the largest collection of e-prints metadata online [28].

The discipline-based repositories such as the PubMed Central and the arXiv serve as the precursor to the current institutional repository movement [29]. In addition to the trending open-access initiative, discipline-based repositories made way to the creation of the Coalition for Networked Information (CNI) organization headed by Clifford A. Lynch. Through the efforts of the Coalition for Networked Information which seeks advancements in scholarly communication assisted by networking technologies, institutional repositories, which are university-based repositories, are becoming more visible in academic institutions [30].

In 2002, Virginia Tech developed the ETD-db repository tailored to host electronic theses and dissertations (ETD). The development was ushered by the increasing acceptance to ETDs as a viable medium in scholarly communication [31]. The strength of this repository can be attributed to its easy authoring and supervision of e-theses. Being a theses management system, besides from the usual title and author information, it also collects information regarding a digital material's defense date and a list of members (committee) that examined the said work. For depositing data, it offers FTP (File Transfer Protocol) support that stores information

externally. However, repositories such as the ETD-db are posed with extensibility problems due to the limited types of files it can cater [32].

To date, there are two popular institutional repository software packages openly available to the public: EPrints and DSpace. The University of Southampton's EPrints institutional repository package is considered one of the first software packages developed to feature institutional data collection. The EPrints software package features seamless single-file uploading, customizable multifaceted data browsing and it makes use of open source software like Linux, Apache, MySQL and Perl. EPrints also features customizable control to submissions in the form of embargoes, in which the time and date of data publication can be specified in advance [33]. DSpace, a project developed by Hewlett-Packard Company and the Massachusetts Institute of Technology (MIT) Libraries, is a repository system that is focused on long-term data preservation. DSpace extends EPrints' user-friendly submission interface to feature bulk file uploading. Labs, centers, schools or departments can have their own communities within the system. Each community will be subject to customized rules on how specific data is stored and accessed. In addition, the system also acknowledges subscribing to updates regarding new submissions, in which an e-mail will be sent to interested users [34].

A prominent repository that utilizes the DSpace repository software package is the Deep Blue. Deep Blue is a project by University of Michigan which is motivated to provide visibility, permanence, comprehensiveness, and control to the university's digital assets. DSpace was adopted because of its incomparable popularity and ease of deployment while being able to meet the university's goals in long-term preservation of files. The system gets its bulk of information

from professors, although students can also deposit their self-authored materials. These materials are mostly in the forms of finished articles, unpublished works and related datasets. To retrieve files, a user needed to be a member of the institution and should be a registered user of Deep Blue. However, Deep Blue is limited by plenty of interface problems which hamper the overall usability of the site [35].

The University of Glasgow's DAEDALUS project explored utilizing both the EPrints and DSpace institutional repository software packages in an attempt to acknowledge the said university's needs in managing content. While EPrints focuses on electronic print materials such as dissertations and research papers in contrast with DSpace's acceptance to video and sound formats, the astounding similarities between the two makes a sufficient argument that the two can complement each other [36].

One major barrier in the success of institutional repositories can be attributed to the presence of convoluted file submission and user registration processes. Another inhibiting factor (in a repository's growth) revolves around a myth claiming that open-access materials deposited in repositories are of low quality compared to the physical and proprietary ones [37]. These are some of the challenges that are acknowledged through an international conference on institutional repositories hosted by the Coalition of Networked Information (CNI) in 2005, although these problems were not fully addressed [38]. Overlooking these problems, current academic institutional repositories have responded greatly towards helping institutions communicate with the world. In a way, these repositories not only serve as a showcase of excellent academic materials but also help in bridging gaps between members of the community

[39]. It provides the opportunity for professors and students alike to be united by offering a unified space for them to deposit their accomplished works.

III. Theoretical Framework

A. Digital Materials

Centuries ago, the ancient Sumerian civilization devised a way to store information by writing symbols in clay tablets with the use of a blunt reed as a stylus [40]. With the invention of the pen followed by the printing press in the year 1440, the modernization stressed the use of paper as a medium for communication. From written forms, people learned to record sound in magnetic tapes and store images in photographic films – both requiring electricity for information retrieval. Today, with the emergence of computers, people have learned to produce digital materials that contain "recorded information structured for human consumption [41]", and continuous research is ongoing on how to store them.

Digital materials refer to digital (electronic) objects or files stored in a physical medium usually accessed through a technological component such as a computer. The word digital stemmed from the nature of electronic files stored in simple binary digits (0 or 1) or states (on or off) which are processed by a computer for us to comprehend [42]. These digital materials could have digital origins like a document or image originally created with the use of a computer application. In several cases, digital materials are derived from physical materials (most of which are paper publications) through the trending practice of digitization. A common example of a digital material is the electronic book or e-book, where (in most instances) an exact copy of the physical book is stored electronically; intangible, but can be easily reproduced perfectly. Other examples of digital materials include:

- Theses/dissertations
- Preprints/e-prints
- Conference proceedings
- Conference presentations
- Tech reports/working papers
- Journal
- Newspapers (born digital)
- Data sets
- University publications
- Departmental materials or records
- Digital images
- Digital audio
- Interview transcripts
- Maps
- Software
- Course content
- Learning objects
- Student papers other than theses or dissertations
- E-portfolios
- Campus blogs
- Newsletters
- Laboratory protocol
- Book manuscript

- Web pages

Note that some of the digital materials enumerated are considered complex digital materials since these can consist of other digital materials. Compared to traditional (paper) print materials, exploiting the benefits of technology to the fullest has made these complex digital materials dynamic, enhanced, richer in content and more interactive [43]. For example, in addition to text material, a web page could also contain images, sounds and/or videos to be able to completely present information. A digital material like the web page has a different internal structure dictated by a container that makes it possible to be conveyed in such manner. Depending on content and how information should be presented, a digital material is contained in a suitable file format (container) which is either open or proprietary. Text documents usually take up the formats .txt, .doc, .docx and .pdf while digital images are presented in .jpg and .png formats.

Digital materials possess several advantages over print materials – and these advantages are inevitably discernible in both corporate and academic communities. A study mentioned by Wu in her paper *Why Print and Electronic Resources Are Essential to the Academic Law Library*, pointed out that the introduction of digital materials, enhanced by the utilization of networking technologies, resulted to a 25% decrease in the amount of time spent doing research within corporations [44]. Although the use of digital materials require an individual effort to be computer-literate, the challenge is seemingly rewarding due to the fact that digital materials can be easily delivered, are readily available anytime and can be freely reproduced, and can quickly provide specific information with the use of indexing and search tools [45]. Users are also given the opportunity to be able to update digital materials instantly with ease, less cost and effort.

Compared to paper publications, digital materials do not take up physical space; although, they could be conveniently stored in portable storage devices such as a CD-ROM that can store huge amount of information.

The (digital) nature of digital materials raises preservation issues in a similar fashion to the same dilemma years ago where people have to deal with preserving information stored in fragile storage media such as nitrate films and acid-based papers. Ideally, the storage for these materials is expected to live on for decades or even centuries, is damage and degradation resistant, and has the ability to be able to store tons of information [46]. However, today's market demands the continuous flow of advancing technologies (hardware, software and storage media required to access digital materials) – with the aim of providing better quality products while decreasing production costs, and this fact (the fact that we are still far from grasping the perfect storage system) serves as a limitation in preserving digital materials [47]. In summary, the advantages and benefits of digital materials don't come without a price – and this serves as a challenge that is constantly being addressed.

i. Searching

When presented with large quantity of digital materials, it would be inefficient to go through all the stored files to look for a particular word or phrase. Instead, we utilize specialized tools that aid in searching through the heap of digital materials. These tools automate the searching of three common digital material information: file name, metadata (which will be discussed later on this section) and its content. Modern software applications

and operating systems have file search functionalities as a feature. A simple search makes use of string matching, where a query by a user is exactly matched against stored information. The significance of the search functionality is that it saves time in finding digital materials. Additionally, it also functions as a filter for results by suggesting relevant materials (based on the user's query).

ii. Indexing

Searching large quantities of digital materials could take a large chunk of time before a result is returned. For a search to be optimized and efficient, an index of files should be created [48]. Traditional libraries make use of file catalogs as an index to physical books. An index gives an overview about a particular book by containing information such as its title and subject, and more importantly, the location of the book. In a similar way, digital files can be indexed by the system. An index contains most relevant information about a digital material which will refer back to the actual file. In this fashion, the scope of the automated search is limited to searching the index, greatly reducing the amount of time results are returned.

iii. Metadata

Long before the advent of computers, metadata exists in the form of cards found in library card catalogs. A card contain a list of vital information about a book which specifies its Dewey Decimal Classification code, title, subject or keyword that best describe the book,

author/s, and year of publication [49]. These vital information gives us a brief explanation or overview about a book and its contents – even before having a direct access to it.

Broadly speaking, metadata is defined as data about data (data about aspects of data, to be precise). In the context of this paper, the term metadata will mostly refer to structured information that describes digital resources. Non-conventional use of the term, such as metadata that deals with information access control, will be noted in their respective sections.

The metadata, being a structured information, is basically composed of the elements (fields) and its contents (values given to the elements). Metadata elements are the predefined representations of specific attributes of a resource like its authorship while the contents are values that give meaning to these representations [50]. Figure 3.1 shows an example of a metadata describing a digital book. "A Day Late" is the title of the book, with "David, Armando" specified as its author and is classified under the "Romance" genre. The metadata elements would be the title, author and genre. The strings "A Day Late", "David, Armando" and "Romance" are considered the contents of their respective elements. For familiarity and clarity purposes, the metadata syntax is made to resemble that of HTML (Hypertext Markup Language) tags (note that this is not exclusive as different metadata standards could have their syntax expressed differently).

```
<book>
  <title>A Day Late</title>
  <author>David, Armando</author>
  <genre>Romance</genre>
</book>
```

Figure 3.1. A simple example of a descriptive metadata

The internet is responsible for the massive wave of digital materials published online (mainly due to ease and convenience) however, it initially became a risky medium for preservation. As the amount of digital materials increases among organizations and institutions, an increase in the level of difficulty to discover relevant materials followed, and a need for newer resource discovery options seems inevitable. Gone are the days where search engines are content in solely indexing the filenames of digital materials as these names, say index.html, doesn't really say much about the content of the file. This problem becomes more critically complex for non-text digital materials such as an audio or video file. The use of metadata deals with the problem of resource discovery since it enhances information and visibility of a material, making it more accessible to people looking for specific data. Optimistically speaking, metadata is integral to the preservation and discovery of digital materials [51].

While some metadata are embedded naturally to digital materials, the discussion of an institutional repository suggests metadata to be stored in some metadata repository or database. In this manner we can ensure that online information that is seemingly lost can still be tracked down since the metadata is independent from its host. The independent metadata could contain information collected from the embedded metadata, or could be a newly-

created entity specified for a particular digital material. The independent metadata features a unique locator element in which the content is an address which serves as a link to the digital material. The content of the locator element is dynamic – adapting to file location changes to ensure no information is lost in the process.

In the case of generating metadata, there exists two possible methods: either manually (traditional) or automatically with the use of computer applications. Automated tools generate metadata from digital materials faster at the expense of varying accuracy and consistency. Information such as the publication date of a certain document can be easily extracted, however fields requiring qualitative or subjective data such as a simple description of say, a picture, is subject to guesswork. In this regard, poor quality metadata may arise leading to confusion and perhaps crippling the discovery of valuable resources. On the other hand, the manual encoding of metadata could pose as a better alternative although it still deals with similar issues regarding the quality of metadata generated due to human error. One solution to this problem, as recommended by a study conducted by Greenberg et al. in 2001, is to create user-friendly web forms that makes use of brief descriptive text guides, pop-up windows and drop-down menus to help in the process of metadata generation [52].

The proposed repository tackled in this paper implements the manual approach to metadata generation with the help of web forms recommended by the aforementioned study. These forms should provide the opportunity for authors of digital materials to be able to create exceptional metadata that will best describe their most cherished works.

a. Dublin Core

A great challenge in building an institutional repository is how to handle metadata for different kinds of interdisciplinary digital materials. Different disciplines have different metadata requirements – same goes for different kinds of digital materials. In a system where the metadata plays an integral role, the selection of the proper metadata schema would lead to the success and quality of that system. A metadata schema defines the core element set (that could be expanded) to satisfy certain requirements. There are hundreds of metadata standards and schemas to choose from although the Dublin Core, which was conceived in 1995, became the established de facto metadata schema in providing solution to the institutional repository predicament [53].

Dublin Core is considered as "the most important standard for the simple description of electronic information resources [54]." It is usually implemented to make use of the XML (Extensible Markup Language) format, which is an extension of the HTML, for an easy exchange of information across the internet. The Dublin Core's set of 15 meticulously selected metadata elements can sufficiently describe a wide array of digital materials. Every element in the set is locally defined, optional and repeatable. Below is a list of elements that makes up the Dublin Core set, each with their own definition [55]:

- *contributor* – An entity responsible for making contributions to the resource.
- *coverage* – The spatial or temporal topic of the resource, the spatial applicability of the resource, or the jurisdiction under which the resource is relevant.

- *creator* – An entity primarily responsible for making the resource.
- *date* – A point or period of time associated with an event in the lifecycle of the resource.
- *description* – An account of the resource.
- *format* – The file format, physical medium, or dimensions of the resource.
- *identifier* – An unambiguous reference to the resource within a given context.
- *language* – A language of the resource.
- *publisher* – An entity responsible for making the resource available.
- *relation* – A related resource.
- *rights* – Information about rights held in and over the resource.
- *source* – A related resource from which the described resource is derived.
- *subject* – The topic of the resource.
- *title* – A name given to the resource.
- *type* – The nature or genre of the resource.

The adaptation of the Dublin Core is also relevant to long-term preservation and discovery needs of an institution, as metadata harvesters such as the one provided by the Open Archives Initiative – Protocol for Metadata Harvesting (OAI-PMH), searches and harvests XML-formatted metadata utilizing the Dublin Core schema. Provided that a repository opens its metadata collection to the harvester, this process of harvesting would lead to indexing and would be crucial to information discovery as this allows data to be publicly available, interoperable and searchable even outside of the hosting institutional repository. However, the proposed institutional repository discussed in this paper is

limited to strictly adapting the Dublin Core schema in preparation for later harvesting, as internal testing would be important in the initial stages of the development of the said repository.

```
<?xml version="1.0"?>
    <!DOCTYPE rdf:RDF PUBLIC "-//DUBLIN CORE//DCMES DTD 2002/07/31//EN"
     "http://dublincore.org/documents/2002/07/31/dcmes-xml/dcmes-xml-dtd.dtd">
    <rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
     xmlns:dc ="http://purl.org/dc/elements/1.1/">
    <rdf:Description rdf:about="http://dublincore.org/">
        <dc:title>A Day Late</dc:title>
        <dc:author>David, Armando</dc:description>
        <dc:type>Romance</dc:date>
        <dc:format>text/html</dc:format>
        <dc:language>en</dc:language>
        <!-- guesses for the translation of the above titles -->
        <dc:title xml:lang="fr">Un jour de retard</dc:title>
        <dc:title xml:lang="de">Een dag te laat</dc:title>
    </rdf:Description>
</rdf:RDF>
```

Figure 3.2. An example of an XML-formatted Dublin Core metadata (based on Figure 3.1)

iv. Versioning

Identical materials within the system can be quite ambiguous. The problem arises when materials tend to be so similar that there is no clear cut about their differences. It could be possible that unrelated materials be given the same description, while others could just be mere duplicates. We could mistakenly identify an updated version of a previously existing material for being the latter that was deposited into the repository at a different date. We could simply fix the last problem by removing the obsolete material from the system, leaving the updated file behind – however, this would be unnecessary as every material deposited in

the system is considered an institutional asset that could possess valuable information. Additionally, while updated materials may follow a trend of being a superior product from their predecessors, this does not generally hold true at all times since updates can be evaluated subjectively. As a conservative alternative, the proposed institutional repository implements file versioning. The term versioning refers to the process that deals with recording the history of file updates and changes happening within the system [56].

Through versioning, we can easily determine the association of materials by looking at its history of changes. The recorded data not only gives us the information about whether a material is the initial/original, intermediate, or the latest/final material, but it also provides a list of its parents (materials that the material in discussion render obsolete), children (materials that make the material in discussion obsolete) or both (if existing). This simplifies the selection of materials in a research process as versioning segregates related materials from non-related ones, and from this we proceed to filter the ones relevant to us. Established information systems associate digital materials with each other by intuitively utilizing metadata that can be easily stored in databases. For example, the Collaboratory for the Multi-scale Chemical Sciences (CMCS) knowledge portal makes use of metadata to provide the versioning requirements of files. Extended metadata elements such as Is Replaced By and Replaces serves as a bridge in linking identical resources [57]. From the contents of these elements, we can effectively deduce the relationship between materials and this serves as the basis in determining the complete history of a file.

We could have an idea about most of the different versioning schemes just by recalling popular software application names. Applications such as Adobe Photoshop CS2 (alphanumeric), Firefox 6 (numeric), and Microsoft Office 2010 (year) all have their versions affixed explicitly. These versions are further extended with the introduction of a *patch* which provides changes to a software application. For applications that employ the numeric versioning scheme, a common practice is through incrementing the current application build's version number. Usually, the extent of an increment is directly proportional to the extent of changes made by a patch. Using Firefox 6 as an example, a version of 6.0.1 denotes that a small fix was applied to the original Firefox 6 application, while a version of 6.1.0 means that some major changes were applied. However, notice that these different kinds of version schemes all needed to be manually addressed (due to the arbitrary nature of such schemes). The best way to alleviate depositors of digital materials from the complexity of version naming is by adapting the date and time versioning scheme. The date and time versioning scheme is dependent on when the digital material is deposited, freeing the depositor from naming responsibilities.

In collaborative environments, versioning serves as a guide to groups working on a specified version of a project. Versioning also adds flexibility to the research process as it gives researchers an opportunity to choose an instance of a file that will best satisfy their needs. As Jenny Brace puts it:

> *"For example, one researcher may need to find the published version in order to give an accurate citation, whilst another would like to view an earlier version of the same work*

in order to assess the development in the thought of the author or content creator. Another person might need to know exactly what information was provided at a particular date to provide evidence, for example in a court case [58]."

In summary, not only is versioning a useful tool in information discovery but it also keeps materials organized and associated. As repositories continue to evolve into more collaborative forms, the flexibility that is expected from versioning would be a useful complement to group-based decision making.

B. Institutional Repositories

Depending on an institution's mission, vision and goals, an institutional repository can be defined in several ways. Below is an interesting view (and possibly one of the earliest definition of the term) from Raym Crow of the Scholarly Publishing and Academic Resource Coalition (SPARC) on institutional repositories:

"any collection of digital material hosted, owned or controlled, or disseminated by a college or university, irrespective of purpose or provenance [59]."

Meanwhile, The Coalition of Networked Information's director, Clifford A. Lynch defines an institutional repository as:

"[a] set of services that a university offers to the members of its community for the management and dissemination of digital materials created by the institution and its community members. It is most essentially an organizational commitment to the stewardship of these digital materials, including long term preservation where appropriate, as well as organizational and access or distribution [60]*."*

It should be noted that both definitions, while providing different views on the subject matter – Crow describing it as a collection in contrast with Lynch's view as a service, are strikingly compatible and share a common ground in capturing the essence of an institutional repository. In addition, it would also be reasonable to define such repositories as open and interoperable as barriers in scholarly communication are continuously being addressed proportionally with the maturing technology [61].

C. Fedora

Fedora or Flexible Extensible Digital Object Repository Architecture is an open source repository software that can serve for a variety of use cases such as digital asset management, institutional repositories, digital archives, content management systems, scholarly publishing enterprises, and digital libraries. In a nutshell, Fedora focuses on the archiving aspect of digital materials. The creation of Fedora was mainly motivated by the problem of storing complex digital materials especially on how to establish relationships between files (e.g., multi-part documents, heterogeneous) [62]. Fedora provides a Digital Object (where digital materials are stored) Model that offers the following advantages:

- *Abstraction* – Fedora treats different kinds of materials, regardless of complexity, equally. This applies to unknown and new digital objects.
- *Flexibility* – Fedora also provides the opportunity to allow implementers to design their model that can best represent their requirements. One example is that an object's metadata schema can make use of other schemas other than Dublin Core.
- *Generic* – Fedora objects containing digital materials and metadata are linked with each other.
- *Aggregation* – Fedora objects refer to data that is stored locally or that is stored on any web accessible server.
- *Extensibility* – Fedora services are associated with data within the object. As service change, so is the object.

The main features of Fedora includes the creation and management of digital content objects, content versioning, simple search and indexing, and the presence of web-based interface that allow data access. Other Fedora functionalities include the capability to export a digital material into an XML-formatted Dublin Core file [63]. Unlike DSpace and EPrints, Fedora is not a complete management service. Instead, it relies on external applications that can connect to it through the APIs provided. This design adds longevity to Fedora, since the repository software is not limited by pre-packaged software making it more viable to accept the newer advances and trends in web services. To date, it still receives support and is still undergoing continuous development.

D. Digital Annotation

The aim of the proposed institutional repository is not restricted to solely storing digital materials – in fact, it is also intended to be a common knowledge space in such a way that information is shared through the communication between users in addition to the interactions with stored materials. In support to this, an annotation service is implemented as a complement to the system's built-in text viewer.

According to Decurtins et al., an annotation is "some means of marking a document to augment its content [64]." It supports several basic cognitive functions such as remembering, thinking and clarifying [65]. An annotation could be an external simple note beside a lengthy news article or a personally highlighted text in a book for a quick review later on.

Much like annotations in a traditional sense, digital annotations could take different forms; they could either be public or private, formal or informal, or could be permanent or non-permanent. These digital counterparts are a separate entity to the document they are trying to define and could be as simple as a short text comment or even as complex as a video annotation. As stated above, these annotations could also be shared to the public which is encouraged by the scholarly community to provide dynamic discussion through sharing of perspective and information on a common interest.

Being a service to the built-in text viewer, the annotation functionality is strictly limited to text-based annotations with highlighting capabilities. The user should be able to create, modify

and delete their own annotations, set the annotation's visibility scope and be able to retrieve a list of viewable annotations from peers. In addition, the user can explicitly annotate specific text by marking its location in the document.

E. Document Access Control

While the proposed institutional repository advocates open-access to digital materials, it would be beneficial to be able to cater to various needs at the other end of the spectrum. A major concern in building information systems lies heavily on how security (integrity and confidentiality) is being handled. For instance, some contributors would prefer that their materials only be shown to select members of the community. Materials such as solution manuals should have access restrictions as their contents are aimed towards instructors. In other cases, we have groups making use of the institutional repository as a form of communication between members, and the materials involved in the research could be at risk if the data is exposed prematurely. Finally, there are existing materials that are held by contracts or obligations (although it is permissible to publicize specific information such as a material's abstract), and the expiration of such will make the material freely available for dissemination.

Majority of the implementation problems that needed to be addressed promptly during the conceptual stages of repository making is related to the examples given above. The implementation of a security service such as a feasible access control system is a necessity in negating unwanted security breaches. Usually, access control deals with exclusively limiting the interaction between users and digital materials, however this is not the case as access control also

encompasses different aspects of the institutional repository such as establishing rules on user management. In some computer systems, access control functions as a service that deals with authorization and auditing [66]. Basically, authorization deals with the verification of a user's identity by requiring the submission of credentials that would validate the user's legitimacy. Modern information systems make use of login screen as a common standard in the authorization process. Auditing on the other hand, adds complexity to the system's security by associating and recording user actions made to system resources.

People today have a brief understanding on how access control works since most systems (digital and physical) are already employing such service. In computer systems, login screens require the input of a username and a corresponding password for identification purposes, and a successful login may grant access to a resource depending on organizational policies. These policies are specifically tailored guidelines that address questions regarding institutional access:

1. *Who*
    - Who can access the digital materials in the repository?
    - Who have the privilege to deposit data?
    - Who are capable of creating system users?
2. *What*
    - What operations can be done on digital materials?
    - What digital materials are accessible to users?
3. *When*
    - When can a user access a file?

The possible interactions between two system resources are encoded in metadata that are typically stored in a database. The structure of the metadata is derived from the guidelines discussed above. The metadata for every system resource are stored in two-dimensional access control lists (ACL) which serves as an important reference for access control. Before a user can delete a certain digital material, the system would conveniently refer to the ACL if a user is allowed to perform such operation. Access control lists are proven to be invaluable and are highly recognized by different organizations because of its simplistic implementation of access control while providing an extra layer of system-wide security.

i. Access Control Models

Different computer systems make use of distinct access control models based on the concept of policy making. These models are based on the relationships between subjects (i.e., users or entities acting on behalf of the users), objects (i.e., protected resources) and permissions to control access. Listed below are the three access models that commonly occur in computer systems:

1. *Mandatory Access Control* (MAC) is a policy that deals with security labels attached to subjects and objects. Access is granted based on these labels which are mandated by the system, hence the term mandatory.
2. *Discretionary Access Control* (DAC) refers to a policy where access is determined by the owner of an object [67].
3. *Role-Based Access Control* (RBAC) is an emerging policy where system administrators create roles (which are given specific system permissions) mirroring

organizational job functions, with users being assigned to a role based on their qualifications and responsibilities [68].

The first two access models, MAC and DAC, have requirements that best fit military information specifications. We often hear the terms Top Secret, Secret, Confidential and Unclassified when referring to military documents. With MAC, a document marked with a Top Secret classification is only accessible to higher-ranking officials. Similarly, with DAC, the owner of a document is capable of selecting specific people to share the information with. Outside of military systems, their applications are quite rare [69].

The RBAC model was the much needed solution in meeting the requirements of many commercial and government organizations [70]. Through the use of roles, the RBAC model can apply techniques innate to MAC and DAC, and this flexibility cemented its importance to almost any computer system. Basically, instead of subjects being given access permissions (DAC approach), subjects are assigned to roles that were given limiting specifications beforehand. A role is a job function that describes the authority and responsibility assigned with a particular working activity, and this concept would be the basis of RBAC in determining system access. Imagine a user that have the credentials to be assigned to a role within the system, say the role of the doctor, and being assigned to that specific role grants you the privilege to prescribe (read and write) medication. Meanwhile, a pharmacist in the same system has only the permission to dispense medication based on a doctor's prescription. System roles replace the need to

assign permissions individually with role-based assignments depending on current organizational needs.

Many institutions like the University of the Philippines Manila have a defined organizational structure that separates the duties and responsibilities of its community. Librarians, professors, department heads and students have different roles within the system. With RBAC, these roles can be transferred to a system owned by an institution. In an institutional repository, librarians would best fit the role of system administrators because the role demands the skill and competency of librarians.

F. Web 2.0

The second incarnation of the Web, Web 2.0, brought changes on how we use the internet technology. With Web 2.0, users can easily generate and publish content than it was during the initial stages of the Web [71]. As we integrate computers into our lives, the technology became more mature to the point that it now allows user-to-user communication. An advantage of this is that it gives users a familiar feel (interaction-wise) towards the virtual community. It also facilitates online activities that require a human-centric approach, where the collective intelligence of users is encouraged. Given the tools necessary for communication, people tend to instinctively participate in social activities and cooperate with each other [72], and this was the case with the emergence of Web 2.0. The Internet, as of today, hosts an incredible amount of applications built towards creating a social network in which users can now view site changes in real-time, send site feedbacks and rate web contents. Combining these elements, we are now

witnessing a myriad of social networking systems lead by services such as Facebook and Twitter.

A good example of a Web 2.0 information system is the Wikipedia. Wikipedia is a web-based encyclopedia where people can have discussions on a separate page while they collaborate on the actual content of an article [73]. Institutional repository implementations can benefit from a design like Wikipedia's. A well-implemented system that supports socializing will make a passive system obsolete. Social functionalities provide more interaction and more learning in the process.

i. Communities

Communities are logical representations for labs, centers, schools, or departments belonging to an institution. Think of communities as separate entities that make up an institutional repository. Each community is designed to be a knowledge space of related information usually deposited by authors belonging to that community, and this nature provides convenience to resource discovery (searching and browsing). Through access control systems, communities can be moderated by community administrators which are responsible for customizing the content, rules, and regulations of their assignments.

IV. Design and Implementation

A. Overview

The University of the Philippines Manila – Institutional Repository is an information system that provides archiving services to digital institutional assets. It serves as a knowledge space that features interactive commenting and annotating.

i. Site Registration

Registration for an account is done off-site. Interested applicants are required to visit the University Librarian and present a valid university ID. Students are required to register an account as part of the enrollment process. Individuals outside of the university are required to submit their picture and fill up an information sheet. They also need to agree with the terms and conditions of the institution. Depending on the need to acquire an account, users can be given a permanent or a timed renewable account. Timed renewable accounts provide access that can last for months and years, in accordance to university regulations.

ii. Material Workflow

Digital materials can be deposited by System Administrators, Community Moderators, and University Users. The UPM-IR website guides the user through a series of steps that will require providing specific details that would enhance the material's searchability. The

website also provides the option to apply embargoes on the materials. Deposited materials require the approval of either System Administrators or Community Moderators before they can be indexed and become publicly available.

There is a limitation on what kinds of digital materials are deposited in the system. Digital materials to be deposited in the repository are defined as products of the institution and are of scholarly work. These include materials such as books and theses in their electronic forms. Additionally, scholarly publications and journals are also desired. Materials that are deemed inappropriate, irrelevant, and personal will be rejected.

Besides from the materials deposited to the system, the repository also gets its data from university processes. The repository also accepts material donations from members of the institution. Graduating students are required to submit a soft copy of their theses and these works are added to the repository for public viewing.

Once a digital material is available for viewing, all users will have access to it. Depending on your access level to the repository, some files are only available for viewing while others can be downloaded. Users who own the digital material have the capability to remove it. The administrators of the system also reserve the right to remove a material. Although it is possible to remove files from the repository, users can still access it upon downloading and storing it in a local computer.

iii. File Versioning

More often than not, digital materials deposited into the system will have updated versions of it being added to the system. Users can submit updates by viewing the history of the file they wish to update. Fortunately for the uninitiated, the update process is strongly similar to that of submitting a new material. Once a material has been submitted and approved by the administrator or moderator, the two files are now considered related, with the latest submission marked as the newest version as represented in the updated file history. The file history presents the user with different versions of a specific file – providing information on whether the file is the latest one, the initial one, or an intermediate between the two.

iv. Indexing and Searching

The repository handles the indexing and searching functionalities to digital materials. Indexing is done to improve the searchability of a file. The metadata is indexed by the system. Additionally, if the digital material is of the formats .txt and .pdf, their contents will also be indexed. Once the indexing is done, a publicly available material can now be searched by making use of Fedora's built-in search functionality.

v. Annotation

Experts and the owner of a digital material have the opportunity to annotate their works. Users annotate text documents by using the built-in file viewer with annotating capabilities. Annotations are done to describe and clarify digital material contents. Each annotation is subject to the approval of System Administrators and/or Community Moderators.

B. User Flow

i. Context Diagram

The UPM-IR is composed of four users: System Administrator, Community Moderator, University User, and Registered User. The System Administrator is responsible for managing user accounts and communities, or in general – managing the site in its entirety. To help with this task, communities are assigned to Community Moderators which are responsible for managing incoming data to their respective communities. University Users are able to deposit digital materials and provide information that would best describe the deposited data.

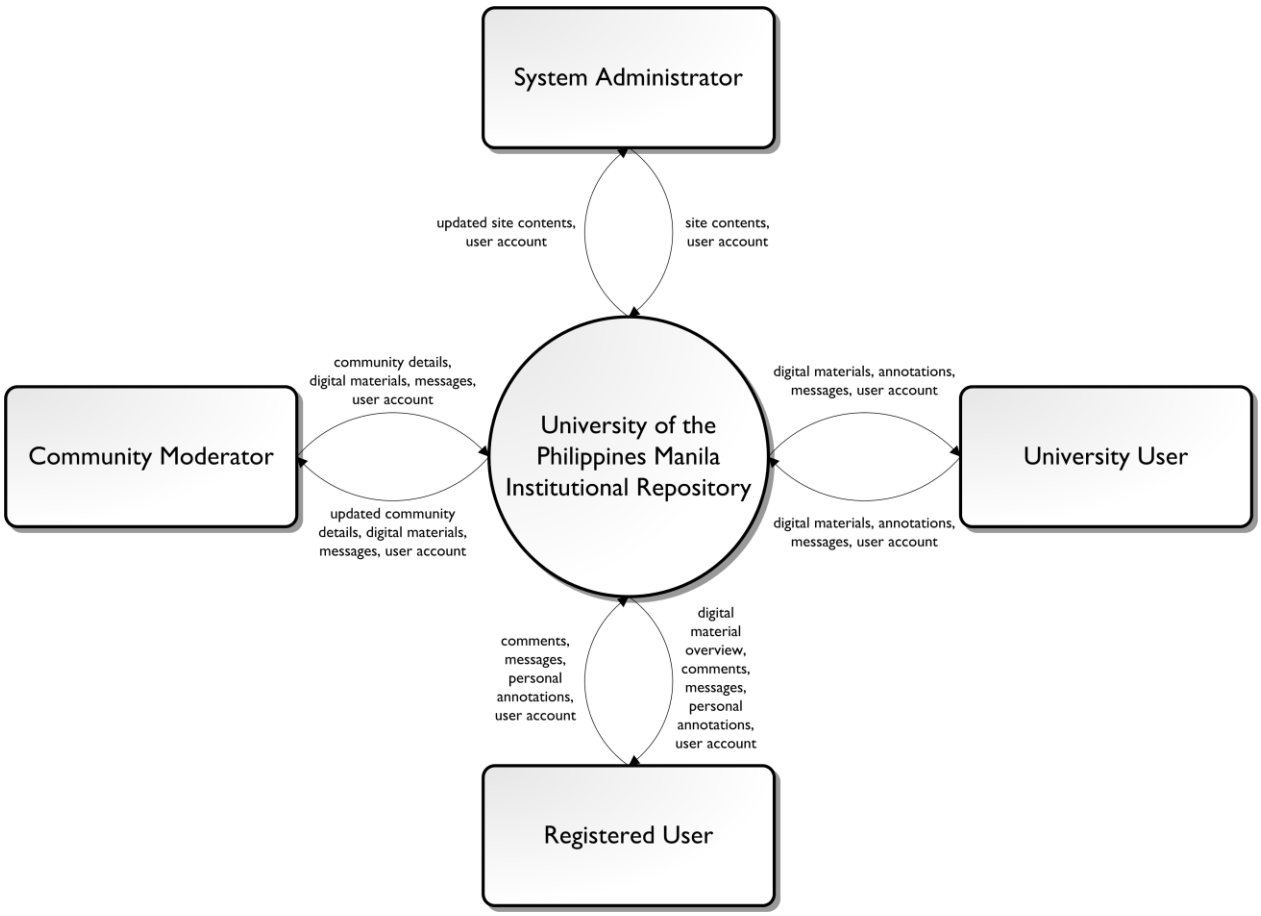

Figure 4.1. Context Diagram, University of the Philippines Manila – Institutional Repository

ii. Data Flow Diagram

Figure 4.2 shows the Top-Level Data Flow Diagram of the repository system. Users start by logging in the system with their appropriate accounts and a successful system verification grants access to the functionalities indicated below. Depending on the user type, some functionality may not be freely accessible such as the management of news which is exclusive to System Administrators.

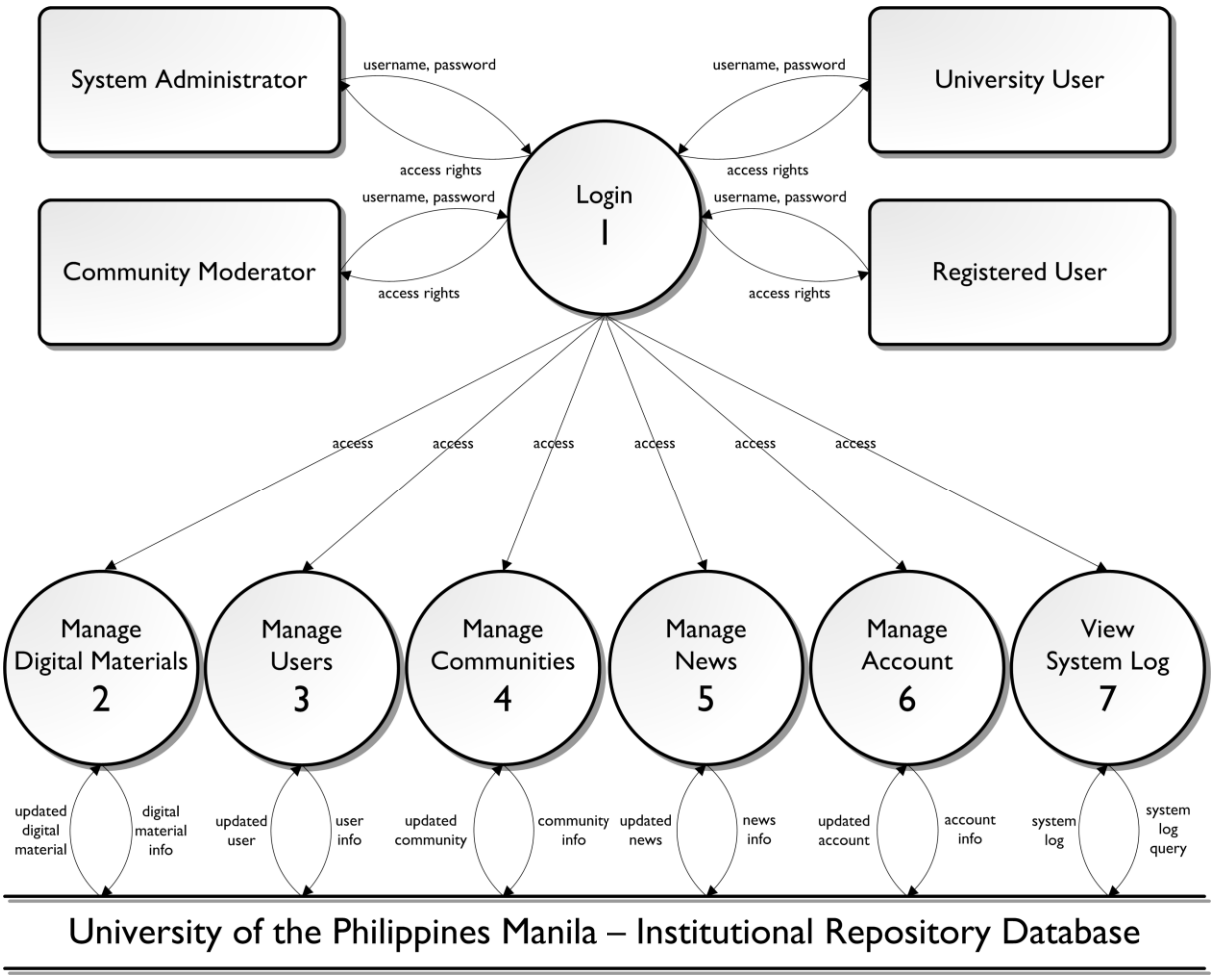

Figure 4.2. Top-Level Data Flow Diagram, University of the Philippines Manila – Institutional Repository

Access to the system is granted once a successful login is made. The user needs to provide his/her own username (e-mail) and password. The system will notify the user if the provided information matched the details stored in the database.

Figure 4.3. Subexplosion of the Process 1: Login, University of the Philippines Manila – Institutional Repository

System Administrators, Community Moderators, and University Users can submit a digital material to the repository. System Administrators and Community Moderators verify the submitted material. Verification ends once the System Administrator or the Community Moderator decides on whether the submitted material should be accepted or denied for repository publishing.

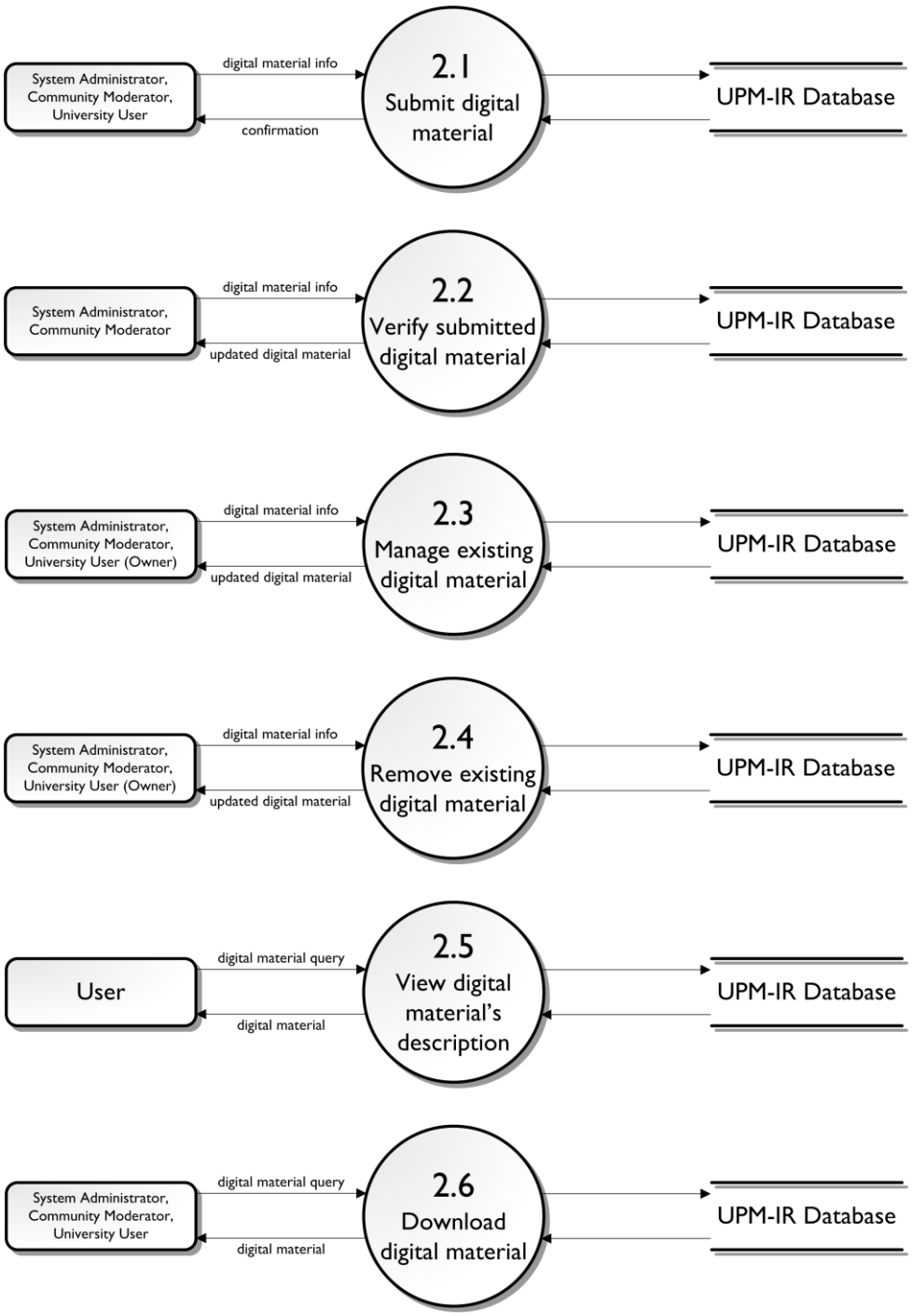

Figure 4.4. Subexplosion of the Process 2: Manage Digital Materials, University of the Philippines Manila – Institutional Repository

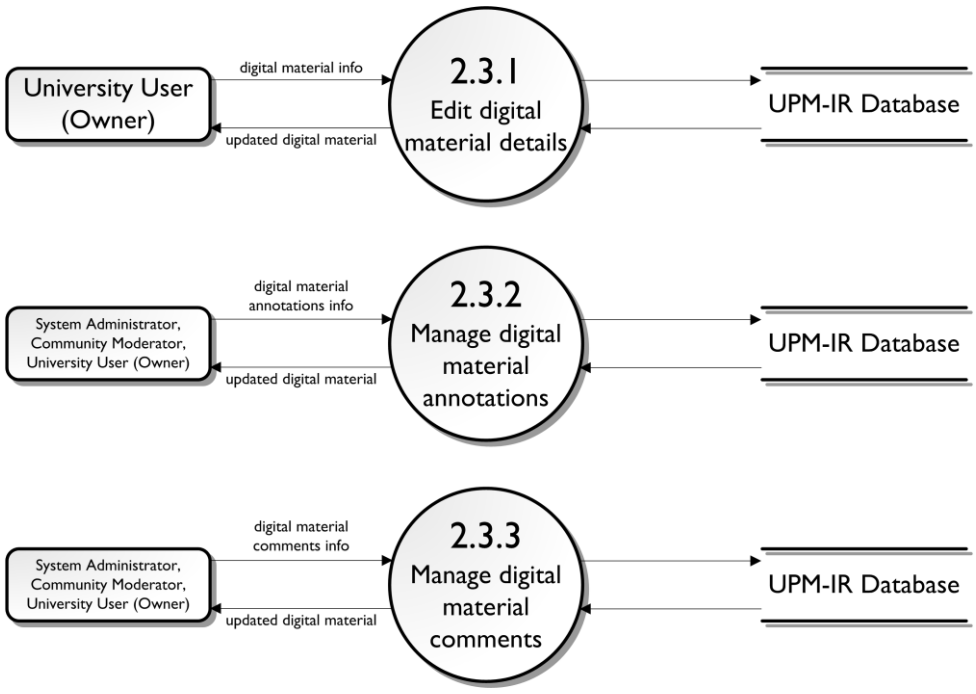

Figure 4.5. Subexplosion of the Process 2.3: Manage Existing Digital Material, University of the Philippines Manila – Institutional Repository

University Users can also add, edit, and delete their own annotations. Annotations can be either personal annotations or annotations that can be viewed by others. The latter ones are subject to the approval of the System Administrator or Community Moderator. Also, this feature is only available for text files.

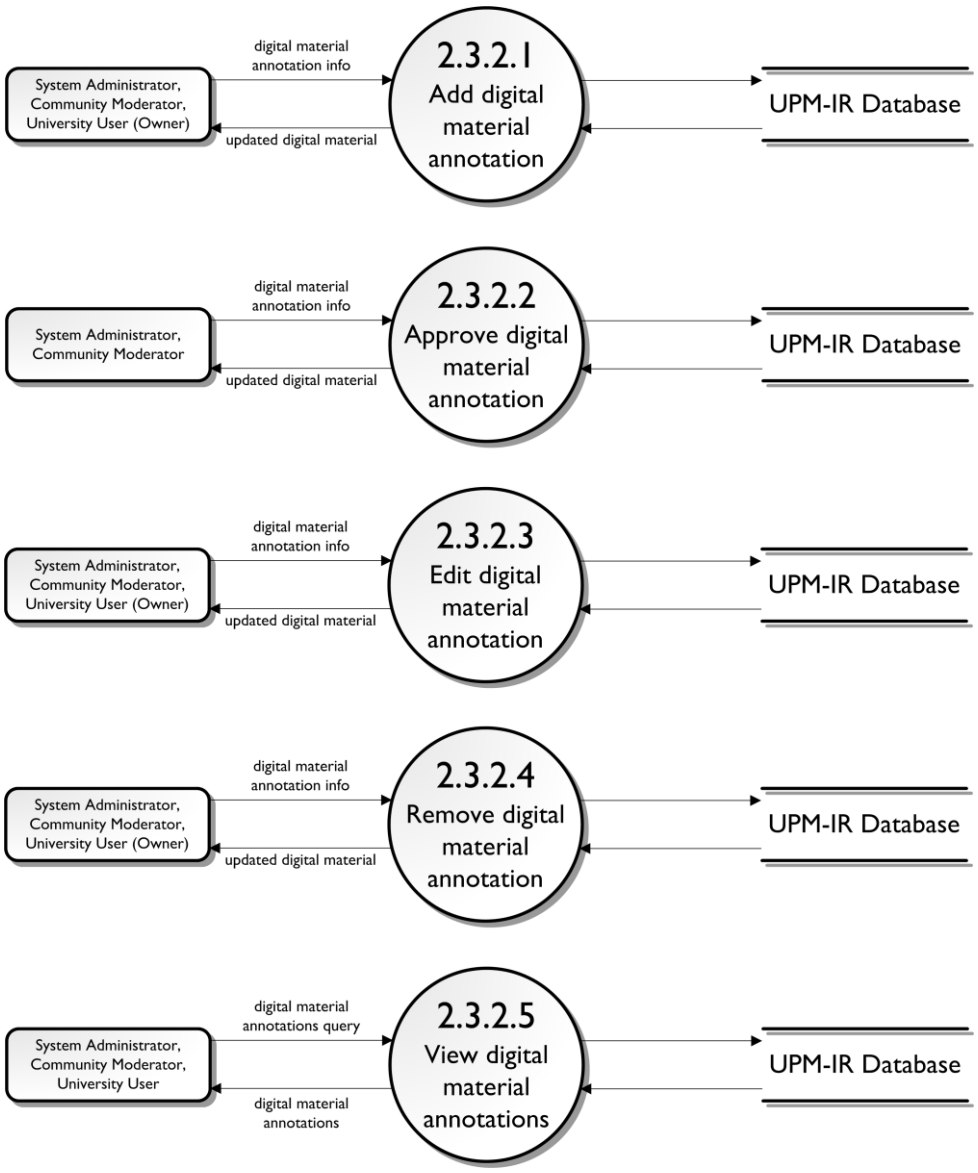

Figure 4.6. Subexplosion of the Process 2.3.2: Manage Digital Material Annotations, University of the Philippines Manila – Institutional Repository

A user can add and delete his/her own comment to any material deposited in the repository.

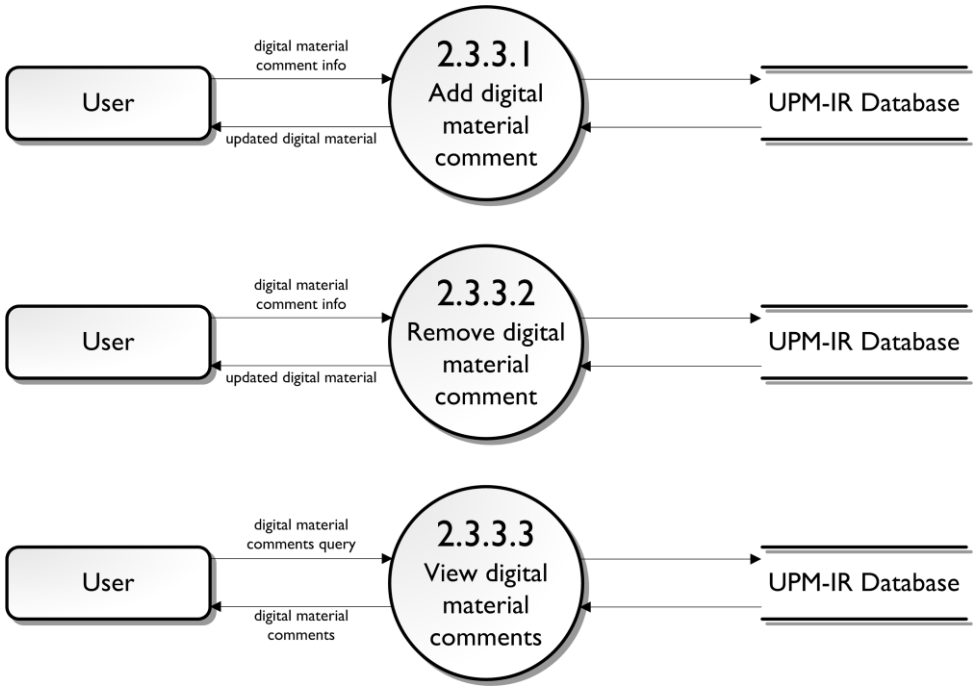

Figure 4.7. Subexplosion of the Process 2.3.3: Manage Digital Material Comments, University of the Philippines Manila – Institutional Repository

Users not registered to the system can apply for system registration by visiting the University librarian. System Administrators have the capability to be able to add users. As part of the System Administrators' responsibility, they can also change an existing user's account type and can revoke access rights to violating users.

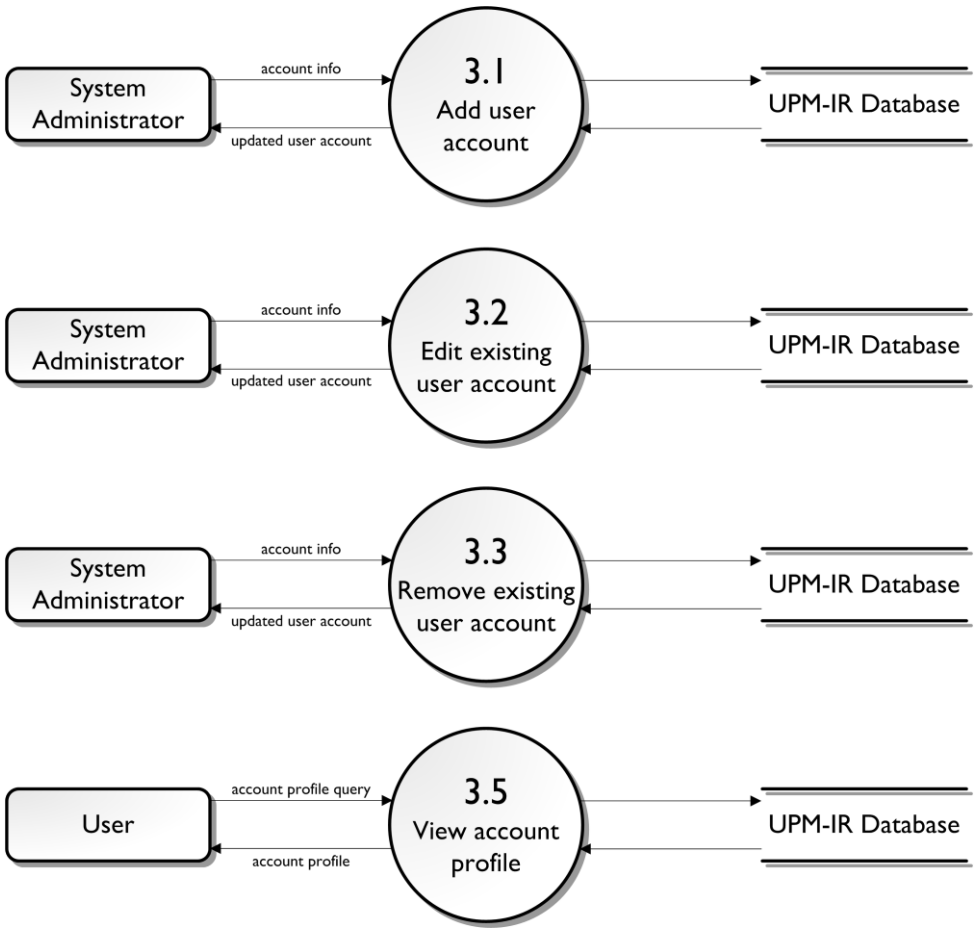

Figure 4.8. Subexplosion of the Process 3: Manage Users, University of the Philippines Manila – Institutional Repository

Communities are entities in the repository belonging to a unit in the UPM system. The System Administrator can create a community and set additional details (e.g., name) about it. With a community already created, the System Administrator can then assign Community Moderators as a step in separating duties within the institutional repository.

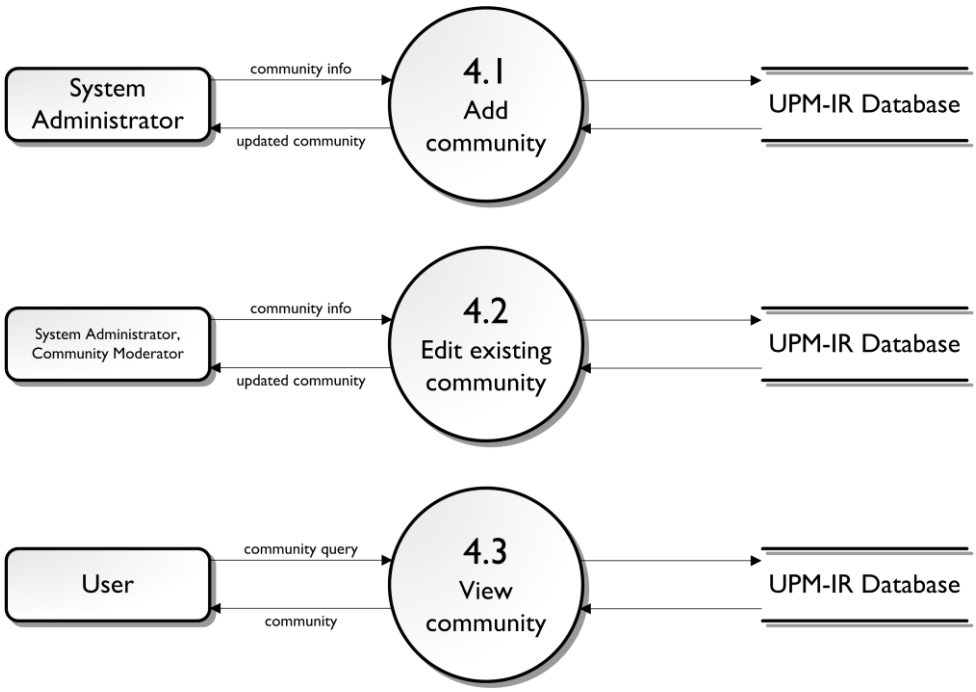

Figure 4.9. Subexplosion of the Process 4: Manage Communities, University of the Philippines Manila – Institutional Repository

System Administrators can provide repository news that will appear on the main page of the repository.

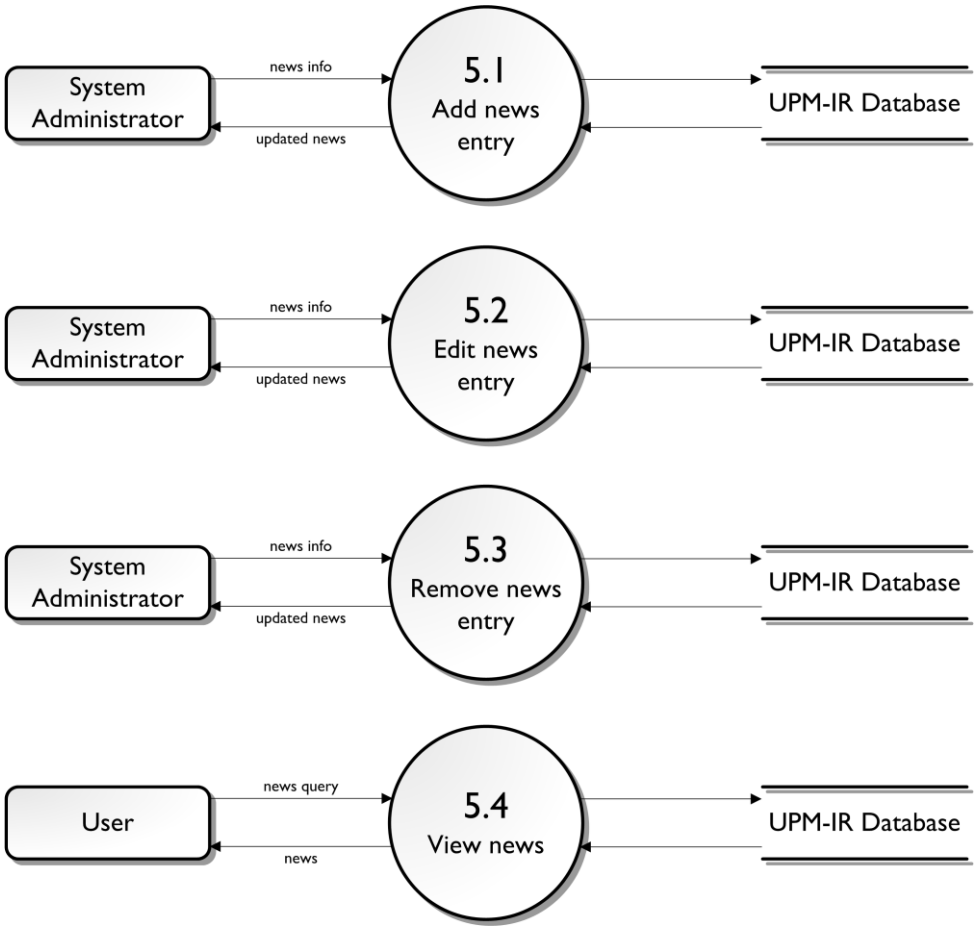

Figure 4.10. Subexplosion of the Process 5: Manage News, University of the Philippines Manila – Institutional Repository

Users registered to the system are privileged to be provided with their own "dashboard" where they can manage their own accounts. This includes managing account details such as changing one's password.

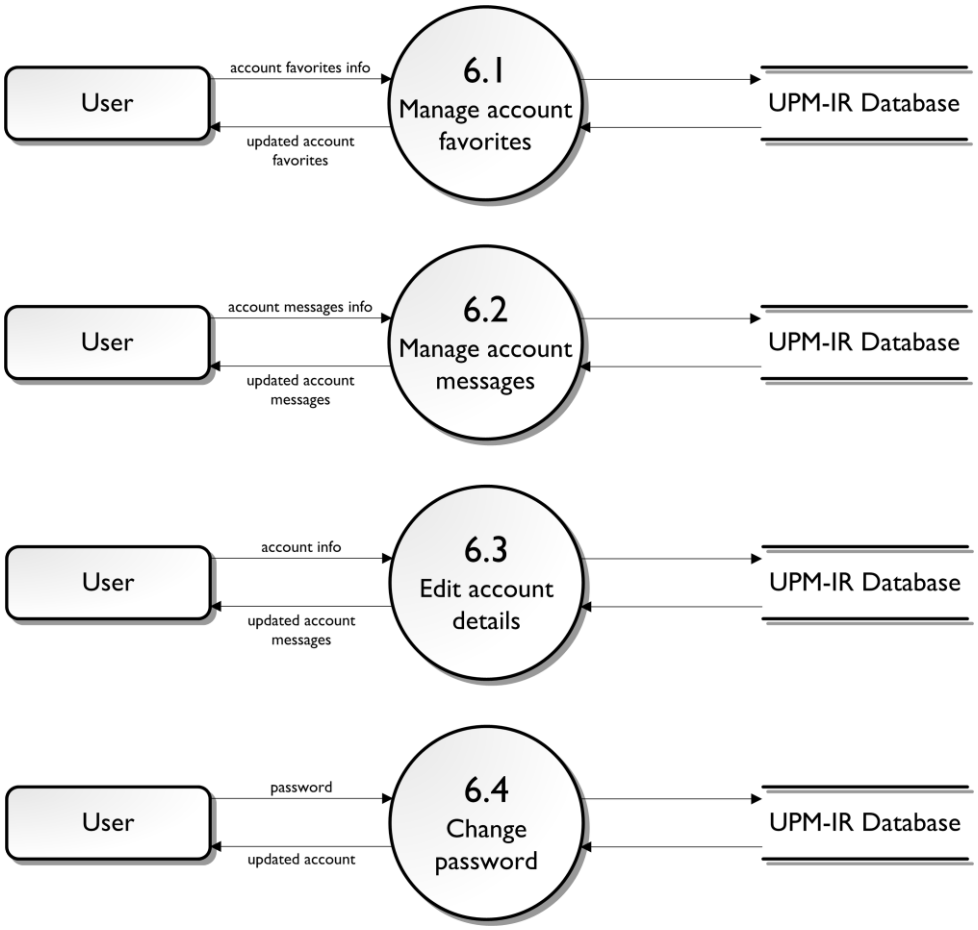

Figure 4.11. Subexplosion of the Process 6: Manage Account, University of the Philippines Manila – Institutional Repository

Digital materials published online can be set as a favorite by a user. The material is added to a user's favorites list which contains links redirecting to a favorite material for easy access later on. Users can describe an item in their favorites list and edit it anytime they like.

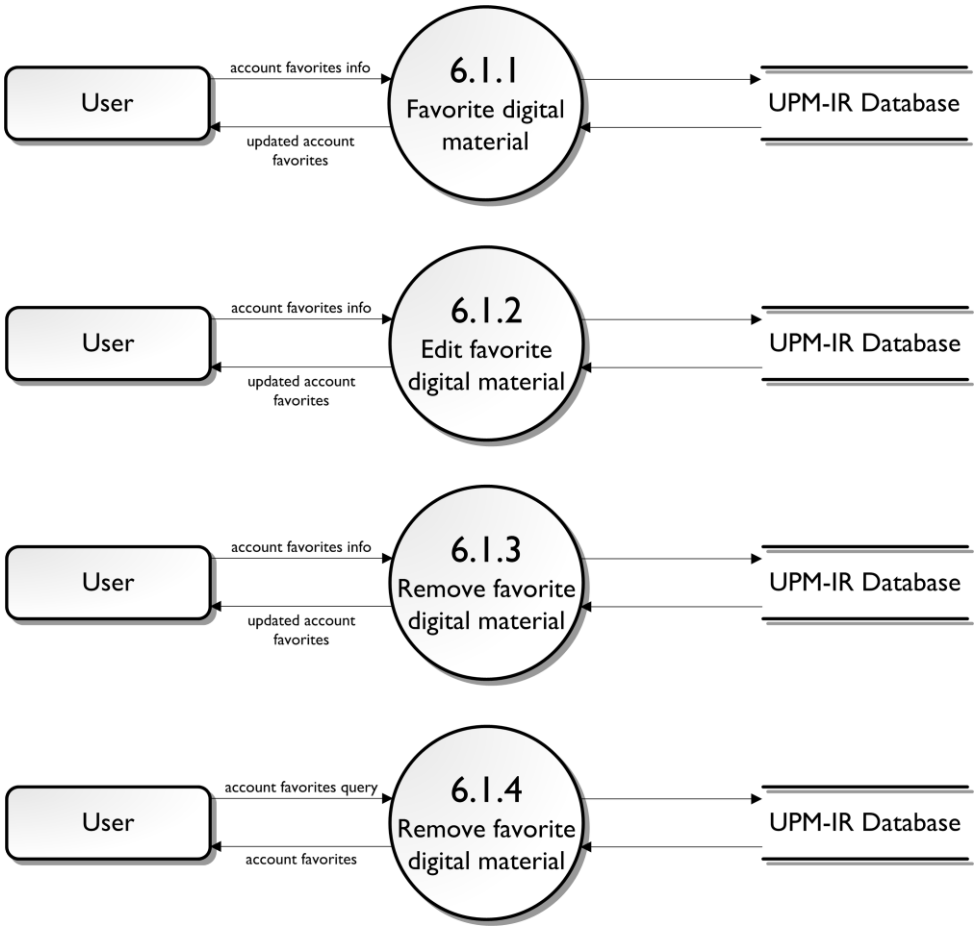

Figure 4.12. Subexplosion of the Process 6.1: Manage Account Favorites, University of the Philippines Manila – Institutional Repository

As an additional functionality to the repository, users are able to compose, send and receive messages sent by other users within the system.

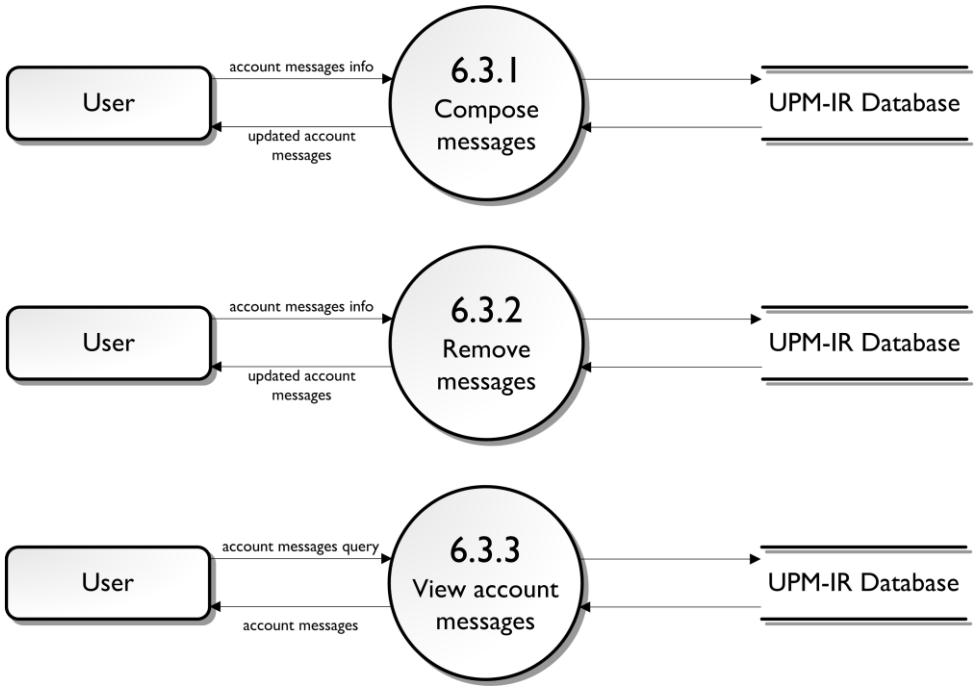

Figure 4.13. Subexplosion of the Process 6.3: Manage Account Messages, University of the Philippines Manila – Institutional Repository

Finally, a System Administration can view all operations that happen within the repository.

Figure 4.14. Subexplosion of the Process 7.1: View System Log, University of the Philippines Manila – Institutional Repository

C. Database Schema Design

i. Entity Relationship Diagrams

The UPM-IR has nine entities: annotation, comment, community, digital material, favorite, message, news, operation and user (System Administrator, Community Moderator, University User, and Registered User). As stated above, users are able to manage their own favorites lists. Figure 4.15 below shows communities being composed of digital materials which have their own comments and annotations.

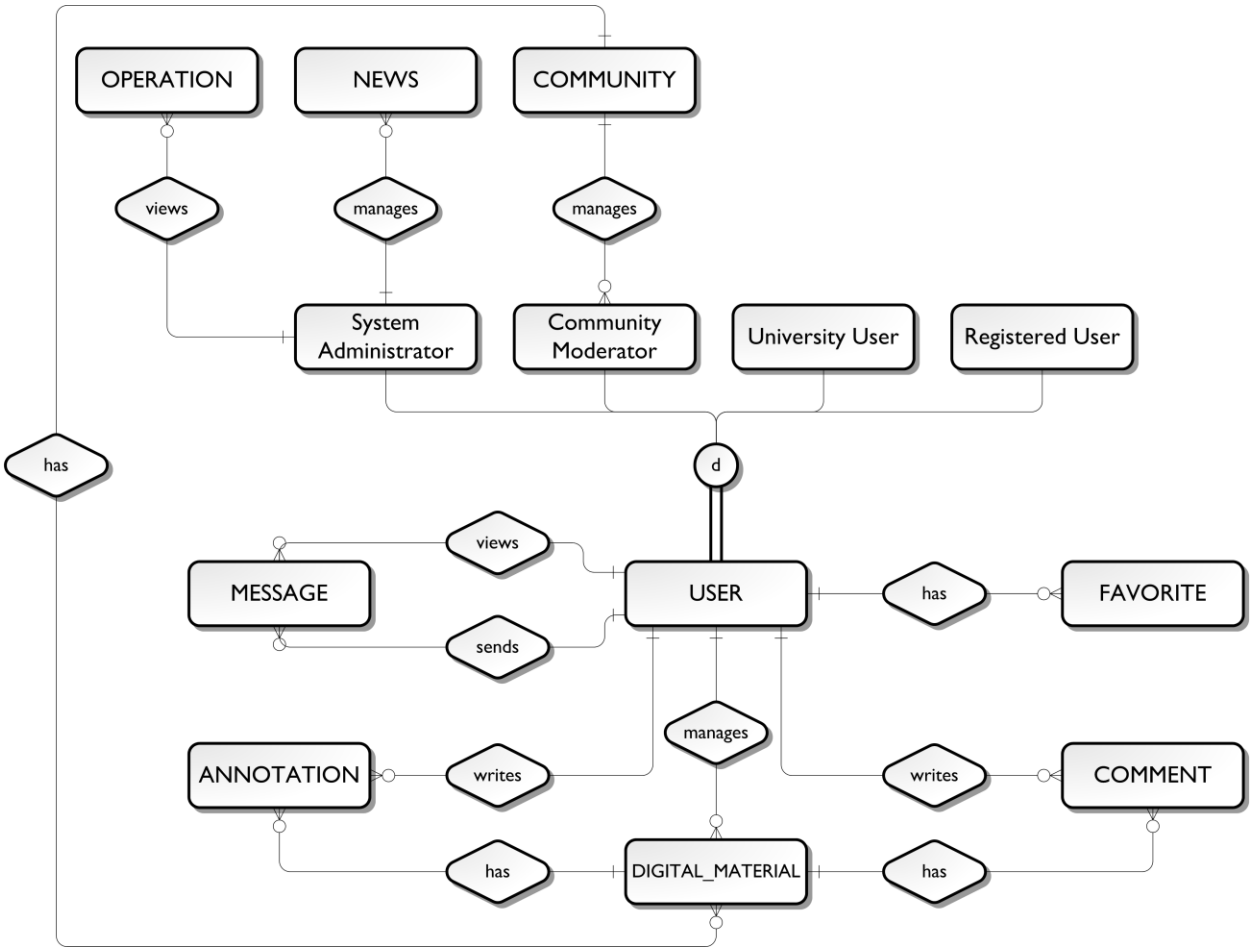

Figure 4.15. Entity Relationship Diagram, University of the Philippines Manila – Institutional Repository

The annotation entity makes use of two distinct fields: `start_location` and `end_location`, which are both utilized by the built-in file viewer with annotating capabilities. These fields serve as storage for the file viewer to remember what text is highlighted by the user.

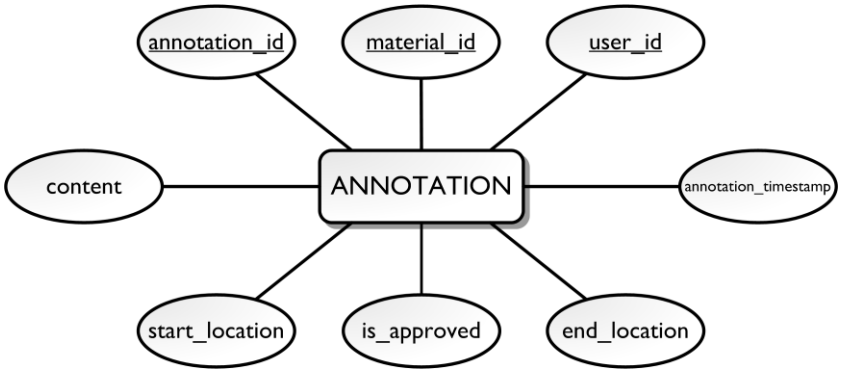

Figure 4.16. Entity Relationship Diagram of `annotation`

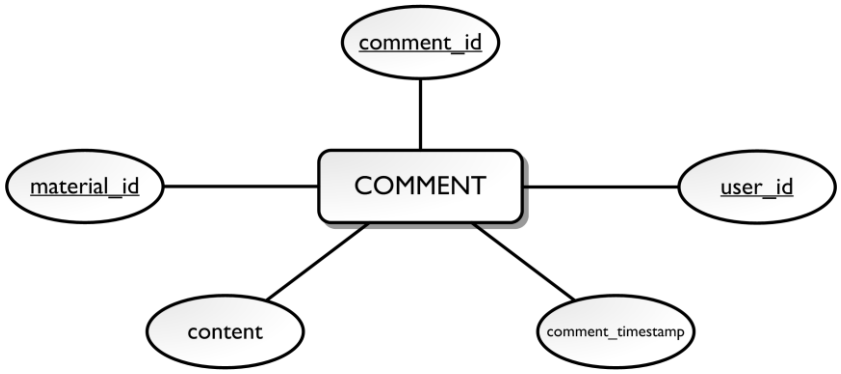

Figure 4.17. Entity Relationship Diagram of `comment`

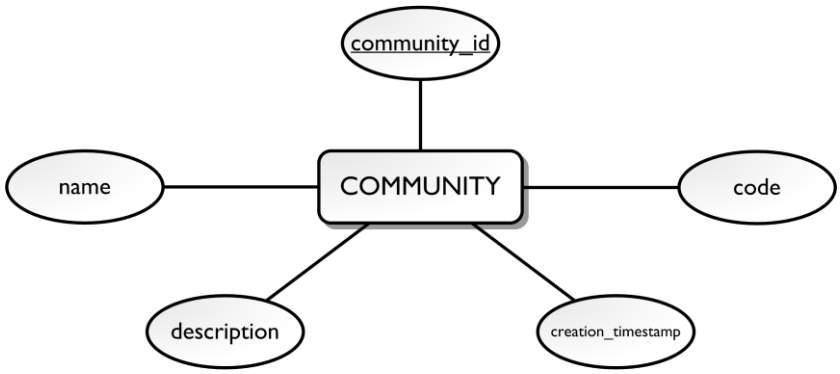

Figure 4.18. Entity Relationship Diagram of `community`

The `digital_material` entity is the most extensive entity in the repository due to its crucial importance to every repository system. Digital materials submitted by users are stored in the database after approval (or without, as mentioned in this section) by System Administrators or Community Moderators. Like an item in a user's favorites list, digital materials can also be tagged as an attempt to enhance its information quality. Material versioning is provided through the `history` field, which links a material to related materials existing in the database. As stated in the previous section, the UPM-IR makes use of the Dublin Core schema for its metadata fields.

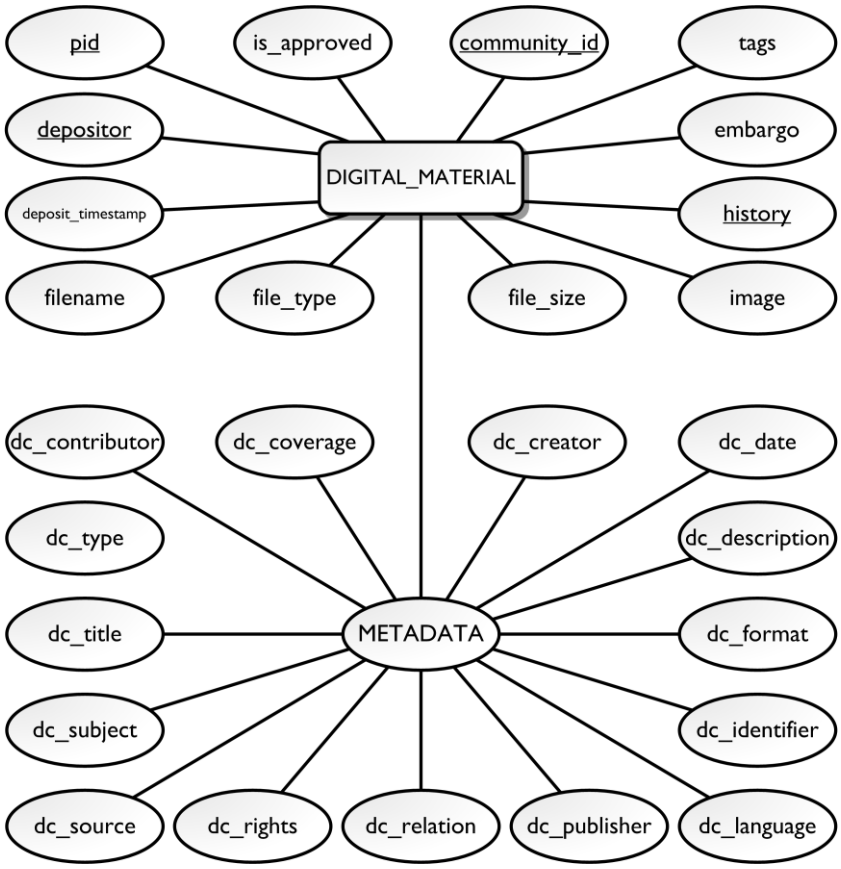

Figure 4.19. Entity Relationship Diagram of `digital_material`

Favorite links can be described for personal reference. Tags can also be added in addition to favorites description for search and browsing purposes.

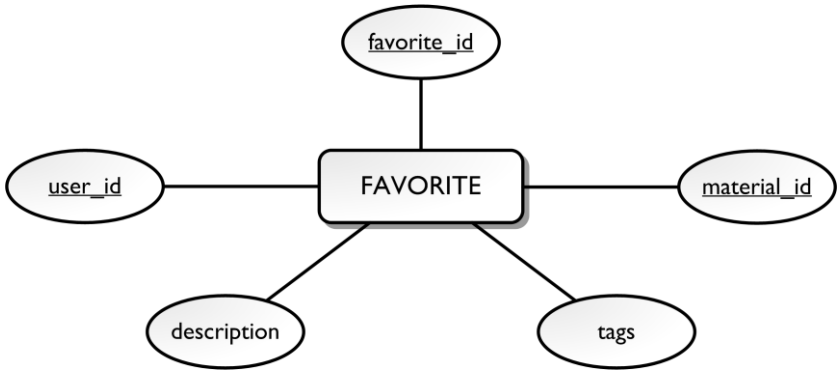

Figure 4.20. Entity Relationship Diagram of `favorite`

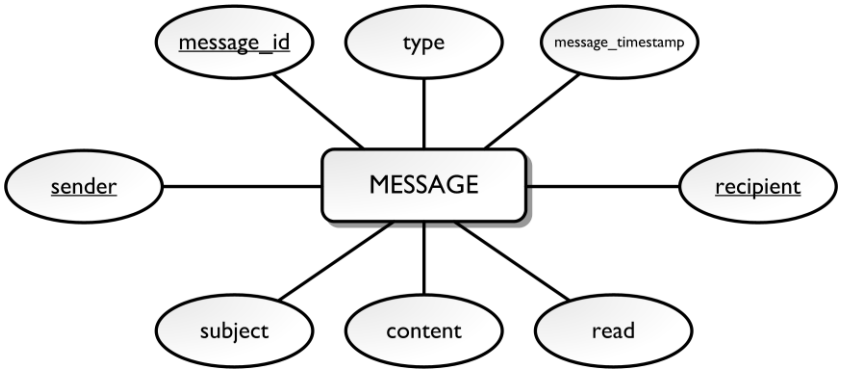

Figure 4.21. Entity Relationship Diagram of `message`

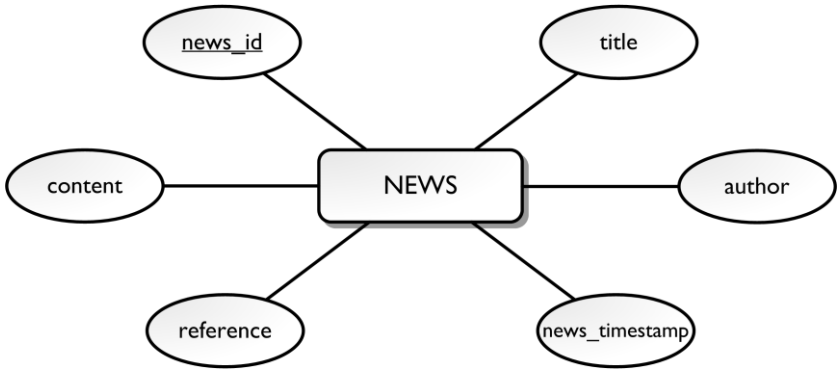

Figure 4.22. Entity Relationship Diagram of `news`

Operations made within the repository are stored in the `operation` table. These operations can be viewed by System Administrators.

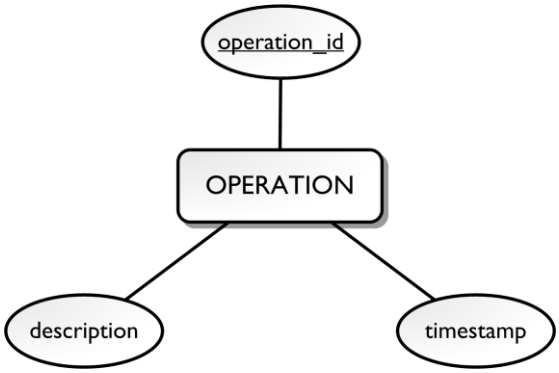

Figure 4.23. Entity Relationship Diagram of `operation`

In the institutional repository, every user is identified through their real names. Members of the University of the Philippines Manila are encouraged to apply as a University User to be able to contribute and retrieve materials. Users can also provide a description of themselves such as stating their occupation and contact number/s.

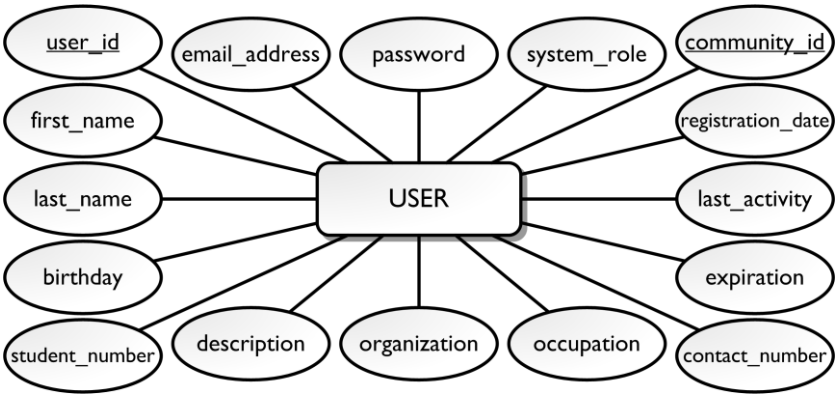

Figure 4.24. Entity Relationship Diagram of `user`

ii. Data Dictionary

Field Name	Data Type	Description
annotation_id	bigint(20)	unique annotation ID (Primary Key)
is_approved	enum('y','n')	single character identifier (y or n) to determine annotation approval
content	varchar(5000)	description of annotation
start_location	int(11)	start highlighting at this location
end_location	int(11)	end highlighting at this location
user_id	bigint(20)	user who created annotation (Foreign Key: user_id from user)
material_id	varchar(25)	ID of digital material (Foreign Key: pid from digital_material)
annotation_timestamp	timestamp	date annotation was created

Table 4.1. annotation

Field Name	Data Type	Description
comment_id	bigint(20)	unique comment ID (Primary Key)
content	varchar(500)	digital material comment
depositor	bigint(20)	user who created comment (Foreign Key: user_id from user)
material_id	varchar(25)	ID of digital material (Foreign Key: pid from digital_material)
comment_timestamp	timestamp	date comment was created

Table 4.2. comment

Field Name	Data Type	Description
community_id	bigint(20)	unique community ID (Primary Key)
code	varchar(10)	community code (usually an acronym for the name of community)
name	varchar(200)	name of community
description	varchar(5000)	community description
creation_timestamp	timestamp	date community was created

Table 4.3. community

Field Name	Data Type	Description
pid	varchar(25)	unique digital material ID (Primary Key)
is_approved	enum('y','n')	single character identifier (y or n) to determine material approval
history	varchar(25)	parent material (keeps track of updates)
dc_contributor	varchar(200)	*
dc_coverage	varchar(200)	*
dc_creator	varchar(200)	*
dc_date	date	*
dc_description	varchar(5000)	*
dc_format	varchar(200)	*
dc_identifier	varchar(200)	*
dc_language	varchar(200)	*
dc_publisher	varchar(200)	*
dc_relation	varchar(200)	*
dc_rights	varchar(200)	*
dc_source	varchar(200)	*
dc_subject	varchar(200)	*
dc_title	varchar(200)	*
dc_type	varchar(200)	*
community_id	bigint(20)	ID of community a material belongs (Foreign Key: community_id from community)
tags	varchar(5000)	user-defined tags to describe material
embargo	date	date the digital material will be publicly and fully available
depositor	bigint(20)	user that deposited the material

		(Foreign Key: user_id from user)
deposit_timestamp	timestamp	date digital material was deposited
filename	varchar(200)	file name
file_type	varchar(200)	file type
image	varchar(200)	reference to image (address)
file_size	bigint(20)	file size in bytes

Table 4.4. digital_material

**descriptions about these fields are provided in the previous section, under the Dublin Core heading*

Field Name	Data Type	Description
favorite_id	bigint(20)	unique favorite ID (Primary Key)
description	varchar(5000)	description of the favorite item
tags	varchar(5000)	tags which the user wants to associate it with
material_id	varchar(25)	ID of digital material (Foreign Key: pid from digital_material)
user_id	bigint(20)	owner (Foreign Key: user_id from user)

Table 4.5. favorite

Field Name	Data Type	Description
message_id	bigint(20)	unique message ID (Primary Key)
read	enum('y','n')	message is either read or unread
subject	varchar(200)	subject of the message
content	varchar(5000)	Message
sender	bigint(20)	sender of the message (Foreign Key: user_id from user)
recipient	bigint(20)	recipient of the message (Foreign Key: user_id from user)

type	varchar(50)	*type of message:* ● user ● support (inquiries - automatically sent to the system) ● flag (simple message to flag inappropriate materials - automatically sent to system administrators and applicable community moderators)
message_timestamp	timestamp	date message was sent

Table 4.6. message

Field Name	Data Type	Description
news_id	bigint(20)	unique news ID (Primary Key)
title	varchar(200)	title of the news
reference	varchar(210)	unique identifier to quickly reference news
content	text	description of the news
author	bigint(20)	author of news (Foreign Key: user_id from user)
news_timestamp	timestamp	date news was created

Table 4.7. news

Field Name	Data Type	Description
operation_id	bigint(20)	unique operation ID (Primary Key)
description	varchar(5000)	description of operation
timestamp	timestamp	date operation was made

Table 4.8. operation

Field Name	Data Type	Description
user_id	bigint(20)	unique user ID (Primary Key)
email_address	varchar(255)	user's e-mail (used for login)
password	varchar(32)	user's password
first_name	varchar(50)	user's first name
last_name	varchar(50)	user's last name
birthday	date	user’s date of birth

description	varchar(500)	self-described info about the user
occupation	varchar(100)	user's occupation
organization	varchar(100)	user's affiliated organization
contact_number	varchar(30)	user's contact number
system_role	varchar(30)	type of user
community_id	bigint(20)	required if user is a Community Moderator (Foreign Key: community_id from community)
student_number	varchar(10)	required if user is a University User (for identification purposes)
registration_date	timestamp	date user registered
expiration	timestamp	set account expiration
last_activity	timestamp	date user last seen active

Table 4.9. user

D. Fedora

The Fedora framework provides exceptional ready-to-use repositorial services. Some of the key functionalities of Fedora are:

1. *Data Collection* – Initially, the submitted material will be collected by Fedora and binds the associated metadata to it. It then creates an object from it which will be stored in the repository. The object's ID will be associated to a persistent identifier which will be used to refer to the stored object regardless of its location. Finally, the system informs the authorized users (administrator and depositors by default) of either the success or failure of the collection process.
2. *Flexibility* – Fedora offers robust flexibility to a repositorial project especially when creating a user interface for it. The availability of two easy to use APIs makes it more appealing to organizations in need of a powerful repositorial backend.

i. API

To be able to deposit and retrieve digital materials and its information, the institutional repository system makes use of Fedora. Data is exposed by making use of Fedora's built-in APIs and accessing those through HTTP methods with the help of a Java client. The repository system utilizes two Fedora APIs – each with different sets of functionalities:

1. *API-A* – This service defines the interface for accessing materials.
2. *API-M* – This service defines the interface for material management. This includes the editing and deletion of files.

E. Technical Architecture

The UPM-IR system has a client-server type of architecture as illustrated in Figure 4.23. All deposited materials are to be stored in the database.

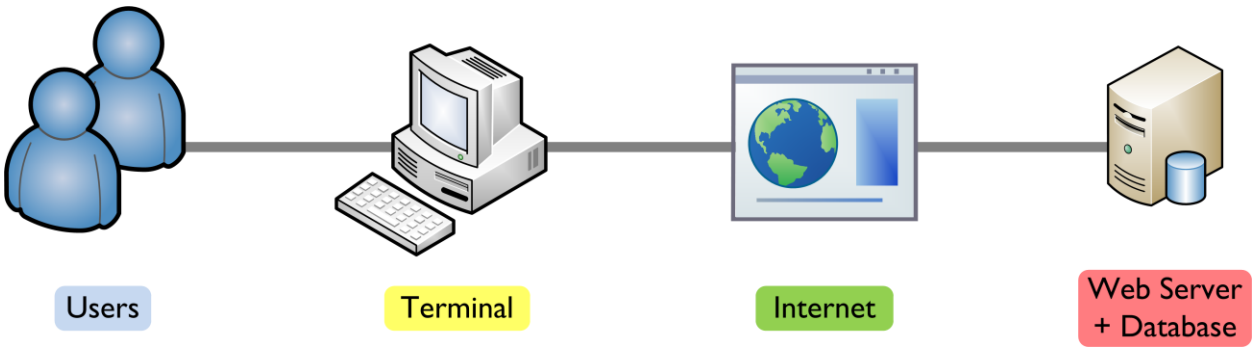

Figure 4.23. Technical Architecture, University of the Philippines Manila – Institutional Repository

V. Results

Without logging in, the UPM-IR homepage screen (Figure 5.1) mainly displays recently submitted materials and recently posted news. On the upper right hand corner is a link which leads to the login screen. Alternatively, visiting a restricted area (e.g. user settings page) of the website will automatically redirect the user to the login screen.

Figure 5.1. Homepage

The login screen (Figure 5.2) simply features two input boxes, one for the user's e-mail address and another for the corresponding password. Input of invalid information (non-existing account, wrong password) will lead to a slightly modified version of the login screen (Figure 5.3) which alerts the user that an error has occurred. On the other hand, a successful login will lead back to the homepage.

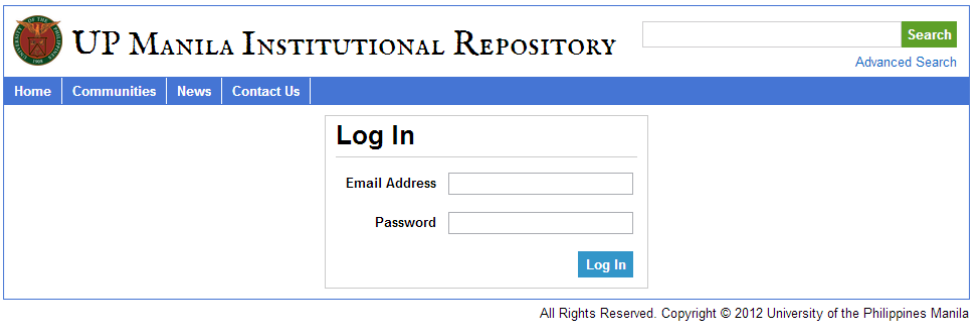

Figure 5.2. Login Page

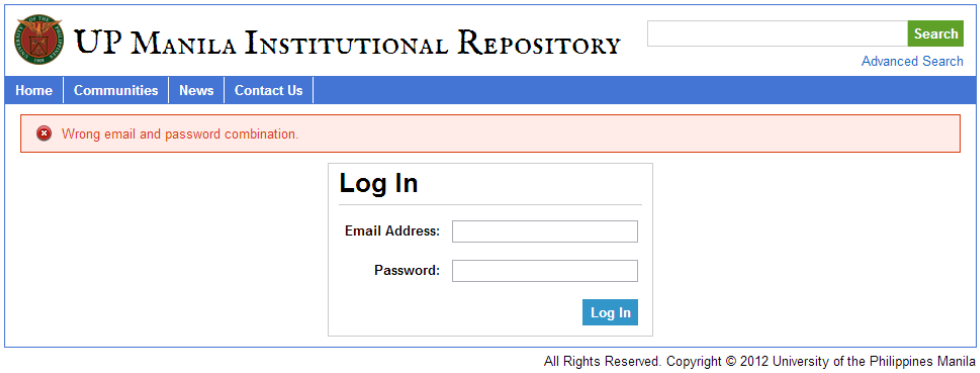

Figure 5.3. Login Page (Wrong email and password combination)

Logged in, the homepage is slightly changed to display account management functionalities on the top right corner of the page and depending on the user's system role, the blue navigation bar under the repository title banner may contain links to important parts of the website. The homepage perspective of the Registered User and the University User are described in Figures 5.4 and 5.5 respectively. Figures 5.6 and 5.7 are shown from the perspective of the System Administrator and a Community Moderator.

Figure 5.4. Homepage (Registered User Perspective)

Figure 5.5. Homepage (University User Perspective)

Figure 5.6. Homepage (System Administrator Perspective)

Figure 5.7. Homepage (Community Moderator Perspective)

For a quick look of the repository's basic features, we first explore the website from the perspective of the Registered User. Clicking Settings on the upper right corner will send the user to the user settings page (Figure 5.8) where he/she can edit personal information.

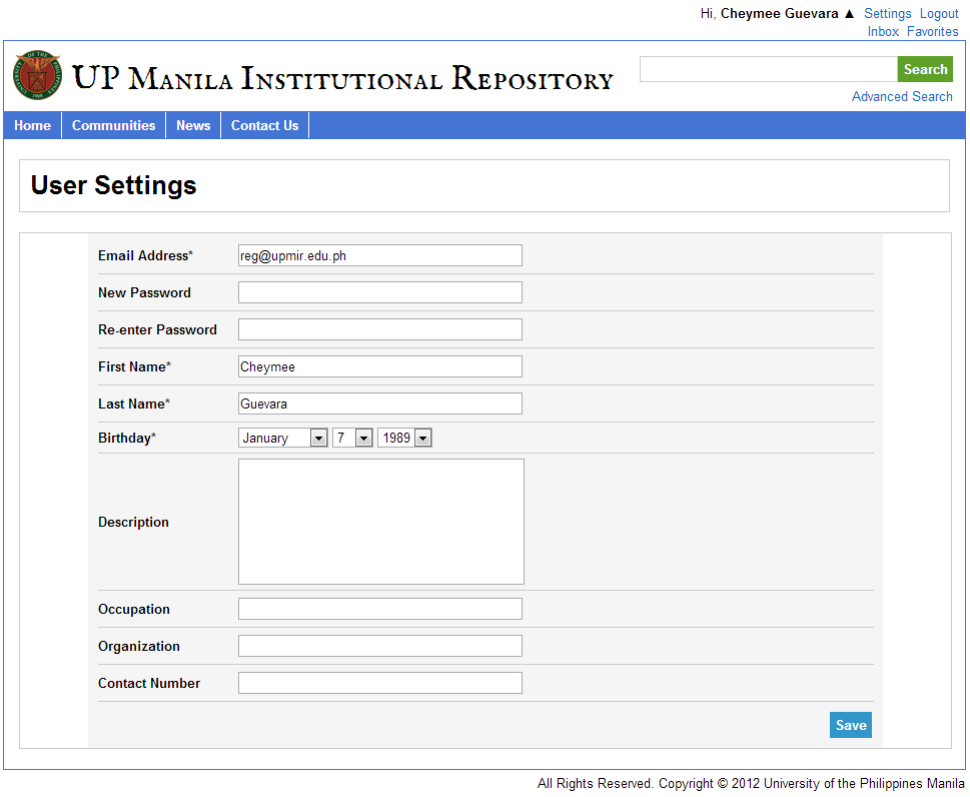

Figure 5.8. User Settings Page

From the homepage, the 15 recently deposited and approved materials are displayed. Clicking one of them will display the material page. The material page (Figure 5.9) contains information about the file (e.g. title, description), an image thumbnail if applicable, options for reporting a material and the ability to save the page as a favorite, two links that allows the user to download the file, and a comment section at the bottom. Every comment you post can be deleted by yourself, a moderator, or an administrator.

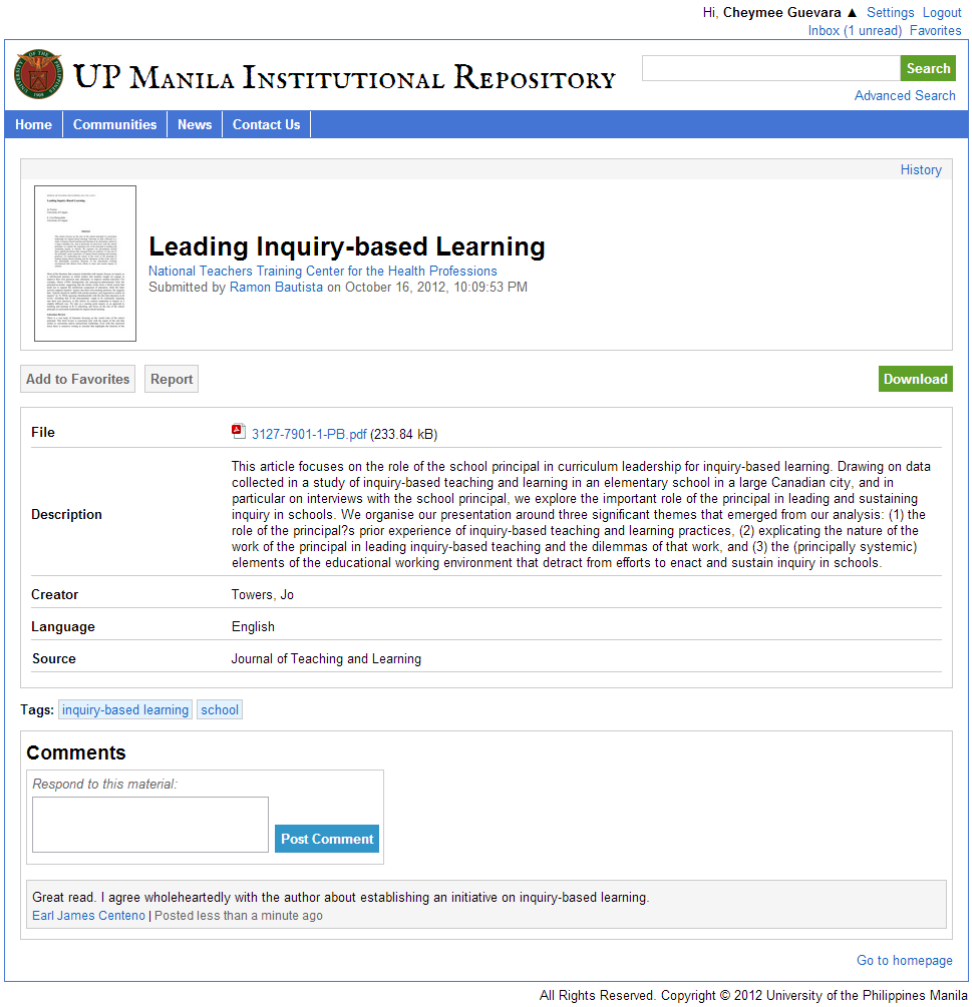

Figure 5.9. Material Page

Clicking the Add to Favorites button displays a pop-up form (Figure 5.10) which asks the user to make a note about the material (optional) before saving it as a favorite. Making a material as a favorite is one way of quickly accessing a material later on.

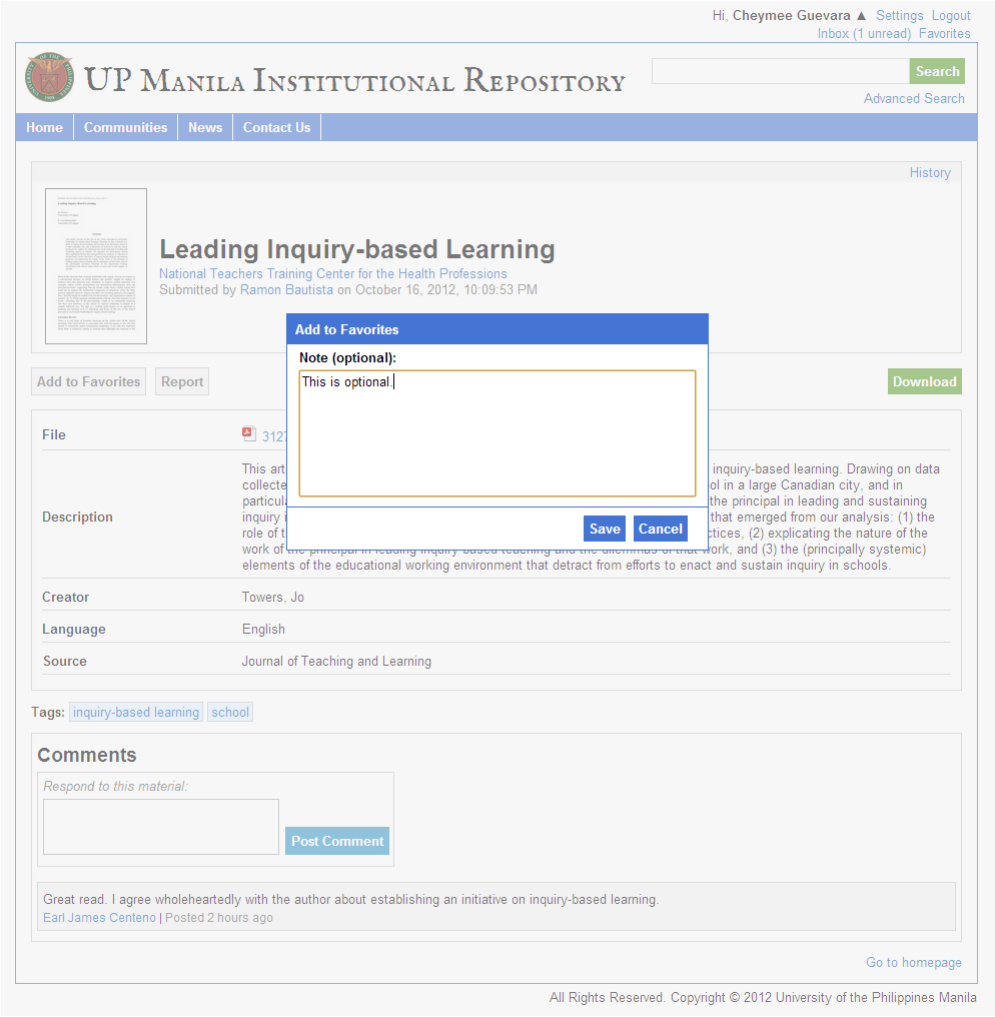

Figure 5.10. Add to Favorites Form

After adding a material as a favorite, the Add to Favorites button will be replaced by the Remove from Favorites button. Clicking this button will remove the material from your favorites list as well as the note that goes with it.

In the same way as adding a material as a favorite, clicking Report will make a pop-up form appear (Figure 5.11). This instance requires the user to input a reason for the report. Once the user is done filling out the form, the report will then be sent to every System Administrator and responsible Community Moderators.

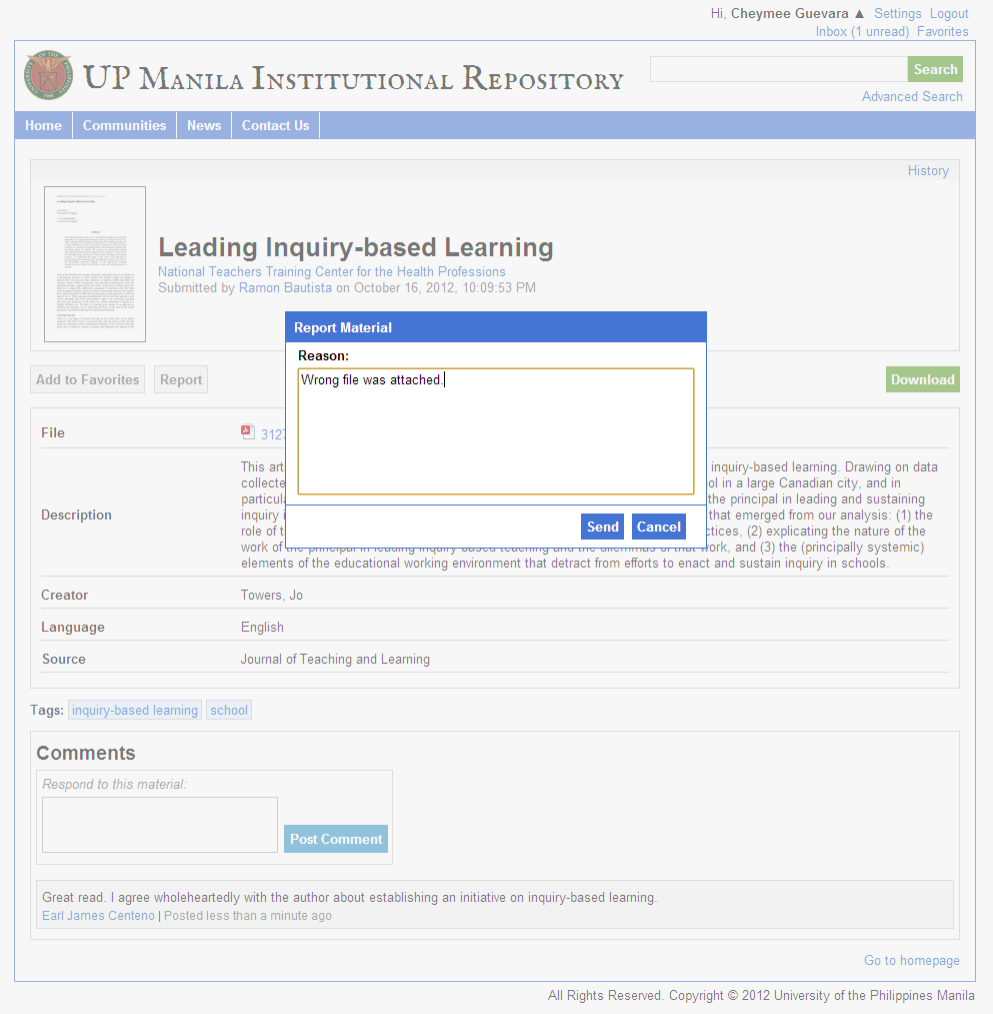

Figure 5.11. Report Material Form

After successfully adding a material as a favorite, we back a step to the front page and click a file named "Analysis of Geographical Data." Like the previous material, this page displays the material's information although for this material, a pink highlighted note at the top is stating that a newer version of it exists (Figure 5.12). Also note of the View button beside the Add to Favorites button. Text files in this repository can be annotated and the annotations can be viewed by clicking the View button.

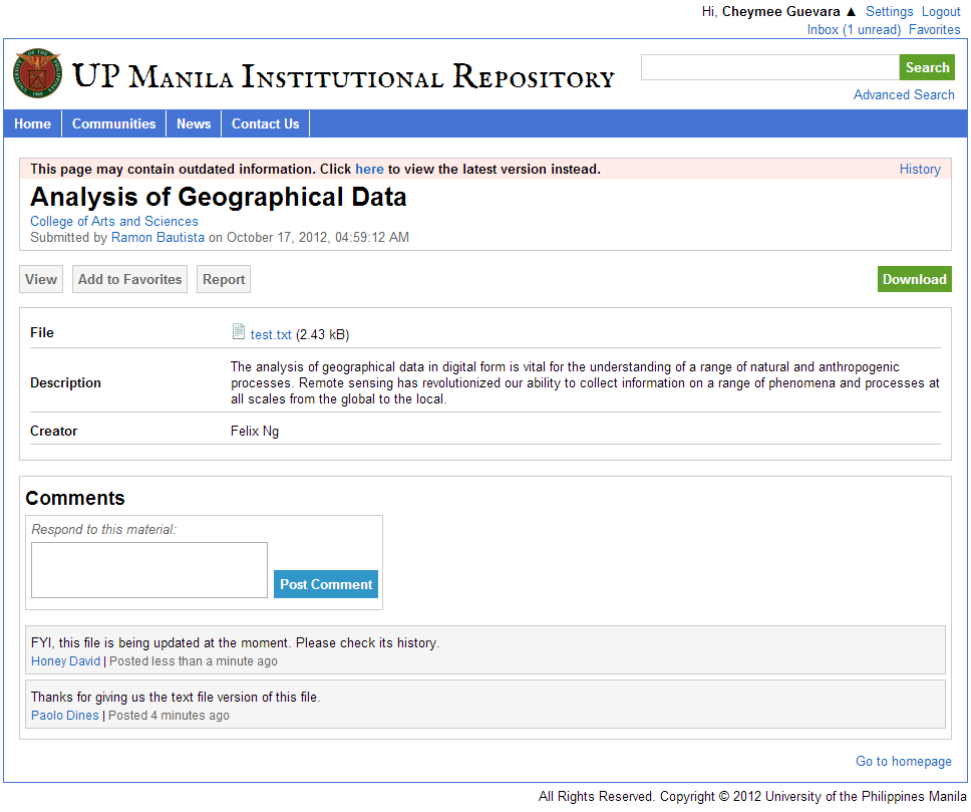

Figure 5.12. Material Page (Viewing an Outdated Material)

Clicking the History link will send the user to material's history page (Figure 5.13). From the data we can deduce that the material has been updated at least five times, with the previous material's page, highlighted in blue, being the second update to the original.

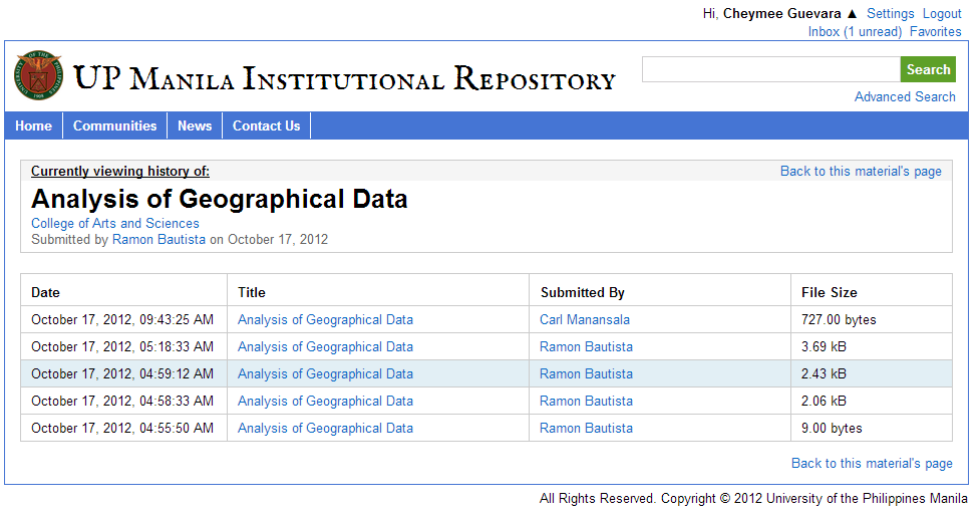

Figure 5.13. History Page

From the six entries, the one at the top of the history is always the latest version. Clicking on it will bring you to the material page of the latest version of "Analysis of Geographical Data." After verifying that this is indeed a new iteration, we click the View button to view its annotation.

Here on the annotations page (Figure 5.14), we can see that the material has only one annotation. Clicking the sole annotation will highlight (Figure 5.15) a part in the file (scrolling to the part if necessary).

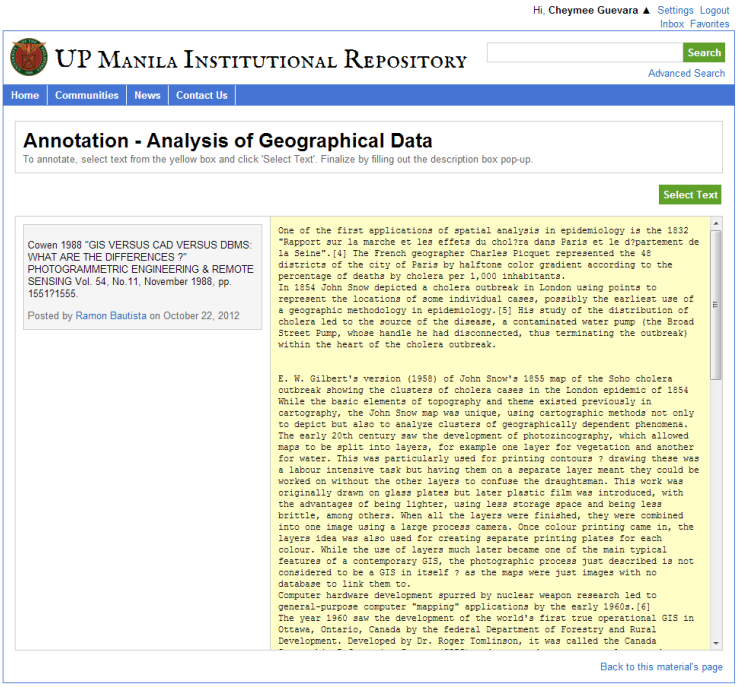

Figure 5.14. Annotation Viewer Page

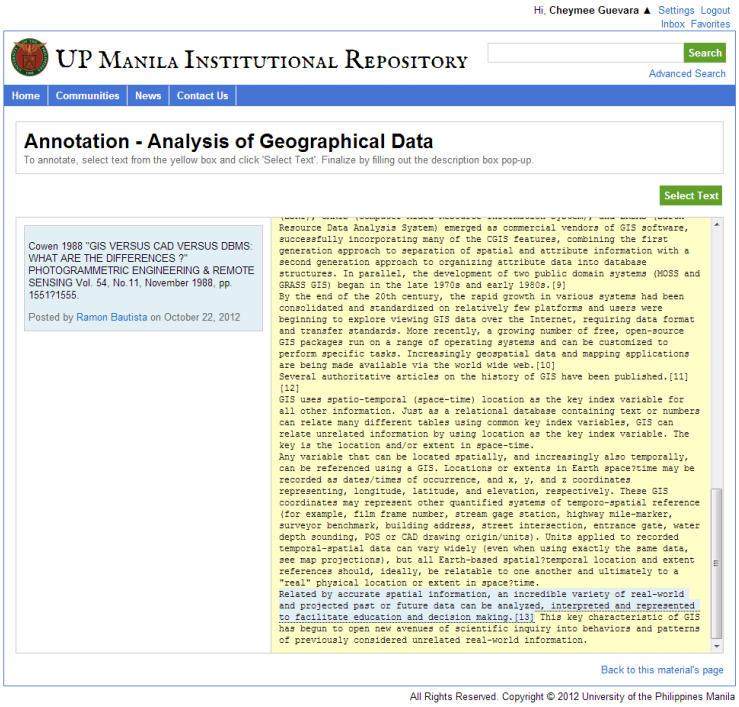

Figure 5.15. Annotation Viewer Page (Clicked Annotation)

Similar to viewing an outdated material's page, the page of a material under an embargo features a slightly different layout (Figure 5.16). Under the embargo rule, a file cannot be downloaded or annotated except for System Administrators and attending Community Moderators. To that sense, the download links together with the View button were removed.

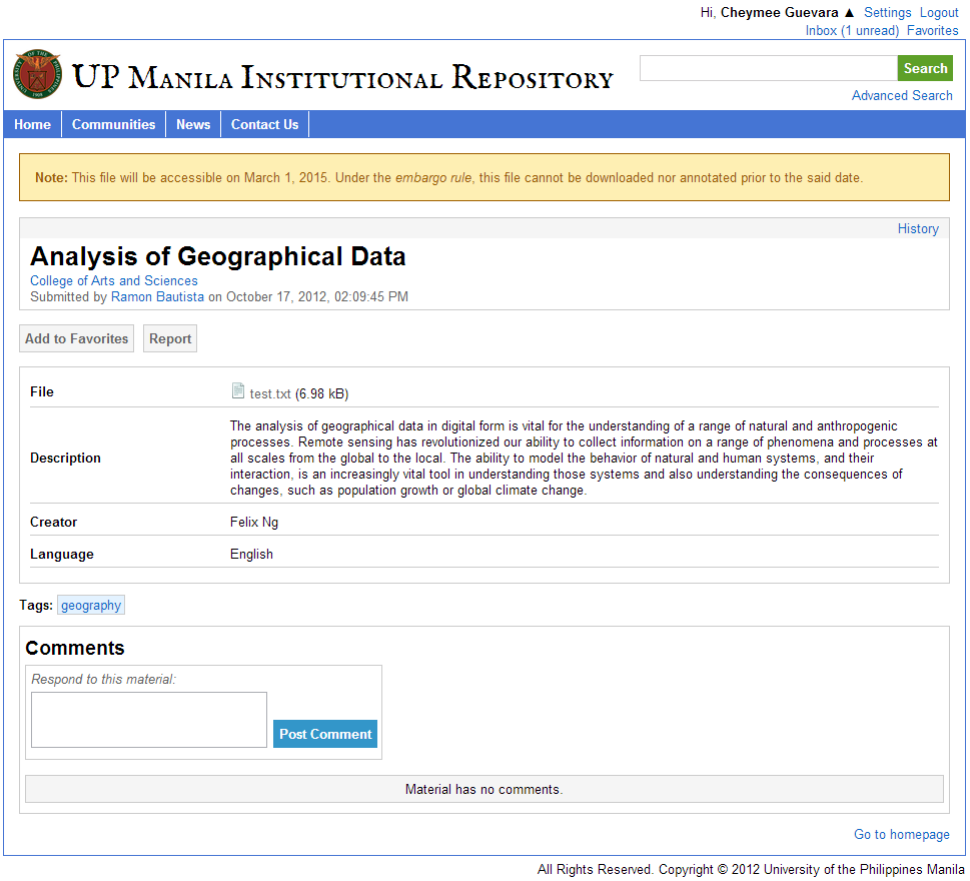

Figure 5.16. Material Page (Viewing a Material Under Embargo)

Below the Settings link on the upper right corner of the screen are two options: Inbox and Favorites. The option Inbox will send the user to a page displaying messages sent to the user which we will fully explore later on. Clicking Favorites opens up the favorites page (Figure 5.17) which displays a list of your saved materials alongside with an optional note (if the user has put one). From the favorites page, one can go to a saved material directly by clicking on an entry's title.

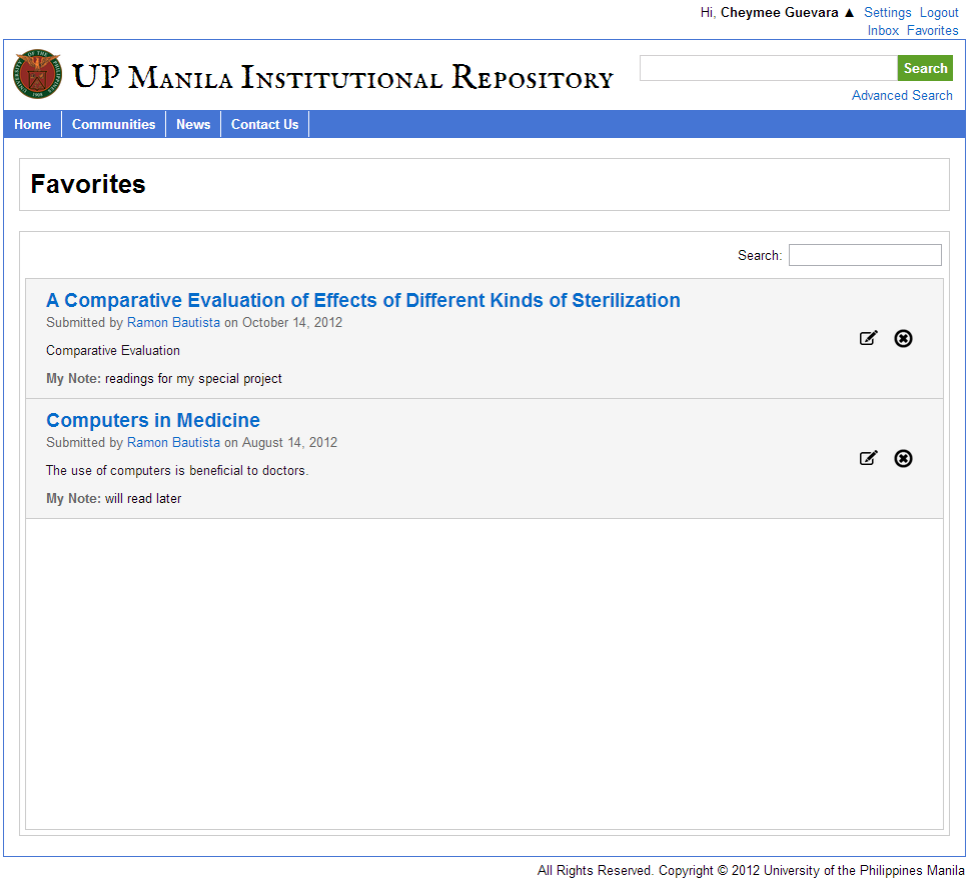

Figure 5.17. Favorites Page

The favorites page allows users to edit favorite notes. Similar to adding a material as a favorite, editing an entry displays a pop-up form (Figure 5.18). Clicking Update submits the form and the changes will be available immediately.

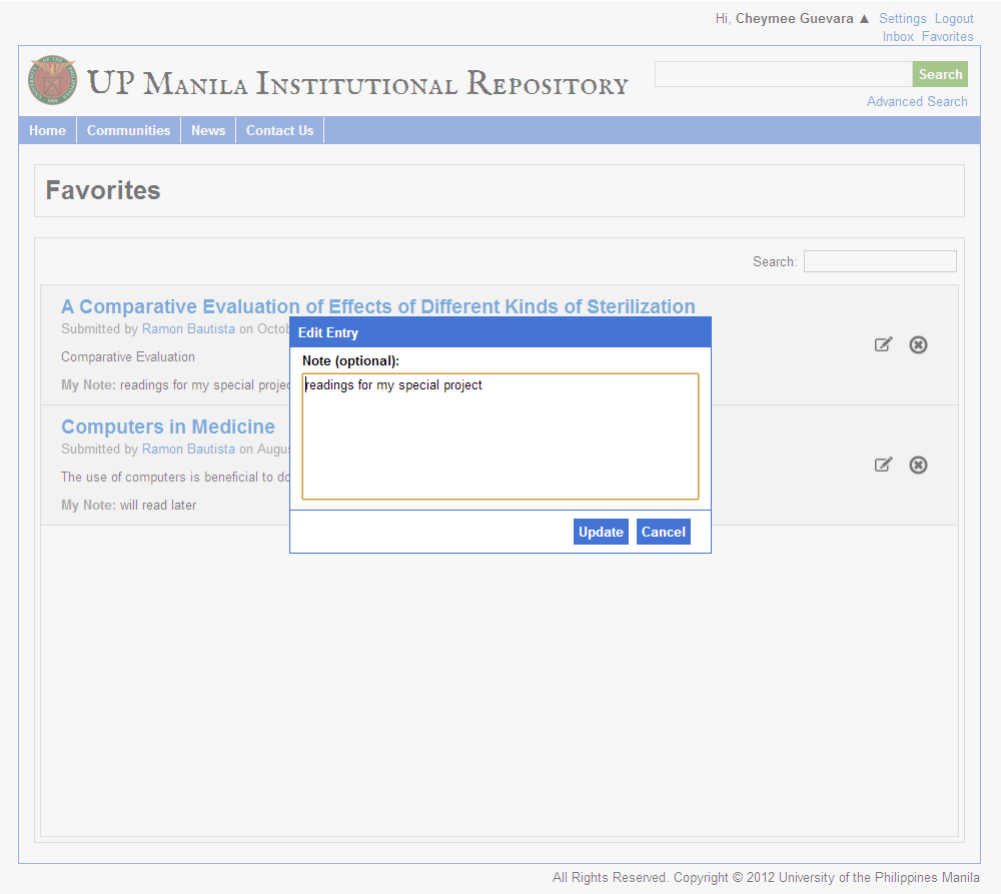

Figure 5.18. Edit Favorite Entry

There are several ways to discover materials in the repository. One is to make use of the search box located in the upper right corner of the screen. After typing a query and clicking Search, the user is sent to the search results page. In Figure 5.19, the user searched for the term "medicine" and got two results.

Figure 5.19. Search results for "medicine" using the search box in the upper right corner of the screen.

To narrow down search results, the user can make use of the advanced search functionality. Clicking the Advanced Search link below the Search button will display the advanced search page (Figure 5.20) which features an array of search options. From here you can choose which metadata fields to search, what file category and community the search result belongs, size of the file and whether or not the file is under an embargo.

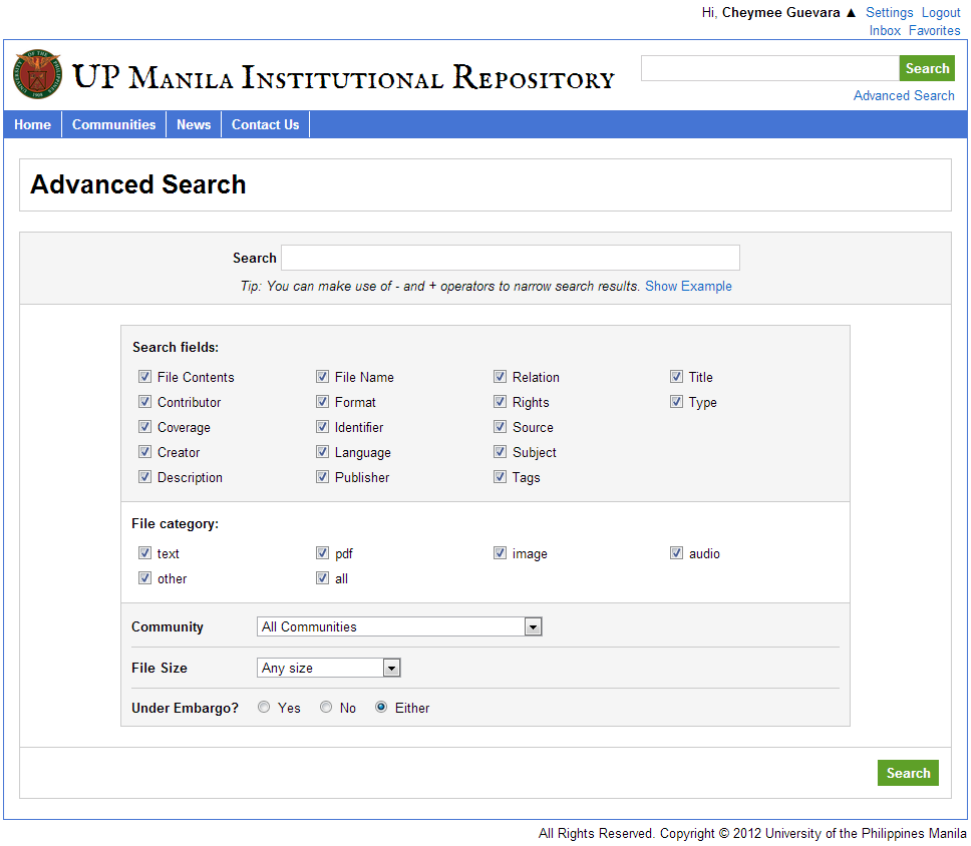

Figure 5.20. Advanced Search Page

Clicking Show Example at the bottom of the middle search box will display some search query examples (Figure 5.21) to the user. The use of the + and – operators can help in narrowing down searches.

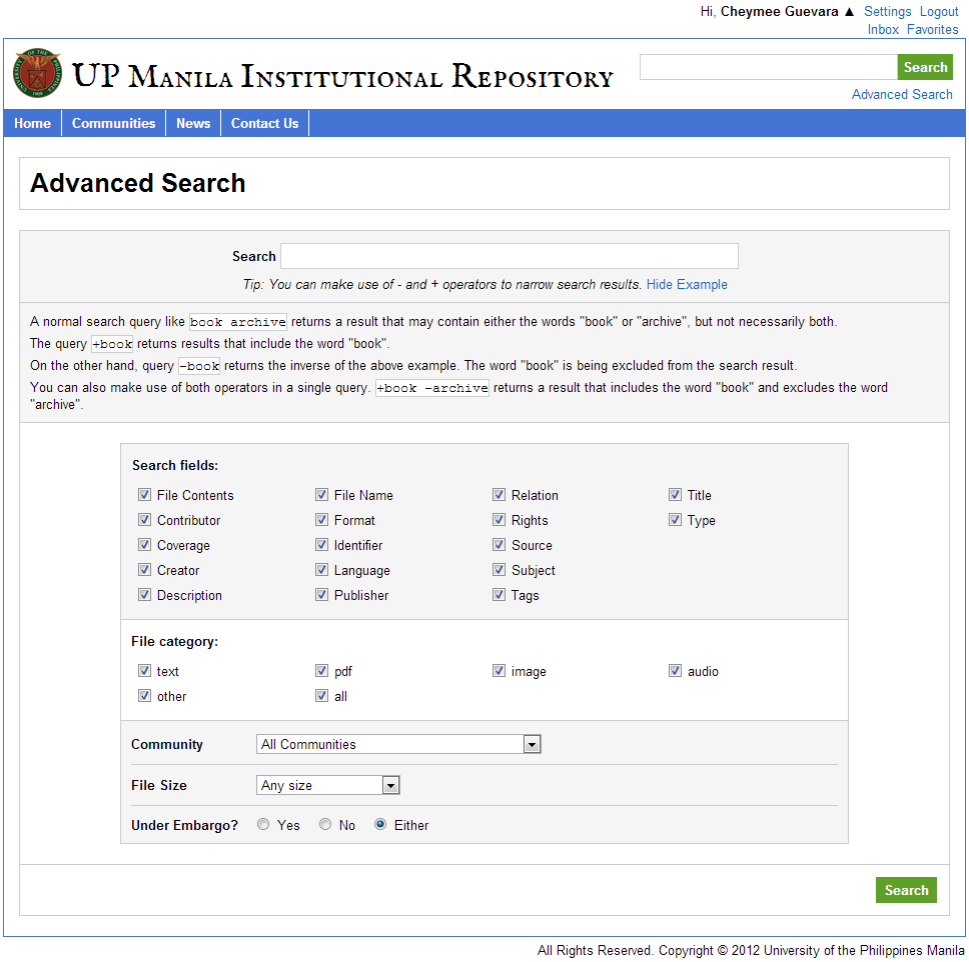

Figure 5.21. Advanced Search Page with Examples

To demonstrate how advanced search can narrow down searches, the user input "medicine" in the search box, but this time, opting to look only for the term in the title of materials. To be able to do this, every checkbox in the search fields are left unchecked with the exception of the Title checkbox (Figure 5.22). The resulting page is displayed in the next figure (Figure 5.23).

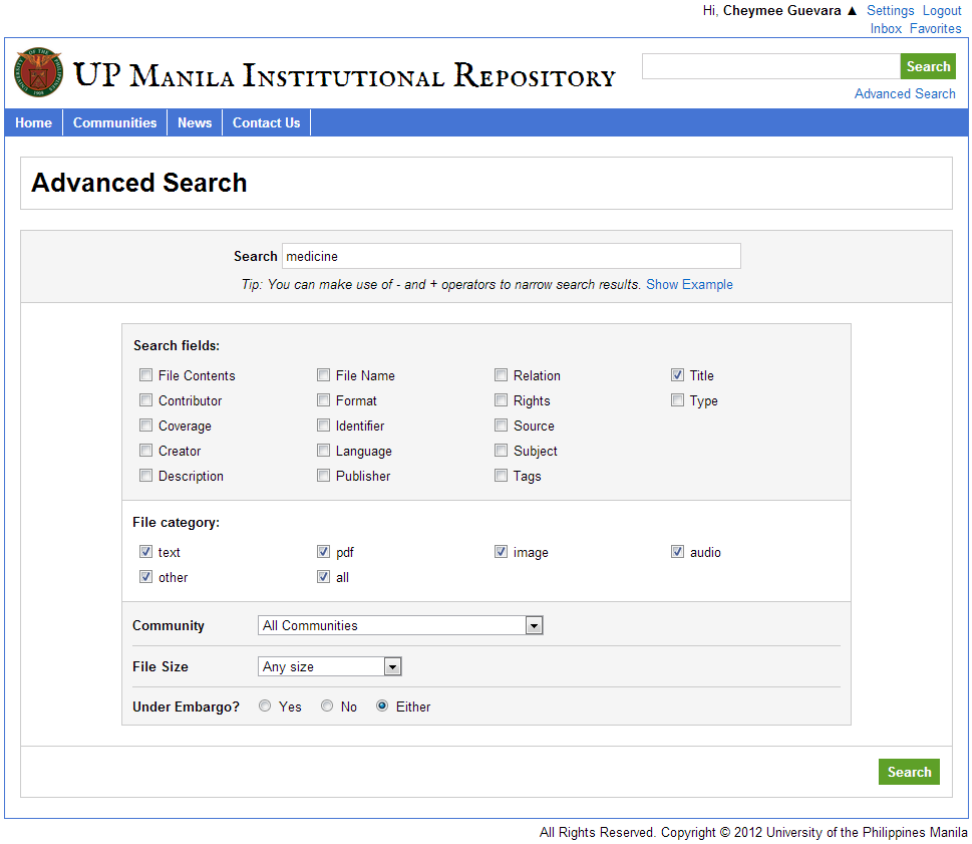

Figure 5.22. Advanced Search Page with the Query "medicine"

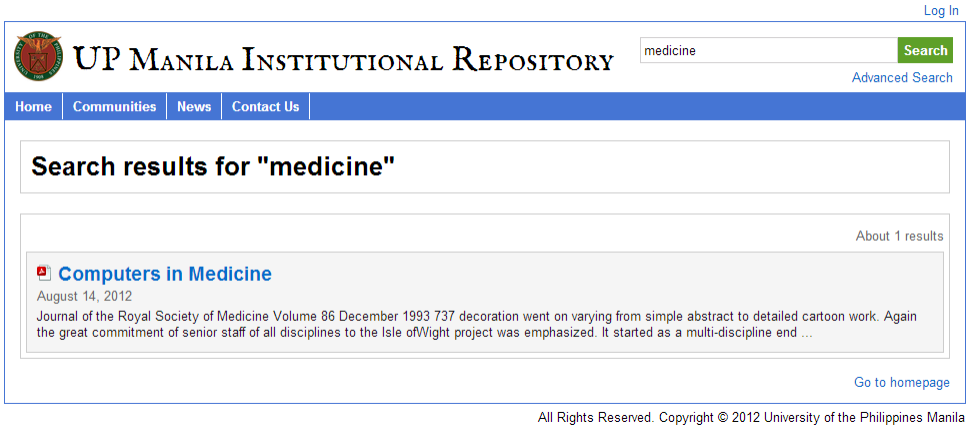

Figure 5.23. Search Results for "medicine" Using Advanced Search

You can also browse communities for materials. Clicking Communities in the long blue navigation bar below the repository title banner will send the user to the community list page (Figure 5.24). From the community list page, clicking a name of a community sends the user to the community page of the selected link. As an alternative, clicking on blue links that start with "college" may lead to a community page. Figure 5.25 shows the community page of the College of Arts and Sciences. The community page displays recent submissions to that community, its community moderators, in addition to the list of materials belonging to that community.

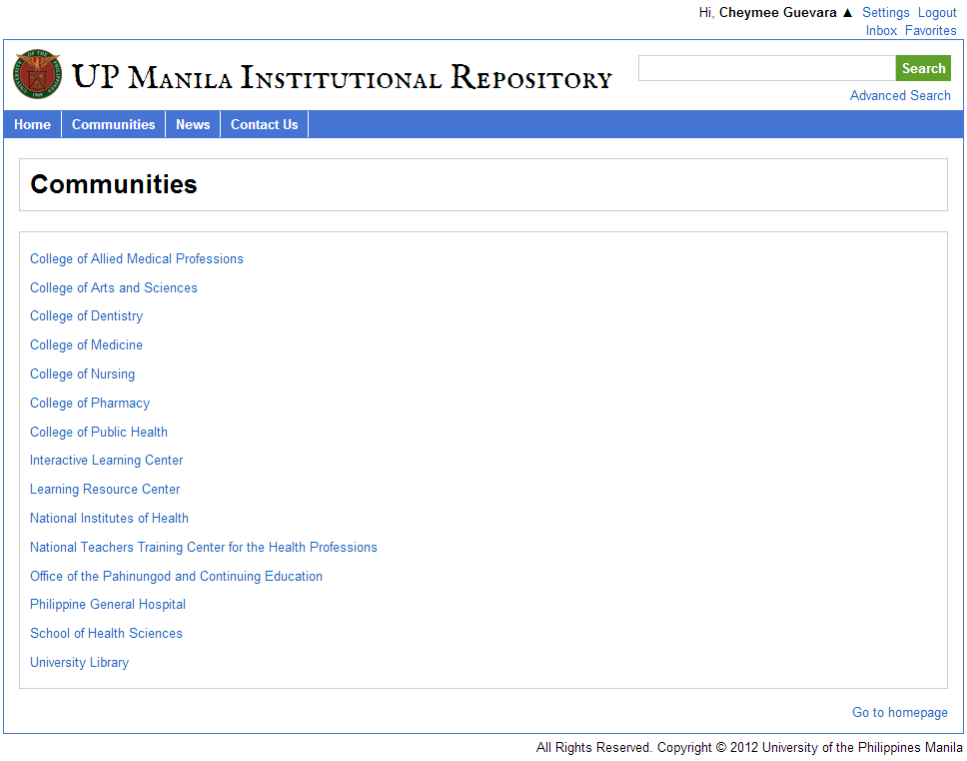

Figure 5.24. Community List Page

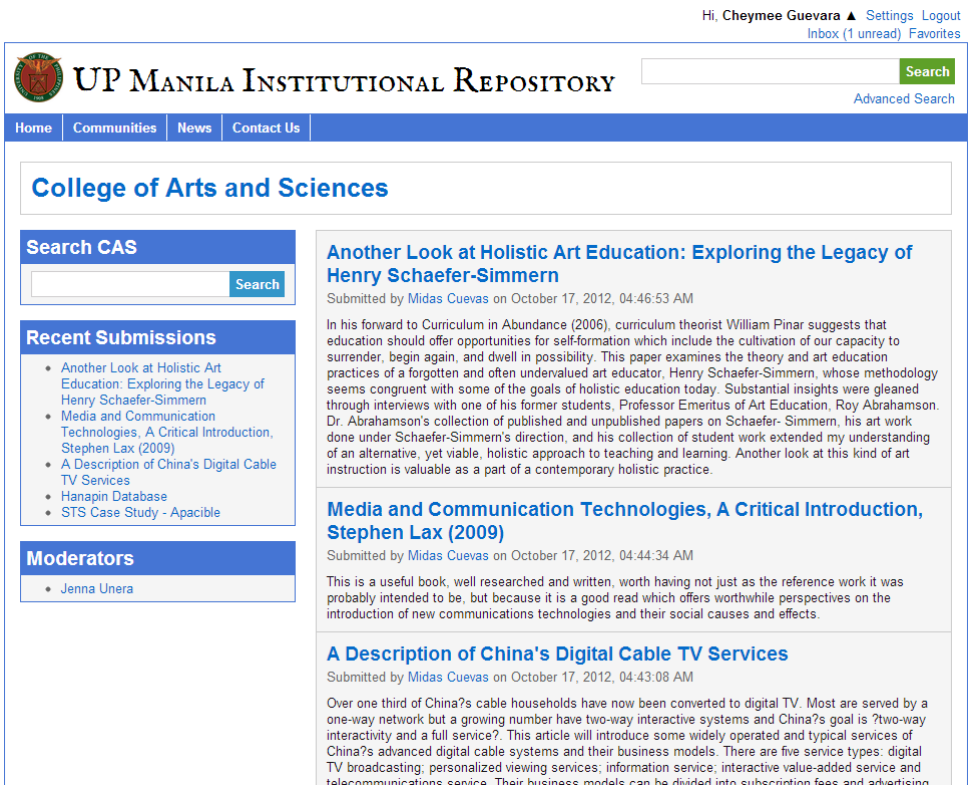

Figure 5.25. Community Page

Besides searching and browsing communities for materials, you can also look for materials in profile pages. If you like a user's material, chances are you will also like their other submissions. To view a user's profile page (Figure 5.26), click their name.

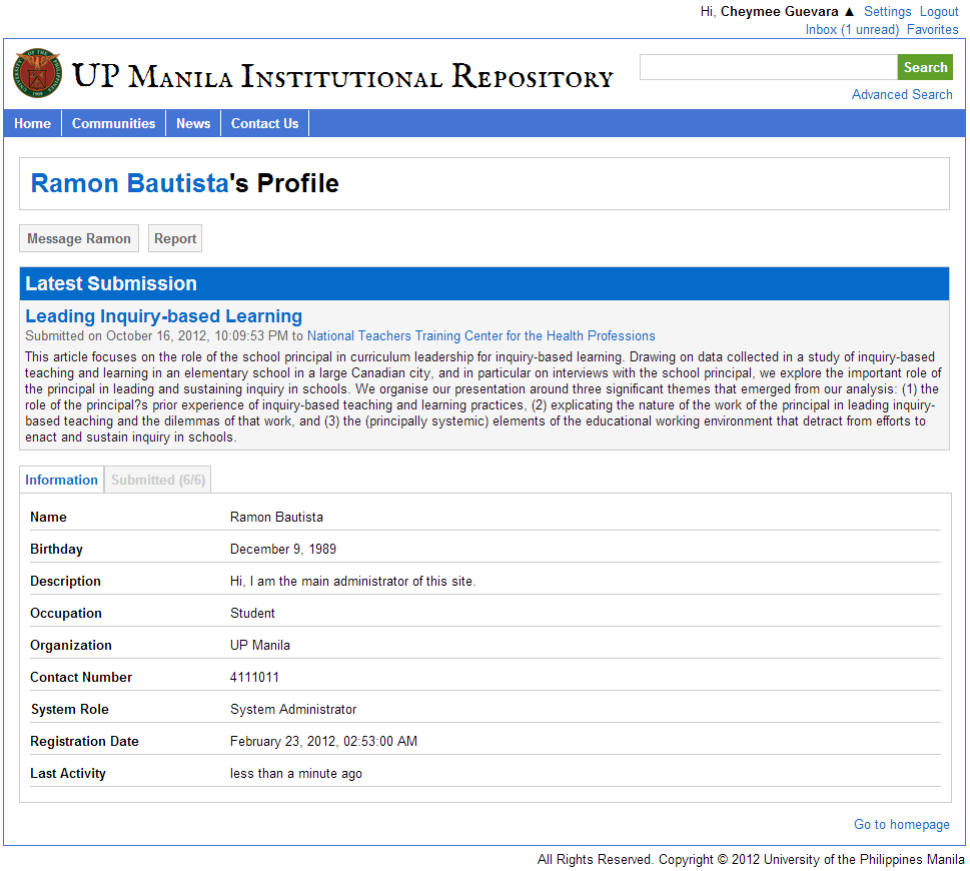

Figure 5.26. Profile Page

The profile page displays information about the user such as name, birthday, and description. The user's latest submission will also be viewable upon opening the profile page, granted that the user has submitted at least one material. Since Ramon Bautista is a System Administrator, the files he submit to the repository no longer need to be approved. Other users need to have at least

one of their materials approved before it will appear to the latest submission part of their profile page.

To view a list of contributions of a specific user, click the Submitted tab located on the user's profile page. The list (Figure 5.27) shares a familiar design to that of your favorites list. Click on an entry's title if you wish to visit its material page.

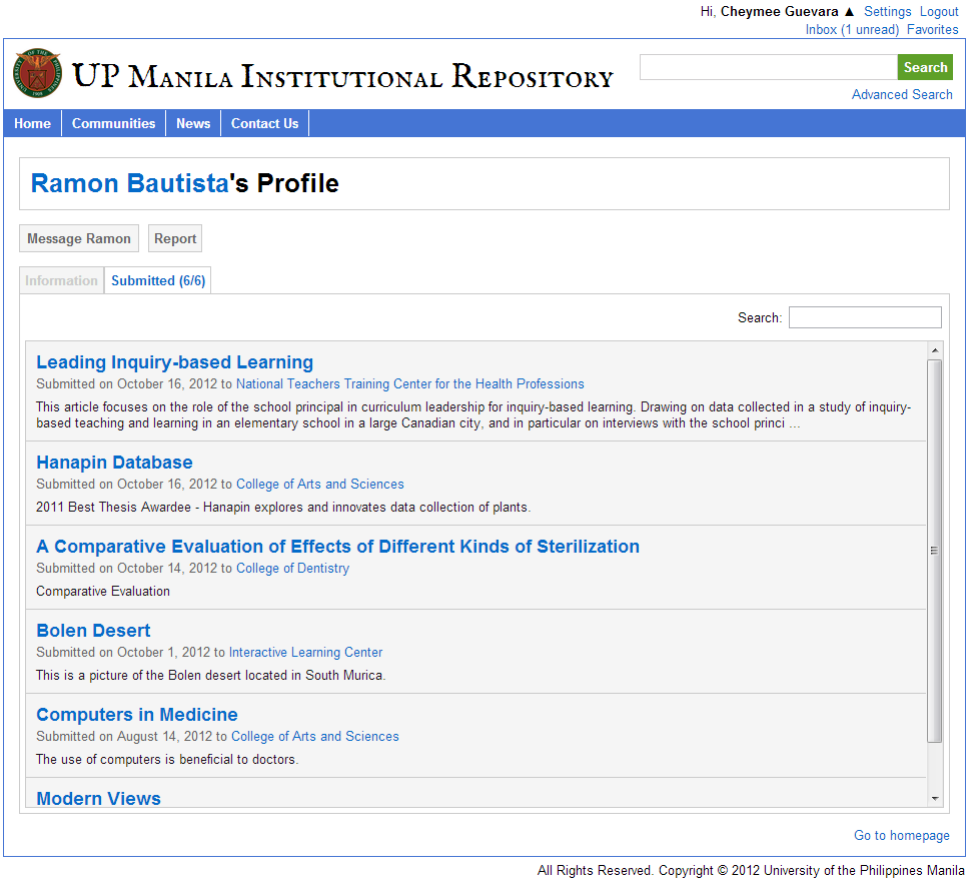

Figure 5.27. Profile Page (Submitted Tab)

Like materials, if you need to report a user, visit his/her profile page and click Report. The reporting process follows the same process as reporting a material. Once you have filled up the pop-up form (Figure 5.28), the message will be sent for evaluation to the System Administrator.

Figure 5.28. Report User

In this institutional repository, only System Administrators can post news although everyone can view them. One way of viewing the latest news is to go back to the front page and click the news links posted. Another method is to access the news list (Figure 5.29) by clicking the News link located at the blue navigation bar below the repository title banner.

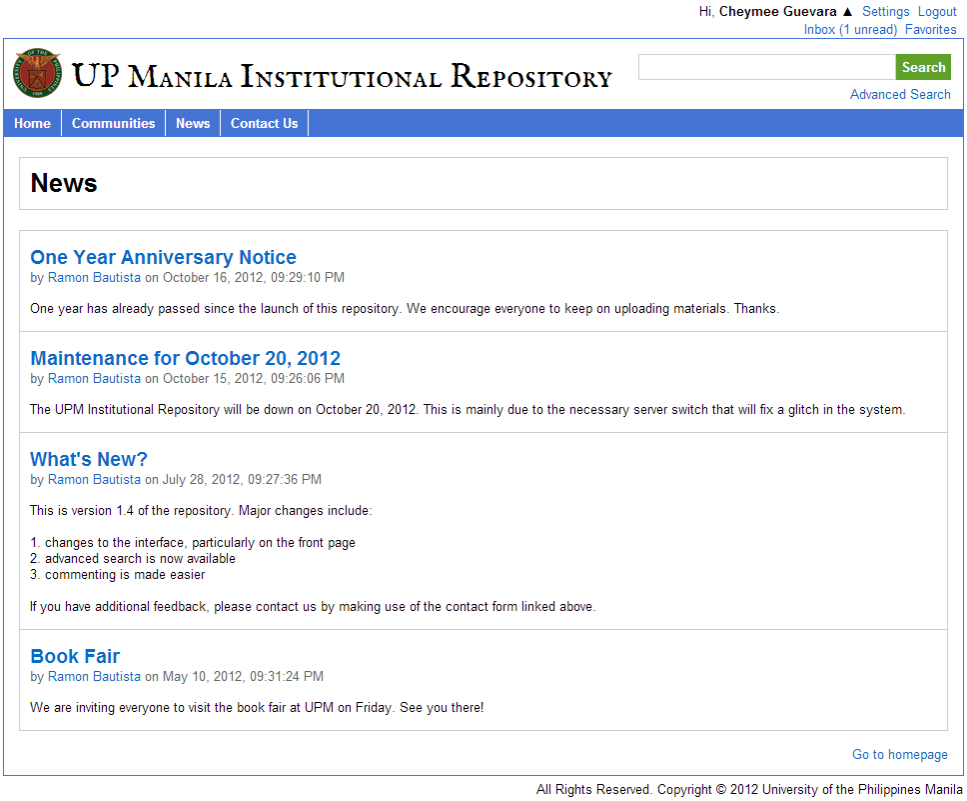

Figure 5.29. News List Page

Clicking a news entry will lead you to an entry's news page. Pictured below (Figure 5.30) is a news page with the title "What's New?"

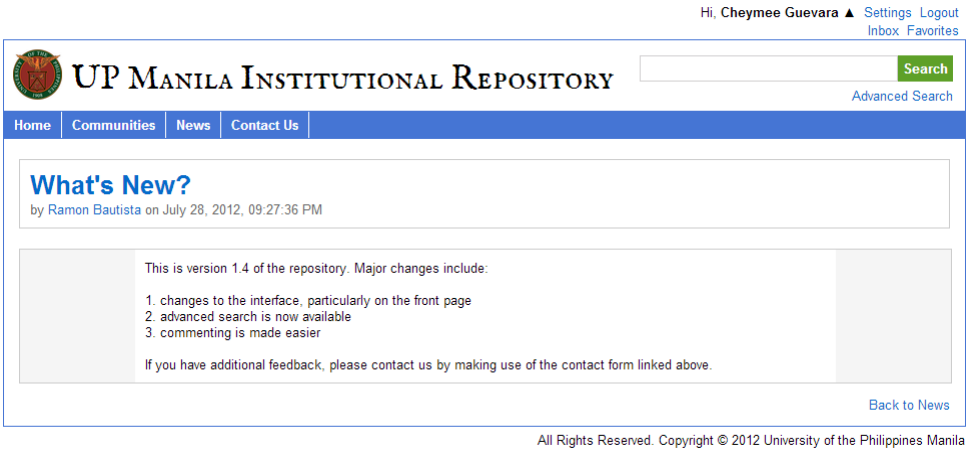

Figure 5.30. News Page

If you have a comment or suggestion about the site, you can click the Contact Us link located at the blue navigation bar at the bottom of the repository title banner. This will send you to a simple page (Figure 5.31) with one input box for your message. Like reports, after filling out and submitting the form, the message will be sent to System Administrators.

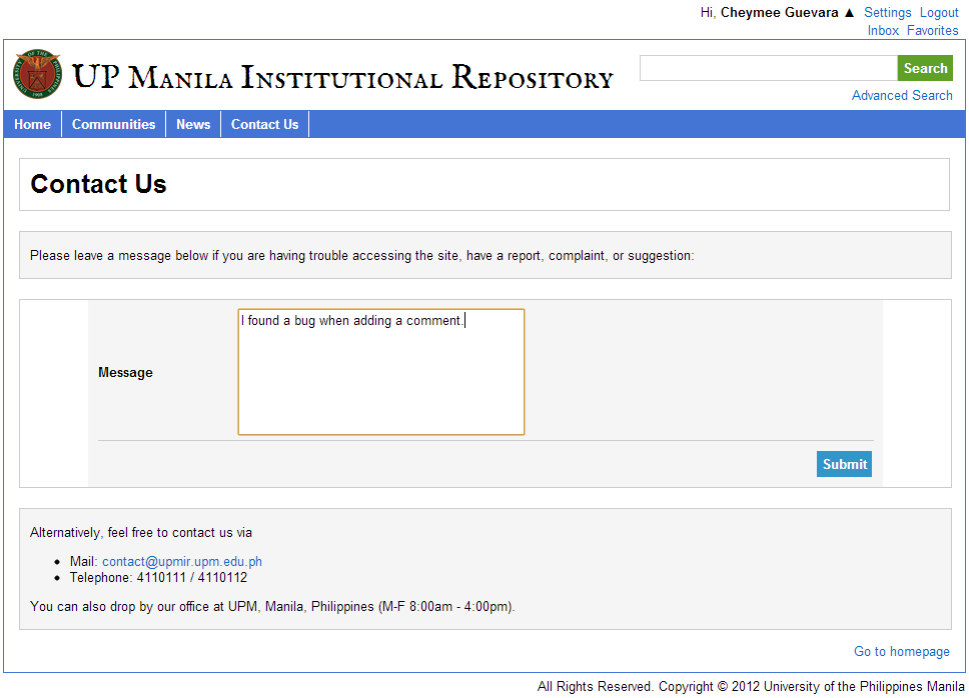

Figure 5.31. Contact Us Page

So far, the basic functionalities of the repository have already been covered from the perspective of the Registered User. The other system role groups University User, Community Moderator, and System Administrator share similar views to these functionalities. To accommodate various system roles imposed by the repository, extending the functionalities of what we've seen so far is necessary.

First off, similar to logging in as a Registered User, logging in as a University User directs the user back to the homepage. The University User's homepage offers few visual changes from a familiar homepage (as pictured back in Figure 5.5). Since University Users have the ability to contribute materials to the repository, a link Contribute can now be seen on the navigation bar

under the repository title banner. A list of the user's contributions, approved or pending, can be viewed under the My Contributions link located at the top right corner of the screen.

Clicking Contribute directs the user to a contributions option page (Figure 5.32). Here we see two options: upload a new file and upload an update to an existing file.

Figure 5.32. Contributions Option Page

Choosing the second option, upload an update, will direct the user to a help page (Figure 5.33) on how to properly update an existing material.

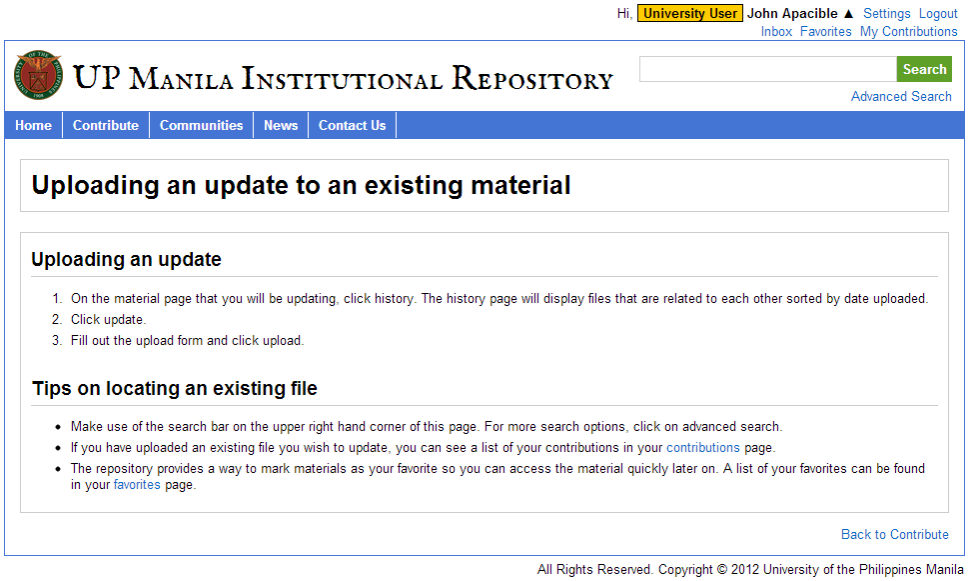

Figure 5.33. Uploading an Update Help Page

Below (Figure 5.34) shows the history page of a material with the Update button visible to the University User.

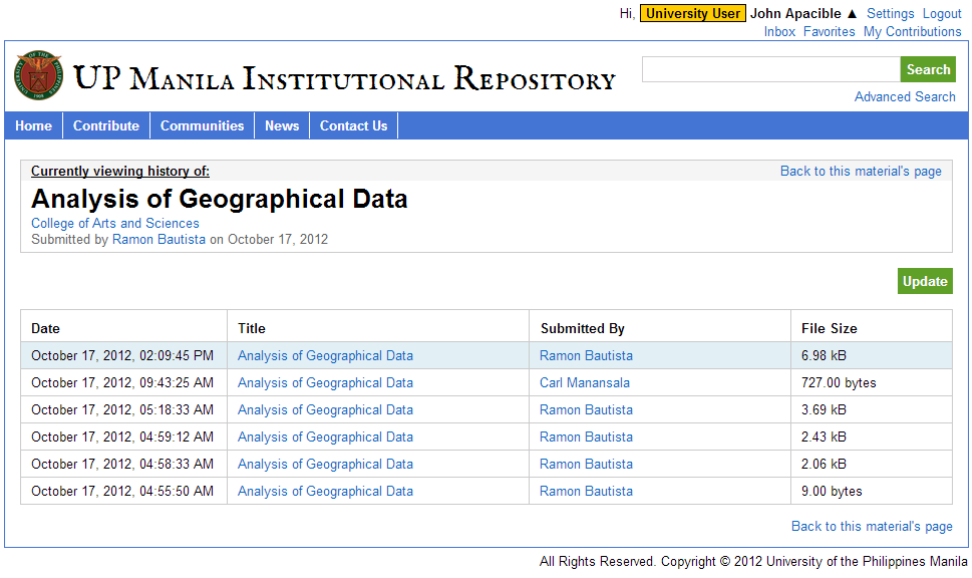

Figure 5.34. History Page (Update Button Visible)

Backtracking to the contributions option page, we now choose the first option (upload a new file) which will lead us to the upload form. An empty upload form can be seen below (Figure 5.35). The user can hover their mouse cursor over the field labels to see a description of the fields. Below, the user is shown hovering over the file label.

The upload form requires the user to be able to provide the file, its title and description, and specify a community for it before the system will let you submit the material. Attempting to submit the form without filling out these required fields will prompt a visible red note (shown in Figure 5.36) at the right side of an empty required field.

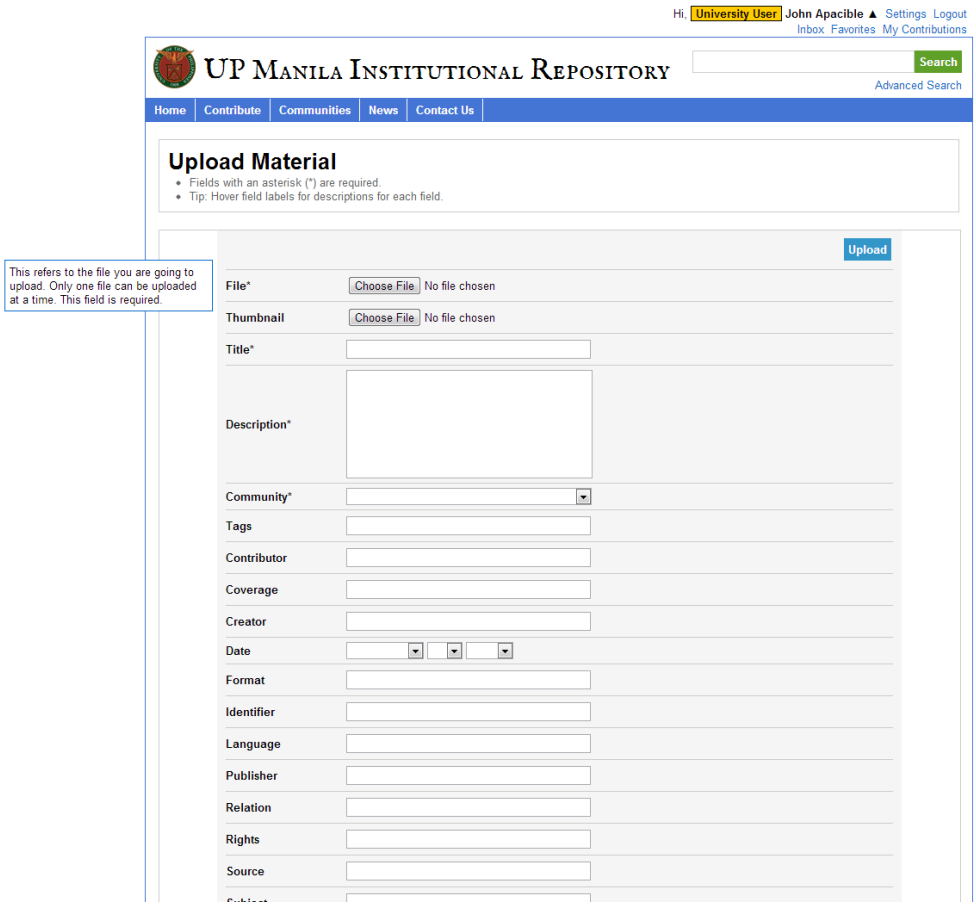

Figure 5.35. Upload Page

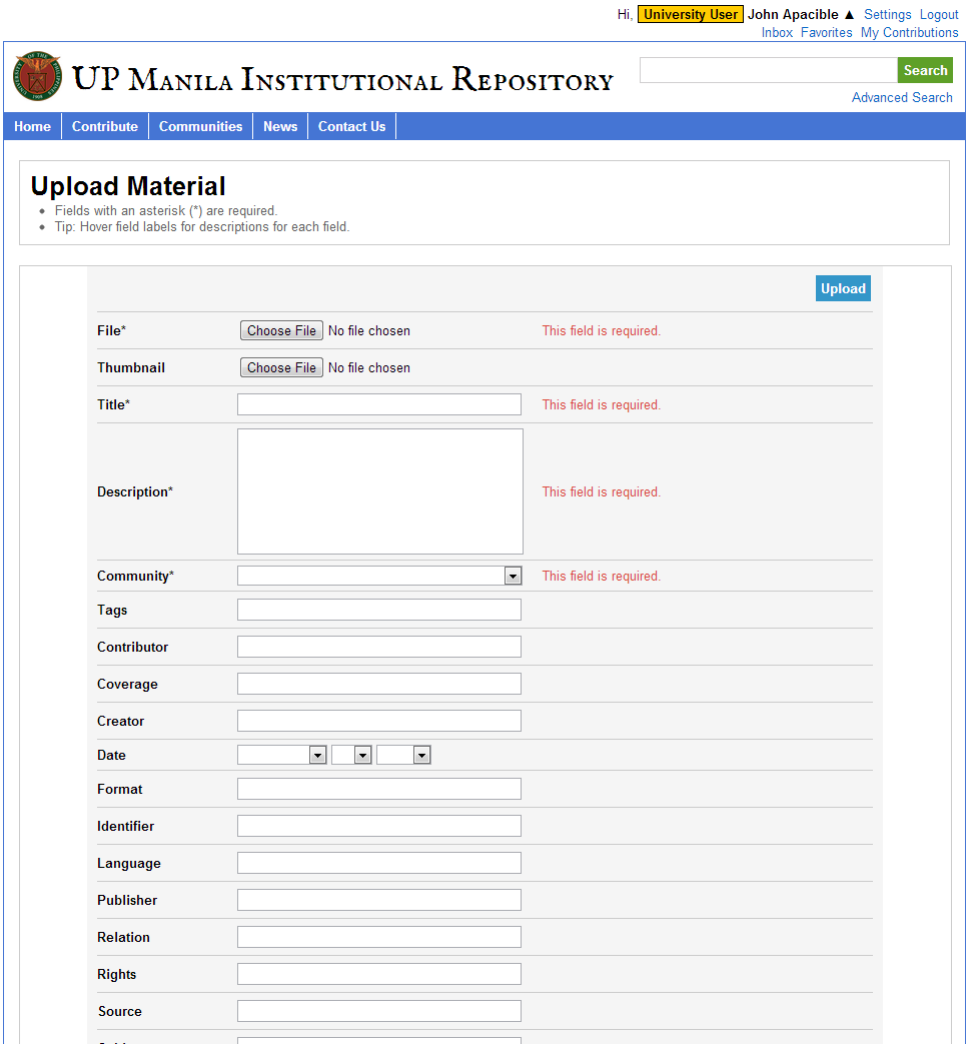

Figure 5.36. Upload Page (Error Checking for Required Fields)

As an aside, following the steps stated in the help page on how to upload an update to an existing material will also send the user to an upload form similar to the images above, although these fields will already be filled with data sans the file from the latest version of the material you wish to upload.

Once the user fills up the necessary fields to be able to submit the material, the user will be directed to his/her contributions page. The contributions page is divided into two separate tabs with one containing approved materials while the other tab contains materials still waiting for approval. Figure 5.37 shows the contributions page of University User John Apacible with a note informing him that the file was successfully sent to the administrators and moderators for approval. An exception here is that upon uploading a material, the System Administrator will be sent back to the material manager page that is only available to site administrators.

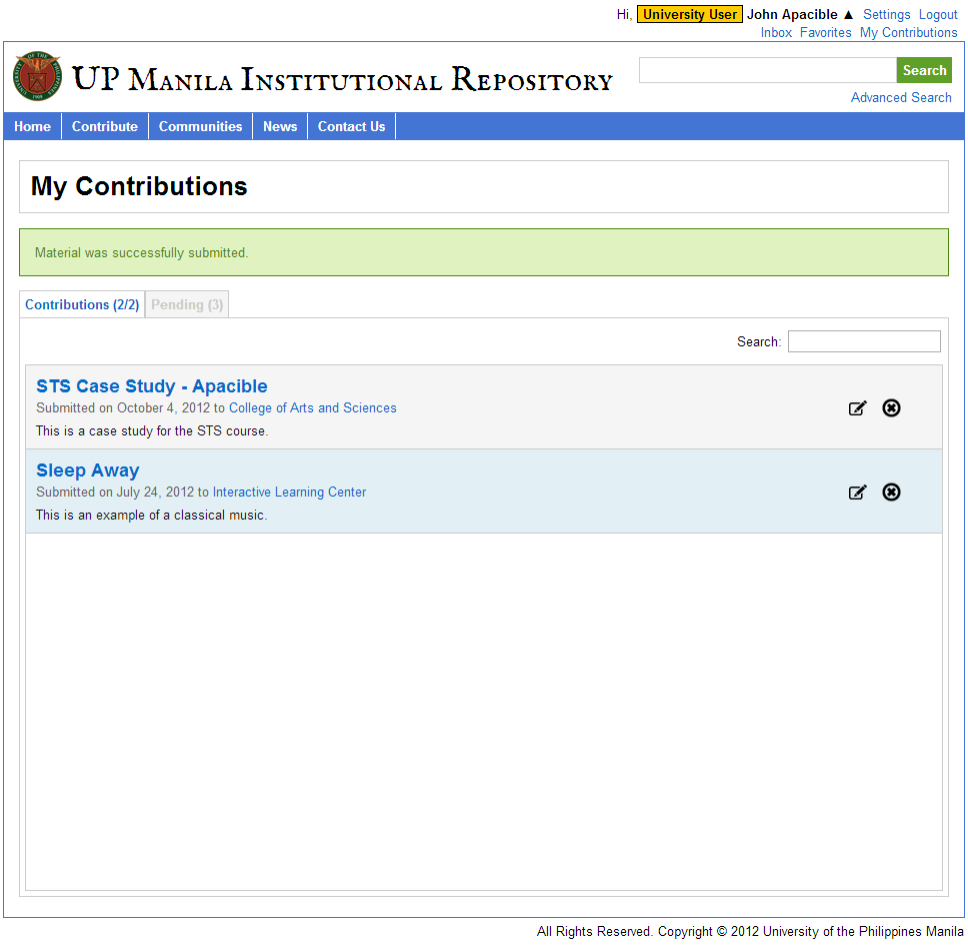

Figure 5.37. Contributions Page

Here in the contributions page, users can directly update or delete their own submissions. For pending materials, only the delete option is available. Clicking the Delete button will reveal a pop-up confirmation box (Figure 5.38). Clicking Yes will delete the material from the system. This action is irreversible.

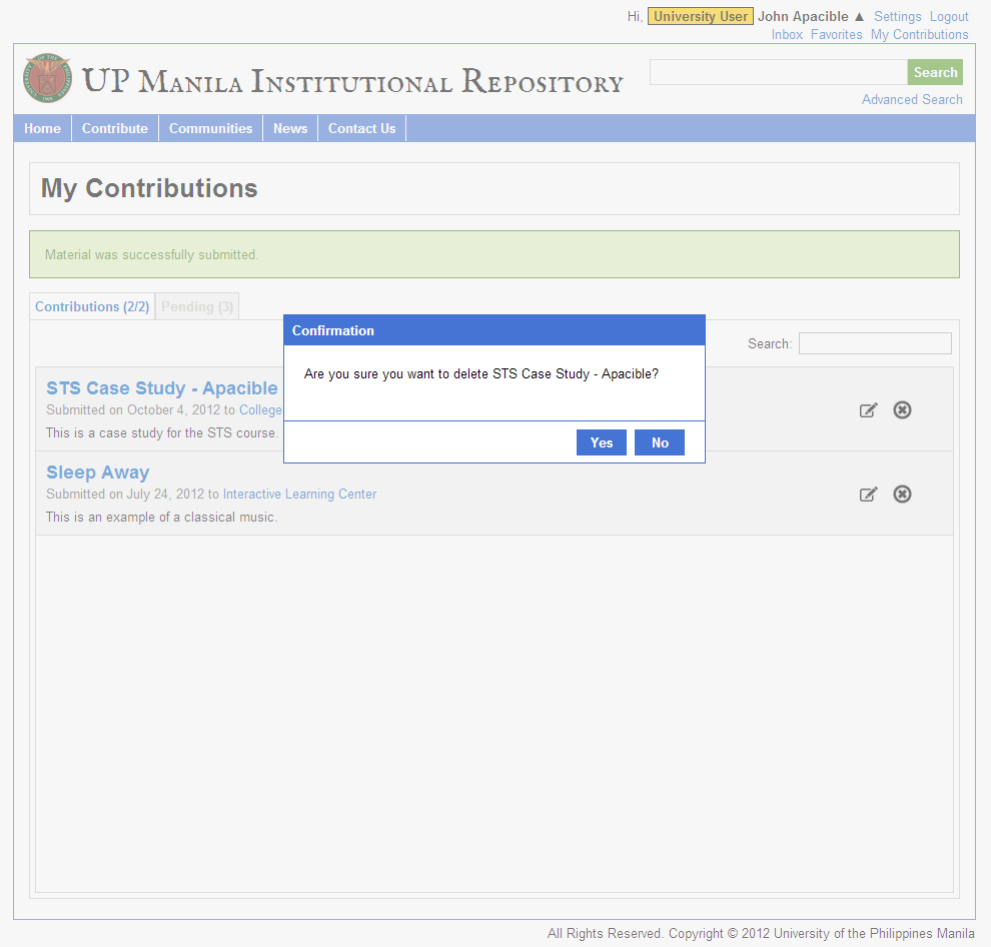

Figure 5.38. Confirmation Box for Deletion of Own Material

Pending materials are not visible to the public except by you (depositor), the System Administrator, and responsible Community Moderators. Clicking the title of your pending material in your contributions page (pending materials tab) will direct you to a special page

(Figure 5.39) intended for reviewing purposes. You can also delete your pending material from the system within the pending material viewer page.

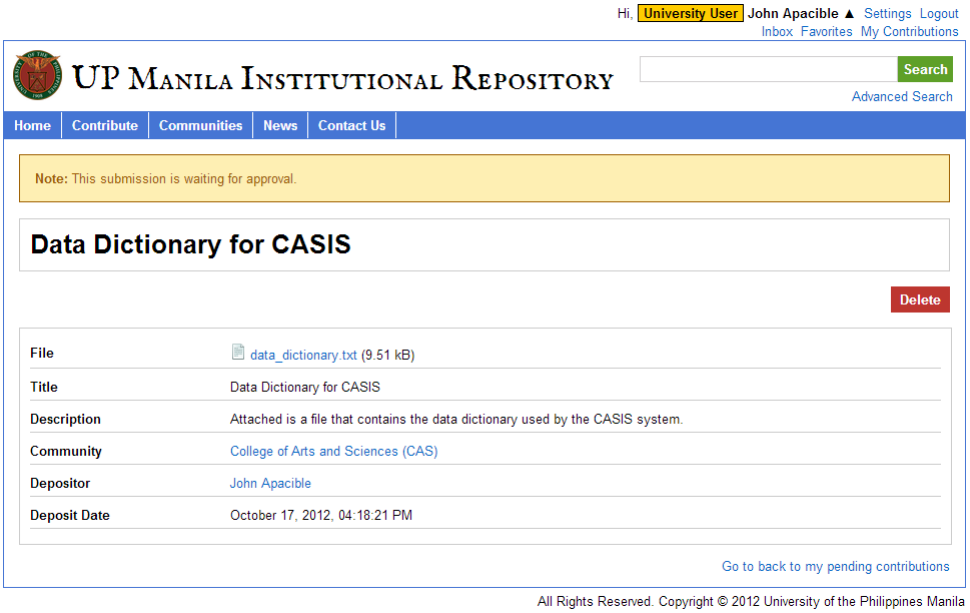

Figure 5.39. Pending Material Page

Heading back to the contributions page by either clicking the link at the bottom right corner of the pending material page or clicking the My Contributions link at the upper right corner, let's view a text file to annotate.

As a University User, now's a good chance to explore about the Select Text button. The Select Text button means that the current user can actually submit an annotation. Like materials, annotations can be subject to approval from the System Administrator or Community Moderators. To create an annotation to submit, the user needs to highlight part of the text from the yellow box and then click the Select Text button. A form (Figure 5.40) will be presented to

the user which displays the text highlighted by user at the top and an input box at the bottom. This input box is where the user should put his/her annotation about the text highlighted earlier.

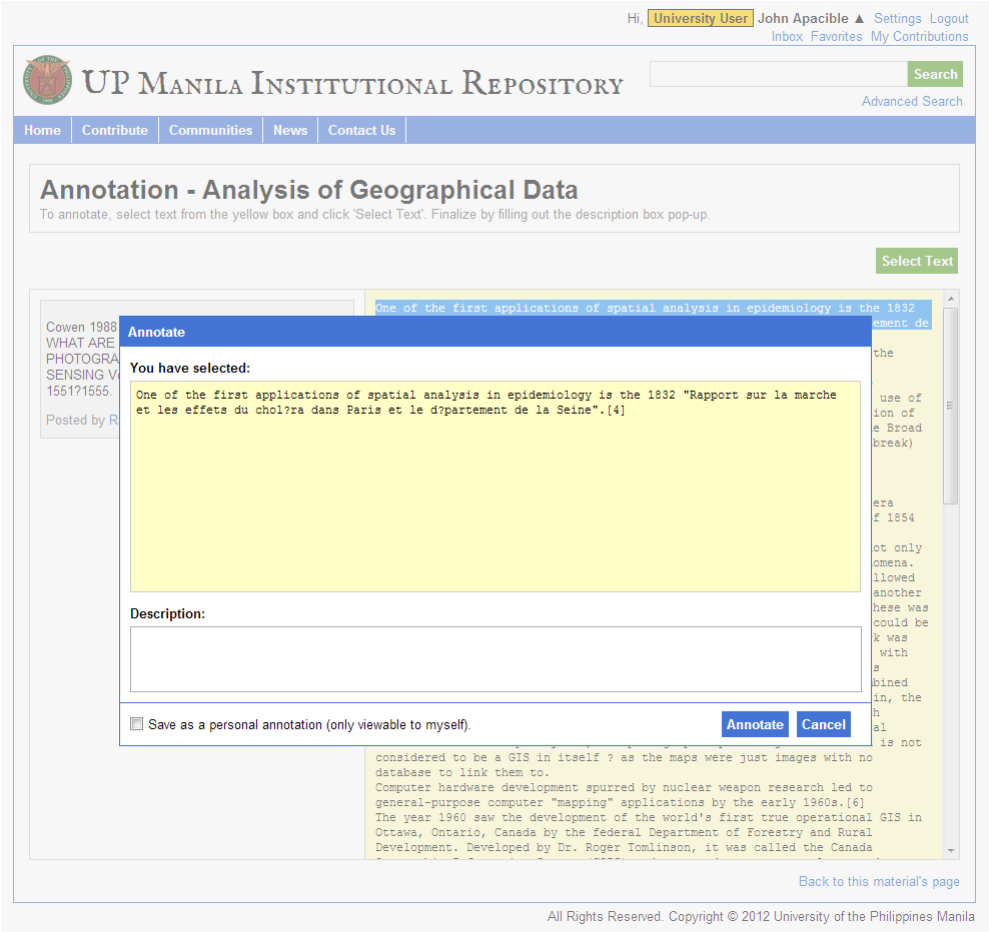

Figure 5.40. Annotation Form

After the annotation has been submitted, the annotations page will look like the one in Figure 5.41 in which the University User's submitted material is waiting for approval. Annotations waiting for approval can be deleted anytime.

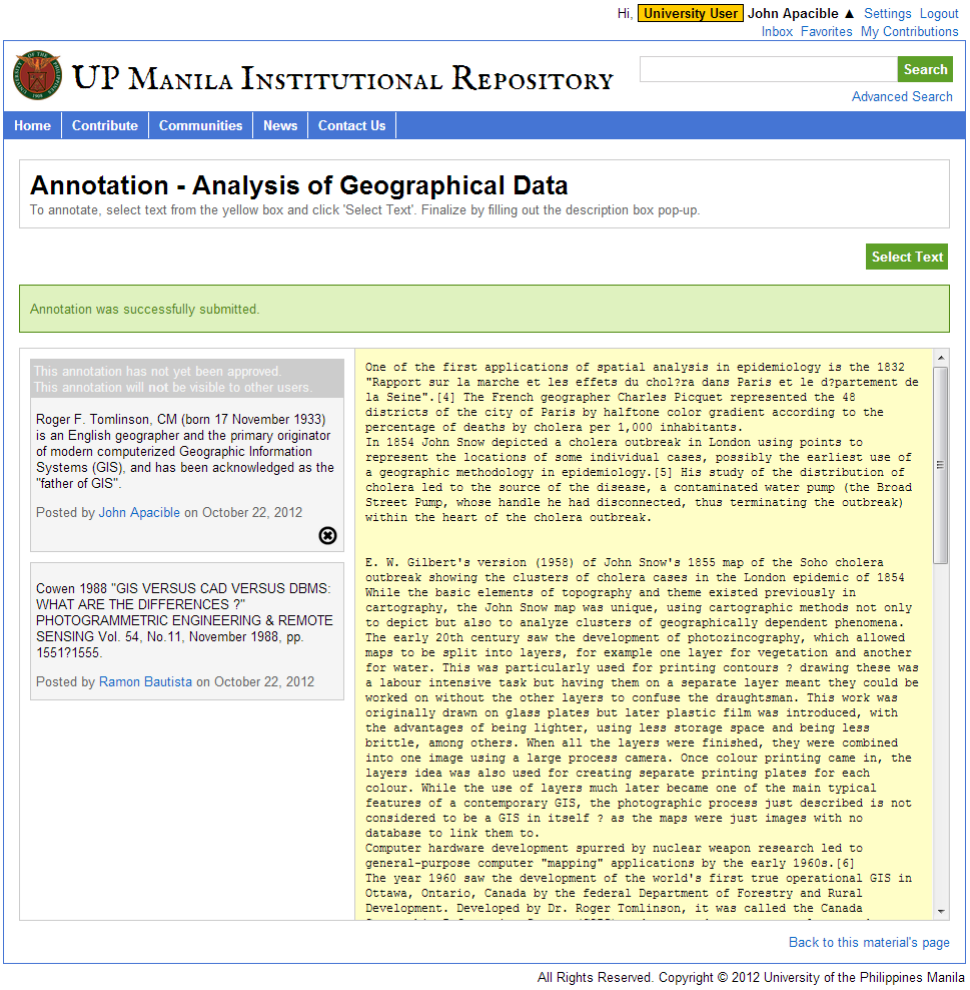

Figure 5.41. Annotation Page (Pending Annotation Visible)

Once the annotation is approved, the annotation will finally be visible to anyone that can view the material and its annotation (Figure 5.42). An approved annotation can be edited and deleted by its owner. As a University User, editing the annotation will change the contents of the annotation although the edited annotation will be resubmitted for another approval.

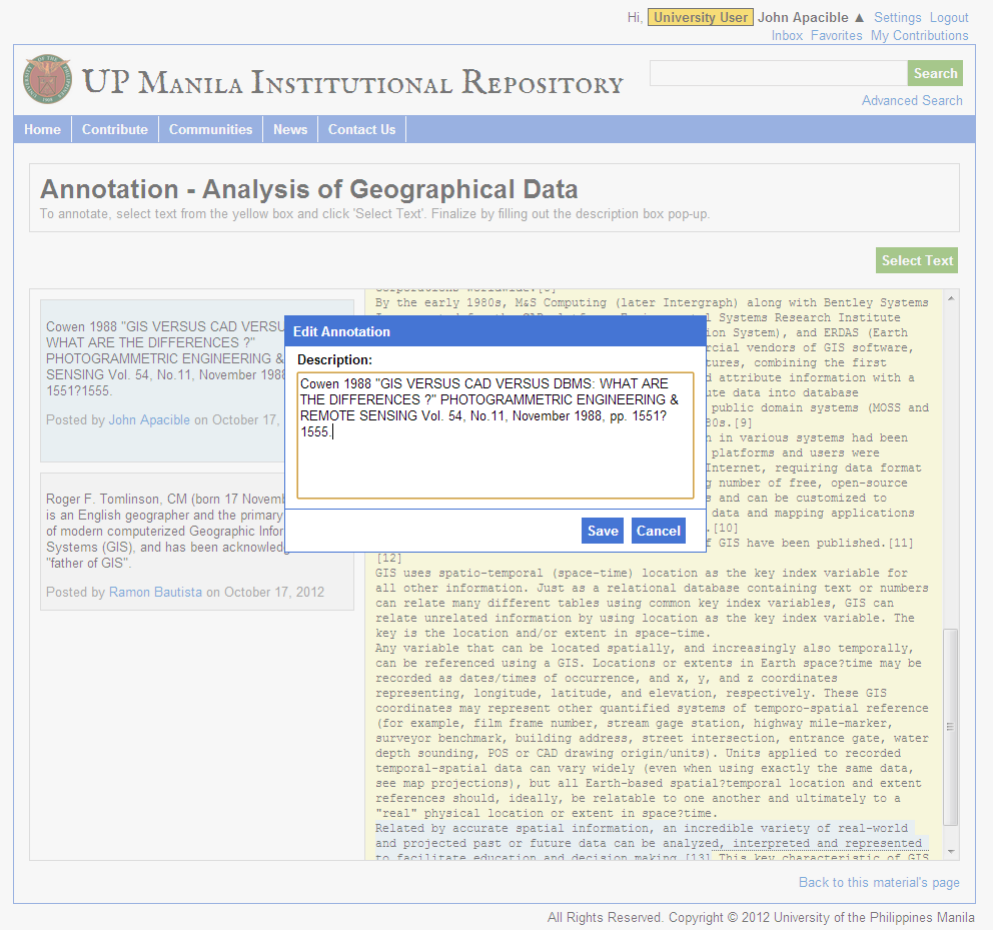

Figure 5.42. Edit Annotation

Having explored the extent of access to system functionalities of both Registered Users and University Users, we move on to the administrative portion of the repository. Making use the stated objectives (section II) of this paper as a guideline, we should know by now that both

System Administrators and Community Moderators are capable of managing materials. This means that both system roles can add, delete, approve materials, and be able to manage the corresponding annotations. On the subject of what the Community Moderator cannot do, a definite rule of the repository is that only System Administrators can manage users, news and add communities. Also, outside of the Community Moderator's scope (e.g. foreign community), the role of the Community Moderator is reduced to that of the University User. In exploring the administrative portion of the repository system, we will utilize an account that has the system role of a System Administrator.

As seen in Figure 5.6, changes like the addition of links to select administrative pages are now visible. For these links, a change in highlight color was made to be easily seen by users and also as a means to differentiate these links from those that direct to non-administrative pages such as Communities (sends user to a list of communities). These links direct the System Administrator to either an upload material page, a user registration page or news creation page while Community Moderators have access to only the upload material page. At the upper right corner of the screen, more options are now available for managing materials, communities, users, and news, and system log viewing. Like Registered Users and University User, System Administrators also have access to the system mail and the favorites page. Community Moderators, as seen in Figure 5.7, have access to his/her Contributions page since it was already stated above that outside of the scope of moderation, Community Moderators functions as a University User.

While the messaging feature of the site is far from being classified as an administrative feature, now is a good time to explore its functionality from a System Administrator's view. Clicking the link Inbox at the upper right corner sends the user to the list of messages sent to the viewer (Figure 5.43). In the inbox page, you can see a Compose Message button and at the bottom is a table listing the user's messages. Rows colored in blue are unread messages, while those that have a white background are messages that were already read. The columns subject, from, and date are table headers and clicking them sorts, in alphabetical order, the respective column entries. Notice that every row starts with an icon. Each icon represents to what an individual message may contain. For example, the image  means that a new annotation has been submitted. For Registered Users and University Users, icons  and  are the message icons that they will be encountering. The  icon is reserved for system notices such as a message stating that a submitted material has been rejected, while messages with the icon  means that a message was created by clicking the Compose Message button.

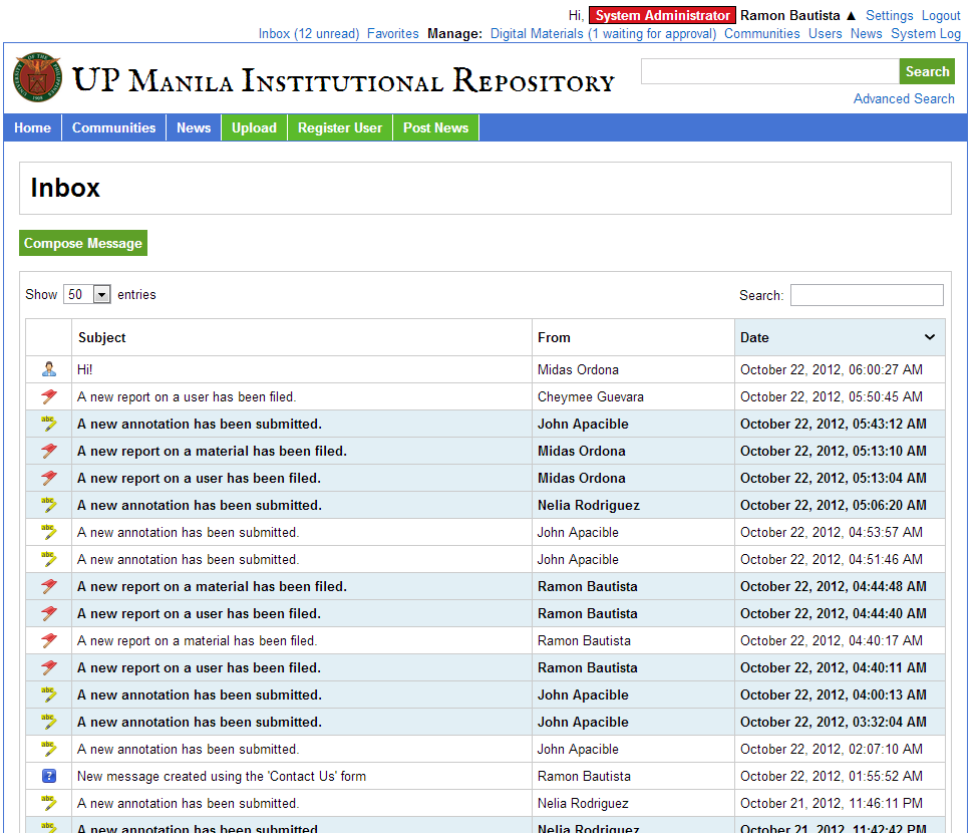

Figure 5.43. Inbox Page

From the inbox page, the user can click a row to go to a message he/she wants to read. In this instance, the user clicks on a user report message which begins with icon . After clicking the message, the user will be directed to the message page. An example of the message page is shown in Figure 5.44. This message is a special type of message since it was generated from reporting a user. Messages that are user reports always have an extra functionality in which you can tag a complaint as resolved. A checkbox with a check on it means that the issue was already resolved. Two buttons are also found on the message page, namely the Reply button and the Delete button.

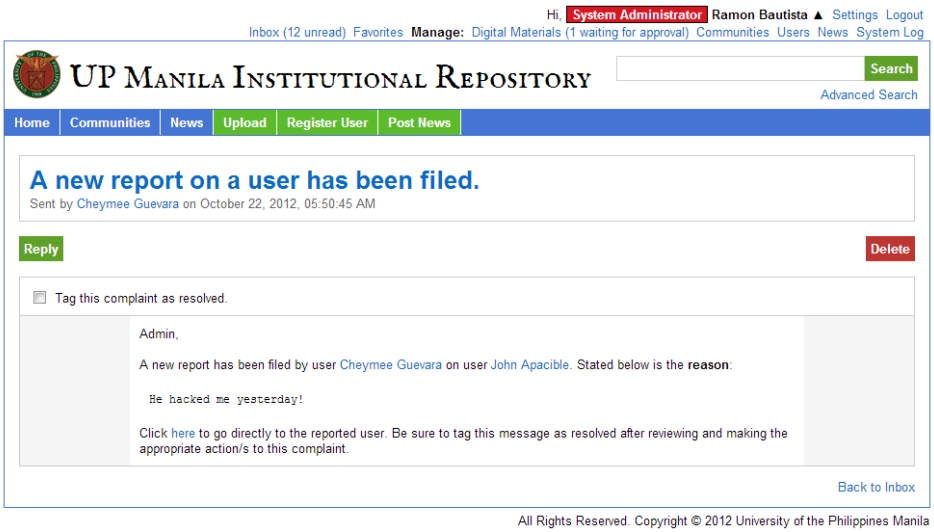

Figure 5.44. Message Page

Clicking Reply is the same as clicking the Compose Message button from the inbox page shown previously, although replying means that the message recipient is already filled up for you. To send a message, make sure that all fields are filled up as they are required fields. Click Send to send the message. Below is an example of a compose message page:

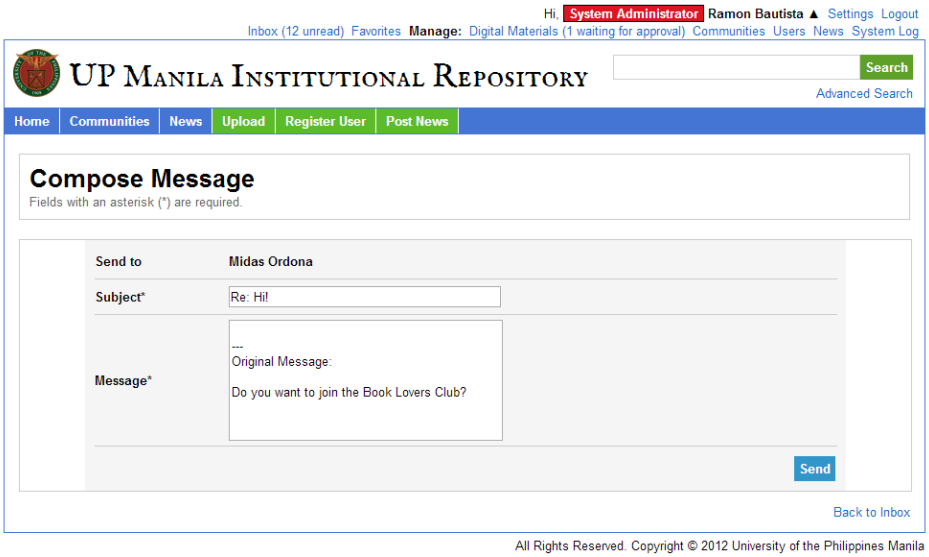

Figure 5.45. Compose Message Page

The manage materials page (Figure 5.46), which is accessed by clicking the Digital Materials link at the upper right corner of the screen, can be viewed by either System Administrator or Community Moderator. System Administrators can access all materials in the system while Community Moderators have access to materials contributed to their respective community. Clicking the Upload Material button will direct the user to the upload page. Below the Upload Material button is a table similar to what we saw in the inbox page. The columns have a built-in sort functionality which triggers on click. Searching the fields in the table is also possible by typing queries in the search box.

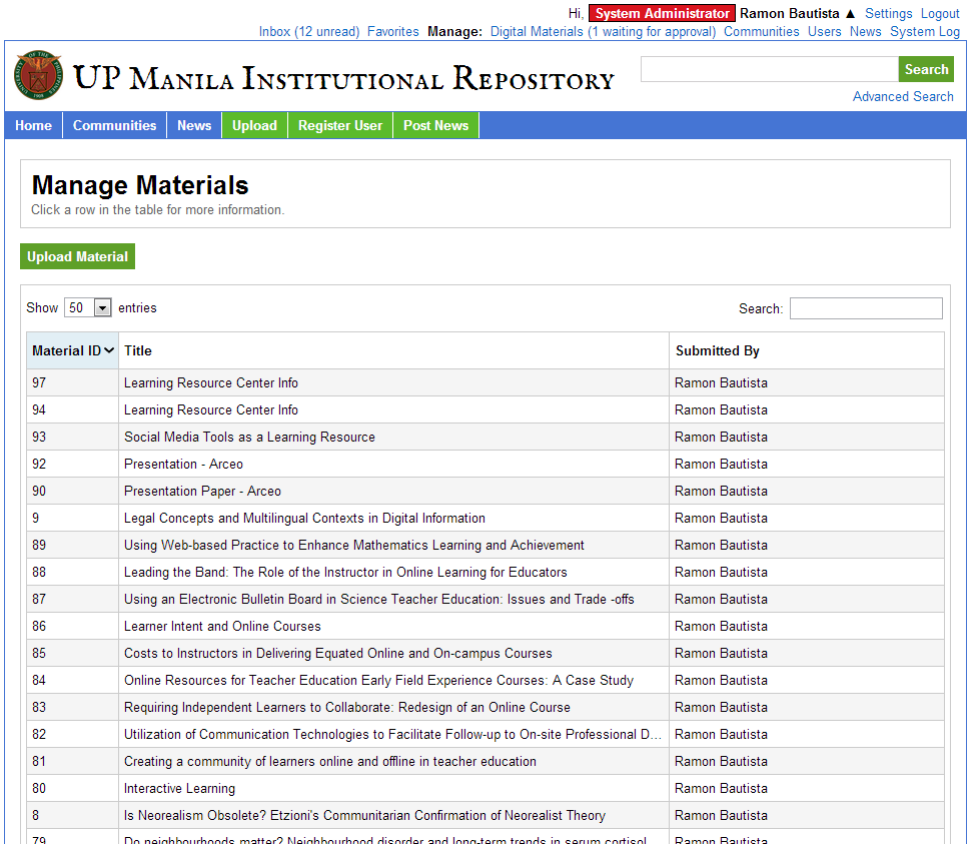

Figure 5.46. Manage Materials Page

Clicking a row in the table will grant access to an entry's material page. For clarity purposes, we will call this page as the administrative material page. For System Administrators and Community Moderators, the material page features several buttons used for material management. Figure 5.47 displays the material page of a text file. In the administrative material page one can annotate, upload a new version of a material, and delete a material.

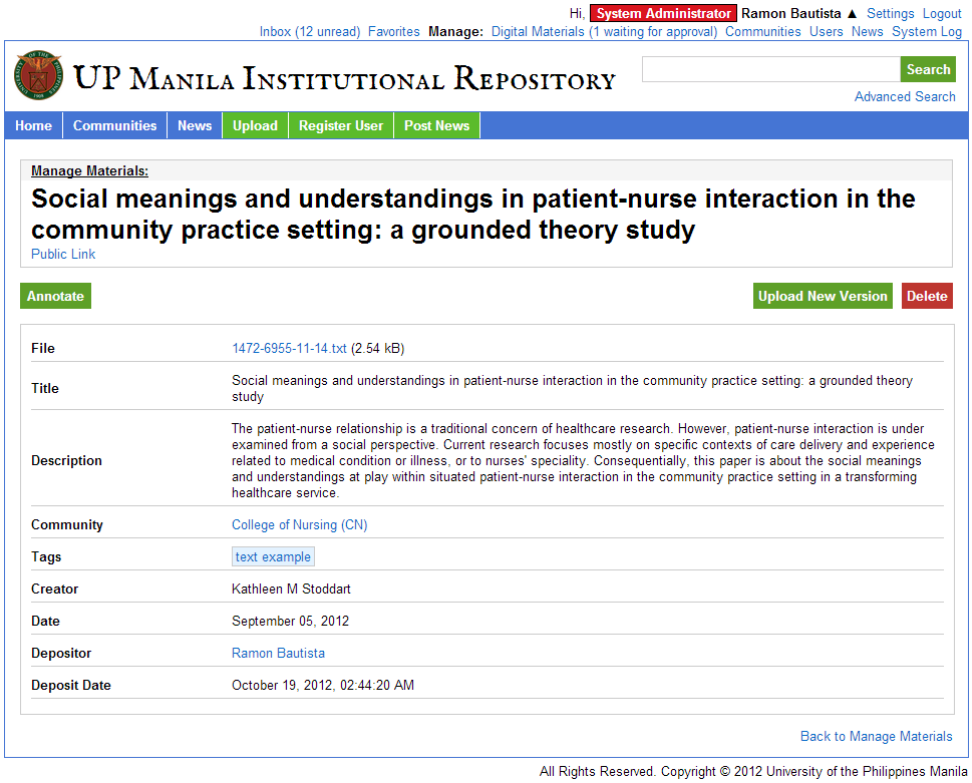

Figure 5.47. Administrative Material Page

Whenever a user submits a material that needs approval from a System Administrator or Community Manager, the pending material link (shown as: # waiting for approval) beside the Digital Materials link at the upper right corner of the screen becomes visible. This link will send the user to the pending materials page (Figure 5.48) where a list of pending materials is shown. The manage materials page and the pending materials page share the same functionalities and layout.

Figure 5.48. Pending Materials Page

Like the manage materials page, clicking on a row show the entry's administrative material page. Here the System Administrator or the Community Moderator can either approve or reject a material. A reason for rejection (Figure 5.49) is required from the user before the material is deleted from the system. On the other hand, accepting the material means that the material will now be visible to the users of the repository system.

Figure 5.49. Reject Material

The user can also manage annotations for text files in the administrative material page. System Administrators and Community Moderators can add, edit, and delete material annotations. There are two ways to approve an annotation. One is by clicking a pending material's Approve link and the other one is by checking the *approve material on edit* (Figure 5.50) checkbox which is available when editing a pending annotation.

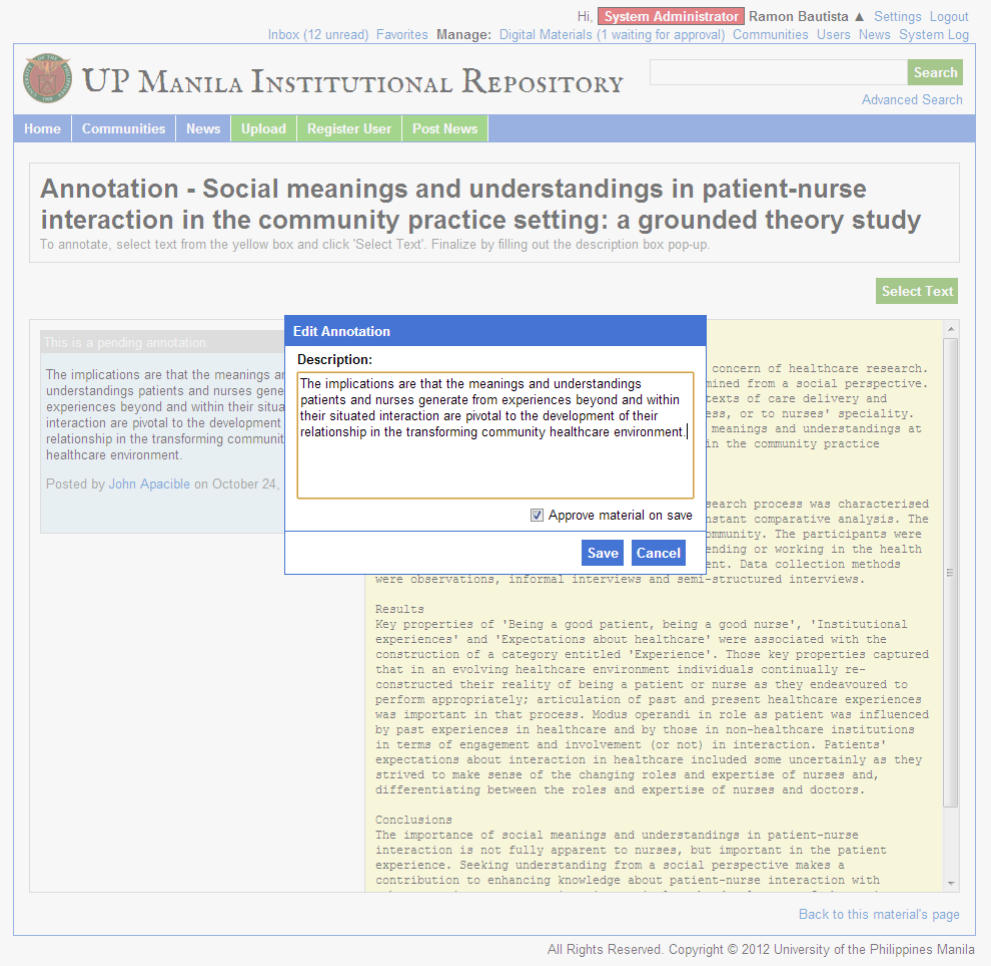

Figure 5.50. Edit Annotation (Approve on Edit Option Visible)

The System Administrator is able to manage communities, users, and news. The Communities link gives access to the community manager page (Figure 5.51).

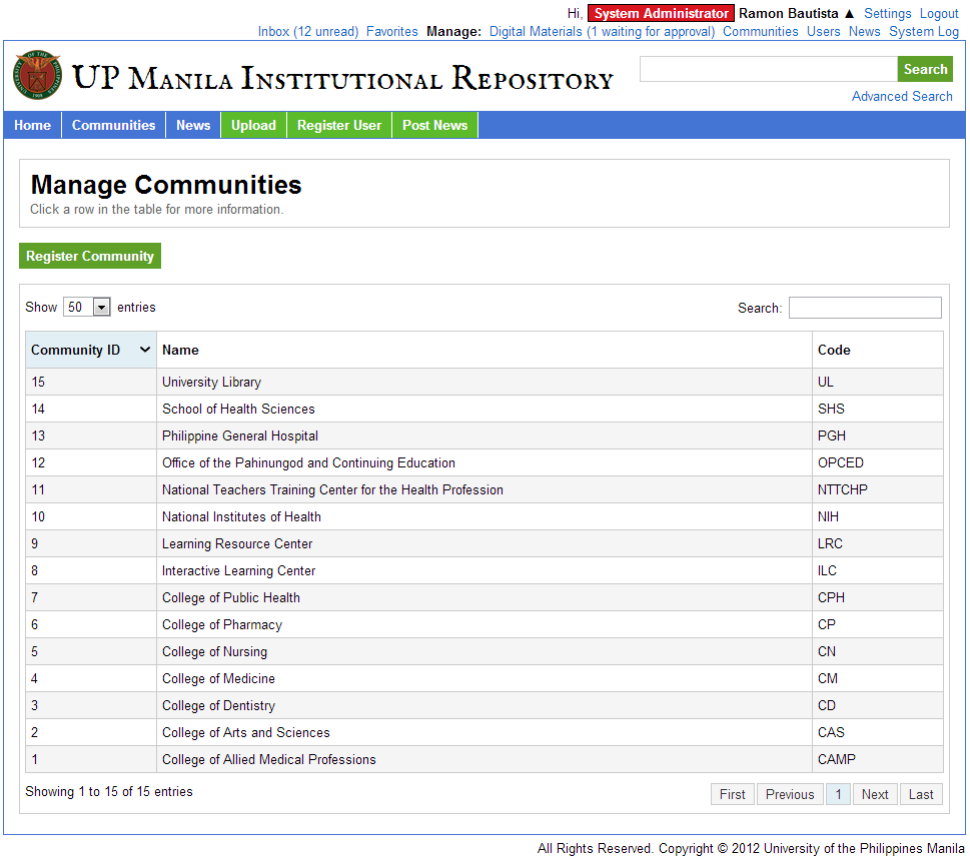

Figure 5.51. Manage Communities Page

Clicking a row will send the user to the community edit page (Figure 5.52) where you can edit a community's description, name, and code.

Figure 5.52. Community Edit Page

Next to Communities, the Users link displays the manage users page.

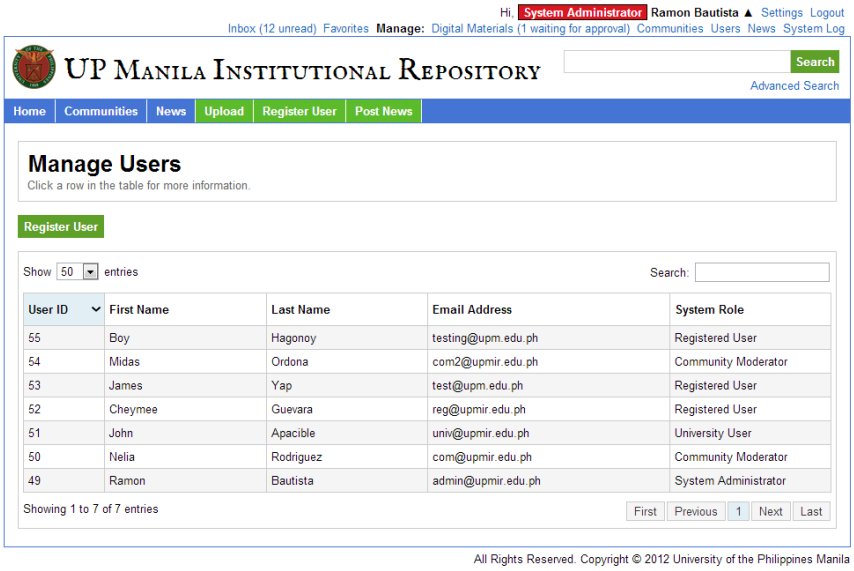

Figure 5.53. Manage Users Page

The Register User link allows the creation of user accounts. Clicking a row in the table while viewing the manage users page displays the user edit page (Figure 5.54).

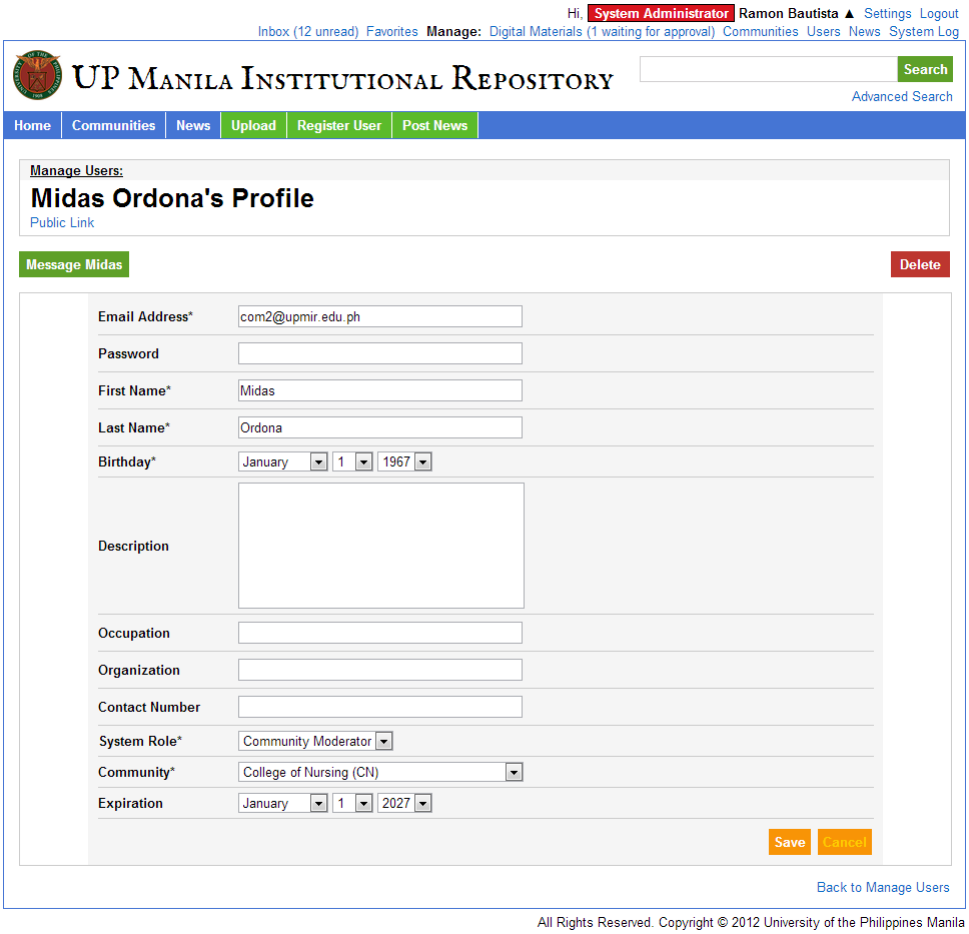

Figure 5.54. User Edit Page

The News link at the top right corner of the site displays a list of news. From this page, you can either create news or manage existing news (Figure 5.55). Clicking on a row in the table lets you edit or delete the selected news (Figure 5.56).

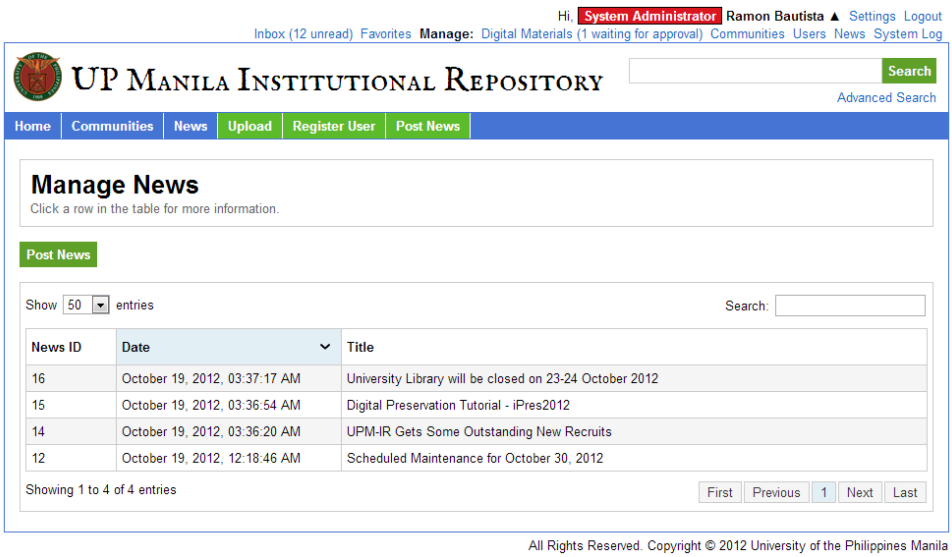

Figure 5.55. Manage News Page

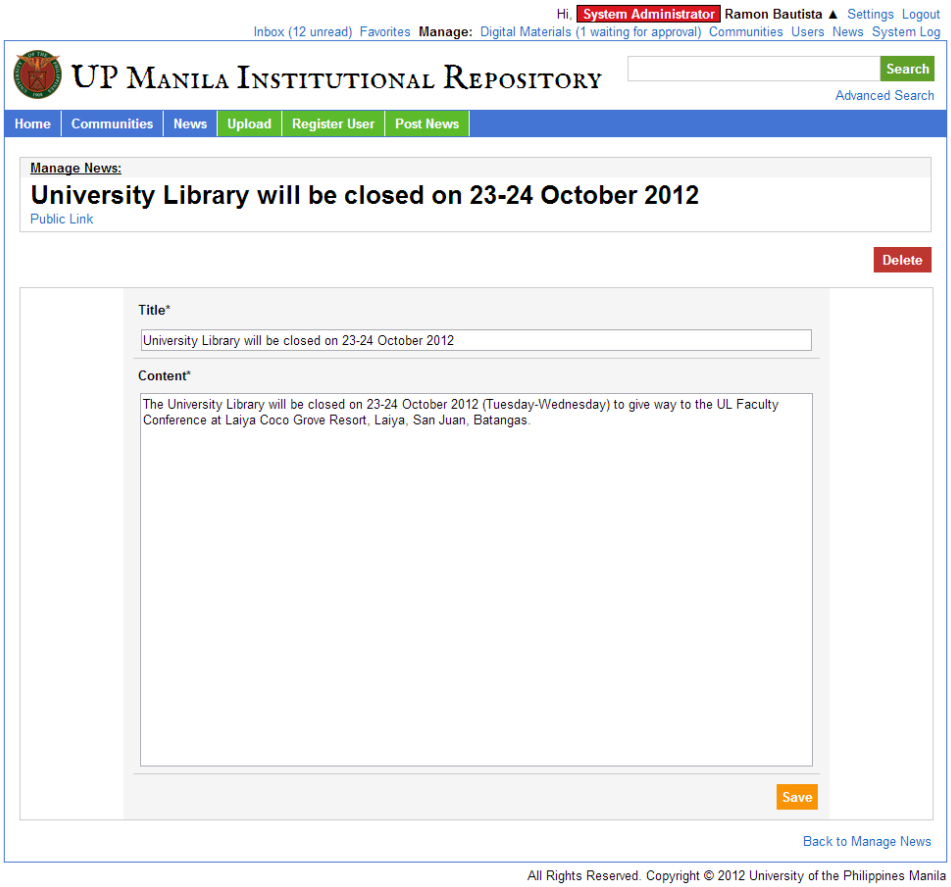

Figure 5.56. News Edit Page

Finally, clicking the System Log link at the top right corner will lead to a page that displays a record of operations made within the system (Figure 5.57). Examples of common operations are adding materials and annotations.

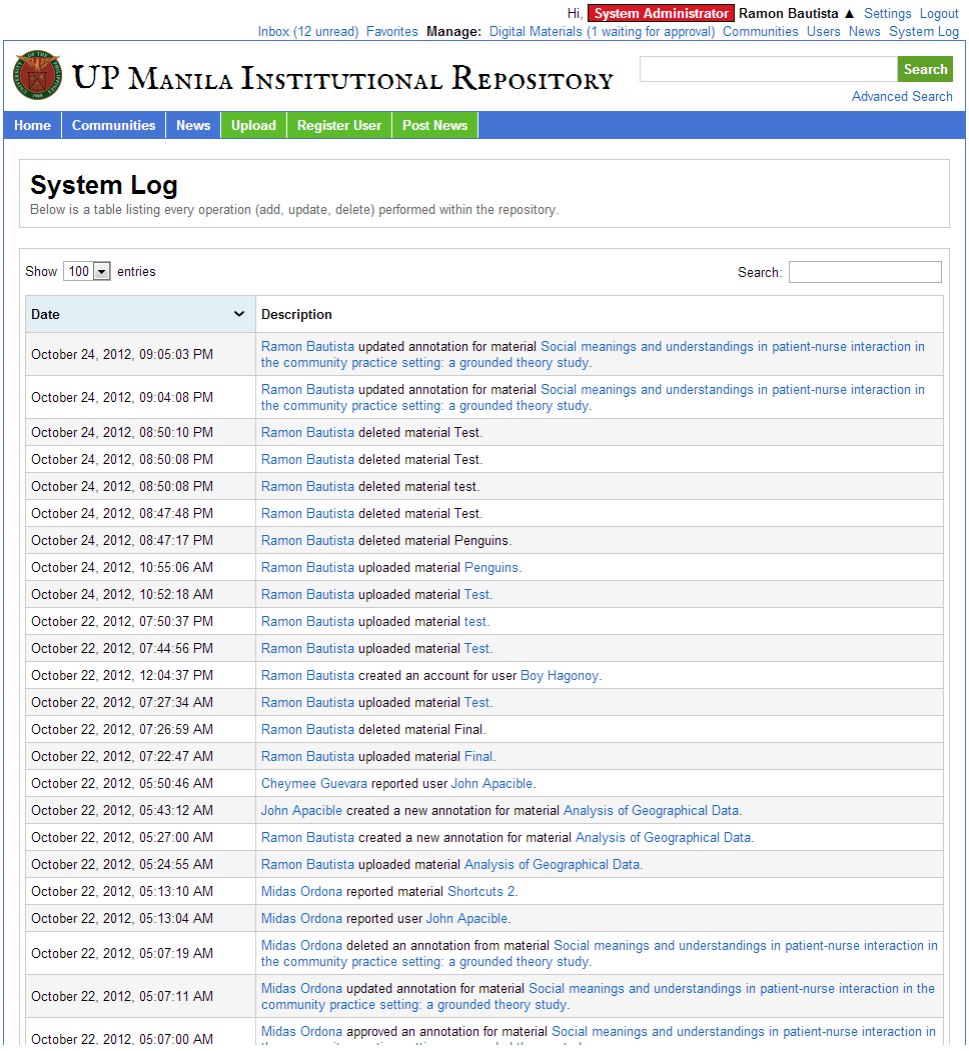

Figure 5.57. System Log Page

VI. Discussion

The purpose of this study is to provide a platform in which digital institutional assets can be efficiently and easily collected and archived. This is enhanced by providing communication tools for user interactions while using the system. For regular users, a normal workflow can be simply summarized as logging in, filling up an upload form and clicking submit. Administrators are then notified of the submissions for them to approve. Once approved, the material is now available for the registered users of the repository.

The system does not solely rely at the hands of the repository administrator. The system administrator can assign moderators to communities to manage. As we've explored, the repository requires and encourages the collaboration between users. Both the administrators and moderators need to properly handle the flow of data while maintaining integrity. Every user in the system is also responsible for the content available publicly. Fortunately, the repository features a messaging feature which allows users to easily report a user or material. This feature also makes it possible for users to contact administrators by making use of the "Contact Us" form.

VII. Conclusion

The implementation of a web-based institutional repository improves a few aspects in the process of archiving and retrieving data. People are no longer required to visit an uploading center on-site (physically) just to be able to contribute materials. The discovery and searching of materials wouldn't be tedious especially if files are indexed and archived collectively in a single repository within an institution. Access to materials means that materials will be ready at any time without limitations (e.g., out of stock).

Though, some may worry about the web-based nature of the repository since every part of the dissemination process can be done exclusively online where user communication and interaction are generally not visible. In fact, the repository does the inverse by providing tools for socializing in the forms of commenting, reporting, and annotating materials.

Finally, the institutional repository serves as an opportunity for long term data preservation. Most of the digital materials created are easily lost. But every institutional asset deserves to be respected and archived properly. Conveniently, all these problems are alleviated (as discussed) with the creation of a repositorial system (presented in this paper) specifically tailored to the institution's requirements.

VIII. Recommendations

As for recommendations, the system could use some customization on some areas particularly on community pages and profile pages. It would help if Community Moderators have the capability to change a community's general look and feel so that users can easily differentiate between communities.

At its current state, the repository system's functionality can only be extended by redeploying another build of the system. By implementing a plugin system, there will be no need for redeployment. The creation of plugins would also be a form of convenience in the testing phases of the repository.

The annotation system can also be improved by providing a means to annotate other file formats. A possible solution would be through the creation of dedicated Java applets which lets the user efficiently select and annotate data.

Similarly, the versioning system could further be extended by providing an interface in which users can actually see the difference between related materials.

Now that more people are shifting towards the utilization of mobile gadgets such as smartphones and tablets, a mobile application that uploads materials through cellular data is recommended. Files stored natively, for example in a user's phone, is just a click away from being stored in the repository.

Lastly, the institutional repository can greatly benefit from the implementation of a file uploader that accepts materials in bulk. To be able to do this, we also need to implement an automatic metadata extractor which will be used to describe various files. The application of the bulk file uploader would also give way to the creation of a convenient material backup and restore functionality for the repository.

IX. References

[1] S. Gulati, "Technology-enhanced learning in developing nations: A review," *International Review of Research in Open and Distance Learning*, vol. 9, no. 1, 2008.

[2] H. Baban and S. Mokhtar, "Online document management system for academic institutes," in *2010 3rd International Conference on Information Management, Innovation Management and Industrial Engineering*, pp. 315–319, IEEE, 2010.

[3] F. Librero, "Digital learning environment in the Philippines: Perspective from the UP Open University," in *Symposium on Digital Learning, Keio University, Tokyo, Japan*, pp. 11–13, 2004.

[4] C. Lynch, "Institutional repositories: Essential infrastructure for the digital age," *Portal*, vol. 3, no. 2, pp. 327–336, 2003.

[5] P. Souto, "E-publishing development and changes in the scholarly communication system," *Ciência da Informação*, vol. 36, no. 1, pp. 158–166, 2007.

[6] K. Garnes, A. Landøy, A. Repanovici, B. Bagge, A. Åsmul, H. Kongshavn, S. Sivertssen, A. Tonning, M. Torras, T. Skagen, *et al.*, *Aspects of the Digital Library*. Alvheim & Eide, 2006.

[7] E. Pacheco, "Academic publishing in the Philippines," in *Academic Publishing in the ASEAN: Problems and Prospects*, pp. 40–55, Chopmen Publishers, 1987.

[8] *http://mainlib.upm.edu.ph/research/*.

[9] *http://ilib.upm.edu.ph/*.

[10] *http://dpsm.cas.upm.edu.ph/~aco/*.

[11] C. Lagoze, S. Payette, E. Shin, and C. Wilper, "Fedora: An architecture for complex objects and their relationships," *International Journal on Digital Libraries*, vol. 6, no. 2, pp. 124–138, 2006.

[12] A. Friedlander, *Dimensions and Use of the Scholarly Information Environment: Introduction to a Data Set*. Digital Library Federation, 2002.

[13] *http://docs.google.com*.

[14] *http://dropbox.com*.

[15] *http://google.com/accounts/TOS?hl=en*.

[16] D. Simberloff, B. Barish, K. Droegemeier, D. Etter, N. Fedoroff, K. Ford, L. Lanzerotti, A. Leshner, J. Lubchenco, M. Rossmann, *et al.*, *Long-lived Digital Data Collections Enabling Research and Education in the 21st Century*. National Science Foundation, 2005.

[17] *http://gutenberg.org/*.

[18] B. Smith, *A quantitative analysis of the impact of e-book format on student acceptance, usage and satisfaction*. Capella University, 2008.

[19] S. Domke, R. Gerlach, and G. Stedman, "Traditional publishers in the digital era," 2006.

[20] *http://commons.wikimedia.org/*.

[21] Wikipedia, "Wikipedia Commons — Wikipedia, the Free Encyclopedia," 2011. [Online; accessed October 21, 2011].

[22] A. Waugh, "The design and implementation of an ingest function to a digital archive," *D-Lib Magazine*, vol. 13, no. 11, pp. 11–12, 2007.

[23] D. Price, "Annotation management in XML content management systems, a case study," in *XML 2002 Conference, Baltimore, MD, December*, pp. 8–12, Citeseer, 2002.

[24] J. Nicholas, "Introduction to Documentum," 2010.

[25] J. Chen, S. Swamidass, Y. Dou, J. Bruand, and P. Baldi, "ChemDB: A public database of small molecules and related chemoinformatics resources," *Bioinformatics*, vol. 21, no. 22, p. 4133, 2005.

[26] R. Patel, M. Hanratty, J. Johnson, P. Saxman, A. Shah, and K. Zheng, "Casepedia: A Web 2.0 case repository enabling collaborative learning," in *AMIA Annual Symposium Proceedings*, 2008.

[27] C. Gay, M. Kayaalp, and A. Aronson, "Semi-automatic indexing of full text biomedical articles," in *AMIA Annual Symposium Proceedings*, vol. 2005, p. 271, American Medical Informatics Association, 2005.

[28] S. Warner, "E-prints and the open archives initiative," *Library Hi Tech*, vol. 21, no. 2, pp. 151–158, 2003.

[29] L. Chan, "Supporting and enhancing scholarship in the digital age: The role of open access institutional repository," *Canadian Journal of Communication*, vol. 29, no. 3, 2004.

[30] C. Lynch and J. Lippincott, "Institutional repository deployment in the United States as of early 2005," *D-Lib Magazine*, vol. 11, no. 9, 2005.

[31] T. Andrew, "Theses Alive!: An e-theses management system for the UK," 2004.

[32] R. Jones, "DSpace vs. ETD-db: Choosing software to manage electronic theses and dissertations," *Ariadne*, no. 38, 2004.

[33] M. Beazley, "EPrints institutional repository software: A review," *Partnership: The Canadian Journal of Library and Information Practice and Research*, vol. 5, no. 2, 2011.

[34] R. Tansley, M. Bass, D. Stuve, M. Branschofsky, D. Chudnov, G. McClellan, and M. Smith, "The DSpace institutional digital repository system: Current functionality," in *Proceedings of the 3rd ACM/IEEE-CS Joint Conference on Digital Libraries*, pp. 87–97, IEEE Computer Society, 2003.

[35] M. Bard, A. Erkan, J. Tyron, and C. Kussmaul, "Deep Blue: A usability assessment," 2006.

[36] W. Nixon, "DAEDALUS: Initial experiences with EPrints and DSpace at the University of Glasgow," *Ariadne*, no. 37, 2003.

[37] G. Van Westrienen and C. Lynch, "Academic institutional repositories," *D-Lib Magazine*, vol. 11, no. 9, 2005.

[38] P. Davis and M. Connolly, "Institutional repositories: Evaluating the reasons for non-use of Cornell University's installation of DSpace," *IR Research*, p. 8, 2007.

[39] S. Gibbons, "Benefits of an institutional repository," *Library Technology Reports*, vol. 40, no. 4, pp. 11–16, 2004.

[40] R. Rossi, *How Writing Began*. Benchmark Books, 2009.

[41] R. Levien and D. Leebaert, "The civilizing currency: Documents and their revolutionary technologies," *Technology 2001: The Future of Computing and Communications*, pp. 205–239, 1991.

[42] S. Keene, "Preserving digital materials: Confronting tomorrow's problems today," *The Conservator*, vol. 26, no. 1, pp. 93–99, 2002.

[43] M. Jones and N. Beagrie, *Preservation Management of Digital Materials: A Handbook*. British Library Publishing, 2001.

[44] M. Wu, "Why print and electronic resources are essential to the academic law library," *Law Library Journal*, vol. 97, p. 233, 2005.

[45] P. Brophy, "Networking in British academic libraries," *British Journal of Academic Librarianship*, vol. 8, no. 1, pp. 49–60, 1993.

[46] M. Hedstrom, "Digital preservation: A time bomb for digital libraries," *Computers and the Humanities*, vol. 31, no. 3, pp. 189–202, 1997.

[47] S. Mao, J. Kim, and G. Thoma, "A dynamic feature generation system for automated metadata extraction in preservation of digital materials," in *Proceedings of the First International Workshop on Document Image Analysis for Libraries (DIAL 2004)*, p. 225, IEEE Computer Society, 2004.

[48] C. Knoblock, "Searching the world wide web," *IEEE Expert: Intelligent Systems and Their Applications*, vol. 12, no. 1, pp. 8–14, 1997.

[49] W. Kerr, "The professor looks at the card catalog," 1943.

[50] C. Taylor, "An introduction to metadata," 2003.

[51] P. Lyman, "Building a national strategy for preservation: Issues in digital media archiving," 2002.

[52] J. Greenberg, M. Pattuelli, B. Parsia, and W. Robertson, "Author-generated Dublin Core metadata for web resources: A baseline study in an organization," *Journal of Digital Information*, vol. 2, no. 2, pp. 38–46, 2001.

[53] S. Weibel and T. Koch, "The Dublin Core metadata initiative," *D-Lib Magazine*, vol. 6, no. 12, pp. 1082–9873, 2000.

[54] S. Darmoni, B. Thirion, J. Leroy, M. Douyere, and J. Piot, "The use of Dublin Core metadata in a structured health resource guide on the internet," *Bulletin of the Medical Library Association*, vol. 89, no. 3, p. 297, 2001.

[55] *http://dublincore.org/documents/dces/*.

[56] K. Muniswamy-Reddy, C. Wright, A. Himmer, and E. Zadok, "A versatile and user-oriented versioning file system," in *Proceedings of the Third USENIX Conference on File and Storage Technologies (FAST 2004)*, pp. 115–128, 2004.

[57] C. Pancerella, J. Hewson, W. Koegler, D. Leahy, M. Lee, L. Rahn, C. Yang, J. Myers, B. Didier, R. McCoy, *et al.*, "Metadata in the collaboratory for multi-scale chemical science," in *Proceedings of the 2003 International Conference on Dublin Core and Metadata Applications*, pp. 1–9, Dublin Core Metadata Initiative, 2003.

[58] J. Brace, "Versioning in repositories: Implementing best practice," *Ariadne*, vol. 56, 2008.

[59] R. Crow, "The case for institutional repositories: A SPARC position paper," *ARL Bimonthly Report 223*, 2002.

[60] P. Wells and P. Wells, "Institutional repositories: Investigating user groups and comparative evaluation using link analysis," *Unpublished)[Thesis]*, 2009.

[61] C. Jones, "Institutional repositories," in *Aspects of the Digital Library*, Alvheim & Eide, 2006.

[62] F. Tutorial, "Introduction to Fedora," 2005.

[63] F. Tutorial, "Getting started: Creating Fedora objects and using disseminators," 2005.

[64] C. Decurtins, M. Norrie, and B. Signer, "Digital annotation of printed documents," in *Proceedings of the 12th International Conference on Information and Knowledge Management*, pp. 552–555, ACM, 2003.

[65] P. Bottoni, R. Civica, S. Levialdi, L. Orso, E. Panizzi, and R. Trinchese, "MADCOW: A multimedia digital annotation system," in *Proceedings of the Working Conference on Advanced Visual Interfaces*, pp. 55–62, ACM, 2004.

[66] R. Sandhu and P. Samarati, "Access control: Principle and practice," *Communications Magazine, IEEE*, vol. 32, no. 9, pp. 40–48, 1994.

[67] J. Lopez, R. Oppliger, and G. Pernul, "Authentication and authorization infrastructures (AAIs): A comparative survey," *Computers & Security*, vol. 23, no. 7, pp. 578–590, 2004.

[68] R. Sandhu, E. Coyne, H. Feinstein, and C. Youman, "Role-based access control models," *Computer*, vol. 29, no. 2, pp. 38–47, 1996.

[69] D. Ferraiolo, J. Cugini, and D. Kuhn, "Role-based access control (RBAC): Features and motivations," in *Proceedings of 11th Annual Computer Security Application Conference*, pp. 11–15, 1995.

[70] M. Ferroiolo, M. Gilbert, and M. Lynch, "An examination of federal and commercial access control policy needs," in *National Computer Security Conference, 16th Proceedings: Information Systems Security: User Choices*, p. 107, DIANE Publishing, 1995.

[71] M. Kamel Boulos and S. Wheeler, "The emerging Web 2.0 social software: An enabling suite of sociable technologies in health and health care education," *Health Information & Libraries Journal*, vol. 24, no. 1, pp. 2–23, 2007.

[72] M. Parameswaran and A. Whinston, "Research issues in social computing," *Journal of the Association for Information Systems*, vol. 8, no. 6, pp. 336–350, 2007.

[73] D. McGuinness, H. Zeng, P. Da Silva, L. Ding, and M. Dhyanesh Narayanan, "Investigations into trust for collaborative information repositories: A Wikipedia case study," *Models of Trust for the Web (MTW '06)*, 2006.

X. Appendix

AcceptAnnotation.java

```
package admin;

import java.io.IOException;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;

import data.AnnotationDB;
import data.OperationDB;
import domain.Annotation;
import domain.Material;
import domain.User;

public class AcceptAnnotation extends HttpServlet {
  public void doPost(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
    if (request.getParameter("material_id") == null ||
request.getParameter("annotation_id") == null) {
      response.sendRedirect(request.getContextPath() + "/admin/m");

      return;
    }
    else {
      HttpSession session = request.getSession();

      int i =
AnnotationDB.approve(Long.parseLong(request.getParameter("annotat
ion_id")));

      if (i > 0) {
        User user = (User)session.getAttribute("me");
        Annotation annotation =
AnnotationDB.select(Long.parseLong(request.getParameter("annotatio
n_id")));
        Material material = annotation.getMaterial();

        String op = "<a href='admin/u/view?id=" + user.getID() + "'>" +
user.getFirstName() + " " + user.getLastName() + "</a> approved an
annotation for material <a href='admin/m/view?id=" + material.getID() +
"'>" + material.getMetadata().getTitle() + "</a>.";
        OperationDB.insert(op);

        session.setAttribute("submit_annotation", "true");
      }
      else {
        session.setAttribute("submit_annotation", "false");
      }
    }

    response.sendRedirect(request.getContextPath() +
"/admin/m/annotate?id=" + request.getParameter("material_id"));
  }

  public void doGet(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
    response.sendRedirect(request.getContextPath() + "/admin/m");
  }
}
```

AcceptMaterial.java

```
package admin;

import java.io.IOException;
import java.util.StringTokenizer;

import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;

import data.MaterialDB;
import data.OperationDB;
import data.TagDB;
import domain.Material;
import domain.User;

public class AcceptMaterial extends HttpServlet {
  public void doPost(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
    if (request.getParameter("material_id") == null) {
      response.sendRedirect(request.getContextPath() +
"/admin/m/pending");

      return;
    }
    else {
      String pid = request.getParameter("material_id");
      HttpSession session = request.getSession();

      int i = 0;

      if (MaterialDB.materialExists(pid)) {
        Material material = MaterialDB.select(pid);

        // Insert tags to the database.
        String tags = material.getTags();
        StringTokenizer st = new StringTokenizer(tags, ",");

        while (st.hasMoreTokens()) {
          String val = st.nextToken().trim();

          // Insert tag if it does not exist
          if (!TagDB.tagExists(val)) {
            TagDB.insert(val);
          }
        }

        // Approve pending submission.
        i = MaterialDB.approve(pid);

        User user = (User)session.getAttribute("me");

        String op = "<a href='admin/u/view?id=" + user.getID() + "'>" +
user.getFirstName() + " " + user.getLastName() + "</a> has approved
```

```
pending material <a href='admin/m/view?id=" + material.getID() + "'>" +
material.getMetadata().getTitle() + "</a>.";
       OperationDB.insert(op);
     }

     if (i > 0) {
       session.setAttribute("approve_material", "true");
     }
     else {
       session.setAttribute("approve_material", "false");
     }
   }

    response.sendRedirect(request.getContextPath() +
"/admin/m/pending");
  }

  public void doGet(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
    response.sendRedirect(request.getContextPath() +
"/admin/m/pending");
  }
}
```

DeleteAnnotation.java

```
package admin;

import java.io.IOException;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;

import data.AnnotationDB;
import data.OperationDB;
import domain.Annotation;
import domain.Material;
import domain.User;

public class DeleteAnnotation extends HttpServlet {
  public void doPost(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
    HttpSession session = request.getSession();
    String material_id = "";

    if (request.getParameter("annotation_id") == null) {
      session.setAttribute("delete_annotation", "false");
    }
    else {
      int i = 0;
      Annotation annotation =
AnnotationDB.select(Long.parseLong(request.getParameter("annotatio
n_id")));
      Material material = annotation.getMaterial();

      i =
AnnotationDB.delete(Long.parseLong(request.getParameter("annotatio
n_id")));

      if (i > 0) {
        User user = (User)session.getAttribute("me");

        String op = "<a href='admin/u/view?id=" + user.getID() + "'>" +
user.getFirstName() + " " + user.getLastName() + "</a> deleted an
annotation from material <a href='admin/m/view?id=" + material.getID()
+ "'>" + material.getMetadata().getTitle() + "</a>.";
        OperationDB.insert(op);

        session.setAttribute("delete_annotation", "true");
      }
      else {
        session.setAttribute("delete_annotation", "false");
      }

      material_id = request.getParameter("material_id");
    }

    response.sendRedirect(request.getContextPath() +
"/admin/m/annotate?id=" + material_id);
  }

  public void doGet(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
    // Do nothing.
  }
}
```

DeleteCommunity.java

```
package admin;

import java.io.IOException;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import data.CommunityDB;

public class DeleteCommunity extends HttpServlet {
  public void doPost(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
    if (request.getParameter("code") == null) {
      response.sendRedirect(request.getContextPath() + "/admin/c");

      return;
    }
    else {
      if (CommunityDB.communityExists(request.getParameter("code")))
{
        CommunityDB.delete(request.getParameter("code"));
      }

      response.sendRedirect(request.getContextPath() + "/admin/c");
    }
  }
}
```

DeleteMaterial.java

```
package admin;
```

```
import java.io.File;
import java.io.IOException;

import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;

import data.FavoriteDB;
import data.MaterialDB;
import data.MaterialFedora;
import data.MessageDB;
import data.OperationDB;
import domain.Material;
import domain.Message;
import domain.User;

public class DeleteMaterial extends HttpServlet {
  public void doPost(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
    HttpSession session = request.getSession();
    Material material = null;

    if (request.getParameter("is_approved").equals("n")) {
      validateMessage(request);
    }

    if (request.getParameter("material_id") == null) {
      session.setAttribute("delete_material", "false");
    }
    else {
      String pid = request.getParameter("material_id");

      material = MaterialDB.select(pid);

      // Delete thumbnail.
      String thumbnail = material.getThumbnail();

      if (thumbnail == null || thumbnail.isEmpty()) {
        // No thumbnail
      }
      else {
        File file = new File("/home/aarceo/thumbnail",
material.getThumbnail());
        file.delete();
      }

      MaterialDB.delete(pid);
      FavoriteDB.delete(pid);

      User user = (User)session.getAttribute("me");

      if (MaterialFedora.materialExists(pid)) {
        MaterialFedora.delete(pid);
      }

      if (!MaterialDB.materialExists(pid) &&
!MaterialFedora.materialExists(pid)) {
        session.setAttribute("delete_material", "true");

        String action = "deleted";

        if (request.getParameter("is_approved").equals("n")) {
          User sender = (User)session.getAttribute("me");

          Message message = new Message();
          message.setSubject("A material you have submitted has been
rejected.");

          String content = "To " + material.getUser().getFirstName() + " " +
material.getUser().getLastName() + ":<br><br>" +
              "We are sorry to inform you that your file submission made
on " + material.getDepositTimestampLong() + ", titled <strong>" +
material.getMetadata().getTitle() + "</strong>, " +
              "has been rejected. Below is the reason:" +
              "<div class=\"read_message_report_description\">" +
request.getParameter("reject_message") + "</div>";

          message.setContent(content);
          message.setSender(sender);
          message.setRecipient(material.getUser());
          message.setType("attention");

          MessageDB.insert(message);

          action = "rejected";
        }

        String op = "<a href='admin/u/view?id=" + user.getID() + "'>" +
user.getFirstName() + " " + user.getLastName() + "</a> " + action + "
material " + material.getMetadata().getTitle() + ".";
        OperationDB.insert(op);
      }
      else {
        session.setAttribute("delete_material", "false");
      }
    }

    if (request.getParameter("is_approved").equals("n")) {
      response.sendRedirect(request.getContextPath() +
"/admin/m/pending");
    }
    else {
      response.sendRedirect(request.getContextPath() + "/admin/m");
    }
  }

  public void doGet(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
    response.sendRedirect(request.getContextPath() + "/admin/m");
  }

  private void validateMessage(HttpServletRequest request) throws
ServletException {
    // Message
    if (request.getParameter("reject_message") == null ||
request.getParameter("reject_message").trim().length() > 5000) {
      if (request.getParameter("reject_message").trim().length() < 1) {
        throw new ServletException("Validation: Please specify the reason
for rejecting the material.");
      }
      else {
```

```
      throw new ServletException("Validation: Your reason for rejecting
the material exceeds character limit of 3000.");
    }
   }
  }
}
```

DeleteNews.java

```
package admin;

import java.io.IOException;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;

import data.NewsDB;
import data.OperationDB;
import domain.News;
import domain.User;

public class DeleteNews extends HttpServlet {
  public void doPost(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
    HttpSession session = request.getSession();

    if (request.getParameter("reference") == null) {
      session.setAttribute("delete_news", "false");
    }
    else {
      int i = 0;

      News news =
NewsDB.select(request.getParameter("reference").trim());

      i = NewsDB.delete(news.getReference());

      if (i > 0) {
        User user = (User)session.getAttribute("me");

        String op = "<a href='admin/u/view?id=" + user.getID() + "'>" +
user.getFirstName() + " " + user.getLastName() + "</a> deleted news "
+ news.getTitle() + ".";
        OperationDB.insert(op);

        session.setAttribute("processed_news", news);
        session.setAttribute("delete_news", "true");
      }
      else {
        session.setAttribute("delete_news", "false");
      }
    }

    response.sendRedirect(request.getContextPath() + "/admin/news");
  }

  public void doGet(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
    response.sendRedirect(request.getContextPath() + "/admin/news");
  }
}
```

DeleteUser.java

```
package admin;

import java.io.IOException;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;

import data.OperationDB;
import data.UserDB;
import domain.User;

public class DeleteUser extends HttpServlet {
  public void doPost(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
    HttpSession session = request.getSession();

    if (request.getParameter("user_id") == null) {
      session.setAttribute("delete_user", "false");
    }
    else {
      int i = 0;

      User user =
UserDB.select(Long.parseLong(request.getParameter("user_id")));

      i =
UserDB.delete(Long.parseLong(request.getParameter("user_id")));

      if (i > 0) {
        User me = (User)session.getAttribute("me");

        String op = "<a href='admin/u/view?id=" + me.getID() + "'>" +
me.getFirstName() + " " + me.getLastName() + "</a> deleted user " +
user.getFirstName() + " " + user.getLastName() + "</a>.";
        OperationDB.insert(op);

        session.setAttribute("processed_user", user);
        session.setAttribute("delete_user", "true");
      }
      else {
        session.setAttribute("delete_user", "false");
      }
    }

    response.sendRedirect(request.getContextPath() + "/admin/u");
  }

  public void doGet(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
    response.sendRedirect(request.getContextPath() + "/admin/u");
  }
}
```

DisplayAnnotation.java

```
package admin;
```

```
import java.io.IOException;
import java.io.InputStream;
import java.util.LinkedHashMap;

import javax.servlet.RequestDispatcher;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;

import org.apache.commons.io.IOUtils;
import org.apache.commons.io.input.BOMInputStream;

import com.yourmediashelf.fedora.client.FedoraClient;
import com.yourmediashelf.fedora.client.FedoraClientException;
import
com.yourmediashelf.fedora.client.request.GetDatastreamDissemination
;
import com.yourmediashelf.fedora.client.response.FedoraResponse;

import data.AnnotationDB;
import data.FedoraConnect;
import data.MaterialDB;
import domain.Annotation;
import domain.Material;
import domain.User;

public class DisplayAnnotation extends HttpServlet {
  public void doGet(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
    if (request.getParameter("id") == null) {
      response.sendRedirect(request.getContextPath() + "/admin/m");

      return;
    }
    else {
      if (MaterialDB.materialExists("get:" + request.getParameter("id"))) {
        String pid = "get:" + request.getParameter("id");

        Material material = MaterialDB.select(pid);

        FedoraConnect connect = new FedoraConnect();
        FedoraClient client = connect.getClient();
        FedoraResponse fedora_response = null;

        InputStream data = null;

        try {
          fedora_response = new
GetDatastreamDissemination(material.getPID(),
"DATA").execute(client);
          data = fedora_response.getEntityInputStream();
        }
        catch (FedoraClientException e) {
          e.printStackTrace();
        }

        BOMInputStream input = new BOMInputStream(data);
        String contents = IOUtils.toString(input, "UTF-8");

        request.setAttribute("m", material);
        request.setAttribute("contents", contents);

        HttpSession session = request.getSession();
        User user = (User)session.getAttribute("me");

        // Get annotations.
        LinkedHashMap<Long, Annotation> annotation_list =
AnnotationDB.selectMaterialAnnotations(material.getPID(), user);

        if (annotation_list != null && annotation_list.size() > 0) {
          request.setAttribute("al", annotation_list);
        }
      }
      else {
        response.sendRedirect(request.getContextPath() + "/admin/m");

        return;
      }

      String url = "/WEB-INF/admin/m/annotate.jsp";
      RequestDispatcher dispatcher =
getServletContext().getRequestDispatcher(url);
      dispatcher.forward(request, response);
    }
  }

  public void doPost(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
    doGet(request, response);
  }
}
```

DisplayCommunity.java

```
package admin;

import java.io.IOException;

import javax.servlet.RequestDispatcher;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import data.CommunityDB;
import domain.Community;

public class DisplayCommunity extends HttpServlet {
  public void doGet(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
    String code = request.getPathInfo();
    code = code.substring(1);

    if (code != null && code.length() > 0 && !code.equals("/")) {
      if (CommunityDB.communityExists(code)) {
        Community community = CommunityDB.select(code);
        request.setAttribute("c", community);

        String url = "/WEB-INF/admin/c/view.jsp";
        RequestDispatcher dispatcher =
getServletContext().getRequestDispatcher(url);
        dispatcher.forward(request, response);
```

```
      return;
    }
   }

   response.sendRedirect(request.getContextPath() + "/admin/c");
  }

  public void doPost(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
    doGet(request, response);
  }
}
```

DisplayCommunityList.java

```
package admin;

import java.io.IOException;
import java.util.ArrayList;

import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import com.google.gson.Gson;

import data.CommunityDB;
import data.SQLUtil;
import domain.Community;

public class DisplayCommunityList extends HttpServlet {
  public void doGet(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
    String[] cols = {"community_id", "name", "code"};
    String filter = "";
    String order = "";
    String limit = "";

    // Filtering
    filter = SQLUtil.getFilter(request, cols);

    // Ordering
    order = SQLUtil.getOrder(request, cols);

    // Paging
    limit = SQLUtil.getLimit(request);

    CommunityJson cj = new CommunityJson();
    cj.setData(CommunityDB.select(filter, order, limit));
    cj.setEcho(Integer.parseInt(request.getParameter("sEcho")));
    cj.setTotalRecords(CommunityDB.count());
    cj.setTotalDisplayRecords(CommunityDB.countFilter(filter));

    response.setContentType("application/json");
    response.setCharacterEncoding("UTF-8");
    response.getWriter().write(new Gson().toJson(cj));
  }

  public void doPost(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
    doGet(request, response);
  }
}

class CommunityJson {
  private int sEcho;
  private int iTotalRecords;
  private int iTotalDisplayRecords;
  private String[][] aaData;

  public CommunityJson() {
    sEcho = 0;
    iTotalRecords = 0;
    iTotalDisplayRecords = 0;
    aaData = null;
  }

  public void setEcho(int sEcho) {
    this.sEcho = sEcho;
  }

  public int getEcho() {
    return sEcho;
  }

  public void setTotalRecords(int iTotalRecords) {
    this.iTotalRecords = iTotalRecords;
  }

  public int getTotalRecords() {
    return iTotalRecords;
  }

  public void setTotalDisplayRecords(int iTotalDisplayRecords) {
    this.iTotalDisplayRecords = iTotalDisplayRecords;
  }

  public int getTotalDisplayRecords() {
    return iTotalDisplayRecords;
  }

  public void setData(ArrayList<Community> community_list) {
    if (community_list != null) {
      aaData = new String[community_list.size()][3];

      for (int i = 0; i < community_list.size(); i++) {
        aaData[i][0] =
String.valueOf(CommunityDB.selectID(community_list.get(i).getCode())
);
        aaData[i][1] = community_list.get(i).getName();
        aaData[i][2] = community_list.get(i).getCode();
      }
    }
    else {
      aaData = null;
    }
  }

  public String[][] getData() {
    return aaData;
  }
}
```

DisplayMaterial.java

```
package admin;

import java.io.IOException;

import javax.servlet.RequestDispatcher;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import data.MaterialDB;
import domain.Material;

public class DisplayMaterial extends HttpServlet {
  public void doGet(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
    if (request.getParameter("id") == null) {
      response.sendRedirect(request.getContextPath() + "/admin/m");

      return;
    }
    else {
      if (MaterialDB.materialExists("get:" + request.getParameter("id"))) {
        Material material = MaterialDB.select("get:" +
request.getParameter("id"));
        request.setAttribute("m", material);

        if (MaterialDB.materialExistsAndApproved(material.getPID())) {
          request.setAttribute("is_approved", "y");
        }
        else {
          request.setAttribute("is_approved", "n");
        }
      }
      else {
        response.sendRedirect(request.getContextPath() + "/admin/m");

        return;
      }

      String url = "/WEB-INF/admin/m/view.jsp";
      RequestDispatcher dispatcher =
getServletContext().getRequestDispatcher(url);
      dispatcher.forward(request, response);
    }
  }

  public void doPost(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
    doGet(request, response);
  }
}
```

DisplayMaterialList.java

```
package admin;

import java.io.IOException;
import java.util.ArrayList;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import com.google.gson.Gson;

import data.MaterialDB;
import data.SQLUtil;
import domain.Material;

public class DisplayMaterialList extends HttpServlet {
  public void doGet(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
    String[] cols = {"pid", "dc_title", "CONCAT(first_name, ' ',
last_name)"};
    String filter = "";
    String order = "";
    String limit = "";

    // Filtering
    filter = SQLUtil.getFilter(request, cols);

    // Ordering
    order = SQLUtil.getOrder(request, cols);

    // Paging
    limit = SQLUtil.getLimit(request);

    MaterialJson mj = new MaterialJson();
    mj.setData(MaterialDB.selectApproved(filter, order, limit));
    mj.setEcho(Integer.parseInt(request.getParameter("sEcho")));
    mj.setTotalRecords(MaterialDB.countApproved());
    mj.setTotalDisplayRecords(MaterialDB.countApprovedFilter(filter));

    response.setContentType("application/json");
    response.setCharacterEncoding("UTF-8");
    response.getWriter().write(new Gson().toJson(mj));
  }

  public void doPost(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
    doGet(request, response);
  }
}

class MaterialJson {
  private int sEcho;
  private int iTotalRecords;
  private int iTotalDisplayRecords;
  private String[][] aaData;

  public MaterialJson() {
    sEcho = 0;
    iTotalRecords = 0;
    iTotalDisplayRecords = 0;
    aaData = null;
  }

  public void setEcho(int sEcho) {
    this.sEcho = sEcho;
```

```
  }

  public int getEcho() {
    return sEcho;
  }

  public void setTotalRecords(int iTotalRecords) {
    this.iTotalRecords = iTotalRecords;
  }

  public int getTotalRecords() {
    return iTotalRecords;
  }

  public void setTotalDisplayRecords(int iTotalDisplayRecords) {
    this.iTotalDisplayRecords = iTotalDisplayRecords;
  }

  public int getTotalDisplayRecords() {
    return iTotalDisplayRecords;
  }

  public void setData(ArrayList<Material> material_list) {
    if (material_list != null) {
      aaData = new String[material_list.size()][3];

      for (int i = 0; i < material_list.size(); i++) {
        aaData[i][0] = material_list.get(i).getID();
        aaData[i][1] = material_list.get(i).getMetadata().getTitle();
        aaData[i][2] = material_list.get(i).getUser().getFirstName() + " " +
material_list.get(i).getUser().getLastName();
      }
    }
    else {
      aaData = null;
    }
  }

  public String[][] getData() {
    return aaData;
  }
}
```

DisplayNews.java

```
package admin;

import java.io.IOException;

import javax.servlet.RequestDispatcher;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import data.NewsDB;
import domain.News;

public class DisplayNews extends HttpServlet {
  public void doGet(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
    String reference = request.getPathInfo();
    reference = reference.substring(1);

    if (reference != null && reference.length() > 0 &&
!reference.equals("/")) {
      if (NewsDB.referenceExists(reference)) {
        News news = NewsDB.select(reference);
        request.setAttribute("n", news);

        String url = "/WEB-INF/admin/news/view.jsp";
        RequestDispatcher dispatcher =
getServletContext().getRequestDispatcher(url);
        dispatcher.forward(request, response);

        return;
      }
      else {
        while (reference.indexOf("/") >= 0) {
          reference = reference.substring(0, reference.indexOf("/"));

          if (NewsDB.referenceExists(reference)) {
            News news = NewsDB.select(reference);
            request.setAttribute("n", news);

            String url = "/WEB-INF/admin/news/view.jsp";
            RequestDispatcher dispatcher =
getServletContext().getRequestDispatcher(url);
            dispatcher.forward(request, response);

            return;
          }
        }
      }
    }

    response.sendRedirect(request.getContextPath() + "/admin/news");
  }

  public void doPost(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
    doGet(request, response);
  }
}
```

DisplayNewsList.java

```
package admin;

import java.io.IOException;
import java.util.ArrayList;

import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import com.google.gson.Gson;

import data.NewsDB;
import data.SQLUtil;
import domain.News;

public class DisplayNewsList extends HttpServlet {
```

```
  public void doGet(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
    String[] cols = {"news_id", "news_timestamp", "title", "reference",
"DATE_FORMAT(news_timestamp, '%M %e, %Y, %k:%i:%s')"};
    String filter = "";
    String order = "";
    String limit = "";

    // Filtering
    filter = SQLUtil.getFilter(request, cols);

    // Ordering
    order = SQLUtil.getOrder(request, cols);

    // Paging
    limit = SQLUtil.getLimit(request);

    NewsJson nj = new NewsJson();
    nj.setData(NewsDB.select(filter, order, limit));
    nj.setEcho(Integer.parseInt(request.getParameter("sEcho")));
    nj.setTotalRecords(NewsDB.count());
    nj.setTotalDisplayRecords(NewsDB.countFilter(filter));

    response.setContentType("application/json");
    response.setCharacterEncoding("UTF-8");
    response.getWriter().write(new Gson().toJson(nj));
  }

  public void doPost(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
    doGet(request, response);
  }
}

class NewsJson {
  private int sEcho;
  private int iTotalRecords;
  private int iTotalDisplayRecords;
  private String[][] aaData;

  public NewsJson() {
    sEcho = 0;
    iTotalRecords = 0;
    iTotalDisplayRecords = 0;
    aaData = null;
  }

  public void setEcho(int sEcho) {
    this.sEcho = sEcho;
  }

  public int getEcho() {
    return sEcho;
  }

  public void setTotalRecords(int iTotalRecords) {
    this.iTotalRecords = iTotalRecords;
  }

  public int getTotalRecords() {
    return iTotalRecords;
  }

  public void setTotalDisplayRecords(int iTotalDisplayRecords) {
    this.iTotalDisplayRecords = iTotalDisplayRecords;
  }

  public int getTotalDisplayRecords() {
    return iTotalDisplayRecords;
  }

  public void setData(ArrayList<News> news_list) {
    if (news_list != null) {
      aaData = new String[news_list.size()][5];

      for (int i = 0; i < news_list.size(); i++) {
        aaData[i][0] =
String.valueOf(NewsDB.selectID(news_list.get(i).getReference()));
        aaData[i][1] = news_list.get(i).getNewsTimestampLong();
        aaData[i][2] = news_list.get(i).getTitle();
        aaData[i][3] = news_list.get(i).getReference();
        aaData[i][4] = news_list.get(i).getNewsTimestampLong();
      }
    }
    else {
      aaData = null;
    }
  }

  public String[][] getData() {
    return aaData;
  }
}
```

DisplayOperationList.java

```
package admin;

import java.io.IOException;
import java.util.ArrayList;

import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import com.google.gson.Gson;

import data.OperationDB;
import data.SQLUtil;
import domain.Operation;


public class DisplayOperationList extends HttpServlet {
  public void doGet(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
    String[] cols = {"operation_id", "description",
"DATE_FORMAT(operation_timestamp, '%M %e, %Y, %k:%i:%s')"};
    String filter = "";
    String order = "";
    String limit = "";

    // Filtering
    filter = SQLUtil.getFilter(request, cols);
```

```
    // Ordering
    order = SQLUtil.getOrder(request, cols);

    // Paging
    limit = SQLUtil.getLimit(request);

    OperationJson oj = new OperationJson();
    oj.setData(OperationDB.select(filter, order, limit));
    oj.setEcho(Integer.parseInt(request.getParameter("sEcho")));
    oj.setTotalRecords(OperationDB.count());
    oj.setTotalDisplayRecords(OperationDB.countFilter(filter));

    response.setContentType("application/json");
    response.setCharacterEncoding("UTF-8");
    response.getWriter().write(new Gson().toJson(oj));
  }

  public void doPost(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
    doGet(request, response);
  }
}

class OperationJson {
  private int sEcho;
  private int iTotalRecords;
  private int iTotalDisplayRecords;
  private String[][] aaData;

  public OperationJson() {
    sEcho = 0;
    iTotalRecords = 0;
    iTotalDisplayRecords = 0;
    aaData = null;
  }

  public void setEcho(int sEcho) {
    this.sEcho = sEcho;
  }

  public int getEcho() {
    return sEcho;
  }

  public void setTotalRecords(int iTotalRecords) {
    this.iTotalRecords = iTotalRecords;
  }

  public int getTotalRecords() {
    return iTotalRecords;
  }

  public void setTotalDisplayRecords(int iTotalDisplayRecords) {
    this.iTotalDisplayRecords = iTotalDisplayRecords;
  }

  public int getTotalDisplayRecords() {
    return iTotalDisplayRecords;
  }

  public void setData(ArrayList<Operation> operation_list) {
    if (operation_list != null) {
      aaData = new String[operation_list.size()][3];

      for (int i = 0; i < operation_list.size(); i++) {
        aaData[i][0] = operation_list.get(i).getOperationTimestampLong();
        aaData[i][1] = operation_list.get(i).getDescription();
        aaData[i][2] = operation_list.get(i).getOperationTimestamp();
      }
    }
    else {
      aaData = null;
    }
  }

  public String[][] getData() {
    return aaData;
  }
}
```

DisplayPendingMaterialList.java

```
package admin;

import java.io.IOException;
import java.util.ArrayList;

import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import com.google.gson.Gson;

import data.MaterialDB;
import data.SQLUtil;
import domain.Material;

public class DisplayPendingMaterialList extends HttpServlet {
  public void doGet(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
    String[] cols = {"pid", "dc_title", "CONCAT(first_name, ' ',
last_name)"};
    String filter = "";
    String order = "";
    String limit = "";

    // Filtering
    filter = SQLUtil.getFilter(request, cols);

    // Ordering
    order = SQLUtil.getOrder(request, cols);

    // Paging
    limit = SQLUtil.getLimit(request);

    PendingJson pj = new PendingJson();
    pj.setData(MaterialDB.selectPending(filter, order, limit));
    pj.setEcho(Integer.parseInt(request.getParameter("sEcho")));
    pj.setTotalRecords(MaterialDB.countPending());
    pj.setTotalDisplayRecords(MaterialDB.countPendingFilter(filter));

    response.setContentType("application/json");
```

```
    response.setCharacterEncoding("UTF-8");
    response.getWriter().write(new Gson().toJson(pj));
  }

  public void doPost(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
    doGet(request, response);
  }
}

class PendingJson {
  private int sEcho;
  private int iTotalRecords;
  private int iTotalDisplayRecords;
  private String[][] aaData;

  public PendingJson() {
    sEcho = 0;
    iTotalRecords = 0;
    iTotalDisplayRecords = 0;
    aaData = null;
  }

  public void setEcho(int sEcho) {
    this.sEcho = sEcho;
  }

  public int getEcho() {
    return sEcho;
  }

  public void setTotalRecords(int iTotalRecords) {
    this.iTotalRecords = iTotalRecords;
  }

  public int getTotalRecords() {
    return iTotalRecords;
  }

  public void setTotalDisplayRecords(int iTotalDisplayRecords) {
    this.iTotalDisplayRecords = iTotalDisplayRecords;
  }

  public int getTotalDisplayRecords() {
    return iTotalDisplayRecords;
  }

  public void setData(ArrayList<Material> material_list) {
    if (material_list != null) {
      aaData = new String[material_list.size()][3];

      for (int i = 0; i < material_list.size(); i++) {
        aaData[i][0] = material_list.get(i).getID();
        aaData[i][1] = material_list.get(i).getMetadata().getTitle();
        aaData[i][2] = material_list.get(i).getUser().getFirstName() + " " +
material_list.get(i).getUser().getLastName();
      }
    }
    else {
      aaData = null;
    }
  }

  public String[][] getData() {
    return aaData;
  }
}
```

DisplayUser.java

```
package admin;

import java.io.IOException;

import javax.servlet.RequestDispatcher;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import data.UserDB;
import domain.User;

public class DisplayUser extends HttpServlet {
  public void doGet(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
    if (request.getParameter("id") == null) {
      response.sendRedirect(request.getContextPath() + "/admin/u");

      return;
    }
    else {
      if
(UserDB.userExists(Long.parseLong(request.getParameter("id").trim())
)) {
        User user =
UserDB.select(Long.parseLong(request.getParameter("id").trim()));
        request.setAttribute("u", user);
      }
      else {
        response.sendRedirect(request.getContextPath() + "/admin/u");

        return;
      }

      String url = "/WEB-INF/admin/u/view.jsp";
      RequestDispatcher dispatcher =
getServletContext().getRequestDispatcher(url);
      dispatcher.forward(request, response);
    }
  }

  public void doPost(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
    doGet(request, response);
  }
}
```

DisplayUserList.java

```
package admin;

import java.io.IOException;
```

```
import java.util.LinkedHashMap;
import java.util.Map;

import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import com.google.gson.Gson;

import data.SQLUtil;
import data.UserDB;
import domain.User;

public class DisplayUserList extends HttpServlet {
  public void doGet(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
    String[] cols = {"user_id", "first_name", "last_name", "email_address",
"system_role"};
    String filter = "";
    String order = "";
    String limit = "";

    // Filtering
    filter = SQLUtil.getFilter(request, cols);

    // Ordering
    order = SQLUtil.getOrder(request, cols);

    // Paging
    limit = SQLUtil.getLimit(request);

    UserJson uj = new UserJson();
    uj.setData(UserDB.select(filter, order, limit));
    uj.setEcho(Integer.parseInt(request.getParameter("sEcho")));
    uj.setTotalRecords(UserDB.count());
    uj.setTotalDisplayRecords(UserDB.countFilter(filter));

    response.setContentType("application/json");
    response.setCharacterEncoding("UTF-8");
    response.getWriter().write(new Gson().toJson(uj));
  }

  public void doPost(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
    doGet(request, response);
  }
}

class UserJson {
  private int sEcho;
  private int iTotalRecords;
  private int iTotalDisplayRecords;
  private String[][] aaData;

  public UserJson() {
    sEcho = 0;
    iTotalRecords = 0;
    iTotalDisplayRecords = 0;
    aaData = null;
  }

  public void setEcho(int sEcho) {
    this.sEcho = sEcho;
  }

  public int getEcho() {
    return sEcho;
  }

  public void setTotalRecords(int iTotalRecords) {
    this.iTotalRecords = iTotalRecords;
  }

  public int getTotalRecords() {
    return iTotalRecords;
  }

  public void setTotalDisplayRecords(int iTotalDisplayRecords) {
    this.iTotalDisplayRecords = iTotalDisplayRecords;
  }

  public int getTotalDisplayRecords() {
    return iTotalDisplayRecords;
  }

  public void setData(LinkedHashMap<Long, User> user_list) {
    if (user_list != null) {
      aaData = new String[user_list.size()][5];

      int i = 0;

      for (Map.Entry<Long, User> u : user_list.entrySet()) {
        aaData[i][0] = u.getKey().toString();
        aaData[i][1] = u.getValue().getFirstName();
        aaData[i][2] = u.getValue().getLastName();
        aaData[i][3] = u.getValue().getEmailAddress();
        aaData[i][4] = u.getValue().getSystemRoleNormal();

        i++;
      }
    }
    else {
      aaData = null;
    }
  }

  public String[][] getData() {
    return aaData;
  }
}
```

FileServlet.java

```
package admin;

import java.io.BufferedInputStream;
import java.io.BufferedOutputStream;
import java.io.Closeable;
import java.io.IOException;
import java.io.InputStream;

import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
```

```
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import com.yourmediashelf.fedora.client.FedoraClient;
import com.yourmediashelf.fedora.client.FedoraClientException;
import
com.yourmediashelf.fedora.client.request.GetDatastreamDissemination
;
import com.yourmediashelf.fedora.client.response.FedoraResponse;

import data.FedoraConnect;
import data.MaterialDB;
import domain.Material;

public class FileServlet extends HttpServlet {
  private static final int DEFAULT_BUFFER_SIZE = 10240;

  public void doGet(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
    String pid = null;

    if (request.getParameter("id") == null) {
      response.sendRedirect(request.getContextPath());

      return;
    }
    else {
      pid = "get:" + request.getParameter("id");
    }

    if (MaterialDB.materialExists(pid)) {
      Material material = MaterialDB.select(pid);

      FedoraConnect connect = new FedoraConnect();
      FedoraClient client = connect.getClient();
      FedoraResponse fedora_response = null;

      // Verify the material first before performing the ingest operation.
      InputStream data = null;

      try {
        fedora_response = new
GetDatastreamDissemination(material.getPID(),
"DATA").execute(client);
        data = fedora_response.getEntityInputStream();
      }
      catch (FedoraClientException e) {
        e.printStackTrace();
      }

      response.reset();
      response.setBufferSize(DEFAULT_BUFFER_SIZE);
      response.setContentType(material.getFileType());
      response.setHeader("Content-Length",
Long.toString(material.getFileSize()));
      response.setHeader("Content-Disposition", "attachment;
filename=\"" + material.getFilename() + "\"");

      BufferedInputStream input = null;
      BufferedOutputStream output = null;

      try {
        // Open streams.
        input = new BufferedInputStream(data,
DEFAULT_BUFFER_SIZE);
        output = new
BufferedOutputStream(response.getOutputStream(),
DEFAULT_BUFFER_SIZE);

        // Write file contents to response.
        byte[] buffer = new byte[DEFAULT_BUFFER_SIZE];
        int length;

        while ((length = input.read(buffer)) > 0) {
          output.write(buffer, 0, length);
        }
      }
      finally {
        close(output);
        close(input);
      }
    }
    else {
      String action_redirect = "";

      if (MaterialDB.materialExistsAndApproved(pid)) {
        action_redirect = "/file?id=" + pid.substring(4);
      }

      response.sendRedirect(request.getContextPath() +
action_redirect);

      return;
    }
  }

  public void doPost(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
    doGet(request, response);
  }

    private static void close(Closeable resource) {
      if (resource != null) {
        try {
          resource.close();
        }
        catch (IOException e) {
          e.printStackTrace();
        }
      }
    }
}
```

ResolveReport.java

```
package admin;

import java.io.IOException;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;
```

```
import data.MessageDB;
import data.OperationDB;
import domain.Message;
import domain.User;

public class ResolveReport extends HttpServlet {
  public void doGet(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
    String check = request.getParameter("check");
    long message_id =
Long.parseLong(request.getParameter("message_id"));

    if (check.equals("1")) {
      MessageDB.resolve("flag_resolved", message_id);
      Message message = MessageDB.select(message_id);

      HttpSession session = request.getSession();
      User user = (User)session.getAttribute("me");

      String op = "<a href='user/inbox/read?id=" + message.getID() +
"'>Report</a> tagged as resolved by user <a href='admin/u/view?id=" +
user.getID() + "'>" + user.getFirstName() + " " + user.getLastName() +
"</a>.";
      OperationDB.insert(op);
    }
    else if (check.equals("0")) {
      MessageDB.resolve("flag", message_id);
      Message message = MessageDB.select(message_id);

      HttpSession session = request.getSession();
      User user = (User)session.getAttribute("me");

      String op = "<a href='user/inbox/read?id=" + message.getID() +
"'>Report</a> tagged as unresolved by user <a
href='admin/u/view?id=" + user.getID() + "'>" + user.getFirstName() + "
" + user.getLastName() + "</a>.";
      OperationDB.insert(op);
    }
  }

  public void doPost(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
    doGet(request, response);
  }
}
```

ResolveSupport.java

```
package admin;

import java.io.IOException;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;

import data.MessageDB;
import data.OperationDB;
import domain.Message;
import domain.User;

public class ResolveSupport extends HttpServlet {
  public void doGet(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
    String check = request.getParameter("check");
    long message_id =
Long.parseLong(request.getParameter("message_id"));

    if (check.equals("1")) {
      MessageDB.resolve("support_resolved", message_id);
      Message message = MessageDB.select(message_id);

      HttpSession session = request.getSession();
      User user = (User)session.getAttribute("me");

      String op = "<a href='admin/u/view?id=" + user.getID() + "'>" +
user.getFirstName() + " " + user.getLastName() + "</a> responded to
<a href='user/inbox/read?id=" + message.getID() + "'>message</a>
created using the 'Contact Us' form.";
      OperationDB.insert(op);
    }
    else if (check.equals("0")) {
      MessageDB.resolve("support", message_id);
      Message message = MessageDB.select(message_id);

      HttpSession session = request.getSession();
      User user = (User)session.getAttribute("me");

      String op = "<a href='admin/u/view?id=" + user.getID() + "'>" +
user.getFirstName() + " " + user.getLastName() + "</a> widthrew
his/her response to <a href='user/inbox/read?id=" + message.getID() +
"'>message</a> created using the 'Contact Us' form. Message is now
waiting for a response.";
      OperationDB.insert(op);
    }
  }

  public void doPost(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
    doGet(request, response);
  }
}
```

SubmitAnnotation.java

```
package admin;

import java.io.IOException;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;

import data.AnnotationDB;
import data.MaterialDB;
import data.OperationDB;
import data.UserDB;
import domain.Annotation;
import domain.User;

public class SubmitAnnotation extends HttpServlet {
```

```
  public void doPost(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
    validateAnnotation(request);

    HttpSession session = request.getSession();
    User user = (User)session.getAttribute("me");

    Annotation annotation = new Annotation();

    annotation.setContent(request.getParameter("content").trim());

annotation.setStartLocation(Integer.parseInt(request.getParameter("sta
rt_location")));

annotation.setEndLocation(Integer.parseInt(request.getParameter("end
_location")));
    annotation.setUser(UserDB.select(user.getEmailAddress()));

annotation.setMaterial(MaterialDB.select(request.getParameter("materi
al_pid")));

    int i = 0;

    if (request.getParameter("set_personal") == null) {
      i = AnnotationDB.insert("y", annotation);
    }
    else {
      i = AnnotationDB.insert("p", annotation);
    }

    if (i > 0) {
      String op = "<a href='admin/u/view?id=" + user.getID() + "'>" +
user.getFirstName() + " " + user.getLastName() + "</a> created a new
annotation for material <a href='admin/m/view?id=" +
annotation.getMaterial().getID() + "'>" +
annotation.getMaterial().getMetadata().getTitle() + "</a>.";
      OperationDB.insert(op);

      session.setAttribute("submit_annotation", "true");
    }
    else {
      session.setAttribute("submit_annotation", "false");
    }

    response.sendRedirect(request.getContextPath() +
"/admin/m/annotate?id=" + annotation.getMaterial().getID());
  }

  public void doGet(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
    response.sendRedirect(request.getContextPath() + "/admin/m");
  }

  private void validateAnnotation(HttpServletRequest request) throws
ServletException {
    // Content
    if (request.getParameter("content") == null ||
request.getParameter("content").trim().length() < 1) {
      throw new ServletException("Validation: Description (field) is
required.");
    }

    if (request.getParameter("content") == null ||
request.getParameter("content").trim().length() > 5000) {
      throw new ServletException("Validation: Description (field) exceeds
character limit of 5000.");
    }

    // Start Location
    if (request.getParameter("start_location") == null ||
Integer.parseInt(request.getParameter("start_location")) < 0) {
      throw new ServletException("Validation: Invalid start location
(field).");
    }

    // End Location
    if (request.getParameter("end_location") == null ||
Integer.parseInt(request.getParameter("end_location")) < 0) {
      throw new ServletException("Validation: Invalid end location
(field).");
    }

    // Material ID
    if (request.getParameter("material_pid") == null) {
      throw new ServletException("Validation: Invalid material (field).");
    }
  }
}
```

SubmitCommunity.java

```
package admin;

import java.io.IOException;

import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;
import javax.servlet.ServletException;

import data.CommunityDB;
import data.OperationDB;
import domain.Community;
import domain.User;

public class SubmitCommunity extends HttpServlet {
  public void doPost(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
    // Validate input. (server-side)
    validateCommunity(request);

    Community community = new Community();

    community.setCode(request.getParameter("code").trim());
    community.setName(request.getParameter("name").trim());

community.setDescription(request.getParameter("description").trim());

    int i = 0;
    String action = "";
    String action_redirect = "";

    if (request.getParameter("submit").equals("register")) {
```

```
      i = CommunityDB.insert(community);
      action_redirect = "/admin/c";

      action = "created";
    }
    else if (request.getParameter("submit").equals("update")) {
      i =
CommunityDB.update(CommunityDB.selectID(request.getParameter("u
pdate_code")), community);

      action_redirect = "/admin/c/view/" + community.getCode();

      action = "updated";
    }
    else {
      throw new ServletException("Can't process registration. Try again
later.");
    }

    HttpSession session = request.getSession();

    if (i > 0) {
      User user = (User)session.getAttribute("me");

      String op = "<a href='admin/u/view?id=" + user.getID() + "'>" +
user.getFirstName() + " " + user.getLastName() + "</a> " + action + "
community <a href='community/" + community.getCode() + "'>" +
community.getName() + "</a>.";
      OperationDB.insert(op);

      session.setAttribute("processed_community", community);
      session.setAttribute("submit_community", "true");
    }
    else {
      session.setAttribute("submit_community", "false");
    }

    response.sendRedirect(request.getContextPath() + action_redirect);
  }

  public void doGet(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
    response.sendRedirect(request.getContextPath() + "/admin/c");
  }

  private void validateCommunity(HttpServletRequest request) throws
ServletException {
    // Code
    if (request.getParameter("code") == null ||
request.getParameter("code").trim().length() < 1 ||
request.getParameter("code").trim().length() > 10) {
      throw new ServletException("Validation: Invalid community code.");
    }

    // Name
    if (request.getParameter("name") == null ||
request.getParameter("name").trim().length() < 1 ||
request.getParameter("name").trim().length() > 200) {
      throw new ServletException("Validation: Invalid community name.");
    }

    // Description
    if(request.getParameter("name").trim().length() > 5000) {
      throw new ServletException("Validation: Description (field) exceeds
character limit of 5000.");
    }
  }
}
```

SubmitMaterial.java

```
package admin;

import java.awt.image.BufferedImage;
import java.io.File;
import java.io.FileInputStream;
import java.io.IOException;
import java.io.InputStream;
import java.text.DateFormat;
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.Date;
import java.util.StringTokenizer;

import javax.imageio.ImageIO;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;

import org.apache.commons.fileupload.FileItem;
import org.apache.commons.fileupload.FileUploadException;
import org.apache.commons.io.FilenameUtils;
import org.apache.commons.io.IOUtils;
import org.apache.pdfbox.pdmodel.PDDocument;
import org.apache.pdfbox.pdmodel.PDPage;
import org.apache.pdfbox.util.PDFTextStripper;
import org.imgscalr.Scalr;

import data.CommunityDB;
import data.FavoriteDB;
import data.MaterialDB;
import data.MaterialFedora;
import data.OperationDB;
import data.TagDB;
import domain.Material;
import domain.Metadata;
import domain.User;

public class SubmitMaterial extends HttpServlet {
  public void doPost(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
    validateMaterial(request);

    File file = null;
    File file_thumbnail = null;
    File upload_path = new File("/home/aarceo/thumbnail");

    upload_path.setReadable(true);
    upload_path.setWritable(true);
    upload_path.setExecutable(true);

    Metadata metadata = new Metadata();
```

```
   Material material = new Material();
   DateFormat df = new SimpleDateFormat("yyyy-MM-dd");

   if (request.getParameter("submit").equals("deposit")) {
     String filename = "";
     String file_type = "";
     long file_size = 0;

     Object object = request.getAttribute("file");
     FileItem item = (FileItem)object;

     String base = FilenameUtils.getName(item.getName());
        String prefix = FilenameUtils.getBaseName(base);
        String suffix = "." + FilenameUtils.getExtension(base);

        // If prefix is < 3 letters in length, append date.
        if (prefix.length() < 3 || (prefix.length() + suffix.length() - 1 < 5)) {
          prefix = prefix + df.format(new Date());
        }

        try {
          file = File.createTempFile(prefix, suffix);

            item.write(file);

            base = prefix + suffix;
            filename = base.replaceAll("[ ]+", "_");
            if (item.getContentType() == null) {
              file_type = "application/octet-stream";
            }
            else {
              file_type = item.getContentType();
            }

            file_size = item.getSize();
        }
        catch (Exception e) {
          e.printStackTrace();
        }

        String index = "";

        if (file_type.substring(0, 5).equals("text/")) {
            FileInputStream input = new FileInputStream(file);

            try {
                index = IOUtils.toString(input, "UTF-8");
            }
            finally {
                input.close();
            }
        }
        else if (file_type.startsWith("image")) {
          Object picture_object = request.getAttribute("picture");

          if (picture_object == null) {
            try {
              file_thumbnail = File.createTempFile(prefix, suffix,
upload_path);

              file_thumbnail.setReadable(true);
              file_thumbnail.setWritable(true);
              file_thumbnail.setExecutable(true);
            }
            catch (Exception e){
              e.printStackTrace();
            }
          }
        }
        else if (file_type.equals("application/pdf")) {
            PDDocument pd = null;

            try {
              pd = PDDocument.load(file);
              PDFTextStripper stripper = new PDFTextStripper();

              index = stripper.getText(pd);
            }
            catch (Exception e) {
              e.printStackTrace();

              throw new ServletException(".pdf file is encrypted. Unable to
index file.");
            }
            finally {
               if (pd != null) {
                 pd.close();
               }
            }
        }

        material.setFilename(filename);
      material.setFileType(file_type);
      material.setFileSize(file_size);
      material.setIndex(index);
    }


metadata.setContributor(request.getParameter("dc_contributor").trim());
    metadata.setCoverage(request.getParameter("dc_coverage").trim());
    metadata.setCreator(request.getParameter("dc_creator").trim());

metadata.setDescription(request.getParameter("dc_description").trim())
;
    metadata.setFormat(request.getParameter("dc_format").trim());
    metadata.setIdentifier(request.getParameter("dc_identifier").trim());
    metadata.setLanguage(request.getParameter("dc_language").trim());
    metadata.setPublisher(request.getParameter("dc_publisher").trim());
    metadata.setRelation(request.getParameter("dc_relation").trim());
    metadata.setRights(request.getParameter("dc_rights").trim());
    metadata.setSource(request.getParameter("dc_source").trim());
    metadata.setSubject(request.getParameter("dc_subject").trim());
    metadata.setTitle(request.getParameter("dc_title").trim());
    metadata.setType(request.getParameter("dc_type").trim());


material.setCommunity(CommunityDB.select(request.getParameter("co
mmunity")));

    // Insert tags to database
    String tags = request.getParameter("tags");
    StringTokenizer st = new StringTokenizer(tags, ",");

    while (st.hasMoreTokens()) {
```

```
      String val = st.nextToken().trim();

      // Insert tag if it does not exist
      if (!TagDB.tagExists(val)) {
        TagDB.insert(val);
      }
    }

    material.setTags(tags);

    material.setPrecededBy(request.getParameter("preceded_by"));

    try {
      // Date
      if (!request.getParameter("material_month").equals("-1")) {
        metadata.setDate(df.parse(request.getParameter("material_year")
+ "-" + request.getParameter("material_month") + "-" +
request.getParameter("material_day")));
      }
      else {
        metadata.setDate(null);
      }

      // Embargo
      if (!request.getParameter("embargo_month").equals("-1")) {

material.setEmbargo(df.parse(request.getParameter("embargo_year")
+ "-" + request.getParameter("embargo_month") + "-" +
request.getParameter("embargo_day")));
      }
      else {
        material.setEmbargo(null);
      }
    }
    catch (ParseException e) {
      e.printStackTrace();
    }

    material.setMetadata(metadata);

    HttpSession session = request.getSession();
    User user = (User)session.getAttribute("me");

    if (!user.getSystemRole().equals("upmir_system_administrator")) {
      throw new ServletException("Access denied.");
    }

    String action = request.getParameter("submit");
    String action_redirect = "";
    String action_new = "";

    if (action.equals("deposit")) {
      try {
        material.setUser(user);

        // Ingest material to Fedora.
        material.setPID(MaterialFedora.insert(material, file));

        // Set material's history.
        if (request.getParameter("history") == null) {
          material.setHistory(material.getPID());
        }
        else {
          String origin_pid = "get:" + request.getParameter("history");
          material.setHistory(origin_pid);
        }

        // Insert material reference to database.
        MaterialDB.insert("y", material);

        action_redirect = "/admin/m";

        action_new = "uploaded";
      }
      catch (Exception e) {
        depositCheck(material);
      }
    }
    else if (action.equals("update")) {
      material.setPID(request.getParameter("material_id"));

      // Update
      MaterialFedora.update(material);

      MaterialDB.update(material);

      action_redirect = "/admin/m/view?id=" + material.getID();

      action_new = "updated";
    }
    else {
      throw new ServletException("Can't process registration. Try again
later.");
    }

    // Thumbnail
    Object picture_object = request.getAttribute("picture");

    if (picture_object != null) {
      FileItem picture_item = (FileItem)picture_object;

      String base = FilenameUtils.getName(picture_item.getName());
        String prefix = material.getID();
        String suffix = "." + FilenameUtils.getExtension(base);

        InputStream instream = null;

        try {
          file = File.createTempFile(prefix, suffix, upload_path);

        file.setReadable(true);
        file.setWritable(true);
        file.setExecutable(true);

          instream = picture_item.getInputStream();
          BufferedImage image = ImageIO.read(instream);
          BufferedImage thumbnail = null;

          if (image.getWidth() > 512 && image.getHeight() > 512) {
            thumbnail = Scalr.resize(image, 512);
          }
          else {
            thumbnail = image;
          }
```

```
        String file_type = picture_item.getContentType();

        if (file_type.equals("image/gif")) {
          ImageIO.write(thumbnail, "gif", file);
        }
      else if (file_type.equals("image/jpeg")) {
        ImageIO.write(thumbnail, "jpg", file);
      }
      else if (file_type.equals("image/png")) {
        ImageIO.write(thumbnail, "png", file);
      }

        // Insert thumbnail file name.
        MaterialDB.insertThumbnail(material, file.getName());
       }
       catch (Exception e) {
        e.printStackTrace();
       }
     finally {
      instream.close();
     }
    }
    else {
     picture_object = request.getAttribute("file");
     FileItem picture_item = (FileItem)picture_object;

     // Image
     if (material.getFileType().startsWith("image")) {
      InputStream instream = null;

      try {
       instream = picture_item.getInputStream();
       BufferedImage image = ImageIO.read(instream);
       BufferedImage thumbnail = null;

       if (image.getWidth() > 128 && image.getHeight() > 128) {
        thumbnail = Scalr.resize(image, 128);
       }
       else {
        thumbnail = image;
       }

       if (material.getFileType().equals("image/gif")) {
        ImageIO.write(thumbnail, "gif", file_thumbnail);
       }
       else if (material.getFileType().equals("image/jpeg")) {
        ImageIO.write(thumbnail, "jpg", file_thumbnail);
       }
       else if (material.getFileType().equals("image/png")) {
        ImageIO.write(thumbnail, "png", file_thumbnail);
       }

          // Insert thumbnail file name.
          MaterialDB.insertThumbnail(material,
file_thumbnail.getName());
       }
       catch (Exception e) {
        e.printStackTrace();
       }
       finally {
        instream.close();
       }
      }
      // PDF
      else if (material.getFileType().equals("application/pdf")) {
       // Create thumbnail
            PDDocument pd = null;

            try {
             file = File.createTempFile("upmir" + material.getID(), ".jpg",
upload_path);

            file.setReadable(true);
            file.setWritable(true);
            file.setExecutable(true);

             picture_item.write(file);

             pd = PDDocument.load(file);
             PDPage first_page =
(PDPage)pd.getDocumentCatalog().getAllPages().get(0);
            BufferedImage image = first_page.convertToImage();
          BufferedImage thumbnail = null;

          if (image.getWidth() > 1024 && image.getHeight() > 1024) {
           thumbnail = Scalr.resize(image, 1024);
          }
          else {
           thumbnail = image;
          }

          ImageIO.write(thumbnail, "jpg", file);

             // Insert thumbnail file name.
             MaterialDB.insertThumbnail(material, file.getName());
             }
             catch (Exception e) {
              e.printStackTrace();
             }
             finally {
                if (pd != null) {
                 pd.close();
                }
             }
      }
     }

     if (depositCheck(material)) {
      String op = "<a href='admin/u/view?id=" + user.getID() + "'>" +
user.getFirstName() + " " + user.getLastName() + "</a> " + action_new
+ " material <a href='admin/m/view?id=" + material.getID() + "'>" +
material.getMetadata().getTitle() + "</a>.";
      OperationDB.insert(op);

      session.setAttribute("submit_material", "true");
     }
     else {
      session.setAttribute("submit_material", "false");
     }

     response.sendRedirect(request.getContextPath() + action_redirect);
    }
```

```
  public void doGet(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
    response.sendRedirect(request.getContextPath() + "/admin/m");
  }

  private void validateMaterial(HttpServletRequest request) throws
ServletException {
    if (request.getParameter("submit").equals("deposit")) {
      // File
      Object object = request.getAttribute("file");

      if (object == null || object instanceof FileUploadException) {
        throw new ServletException("Validation: Invalid file.");
      }
      else {
        FileItem item = (FileItem)object;

        String base = FilenameUtils.getName(item.getName());
            String prefix = FilenameUtils.getBaseName(base);
            String suffix = "." + FilenameUtils.getExtension(base);

            if (prefix.length() + suffix.length() + 1 > 200) {
              throw new ServletException("Validation: File name is too
long.");
            }
      }
    }

    // Thumbnail
    Object picture_object = request.getAttribute("picture");

    if (picture_object instanceof FileUploadException) {
      throw new ServletException("Validation: Invalid thumbnail picture.");
    }
    else {
      if (picture_object != null) {
        FileItem picture_item = (FileItem)picture_object;
        String content_type = picture_item.getContentType();

        if (!content_type.equals("image/gif") &&
!content_type.equals("image/jpeg") &&
!content_type.equals("image/png")) {
          throw new ServletException("Validation: Invalid thumbnail picture
content type. Please make sure the file selected for use as a thumbnail
is either gif, jpeg, or png.");
        }
      }
    }

    // Contributor
    if (request.getParameter("dc_contributor") == null) {
      throw new ServletException("Validation: Contributor (field) is null.");
    }
    if (request.getParameter("dc_contributor").trim().length() > 200) {
      throw new ServletException("Validation: Please enter no more than
200 characters for contributor (field).");
    }

    // Coverage
    if (request.getParameter("dc_coverage") == null) {
      throw new ServletException("Validation: Coverage (field) is null.");
    }
    if (request.getParameter("dc_coverage").trim().length() > 200) {
      throw new ServletException("Validation: Please enter no more than
200 characters for coverage (field).");
    }

    // Creator
    if (request.getParameter("dc_creator") == null) {
      throw new ServletException("Validation: Creator (field) is null.");
    }
    if (request.getParameter("dc_creator").trim().length() > 200) {
      throw new ServletException("Validation: Please enter no more than
200 characters for creator (field).");
    }

    DateFormat df = new SimpleDateFormat("yyyy-MM-dd");
    df.setLenient(false);
    Date current_date = new Date();
    Date test_date;

    // Date
    if (request.getParameter("material_month") == null ||
request.getParameter("material_day") == null ||
request.getParameter("material_year") == null) {
      throw new ServletException("Validation: Date (field) is null.");
    }

    if (!request.getParameter("material_month").equals("-1") ||
!request.getParameter("material_day").equals("-1") ||
!request.getParameter("material_year").equals("-1")) {
      try {
        test_date = df.parse(request.getParameter("material_year") + "-" +
request.getParameter("material_month") + "-" +
request.getParameter("material_day"));

        if (test_date.after(current_date)) {
          throw new ServletException("Validation: Invalid date (field).");
        }
      }
      catch (ParseException e) {
        throw new ServletException("Validation: Invalid date (field).");
      }
    }

    // Description
    if (request.getParameter("dc_description") == null) {
      throw new ServletException("Validation: Description (field) is null.");
    }
    if (request.getParameter("dc_description").trim().length() < 1) {
      throw new ServletException("Validation: Description (field) is
blank.");
    }
    if (request.getParameter("dc_description").trim().length() > 5000) {
      throw new ServletException("Validation: Please enter no more than
5000 characters for description (field).");
    }

    // Format
    if (request.getParameter("dc_format") == null) {
      throw new ServletException("Validation: Format (field) is null.");
    }
    if (request.getParameter("dc_format").trim().length() > 200) {
```

```
    throw new ServletException("Validation: Please enter no more than
200 characters for format (field).");
   }

   // Identifier
   if (request.getParameter("dc_identifier") == null) {
    throw new ServletException("Validation: Identifier (field) is null.");
   }
   if (request.getParameter("dc_identifier").trim().length() > 200) {
    throw new ServletException("Validation: Please enter no more than
200 characters for identifier (field).");
   }

   // Language
   if (request.getParameter("dc_language") == null) {
    throw new ServletException("Validation: Language (field) is null.");
   }
   if (request.getParameter("dc_language").trim().length() > 200) {
    throw new ServletException("Validation: Please enter no more than
200 characters for language (field).");
   }

   // Publisher
   if (request.getParameter("dc_publisher") == null) {
    throw new ServletException("Validation: Publisher (field) is null.");
   }
   if (request.getParameter("dc_publisher").trim().length() > 200) {
    throw new ServletException("Validation: Please enter no more than
200 characters for publisher (field).");
   }

   // Relation
   if (request.getParameter("dc_relation") == null) {
    throw new ServletException("Validation: Relation (field) is null.");
   }
   if (request.getParameter("dc_relation").trim().length() > 200) {
    throw new ServletException("Validation: Please enter no more than
200 characters for relation (field).");
   }

   // Rights
   if (request.getParameter("dc_rights") == null) {
    throw new ServletException("Validation: Rights (field) is null.");
   }
   if (request.getParameter("dc_rights").trim().length() > 200) {
    throw new ServletException("Validation: Please enter no more than
200 characters for rights (field).");
   }

   // Source
   if (request.getParameter("dc_source") == null) {
    throw new ServletException("Validation: Source (field) is null.");
   }
   if (request.getParameter("dc_source").trim().length() > 200) {
    throw new ServletException("Validation: Please enter no more than
200 characters for source (field).");
   }

   // Subject
   if (request.getParameter("dc_subject") == null) {
    throw new ServletException("Validation: Subject (field) is null.");
   }
   if (request.getParameter("dc_subject").trim().length() > 200) {
    throw new ServletException("Validation: Please enter no more than
200 characters for subject (field).");
   }

   // Title
   if (request.getParameter("dc_title") == null) {
    throw new ServletException("Validation: Title (field) is null.");
   }
   if (request.getParameter("dc_title").trim().length() < 1) {
    throw new ServletException("Validation: Title (field) is blank.");
   }
   if (request.getParameter("dc_title").trim().length() > 200) {
    throw new ServletException("Validation: Please enter no more than
200 characters for title (field).");
   }

   // Type
   if (request.getParameter("dc_type") == null) {
    throw new ServletException("Validation: Type (field) is null.");
   }
   if (request.getParameter("dc_type").trim().length() > 200) {
    throw new ServletException("Validation: Please enter no more than
200 characters for type (field).");
   }

   // Community
   if (request.getParameter("community") == null) {
    throw new ServletException("Validation: Community (field) is null.");
   }
   if (request.getParameter("community").equals("")) {
    throw new ServletException("Validation: No community selected.");
   }

   // Tags
   if (request.getParameter("tags") == null) {
    throw new ServletException("Validation: Tags (field) is null.");
   }
   if (request.getParameter("tags").trim().length() > 5000) {
    throw new ServletException("Validation: Please enter no more than
5000 characters for tags (field).");
   }

   // Embargo
   if (request.getParameter("embargo_month") == null ||
request.getParameter("embargo_day") == null ||
request.getParameter("embargo_year") == null) {
    throw new ServletException("Validation: Embargo (field) is null.");
   }

   if (!request.getParameter("embargo_month").equals("-1") ||
!request.getParameter("embargo_day").equals("-1") ||
!request.getParameter("embargo_year").equals("-1")) {
     try {
      test_date = df.parse(request.getParameter("embargo_year") + "-"
+ request.getParameter("embargo_month") + "-" +
request.getParameter("embargo_day"));

      if (test_date.before(current_date)) {
       throw new ServletException("Validation: Invalid date for embargo
(field).");
      }
```

```
    }
    catch (ParseException e) {
      throw new ServletException("Validation: Invalid date for embargo
(field).");
    }
   }
  }

  private boolean depositCheck(Material material) {
    if (MaterialDB.materialExists(material.getPID()) &&
MaterialFedora.materialExists(material.getPID())) {
     return true;
    }
    else {
     MaterialDB.delete(material.getPID());
     FavoriteDB.delete(material.getPID());

     if (MaterialFedora.materialExists(material.getPID())) {
       MaterialFedora.delete(material.getPID());
     }

     return false;
    }
  }
}
```

SubmitNews.java

```
package admin;

import java.io.IOException;

import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;

import org.springframework.util.StringUtils;

import data.NewsDB;
import data.OperationDB;
import domain.News;
import domain.User;

public class SubmitNews extends HttpServlet {
  public void doPost(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
    // Validate input. (server-side)
    validateNews(request);

    String title = request.getParameter("title").trim();
    String reference = "";

    reference = title;
    reference = reference.toLowerCase().replaceAll("[ ]+", "-");
    reference = reference.replaceAll("[^A-Za-z0-9-]", "");

    while (StringUtils.countOccurrencesOf(reference, "-") > 4) {
      reference = reference.substring(0, reference.lastIndexOf("-"));
    }

    if (NewsDB.referenceExists(reference)) {
      int counter = 0;

      while (NewsDB.referenceExists(reference + "-" + counter)) {
        counter++;
      }

      // If it exists, append number/s at the end.
      reference = reference + "-" + counter;
    }

    HttpSession session = request.getSession();

    News news = new News();

    news.setTitle(title);
    news.setReference(reference);
    news.setContent(request.getParameter("news_content").trim());
    news.setUser((User)session.getAttribute("me"));

    int i = 0;
    String action_redirect = "";
    String action = "";

    if (request.getParameter("submit").equals("post")) {
      i = NewsDB.insert(news);
      action_redirect = "/admin/news";

      action = "posted";
    }
    else if (request.getParameter("submit").equals("update")) {
      reference = request.getParameter("reference");

      if (reference != null &&
NewsDB.delete(request.getParameter("reference")) > 0) {
        news.setReference(reference);
        i = NewsDB.insert(news);
      }

      action_redirect = "/admin/news/edit/" + news.getReference();

      action = "updated";
    }
    else {
      throw new ServletException("Can't process news submission. Try
again later.");
    }

    if (i > 0) {
      User user = (User)session.getAttribute("me");

      String op = "<a href='admin/u/view?id=" + user.getID() + "'>" +
user.getFirstName() + " " + user.getLastName() + "</a> " + action + "
news <a href='news?id=" + NewsDB.selectID(news.getReference()) +
"'>" + news.getTitle() + "</a>.";
      OperationDB.insert(op);

      session.setAttribute("submit_news", "true");
    }
    else {
      session.setAttribute("submit_news", "false");
    }
```

```
    response.sendRedirect(request.getContextPath() + action_redirect);
  }

  public void doGet(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
    response.sendRedirect(request.getContextPath() + "/admin/news");
  }

  private void validateNews(HttpServletRequest request) throws
ServletException {
    // Title
    if (request.getParameter("title") == null ||
request.getParameter("title").trim().length() < 1 ||
request.getParameter("title").trim().length() > 200) {
      throw new ServletException("Validation: Invalid news title.");
    }

    // Content
    if (request.getParameter("news_content") == null ||
request.getParameter("news_content").trim().length() < 1 ||
request.getParameter("news_content").trim().length() > 65000) {
      throw new ServletException("Validation: Invalid news content.");
    }
  }
}
```

SubmitUser.java

```
package admin;

import java.io.IOException;
import java.text.DateFormat;
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.Calendar;
import java.util.Date;
import java.util.TimeZone;

import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;
import javax.servlet.ServletException;

import org.apache.commons.codec.digest.DigestUtils;
import org.apache.commons.validator.routines.EmailValidator;

import data.CommunityDB;
import data.OperationDB;
import data.UserDB;
import domain.User;

public class SubmitUser extends HttpServlet {
  public void doPost(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
    // Validate input. (server-side)
    validateUser(request);

    User user = new User();

user.setEmailAddress(request.getParameter("email_address").trim());

user.setPassword(DigestUtils.md5Hex(request.getParameter("passwor
d")));
    user.setFirstName(request.getParameter("first_name").trim());
    user.setLastName(request.getParameter("last_name").trim());
    user.setDescription(request.getParameter("description").trim());
    user.setOccupation(request.getParameter("occupation").trim());
    user.setOrganization(request.getParameter("organization").trim());

user.setContactNumber(request.getParameter("contact_number").trim(
));
    user.setSystemRole(request.getParameter("system_role").trim());

    if (user.getSystemRole().equals("upmir_community_moderator")) {

user.setCommunity(CommunityDB.select(request.getParameter("comm
unity").trim()));
    }
    else if (user.getSystemRole().equals("upmir_university_user")) {

user.setStudentNumber(request.getParameter("student_number").trim(
));
    }

    DateFormat df = new SimpleDateFormat("yyyy-MM-dd");
    df.setTimeZone(TimeZone.getTimeZone("Asia/Manila"));
    df.setLenient(false);

    Date today = new Date();
    Date birthday = new Date();
    Date expiration = new Date();

    try {
      // Birthday
      birthday = df.parse(request.getParameter("birthday_year") + "-" +
request.getParameter("birthday_month") + "-" +
request.getParameter("birthday_day"));

      // Expiration
      if (!(request.getParameter("expiration_year").equals("-1") &&
request.getParameter("expiration_month").equals("-1") &&
request.getParameter("expiration_day").equals("-1"))) {
        expiration = df.parse(request.getParameter("expiration_year") + "-
" + request.getParameter("expiration_month") + "-" +
request.getParameter("expiration_day"));
      }
      else {
        expiration =
df.parse(Integer.toString(Calendar.getInstance().get(Calendar.YEAR) +
15) + "-01-01");
      }
    }
    catch (ParseException e) {
      e.printStackTrace();
    }

    user.setBirthday(birthday);
    user.setExpiration(expiration);
    user.setLastActivity(today);
```

```
    int i = 0;
    String action = request.getParameter("submit");
    String action_redirect = "";
    String action_new = "";

    if (action.equals("register")) {
      i = UserDB.insert(user);
      action_redirect = "/admin/u";
      action_new = "created an account for user";
    }
    else if (action.equals("update")) {
      if
(UserDB.userExists(Long.parseLong(request.getParameter("user_id")))
) {
        i =
UserDB.update(Long.parseLong(request.getParameter("user_id")),
user);
        action_redirect = "/admin/u/view?id=" +
request.getParameter("user_id");

        action_new = "updated account details of user";
      }
      else {
        action_redirect = "/admin/u";
      }
    }
    else {
      throw new ServletException("Can't process registration. Try again
later.");
    }

    HttpSession session = request.getSession();

    if (i > 0) {
      User me = (User)session.getAttribute("me");

      String op = "<a href='admin/u/view?id=" + me.getID() + "'>" +
me.getFirstName() + " " + me.getLastName() + "</a> " + action_new +
" <a href='admin/u/view?id=" + user.getID() + "'>" +
user.getFirstName() + " " + user.getLastName() + "</a>.";
      OperationDB.insert(op);

      session.setAttribute("processed_user", user);
      session.setAttribute("submit_user", "true");
    }
    else {
      session.setAttribute("submit_user", "false");
    }

    response.sendRedirect(request.getContextPath() + action_redirect);
  }

  public void doGet(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
    response.sendRedirect(request.getContextPath() + "/admin/u");
  }

  private void validateUser(HttpServletRequest request) throws
ServletException {
    // Email Address
    if
(!EmailValidator.getInstance().isValid(request.getParameter("email_add
ress").trim())) {
      throw new ServletException("Validation: Invalid email address.");
    }

    // Password
    if (request.getParameter("password").length() == 0) {
      if (!request.getParameter("submit").equals("update")) {
        throw new ServletException("Validation: Invalid password.");
      }
    }
    else if (request.getParameter("password").length() < 5) {
      throw new ServletException("Validation: Invalid password.");
    }

    // First Name
    if (request.getParameter("first_name").trim().length() < 1) {
      throw new ServletException("Validation: First name (field) is
blank.");
    }

    // Last Name
    if (request.getParameter("last_name").trim().length() < 1) {
      throw new ServletException("Validation: Last name (field) is
blank.");
    }

    DateFormat df = new SimpleDateFormat("yyyy-MM-dd");
    df.setLenient(false);
    Date current_date = new Date();
    Date test_date;

    // Birthday
    try {
      test_date = df.parse(request.getParameter("birthday_year") + "-" +
request.getParameter("birthday_month") + "-" +
request.getParameter("birthday_day"));

      if (test_date.after(current_date)) {
        throw new ServletException("Validation: Invalid date for birthday
(field).");
      }
    }
    catch (ParseException e) {
      throw new ServletException("Validation: Invalid date for birthday
(field).");
    }

    // Description
    if (request.getParameter("description").trim().length() > 500) {
      throw new ServletException("Validation: Description (field) exceeds
character limit of 500.");
    }

    // Occupation
    if (request.getParameter("occupation").trim().length() > 100) {
      throw new ServletException("Validation: Occupation (field) exceeds
character limit of 100.");
    }

    // Organization
```

```
    if (request.getParameter("organization").trim().length() > 100) {
      throw new ServletException("Validation: Organization (field)
exceeds character limit of 100.");
    }

    // Contact Number
    if (request.getParameter("contact_number").trim().length() > 25) {
      throw new ServletException("Validation: Contact number (field)
exceeds character limit of 25.");
    }

    // System Role
    if
(!request.getParameter("system_role").equals("upmir_system_administ
rator") &&

!request.getParameter("system_role").equals("upmir_community_mode
rator") &&

!request.getParameter("system_role").equals("upmir_university_user")
&&

!request.getParameter("system_role").equals("upmir_registered_user"))
{

      throw new ServletException("Validation: No system role provided.");
    }

    // Community
    if (request.getParameter("community").trim().length() > 10) {
      throw new ServletException("Validation: Invalid community code.");
    }

    if
(request.getParameter("system_role").equals("upmir_community_mode
rator")) {
      if (request.getParameter("community").trim().length() < 1) {
        throw new ServletException("Validation: Missing student/faculty
number.");
      }
      if
(!CommunityDB.communityExists(request.getParameter("community").t
rim())) {
        throw new ServletException("Validation: No such community.");
      }
    }

    // Student Number
    if (request.getParameter("student_number").trim().length() > 10) {
      throw new ServletException("Validation: Invalid student/faculty
number.");
    }

    if
(request.getParameter("system_role").equals("upmir_university_user"))
{
      if (request.getParameter("student_number").trim().length() < 1) {
        throw new ServletException("Validation: Missing student/faculty
number.");
      }
      if
(!request.getParameter("student_number").trim().matches("^\\d\\d\\d\\d-
\\d\\d\\d\\d\\d$")) {
        throw new ServletException("Validation: Invalid student/faculty
number.");
      }
    }

    // Expiration
    if (request.getParameter("expiration_year").equals("-1") &&
request.getParameter("expiration_month").equals("-1") &&
request.getParameter("expiration_day").equals("-1")) {
      return;
    }
    else {
      try {
        test_date = df.parse(request.getParameter("expiration_year") + "-"
+ request.getParameter("expiration_month") + "-" +
request.getParameter("expiration_day"));

        if (test_date.before(current_date)) {
          throw new ServletException("Validation: Invalid date for
expiration (field).");
        }
      }
      catch (ParseException e) {
        throw new ServletException("Validation: Invalid date for expiration
(field).");
      }
    }
  }
}
```

UpdateAnnotation.java

```
package admin;

import java.io.IOException;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;

import data.AnnotationDB;
import data.OperationDB;
import domain.Annotation;
import domain.User;

public class UpdateAnnotation extends HttpServlet {
  public void doPost(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
    validateAnnotation(request);

    HttpSession session = request.getSession();

    String is_approved = "n";

    long annotation_id =
Long.parseLong(request.getParameter("annotation_id"));
    Annotation annotation = AnnotationDB.select(annotation_id);
    String content = request.getParameter("update_description");
```

```
    if (request.getParameter("approve") != null ||
annotation.getApproved().equals("y")) {
      is_approved = "y";
    }
    else if (annotation.getApproved().equals("p")) {
      is_approved = "p";
    }

    int i = AnnotationDB.update(is_approved, annotation_id, content);

    if (i > 0) {
      User user = (User)session.getAttribute("me");

      String op = "<a href='admin/u/view?id=" + user.getID() + "'>" +
user.getFirstName() + " " + user.getLastName() + "</a> updated
annotation for material <a href='admin/m/view?id=" +
annotation.getMaterial().getID() + "'>" +
annotation.getMaterial().getMetadata().getTitle() + "</a>.";
      OperationDB.insert(op);

      session.setAttribute("submit_annotation", "true");
    }
    else {
      session.setAttribute("submit_annotation", "false");
    }

    response.sendRedirect(request.getContextPath() +
"/admin/m/annotate?id=" + annotation.getMaterial().getID());
  }

  public void doGet(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
    response.sendRedirect(request.getContextPath());
  }

  private void validateAnnotation(HttpServletRequest request) throws
ServletException {
    // Content
    if (request.getParameter("update_description") == null ||
request.getParameter("update_description").trim().length() < 1) {
      throw new ServletException("Validation: Description (field) is
required.");
    }
    if (request.getParameter("update_description") == null ||
request.getParameter("update_description").trim().length() > 5000) {
      throw new ServletException("Validation: Description (field) exceeds
character limit of 5000.");
    }

    // Annotation ID
    if (request.getParameter("annotation_id") == null ||
request.getParameter("annotation_id").equals("-1")) {
      throw new ServletException("Validation: Invalid annotation.");
    }
  }
}
```

DeletePersonalAnnotation.java

```
package annotation;

import java.io.IOException;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;

import data.AnnotationDB;
import domain.Annotation;
import domain.User;

public class DeletePersonalAnnotation extends HttpServlet {
  public void doPost(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
    HttpSession session = request.getSession();
    String material_id = "";
    User user = (User)session.getAttribute("me");

    if (request.getParameter("annotation_id") == null) {
      session.setAttribute("delete_annotation", "false");
    }
    else {
      Annotation annotation =
AnnotationDB.select(Long.parseLong(request.getParameter("annotatio
n_id")));

      int i = 0;

      if (user.getID() == annotation.getUser().getID()) {
        i =
AnnotationDB.delete(Long.parseLong(request.getParameter("annotatio
n_id")));
      }

      if (i > 0) {
        session.setAttribute("delete_annotation", "true");
      }
      else {
        session.setAttribute("delete_annotation", "false");
      }

      material_id = request.getParameter("material_id");
    }

    response.sendRedirect(request.getContextPath() + "/annotation?id="
+ material_id);
  }
}
```

DisplayAnnotation.java

```
package annotation;

import java.io.IOException;
import java.io.InputStream;
import java.util.Date;
import java.util.LinkedHashMap;

import javax.servlet.RequestDispatcher;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
```

```
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;

import org.apache.commons.io.IOUtils;
import org.apache.commons.io.input.BOMInputStream;
import org.joda.time.DateTime;
import org.joda.time.Days;

import com.yourmediashelf.fedora.client.FedoraClient;
import com.yourmediashelf.fedora.client.FedoraClientException;
import
com.yourmediashelf.fedora.client.request.GetDatastreamDissemination
;
import com.yourmediashelf.fedora.client.response.FedoraResponse;

import data.AnnotationDB;
import data.FedoraConnect;
import data.MaterialDB;
import domain.Annotation;
import domain.Material;
import domain.User;

public class DisplayAnnotation extends HttpServlet {
  public void doGet(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
    if (request.getParameter("id") == null) {
      response.sendRedirect(request.getContextPath());

      return;
    }
    else if (MaterialDB.materialExists("get:" +
request.getParameter("id"))) {
      Material material = MaterialDB.select("get:" +
request.getParameter("id"));

      DateTime date = new DateTime(material.getEmbargo());
      DateTime current = new DateTime(new Date());

      Days d = Days.daysBetween(current, date);

      if (d.getDays() > 0) {
        response.sendRedirect(request.getContextPath() + "/file?id=" +
material.getID());

        return;
      }

      if (material.getFileType().equals("text/plain")) {
        FedoraConnect connect = new FedoraConnect();
        FedoraClient client = connect.getClient();
        FedoraResponse fedora_response = null;

        InputStream data = null;

        try {
          fedora_response = new
GetDatastreamDissemination(material.getPID(),
"DATA").execute(client);
          data = fedora_response.getEntityInputStream();
        }
        catch (FedoraClientException e) {
          e.printStackTrace();
        }

        BOMInputStream input = new BOMInputStream(data);
        String contents = IOUtils.toString(input, "UTF-8");

        request.setAttribute("m", material);
        request.setAttribute("contents", contents);

        HttpSession session = request.getSession();
        User user = (User)session.getAttribute("me");

        // Get annotations.
        LinkedHashMap<Long, Annotation> annotation_list =
AnnotationDB.selectViewableMaterialAnnotations(material.getPID(),
user.getID());

        if (annotation_list != null && annotation_list.size() > 0) {
          request.setAttribute("al", annotation_list);
        }
      }
      else {
        response.sendRedirect(request.getContextPath() + "/file?id=" +
material.getID());

        return;
      }
    }
    else {
      response.sendRedirect(request.getContextPath());

      return;
    }

    String action_redirect = "view_edit.jsp";

    String url = "/WEB-INF/material/annotation/" + action_redirect;
    RequestDispatcher dispatcher =
getServletContext().getRequestDispatcher(url);
    dispatcher.forward(request, response);
  }

  public void doPost(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
    doGet(request, response);
  }
}
```

SubmitPersonalAnnotation.java

```
package annotation;

import java.io.IOException;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;

import data.AnnotationDB;
import data.MaterialDB;
import domain.Annotation;
import domain.User;
```

```
public class SubmitPersonalAnnotation extends HttpServlet {
  public void doPost(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
    validateAnnotation(request);

    HttpSession session = request.getSession();
    User user = (User)session.getAttribute("me");

    Annotation annotation = new Annotation();

    annotation.setContent(request.getParameter("content").trim());

annotation.setStartLocation(Integer.parseInt(request.getParameter("sta
rt_location")));

annotation.setEndLocation(Integer.parseInt(request.getParameter("end
_location")));
    annotation.setUser(user);

annotation.setMaterial(MaterialDB.select(request.getParameter("materi
al_pid")));

    int i = AnnotationDB.insert("p", annotation);

    if (i > 0) {
      session.setAttribute("submit_annotation", "true");
    }
    else {
      session.setAttribute("submit_annotation", "false");
    }

    response.sendRedirect(request.getContextPath() + "/annotation?id="
+ annotation.getMaterial().getID());
  }

  public void doGet(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
    response.sendRedirect(request.getContextPath());
  }

  private void validateAnnotation(HttpServletRequest request) throws
ServletException {
    // Content
    if (request.getParameter("content") == null ||
request.getParameter("content").trim().length() < 1) {
      throw new ServletException("Validation: Description (field) is
required.");
    }
    if (request.getParameter("content") == null ||
request.getParameter("content").trim().length() > 5000) {
      throw new ServletException("Validation: Description (field) exceeds
character limit of 5000.");
    }

    // Start Location
    if (request.getParameter("start_location") == null ||
Integer.parseInt(request.getParameter("start_location")) < 0) {
      throw new ServletException("Validation: Invalid start location
(field).");
    }

    // End Location
    if (request.getParameter("end_location") == null ||
Integer.parseInt(request.getParameter("end_location")) < 0) {
      throw new ServletException("Validation: Invalid end location
(field).");
    }

    // Material ID
    if (request.getParameter("material_pid") == null) {
      throw new ServletException("Validation: Invalid material (field).");
    }
  }
}
```

UpdatePersonalAnnotation.java

```
package annotation;

import java.io.IOException;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;

import data.AnnotationDB;
import domain.Annotation;
import domain.User;

public class UpdatePersonalAnnotation extends HttpServlet {
  public void doPost(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
    validateAnnotation(request);

    HttpSession session = request.getSession();
    User user = (User)session.getAttribute("me");

    long annotation_id =
Long.parseLong(request.getParameter("annotation_id"));
    Annotation annotation = AnnotationDB.select(annotation_id);
    String content = request.getParameter("update_description");

    int i = AnnotationDB.updatePersonal(annotation_id, user, content);

    if (i > 0) {
      session.setAttribute("submit_annotation", "true");
    }
    else {
      session.setAttribute("submit_annotation", "false");
    }

    response.sendRedirect(request.getContextPath() + "/annotation?id="
+ annotation.getMaterial().getID());
  }

  public void doGet(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
    response.sendRedirect(request.getContextPath());
  }

  private void validateAnnotation(HttpServletRequest request) throws
ServletException {
```

```
    // Content
    if (request.getParameter("update_description") == null ||
request.getParameter("update_description").trim().length() < 1) {
      throw new ServletException("Validation: Description (field) is
required.");
    }
    if (request.getParameter("update_description") == null ||
request.getParameter("update_description").trim().length() > 5000) {
      throw new ServletException("Validation: Description (field) exceeds
character limit of 5000.");
    }

    // Annotation ID
    if (request.getParameter("annotation_id") == null ||
request.getParameter("annotation_id").equals("-1")) {
      throw new ServletException("Validation: Invalid annotation.");
    }
  }
}
```

DisplayMaterialHistory.java

```
package catalog;

import java.io.IOException;
import java.util.ArrayList;

import javax.servlet.RequestDispatcher;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import data.MaterialDB;
import domain.Material;

public class DisplayMaterialHistory extends HttpServlet {
  public void doGet(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
    String pid = null;

    if (request.getParameter("id") == null) {
      response.sendRedirect(request.getContextPath());

      return;
    }
    else {
      pid = "get:" + request.getParameter("id");
    }

    if (MaterialDB.materialExistsAndApproved(pid)) {
      Material material = MaterialDB.select(pid);
      request.setAttribute("m", material);

      // Get a list of materials related to the requested material.
      ArrayList<Material> material_list =
MaterialDB.selectCollection(material.getHistory());
      request.setAttribute("ml", material_list);
    }

    String url = "/WEB-INF/material/history/view.jsp";
    RequestDispatcher dispatcher =
getServletContext().getRequestDispatcher(url);
    dispatcher.forward(request, response);
  }

  public void doPost(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
    doGet(request, response);
  }
}
```

DisplayTagList.java

```
package catalog;

import java.io.IOException;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import com.google.gson.Gson;

import data.TagDB;

public class DisplayTagList extends HttpServlet {
  public void doGet(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
    String query = request.getParameter("query");

    TagJson tj = new TagJson();
    tj.setQuery(query);
    tj.setSuggestions(TagDB.select(query));

    response.setContentType("application/json");
    response.setCharacterEncoding("UTF-8");
    response.getWriter().write(new Gson().toJson(tj));
  }

  public void doPost(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
    doGet(request, response);
  }
}

class TagJson {
  private String query;
  private String[] suggestions;

  public TagJson() {
    query = "";
    suggestions = null;
  }

  public void setQuery(String query) {
    this.query = query;
  }

  public String getQuery() {
    return query;
  }
```

```
  public void setSuggestions(String[] suggestions) {
    this.suggestions = suggestions;
  }

  public String[] getSuggestions() {
    return suggestions;
  }
}
```

ImageServlet.java

```
package catalog;

import java.io.BufferedInputStream;
import java.io.BufferedOutputStream;
import java.io.Closeable;
import java.io.File;
import java.io.FileInputStream;
import java.io.IOException;
import java.net.URLDecoder;

import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

public class ImageServlet extends HttpServlet {
  private static final int DEFAULT_BUFFER_SIZE = 10240;

  public void doGet(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
    String requested_image = request.getPathInfo();

    if (requested_image == null) {
      return;
    }

    File image = new File("/home/aarceo/thumbnail",
URLDecoder.decode(requested_image, "UTF-8"));

    if (!image.exists()) {
      return;
    }

    String content_type =
getServletContext().getMimeType(image.getName());

    if (content_type == null || !content_type.startsWith("image")) {
      return;
    }

        response.reset();
        response.setBufferSize(DEFAULT_BUFFER_SIZE);
        response.setContentType(content_type);
        response.setHeader("Content-Length",
String.valueOf(image.length()));
        response.setHeader("Content-Disposition", "inline; filename=\"" +
image.getName() + "\"");

        BufferedInputStream input = null;
        BufferedOutputStream output = null;

        try {
          input = new BufferedInputStream(new FileInputStream(image),
DEFAULT_BUFFER_SIZE);
          output = new
BufferedOutputStream(response.getOutputStream(),
DEFAULT_BUFFER_SIZE);

          byte[] buffer = new byte[DEFAULT_BUFFER_SIZE];
          int length;

          while ((length = input.read(buffer)) > 0) {
            output.write(buffer, 0, length);
          }
        }
        finally {
          close(output);
          close(input);
        }

  }

  public void doPost(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
    doGet(request, response);
  }

    private static void close(Closeable resource) {
      if (resource != null) {
        try {
          resource.close();
        }
        catch (IOException e) {
          e.printStackTrace();
        }
      }
    }
}
```

AnnotationDB.java

```
package data;

import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.LinkedHashMap;

import domain.Annotation;
import domain.User;

public class AnnotationDB {
  public static Annotation select(long annotation_id) {
    ConnectionPool pool = ConnectionPool.getInstance();
    Connection connection = pool.getConnection();
    PreparedStatement ps = null;
    ResultSet rs = null;

    String query = "SELECT * FROM annotation WHERE annotation_id
= ?";

    try {
```

```
      ps = connection.prepareStatement(query);
      ps.setLong(1, annotation_id);
      rs = ps.executeQuery();

      if (rs.next()) {
        Annotation annotation = new Annotation();

        annotation.setApproved(rs.getString("is_approved"));
        annotation.setContent(rs.getString("content"));
        annotation.setStartLocation(rs.getInt("start_location"));
        annotation.setEndLocation(rs.getInt("end_location"));
        annotation.setUser(UserDB.select(rs.getLong("user_id")));

annotation.setMaterial(MaterialDB.select(rs.getString("material_id")));

annotation.setAnnotationTimestamp(rs.getTimestamp("annotation_time
stamp"));

        return annotation;
      }
      else {
        return null;
      }
    }
    catch (SQLException e) {
      e.printStackTrace();

      return null;
    }
    finally {
      DBUtil.closeResultSet(rs);
      DBUtil.closePreparedStatement(ps);
      pool.freeConnection(connection);
    }
  }

  public static LinkedHashMap<Long, Annotation>
selectMaterialAnnotations(String material_id) {
    ConnectionPool pool = ConnectionPool.getInstance();
    Connection connection = pool.getConnection();
    PreparedStatement ps = null;
    ResultSet rs = null;

    String query = "SELECT * FROM annotation WHERE material_id = ?
AND (is_approved = 'y' OR is_approved = 'n') ORDER BY
annotation_timestamp DESC";

    try {
      ps = connection.prepareStatement(query);
      ps.setString(1, material_id);
      rs = ps.executeQuery();

      LinkedHashMap<Long, Annotation> annotation_list = new
LinkedHashMap<Long, Annotation>();

      while (rs.next()) {
        Annotation annotation = new Annotation();

        annotation.setApproved(rs.getString("is_approved"));
        annotation.setContent(rs.getString("content"));
        annotation.setStartLocation(rs.getInt("start_location"));
        annotation.setEndLocation(rs.getInt("end_location"));
        annotation.setUser(UserDB.select(rs.getLong("user_id")));

annotation.setMaterial(MaterialDB.select(rs.getString("material_id")));

annotation.setAnnotationTimestamp(rs.getTimestamp("annotation_time
stamp"));

        annotation_list.put(rs.getLong("annotation_id"), annotation);
      }

      return annotation_list;
    }
    catch (SQLException e) {
      e.printStackTrace();

      return null;
    }
    finally {
      DBUtil.closeResultSet(rs);
      DBUtil.closePreparedStatement(ps);
      pool.freeConnection(connection);
    }
  }

  public static LinkedHashMap<Long, Annotation>
selectMaterialAnnotations(String material_id, User user) {
    ConnectionPool pool = ConnectionPool.getInstance();
    Connection connection = pool.getConnection();
    PreparedStatement ps = null;
    ResultSet rs = null;

    String query = "SELECT * FROM annotation WHERE material_id = ?
AND (is_approved = 'y' OR is_approved = 'n') OR (is_approved = 'p'
AND user_id = ?) ORDER BY annotation_timestamp DESC";

    try {
      ps = connection.prepareStatement(query);
      ps.setString(1, material_id);
      ps.setLong(2, user.getID());
      rs = ps.executeQuery();

      LinkedHashMap<Long, Annotation> annotation_list = new
LinkedHashMap<Long, Annotation>();

      while (rs.next()) {
        Annotation annotation = new Annotation();

        annotation.setApproved(rs.getString("is_approved"));
        annotation.setContent(rs.getString("content"));
        annotation.setStartLocation(rs.getInt("start_location"));
        annotation.setEndLocation(rs.getInt("end_location"));
        annotation.setUser(UserDB.select(rs.getLong("user_id")));

annotation.setMaterial(MaterialDB.select(rs.getString("material_id")));

annotation.setAnnotationTimestamp(rs.getTimestamp("annotation_time
stamp"));

        annotation_list.put(rs.getLong("annotation_id"), annotation);
      }

      return annotation_list;
```

```
    }
    catch (SQLException e) {
      e.printStackTrace();

      return null;
    }
    finally {
      DBUtil.closeResultSet(rs);
      DBUtil.closePreparedStatement(ps);
      pool.freeConnection(connection);
    }
  }

  public static LinkedHashMap<Long, Annotation>
selectApprovedMaterialAnnotations(String material_id) {
    ConnectionPool pool = ConnectionPool.getInstance();
    Connection connection = pool.getConnection();
    PreparedStatement ps = null;
    ResultSet rs = null;

    String query = "SELECT * FROM annotation WHERE material_id = ?
AND is_approved = 'y' ORDER BY annotation_timestamp DESC";

    try {
      ps = connection.prepareStatement(query);
      ps.setString(1, material_id);
      rs = ps.executeQuery();

      LinkedHashMap<Long, Annotation> annotation_list = new
LinkedHashMap<Long, Annotation>();

      while (rs.next()) {
        Annotation annotation = new Annotation();

        annotation.setApproved(rs.getString("is_approved"));
        annotation.setContent(rs.getString("content"));
        annotation.setStartLocation(rs.getInt("start_location"));
        annotation.setEndLocation(rs.getInt("end_location"));
        annotation.setUser(UserDB.select(rs.getLong("user_id")));

annotation.setMaterial(MaterialDB.select(rs.getString("material_id")));

annotation.setAnnotationTimestamp(rs.getTimestamp("annotation_time
stamp"));

        annotation_list.put(rs.getLong("annotation_id"), annotation);
      }

      return annotation_list;
    }
    catch (SQLException e) {
      e.printStackTrace();

      return null;
    }
    finally {
      DBUtil.closeResultSet(rs);
      DBUtil.closePreparedStatement(ps);
      pool.freeConnection(connection);
    }
  }

  // Select the material's approved annotations plus your pending
annotations
  public static LinkedHashMap<Long, Annotation>
selectViewableMaterialAnnotations(String material_id, long user_id) {
    ConnectionPool pool = ConnectionPool.getInstance();
    Connection connection = pool.getConnection();
    PreparedStatement ps = null;
    ResultSet rs = null;

    String query = "SELECT * FROM annotation WHERE material_id = ?
AND (is_approved = 'y' OR user_id = ?) OR (is_approved = 'p' AND
user_id = ?) ORDER BY annotation_timestamp DESC";

    try {
      ps = connection.prepareStatement(query);

      ps.setString(1, material_id);
      ps.setLong(2, user_id);
      ps.setLong(3, user_id);

      rs = ps.executeQuery();

      LinkedHashMap<Long, Annotation> annotation_list = new
LinkedHashMap<Long, Annotation>();

      while (rs.next()) {
        Annotation annotation = new Annotation();

        annotation.setApproved(rs.getString("is_approved"));
        annotation.setContent(rs.getString("content"));
        annotation.setStartLocation(rs.getInt("start_location"));
        annotation.setEndLocation(rs.getInt("end_location"));
        annotation.setUser(UserDB.select(rs.getLong("user_id")));

annotation.setMaterial(MaterialDB.select(rs.getString("material_id")));

annotation.setAnnotationTimestamp(rs.getTimestamp("annotation_time
stamp"));

        annotation_list.put(rs.getLong("annotation_id"), annotation);
      }

      return annotation_list;
    }
    catch (SQLException e) {
      e.printStackTrace();

      return null;
    }
    finally {
      DBUtil.closeResultSet(rs);
      DBUtil.closePreparedStatement(ps);
      pool.freeConnection(connection);
    }
  }

  public static int insert(String is_approved, Annotation annotation) {
    ConnectionPool pool = ConnectionPool.getInstance();
      Connection connection = pool.getConnection();
      PreparedStatement ps = null;
      ResultSet rs = null;
```

```
        String query = "INSERT INTO annotation " +
                "(content, is_approved, start_location, end_location," +
                " user_id, material_id, annotation_timestamp)"        +
                " VALUES (?, ?, ?, ?, ?, ?, NOW())";

        try {
            ps = connection.prepareStatement(query);

            ps.setString(1, annotation.getContent());
            ps.setString(2, is_approved);
            ps.setInt(3, annotation.getStartLocation());
            ps.setInt(4, annotation.getEndLocation());
            ps.setLong(5,
UserDB.selectID(annotation.getUser().getEmailAddress()));
            ps.setString(6, annotation.getMaterial().getPID());

            return ps.executeUpdate();
        }
        catch (SQLException e) {
            e.printStackTrace();

            return 0;
        }
        finally {
            DBUtil.closeResultSet(rs);
            DBUtil.closePreparedStatement(ps);
            pool.freeConnection(connection);
        }
    }

    public static int approve(long annotation_id) {
      ConnectionPool pool = ConnectionPool.getInstance();
        Connection connection = pool.getConnection();
        PreparedStatement ps = null;
        ResultSet rs = null;

        String query = "UPDATE annotation SET is_approved = 'y' WHERE
annotation_id = ?";

        try {
          ps = connection.prepareStatement(query);

          ps.setLong(1, annotation_id);

            return ps.executeUpdate();
        }
        catch (SQLException e) {
          e.printStackTrace();

            return 0;
        }
        finally {
          DBUtil.closeResultSet(rs);
            DBUtil.closePreparedStatement(ps);
            pool.freeConnection(connection);
        }
    }

    public static int update(String is_approved, long annotation_id, String
content) {
      ConnectionPool pool = ConnectionPool.getInstance();
        Connection connection = pool.getConnection();
        PreparedStatement ps = null;
        ResultSet rs = null;

        String query = "UPDATE annotation SET is_approved = ?, content
= ?, annotation_timestamp = NOW() WHERE annotation_id = ?";

        try {
          ps = connection.prepareStatement(query);

          ps.setString(1, is_approved);
          ps.setString(2, content);
          ps.setLong(3, annotation_id);

            return ps.executeUpdate();
        }
        catch (SQLException e) {
          e.printStackTrace();

            return 0;
        }
        finally {
          DBUtil.closeResultSet(rs);
            DBUtil.closePreparedStatement(ps);
            pool.freeConnection(connection);
        }
    }

    public static int updateRegular(long annotation_id, String content) {
      ConnectionPool pool = ConnectionPool.getInstance();
        Connection connection = pool.getConnection();
        PreparedStatement ps = null;
        ResultSet rs = null;

        String query = "UPDATE annotation SET is_approved = 'n', content
= ?, annotation_timestamp = NOW() WHERE annotation_id = ?";

        try {
          ps = connection.prepareStatement(query);

          ps.setString(1, content);
          ps.setLong(2, annotation_id);

            return ps.executeUpdate();
        }
        catch (SQLException e) {
          e.printStackTrace();

            return 0;
        }
        finally {
          DBUtil.closeResultSet(rs);
            DBUtil.closePreparedStatement(ps);
            pool.freeConnection(connection);
        }
    }

    public static int updatePersonal(long annotation_id, User user, String
content) {
      ConnectionPool pool = ConnectionPool.getInstance();
        Connection connection = pool.getConnection();
        PreparedStatement ps = null;
        ResultSet rs = null;
```

```
    String query = "UPDATE annotation SET content = ?,
annotation_timestamp = NOW() WHERE annotation_id = ? AND
is_approved = 'p' AND user_id = ?";

    try {
      ps = connection.prepareStatement(query);

      ps.setString(1, content);
      ps.setLong(2, annotation_id);
      ps.setLong(3, user.getID());

        return ps.executeUpdate();
    }
    catch (SQLException e) {
      e.printStackTrace();

        return 0;
    }
    finally {
      DBUtil.closeResultSet(rs);
        DBUtil.closePreparedStatement(ps);
        pool.freeConnection(connection);
    }
  }

  public static int delete(long annotation_id) {
    ConnectionPool pool = ConnectionPool.getInstance();
    Connection connection = pool.getConnection();
    PreparedStatement ps = null;
    ResultSet rs = null;

    String query = "DELETE FROM annotation WHERE annotation_id
= ?";

    try {
      ps = connection.prepareStatement(query);
        ps.setLong(1, annotation_id);

        return ps.executeUpdate();
    }
    catch (SQLException e) {
      e.printStackTrace();

        return 0;
    }
    finally {
      DBUtil.closeResultSet(rs);
        DBUtil.closePreparedStatement(ps);
        pool.freeConnection(connection);
    }
  }
}
```

CommentDB.java

```
package data;

import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.LinkedHashMap;

import domain.Comment;

public class CommentDB {
  public static boolean commentExists(long comment_id) {
    ConnectionPool pool = ConnectionPool.getInstance();
    Connection connection = pool.getConnection();
    PreparedStatement ps = null;
    ResultSet rs = null;

    String query = "SELECT comment_id FROM comment WHERE
comment_id = ?";

    try {
      ps = connection.prepareStatement(query);
      ps.setLong(1, comment_id);
      rs = ps.executeQuery();

      return rs.next();
    }
    catch (SQLException e) {
      e.printStackTrace();

      return false;
    }
    finally {
      DBUtil.closeResultSet(rs);
          DBUtil.closePreparedStatement(ps);
          pool.freeConnection(connection);
    }
  }

  public static LinkedHashMap<Long, Comment>
selectMaterialComments(String pid) {
    ConnectionPool pool = ConnectionPool.getInstance();
    Connection connection = pool.getConnection();
    PreparedStatement ps = null;
    ResultSet rs = null;

    String query = "SELECT * FROM comment WHERE material_id = ?
ORDER BY comment_timestamp DESC";

    try {
      ps = connection.prepareStatement(query);
      ps.setString(1, pid);
      rs = ps.executeQuery();

      LinkedHashMap<Long, Comment> comment_list = new
LinkedHashMap<Long, Comment>();

      while (rs.next()) {
        Comment comment = new Comment();

        comment.setContent(rs.getString("content"));
        comment.setUser(UserDB.select(rs.getLong("depositor")));

comment.setMaterial(MaterialDB.select(rs.getString("material_id")));

comment.setCommentTimestamp(rs.getTimestamp("comment_timesta
mp"));
```

```
        comment_list.put(rs.getLong("comment_id"), comment);
      }

      return comment_list;
    }
    catch (SQLException e) {
      e.printStackTrace();

      return null;
    }
    finally {
      DBUtil.closeResultSet(rs);
      DBUtil.closePreparedStatement(ps);
      pool.freeConnection(connection);
    }
  }

  public static int insert(Comment comment) {
    ConnectionPool pool = ConnectionPool.getInstance();
        Connection connection = pool.getConnection();
        PreparedStatement ps = null;
        ResultSet rs = null;

        String query = "INSERT INTO comment " +
                "(content, depositor, material_id, comment_timestamp)" +
                " VALUES (?, ?, ?, NOW())";

        try {
            ps = connection.prepareStatement(query);

            ps.setString(1, comment.getContent());
            ps.setLong(2,
UserDB.selectID(comment.getUser().getEmailAddress()));
            ps.setString(3, comment.getMaterial().getPID());

            return ps.executeUpdate();
        }
        catch (SQLException e) {
            e.printStackTrace();

            return 0;
        }
        finally {
            DBUtil.closeResultSet(rs);
            DBUtil.closePreparedStatement(ps);
            pool.freeConnection(connection);
        }
  }

  public static int delete(long comment_id) {
    ConnectionPool pool = ConnectionPool.getInstance();
      Connection connection = pool.getConnection();
      PreparedStatement ps = null;
      ResultSet rs = null;

      String query = "DELETE FROM comment WHERE comment_id =
?";

      try {
        ps = connection.prepareStatement(query);
          ps.setLong(1, comment_id);

          return ps.executeUpdate();
        }
        catch (SQLException e) {
          e.printStackTrace();

          return 0;
        }
        finally {
          DBUtil.closeResultSet(rs);
            DBUtil.closePreparedStatement(ps);
            pool.freeConnection(connection);
        }
    }
}
```

CommunityDB.java

```
package data;

import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.ArrayList;

import domain.Community;

public class CommunityDB {
  public static int count() {
    ConnectionPool pool = ConnectionPool.getInstance();
    Connection connection = pool.getConnection();
    PreparedStatement ps = null;
    ResultSet rs = null;

    String query = "SELECT COUNT(community_id) FROM community";

    try {
      ps = connection.prepareStatement(query);
      rs = ps.executeQuery();

      if (rs.next()) {
        return rs.getInt(1);
      }
      else {
        return 0;
      }
    }
    catch (SQLException e) {
      e.printStackTrace();

      return 0;
    }
    finally {
      DBUtil.closeResultSet(rs);
          DBUtil.closePreparedStatement(ps);
          pool.freeConnection(connection);
    }
  }

  public static int countFilter(String filter) {
    ConnectionPool pool = ConnectionPool.getInstance();
    Connection connection = pool.getConnection();
```

```
    PreparedStatement ps = null;
    ResultSet rs = null;

    String query = "SELECT COUNT(community_id) FROM community "
+ filter;

    try {
      ps = connection.prepareStatement(query);
      rs = ps.executeQuery();

      if (rs.next()) {
        return rs.getInt(1);
      }
      else {
        return 0;
      }
    }
    catch (SQLException e) {
      e.printStackTrace();

      return 0;
    }
    finally {
      DBUtil.closeResultSet(rs);
          DBUtil.closePreparedStatement(ps);
          pool.freeConnection(connection);
    }
  }

  public static boolean communityExists(String code) {
    ConnectionPool pool = ConnectionPool.getInstance();
    Connection connection = pool.getConnection();
    PreparedStatement ps = null;
    ResultSet rs = null;

    String query = "SELECT code FROM community WHERE code = ?";

    try {
      ps = connection.prepareStatement(query);
      ps.setString(1, code);
      rs = ps.executeQuery();

      return rs.next();
    }
    catch (SQLException e) {
      e.printStackTrace();

      return false;
    }
    finally {
      DBUtil.closeResultSet(rs);
          DBUtil.closePreparedStatement(ps);
          pool.freeConnection(connection);
    }
  }


  public static boolean communityExists(long community_id) {
    ConnectionPool pool = ConnectionPool.getInstance();
    Connection connection = pool.getConnection();
    PreparedStatement ps = null;
    ResultSet rs = null;

    String query = "SELECT code FROM community WHERE
community_id = ?";

    try {
      ps = connection.prepareStatement(query);
      ps.setLong(1, community_id);
      rs = ps.executeQuery();

      return rs.next();
    }
    catch (SQLException e) {
      e.printStackTrace();

      return false;
    }
    finally {
      DBUtil.closeResultSet(rs);
          DBUtil.closePreparedStatement(ps);
          pool.freeConnection(connection);
    }
  }

  public static long selectID(String code) {
    ConnectionPool pool = ConnectionPool.getInstance();
    Connection connection = pool.getConnection();
    PreparedStatement ps = null;
    ResultSet rs = null;

    String query = "SELECT community_id FROM community WHERE
code = ?";

    try {
      ps = connection.prepareStatement(query);
      ps.setString(1, code);
      rs = ps.executeQuery();

      if (rs.next()) {
        return rs.getLong("community_id");
      }
      else {
        return -1;
      }
    }
    catch (SQLException e) {
      e.printStackTrace();

      return -1;
    }
    finally {
      DBUtil.closeResultSet(rs);
      DBUtil.closePreparedStatement(ps);
      pool.freeConnection(connection);
    }
  }

  public static Community select(String code) {
    ConnectionPool pool = ConnectionPool.getInstance();
    Connection connection = pool.getConnection();
    PreparedStatement ps = null;
    ResultSet rs = null;
```

```
    String query = "SELECT * FROM community WHERE code = ?";

    try {
      ps = connection.prepareStatement(query);
      ps.setString(1, code);
      rs = ps.executeQuery();

      if (rs.next()) {
        Community community = new Community();

        community.setCode(rs.getString("code"));
        community.setName(rs.getString("name"));
        community.setDescription(rs.getString("description"));

community.setCreationTimestamp(rs.getTimestamp("creation_timesta
mp"));

        return community;
      }
      else {
        return null;
      }
    }
    catch (SQLException e) {
      e.printStackTrace();

      return null;
    }
    finally {
      DBUtil.closeResultSet(rs);
      DBUtil.closePreparedStatement(ps);
      pool.freeConnection(connection);
    }
  }

  public static Community select(long community_id) {
    ConnectionPool pool = ConnectionPool.getInstance();
    Connection connection = pool.getConnection();
    PreparedStatement ps = null;
    ResultSet rs = null;

    String query = "SELECT * FROM community WHERE community_id
= ?";

    try {
      ps = connection.prepareStatement(query);
      ps.setLong(1, community_id);
      rs = ps.executeQuery();

      if (rs.next()) {
        Community community = new Community();

        community.setCode(rs.getString("code"));
        community.setName(rs.getString("name"));
        community.setDescription(rs.getString("description"));

community.setCreationTimestamp(rs.getTimestamp("creation_timesta
mp"));

        return community;
      }
      else {
        return null;
      }
    }
    catch (SQLException e) {
      e.printStackTrace();

      return null;
    }
    finally {
      DBUtil.closeResultSet(rs);
      DBUtil.closePreparedStatement(ps);
      pool.freeConnection(connection);
    }
  }

  public static ArrayList<Community> select(String filter, String order,
String limit) {
    ConnectionPool pool = ConnectionPool.getInstance();
    Connection connection = pool.getConnection();
    PreparedStatement ps = null;
    ResultSet rs = null;

    String query = "SELECT * FROM community " + filter + " " + order + "
" + limit;

    try {
      ps = connection.prepareStatement(query);
      rs = ps.executeQuery();
      ArrayList<Community> community_list = new
ArrayList<Community>();

      while (rs.next()) {
        Community community = new Community();

        community.setCode(rs.getString("code"));
        community.setName(rs.getString("name"));
        community.setDescription(rs.getString("description"));

community.setCreationTimestamp(rs.getTimestamp("creation_timesta
mp"));

        community_list.add(community);
      }

      return community_list;
    }
    catch (SQLException e) {
      e.printStackTrace();

      return null;
    }
    finally {
      DBUtil.closeResultSet(rs);
      DBUtil.closePreparedStatement(ps);
      pool.freeConnection(connection);
    }
  }

  public static ArrayList<Community> selectAll() {
    ConnectionPool pool = ConnectionPool.getInstance();
    Connection connection = pool.getConnection();
    PreparedStatement ps = null;
```

```
    ResultSet rs = null;

    String query = "SELECT * FROM community ORDER BY name
ASC";

    try {
      ps = connection.prepareStatement(query);
      rs = ps.executeQuery();
      ArrayList<Community> community_list = new
ArrayList<Community>();

      while (rs.next()) {
        Community community = new Community();

        community.setCode(rs.getString("code"));
        community.setName(rs.getString("name"));
        community.setDescription(rs.getString("description"));

community.setCreationTimestamp(rs.getTimestamp("creation_timesta
mp"));

        community_list.add(community);
      }

      return community_list;
    }
    catch (SQLException e) {
      e.printStackTrace();

      return null;
    }
    finally {
      DBUtil.closeResultSet(rs);
      DBUtil.closePreparedStatement(ps);
      pool.freeConnection(connection);
    }
  }

  public static int insert(Community community) {
    ConnectionPool pool = ConnectionPool.getInstance();
      Connection connection = pool.getConnection();
      PreparedStatement ps = null;
      ResultSet rs = null;

      String query = "INSERT INTO community "       +
                "(code,        name,"          +
                " description, creation_timestamp)" +
                " VALUES (?, ?, ?, NOW())";

      try {
         ps = connection.prepareStatement(query);

         ps.setString(1, community.getCode());
         ps.setString(2, community.getName());
         ps.setString(3, community.getDescription());

         return ps.executeUpdate();
      }
      catch (SQLException e) {
         e.printStackTrace();

         return 0;
      }
      finally {
         DBUtil.closeResultSet(rs);
         DBUtil.closePreparedStatement(ps);
         pool.freeConnection(connection);
      }
  }

  public static int update(long community_id, Community community) {
    ConnectionPool pool = ConnectionPool.getInstance();
      Connection connection = pool.getConnection();
      PreparedStatement ps = null;
      ResultSet rs = null;

      String query = "UPDATE community SET " +
             "code = ?, "            +
                  "name = ?, "           +
                  "description = ? "     +
                  "WHERE community_id = ?";

      try {
        ps = connection.prepareStatement(query);

          ps.setString(1, community.getCode());
          ps.setString(2, community.getName());
          ps.setString(3, community.getDescription());
          ps.setLong(4, community_id);

         return ps.executeUpdate();
      }
      catch (SQLException e) {
        e.printStackTrace();

         return 0;
      }
      finally {
        DBUtil.closeResultSet(rs);
         DBUtil.closePreparedStatement(ps);
         pool.freeConnection(connection);
      }
  }

  public static int delete(String code) {
    ConnectionPool pool = ConnectionPool.getInstance();
      Connection connection = pool.getConnection();
      PreparedStatement ps = null;
      ResultSet rs = null;

      String query = "DELETE FROM community WHERE code = ?";

      try {
        ps = connection.prepareStatement(query);
         ps.setString(1, code);

         return ps.executeUpdate();
      }
      catch (SQLException e) {
        e.printStackTrace();

         return 0;
      }
      finally {
```

```
      DBUtil.closeResultSet(rs);
       DBUtil.closePreparedStatement(ps);
       pool.freeConnection(connection);
    }
  }

  public ArrayList<Community> getCommunities() {
    return selectAll();
  }
}
```

ConnectionPool.java

```
package data;

import java.sql.*;
import javax.sql.DataSource;
import javax.naming.InitialContext;

public class ConnectionPool {
  private static ConnectionPool pool = null;
  private static DataSource data_source = null;

  private ConnectionPool() {
    try {
      InitialContext ic = new InitialContext();
      data_source =
(DataSource)ic.lookup("java:/comp/env/jdbc/fedora");
    }
    catch(Exception e) {
      e.printStackTrace();
    }
  }

  public static ConnectionPool getInstance() {
    if(pool == null) {
      pool = new ConnectionPool();
    }
    return pool;
  }

  public Connection getConnection() {
    try {
      return data_source.getConnection();
    }
    catch(SQLException sqle) {
      sqle.printStackTrace();
      return null;
    }
  }

  public void freeConnection(Connection c) {
    try {
      c.close();
    }
    catch(SQLException sqle) {
      sqle.printStackTrace();
    }
  }
}
```

DBUtil.java

```
package data;

import java.sql.*;

public class DBUtil {
  public static void closeStatement(Statement s) {
    try {
      if(s != null) s.close();
    }
    catch(SQLException e) {
      e.printStackTrace();
    }
  }

  public static void closePreparedStatement(Statement ps) {
    try {
      if(ps != null) ps.close();
    }
    catch(SQLException e) {
      e.printStackTrace();
    }
  }

  public static void closeResultSet(ResultSet rs) {
    try {
      if(rs != null) rs.close();
    }
    catch(SQLException e) {
      e.printStackTrace();
    }
  }
}
```

FavoriteDB.java

```
package data;

import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.LinkedHashMap;

import domain.Favorite;
import domain.User;

public class FavoriteDB {
  public static boolean favoriteExists(Favorite favorite) {
    ConnectionPool pool = ConnectionPool.getInstance();
    Connection connection = pool.getConnection();
    PreparedStatement ps = null;
    ResultSet rs = null;

    String query = "SELECT favorite_id FROM favorite WHERE
material_id = ? AND user_id = ?";

    try {
      ps = connection.prepareStatement(query);

      ps.setString(1, favorite.getMaterial().getPID());
```

```
    ps.setLong(2,
UserDB.selectID(favorite.getUser().getEmailAddress()));

    rs = ps.executeQuery();

    return rs.next();
   }
   catch (SQLException e) {
    e.printStackTrace();

    return false;
   }
   finally {
    DBUtil.closeResultSet(rs);
        DBUtil.closePreparedStatement(ps);
        pool.freeConnection(connection);
   }
  }

  public static boolean favoriteExists(User user, String pid) {
   ConnectionPool pool = ConnectionPool.getInstance();
   Connection connection = pool.getConnection();
   PreparedStatement ps = null;
   ResultSet rs = null;

   String query = "SELECT favorite_id FROM favorite WHERE
material_id = ? AND user_id = ?";

   try {
    ps = connection.prepareStatement(query);

    ps.setString(1, pid);
    ps.setLong(2, UserDB.selectID(user.getEmailAddress()));

    rs = ps.executeQuery();

    return rs.next();
   }
   catch (SQLException e) {
    e.printStackTrace();

    return false;
   }
   finally {
    DBUtil.closeResultSet(rs);
        DBUtil.closePreparedStatement(ps);
        pool.freeConnection(connection);
   }
  }

  public static LinkedHashMap<Long, Favorite> selectMine(User user) {
   ConnectionPool pool = ConnectionPool.getInstance();
      Connection connection = pool.getConnection();
      PreparedStatement ps = null;
      ResultSet rs = null;

      String query = "SELECT * FROM favorite WHERE user_id = ?";

      try {
       ps = connection.prepareStatement(query);
       ps.setLong(1, UserDB.selectID(user.getEmailAddress()));
       rs = ps.executeQuery();

       LinkedHashMap<Long, Favorite> favorite_list = new
LinkedHashMap<Long, Favorite>();

       while (rs.next()) {
        Favorite favorite = new Favorite();

        favorite.setDescription(rs.getString("description"));
        favorite.setTags(rs.getString("tags"));

favorite.setMaterial(MaterialDB.select(rs.getString("material_id")));
        favorite.setAuthor(favorite.getMaterial().getUser().getID());
        favorite.setUser(user);

        favorite_list.put(rs.getLong("favorite_id"), favorite);
       }

       return favorite_list;
      }
      catch (SQLException e) {
       e.printStackTrace();

       return null;
      }
      finally {
       DBUtil.closeResultSet(rs);
       DBUtil.closePreparedStatement(ps);
       pool.freeConnection(connection);
      }
  }

  public static int insert(Favorite favorite) {
   ConnectionPool pool = ConnectionPool.getInstance();
      Connection connection = pool.getConnection();
      PreparedStatement ps = null;
      ResultSet rs = null;

      String query = "INSERT INTO favorite (description, tags,
material_id, user_id) " +
            "VALUES (?, ?, ?, ?)";

      try {
       ps = connection.prepareStatement(query);

       ps.setString(1, favorite.getDescription());
       ps.setString(2, favorite.getTags());
       ps.setString(3, favorite.getMaterial().getPID());
       ps.setLong(4,
UserDB.selectID(favorite.getUser().getEmailAddress()));

       return ps.executeUpdate();
      }
      catch (SQLException e) {
       e.printStackTrace();

       return 0;
      }
      finally {
        DBUtil.closeResultSet(rs);
        DBUtil.closePreparedStatement(ps);
        pool.freeConnection(connection);
      }
```

```
  }

  public static int update(Favorite favorite) {
    ConnectionPool pool = ConnectionPool.getInstance();
      Connection connection = pool.getConnection();
      PreparedStatement ps = null;
      ResultSet rs = null;

      String query = "UPDATE favorite SET description = ? WHERE
material_id = ? AND user_id = ?";

      try {
        ps = connection.prepareStatement(query);

        ps.setString(1, favorite.getDescription());
          ps.setString(2, favorite.getMaterial().getPID());
          ps.setLong(3,
UserDB.selectID(favorite.getUser().getEmailAddress()));

        return ps.executeUpdate();
      }
      catch (SQLException e) {
        e.printStackTrace();

          return 0;
      }
      finally {
        DBUtil.closeResultSet(rs);
          DBUtil.closePreparedStatement(ps);
          pool.freeConnection(connection);
      }
  }

  public static int delete(String pid) {
    ConnectionPool pool = ConnectionPool.getInstance();
      Connection connection = pool.getConnection();
      PreparedStatement ps = null;
      ResultSet rs = null;

      String query = "DELETE FROM favorite WHERE material_id = ?";

      try {
        ps = connection.prepareStatement(query);
          ps.setString(1, pid);
          return ps.executeUpdate();
      }
      catch (SQLException e) {
        e.printStackTrace();

        return 0;
      }
      finally {
        DBUtil.closeResultSet(rs);
          DBUtil.closePreparedStatement(ps);
          pool.freeConnection(connection);
      }
  }

  public static int delete(User user, String pid) {
    ConnectionPool pool = ConnectionPool.getInstance();
      Connection connection = pool.getConnection();
      PreparedStatement ps = null;
      ResultSet rs = null;

      String query = "DELETE FROM favorite WHERE material_id = ?
AND user_id = ?";

      try {
        ps = connection.prepareStatement(query);
          ps.setString(1, pid);
          ps.setLong(2, UserDB.selectID(user.getEmailAddress()));

          return ps.executeUpdate();
      }
      catch (SQLException e) {
        e.printStackTrace();

        return 0;
      }
      finally {
        DBUtil.closeResultSet(rs);
          DBUtil.closePreparedStatement(ps);
          pool.freeConnection(connection);
      }
  }
}
```

FedoraConnect.java

```
package data;

import com.yourmediashelf.fedora.client.FedoraClient;
import com.yourmediashelf.fedora.client.FedoraCredentials;
import java.net.MalformedURLException;

public class FedoraConnect {
  FedoraCredentials credentials = null;
  FedoraClient fedora = null;
  private String base_url = "http://localhost:8084/fedora/";
  private String fdb_username = "fedoraAdmin";
  private String fdb_password = "local";

  public FedoraConnect() {
    try {
      credentials = new FedoraCredentials(base_url, fdb_username,
fdb_password);
      fedora = new FedoraClient(credentials);
    }
    catch(MalformedURLException e) {
      e.printStackTrace();
    }
  }

  public FedoraClient getClient() {
    return fedora;
  }
}
```

MaterialDB.java

```
package data;

import java.sql.Connection;
import java.sql.PreparedStatement;
```

```
import java.sql.ResultSet;
import java.sql.SQLException;

import java.util.ArrayList;

import domain.Community;
import domain.Material;
import domain.Metadata;
import domain.User;

public class MaterialDB {
  public static int countApproved() {
    ConnectionPool pool = ConnectionPool.getInstance();
    Connection connection = pool.getConnection();
    PreparedStatement ps = null;
    ResultSet rs = null;

    String query = "SELECT COUNT(pid) FROM digital_material
WHERE is_approved = 'y'";

    try {
      ps = connection.prepareStatement(query);
      rs = ps.executeQuery();

      if (rs.next()) {
        return rs.getInt(1);
      }
      else {
        return 0;
      }
    }
    catch (SQLException e) {
      e.printStackTrace();

      return 0;
    }
    finally {
      DBUtil.closeResultSet(rs);
          DBUtil.closePreparedStatement(ps);
          pool.freeConnection(connection);
    }
  }

  public static int countApproved(Community community) {
    ConnectionPool pool = ConnectionPool.getInstance();
    Connection connection = pool.getConnection();
    PreparedStatement ps = null;
    ResultSet rs = null;

    String query = "SELECT COUNT(pid) FROM digital_material
WHERE is_approved = 'y' AND community_id = ?";

    try {
      ps = connection.prepareStatement(query);
      ps.setLong(1, community.getID());
      rs = ps.executeQuery();

      if (rs.next()) {
        return rs.getInt(1);
      }
      else {
        return 0;
      }
    }
    catch (SQLException e) {
      e.printStackTrace();

      return 0;
    }
    finally {
      DBUtil.closeResultSet(rs);
          DBUtil.closePreparedStatement(ps);
          pool.freeConnection(connection);
    }
  }

  public static int countPending() {
    ConnectionPool pool = ConnectionPool.getInstance();
    Connection connection = pool.getConnection();
    PreparedStatement ps = null;
    ResultSet rs = null;

    String query = "SELECT COUNT(pid) FROM digital_material
WHERE is_approved = 'n'";

    try {
      ps = connection.prepareStatement(query);
      rs = ps.executeQuery();

      if (rs.next()) {
        return rs.getInt(1);
      }
      else {
        return 0;
      }
    }
    catch (SQLException e) {
      e.printStackTrace();

      return 0;
    }
    finally {
      DBUtil.closeResultSet(rs);
          DBUtil.closePreparedStatement(ps);
          pool.freeConnection(connection);
    }
  }

  public static int countPending(Community community) {
    ConnectionPool pool = ConnectionPool.getInstance();
    Connection connection = pool.getConnection();
    PreparedStatement ps = null;
    ResultSet rs = null;

    String query = "SELECT COUNT(pid) FROM digital_material
WHERE is_approved = 'n' AND community_id = ?";

    try {
      ps = connection.prepareStatement(query);
      ps.setLong(1, community.getID());
      rs = ps.executeQuery();

      if (rs.next()) {
        return rs.getInt(1);
```

```
    }
    else {
      return 0;
    }
  }
  catch (SQLException e) {
    e.printStackTrace();

    return 0;
  }
  finally {
    DBUtil.closeResultSet(rs);
        DBUtil.closePreparedStatement(ps);
        pool.freeConnection(connection);
  }
}

 public static int countApprovedFilter(String filter) {
   ConnectionPool pool = ConnectionPool.getInstance();
   Connection connection = pool.getConnection();
   PreparedStatement ps = null;
   ResultSet rs = null;

   if (filter.trim().equals("")) {

   }
   else {

   }

   String query = "SELECT COUNT(pid) FROM (SELECT
digital_material.pid, digital_material.dc_title, user.first_name,
user.last_name FROM digital_material INNER JOIN user ON
digital_material.depositor = user.user_id WHERE is_approved = 'y') AS
nested_table " + filter;

   try {
     ps = connection.prepareStatement(query);
     rs = ps.executeQuery();

     if (rs.next()) {
       return rs.getInt(1);
     }
     else {
       return 0;
     }
   }
   catch (SQLException e) {
     e.printStackTrace();

     return 0;
   }
   finally {
     DBUtil.closeResultSet(rs);
         DBUtil.closePreparedStatement(ps);
         pool.freeConnection(connection);
   }
 }

  public static int countApprovedFilter(Community community, String
filter) {
    ConnectionPool pool = ConnectionPool.getInstance();
    Connection connection = pool.getConnection();
    PreparedStatement ps = null;
    ResultSet rs = null;

    if (filter.trim().equals("")) {

    }
    else {

    }

    String query = "SELECT COUNT(pid) FROM (SELECT
digital_material.pid, digital_material.dc_title, user.first_name,
user.last_name FROM digital_material INNER JOIN user ON
digital_material.depositor = user.user_id WHERE is_approved = 'y'
AND digital_material.community_id = ?) AS nested_table " + filter;

    try {
      ps = connection.prepareStatement(query);
      ps.setLong(1, community.getID());
      rs = ps.executeQuery();

      if (rs.next()) {
        return rs.getInt(1);
      }
      else {
        return 0;
      }
    }
    catch (SQLException e) {
      e.printStackTrace();

      return 0;
    }
    finally {
      DBUtil.closeResultSet(rs);
          DBUtil.closePreparedStatement(ps);
          pool.freeConnection(connection);
    }
  }

  public static int countPendingFilter(String filter) {
    ConnectionPool pool = ConnectionPool.getInstance();
    Connection connection = pool.getConnection();
    PreparedStatement ps = null;
    ResultSet rs = null;

    if (filter.trim().equals("")) {

    }
    else {

    }

    String query = "SELECT COUNT(pid) FROM (SELECT
digital_material.pid, digital_material.dc_title, user.first_name,
user.last_name FROM digital_material INNER JOIN user ON
digital_material.depositor = user.user_id WHERE is_approved = 'n') AS
nested_table " + filter;

    try {
      ps = connection.prepareStatement(query);
```

```
      rs = ps.executeQuery();

      if (rs.next()) {
        return rs.getInt(1);
      }
      else {
        return 0;
      }
    }
    catch (SQLException e) {
      e.printStackTrace();

      return 0;
    }
    finally {
      DBUtil.closeResultSet(rs);
          DBUtil.closePreparedStatement(ps);
          pool.freeConnection(connection);
    }
  }

  public static int countPendingFilter(Community community, String
filter) {
    ConnectionPool pool = ConnectionPool.getInstance();
    Connection connection = pool.getConnection();
    PreparedStatement ps = null;
    ResultSet rs = null;

    if (filter.trim().equals("")) {

    }
    else {

    }

    String query = "SELECT COUNT(pid) FROM (SELECT
digital_material.pid, digital_material.dc_title, user.first_name,
user.last_name FROM digital_material INNER JOIN user ON
digital_material.depositor = user.user_id WHERE is_approved = 'n'
AND digital_material.community_id = ?) AS nested_table " + filter;

    try {
      ps = connection.prepareStatement(query);
      ps.setLong(1, community.getID());
      rs = ps.executeQuery();

      if (rs.next()) {
        return rs.getInt(1);
      }
      else {
        return 0;
      }
    }
    catch (SQLException e) {
      e.printStackTrace();

      return 0;
    }
    finally {
      DBUtil.closeResultSet(rs);
          DBUtil.closePreparedStatement(ps);
          pool.freeConnection(connection);
    }
  }

  public static int countMineApproved(User user) {
    ConnectionPool pool = ConnectionPool.getInstance();
    Connection connection = pool.getConnection();
    PreparedStatement ps = null;
    ResultSet rs = null;

    String query = "SELECT COUNT(pid) FROM digital_material
WHERE is_approved = 'y' AND depositor = ?";

    try {
      ps = connection.prepareStatement(query);

      ps.setLong(1, user.getID());

      rs = ps.executeQuery();

      if (rs.next()) {
        return rs.getInt(1);
      }
      else {
        return 0;
      }
    }
    catch (SQLException e) {
      e.printStackTrace();

      return 0;
    }
    finally {
      DBUtil.closeResultSet(rs);
          DBUtil.closePreparedStatement(ps);
          pool.freeConnection(connection);
    }
  }

  public static int countMinePending(User user) {
    ConnectionPool pool = ConnectionPool.getInstance();
    Connection connection = pool.getConnection();
    PreparedStatement ps = null;
    ResultSet rs = null;

    String query = "SELECT COUNT(pid) FROM digital_material
WHERE is_approved = 'n' AND depositor = ?";

    try {
      ps = connection.prepareStatement(query);

      ps.setLong(1, user.getID());

      rs = ps.executeQuery();

      if (rs.next()) {
        return rs.getInt(1);
      }
      else {
        return 0;
      }
    }
    catch (SQLException e) {
```

```
    e.printStackTrace();

    return 0;
   }
   finally {
    DBUtil.closeResultSet(rs);
        DBUtil.closePreparedStatement(ps);
        pool.freeConnection(connection);
   }
  }

  public static boolean materialExists(String pid) {
   ConnectionPool pool = ConnectionPool.getInstance();
   Connection connection = pool.getConnection();
   PreparedStatement ps = null;
   ResultSet rs = null;

   String query = "SELECT pid FROM digital_material WHERE pid = ?";

   try {
    ps = connection.prepareStatement(query);
    ps.setString(1, pid);
    rs = ps.executeQuery();

    return rs.next();
   }
   catch (SQLException e) {
    e.printStackTrace();

    return false;
   }
   finally {
    DBUtil.closeResultSet(rs);
        DBUtil.closePreparedStatement(ps);
        pool.freeConnection(connection);
   }
  }

  public static boolean materialExists(String pid, User user) {
   ConnectionPool pool = ConnectionPool.getInstance();
   Connection connection = pool.getConnection();
   PreparedStatement ps = null;
   ResultSet rs = null;

   String query = "SELECT pid FROM digital_material WHERE pid = ?
AND depositor = ?";

   try {
    ps = connection.prepareStatement(query);
    ps.setString(1, pid);
    ps.setLong(2, user.getID());
    rs = ps.executeQuery();

    return rs.next();
   }
   catch (SQLException e) {
    e.printStackTrace();

    return false;
   }
   finally {
    DBUtil.closeResultSet(rs);
        DBUtil.closePreparedStatement(ps);
        pool.freeConnection(connection);
   }
  }

  public static boolean materialExistsAndApproved(String pid) {
   ConnectionPool pool = ConnectionPool.getInstance();
   Connection connection = pool.getConnection();
   PreparedStatement ps = null;
   ResultSet rs = null;

   String query = "SELECT pid FROM digital_material WHERE pid = ?
AND is_approved = 'y'";

   try {
    ps = connection.prepareStatement(query);
    ps.setString(1, pid);
    rs = ps.executeQuery();

    return rs.next();
   }
   catch (SQLException e) {
    e.printStackTrace();

    return false;
   }
   finally {
    DBUtil.closeResultSet(rs);
        DBUtil.closePreparedStatement(ps);
        pool.freeConnection(connection);
   }
  }

  // Check - material exists, approved and embargo period is over
  public static boolean materialAccessible(String pid) {
   ConnectionPool pool = ConnectionPool.getInstance();
   Connection connection = pool.getConnection();
   PreparedStatement ps = null;
   ResultSet rs = null;

   String query = "SELECT pid FROM digital_material WHERE pid = ?
AND is_approved = 'y' AND (NOW() >= embargo OR embargo IS
NULL)";

   try {
    ps = connection.prepareStatement(query);
    ps.setString(1, pid);
    rs = ps.executeQuery();

    return rs.next();
   }
   catch (SQLException e) {
    e.printStackTrace();

    return false;
   }
   finally {
    DBUtil.closeResultSet(rs);
        DBUtil.closePreparedStatement(ps);
        pool.freeConnection(connection);
   }
  }
```

```
  public static boolean pendingMaterialExists(String pid, User user) {
    ConnectionPool pool = ConnectionPool.getInstance();
    Connection connection = pool.getConnection();
    PreparedStatement ps = null;
    ResultSet rs = null;

    String query = "SELECT pid FROM digital_material WHERE pid = ?
AND is_approved = 'n' AND depositor = ?";

    try {
      ps = connection.prepareStatement(query);
      ps.setString(1, pid);
      ps.setLong(2, user.getID());
      rs = ps.executeQuery();

      return rs.next();
    }
    catch (SQLException e) {
      e.printStackTrace();

      return false;
    }
    finally {
      DBUtil.closeResultSet(rs);
          DBUtil.closePreparedStatement(ps);
          pool.freeConnection(connection);
    }
  }

  public static Material select(String pid) {
    ConnectionPool pool = ConnectionPool.getInstance();
    Connection connection = pool.getConnection();
    PreparedStatement ps = null;
    ResultSet rs = null;

    String query = "SELECT * FROM digital_material WHERE pid = ?";

    try {
      ps = connection.prepareStatement(query);
      ps.setString(1, pid);
      rs = ps.executeQuery();

      if (rs.next()) {
        Material material = new Material();
        Metadata metadata = new Metadata();

        material.setPID(rs.getString("pid"));
        metadata.setContributor(rs.getString("dc_contributor"));
        metadata.setCoverage(rs.getString("dc_coverage"));
        metadata.setCreator(rs.getString("dc_creator"));
        metadata.setDate(rs.getDate("dc_date"));
        metadata.setDescription(rs.getString("dc_description"));
        metadata.setFormat(rs.getString("dc_format"));
        metadata.setIdentifier(rs.getString("dc_identifier"));
        metadata.setLanguage(rs.getString("dc_language"));
        metadata.setPublisher(rs.getString("dc_publisher"));
        metadata.setRelation(rs.getString("dc_relation"));
        metadata.setRights(rs.getString("dc_rights"));
        metadata.setSource(rs.getString("dc_source"));
        metadata.setSubject(rs.getString("dc_subject"));
        metadata.setTitle(rs.getString("dc_title"));
        metadata.setType(rs.getString("dc_type"));
        material.setMetadata(metadata);

material.setCommunity(CommunityDB.select(rs.getLong("community_i
d")));
        material.setTags(rs.getString("tags"));
        material.setHistory(rs.getString("history"));
        material.setEmbargo(rs.getDate("embargo"));
        material.setUser(UserDB.select(rs.getLong("depositor")));

material.setDepositTimestamp(rs.getTimestamp("deposit_timestamp"));
        material.setFilename(rs.getString("filename"));
        material.setFileType(rs.getString("file_type"));
        material.setFileSize(rs.getLong("file_size"));
        material.setIndex(rs.getString("index"));

        return material;
      }
      else {
        return null;
      }
    }
    catch (SQLException e) {
      e.printStackTrace();

      return null;
    }
    finally {
      DBUtil.closeResultSet(rs);
      DBUtil.closePreparedStatement(ps);
      pool.freeConnection(connection);
    }
  }

  public static Material selectLatest() {
    ConnectionPool pool = ConnectionPool.getInstance();
    Connection connection = pool.getConnection();
    PreparedStatement ps = null;
    ResultSet rs = null;

    String query = "SELECT * FROM digital_material WHERE
is_approved = 'y' ORDER BY deposit_timestamp DESC LIMIT 1";

    try {
      ps = connection.prepareStatement(query);
      rs = ps.executeQuery();

      if (rs.next()) {
        Material material = new Material();
        Metadata metadata = new Metadata();

        material.setPID(rs.getString("pid"));
        metadata.setContributor(rs.getString("dc_contributor"));
        metadata.setCoverage(rs.getString("dc_coverage"));
        metadata.setCreator(rs.getString("dc_creator"));
        metadata.setDate(rs.getDate("dc_date"));
        metadata.setDescription(rs.getString("dc_description"));
        metadata.setFormat(rs.getString("dc_format"));
        metadata.setIdentifier(rs.getString("dc_identifier"));
        metadata.setLanguage(rs.getString("dc_language"));
        metadata.setPublisher(rs.getString("dc_publisher"));
        metadata.setRelation(rs.getString("dc_relation"));
```

```
        metadata.setRights(rs.getString("dc_rights"));
        metadata.setSource(rs.getString("dc_source"));
        metadata.setSubject(rs.getString("dc_subject"));
        metadata.setTitle(rs.getString("dc_title"));
        metadata.setType(rs.getString("dc_type"));
        material.setMetadata(metadata);

material.setCommunity(CommunityDB.select(rs.getLong("community_i
d")));
        material.setTags(rs.getString("tags"));
        material.setHistory(rs.getString("history"));
        material.setEmbargo(rs.getDate("embargo"));
        material.setUser(UserDB.select(rs.getLong("depositor")));

material.setDepositTimestamp(rs.getTimestamp("deposit_timestamp"));
        material.setFilename(rs.getString("filename"));
        material.setFileType(rs.getString("file_type"));
        material.setFileSize(rs.getLong("file_size"));

        return material;
      }
      else {
        return null;
      }
    }
    catch (SQLException e) {
      e.printStackTrace();

      return null;
    }
    finally {
      DBUtil.closeResultSet(rs);
      DBUtil.closePreparedStatement(ps);
      pool.freeConnection(connection);
    }
  }

  public static Material selectLatest(User user) {
    ConnectionPool pool = ConnectionPool.getInstance();
    Connection connection = pool.getConnection();
    PreparedStatement ps = null;
    ResultSet rs = null;

    String query = "SELECT * FROM digital_material WHERE
is_approved = 'y' AND depositor = ? ORDER BY deposit_timestamp
DESC LIMIT 1";

    try {
      ps = connection.prepareStatement(query);
      ps.setLong(1, UserDB.selectID(user.getEmailAddress()));
      rs = ps.executeQuery();

      if (rs.next()) {
        Material material = new Material();
        Metadata metadata = new Metadata();

        material.setPID(rs.getString("pid"));
        metadata.setContributor(rs.getString("dc_contributor"));
        metadata.setCoverage(rs.getString("dc_coverage"));
        metadata.setCreator(rs.getString("dc_creator"));
        metadata.setDate(rs.getDate("dc_date"));
        metadata.setDescription(rs.getString("dc_description"));
        metadata.setFormat(rs.getString("dc_format"));
        metadata.setIdentifier(rs.getString("dc_identifier"));
        metadata.setLanguage(rs.getString("dc_language"));
        metadata.setPublisher(rs.getString("dc_publisher"));
        metadata.setRelation(rs.getString("dc_relation"));
        metadata.setRights(rs.getString("dc_rights"));
        metadata.setSource(rs.getString("dc_source"));
        metadata.setSubject(rs.getString("dc_subject"));
        metadata.setTitle(rs.getString("dc_title"));
        metadata.setType(rs.getString("dc_type"));
        material.setMetadata(metadata);

material.setCommunity(CommunityDB.select(rs.getLong("community_i
d")));
        material.setTags(rs.getString("tags"));
        material.setHistory(rs.getString("history"));
        material.setEmbargo(rs.getDate("embargo"));
        material.setUser(UserDB.select(rs.getLong("depositor")));

material.setDepositTimestamp(rs.getTimestamp("deposit_timestamp"));
        material.setFilename(rs.getString("filename"));
        material.setFileType(rs.getString("file_type"));
        material.setFileSize(rs.getLong("file_size"));

        return material;
      }
      else {
        return null;
      }
    }
    catch (SQLException e) {
      e.printStackTrace();

      return null;
    }
    finally {
      DBUtil.closeResultSet(rs);
      DBUtil.closePreparedStatement(ps);
      pool.freeConnection(connection);
    }
  }

  public static ArrayList<Material> selectApproved(String filter, String
order, String limit) {
    ConnectionPool pool = ConnectionPool.getInstance();
    Connection connection = pool.getConnection();
    PreparedStatement ps = null;
    ResultSet rs = null;

    String query = "SELECT * FROM (SELECT * FROM digital_material
WHERE is_approved = 'y') AS nested_table LEFT JOIN user ON
nested_table.depositor = user.user_id " + filter + " " + order + " " + limit;

    try {
      ps = connection.prepareStatement(query);
      rs = ps.executeQuery();
      ArrayList<Material> material_list = new ArrayList<Material>();

      while (rs.next()) {
        Material material = new Material();
        Metadata metadata = new Metadata();
```

```
        material.setPID(rs.getString("pid"));
        metadata.setContributor(rs.getString("dc_contributor"));
        metadata.setCoverage(rs.getString("dc_coverage"));
        metadata.setCreator(rs.getString("dc_creator"));
        metadata.setDate(rs.getDate("dc_date"));
        metadata.setDescription(rs.getString("dc_description"));
        metadata.setFormat(rs.getString("dc_format"));
        metadata.setIdentifier(rs.getString("dc_identifier"));
        metadata.setLanguage(rs.getString("dc_language"));
        metadata.setPublisher(rs.getString("dc_publisher"));
        metadata.setRelation(rs.getString("dc_relation"));
        metadata.setRights(rs.getString("dc_rights"));
        metadata.setSource(rs.getString("dc_source"));
        metadata.setSubject(rs.getString("dc_subject"));
        metadata.setTitle(rs.getString("dc_title"));
        metadata.setType(rs.getString("dc_type"));
        material.setMetadata(metadata);

material.setCommunity(CommunityDB.select(rs.getLong("community_i
d")));
        material.setTags(rs.getString("tags"));
        material.setHistory(rs.getString("history"));
        material.setEmbargo(rs.getDate("embargo"));
        material.setUser(UserDB.select(rs.getLong("depositor")));

material.setDepositTimestamp(rs.getTimestamp("deposit_timestamp"));
        material.setFilename(rs.getString("filename"));
        material.setFileType(rs.getString("file_type"));
        material.setFileSize(rs.getLong("file_size"));

        material_list.add(material);
      }

      return material_list;
    }
    catch (SQLException e) {
      e.printStackTrace();

      return null;
    }
    finally {
      DBUtil.closeResultSet(rs);
      DBUtil.closePreparedStatement(ps);
      pool.freeConnection(connection);
    }
  }

  public static ArrayList<Material> selectApproved(Community
community, String filter, String order, String limit) {
    ConnectionPool pool = ConnectionPool.getInstance();
    Connection connection = pool.getConnection();
    PreparedStatement ps = null;
    ResultSet rs = null;

    String query = "SELECT * FROM (SELECT * FROM digital_material
WHERE is_approved = 'y' AND digital_material.community_id = ?) AS
nested_table LEFT JOIN user ON nested_table.depositor =
user.user_id " + filter + " " + order + " " + limit;

    try {
      ps = connection.prepareStatement(query);
      ps.setLong(1, community.getID());
      rs = ps.executeQuery();
      ArrayList<Material> material_list = new ArrayList<Material>();

      while (rs.next()) {
        Material material = new Material();
        Metadata metadata = new Metadata();

        material.setPID(rs.getString("pid"));
        metadata.setContributor(rs.getString("dc_contributor"));
        metadata.setCoverage(rs.getString("dc_coverage"));
        metadata.setCreator(rs.getString("dc_creator"));
        metadata.setDate(rs.getDate("dc_date"));
        metadata.setDescription(rs.getString("dc_description"));
        metadata.setFormat(rs.getString("dc_format"));
        metadata.setIdentifier(rs.getString("dc_identifier"));
        metadata.setLanguage(rs.getString("dc_language"));
        metadata.setPublisher(rs.getString("dc_publisher"));
        metadata.setRelation(rs.getString("dc_relation"));
        metadata.setRights(rs.getString("dc_rights"));
        metadata.setSource(rs.getString("dc_source"));
        metadata.setSubject(rs.getString("dc_subject"));
        metadata.setTitle(rs.getString("dc_title"));
        metadata.setType(rs.getString("dc_type"));
        material.setMetadata(metadata);

material.setCommunity(CommunityDB.select(rs.getLong("community_i
d")));
        material.setTags(rs.getString("tags"));
        material.setHistory(rs.getString("history"));
        material.setEmbargo(rs.getDate("embargo"));
        material.setUser(UserDB.select(rs.getLong("depositor")));

material.setDepositTimestamp(rs.getTimestamp("deposit_timestamp"));
        material.setFilename(rs.getString("filename"));
        material.setFileType(rs.getString("file_type"));
        material.setFileSize(rs.getLong("file_size"));

        material_list.add(material);
      }

      return material_list;
    }
    catch (SQLException e) {
      e.printStackTrace();

      return null;
    }
    finally {
      DBUtil.closeResultSet(rs);
      DBUtil.closePreparedStatement(ps);
      pool.freeConnection(connection);
    }
  }

  public static ArrayList<Material> selectPending(String filter, String
order, String limit) {
    ConnectionPool pool = ConnectionPool.getInstance();
    Connection connection = pool.getConnection();
    PreparedStatement ps = null;
    ResultSet rs = null;
```

```
    String query = "SELECT * FROM (SELECT * FROM digital_material
WHERE is_approved = 'n') AS nested_table LEFT JOIN user ON
nested_table.depositor = user.user_id " + filter + " " + order + " " + limit;

    try {
      ps = connection.prepareStatement(query);
      rs = ps.executeQuery();
      ArrayList<Material> material_list = new ArrayList<Material>();

      while (rs.next()) {
        Material material = new Material();
        Metadata metadata = new Metadata();

        material.setPID(rs.getString("pid"));
        metadata.setContributor(rs.getString("dc_contributor"));
        metadata.setCoverage(rs.getString("dc_coverage"));
        metadata.setCreator(rs.getString("dc_creator"));
        metadata.setDate(rs.getDate("dc_date"));
        metadata.setDescription(rs.getString("dc_description"));
        metadata.setFormat(rs.getString("dc_format"));
        metadata.setIdentifier(rs.getString("dc_identifier"));
        metadata.setLanguage(rs.getString("dc_language"));
        metadata.setPublisher(rs.getString("dc_publisher"));
        metadata.setRelation(rs.getString("dc_relation"));
        metadata.setRights(rs.getString("dc_rights"));
        metadata.setSource(rs.getString("dc_source"));
        metadata.setSubject(rs.getString("dc_subject"));
        metadata.setTitle(rs.getString("dc_title"));
        metadata.setType(rs.getString("dc_type"));
        material.setMetadata(metadata);

material.setCommunity(CommunityDB.select(rs.getLong("community_i
d")));
        material.setTags(rs.getString("tags"));
        material.setHistory(rs.getString("history"));
        material.setEmbargo(rs.getDate("embargo"));
        material.setUser(UserDB.select(rs.getLong("depositor")));

material.setDepositTimestamp(rs.getTimestamp("deposit_timestamp"));
        material.setFilename(rs.getString("filename"));
        material.setFileType(rs.getString("file_type"));
        material.setFileSize(rs.getLong("file_size"));

        material_list.add(material);
      }

      return material_list;
    }
    catch (SQLException e) {
      e.printStackTrace();

      return null;
    }
    finally {
      DBUtil.closeResultSet(rs);
      DBUtil.closePreparedStatement(ps);
      pool.freeConnection(connection);
    }
  }

  public static ArrayList<Material> selectPending(Community
community, String filter, String order, String limit) {
    ConnectionPool pool = ConnectionPool.getInstance();
    Connection connection = pool.getConnection();
    PreparedStatement ps = null;
    ResultSet rs = null;

    String query = "SELECT * FROM (SELECT * FROM digital_material
WHERE is_approved = 'n' AND digital_material.community_id = ?) AS
nested_table LEFT JOIN user ON nested_table.depositor =
user.user_id " + filter + " " + order + " " + limit;

    try {
      ps = connection.prepareStatement(query);
      ps.setLong(1, community.getID());
      rs = ps.executeQuery();
      ArrayList<Material> material_list = new ArrayList<Material>();

      while (rs.next()) {
        Material material = new Material();
        Metadata metadata = new Metadata();

        material.setPID(rs.getString("pid"));
        metadata.setContributor(rs.getString("dc_contributor"));
        metadata.setCoverage(rs.getString("dc_coverage"));
        metadata.setCreator(rs.getString("dc_creator"));
        metadata.setDate(rs.getDate("dc_date"));
        metadata.setDescription(rs.getString("dc_description"));
        metadata.setFormat(rs.getString("dc_format"));
        metadata.setIdentifier(rs.getString("dc_identifier"));
        metadata.setLanguage(rs.getString("dc_language"));
        metadata.setPublisher(rs.getString("dc_publisher"));
        metadata.setRelation(rs.getString("dc_relation"));
        metadata.setRights(rs.getString("dc_rights"));
        metadata.setSource(rs.getString("dc_source"));
        metadata.setSubject(rs.getString("dc_subject"));
        metadata.setTitle(rs.getString("dc_title"));
        metadata.setType(rs.getString("dc_type"));
        material.setMetadata(metadata);

material.setCommunity(CommunityDB.select(rs.getLong("community_i
d")));
        material.setTags(rs.getString("tags"));
        material.setHistory(rs.getString("history"));
        material.setEmbargo(rs.getDate("embargo"));
        material.setUser(UserDB.select(rs.getLong("depositor")));

material.setDepositTimestamp(rs.getTimestamp("deposit_timestamp"));
        material.setFilename(rs.getString("filename"));
        material.setFileType(rs.getString("file_type"));
        material.setFileSize(rs.getLong("file_size"));

        material_list.add(material);
      }

      return material_list;
    }
    catch (SQLException e) {
      e.printStackTrace();

      return null;
    }
    finally {
      DBUtil.closeResultSet(rs);
```

```
      DBUtil.closePreparedStatement(ps);
      pool.freeConnection(connection);
    }
  }

  public static ArrayList<Material> selectFromCommunity(Community
community, String limit) {
    ConnectionPool pool = ConnectionPool.getInstance();
    Connection connection = pool.getConnection();
    PreparedStatement ps = null;
    ResultSet rs = null;

    String query = "SELECT * FROM digital_material WHERE
community_id = ? AND is_approved = 'y' ORDER BY
deposit_timestamp DESC " + limit;
    try {
      ps = connection.prepareStatement(query);
      ps.setLong(1, community.getID());
      rs = ps.executeQuery();

      ArrayList<Material> material_list = new ArrayList<Material>();

      while (rs.next()) {
        Material material = new Material();
        Metadata metadata = new Metadata();

        material.setPID(rs.getString("pid"));
        metadata.setContributor(rs.getString("dc_contributor"));
        metadata.setCoverage(rs.getString("dc_coverage"));
        metadata.setCreator(rs.getString("dc_creator"));
        metadata.setDate(rs.getDate("dc_date"));
        metadata.setDescription(rs.getString("dc_description"));
        metadata.setFormat(rs.getString("dc_format"));
        metadata.setIdentifier(rs.getString("dc_identifier"));
        metadata.setLanguage(rs.getString("dc_language"));
        metadata.setPublisher(rs.getString("dc_publisher"));
        metadata.setRelation(rs.getString("dc_relation"));
        metadata.setRights(rs.getString("dc_rights"));
        metadata.setSource(rs.getString("dc_source"));
        metadata.setSubject(rs.getString("dc_subject"));
        metadata.setTitle(rs.getString("dc_title"));
        metadata.setType(rs.getString("dc_type"));
        material.setMetadata(metadata);

material.setCommunity(CommunityDB.select(rs.getLong("community_i
d")));
        material.setTags(rs.getString("tags"));
        material.setHistory(rs.getString("history"));
        material.setEmbargo(rs.getDate("embargo"));
        material.setUser(UserDB.select(rs.getLong("depositor")));

material.setDepositTimestamp(rs.getTimestamp("deposit_timestamp"));
        material.setFilename(rs.getString("filename"));
        material.setFileType(rs.getString("file_type"));
        material.setFileSize(rs.getLong("file_size"));

        material_list.add(material);
      }

      return material_list;
    }
    catch (SQLException e) {
      e.printStackTrace();

      return null;
    }
    finally {
      DBUtil.closeResultSet(rs);
      DBUtil.closePreparedStatement(ps);
      pool.freeConnection(connection);
    }
  }

  public static ArrayList<Material> selectAll() {
    ConnectionPool pool = ConnectionPool.getInstance();
    Connection connection = pool.getConnection();
    PreparedStatement ps = null;
    ResultSet rs = null;

    String query = "SELECT * FROM digital_material";

    try {
      ps = connection.prepareStatement(query);
      rs = ps.executeQuery();
      ArrayList<Material> material_list = new ArrayList<Material>();

      while (rs.next()) {
        Material material = new Material();
        Metadata metadata = new Metadata();

        material.setPID(rs.getString("pid"));
        metadata.setContributor(rs.getString("dc_contributor"));
        metadata.setCoverage(rs.getString("dc_coverage"));
        metadata.setCreator(rs.getString("dc_creator"));
        metadata.setDate(rs.getDate("dc_date"));
        metadata.setDescription(rs.getString("dc_description"));
        metadata.setFormat(rs.getString("dc_format"));
        metadata.setIdentifier(rs.getString("dc_identifier"));
        metadata.setLanguage(rs.getString("dc_language"));
        metadata.setPublisher(rs.getString("dc_publisher"));
        metadata.setRelation(rs.getString("dc_relation"));
        metadata.setRights(rs.getString("dc_rights"));
        metadata.setSource(rs.getString("dc_source"));
        metadata.setSubject(rs.getString("dc_subject"));
        metadata.setTitle(rs.getString("dc_title"));
        metadata.setType(rs.getString("dc_type"));
        material.setMetadata(metadata);

material.setCommunity(CommunityDB.select(rs.getString("community_
code")));
        material.setTags(rs.getString("tags"));
        material.setHistory(rs.getString("history"));
        material.setEmbargo(rs.getDate("embargo"));
        material.setUser(UserDB.select(rs.getString("depositor")));

material.setDepositTimestamp(rs.getTimestamp("deposit_timestamp"));
        material.setFilename(rs.getString("filename"));
        material.setFileType(rs.getString("file_type"));
        material.setFileSize(rs.getLong("file_size"));

        material_list.add(material);
      }

      return material_list;
```

```
    }
    catch (SQLException e) {
      e.printStackTrace();

      return null;
    }
    finally {
      DBUtil.closeResultSet(rs);
      DBUtil.closePreparedStatement(ps);
      pool.freeConnection(connection);
    }
  }

  public static ArrayList<Material> selectRecent() {
    ConnectionPool pool = ConnectionPool.getInstance();
    Connection connection = pool.getConnection();
    PreparedStatement ps = null;
    ResultSet rs = null;

    String query = "SELECT * FROM digital_material WHERE
is_approved = 'y' ORDER BY deposit_timestamp DESC LIMIT 0, 15";

    try {
      ps = connection.prepareStatement(query);
      rs = ps.executeQuery();
      ArrayList<Material> material_list = new ArrayList<Material>();

      while (rs.next()) {
        Material material = new Material();
        Metadata metadata = new Metadata();

        material.setPID(rs.getString("pid"));
        metadata.setContributor(rs.getString("dc_contributor"));
        metadata.setCoverage(rs.getString("dc_coverage"));
        metadata.setCreator(rs.getString("dc_creator"));
        metadata.setDate(rs.getDate("dc_date"));
        metadata.setDescription(rs.getString("dc_description"));
        metadata.setFormat(rs.getString("dc_format"));
        metadata.setIdentifier(rs.getString("dc_identifier"));
        metadata.setLanguage(rs.getString("dc_language"));
        metadata.setPublisher(rs.getString("dc_publisher"));
        metadata.setRelation(rs.getString("dc_relation"));
        metadata.setRights(rs.getString("dc_rights"));
        metadata.setSource(rs.getString("dc_source"));
        metadata.setSubject(rs.getString("dc_subject"));
        metadata.setTitle(rs.getString("dc_title"));
        metadata.setType(rs.getString("dc_type"));
        material.setMetadata(metadata);

material.setCommunity(CommunityDB.select(rs.getLong("community_i
d")));
        material.setTags(rs.getString("tags"));
        material.setHistory(rs.getString("history"));
        material.setEmbargo(rs.getDate("embargo"));
        material.setUser(UserDB.select(rs.getLong("depositor")));

material.setDepositTimestamp(rs.getTimestamp("deposit_timestamp"));
        material.setFilename(rs.getString("filename"));
        material.setFileType(rs.getString("file_type"));
        material.setFileSize(rs.getLong("file_size"));

        material_list.add(material);
      }

      return material_list;
    }
    catch (SQLException e) {
      e.printStackTrace();

      return null;
    }
    finally {
      DBUtil.closeResultSet(rs);
      DBUtil.closePreparedStatement(ps);
      pool.freeConnection(connection);
    }
  }

  public static ArrayList<Material> selectRecent(Community community)
{
    ConnectionPool pool = ConnectionPool.getInstance();
    Connection connection = pool.getConnection();
    PreparedStatement ps = null;
    ResultSet rs = null;

    String query = "SELECT * FROM digital_material WHERE
is_approved = 'y' AND community_id = ? ORDER BY
deposit_timestamp DESC LIMIT 0, 5";

    try {
      ps = connection.prepareStatement(query);
      ps.setLong(1, community.getID());
      rs = ps.executeQuery();

      ArrayList<Material> material_list = new ArrayList<Material>();

      while (rs.next()) {
        Material material = new Material();
        Metadata metadata = new Metadata();

        material.setPID(rs.getString("pid"));
        metadata.setContributor(rs.getString("dc_contributor"));
        metadata.setCoverage(rs.getString("dc_coverage"));
        metadata.setCreator(rs.getString("dc_creator"));
        metadata.setDate(rs.getDate("dc_date"));
        metadata.setDescription(rs.getString("dc_description"));
        metadata.setFormat(rs.getString("dc_format"));
        metadata.setIdentifier(rs.getString("dc_identifier"));
        metadata.setLanguage(rs.getString("dc_language"));
        metadata.setPublisher(rs.getString("dc_publisher"));
        metadata.setRelation(rs.getString("dc_relation"));
        metadata.setRights(rs.getString("dc_rights"));
        metadata.setSource(rs.getString("dc_source"));
        metadata.setSubject(rs.getString("dc_subject"));
        metadata.setTitle(rs.getString("dc_title"));
        metadata.setType(rs.getString("dc_type"));
        material.setMetadata(metadata);

material.setCommunity(CommunityDB.select(rs.getLong("community_i
d")));
        material.setTags(rs.getString("tags"));
        material.setHistory(rs.getString("history"));
        material.setEmbargo(rs.getDate("embargo"));
        material.setUser(UserDB.select(rs.getLong("depositor")));
```

```
material.setDepositTimestamp(rs.getTimestamp("deposit_timestamp"));
        material.setFilename(rs.getString("filename"));
        material.setFileType(rs.getString("file_type"));
        material.setFileSize(rs.getLong("file_size"));

        material_list.add(material);
      }

      return material_list;
    }
    catch (SQLException e) {
      e.printStackTrace();

      return null;
    }
    finally {
      DBUtil.closeResultSet(rs);
      DBUtil.closePreparedStatement(ps);
      pool.freeConnection(connection);
    }
  }

  public static Material selectLatestVersion(String origin_pid) {
    ConnectionPool pool = ConnectionPool.getInstance();
    Connection connection = pool.getConnection();
    PreparedStatement ps = null;
    ResultSet rs = null;

    String query = "SELECT * FROM digital_material WHERE
is_approved = 'y' AND history = ? ORDER BY deposit_timestamp
DESC LIMIT 1";

    try {
      ps = connection.prepareStatement(query);
      ps.setString(1, origin_pid);
      rs = ps.executeQuery();

      if (rs.next()) {
        Material material = new Material();
        Metadata metadata = new Metadata();

        material.setPID(rs.getString("pid"));
        metadata.setContributor(rs.getString("dc_contributor"));
        metadata.setCoverage(rs.getString("dc_coverage"));
        metadata.setCreator(rs.getString("dc_creator"));
        metadata.setDate(rs.getDate("dc_date"));
        metadata.setDescription(rs.getString("dc_description"));
        metadata.setFormat(rs.getString("dc_format"));
        metadata.setIdentifier(rs.getString("dc_identifier"));
        metadata.setLanguage(rs.getString("dc_language"));
        metadata.setPublisher(rs.getString("dc_publisher"));
        metadata.setRelation(rs.getString("dc_relation"));
        metadata.setRights(rs.getString("dc_rights"));
        metadata.setSource(rs.getString("dc_source"));
        metadata.setSubject(rs.getString("dc_subject"));
        metadata.setTitle(rs.getString("dc_title"));
        metadata.setType(rs.getString("dc_type"));
        material.setMetadata(metadata);

material.setCommunity(CommunityDB.select(rs.getLong("community_i
d")));
        material.setTags(rs.getString("tags"));
        material.setHistory(rs.getString("history"));
        material.setEmbargo(rs.getDate("embargo"));
        material.setUser(UserDB.select(rs.getLong("depositor")));

material.setDepositTimestamp(rs.getTimestamp("deposit_timestamp"));
        material.setFilename(rs.getString("filename"));
        material.setFileType(rs.getString("file_type"));
        material.setFileSize(rs.getLong("file_size"));

        return material;
      }
      else {
        return null;
      }
    }
    catch (SQLException e) {
      e.printStackTrace();

      return null;
    }
    finally {
      DBUtil.closeResultSet(rs);
      DBUtil.closePreparedStatement(ps);
      pool.freeConnection(connection);
    }
  }

  public static Material selectLatestVersionAdmin(String origin_pid) {
    ConnectionPool pool = ConnectionPool.getInstance();
    Connection connection = pool.getConnection();
    PreparedStatement ps = null;
    ResultSet rs = null;

    String query = "SELECT * FROM digital_material WHERE history = ?
ORDER BY deposit_timestamp DESC LIMIT 1";

    try {
      ps = connection.prepareStatement(query);
      ps.setString(1, origin_pid);
      rs = ps.executeQuery();

      if (rs.next()) {
        Material material = new Material();
        Metadata metadata = new Metadata();

        material.setPID(rs.getString("pid"));
        metadata.setContributor(rs.getString("dc_contributor"));
        metadata.setCoverage(rs.getString("dc_coverage"));
        metadata.setCreator(rs.getString("dc_creator"));
        metadata.setDate(rs.getDate("dc_date"));
        metadata.setDescription(rs.getString("dc_description"));
        metadata.setFormat(rs.getString("dc_format"));
        metadata.setIdentifier(rs.getString("dc_identifier"));
        metadata.setLanguage(rs.getString("dc_language"));
        metadata.setPublisher(rs.getString("dc_publisher"));
        metadata.setRelation(rs.getString("dc_relation"));
        metadata.setRights(rs.getString("dc_rights"));
        metadata.setSource(rs.getString("dc_source"));
        metadata.setSubject(rs.getString("dc_subject"));
        metadata.setTitle(rs.getString("dc_title"));
        metadata.setType(rs.getString("dc_type"));
```

```
        material.setMetadata(metadata);

material.setCommunity(CommunityDB.select(rs.getLong("community_i
d")));
        material.setTags(rs.getString("tags"));
        material.setHistory(rs.getString("history"));
        material.setEmbargo(rs.getDate("embargo"));
        material.setUser(UserDB.select(rs.getLong("depositor")));

material.setDepositTimestamp(rs.getTimestamp("deposit_timestamp"));
        material.setFilename(rs.getString("filename"));
        material.setFileType(rs.getString("file_type"));
        material.setFileSize(rs.getLong("file_size"));

        return material;
      }
      else {
        return null;
      }
    }
    catch (SQLException e) {
      e.printStackTrace();

      return null;
    }
    finally {
      DBUtil.closeResultSet(rs);
      DBUtil.closePreparedStatement(ps);
      pool.freeConnection(connection);
    }
  }

  public static ArrayList<Material> selectCollection(String origin_pid) {
    ConnectionPool pool = ConnectionPool.getInstance();
    Connection connection = pool.getConnection();
    PreparedStatement ps = null;
    ResultSet rs = null;

    String query = "SELECT * FROM digital_material WHERE
is_approved = 'y' AND history = ? ORDER BY deposit_timestamp
DESC";

    try {
      ps = connection.prepareStatement(query);
      ps.setString(1, origin_pid);
      rs = ps.executeQuery();

      ArrayList<Material> material_list = new ArrayList<Material>();

      while (rs.next()) {
        Material material = new Material();
        Metadata metadata = new Metadata();

        material.setPID(rs.getString("pid"));
        metadata.setContributor(rs.getString("dc_contributor"));
        metadata.setCoverage(rs.getString("dc_coverage"));
        metadata.setCreator(rs.getString("dc_creator"));
        metadata.setDate(rs.getDate("dc_date"));
        metadata.setDescription(rs.getString("dc_description"));
        metadata.setFormat(rs.getString("dc_format"));
        metadata.setIdentifier(rs.getString("dc_identifier"));
        metadata.setLanguage(rs.getString("dc_language"));
        metadata.setPublisher(rs.getString("dc_publisher"));
        metadata.setRelation(rs.getString("dc_relation"));
        metadata.setRights(rs.getString("dc_rights"));
        metadata.setSource(rs.getString("dc_source"));
        metadata.setSubject(rs.getString("dc_subject"));
        metadata.setTitle(rs.getString("dc_title"));
        metadata.setType(rs.getString("dc_type"));
        material.setMetadata(metadata);

material.setCommunity(CommunityDB.select(rs.getLong("community_i
d")));
        material.setTags(rs.getString("tags"));
        material.setHistory(rs.getString("history"));
        material.setEmbargo(rs.getDate("embargo"));
        material.setUser(UserDB.select(rs.getLong("depositor")));

material.setDepositTimestamp(rs.getTimestamp("deposit_timestamp"));
        material.setFilename(rs.getString("filename"));
        material.setFileType(rs.getString("file_type"));
        material.setFileSize(rs.getLong("file_size"));

        material_list.add(material);
      }

      return material_list;
    }
    catch (SQLException e) {
      e.printStackTrace();

      return null;
    }
    finally {
      DBUtil.closeResultSet(rs);
      DBUtil.closePreparedStatement(ps);
      pool.freeConnection(connection);
    }
  }

  public static ArrayList<Material> selectMineApproved(User user) {
    ConnectionPool pool = ConnectionPool.getInstance();
    Connection connection = pool.getConnection();
    PreparedStatement ps = null;
    ResultSet rs = null;

    String query = "SELECT * FROM digital_material WHERE
is_approved = 'y' AND depositor = ? ORDER BY deposit_timestamp
DESC";

    try {
      ps = connection.prepareStatement(query);
      ps.setLong(1, UserDB.selectID(user.getEmailAddress()));
      rs = ps.executeQuery();

      ArrayList<Material> material_list = new ArrayList<Material>();

      while (rs.next()) {
        Material material = new Material();
        Metadata metadata = new Metadata();

        material.setPID(rs.getString("pid"));
        metadata.setContributor(rs.getString("dc_contributor"));
        metadata.setCoverage(rs.getString("dc_coverage"));
```

```
        metadata.setCreator(rs.getString("dc_creator"));
        metadata.setDate(rs.getDate("dc_date"));
        metadata.setDescription(rs.getString("dc_description"));
        metadata.setFormat(rs.getString("dc_format"));
        metadata.setIdentifier(rs.getString("dc_identifier"));
        metadata.setLanguage(rs.getString("dc_language"));
        metadata.setPublisher(rs.getString("dc_publisher"));
        metadata.setRelation(rs.getString("dc_relation"));
        metadata.setRights(rs.getString("dc_rights"));
        metadata.setSource(rs.getString("dc_source"));
        metadata.setSubject(rs.getString("dc_subject"));
        metadata.setTitle(rs.getString("dc_title"));
        metadata.setType(rs.getString("dc_type"));
        material.setMetadata(metadata);

material.setCommunity(CommunityDB.select(rs.getLong("community_i
d")));
        material.setTags(rs.getString("tags"));
        material.setHistory(rs.getString("history"));
        material.setEmbargo(rs.getDate("embargo"));
        material.setUser(UserDB.select(rs.getLong("depositor")));

material.setDepositTimestamp(rs.getTimestamp("deposit_timestamp"));
        material.setFilename(rs.getString("filename"));
        material.setFileType(rs.getString("file_type"));
        material.setFileSize(rs.getLong("file_size"));

        material_list.add(material);
      }

      return material_list;
    }
    catch (SQLException e) {
      e.printStackTrace();

      return null;
    }
    finally {
      DBUtil.closeResultSet(rs);
      DBUtil.closePreparedStatement(ps);
      pool.freeConnection(connection);
    }
  }

  public static ArrayList<Material> selectMinePending(User user) {
    ConnectionPool pool = ConnectionPool.getInstance();
    Connection connection = pool.getConnection();
    PreparedStatement ps = null;
    ResultSet rs = null;

    String query = "SELECT * FROM digital_material WHERE
is_approved = 'n' AND depositor = ? ORDER BY deposit_timestamp
DESC";

    try {
      ps = connection.prepareStatement(query);
      ps.setLong(1, UserDB.selectID(user.getEmailAddress()));
      rs = ps.executeQuery();

      ArrayList<Material> material_list = new ArrayList<Material>();

      while (rs.next()) {
        Material material = new Material();
        Metadata metadata = new Metadata();

        material.setPID(rs.getString("pid"));
        metadata.setContributor(rs.getString("dc_contributor"));
        metadata.setCoverage(rs.getString("dc_coverage"));
        metadata.setCreator(rs.getString("dc_creator"));
        metadata.setDate(rs.getDate("dc_date"));
        metadata.setDescription(rs.getString("dc_description"));
        metadata.setFormat(rs.getString("dc_format"));
        metadata.setIdentifier(rs.getString("dc_identifier"));
        metadata.setLanguage(rs.getString("dc_language"));
        metadata.setPublisher(rs.getString("dc_publisher"));
        metadata.setRelation(rs.getString("dc_relation"));
        metadata.setRights(rs.getString("dc_rights"));
        metadata.setSource(rs.getString("dc_source"));
        metadata.setSubject(rs.getString("dc_subject"));
        metadata.setTitle(rs.getString("dc_title"));
        metadata.setType(rs.getString("dc_type"));
        material.setMetadata(metadata);

material.setCommunity(CommunityDB.select(rs.getLong("community_i
d")));
        material.setTags(rs.getString("tags"));
        material.setHistory(rs.getString("history"));
        material.setEmbargo(rs.getDate("embargo"));
        material.setUser(UserDB.select(rs.getLong("depositor")));

material.setDepositTimestamp(rs.getTimestamp("deposit_timestamp"));
        material.setFilename(rs.getString("filename"));
        material.setFileType(rs.getString("file_type"));
        material.setFileSize(rs.getLong("file_size"));

        material_list.add(material);
      }

      return material_list;
    }
    catch (SQLException e) {
      e.printStackTrace();

      return null;
    }
    finally {
      DBUtil.closeResultSet(rs);
      DBUtil.closePreparedStatement(ps);
      pool.freeConnection(connection);
    }
  }

  public static String selectThumbnail(String pid) {
    ConnectionPool pool = ConnectionPool.getInstance();
    Connection connection = pool.getConnection();
    PreparedStatement ps = null;
    ResultSet rs = null;

    String query = "SELECT `image` FROM digital_material WHERE pid
= ?";

    try {
      ps = connection.prepareStatement(query);
      ps.setString(1, pid);
```

```
    rs = ps.executeQuery();

    if (rs.next()) {
      return rs.getString("image");
    }
    else {
      return null;
    }
   }
   catch (SQLException e) {
    e.printStackTrace();

    return null;
   }
   finally {
    DBUtil.closeResultSet(rs);
    DBUtil.closePreparedStatement(ps);
    pool.freeConnection(connection);
   }
  }

  public static ArrayList<Material> search(String search_query) {
    ConnectionPool pool = ConnectionPool.getInstance();
    Connection connection = pool.getConnection();
    PreparedStatement ps = null;
    ResultSet rs = null;

    String query = "SELECT *, MATCH (dc_contributor, dc_coverage,
dc_creator, dc_description, dc_format, dc_identifier, dc_language,
dc_publisher, dc_relation, dc_rights, dc_source, dc_subject, dc_title,
dc_type, tags, filename, `index`) AGAINST (? IN BOOLEAN MODE) AS
score FROM digital_material WHERE MATCH(dc_contributor,
dc_coverage, dc_creator, dc_description, dc_format, dc_identifier,
dc_language, dc_publisher, dc_relation, dc_rights, dc_source,
dc_subject, dc_title, dc_type, tags, filename, `index`) AGAINST (? IN
BOOLEAN MODE) AND is_approved = 'y' ORDER BY score DESC";

    try {
      ps = connection.prepareStatement(query);
      ps.setString(1, search_query);
      ps.setString(2, search_query);
      rs = ps.executeQuery();

      ArrayList<Material> material_list = new ArrayList<Material>();

      while (rs.next()) {
        Material material = new Material();
        Metadata metadata = new Metadata();

        material.setPID(rs.getString("pid"));
        metadata.setContributor(rs.getString("dc_contributor"));
        metadata.setCoverage(rs.getString("dc_coverage"));
        metadata.setCreator(rs.getString("dc_creator"));
        metadata.setDate(rs.getDate("dc_date"));
        metadata.setDescription(rs.getString("dc_description"));
        metadata.setFormat(rs.getString("dc_format"));
        metadata.setIdentifier(rs.getString("dc_identifier"));
        metadata.setLanguage(rs.getString("dc_language"));
        metadata.setPublisher(rs.getString("dc_publisher"));
        metadata.setRelation(rs.getString("dc_relation"));
        metadata.setRights(rs.getString("dc_rights"));
        metadata.setSource(rs.getString("dc_source"));
        metadata.setSubject(rs.getString("dc_subject"));
        metadata.setTitle(rs.getString("dc_title"));
        metadata.setType(rs.getString("dc_type"));
        material.setMetadata(metadata);

material.setCommunity(CommunityDB.select(rs.getLong("community_i
d")));
        material.setTags(rs.getString("tags"));
        material.setHistory(rs.getString("history"));
        material.setEmbargo(rs.getDate("embargo"));
        material.setUser(UserDB.select(rs.getLong("depositor")));

material.setDepositTimestamp(rs.getTimestamp("deposit_timestamp"));
        material.setFilename(rs.getString("filename"));
        material.setFileType(rs.getString("file_type"));
        material.setFileSize(rs.getLong("file_size"));
        material.setIndex(rs.getString("index"));

        material_list.add(material);
      }

      return material_list;
    }
    catch (SQLException e) {
      e.printStackTrace();

      return null;
    }
    finally {
      DBUtil.closeResultSet(rs);
      DBUtil.closePreparedStatement(ps);
      pool.freeConnection(connection);
    }
   }

   public static ArrayList<Material> search(String search_query,
Community community) {
    ConnectionPool pool = ConnectionPool.getInstance();
    Connection connection = pool.getConnection();
    PreparedStatement ps = null;
    ResultSet rs = null;

    String query = "SELECT *, MATCH (dc_contributor, dc_coverage,
dc_creator, dc_description, dc_format, dc_identifier, dc_language,
dc_publisher, dc_relation, dc_rights, dc_source, dc_subject, dc_title,
dc_type, tags, filename, `index`) AGAINST (? IN BOOLEAN MODE) AS
score FROM digital_material WHERE MATCH(dc_contributor,
dc_coverage, dc_creator, dc_description, dc_format, dc_identifier,
dc_language, dc_publisher, dc_relation, dc_rights, dc_source,
dc_subject, dc_title, dc_type, tags, filename, `index`) AGAINST (? IN
BOOLEAN MODE) AND is_approved = 'y' AND community_id = ?
ORDER BY score DESC";

    try {
      ps = connection.prepareStatement(query);
      ps.setString(1, search_query);
      ps.setString(2, search_query);
      ps.setLong(3, community.getID());
      rs = ps.executeQuery();

      ArrayList<Material> material_list = new ArrayList<Material>();
```

```
      while (rs.next()) {
        Material material = new Material();
        Metadata metadata = new Metadata();

        material.setPID(rs.getString("pid"));
        metadata.setContributor(rs.getString("dc_contributor"));
        metadata.setCoverage(rs.getString("dc_coverage"));
        metadata.setCreator(rs.getString("dc_creator"));
        metadata.setDate(rs.getDate("dc_date"));
        metadata.setDescription(rs.getString("dc_description"));
        metadata.setFormat(rs.getString("dc_format"));
        metadata.setIdentifier(rs.getString("dc_identifier"));
        metadata.setLanguage(rs.getString("dc_language"));
        metadata.setPublisher(rs.getString("dc_publisher"));
        metadata.setRelation(rs.getString("dc_relation"));
        metadata.setRights(rs.getString("dc_rights"));
        metadata.setSource(rs.getString("dc_source"));
        metadata.setSubject(rs.getString("dc_subject"));
        metadata.setTitle(rs.getString("dc_title"));
        metadata.setType(rs.getString("dc_type"));
        material.setMetadata(metadata);

material.setCommunity(CommunityDB.select(rs.getLong("community_i
d")));
        material.setTags(rs.getString("tags"));
        material.setHistory(rs.getString("history"));
        material.setEmbargo(rs.getDate("embargo"));
        material.setUser(UserDB.select(rs.getLong("depositor")));

material.setDepositTimestamp(rs.getTimestamp("deposit_timestamp"));
        material.setFilename(rs.getString("filename"));
        material.setFileType(rs.getString("file_type"));
        material.setFileSize(rs.getLong("file_size"));
        material.setIndex(rs.getString("index"));

        material_list.add(material);
      }

      return material_list;
    }
    catch (SQLException e) {
      e.printStackTrace();

      return null;
    }
    finally {
      DBUtil.closeResultSet(rs);
      DBUtil.closePreparedStatement(ps);
      pool.freeConnection(connection);
    }
  }

  public static ArrayList<Material> search(String search_query, String
mysql_query) {
    ConnectionPool pool = ConnectionPool.getInstance();
    Connection connection = pool.getConnection();
    PreparedStatement ps = null;
    ResultSet rs = null;

    String query = mysql_query;

    try {
      ps = connection.prepareStatement(query);
      ps.setString(1, search_query);
      ps.setString(2, search_query);
      rs = ps.executeQuery();

      ArrayList<Material> material_list = new ArrayList<Material>();

      while (rs.next()) {
        Material material = new Material();
        Metadata metadata = new Metadata();

        material.setPID(rs.getString("pid"));
        metadata.setContributor(rs.getString("dc_contributor"));
        metadata.setCoverage(rs.getString("dc_coverage"));
        metadata.setCreator(rs.getString("dc_creator"));
        metadata.setDate(rs.getDate("dc_date"));
        metadata.setDescription(rs.getString("dc_description"));
        metadata.setFormat(rs.getString("dc_format"));
        metadata.setIdentifier(rs.getString("dc_identifier"));
        metadata.setLanguage(rs.getString("dc_language"));
        metadata.setPublisher(rs.getString("dc_publisher"));
        metadata.setRelation(rs.getString("dc_relation"));
        metadata.setRights(rs.getString("dc_rights"));
        metadata.setSource(rs.getString("dc_source"));
        metadata.setSubject(rs.getString("dc_subject"));
        metadata.setTitle(rs.getString("dc_title"));
        metadata.setType(rs.getString("dc_type"));
        material.setMetadata(metadata);

material.setCommunity(CommunityDB.select(rs.getLong("community_i
d")));
        material.setTags(rs.getString("tags"));
        material.setHistory(rs.getString("history"));
        material.setEmbargo(rs.getDate("embargo"));
        material.setUser(UserDB.select(rs.getLong("depositor")));

material.setDepositTimestamp(rs.getTimestamp("deposit_timestamp"));
        material.setFilename(rs.getString("filename"));
        material.setFileType(rs.getString("file_type"));
        material.setFileSize(rs.getLong("file_size"));
        material.setIndex(rs.getString("index"));

        material_list.add(material);
      }

      return material_list;
    }
    catch (SQLException e) {
      e.printStackTrace();

      return null;
    }
    finally {
      DBUtil.closeResultSet(rs);
      DBUtil.closePreparedStatement(ps);
      pool.freeConnection(connection);
    }
  }

  public static int insert(String is_approved, Material material) {
    ConnectionPool pool = ConnectionPool.getInstance();
      Connection connection = pool.getConnection();
```

```
        PreparedStatement ps = null;
        ResultSet rs = null;

        String query = "INSERT INTO digital_material " +
                    "(pid,          is_approved, dc_contributor, dc_coverage,
dc_creator,       dc_date,"     +
                    " dc_description, dc_format,   dc_identifier,  dc_language,
dc_publisher,     dc_relation,"  +
                    " dc_rights,     dc_source,   dc_subject,     dc_title,
dc_type,          community_id," +
                    " tags,          history,     embargo,       depositor,
deposit_timestamp, filename,    " +
                    " file_type,     file_size,   `index`)" +
                    " VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?,
?, ?, ?, NOW(), ?, ?, ?, ?)";

        try {
            ps = connection.prepareStatement(query);

            ps.setString(1, material.getPID());
            ps.setString(2, is_approved);
            ps.setString(3, material.getMetadata().getContributor());
            ps.setString(4, material.getMetadata().getCoverage());
            ps.setString(5, material.getMetadata().getCreator());
            ps.setString(6, material.getMetadata().getDate());
            ps.setString(7, material.getMetadata().getDescription());
            ps.setString(8, material.getMetadata().getFormat());
            ps.setString(9, material.getMetadata().getIdentifier());
            ps.setString(10, material.getMetadata().getLanguage());
            ps.setString(11, material.getMetadata().getPublisher());
            ps.setString(12, material.getMetadata().getRelation());
            ps.setString(13, material.getMetadata().getRights());
            ps.setString(14, material.getMetadata().getSource());
            ps.setString(15, material.getMetadata().getSubject());
            ps.setString(16, material.getMetadata().getTitle());
            ps.setString(17, material.getMetadata().getType());
            ps.setLong(18,
CommunityDB.selectID(material.getCommunity().getCode()));
            ps.setString(19, material.getTags());
            ps.setString(20, material.getHistory());
            ps.setString(21, material.getEmbargo());
            ps.setLong(22,
UserDB.selectID(material.getUser().getEmailAddress()));
            ps.setString(23, material.getFilename());
            ps.setString(24, material.getFileType());
            ps.setLong(25, material.getFileSize());
            ps.setString(26, material.getIndex());

            return ps.executeUpdate();
        }
        catch (SQLException e) {
            e.printStackTrace();

            return 0;
        }
        finally {
            DBUtil.closeResultSet(rs);
            DBUtil.closePreparedStatement(ps);
            pool.freeConnection(connection);
        }
    }
```

```
    public static int approve(String pid) {
      ConnectionPool pool = ConnectionPool.getInstance();
        Connection connection = pool.getConnection();
        PreparedStatement ps = null;
        ResultSet rs = null;

        String query = "UPDATE digital_material SET " +
                    "is_approved = 'y', deposit_timestamp = NOW() " +
                    "WHERE pid = ?";

        try {
          ps = connection.prepareStatement(query);

            ps.setString(1, pid);

            return ps.executeUpdate();
        }
        catch (SQLException e) {
          e.printStackTrace();

            return 0;
        }
        finally {
          DBUtil.closeResultSet(rs);
            DBUtil.closePreparedStatement(ps);
            pool.freeConnection(connection);
        }
    }

    public static int approve(Material material) {
      ConnectionPool pool = ConnectionPool.getInstance();
        Connection connection = pool.getConnection();
        PreparedStatement ps = null;
        ResultSet rs = null;

        String query = "UPDATE digital_material SET " +
                    "is_approved = 'y' " +
                    "WHERE pid = ?";

        try {
          ps = connection.prepareStatement(query);

            ps.setString(1, material.getPID());

            return ps.executeUpdate();
        }
        catch (SQLException e) {
          e.printStackTrace();

            return 0;
        }
        finally {
          DBUtil.closeResultSet(rs);
            DBUtil.closePreparedStatement(ps);
            pool.freeConnection(connection);
        }
    }

    public static int insertThumbnail(Material material, String name) {
      ConnectionPool pool = ConnectionPool.getInstance();
        Connection connection = pool.getConnection();
        PreparedStatement ps = null;
```

```
    ResultSet rs = null;

    String query = "UPDATE digital_material SET " +
                "`image` = ? " +
                "WHERE pid = ?";

    try {
      ps = connection.prepareStatement(query);

      ps.setString(1, name);
         ps.setString(2, material.getPID());

        return ps.executeUpdate();
    }
    catch (SQLException e) {
      e.printStackTrace();

        return 0;
    }
    finally {
      DBUtil.closeResultSet(rs);
        DBUtil.closePreparedStatement(ps);
        pool.freeConnection(connection);
    }
}

public static int update(Material material) {
  ConnectionPool pool = ConnectionPool.getInstance();
    Connection connection = pool.getConnection();
    PreparedStatement ps = null;
    ResultSet rs = null;

    String query = "UPDATE digital_material SET " +
                "dc_contributor = ?, " +
                "dc_coverage = ?, "    +
                "dc_creator = ?, "     +
                "dc_date = ?, "        +
                "dc_description = ?, " +
                "dc_format = ?, "      +
                "dc_identifier = ?, "  +
                "dc_language = ?, "    +
                "dc_publisher = ?, "   +
                "dc_relation = ?, "    +
                "dc_rights = ?, "      +
                "dc_source = ?, "      +
                "dc_subject = ?, "     +
                "dc_title = ?, "       +
                "dc_type = ?, "        +
                "community_id = ?, "   +
                "tags = ?, "           +
                "embargo = ? "         +
                "WHERE pid = ?";

    try {
      ps = connection.prepareStatement(query);

        ps.setString(1, material.getMetadata().getContributor());
        ps.setString(2, material.getMetadata().getCoverage());
        ps.setString(3, material.getMetadata().getCreator());
        ps.setString(4, material.getMetadata().getDate());
        ps.setString(5, material.getMetadata().getDescription());
        ps.setString(6, material.getMetadata().getFormat());
        ps.setString(7, material.getMetadata().getIdentifier());
        ps.setString(8, material.getMetadata().getLanguage());
        ps.setString(9, material.getMetadata().getPublisher());
        ps.setString(10, material.getMetadata().getRelation());
        ps.setString(11, material.getMetadata().getRights());
        ps.setString(12, material.getMetadata().getSource());
        ps.setString(13, material.getMetadata().getSubject());
        ps.setString(14, material.getMetadata().getTitle());
        ps.setString(15, material.getMetadata().getType());
        ps.setLong(16,
CommunityDB.selectID(material.getCommunity().getCode()));
        ps.setString(17, material.getTags());
        ps.setString(18, material.getEmbargo());
        ps.setString(19, material.getPID());

        return ps.executeUpdate();
      }
      catch (SQLException e) {
        e.printStackTrace();

        return 0;
      }
      finally {
        DBUtil.closeResultSet(rs);
        DBUtil.closePreparedStatement(ps);
        pool.freeConnection(connection);
      }
    }

    public static int delete(String pid) {
      ConnectionPool pool = ConnectionPool.getInstance();
      Connection connection = pool.getConnection();
      PreparedStatement ps = null;
      ResultSet rs = null;

      String query = "DELETE FROM digital_material WHERE pid = ?";

      try {
        ps = connection.prepareStatement(query);
        ps.setString(1, pid);

        return ps.executeUpdate();
      }
      catch (SQLException e) {
        e.printStackTrace();

        return 0;
      }
      finally {
        DBUtil.closeResultSet(rs);
        DBUtil.closePreparedStatement(ps);
        pool.freeConnection(connection);
      }
    }

    public static int delete(User user, String pid) {
      ConnectionPool pool = ConnectionPool.getInstance();
      Connection connection = pool.getConnection();
      PreparedStatement ps = null;
      ResultSet rs = null;
```

```
    String query = "DELETE FROM digital_material WHERE depositor
= ? AND pid = ?";

    try {
      ps = connection.prepareStatement(query);

      ps.setLong(1, user.getID());
        ps.setString(2, pid);

        return ps.executeUpdate();
    }
    catch (SQLException e) {
      e.printStackTrace();

        return 0;
    }
    finally {
      DBUtil.closeResultSet(rs);
        DBUtil.closePreparedStatement(ps);
        pool.freeConnection(connection);
    }
  }

  public Material getLatestMaterial() {
    return selectLatest();
  }

  public ArrayList<Material> getRecentMaterial() {
    return selectRecent();
  }

  public int getPendingCount() {
    return countPending();
  }
}
```

MaterialFedora.java

```
package data;

import java.io.File;

import java.util.Date;

import org.apache.commons.io.FilenameUtils;

import com.yourmediashelf.fedora.client.FedoraClient;
import com.yourmediashelf.fedora.client.FedoraClientException;
import com.yourmediashelf.fedora.client.request.GetDatastream;
import com.yourmediashelf.fedora.client.request.Ingest;

import
com.yourmediashelf.fedora.client.response.GetDatastreamResponse;
import com.yourmediashelf.fedora.client.response.IngestResponse;

import static
com.yourmediashelf.fedora.client.FedoraClient.addDatastream;
import static
com.yourmediashelf.fedora.client.FedoraClient.modifyObject;
import static
com.yourmediashelf.fedora.client.FedoraClient.purgeObject;

import domain.Material;

public class MaterialFedora {
  public static boolean materialExists(String pid) {
    FedoraConnect connect = new FedoraConnect();
    FedoraClient client = connect.getClient();
    GetDatastreamResponse response = null;

    try {
      response = new GetDatastream(pid, "DATA").execute(client);

      return true;
    }
    catch (FedoraClientException e) {
      e.printStackTrace();

      return false;
    }
  }

  public static String insert(Material material, File file) {
    FedoraConnect connect = new FedoraConnect();
    FedoraClient client = connect.getClient();
    IngestResponse response = null;

    try {
      String ds =
FilenameUtils.removeExtension(material.getFilename());

      response = new
Ingest().namespace("get").label(material.getMetadata().getTitle()).exec
ute(client);
      addDatastream(response.getPid(),
"DATA").dsLabel(ds).controlGroup("M").mimeType(material.getFileTyp
e()).content(file).execute(client);

      return response.getPid();
    }
    catch (FedoraClientException e) {
      e.printStackTrace();

      return null;
    }
  }

  // Update - later

  public static boolean update(Material material) {
    FedoraConnect connect = new FedoraConnect();
    FedoraClient client = connect.getClient();

    try {
      Date lastModifiedDate =
client.getLastModifiedDate(material.getPID());


modifyObject(material.getPID()).label(material.getMetadata().getTitle()).
lastModifiedDate(lastModifiedDate).execute(client);

      return true;
    }
    catch (FedoraClientException e) {
```

```
      e.printStackTrace();

      return false;
    }
  }

  public static boolean update(Material material, File file) {
    FedoraConnect connect = new FedoraConnect();
    FedoraClient client = connect.getClient();

    try {
      Date lastModifiedDate =
client.getLastModifiedDate(material.getPID());


modifyObject(material.getPID()).label(material.getMetadata().getTitle()).
lastModifiedDate(lastModifiedDate).execute(client);

      return true;
    }
    catch (FedoraClientException e) {
      e.printStackTrace();

      return false;
    }
  }

  public static boolean delete(String pid) {
    FedoraConnect connect = new FedoraConnect();
    FedoraClient client = connect.getClient();

    try {
      purgeObject(pid).execute(client);

      return true;
    }
    catch (FedoraClientException e) {
      e.printStackTrace();

      return false;
    }
  }
}
```

MessageDB.java

```
package data;

import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.ArrayList;

import domain.Message;
import domain.User;

public class MessageDB {
  public static int count(User user) {
    ConnectionPool pool = ConnectionPool.getInstance();
    Connection connection = pool.getConnection();
    PreparedStatement ps = null;
    ResultSet rs = null;

    String query = "SELECT COUNT(message_id) FROM message
WHERE recipient = ?";

    try {
      ps = connection.prepareStatement(query);

      ps.setLong(1, user.getID());

      rs = ps.executeQuery();

      if (rs.next()) {
        return rs.getInt(1);
      }
      else {
        return 0;
      }
    }
    catch (SQLException e) {
      e.printStackTrace();

      return 0;
    }
    finally {
      DBUtil.closeResultSet(rs);
          DBUtil.closePreparedStatement(ps);
          pool.freeConnection(connection);
    }
  }

  public static int countModerator(User user) {
    ConnectionPool pool = ConnectionPool.getInstance();
    Connection connection = pool.getConnection();
    PreparedStatement ps = null;
    ResultSet rs = null;

    String query = "SELECT COUNT(message_id) FROM message
LEFT JOIN digital_material ON message.material_id =
digital_material.pid WHERE recipient = ? OR community_id = ?";

    try {
      ps = connection.prepareStatement(query);

      ps.setLong(1, user.getID());
      ps.setLong(2, user.getCommunity().getID());

      rs = ps.executeQuery();

      if (rs.next()) {
        return rs.getInt(1);
      }
      else {
        return 0;
      }
    }
    catch (SQLException e) {
      e.printStackTrace();

      return 0;
    }
    finally {
```

```
    DBUtil.closeResultSet(rs);
        DBUtil.closePreparedStatement(ps);
        pool.freeConnection(connection);
  }
 }

 public static int countAdmin(User user) {
  ConnectionPool pool = ConnectionPool.getInstance();
  Connection connection = pool.getConnection();
  PreparedStatement ps = null;
  ResultSet rs = null;

  String query = "SELECT COUNT(message_id) FROM message
WHERE recipient = ? OR type = 'flag' OR type = 'flag_resolved' OR
type = 'support' OR type = 'support_resolved' OR type = 'annotation'";

  try {
   ps = connection.prepareStatement(query);

   ps.setLong(1, user.getID());

   rs = ps.executeQuery();

   if (rs.next()) {
    return rs.getInt(1);
   }
   else {
    return 0;
   }
  }
  catch (SQLException e) {
   e.printStackTrace();

   return 0;
  }
  finally {
   DBUtil.closeResultSet(rs);
        DBUtil.closePreparedStatement(ps);
        pool.freeConnection(connection);
  }
 }

 public static int countFilter(String filter, User user) {
  ConnectionPool pool = ConnectionPool.getInstance();
  Connection connection = pool.getConnection();
  PreparedStatement ps = null;
  ResultSet rs = null;

  String query = "SELECT COUNT(message_id) FROM (SELECT *
FROM message WHERE recipient = ?) AS message INNER JOIN user
ON sender = user_id " + filter;

  try {
   ps = connection.prepareStatement(query);

   ps.setLong(1, user.getID());

   rs = ps.executeQuery();

   if (rs.next()) {
    return rs.getInt(1);
   }
   else {
    return 0;
   }
  }
  catch (SQLException e) {
   e.printStackTrace();

   return 0;
  }
  finally {
   DBUtil.closeResultSet(rs);
        DBUtil.closePreparedStatement(ps);
        pool.freeConnection(connection);
  }
 }

 public static int countFilterModerator(String filter, User user) {
  ConnectionPool pool = ConnectionPool.getInstance();
  Connection connection = pool.getConnection();
  PreparedStatement ps = null;
  ResultSet rs = null;

  String query = "SELECT COUNT(message_id) FROM (SELECT *
FROM message LEFT JOIN digital_material ON message.material_id
= digital_material.pid WHERE recipient = ? OR community_id = ?) AS
message INNER JOIN user ON sender = user_id " + filter;

  try {
   ps = connection.prepareStatement(query);

   ps.setLong(1, user.getID());
   ps.setLong(2, user.getCommunity().getID());

   rs = ps.executeQuery();

   if (rs.next()) {
    return rs.getInt(1);
   }
   else {
    return 0;
   }
  }
  catch (SQLException e) {
   e.printStackTrace();

   return 0;
  }
  finally {
   DBUtil.closeResultSet(rs);
        DBUtil.closePreparedStatement(ps);
        pool.freeConnection(connection);
  }
 }

 public static int countFilterAdmin(String filter, User user) {
  ConnectionPool pool = ConnectionPool.getInstance();
  Connection connection = pool.getConnection();
  PreparedStatement ps = null;
  ResultSet rs = null;

  String query = "SELECT COUNT(message_id) FROM (SELECT *
FROM message WHERE recipient = ? OR type = 'flag' OR type =
```

```
'flag_resolved' OR type = 'support' OR type = 'support_resolved' OR
type = 'annotation') AS message INNER JOIN user ON sender =
user_id " + filter;

    try {
      ps = connection.prepareStatement(query);

      ps.setLong(1, user.getID());

      rs = ps.executeQuery();

      if (rs.next()) {
        return rs.getInt(1);
      }
      else {
        return 0;
      }
    }
    catch (SQLException e) {
      e.printStackTrace();

      return 0;
    }
    finally {
      DBUtil.closeResultSet(rs);
          DBUtil.closePreparedStatement(ps);
          pool.freeConnection(connection);
    }
  }

  public static int countMineUnread(User user) {
    ConnectionPool pool = ConnectionPool.getInstance();
    Connection connection = pool.getConnection();
    PreparedStatement ps = null;
    ResultSet rs = null;

    String query = "SELECT COUNT(message_id) FROM message
WHERE recipient = ? AND `read` = 'n'";

    try {
      ps = connection.prepareStatement(query);

      ps.setLong(1, user.getID());

      rs = ps.executeQuery();

      if (rs.next()) {
        return rs.getInt(1);
      }
      else {
        return 0;
      }
    }
    catch (SQLException e) {
      e.printStackTrace();

      return 0;
    }
    finally {
      DBUtil.closeResultSet(rs);
          DBUtil.closePreparedStatement(ps);
          pool.freeConnection(connection);
    }
  }

  public static int countMineUnreadModerator(User user) {
    ConnectionPool pool = ConnectionPool.getInstance();
    Connection connection = pool.getConnection();
    PreparedStatement ps = null;
    ResultSet rs = null;

    String query = "SELECT COUNT(message_id) FROM message
LEFT JOIN digital_material ON message.material_id =
digital_material.pid WHERE (recipient = ? AND `read` = 'n') OR
(community_id = ? AND `read` = 'n')";

    try {
      ps = connection.prepareStatement(query);

      ps.setLong(1, user.getID());
      ps.setLong(2, user.getCommunity().getID());

      rs = ps.executeQuery();

      if (rs.next()) {
        return rs.getInt(1);
      }
      else {
        return 0;
      }
    }
    catch (SQLException e) {
      e.printStackTrace();

      return 0;
    }
    finally {
      DBUtil.closeResultSet(rs);
          DBUtil.closePreparedStatement(ps);
          pool.freeConnection(connection);
    }
  }

  public static int countMineUnreadAdmin(User user) {
    ConnectionPool pool = ConnectionPool.getInstance();
    Connection connection = pool.getConnection();
    PreparedStatement ps = null;
    ResultSet rs = null;

    String query = "SELECT COUNT(message_id) FROM message
WHERE (recipient = ? AND `read` = 'n') OR (type = 'flag' AND `read` =
'n') OR (type = 'flag_resolved' AND `read` = 'n') OR (type = 'support'
AND `read` = 'n') OR (type = 'support_resolved' AND `read` = 'n') OR
(type = 'annotation' AND `read` = 'n')";

    try {
      ps = connection.prepareStatement(query);

      ps.setLong(1, user.getID());

      rs = ps.executeQuery();

      if (rs.next()) {
        return rs.getInt(1);
```

```
    }
    else {
      return 0;
    }
  }
  catch (SQLException e) {
    e.printStackTrace();

    return 0;
  }
  finally {
    DBUtil.closeResultSet(rs);
        DBUtil.closePreparedStatement(ps);
        pool.freeConnection(connection);
  }
 }

 public static Message select(long message_id) {
   ConnectionPool pool = ConnectionPool.getInstance();
   Connection connection = pool.getConnection();
   PreparedStatement ps = null;
   ResultSet rs = null;

   String query = "SELECT * FROM message WHERE message_id =
?";

   try {
     ps = connection.prepareStatement(query);

     ps.setLong(1, message_id);

     rs = ps.executeQuery();

     if (rs.next()) {
       Message message = new Message();

       message.setID(rs.getLong("message_id"));
       message.setRead(rs.getString("read"));
       message.setSubject(rs.getString("subject"));
       message.setContent(rs.getString("content"));
       message.setSender(UserDB.select(rs.getLong("sender")));
       message.setRecipient(UserDB.select(rs.getLong("recipient")));
       message.setType(rs.getString("type"));

message.setMessageTimestamp(rs.getTimestamp("message_timesta
mp"));

       return message;
     }
     else {
       return null;
     }
   }
   catch (SQLException e) {
     e.printStackTrace();

     return null;
   }
   finally {
     DBUtil.closeResultSet(rs);
     DBUtil.closePreparedStatement(ps);
     pool.freeConnection(connection);
   }
 }

 public static Message select(long message_id, User user) {
   ConnectionPool pool = ConnectionPool.getInstance();
   Connection connection = pool.getConnection();
   PreparedStatement ps = null;
   ResultSet rs = null;

   String query = "SELECT * FROM message WHERE recipient = ?
AND message_id = ?";

   try {
     ps = connection.prepareStatement(query);

     ps.setLong(1, user.getID());
     ps.setLong(2, message_id);

     rs = ps.executeQuery();

     if (rs.next()) {
       Message message = new Message();

       message.setID(rs.getLong("message_id"));
       message.setRead(rs.getString("read"));
       message.setSubject(rs.getString("subject"));
       message.setContent(rs.getString("content"));
       message.setSender(UserDB.select(rs.getLong("sender")));
       message.setRecipient(UserDB.select(rs.getLong("recipient")));
       message.setType(rs.getString("type"));

message.setMessageTimestamp(rs.getTimestamp("message_timesta
mp"));

       return message;
     }
     else {
       return null;
     }
   }
   catch (SQLException e) {
     e.printStackTrace();

     return null;
   }
   finally {
     DBUtil.closeResultSet(rs);
     DBUtil.closePreparedStatement(ps);
     pool.freeConnection(connection);
   }
 }

 public static Message selectModerator(long message_id, User user) {
   ConnectionPool pool = ConnectionPool.getInstance();
   Connection connection = pool.getConnection();
   PreparedStatement ps = null;
   ResultSet rs = null;

   String query = "SELECT * FROM message LEFT JOIN
digital_material ON message.material_id = digital_material.pid WHERE
(recipient = ? AND message_id = ?) OR (community_id = ? AND
message_id = ?)";
```

```
    try {
      ps = connection.prepareStatement(query);

      ps.setLong(1, user.getID());
      ps.setLong(2, message_id);
      ps.setLong(3, user.getCommunity().getID());
      ps.setLong(4, message_id);

      rs = ps.executeQuery();

      if (rs.next()) {
        Message message = new Message();

        message.setID(rs.getLong("message_id"));
        message.setRead(rs.getString("read"));
        message.setSubject(rs.getString("subject"));
        message.setContent(rs.getString("content"));
        message.setSender(UserDB.select(rs.getLong("sender")));
        message.setRecipient(UserDB.select(rs.getLong("recipient")));
        message.setType(rs.getString("type"));

message.setMessageTimestamp(rs.getTimestamp("message_timesta
mp"));

        return message;
      }
      else {
        return null;
      }
    }
    catch (SQLException e) {
      e.printStackTrace();

      return null;
    }
    finally {
      DBUtil.closeResultSet(rs);
      DBUtil.closePreparedStatement(ps);
      pool.freeConnection(connection);
    }
  }

  public static Message selectAdmin(long message_id, User user) {
    ConnectionPool pool = ConnectionPool.getInstance();
    Connection connection = pool.getConnection();
    PreparedStatement ps = null;
    ResultSet rs = null;

    String query = "SELECT * FROM message WHERE (recipient = ?
AND message_id = ?) OR (type = 'flag' AND message_id = ?) OR (type
= 'flag_resolved' AND message_id = ?) OR (type = 'support' AND
message_id = ?) OR (type = 'support_resolved' AND message_id = ?)
OR (type = 'annotation' AND message_id = ?)";

    try {
      ps = connection.prepareStatement(query);

      ps.setLong(1, user.getID());
      ps.setLong(2, message_id);
      ps.setLong(3, message_id);
      ps.setLong(4, message_id);
      ps.setLong(5, message_id);
      ps.setLong(6, message_id);
      ps.setLong(7, message_id);

      rs = ps.executeQuery();

      if (rs.next()) {
        Message message = new Message();

        message.setID(rs.getLong("message_id"));
        message.setRead(rs.getString("read"));
        message.setSubject(rs.getString("subject"));
        message.setContent(rs.getString("content"));
        message.setSender(UserDB.select(rs.getLong("sender")));
        message.setRecipient(UserDB.select(rs.getLong("recipient")));
        message.setType(rs.getString("type"));

message.setMessageTimestamp(rs.getTimestamp("message_timesta
mp"));

        return message;
      }
      else {
        return null;
      }
    }
    catch (SQLException e) {
      e.printStackTrace();

      return null;
    }
    finally {
      DBUtil.closeResultSet(rs);
      DBUtil.closePreparedStatement(ps);
      pool.freeConnection(connection);
    }
  }

  public static ArrayList<Message> selectMine(String filter, String order,
String limit, User user) {
    ConnectionPool pool = ConnectionPool.getInstance();
    Connection connection = pool.getConnection();
    PreparedStatement ps = null;
    ResultSet rs = null;

    String query = "SELECT message_id, `read`, subject, content,
sender, first_name, last_name, recipient, type, " +
          "message_timestamp FROM (SELECT * FROM message
WHERE recipient = ?) AS message INNER JOIN user ON sender =
user_id " +
          filter + " " + order + " " + limit;

    try {
      ps = connection.prepareStatement(query);

      ps.setLong(1, user.getID());

      rs = ps.executeQuery();
      ArrayList<Message> message_list = new ArrayList<Message>();

      while (rs.next()) {
        Message message = new Message();
```

```
      message.setID(rs.getLong("message_id"));
      message.setRead(rs.getString("read"));
      message.setSubject(rs.getString("subject"));
      message.setContent(rs.getString("content"));
      message.setSender(UserDB.select(rs.getLong("sender")));
      message.setRecipient(UserDB.select(rs.getLong("recipient")));
      message.setType(rs.getString("type"));

message.setMessageTimestamp(rs.getTimestamp("message_timesta
mp"));

      message_list.add(message);
    }

    return message_list;
  }
  catch (SQLException e) {
    e.printStackTrace();

    return null;
  }
  finally {
    DBUtil.closeResultSet(rs);
    DBUtil.closePreparedStatement(ps);
    pool.freeConnection(connection);
  }
 }

 public static ArrayList<Message> selectMineModerator(String filter,
String order, String limit, User user) {
   ConnectionPool pool = ConnectionPool.getInstance();
   Connection connection = pool.getConnection();
   PreparedStatement ps = null;
   ResultSet rs = null;

   String query = "SELECT message_id, `read`, subject, content,
sender, first_name, last_name, recipient, type, " +
          "message_timestamp FROM (SELECT * FROM message
LEFT JOIN digital_material ON message.material_id =
digital_material.pid WHERE recipient = ? OR community_id = ?) AS
message INNER JOIN user ON sender = user_id " +
          filter + " " + order + " " + limit;

   try {
    ps = connection.prepareStatement(query);

    ps.setLong(1, user.getID());
    ps.setLong(2, user.getCommunity().getID());

    rs = ps.executeQuery();
    ArrayList<Message> message_list = new ArrayList<Message>();

    while (rs.next()) {
     Message message = new Message();

     message.setID(rs.getLong("message_id"));
     message.setRead(rs.getString("read"));
     message.setSubject(rs.getString("subject"));
     message.setContent(rs.getString("content"));
     message.setSender(UserDB.select(rs.getLong("sender")));
     message.setRecipient(UserDB.select(rs.getLong("recipient")));
     message.setType(rs.getString("type"));

message.setMessageTimestamp(rs.getTimestamp("message_timesta
mp"));

     message_list.add(message);
    }

    return message_list;
   }
   catch (SQLException e) {
    e.printStackTrace();

    return null;
   }
   finally {
    DBUtil.closeResultSet(rs);
    DBUtil.closePreparedStatement(ps);
    pool.freeConnection(connection);
   }
  }

  public static ArrayList<Message> selectMineAdmin(String filter, String
order, String limit, User user) {
   ConnectionPool pool = ConnectionPool.getInstance();
   Connection connection = pool.getConnection();
   PreparedStatement ps = null;
   ResultSet rs = null;

   String query = "SELECT message_id, `read`, subject, content,
sender, first_name, last_name, recipient, type, " +
          "message_timestamp FROM (SELECT * FROM message
WHERE recipient = ? OR type = 'flag' OR type = 'flag_resolved' OR
type = 'support' OR type = 'support_resolved' OR type = 'annotation')
AS message INNER JOIN user ON sender = user_id " +
          filter + " " + order + " " + limit;

   try {
    ps = connection.prepareStatement(query);

    ps.setLong(1, user.getID());

    rs = ps.executeQuery();
    ArrayList<Message> message_list = new ArrayList<Message>();

    while (rs.next()) {
     Message message = new Message();

     message.setID(rs.getLong("message_id"));
     message.setRead(rs.getString("read"));
     message.setSubject(rs.getString("subject"));
     message.setContent(rs.getString("content"));
     message.setSender(UserDB.select(rs.getLong("sender")));
     message.setRecipient(UserDB.select(rs.getLong("recipient")));
     message.setType(rs.getString("type"));

message.setMessageTimestamp(rs.getTimestamp("message_timesta
mp"));

     message_list.add(message);
    }
```

```
    return message_list;
   }
   catch (SQLException e) {
    e.printStackTrace();

    return null;
   }
   finally {
    DBUtil.closeResultSet(rs);
    DBUtil.closePreparedStatement(ps);
    pool.freeConnection(connection);
   }
  }

  public static int insert(Message message) {
   ConnectionPool pool = ConnectionPool.getInstance();
      Connection connection = pool.getConnection();
      PreparedStatement ps = null;
      ResultSet rs = null;

      String query = "INSERT INTO message (`read`, subject, content,
sender, recipient, type, message_timestamp, material_id) VALUES ('n',
?, ?, ?, ?, ?, NOW(), ?)";

      try {
         ps = connection.prepareStatement(query);

         ps.setString(1, message.getSubject());
         ps.setString(2, message.getContent());
         ps.setLong(3,
UserDB.selectID(message.getSender().getEmailAddress()));
         ps.setLong(4,
UserDB.selectID(message.getRecipient().getEmailAddress()));
         ps.setString(5, message.getType());
         ps.setString(6, message.getMaterialID());

         return ps.executeUpdate();
      }
      catch (SQLException e) {
         e.printStackTrace();

         return 0;
      }
      finally {
         DBUtil.closeResultSet(rs);
         DBUtil.closePreparedStatement(ps);
         pool.freeConnection(connection);
      }
  }

  public static int read(long message_id, User user) {
   ConnectionPool pool = ConnectionPool.getInstance();
    Connection connection = pool.getConnection();
    PreparedStatement ps = null;
    ResultSet rs = null;

    String query = "UPDATE message SET " +
                "`read` = 'y' " +
                "WHERE recipient = ? AND message_id = ?";

    try {
     ps = connection.prepareStatement(query);

         ps.setLong(1, user.getID());
         ps.setLong(2, message_id);

        return ps.executeUpdate();
    }
    catch (SQLException e) {
     e.printStackTrace();

        return 0;
    }
    finally {
      DBUtil.closeResultSet(rs);
        DBUtil.closePreparedStatement(ps);
        pool.freeConnection(connection);
    }
  }

  public static int readAdmin(long message_id) {
   ConnectionPool pool = ConnectionPool.getInstance();
    Connection connection = pool.getConnection();
    PreparedStatement ps = null;
    ResultSet rs = null;

    String query = "UPDATE message SET " +
                "`read` = 'y' " +
                "WHERE message_id = ?";

    try {
     ps = connection.prepareStatement(query);

         ps.setLong(1, message_id);

        return ps.executeUpdate();
    }
    catch (SQLException e) {
     e.printStackTrace();

        return 0;
    }
    finally {
      DBUtil.closeResultSet(rs);
        DBUtil.closePreparedStatement(ps);
        pool.freeConnection(connection);
    }
  }

  public static int resolve(String action, long message_id) {
   ConnectionPool pool = ConnectionPool.getInstance();
    Connection connection = pool.getConnection();
    PreparedStatement ps = null;
    ResultSet rs = null;

    String query = "UPDATE message SET type = ? WHERE
message_id = ?";

    try {
     ps = connection.prepareStatement(query);

     ps.setString(1, action);
        ps.setLong(2, message_id);
```

```
            return ps.executeUpdate();
        }
        catch (SQLException e) {
          e.printStackTrace();

            return 0;
        }
        finally {
          DBUtil.closeResultSet(rs);
            DBUtil.closePreparedStatement(ps);
            pool.freeConnection(connection);
        }
    }

    public static int delete(long message_id, User user) {
      ConnectionPool pool = ConnectionPool.getInstance();
        Connection connection = pool.getConnection();
        PreparedStatement ps = null;
        ResultSet rs = null;

        String query = "DELETE FROM message WHERE recipient = ?
AND message_id = ?";

        try {
          ps = connection.prepareStatement(query);

            ps.setLong(1, user.getID());
            ps.setLong(2, message_id);

            return ps.executeUpdate();
        }
        catch (SQLException e) {
          e.printStackTrace();

            return 0;
        }
        finally {
          DBUtil.closeResultSet(rs);
            DBUtil.closePreparedStatement(ps);
            pool.freeConnection(connection);
        }
    }

    public static int deleteModerator(long message_id, User user) {
      ConnectionPool pool = ConnectionPool.getInstance();
        Connection connection = pool.getConnection();
        PreparedStatement ps = null;
        ResultSet rs = null;

        String query = "DELETE message.* FROM message LEFT JOIN
digital_material ON message.material_id = digital_material.pid WHERE
(recipient = ? AND message_id = ?) OR (community_id = ? AND
message_id = ?)";

        try {
          ps = connection.prepareStatement(query);

            ps.setLong(1, user.getID());
            ps.setLong(2, message_id);
            ps.setLong(3, user.getCommunity().getID());
            ps.setLong(4, message_id);
            return ps.executeUpdate();
        }
        catch (SQLException e) {
          e.printStackTrace();

            return 0;
        }
        finally {
          DBUtil.closeResultSet(rs);
            DBUtil.closePreparedStatement(ps);
            pool.freeConnection(connection);
        }
    }

    public static int deleteAdmin(long message_id, User user) {
      ConnectionPool pool = ConnectionPool.getInstance();
        Connection connection = pool.getConnection();
        PreparedStatement ps = null;
        ResultSet rs = null;

        String query = "DELETE FROM message WHERE (recipient = ?
AND message_id = ?) OR (type = 'flag' AND message_id = ?) OR (type
= 'flag_resolved' AND message_id = ?) OR (type = 'support' AND
message_id = ?) OR (type = 'support_resolved' AND message_id = ?)
OR (type = 'annotation' AND message_id = ?)";

        try {
          ps = connection.prepareStatement(query);

            ps.setLong(1, user.getID());
            ps.setLong(2, message_id);
            ps.setLong(3, message_id);
            ps.setLong(4, message_id);
            ps.setLong(5, message_id);
            ps.setLong(6, message_id);
            ps.setLong(7, message_id);

            return ps.executeUpdate();
        }
        catch (SQLException e) {
          e.printStackTrace();

            return 0;
        }
        finally {
          DBUtil.closeResultSet(rs);
            DBUtil.closePreparedStatement(ps);
            pool.freeConnection(connection);
        }
    }
}
```

NewsDB.java

```
package data;

import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.ArrayList;
```

```
import domain.News;

public class NewsDB {
  public static int count() {
    ConnectionPool pool = ConnectionPool.getInstance();
    Connection connection = pool.getConnection();
    PreparedStatement ps = null;
    ResultSet rs = null;

    String query = "SELECT COUNT(news_id) FROM news";

    try {
      ps = connection.prepareStatement(query);
      rs = ps.executeQuery();

      if (rs.next()) {
        return rs.getInt(1);
      }
      else {
        return 0;
      }
    }
    catch (SQLException e) {
      e.printStackTrace();

      return 0;
    }
    finally {
      DBUtil.closeResultSet(rs);
          DBUtil.closePreparedStatement(ps);
          pool.freeConnection(connection);
    }
  }

  public static int countFilter(String filter) {
    ConnectionPool pool = ConnectionPool.getInstance();
    Connection connection = pool.getConnection();
    PreparedStatement ps = null;
    ResultSet rs = null;

    String query = "SELECT COUNT(news_id) FROM news " + filter;

    try {
      ps = connection.prepareStatement(query);
      rs = ps.executeQuery();

      if (rs.next()) {
        return rs.getInt(1);
      }
      else {
        return 0;
      }
    }
    catch (SQLException e) {
      e.printStackTrace();

      return 0;
    }
    finally {
      DBUtil.closeResultSet(rs);
          DBUtil.closePreparedStatement(ps);
          pool.freeConnection(connection);
    }
  }

  public static long selectID(String reference) {
    ConnectionPool pool = ConnectionPool.getInstance();
    Connection connection = pool.getConnection();
    PreparedStatement ps = null;
    ResultSet rs = null;

    String query = "SELECT news_id FROM news WHERE reference =
?";

    try {
      ps = connection.prepareStatement(query);
      ps.setString(1, reference);
      rs = ps.executeQuery();

      if (rs.next()) {
        return rs.getLong("news_id");
      }
      else {
        return -1;
      }
    }
    catch (SQLException e) {
      e.printStackTrace();

      return -1;
    }
    finally {
      DBUtil.closeResultSet(rs);
          DBUtil.closePreparedStatement(ps);
          pool.freeConnection(connection);
    }
  }

  public static String selectTitle(String id) {
    ConnectionPool pool = ConnectionPool.getInstance();
    Connection connection = pool.getConnection();
    PreparedStatement ps = null;
    ResultSet rs = null;

    String query = "SELECT title FROM news WHERE news_id = ?";

    try {
      ps = connection.prepareStatement(query);
      ps.setString(1, id);
      rs = ps.executeQuery();

      if (rs.next()) {
        return rs.getString("title");
      }
      else {
        return null;
      }
    }
    catch (SQLException e) {
      e.printStackTrace();

      return null;
    }
    finally {
```

```
    DBUtil.closeResultSet(rs);
    DBUtil.closePreparedStatement(ps);
    pool.freeConnection(connection);
  }
 }

 public static boolean newsExists(long news_id) {
  ConnectionPool pool = ConnectionPool.getInstance();
  Connection connection = pool.getConnection();
  PreparedStatement ps = null;
  ResultSet rs = null;

  String query = "SELECT news_id FROM news WHERE news_id =
?";

  try {
    ps = connection.prepareStatement(query);
    ps.setLong(1, news_id);
    rs = ps.executeQuery();

    return rs.next();
  }
  catch (SQLException e) {
    e.printStackTrace();

    return false;
  }
  finally {
    DBUtil.closeResultSet(rs);
    DBUtil.closePreparedStatement(ps);
    pool.freeConnection(connection);
  }
 }

 public static boolean referenceExists(String reference) {
  ConnectionPool pool = ConnectionPool.getInstance();
  Connection connection = pool.getConnection();
  PreparedStatement ps = null;
  ResultSet rs = null;

  String query = "SELECT news_id FROM news WHERE reference =
?";

  try {
    ps = connection.prepareStatement(query);
    ps.setString(1, reference);
    rs = ps.executeQuery();

    return rs.next();
  }
  catch (SQLException e) {
    e.printStackTrace();

    return false;
  }
  finally {
    DBUtil.closeResultSet(rs);
        DBUtil.closePreparedStatement(ps);
        pool.freeConnection(connection);
  }
 }

 public static News select(String reference) {
  ConnectionPool pool = ConnectionPool.getInstance();
  Connection connection = pool.getConnection();
  PreparedStatement ps = null;
  ResultSet rs = null;

  String query = "SELECT * FROM news WHERE reference = ?";

  try {
    ps = connection.prepareStatement(query);
    ps.setString(1, reference);
    rs = ps.executeQuery();

    if (rs.next()) {
      News news = new News();

      news.setTitle(rs.getString("title"));
      news.setReference(rs.getString("reference"));
      news.setContent(rs.getString("content"));
      news.setUser(UserDB.select(rs.getLong("author")));
      news.setNewsTimestamp(rs.getTimestamp("news_timestamp"));

      return news;
    }
    else {
      return null;
    }
  }
  catch (SQLException e) {
    e.printStackTrace();

    return null;
  }
  finally {
    DBUtil.closeResultSet(rs);
    DBUtil.closePreparedStatement(ps);
    pool.freeConnection(connection);
  }
 }

 public static News select(long news_id) {
  ConnectionPool pool = ConnectionPool.getInstance();
  Connection connection = pool.getConnection();
  PreparedStatement ps = null;
  ResultSet rs = null;

  String query = "SELECT * FROM news WHERE news_id = ?";

  try {
    ps = connection.prepareStatement(query);
    ps.setLong(1, news_id);
    rs = ps.executeQuery();

    if (rs.next()) {
      News news = new News();

      news.setTitle(rs.getString("title"));
      news.setReference(rs.getString("reference"));
      news.setContent(rs.getString("content"));
      news.setUser(UserDB.select(rs.getLong("author")));
      news.setNewsTimestamp(rs.getTimestamp("news_timestamp"));
```

```
      return news;
    }
    else {
      return null;
    }
   }
   catch (SQLException e) {
    e.printStackTrace();

    return null;
   }
   finally {
    DBUtil.closeResultSet(rs);
    DBUtil.closePreparedStatement(ps);
    pool.freeConnection(connection);
   }
  }

  public static ArrayList<News> select(String filter, String order, String
limit) {
   ConnectionPool pool = ConnectionPool.getInstance();
   Connection connection = pool.getConnection();
   PreparedStatement ps = null;
   ResultSet rs = null;

   String query = "SELECT * FROM news " + filter + " " + order + " " +
limit;

   try {
    ps = connection.prepareStatement(query);
    rs = ps.executeQuery();
    ArrayList<News> news_list = new ArrayList<News>();

    while (rs.next()) {
     News news = new News();

     news.setTitle(rs.getString("title"));
     news.setReference(rs.getString("reference"));
     news.setContent(rs.getString("content"));
     news.setUser(UserDB.select(rs.getLong("author")));
     news.setNewsTimestamp(rs.getTimestamp("news_timestamp"));

     news_list.add(news);
    }

    return news_list;
   }
   catch (SQLException e) {
    e.printStackTrace();

    return null;
   }
   finally {
    DBUtil.closeResultSet(rs);
    DBUtil.closePreparedStatement(ps);
    pool.freeConnection(connection);
   }
  }

  public static ArrayList<News> selectAll() {
   ConnectionPool pool = ConnectionPool.getInstance();
   Connection connection = pool.getConnection();
   PreparedStatement ps = null;
   ResultSet rs = null;

   String query = "SELECT * FROM news ORDER BY news_timestamp
DESC";

   try {
    ps = connection.prepareStatement(query);
    rs = ps.executeQuery();
    ArrayList<News> news_list = new ArrayList<News>();

    while (rs.next()) {
     News news = new News();

     news.setTitle(rs.getString("title"));
     news.setReference(rs.getString("reference"));
     news.setContent(rs.getString("content"));
     news.setUser(UserDB.select(rs.getLong("author")));
     news.setNewsTimestamp(rs.getTimestamp("news_timestamp"));

     news_list.add(news);
    }

    return news_list;
   }
   catch (SQLException e) {
    e.printStackTrace();

    return null;
   }
   finally {
    DBUtil.closeResultSet(rs);
    DBUtil.closePreparedStatement(ps);
    pool.freeConnection(connection);
   }
  }

  public static ArrayList<News> selectRecent() {
   ConnectionPool pool = ConnectionPool.getInstance();
   Connection connection = pool.getConnection();
   PreparedStatement ps = null;
   ResultSet rs = null;

   String query = "SELECT * FROM news ORDER BY news_timestamp
DESC LIMIT 5";

   try {
    ps = connection.prepareStatement(query);
    rs = ps.executeQuery();
    ArrayList<News> news_list = new ArrayList<News>();

    while (rs.next()) {
     News news = new News();

     news.setTitle(rs.getString("title"));
     news.setReference(rs.getString("reference"));
     news.setContent(rs.getString("content"));
     news.setUser(UserDB.select(rs.getString("author")));
     news.setNewsTimestamp(rs.getTimestamp("news_timestamp"));

     news_list.add(news);
    }
```

```
    return news_list;
   }
   catch (SQLException e) {
    e.printStackTrace();

    return null;
   }
   finally {
    DBUtil.closeResultSet(rs);
    DBUtil.closePreparedStatement(ps);
    pool.freeConnection(connection);
   }
  }

  public static int insert(News news) {
    ConnectionPool pool = ConnectionPool.getInstance();
      Connection connection = pool.getConnection();
      PreparedStatement ps = null;
      ResultSet rs = null;

      String query = "INSERT INTO news " +
              "(title, reference, content, author, news_timestamp)" +
              " VALUES (?, ?, ?, ?, NOW())";

      try {
        ps = connection.prepareStatement(query);

        ps.setString(1, news.getTitle());
        ps.setString(2, news.getReference());
        ps.setString(3, news.getContent());
        ps.setLong(4,
UserDB.selectID(news.getUser().getEmailAddress()));

        return ps.executeUpdate();
      }
      catch (SQLException e) {
        e.printStackTrace();

        return 0;
      }
      finally {
        DBUtil.closeResultSet(rs);
        DBUtil.closePreparedStatement(ps);
        pool.freeConnection(connection);
      }
  }

  public static int delete(String reference) {
    ConnectionPool pool = ConnectionPool.getInstance();
     Connection connection = pool.getConnection();
     PreparedStatement ps = null;
     ResultSet rs = null;

     String query = "DELETE FROM news WHERE reference = ?";

     try {
       ps = connection.prepareStatement(query);
         ps.setString(1, reference);

         return ps.executeUpdate();
     }
     catch (SQLException e) {
       e.printStackTrace();

         return 0;
     }
     finally {
       DBUtil.closeResultSet(rs);
         DBUtil.closePreparedStatement(ps);
         pool.freeConnection(connection);
     }
  }

  public ArrayList<News> getRecentNews() {
    return selectRecent();
  }
}
```

OperationDB.java

```
package data;

import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.ArrayList;

import domain.Operation;

public class OperationDB {
  public static int count() {
    ConnectionPool pool = ConnectionPool.getInstance();
    Connection connection = pool.getConnection();
    PreparedStatement ps = null;
    ResultSet rs = null;

    String query = "SELECT COUNT(operation_id) FROM operation";

    try {
      ps = connection.prepareStatement(query);
      rs = ps.executeQuery();

      if (rs.next()) {
        return rs.getInt(1);
      }
      else {
        return 0;
      }
    }
    catch (SQLException e) {
      e.printStackTrace();

      return 0;
    }
    finally {
      DBUtil.closeResultSet(rs);
          DBUtil.closePreparedStatement(ps);
          pool.freeConnection(connection);
    }
  }

  public static int countFilter(String filter) {
```

```
    ConnectionPool pool = ConnectionPool.getInstance();
    Connection connection = pool.getConnection();
    PreparedStatement ps = null;
    ResultSet rs = null;

    String query = "SELECT COUNT(operation_id) FROM operation " +
filter;

    try {
      ps = connection.prepareStatement(query);
      rs = ps.executeQuery();

      if (rs.next()) {
        return rs.getInt(1);
      }
      else {
        return 0;
      }
    }
    catch (SQLException e) {
      e.printStackTrace();

      return 0;
    }
    finally {
      DBUtil.closeResultSet(rs);
          DBUtil.closePreparedStatement(ps);
          pool.freeConnection(connection);
    }
  }

  public static ArrayList<Operation> select(String filter, String order,
String limit) {
    ConnectionPool pool = ConnectionPool.getInstance();
    Connection connection = pool.getConnection();
    PreparedStatement ps = null;
    ResultSet rs = null;

    String query = "SELECT * FROM operation " + filter + " " + order + " "
+ limit;

    try {
      ps = connection.prepareStatement(query);
      rs = ps.executeQuery();
      ArrayList<Operation> operation_list = new ArrayList<Operation>();

      while (rs.next()) {
        Operation operation = new Operation();

        operation.setDescription(rs.getString("description"));

operation.setOperationTimestamp(rs.getTimestamp("operation_timesta
mp"));

        operation_list.add(operation);
      }

      return operation_list;
    }
    catch (SQLException e) {
      e.printStackTrace();

      return null;
    }
    finally {
      DBUtil.closeResultSet(rs);
      DBUtil.closePreparedStatement(ps);
      pool.freeConnection(connection);
    }
  }

  public static int insert(String description) {
    ConnectionPool pool = ConnectionPool.getInstance();
        Connection connection = pool.getConnection();
        PreparedStatement ps = null;
        ResultSet rs = null;

        String query = "INSERT INTO operation (description,
operation_timestamp) VALUES (?, NOW())";

        try {
          ps = connection.prepareStatement(query);

          ps.setString(1, description);

          return ps.executeUpdate();
        }
        catch (SQLException e) {
          e.printStackTrace();

          return 0;
        }
        finally {
          DBUtil.closeResultSet(rs);
          DBUtil.closePreparedStatement(ps);
          pool.freeConnection(connection);
        }
  }
}
```

SQLUtil.java

```
package data;

import java.sql.*;

import javax.servlet.http.HttpServletRequest;

public class SQLUtil {
  public static String getFilter(HttpServletRequest request, String[] cols)
{
    String filter = "";

    // Filtering
    if (!request.getParameter("sSearch").equals("")) {
      filter = "WHERE (";

      for (int i = 0; i < cols.length; i++) {
        if (request.getParameter("bSearchable_" + i).equals("true")) {
          filter = filter + cols[i] + " LIKE '%" +
request.getParameter("sSearch") + "%' OR ";
        }
      }
```

```
      filter = filter.substring(0, filter.length() - 3);
      filter = filter + ")";
    }

    // Individual column filtering
    for (int i = 0; i < cols.length; i++) {
      if (request.getParameter("bSearchable_" + i).equals("true") &&
!request.getParameter("sSearch_" + i).equals("")) {
        if (filter.equals("")) {
          filter = "WHERE ";
        }
        else {
          filter = filter + " AND ";
        }
        filter = filter + cols[i] + " LIKE '%" +
request.getParameter("sSearch_" + i) + "%' ";
      }
    }

    return filter;
  }

  public static String getOrder(HttpServletRequest request, String[] cols)
{
    String order = "";

    if (request.getParameter("iSortCol_0") != null) {
      order = "ORDER BY  ";
      int sorting_cols =
Integer.parseInt(request.getParameter("iSortingCols"));

      for (int i = 0; i < sorting_cols; i++) {
        String sort_col = request.getParameter("iSortCol_" + i);
        String sortable = request.getParameter("bSortable_" + sort_col);

        if (sortable.equals("true")) {
          order = order + cols[Integer.parseInt(sort_col)] + " " +
request.getParameter("sSortDir_" + i) + ", ";
        }
      }

      order = order.substring(0, order.length() - 2);

      if (order.equals("ORDER BY")) {
        order = "";
      }
    }

    return order;
  }

  public static String getLimit(HttpServletRequest request) {
    String limit = "";

    if (request.getParameter("iDisplayStart") != null &&
request.getParameter("iDisplayLength") != "-1") {
      limit = "LIMIT " + request.getParameter("iDisplayStart") + ", " +
request.getParameter("iDisplayLength");
    }

    return limit;
  }

    public static String getHtmlRows(ResultSet results) throws
SQLException {
        StringBuffer htmlRows = new StringBuffer();
        ResultSetMetaData metaData = results.getMetaData();
        int columnCount = metaData.getColumnCount();

        htmlRows.append("<tr>");
        for(int i = 1; i <= columnCount; i++) {
            htmlRows.append("<td><b>" + metaData.getColumnName(i) +
"</td>");
        }
        htmlRows.append("</tr>");

        while(results.next()) {
            htmlRows.append("<tr>");
            for(int i = 1; i <= columnCount; i++) {
                htmlRows.append("<td>" + results.getString(i) + "</td>");
            }
        }
        htmlRows.append("</tr>");

        return htmlRows.toString();
    }

    public static String encode(String s) {
        if(s == null) return s;
        StringBuffer sb = new StringBuffer(s);
        for(int i = 0; i < sb.length(); i++) {
            char ch = sb.charAt(i);
            if(ch == 39) {
              // 39 is the ASCII code for an apostrophe
                sb.insert(i++, "'");
            }
        }
        return sb.toString();
    }
}
```

TagDB.java

```
package data;

import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.ArrayList;

public class TagDB {
  public static boolean tagExists(String tag) {
    ConnectionPool pool = ConnectionPool.getInstance();
    Connection connection = pool.getConnection();
    PreparedStatement ps = null;
    ResultSet rs = null;

    String query = "SELECT name FROM tags WHERE name = ?";

    try {
      ps = connection.prepareStatement(query);
      ps.setString(1, tag);
      rs = ps.executeQuery();
```

```
    return rs.next();
   }
   catch (SQLException e) {
    e.printStackTrace();

    return false;
   }
   finally {
    DBUtil.closeResultSet(rs);
        DBUtil.closePreparedStatement(ps);
        pool.freeConnection(connection);
   }
  }

  public static String[] select(String input) {
   ConnectionPool pool = ConnectionPool.getInstance();
   Connection connection = pool.getConnection();
   PreparedStatement ps = null;
   ResultSet rs = null;

   input = "'" + input + "%' ";
   String query = "SELECT * FROM tags WHERE name LIKE " + input
+ "ORDER BY name ASC";

   try {
    ps = connection.prepareStatement(query);
    rs = ps.executeQuery();

    ArrayList<String> tag_list = new ArrayList<String>();

    while (rs.next()) {
     tag_list.add(rs.getString("name"));
    }

    String[] tags = new String[tag_list.size()];
    tags = tag_list.toArray(tags);

    return tags;
   }
   catch (SQLException e) {
    e.printStackTrace();

    return null;
   }
   finally {
    DBUtil.closeResultSet(rs);
    DBUtil.closePreparedStatement(ps);
    pool.freeConnection(connection);
   }
  }

  public static int insert(String tag) {
   ConnectionPool pool = ConnectionPool.getInstance();
      Connection connection = pool.getConnection();
      PreparedStatement ps = null;
      ResultSet rs = null;

      String query = "INSERT INTO tags (name) VALUES (?)";

      try {
       ps = connection.prepareStatement(query);
       ps.setString(1, tag);

       return ps.executeUpdate();
      }
      catch (SQLException e) {
       e.printStackTrace();

       return 0;
      }
      finally {
       DBUtil.closeResultSet(rs);
       DBUtil.closePreparedStatement(ps);
       pool.freeConnection(connection);
      }
   }
  }
```

UserDB.java

```
package data;

import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.ArrayList;
import java.util.LinkedHashMap;

import org.apache.commons.codec.digest.DigestUtils;

import domain.Community;
import domain.User;

public class UserDB {
 public static int count() {
   ConnectionPool pool = ConnectionPool.getInstance();
   Connection connection = pool.getConnection();
   PreparedStatement ps = null;
   ResultSet rs = null;

   String query = "SELECT COUNT(user_id) FROM user";

   try {
    ps = connection.prepareStatement(query);
    rs = ps.executeQuery();

    if (rs.next()) {
     return rs.getInt(1);
    }
    else {
     return 0;
    }
   }
   catch (SQLException e) {
    e.printStackTrace();

    return 0;
   }
   finally {
    DBUtil.closeResultSet(rs);
        DBUtil.closePreparedStatement(ps);
```

```
        pool.freeConnection(connection);
    }
}

public static int countFilter(String filter) {
    ConnectionPool pool = ConnectionPool.getInstance();
    Connection connection = pool.getConnection();
    PreparedStatement ps = null;
    ResultSet rs = null;

    String query = "SELECT COUNT(user_id) FROM user " + filter;

    try {
      ps = connection.prepareStatement(query);
      rs = ps.executeQuery();

      if (rs.next()) {
        return rs.getInt(1);
      }
      else {
        return 0;
      }
    }
    catch (SQLException e) {
      e.printStackTrace();

      return 0;
    }
    finally {
      DBUtil.closeResultSet(rs);
          DBUtil.closePreparedStatement(ps);
          pool.freeConnection(connection);
    }
}

public static boolean userExists(long user_id) {
    ConnectionPool pool = ConnectionPool.getInstance();
    Connection connection = pool.getConnection();
    PreparedStatement ps = null;
    ResultSet rs = null;

    String query = "SELECT user_id FROM user WHERE user_id = ?";

    try {
      ps = connection.prepareStatement(query);
      ps.setLong(1, user_id);
      rs = ps.executeQuery();

      return rs.next();
    }
    catch (SQLException e) {
      e.printStackTrace();

      return false;
    }
    finally {
      DBUtil.closeResultSet(rs);
          DBUtil.closePreparedStatement(ps);
          pool.freeConnection(connection);
    }
}

public static boolean userExists(String email_address) {
    ConnectionPool pool = ConnectionPool.getInstance();
    Connection connection = pool.getConnection();
    PreparedStatement ps = null;
    ResultSet rs = null;

    String query = "SELECT email_address FROM user WHERE
email_address = ?";

    try {
      ps = connection.prepareStatement(query);
      ps.setString(1, email_address);
      rs = ps.executeQuery();

      return rs.next();
    }
    catch (SQLException e) {
      e.printStackTrace();

      return false;
    }
    finally {
      DBUtil.closeResultSet(rs);
          DBUtil.closePreparedStatement(ps);
          pool.freeConnection(connection);
    }
}

public static long selectID(String email_address) {
    ConnectionPool pool = ConnectionPool.getInstance();
    Connection connection = pool.getConnection();
    PreparedStatement ps = null;
    ResultSet rs = null;

    String query = "SELECT user_id FROM user WHERE email_address
= ?";

    try {
      ps = connection.prepareStatement(query);
      ps.setString(1, email_address);
      rs = ps.executeQuery();

      if (rs.next()) {
        return rs.getLong("user_id");
      }
      else {
        return -1;
      }
    }
    catch (SQLException e) {
      e.printStackTrace();

      return -1;
    }
    finally {
      DBUtil.closeResultSet(rs);
      DBUtil.closePreparedStatement(ps);
      pool.freeConnection(connection);
    }
}

public static User select(long user_id) {
```

```
    ConnectionPool pool = ConnectionPool.getInstance();
    Connection connection = pool.getConnection();
    PreparedStatement ps = null;
    ResultSet rs = null;

    String query = "SELECT * FROM user WHERE user_id = ?";

    try {
      ps = connection.prepareStatement(query);
      ps.setLong(1, user_id);
      rs = ps.executeQuery();

      if (rs.next()) {
        User user = new User();

        user.setEmailAddress(rs.getString("email_address"));
        user.setPassword(rs.getString("password"));
        user.setFirstName(rs.getString("first_name"));
        user.setLastName(rs.getString("last_name"));
        user.setBirthday(rs.getDate("birthday"));
        user.setDescription(rs.getString("description"));
        user.setOccupation(rs.getString("occupation"));
        user.setOrganization(rs.getString("organization"));
        user.setContactNumber(rs.getString("contact_number"));
        user.setSystemRole(rs.getString("system_role"));

user.setCommunity(CommunityDB.select(rs.getLong("community_id")))
;
        user.setStudentNumber(rs.getString("student_number"));

user.setRegistrationTimestamp(rs.getTimestamp("registration_timesta
mp"));
        user.setExpiration(rs.getTimestamp("expiration"));
        user.setLastActivity(rs.getTimestamp("last_activity"));

        return user;
      }
      else {
        return null;
      }
    }
    catch (SQLException e) {
      e.printStackTrace();

      return null;
    }
    finally {
      DBUtil.closeResultSet(rs);
      DBUtil.closePreparedStatement(ps);
      pool.freeConnection(connection);
    }
  }

  public static User select(String email_address) {
    ConnectionPool pool = ConnectionPool.getInstance();
    Connection connection = pool.getConnection();
    PreparedStatement ps = null;
    ResultSet rs = null;

    String query = "SELECT * FROM user WHERE email_address = ?";

    try {
      ps = connection.prepareStatement(query);
      ps.setString(1, email_address);
      rs = ps.executeQuery();

      if (rs.next()) {
        User user = new User();

        user.setEmailAddress(rs.getString("email_address"));
        user.setPassword(rs.getString("password"));
        user.setFirstName(rs.getString("first_name"));
        user.setLastName(rs.getString("last_name"));
        user.setBirthday(rs.getDate("birthday"));
        user.setDescription(rs.getString("description"));
        user.setOccupation(rs.getString("occupation"));
        user.setOrganization(rs.getString("organization"));
        user.setContactNumber(rs.getString("contact_number"));
        user.setSystemRole(rs.getString("system_role"));

user.setCommunity(CommunityDB.select(rs.getLong("community_id")))
;
        user.setStudentNumber(rs.getString("student_number"));

user.setRegistrationTimestamp(rs.getTimestamp("registration_timesta
mp"));
        user.setExpiration(rs.getTimestamp("expiration"));
        user.setLastActivity(rs.getTimestamp("last_activity"));

        return user;
      }
      else {
        return null;
      }
    }
    catch (SQLException e) {
      e.printStackTrace();

      return null;
    }
    finally {
      DBUtil.closeResultSet(rs);
      DBUtil.closePreparedStatement(ps);
      pool.freeConnection(connection);
    }
  }

  public static LinkedHashMap<Long, User> select(String filter, String
order, String limit) {
    ConnectionPool pool = ConnectionPool.getInstance();
    Connection connection = pool.getConnection();
    PreparedStatement ps = null;
    ResultSet rs = null;

    String query = "SELECT * FROM user " + filter + " " + order + " " +
limit;

    try {
      ps = connection.prepareStatement(query);
      rs = ps.executeQuery();
      LinkedHashMap<Long, User> user_list = new
LinkedHashMap<Long, User>();

      while (rs.next()) {
```

```
        User user = new User();

        user.setEmailAddress(rs.getString("email_address"));
        user.setPassword(rs.getString("password"));
        user.setFirstName(rs.getString("first_name"));
        user.setLastName(rs.getString("last_name"));
        user.setBirthday(rs.getDate("birthday"));
        user.setDescription(rs.getString("description"));
        user.setOccupation(rs.getString("occupation"));
        user.setOrganization(rs.getString("organization"));
        user.setContactNumber(rs.getString("contact_number"));
        user.setSystemRole(rs.getString("system_role"));

user.setCommunity(CommunityDB.select(rs.getLong("community_id")))
;
        user.setStudentNumber(rs.getString("student_number"));

user.setRegistrationTimestamp(rs.getTimestamp("registration_timesta
mp"));
        user.setExpiration(rs.getTimestamp("expiration"));
        user.setLastActivity(rs.getTimestamp("last_activity"));

        user_list.put(rs.getLong("user_id"), user);
      }

      return user_list;
    }
    catch (SQLException e) {
      e.printStackTrace();

      return null;
    }
    finally {
      DBUtil.closeResultSet(rs);
      DBUtil.closePreparedStatement(ps);
      pool.freeConnection(connection);
    }
  }

  public static ArrayList<User> selectAdministrators() {
    ConnectionPool pool = ConnectionPool.getInstance();
    Connection connection = pool.getConnection();
    PreparedStatement ps = null;
    ResultSet rs = null;

    String query = "SELECT * FROM user WHERE system_role =
'upmir_system_administrator'";

    try {
      ps = connection.prepareStatement(query);
      rs = ps.executeQuery();

      ArrayList<User> user_list = new ArrayList<User>();

      while (rs.next()) {
        User user = new User();

        user.setEmailAddress(rs.getString("email_address"));
        user.setPassword(rs.getString("password"));
        user.setFirstName(rs.getString("first_name"));
        user.setLastName(rs.getString("last_name"));
        user.setBirthday(rs.getDate("birthday"));
        user.setDescription(rs.getString("description"));
        user.setOccupation(rs.getString("occupation"));
        user.setOrganization(rs.getString("organization"));
        user.setContactNumber(rs.getString("contact_number"));
        user.setSystemRole(rs.getString("system_role"));

user.setCommunity(CommunityDB.select(rs.getLong("community_id")))
;
        user.setStudentNumber(rs.getString("student_number"));

user.setRegistrationTimestamp(rs.getTimestamp("registration_timesta
mp"));
        user.setExpiration(rs.getTimestamp("expiration"));
        user.setLastActivity(rs.getTimestamp("last_activity"));

        user_list.add(user);
      }

      return user_list;
    }
    catch (SQLException e) {
      e.printStackTrace();

      return null;
    }
    finally {
      DBUtil.closeResultSet(rs);
      DBUtil.closePreparedStatement(ps);
      pool.freeConnection(connection);
    }
  }

  public static ArrayList<User> selectModerators(Community
community) {
    ConnectionPool pool = ConnectionPool.getInstance();
    Connection connection = pool.getConnection();
    PreparedStatement ps = null;
    ResultSet rs = null;

    String query = "SELECT * FROM user WHERE system_role =
'upmir_community_moderator' AND community_id = ?";

    try {
      ps = connection.prepareStatement(query);
      ps.setLong(1, community.getID());
      rs = ps.executeQuery();

      ArrayList<User> user_list = new ArrayList<User>();

      while (rs.next()) {
        User user = new User();

        user.setEmailAddress(rs.getString("email_address"));
        user.setPassword(rs.getString("password"));
        user.setFirstName(rs.getString("first_name"));
        user.setLastName(rs.getString("last_name"));
        user.setBirthday(rs.getDate("birthday"));
        user.setDescription(rs.getString("description"));
        user.setOccupation(rs.getString("occupation"));
        user.setOrganization(rs.getString("organization"));
        user.setContactNumber(rs.getString("contact_number"));
        user.setSystemRole(rs.getString("system_role"));
```

```
user.setCommunity(CommunityDB.select(rs.getLong("community_id")))
;
       user.setStudentNumber(rs.getString("student_number"));

user.setRegistrationTimestamp(rs.getTimestamp("registration_timesta
mp"));
       user.setExpiration(rs.getTimestamp("expiration"));
       user.setLastActivity(rs.getTimestamp("last_activity"));

       user_list.add(user);
     }

     return user_list;
   }
   catch (SQLException e) {
     e.printStackTrace();

     return null;
   }
   finally {
     DBUtil.closeResultSet(rs);
     DBUtil.closePreparedStatement(ps);
     pool.freeConnection(connection);
   }
 }

  public static int insert(User user) {
    ConnectionPool pool = ConnectionPool.getInstance();
       Connection connection = pool.getConnection();
       PreparedStatement ps = null;
       ResultSet rs = null;

       String query = "INSERT INTO user " +
                 "(email_address, password,       first_name,
last_name,      birthday,"     +
                 " description,   occupation,     organization,
contact_number, system_role,"   +
                 " community_id,  student_number, registration_timestamp,
expiration,     last_activity)" +
                 " VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, NOW(), ?,
NOW())";

       try {
          ps = connection.prepareStatement(query);

          ps.setString(1, user.getEmailAddress());
          ps.setString(2, user.getPassword());
          ps.setString(3, user.getFirstName());
          ps.setString(4, user.getLastName());
          ps.setString(5, user.getBirthday());
          ps.setString(6, user.getDescription());
          ps.setString(7, user.getOccupation());
          ps.setString(8, user.getOrganization());
          ps.setString(9, user.getContactNumber());
          ps.setString(10, user.getSystemRole());
          ps.setLong(11,
CommunityDB.selectID(user.getCommunity().getCode()));
          ps.setString(12, user.getStudentNumber());
          ps.setString(13, user.getExpiration());

          return ps.executeUpdate();
       }
       catch (SQLException e) {
          e.printStackTrace();

          return 0;
       }
       finally {
          DBUtil.closeResultSet(rs);
          DBUtil.closePreparedStatement(ps);
          pool.freeConnection(connection);
       }
  }

  public static int update(User user) {
    ConnectionPool pool = ConnectionPool.getInstance();
       Connection connection = pool.getConnection();
       PreparedStatement ps = null;
       ResultSet rs = null;

       String query = "UPDATE user SET "       +
                 "email_address = ?, "     +
                       "password = ?, "          +
                       "first_name = ?, "        +
                       "last_name = ?, "         +
                       "birthday = ?, "          +
                       "description = ?, "       +
                       "occupation = ?, "        +
                       "organization = ?, "      +
                       "contact_number = ?, "    +
                       "system_role = ?, "       +
                       "community_id = ?, "      +
                       "student_number = ?, "    +
                       "expiration = ?, "        +
                       "last_activity = ? "      +
                       "WHERE email_address = ?";

       try {
         ps = connection.prepareStatement(query);

          ps.setString(1, user.getEmailAddress());
          ps.setString(2, user.getPassword());
          ps.setString(3, user.getFirstName());
          ps.setString(4, user.getLastName());
          ps.setString(5, user.getBirthday());
          ps.setString(6, user.getDescription());
          ps.setString(7, user.getOccupation());
          ps.setString(8, user.getOrganization());
          ps.setString(9, user.getContactNumber());
          ps.setString(10, user.getSystemRole());
          ps.setLong(11,
CommunityDB.selectID(user.getCommunity().getCode()));
          ps.setString(12, user.getStudentNumber());
          ps.setString(13, user.getExpiration());
          ps.setString(14, user.getLastActivity());
          ps.setString(15, user.getEmailAddress());

          return ps.executeUpdate();
       }
       catch (SQLException e) {
         e.printStackTrace();

          return 0;
```

```
    }
    finally {
      DBUtil.closeResultSet(rs);
        DBUtil.closePreparedStatement(ps);
        pool.freeConnection(connection);
    }
  }

  public static int update(long user_id, User user) {
    ConnectionPool pool = ConnectionPool.getInstance();
    Connection connection = pool.getConnection();
    PreparedStatement ps = null;
    ResultSet rs = null;

    String query = "UPDATE user SET "       +
           "email_address = ?, "     +
                "password = ?, "         +
                "first_name = ?, "       +
                "last_name = ?, "        +
                "birthday = ?, "         +
                "description = ?, "      +
                "occupation = ?, "       +
                "organization = ?, "     +
                "contact_number = ?, "   +
                "system_role = ?, "      +
                "community_id = ?, "     +
                "student_number = ?, "   +
                "expiration = ? "        +
                "WHERE user_id = ?";

    try {
      ps = connection.prepareStatement(query);

        ps.setString(1, user.getEmailAddress());

        String password = "";

        if (user.getPassword().equals(DigestUtils.md5Hex(""))) {
          password = select(user_id).getPassword();
        }
        else {
          password = user.getPassword();
        }

        ps.setString(2, password);
        ps.setString(3, user.getFirstName());
        ps.setString(4, user.getLastName());
        ps.setString(5, user.getBirthday());
        ps.setString(6, user.getDescription());
        ps.setString(7, user.getOccupation());
        ps.setString(8, user.getOrganization());
        ps.setString(9, user.getContactNumber());
        ps.setString(10, user.getSystemRole());
        ps.setLong(11,
CommunityDB.selectID(user.getCommunity().getCode()));
        ps.setString(12, user.getStudentNumber());
        ps.setString(13, user.getExpiration());
        ps.setLong(14, user_id);

       return ps.executeUpdate();
    }
    catch (SQLException e) {
      e.printStackTrace();

       return 0;
    }
    finally {
      DBUtil.closeResultSet(rs);
        DBUtil.closePreparedStatement(ps);
        pool.freeConnection(connection);
    }
  }

  public static int delete(long user_id) {
    ConnectionPool pool = ConnectionPool.getInstance();
    Connection connection = pool.getConnection();
    PreparedStatement ps = null;
    ResultSet rs = null;

    String query = "DELETE FROM user WHERE user_id = ?";

    try {
      ps = connection.prepareStatement(query);
        ps.setLong(1, user_id);

       return ps.executeUpdate();
    }
    catch (SQLException e) {
      e.printStackTrace();

       return 0;
    }
    finally {
      DBUtil.closeResultSet(rs);
        DBUtil.closePreparedStatement(ps);
        pool.freeConnection(connection);
    }
  }

  public static int refreshActivity(User user) {
    ConnectionPool pool = ConnectionPool.getInstance();
    Connection connection = pool.getConnection();
    PreparedStatement ps = null;
    ResultSet rs = null;

    String query = "UPDATE user SET last_activity = NOW() WHERE
email_address = ?";

    try {
      ps = connection.prepareStatement(query);

         ps.setString(1, user.getEmailAddress());

       return ps.executeUpdate();
    }
    catch (SQLException e) {
      e.printStackTrace();

       return 0;
    }
    finally {
      DBUtil.closeResultSet(rs);
        DBUtil.closePreparedStatement(ps);
        pool.freeConnection(connection);
```

```
    }
  }
}
```

Annotation.java

```
package domain;

import java.io.Serializable;
import java.text.DateFormat;
import java.text.SimpleDateFormat;
import java.util.Date;

public class Annotation implements Serializable {
  private String approved;
  private String content;
  private int start_location;
  private int end_location;
  private User user;
  private Material material;
  private Date annotation_timestamp;

  public Annotation() {
    approved = "n";
    content = "";
    start_location = -1;
    end_location = -1;
    user = new User();
    material = new Material();
    annotation_timestamp = new Date();
  }

  public void setApproved(String approved) {
    this.approved = approved;
  }

  public String getApproved() {
    return approved;
  }

  public void setContent(String content) {
    this.content = content;
  }

  public String getContent() {
    return content;
  }

  public void setStartLocation(int start_location) {
    this.start_location = start_location;
  }

  public int getStartLocation() {
    return start_location;
  }

  public void setEndLocation(int end_location) {
    this.end_location = end_location;
  }

  public int getEndLocation() {
    return end_location;
  }

  public void setUser(User user) {
    this.user = user;
  }

  public User getUser() {
    return user;
  }

  public void setMaterial(Material material) {
    this.material = material;
  }

  public Material getMaterial() {
    return material;
  }

  public void setAnnotationTimestamp(Date annotation_timestamp) {
    this.annotation_timestamp = annotation_timestamp;
  }

  public String getAnnotationTimestamp() {
      DateFormat date_format = new SimpleDateFormat("yyyy-MM-dd
HH:mm:ss");
      String annotation_timestamp_formatted =
date_format.format(annotation_timestamp);
      return annotation_timestamp_formatted;
  }

  public String getAnnotationDateLong() {
      DateFormat date_format = new SimpleDateFormat("MMMM d,
yyyy");
      String annotation_timestamp_formatted =
date_format.format(annotation_timestamp);
      return annotation_timestamp_formatted;
  }

  public String getAnnotationDateAlt() {
      DateFormat date_format = new SimpleDateFormat("MM/dd/yy");
      String annotation_timestamp_formatted =
date_format.format(annotation_timestamp);
      return annotation_timestamp_formatted;
  }

  public String getAnnotationTimestampLong() {
      DateFormat date_format = new SimpleDateFormat("MMMM d,
yyyy, hh:mm:ss a");
      String annotation_timestamp_formatted =
date_format.format(annotation_timestamp);
      return annotation_timestamp_formatted;
  }
}
```

Comment.java

```
package domain;

import java.io.Serializable;
import java.text.DateFormat;
import java.text.SimpleDateFormat;
import java.util.Date;
```

```
import util.PrettyTime;

public class Comment implements Serializable {
  private String content;
  private User user;
  private Material material;
  private Date comment_timestamp;

  public Comment() {
    content = "";
    user = new User();
    material = new Material();
    comment_timestamp = new Date();
  }

  public void setContent(String content) {
    this.content = content;
  }

  public String getContent() {
    return content;
  }

  public void setUser(User user) {
    this.user = user;
  }

  public User getUser() {
    return user;
  }

  public void setMaterial(Material material) {
    this.material = material;
  }

  public Material getMaterial() {
    return material;
  }

  public void setCommentTimestamp(Date comment_timestamp) {
    this.comment_timestamp = comment_timestamp;
  }

  public String getCommentTimestamp() {
      DateFormat date_format = new SimpleDateFormat("yyyy-MM-dd
HH:mm:ss");
      String comment_timestamp_formatted =
date_format.format(comment_timestamp);
      return comment_timestamp_formatted;
  }

  public String getCommentTimestampLong() {
      DateFormat date_format = new SimpleDateFormat("MMMM d,
yyyy, hh:mm:ss a");
      String comment_timestamp_formatted =
date_format.format(comment_timestamp);
      return comment_timestamp_formatted;
  }

  public String getCommentTimestampPretty() {
    return PrettyTime.format(comment_timestamp, new Date());
  }
}
```

Community.java

```
package domain;

import java.io.Serializable;
import java.text.DateFormat;
import java.text.SimpleDateFormat;
import java.util.Date;

import data.CommunityDB;
import data.MaterialDB;

public class Community implements Serializable {
  private String code;
  private String name;
  private String description;
  private Date creation_timestamp;

  public Community() {
    code = "";
    name = "";
    description = "";
    creation_timestamp = new Date();
  }

  public void setCode(String code) {
    this.code = code;
  }

  public String getCode() {
    return code;
  }

  public void setName(String name) {
    this.name = name;
  }

  public String getName() {
    return name;
  }

  public void setDescription(String description) {
    this.description = description;
  }

  public String getDescription() {
    return description;
  }

  public void setCreationTimestamp(Date creation_timestamp) {
    this.creation_timestamp = creation_timestamp;
  }

  public String getCreationTimestamp() {
      DateFormat date_format = new SimpleDateFormat("yyyy-MM-dd
HH:mm:ss");
      String creation_timestamp_formatted =
date_format.format(creation_timestamp);
      return creation_timestamp_formatted;
```

```
  }

  public String getCreationDateLong() {
       DateFormat date_format = new SimpleDateFormat("MMMM d,
yyyy, hh:mm:ss a");
       String creation_timestamp_formatted =
date_format.format(creation_timestamp);
       return creation_timestamp_formatted;
  }

  public String getOptionName() {
    String option_name = name + " (" + code + ")";

    return option_name;
  }

  public long getID() {
    return CommunityDB.selectID(code);
  }

  public int getPendingCount() {
    return MaterialDB.countPending(this);
  }
}
```

Favorite.java

```
package domain;

import java.io.Serializable;

public class Favorite implements Serializable {
  private String description;
  private String tags;
  private Material material;
  private long author;
  private User user;

  public Favorite() {
    description = "";
    tags = "";
    material = new Material();
    author = -1;
    user = new User();
  }

  public void setDescription(String description) {
    this.description = description;
  }

  public String getDescription() {
    return description;
  }

  public void setTags(String tags) {
    this.tags = tags;
  }

  public String getTags() {
    return tags;
  }

  public void setMaterial(Material material) {
    this.material = material;
  }

  public Material getMaterial() {
    return material;
  }

  public void setAuthor(long author) {
    this.author = author;
  }

  public long getAuthor() {
    return author;
  }

  public void setUser(User user) {
    this.user = user;
  }

  public User getUser() {
    return user;
  }
}
```

Material.java

```
package domain;

import java.io.Serializable;
import java.text.DateFormat;
import java.text.SimpleDateFormat;
import java.util.Calendar;
import java.util.Date;
import java.util.StringTokenizer;

import util.PrettyTime;

import data.MaterialDB;
import domain.User;

public class Material implements Serializable {
  private String pid;
  private Metadata metadata;
  private Community community;
  private String tags;
  private String preceded_by;
  private String history;
  private Date embargo;
  private User user;
  private Date deposit_timestamp;
  private String filename;
  private String file_type;
  private long file_size;
  private String index;

  public Material() {
    pid = "";
    metadata = new Metadata();
    community = new Community();
    tags = "";
    preceded_by = "";
```

```
  history = "";
  embargo = new Date();
  user = new User();
  deposit_timestamp = new Date();
  filename = "";
  file_type = "";
  file_size = 0;
  index = "";
}

public void setPID(String pid) {
  this.pid = pid;
}

public String getPID() {
  return pid;
}

public void setMetadata(Metadata metadata) {
  this.metadata = metadata;
}

public Metadata getMetadata() {
  return metadata;
}

public void setCommunity(Community community) {
  this.community = community;
}

public Community getCommunity() {
  return community;
}

public void setTags(String tags) {
  this.tags = tags;
}

public String getTags() {
  return tags;
}

public void setPrecededBy(String preceded_by) {
  this.preceded_by = preceded_by;
}

public String getPrecededBy() {
  return preceded_by;
}

public void setHistory(String history) {
  this.history = history;
}

public String getHistory() {
  return history;
}

public void setEmbargo(Date embargo) {
  this.embargo = embargo;
}

  public String getEmbargo() {
    if (embargo == null) {
      return null;
    }
    else {
        DateFormat date_format = new SimpleDateFormat("yyyy-MM-
dd");
        String embargo_formatted = date_format.format(embargo);
        return embargo_formatted;
    }
  }

  public void setUser(User user) {
    this.user = user;
  }

  public User getUser() {
    return user;
  }

  public void setDepositTimestamp(Date deposit_timestamp) {
    this.deposit_timestamp = deposit_timestamp;
  }

  public String getDepositTimestamp() {
      DateFormat date_format = new SimpleDateFormat("yyyy-MM-
dd");
      String deposit_timestamp_formatted =
date_format.format(deposit_timestamp);
      return deposit_timestamp_formatted;
  }

  public void setFilename(String filename) {
    this.filename = filename;
  }

  public String getFilename() {
    return filename;
  }

  public void setFileType(String file_type) {
    this.file_type = file_type;
  }

  public String getFileType() {
    return file_type;
  }

  public void setFileSize(long file_size) {
    this.file_size = file_size;
  }

  public long getFileSize() {
    return file_size;
  }

  public void setIndex(String index) {
    this.index = index;
  }

  public String getIndex() {
    return index;
```

```
  }

  public String getID() {
    return pid.substring(4);
  }

  public String getFileSizeDetails() {
    float size = file_size;
    if (file_size/1048576 > 0) {
      return String.format("%.2f MB", size/1048576);
    }
    else if (file_size/1024 > 0) {
      return String.format("%.2f kB", size/1024);
    }
    else {
      return String.format("%.2f bytes", size);
    }
  }

  public String getDepositDateLong() {
       DateFormat date_format = new SimpleDateFormat("MMMM d,
yyyy");
       String deposit_timestamp_formatted =
date_format.format(deposit_timestamp);
       return deposit_timestamp_formatted;
  }

  public String getDepositTimestampLong() {
       DateFormat date_format = new SimpleDateFormat("MMMM d,
yyyy, hh:mm:ss a");
       String deposit_timestamp_formatted =
date_format.format(deposit_timestamp);
       return deposit_timestamp_formatted;
  }

  public String getDepositTimestampPretty() {
    return PrettyTime.format(deposit_timestamp, new Date());
  }

  public String getEmbargoLong() {
    if (embargo == null) {
      return null;
    }
    else {
      DateFormat date_format = new SimpleDateFormat("MMMM d,
yyyy");
        String embargo_formatted = date_format.format(embargo);
        return embargo_formatted;
    }
  }

  public int getEmbargoMonth() {
       if (embargo == null) {
         return -1;
       }
       else {
      Calendar calendar = Calendar.getInstance();
      calendar.setTime(embargo);
      return calendar.get(Calendar.MONTH) + 1;
       }
  }

  public int getEmbargoDay() {
       if (embargo == null) {
         return -1;
       }
       else {
      Calendar calendar = Calendar.getInstance();
      calendar.setTime(embargo);
      return calendar.get(Calendar.DATE);
       }
  }

  public int getEmbargoYear() {
       if (embargo == null) {
         return -1;
       }
       else {
      Calendar calendar = Calendar.getInstance();
      calendar.setTime(embargo);
      return calendar.get(Calendar.YEAR);
       }
  }

  public String getIndexSearch() {
    if (index.length() > 290) {
      return index.substring(0, 290) + " ...";
    }

    return index;
  }

  public String getTagLinks() {
    String tag_links = "";

    StringTokenizer st = new StringTokenizer(tags, ",");

    while (st.hasMoreTokens()) {
      String val = st.nextToken();
      tag_links += "<a class='tag_style' href='search?query=" + val.trim()
+ "'>" + val.trim() + "</a> ";
    }

    return tag_links;
  }

  public String getHistoryID() {
    return history.substring(4);
  }

  public Material getLatestVersion() {
    return MaterialDB.selectLatestVersion(history);
  }

  public String getFileIcon() {
    String icon = "";

    // txt
    if (file_type.trim().equals("text/plain")) {
      icon = "<img src='images/icons/text.png' title='text'> ";
    }

    // pdf
    else if (file_type.trim().equals("application/pdf")) {
```

```
    icon = "<img src='images/icons/pdf.png' title='pdf'> ";
  }

  // mpeg (audio)
  else if (file_type.trim().equals("audio/mpeg")) {
    icon = "<img src='images/icons/music.png' title='mpeg'> ";
  }

  // mp4 (audio)
  else if (file_type.trim().equals("audio/mp4")) {
    icon = "<img src='images/icons/music.png' title='mp4'> ";
  }

  // gif
  else if (file_type.trim().equals("image/gif")) {
    icon = "<img src='images/icons/image.png' title='gif'> ";
  }

  // jpeg
  else if (file_type.trim().equals("image/jpeg")) {
    icon = "<img src='images/icons/image.png' title='jpeg'> ";
  }

  // png
  else if (file_type.trim().equals("image")) {
    icon = "<img src='images/icons/image.png' title='png'> ";
  }

  // doc or docx
  else if (file_type.trim().equals("application/msword")) {
    icon = "<img src='images/icons/word.png' title='doc/docx'> ";
  }

  // ppt
  else if (file_type.trim().equals("application/vnd.ms-powerpoint")) {
    icon = "<img src='images/icons/powerpoint.png' title='ppt/pptx'> ";
  }

  // xls
  else if (file_type.trim().equals("application/vnd.ms-excel")) {
    icon = "<img src='images/icons/powerpoint.png' title='xls'> ";
  }

  // mpeg (video)
  else if (file_type.trim().equals("video/mpeg")) {
    icon = "<img src='images/icons/film.png' title='mpeg'> ";
  }

  // mp4 (video)
  else if (file_type.trim().equals("video/mp4")) {
    icon = "<img src='images/icons/film.png' title='mp4'> ";
  }

  return icon;
}

public String getThumbnail() {
  String thumbnail = MaterialDB.selectThumbnail(pid);

  if (thumbnail == null || thumbnail.isEmpty()) {
    return null;
  }
  else {
    return thumbnail;
  }
 }
}
```

Message.java

```
package domain;

import java.io.Serializable;
import java.text.DateFormat;
import java.text.SimpleDateFormat;
import java.util.Date;

public class Message implements Serializable {
  private long message_id;
  private boolean read;
  private String subject;
  private String content;
  private User sender;
  private User recipient;
  private String type;
  private Date message_timestamp;
  private String material_id;

  public Message() {
    message_id = -1;
    read = false;
    subject = "";
    content = "";
    sender = new User();
    recipient = new User();
    type = "plain";
    message_timestamp = new Date();
    material_id = null;
  }

  public void setID(long message_id) {
    this.message_id = message_id;
  }

  public Long getID() {
    return message_id;
  }

  public void setRead(String read) {
    if (read.equals("y")) {
      this.read = true;
    }
    else {
      this.read = false;
    }
  }

  public boolean getRead() {
    return read;
  }

  public void setSubject(String subject) {
    this.subject = subject;
  }
```

```
 public String getSubject() {
  return subject;
 }

 public void setContent(String content) {
  this.content = content;
 }

 public String getContent() {
  return content;
 }

 public void setSender(User sender) {
  this.sender = sender;
 }

 public User getSender() {
  return sender;
 }

 public void setRecipient(User recipient) {
  this.recipient = recipient;
 }

 public User getRecipient() {
  return recipient;
 }

 public void setType(String type) {
  this.type = type;
 }

 public String getType() {
  return type;
 }

 public void setMessageTimestamp(Date message_timestamp) {
  this.message_timestamp = message_timestamp;
 }

 public String getMessageTimestamp() {
      DateFormat date_format = new SimpleDateFormat("yyyy-MM-
dd");
      String message_timestamp_formatted =
date_format.format(message_timestamp);
      return message_timestamp_formatted;
 }

 public void setMaterialID(String material_id) {
  this.material_id = material_id;
 }

 public String getMaterialID() {
  return material_id;
 }

 public String getMessageTimestampLong() {
      DateFormat date_format = new SimpleDateFormat("MMMM d,
yyyy, hh:mm:ss a");
      String message_timestamp_formatted =
date_format.format(message_timestamp);
      return message_timestamp_formatted;
 }

 public String getSubjectShort() {
  if (subject.length() > 30) {
   return subject.substring(0, 30) + " ...";
  }

  return subject;
 }
}
```

Metadata.java

```
package domain;

import java.io.Serializable;
import java.text.DateFormat;
import java.text.SimpleDateFormat;
import java.util.Calendar;
import java.util.Date;

public class Metadata implements Serializable {
 private String dc_contributor;
 private String dc_coverage;
 private String dc_creator;
 private Date dc_date;
 private String dc_description;
 private String dc_format;
 private String dc_identifier;
 private String dc_language;
 private String dc_publisher;
 private String dc_relation;
 private String dc_rights;
 private String dc_source;
 private String dc_subject;
 private String dc_title;
 private String dc_type;

 public Metadata() {
  dc_contributor = "";
  dc_coverage = "";
  dc_creator = "";
  dc_date = new Date();
  dc_description = "";
  dc_format = "";
  dc_identifier = "";
  dc_language = "";
  dc_publisher = "";
  dc_relation = "";
  dc_rights = "";
  dc_source = "";
  dc_subject = "";
  dc_title = "";
  dc_type = "";
 }

 public void setContributor(String dc_contributor) {
  this.dc_contributor = dc_contributor;
 }

 public String getContributor() {
```

```
  return dc_contributor;
 }

 public void setCoverage(String dc_coverage) {
  this.dc_coverage = dc_coverage;
 }

 public String getCoverage() {
  return dc_coverage;
 }

 public void setCreator(String dc_creator) {
  this.dc_creator = dc_creator;
 }

 public String getCreator() {
  return dc_creator;
 }

 public void setDate(Date dc_date) {
  this.dc_date = dc_date;
 }

 public String getDate() {
      if (dc_date == null) {
        return null;
      }
      else {
        DateFormat date_format = new SimpleDateFormat("yyyy-MM-
dd");
         String dc_date_formatted = date_format.format(dc_date);
         return dc_date_formatted;
      }
 }

 public void setDescription(String dc_description) {
  this.dc_description = dc_description;
 }

 public String getDescription() {
  return dc_description;
 }

 public void setFormat(String dc_format) {
  this.dc_format = dc_format;
 }

 public String getFormat() {
  return dc_format;
 }

 public void setIdentifier(String dc_identifier) {
  this.dc_identifier = dc_identifier;
 }

 public String getIdentifier() {
  return dc_identifier;
 }

 public void setLanguage(String dc_language) {
  this.dc_language = dc_language;
 }

 public String getLanguage() {
  return dc_language;
 }

 public void setPublisher(String dc_publisher) {
  this.dc_publisher = dc_publisher;
 }

 public String getPublisher() {
  return dc_publisher;
 }

 public void setRelation(String dc_relation) {
  this.dc_relation = dc_relation;
 }

 public String getRelation() {
  return dc_relation;
 }

 public void setRights(String dc_rights) {
  this.dc_rights = dc_rights;
 }

 public String getRights() {
  return dc_rights;
 }

 public void setSource(String dc_source) {
  this.dc_source = dc_source;
 }

 public String getSource() {
  return dc_source;
 }

 public void setSubject(String dc_subject) {
  this.dc_subject = dc_subject;
 }

 public String getSubject() {
  return dc_subject;
 }

 public void setTitle(String dc_title) {
  this.dc_title = dc_title;
 }

 public String getTitle() {
  return dc_title;
 }

 public void setType(String dc_type) {
  this.dc_type = dc_type;
 }

 public String getType() {
  return dc_type;
 }

 public String getDateLong() {
```

```
      if (dc_date == null) {
        return null;
      }
      else {
        DateFormat date_format = new SimpleDateFormat("MMMM dd,
yyyy");
          String dc_date_formatted = date_format.format(dc_date);
          return dc_date_formatted;
      }
  }

  public int getDateMonth() {
      if (dc_date == null) {
        return -1;
      }
      else {
     Calendar calendar = Calendar.getInstance();
     calendar.setTime(dc_date);
     return calendar.get(Calendar.MONTH) + 1;
      }
  }

  public int getDateDay() {
      if (dc_date == null) {
        return -1;
      }
      else {
     Calendar calendar = Calendar.getInstance();
     calendar.setTime(dc_date);
     return calendar.get(Calendar.DATE);
      }
  }

  public int getDateYear() {
      if (dc_date == null) {
        return -1;
      }
      else {
     Calendar calendar = Calendar.getInstance();
     calendar.setTime(dc_date);
     return calendar.get(Calendar.YEAR);
      }
  }
}
```

News.java

```
package domain;

import java.io.Serializable;
import java.text.DateFormat;
import java.text.SimpleDateFormat;
import java.util.Date;

public class News implements Serializable {
  private String title;
  private String reference;
  private String content;
  private User user;
  private Date news_timestamp;

  public News() {
    title = "";
    reference = "";
    content = "";
    user = new User();
    news_timestamp = new Date();
  }

  public void setTitle(String title) {
    this.title = title;
  }

  public String getTitle() {
    return title;
  }

  public void setReference(String reference) {
    this.reference = reference;
  }

  public String getReference() {
    return reference;
  }

  public void setContent(String content) {
    this.content = content;
  }

  public String getContent() {
    return content;
  }

  public void setUser(User user) {
    this.user = user;
  }

  public User getUser() {
    return user;
  }

  public void setNewsTimestamp(Date news_timestamp) {
    this.news_timestamp = news_timestamp;
  }

  public String getNewsTimestamp() {
      DateFormat date_format = new SimpleDateFormat("yyyy-MM-dd
HH:mm:ss");
      String news_timestamp_formatted =
date_format.format(news_timestamp);
      return news_timestamp_formatted;
  }

  public String getNewsDateAlt() {
    DateFormat date_format = new SimpleDateFormat("MM-dd-yyyy");
      String news_timestamp_formatted =
date_format.format(news_timestamp);
      return news_timestamp_formatted;
  }

  public String getNewsTimestampLong() {
      DateFormat date_format = new SimpleDateFormat("MMMM d,
yyyy, hh:mm:ss a");
```

```
        String news_timestamp_formatted =
date_format.format(news_timestamp);
        return news_timestamp_formatted;
  }

  public String getMonth() {
    DateFormat date_format = new SimpleDateFormat("MM");
    return date_format.format(news_timestamp);
  }

  public String getDay() {
    DateFormat date_format = new SimpleDateFormat("dd");
    return date_format.format(news_timestamp);
  }

  public String getYear() {
    DateFormat date_format = new SimpleDateFormat("yy");
    return date_format.format(news_timestamp);
  }
}
```

Operation.java

```
package domain;

import java.io.Serializable;
import java.text.DateFormat;
import java.text.SimpleDateFormat;
import java.util.Date;

public class Operation implements Serializable {
  private String description;
  private Date operation_timestamp;

  public Operation() {
    description = "";
    operation_timestamp = new Date();
  }

  public Operation(String description) {
    this.description = description;
    operation_timestamp = new Date();
  }

  public void setDescription(String description) {
    this.description = description;
  }

  public String getDescription() {
    return description;
  }

  public void setOperationTimestamp(Date operation_timestamp) {
    this.operation_timestamp = operation_timestamp;
  }

  public String getOperationTimestamp() {
        DateFormat date_format = new SimpleDateFormat("yyyy-MM-dd
HH:mm:ss");
        String operation_timestamp_formatted =
date_format.format(operation_timestamp);
        return operation_timestamp_formatted;
  }

  public String getOperationTimestampLong() {
        DateFormat date_format = new SimpleDateFormat("MMMM d,
yyyy, hh:mm:ss a");
        String operation_timestamp_formatted =
date_format.format(operation_timestamp);
        return operation_timestamp_formatted;
  }
}
```

User.java

```
package domain;

import java.io.Serializable;
import java.text.DateFormat;
import java.text.SimpleDateFormat;
import java.util.Calendar;
import java.util.Date;

import util.PrettyTime;

import data.UserDB;

public class User implements Serializable {
  private String email_address;
  private String password;
  private String first_name;
  private String last_name;
  private Date birthday;
  private String description;
  private String occupation;
  private String organization;
  private String contact_number;
  private String system_role;
  private Community community;
  private String student_number;
  private Date registration_timestamp;
  private Date expiration;
  private Date last_activity;

  public User() {
    email_address = "";
    password = "";
    first_name = "";
    last_name = "";
    birthday = new Date();
    description = "";
    occupation = "";
    organization = "";
    contact_number = "";
    system_role = "";
    community = new Community();
    student_number = "";
    registration_timestamp = new Date();
    expiration = new Date();
    last_activity = new Date();
  }

  public void setEmailAddress(String email_address) {
    this.email_address = email_address;
```

```
}

public String getEmailAddress() {
  return email_address;
}

public void setPassword(String password) {
  this.password = password;
}

public String getPassword() {
  return password;
}

public void setFirstName(String first_name) {
  this.first_name = first_name;
}

public String getFirstName() {
  return first_name;
}

public void setLastName(String last_name) {
  this.last_name = last_name;
}

public String getLastName() {
  return last_name;
}

public void setBirthday(Date birthday) {
  this.birthday = birthday;
}

public String getBirthday() {
     DateFormat date_format = new SimpleDateFormat("yyyy-MM-
dd");
     String birthday_formatted = date_format.format(birthday);
     return birthday_formatted;
}

public void setDescription(String description) {
  this.description = description;
}

public String getDescription() {
  return description;
}

public void setOccupation(String occupation) {
  this.occupation = occupation;
}

public String getOccupation() {
  return occupation;
}

public void setOrganization(String organization) {
  this.organization = organization;
}

public String getOrganization() {
  return organization;
}

public void setContactNumber(String contact_number) {
  this.contact_number = contact_number;
}

public String getContactNumber() {
  return contact_number;
}

public void setSystemRole(String system_role) {
  this.system_role = system_role;
}

public String getSystemRole() {
  return system_role;
}

public void setCommunity(Community community) {
  this.community = community;
}

public Community getCommunity() {
  return community;
}

public void setStudentNumber(String student_number) {
  this.student_number = student_number;
}

public String getStudentNumber() {
  return student_number;
}

public void setRegistrationTimestamp(Date registration_timestamp) {
  this.registration_timestamp = registration_timestamp;
}

public String getRegistrationTimestamp() {
     DateFormat date_format = new SimpleDateFormat("yyyy-MM-
dd");
     String registration_timestamp_formatted =
date_format.format(registration_timestamp);
     return registration_timestamp_formatted;
}

public void setExpiration(Date expiration) {
  this.expiration = expiration;
}

public String getExpiration() {
     DateFormat date_format = new SimpleDateFormat("yyyy-MM-
dd");
     String expiration_formatted = date_format.format(expiration);
     return expiration_formatted;
}

public void setLastActivity(Date last_activity) {
  this.last_activity = last_activity;
}
```

```
  public String getLastActivity() {
      DateFormat date_format = new SimpleDateFormat("yyyy-MM-
dd");
      String last_activity_formatted = date_format.format(last_activity);
      return last_activity_formatted;
  }

  public String getSystemRoleNormal() {
    if (system_role.equals("upmir_system_administrator")) {
      return "System Administrator";
    }
    else if (system_role.equals("upmir_community_moderator")) {
      return "Community Moderator";
    }
    else if (system_role.equals("upmir_university_user")) {
      return "University User";
    }
    else if (system_role.equals("upmir_registered_user")) {
      return "Registered User";
    }

    return "n/a";
  }

  public String getSystemRoleStylized() {
    if (system_role.equals("upmir_system_administrator")) {
      return "<span class=\"role_style_1\">System
Administrator</span>";
    }
    else if (system_role.equals("upmir_community_moderator")) {
      return "<span class=\"role_style_2\" title=\"Community Moderator of
" + community.getCode() + "\">Community Moderator</span>";
    }
    else if (system_role.equals("upmir_university_user")) {
      return "<span class=\"role_style_3\">University User</span>";
    }

    return "";
  }

  public String getSystemRoleStylizedShort() {
    if (system_role.equals("upmir_system_administrator")) {
      return "<span class=\"role_style_1\" title=\"System
Administrator\">SA</span>";
    }
    else if (system_role.equals("upmir_community_moderator")) {
      return "<span class=\"role_style_2\" title=\"" + community.getCode()
+ " Community Moderator\">CM</span>";
    }
    else if (system_role.equals("upmir_university_user")) {
      return "<span class=\"role_style_3\" title=\"University
User\">U</span>";
    }

    return "";
  }

  public String getCommunityDetails() {
    if (community == null) {
      return "<i>deleted</i>";
    }
    else {
      return community.getCode() + " - " + community.getName();
    }
  }

  public String getBirthdayLong() {
      DateFormat date_format = new SimpleDateFormat("MMMM d,
yyyy");
      String birthday_formatted = date_format.format(birthday);
      return birthday_formatted;
  }

  public int getBirthdayMonth() {
    Calendar calendar = Calendar.getInstance();
    calendar.setTime(birthday);
    return calendar.get(Calendar.MONTH) + 1;
  }

  public int getBirthdayDay() {
    Calendar calendar = Calendar.getInstance();
    calendar.setTime(birthday);
    return calendar.get(Calendar.DATE);
  }

  public int getBirthdayYear() {
    Calendar calendar = Calendar.getInstance();
    calendar.setTime(birthday);
    return calendar.get(Calendar.YEAR);
  }

  public String getRegistrationTimestampLong() {
      DateFormat date_format = new SimpleDateFormat("MMMM d,
yyyy, hh:mm:ss a");
      String registration_timestamp_formatted =
date_format.format(registration_timestamp);
      return registration_timestamp_formatted;
  }

  public String getExpirationLong() {
      DateFormat date_format = new SimpleDateFormat("MMMM d,
yyyy");
      String expiration_formatted = date_format.format(expiration);
      return expiration_formatted;
  }

  public int getExpirationMonth() {
    Calendar calendar = Calendar.getInstance();
    calendar.setTime(expiration);
    return calendar.get(Calendar.MONTH) + 1;
  }

  public int getExpirationDay() {
    Calendar calendar = Calendar.getInstance();
    calendar.setTime(expiration);
    return calendar.get(Calendar.DATE);
  }

  public int getExpirationYear() {
    Calendar calendar = Calendar.getInstance();
    calendar.setTime(expiration);
    return calendar.get(Calendar.YEAR);
  }
```

```
  public String getLastActivityPretty() {
      return PrettyTime.format(last_activity, new Date());
  }

  public long getID() {
    return UserDB.selectID(email_address);
  }
}
```

DeleteComment.java

```
package user;

import java.io.IOException;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import data.CommentDB;

public class DeleteComment extends HttpServlet {
  public void doPost(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
    if (request.getParameter("comment_id") == null) {
      response.sendRedirect(request.getHeader("Referer"));

      return;
    }
    else {
      if
(CommentDB.commentExists(Long.parseLong(request.getParameter("
comment_id")))) {

CommentDB.delete(Long.parseLong(request.getParameter("comment_
id")));
      }

      response.sendRedirect(request.getHeader("Referer"));
    }
  }
}
```

DeleteFavorite.java

```
package user;

import java.io.IOException;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;

import data.FavoriteDB;
import domain.User;

public class DeleteFavorite extends HttpServlet {
  public void doPost(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
    HttpSession session = request.getSession();
    User user = (User)session.getAttribute("me");

    if (request.getParameter("material_id") == null ||
request.getParameter("delete_favorite_referer") == null) {
      response.sendRedirect(request.getContextPath() + "/user/favorite");

      return;
    }
    else {
      String pid = "get:" + request.getParameter("material_id");

      FavoriteDB.delete(user, pid);

      // Deleted on material page.
      if
(request.getParameter("delete_favorite_referer").equals("material")) {
        response.sendRedirect(request.getContextPath() + "/file?id=" +
request.getParameter("material_id"));

        return;
      }
      // Deleted on favorite page.
      else {
        response.sendRedirect(request.getContextPath() +
"/user/favorite");

        return;
      }
    }
  }

  public void doGet(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
    response.sendRedirect(request.getContextPath() + "/user/favorite");
  }
}
```

DeleteMessage.java

```
package user;

import java.io.IOException;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;

import data.MessageDB;
import domain.User;

public class DeleteMessage extends HttpServlet {
  public void doPost(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
    HttpSession session = request.getSession();

    if (request.getParameter("message_id") == null) {
      session.setAttribute("delete_message", "false");
    }
    else {
      User user = (User)session.getAttribute("me");

      int i = 0;
```

```
    long message_id =
Long.parseLong(request.getParameter("message_id"));

    if (user.getSystemRole().equals("upmir_system_administrator")) {
      i = MessageDB.deleteAdmin(message_id, user);
    }
    else if
(user.getSystemRole().equals("upmir_community_moderator")) {
      i = MessageDB.deleteModerator(message_id, user);
    }
    else {
      i = MessageDB.delete(message_id, user);
    }

    if (i > 0) {
      session.setAttribute("delete_message", "true");
    }
    else {
      session.setAttribute("delete_message", "false");
    }
   }

   response.sendRedirect(request.getContextPath() + "/user/inbox");
  }

  public void doGet(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
   response.sendRedirect(request.getContextPath() + "/user/inbox");
  }
}
```

DisplayFavoriteList.java

```
package user;

import java.io.IOException;
import java.util.LinkedHashMap;

import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;

import com.google.gson.Gson;

import data.FavoriteDB;
import domain.Favorite;
import domain.User;

public class DisplayFavoriteList extends HttpServlet {
  public void doGet(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
    HttpSession session = request.getSession();
    User user = (User)session.getAttribute("me");

    LinkedHashMap<Long, Favorite> favorite_list =
FavoriteDB.selectMine(user);

     String json = new Gson().toJson(favorite_list);

     response.setContentType("application/json");
     response.setCharacterEncoding("UTF-8");
     response.getWriter().write(json);
  }

  public void doPost(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
    doGet(request, response);
  }
}
```

DisplayMessage.java

```
package user;

import java.io.IOException;

import javax.servlet.RequestDispatcher;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;

import data.MessageDB;
import domain.Message;
import domain.User;

public class DisplayMessage extends HttpServlet {
  public void doGet(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
    if (request.getParameter("id") == null) {
     response.sendRedirect(request.getContextPath() + "/user/inbox");

     return;
    }
    else {
     HttpSession session = request.getSession();
     User user = (User)session.getAttribute("me");

     long message_id = Long.parseLong(request.getParameter("id"));

     // Tag message as read.
     if (user.getSystemRole().equals("upmir_system_administrator")) {
       MessageDB.readAdmin(message_id);

       // Update message number count.
       long unread_messages_count =
MessageDB.countMineUnreadAdmin(user);
       if (unread_messages_count > 0) {
         session.setAttribute("unread_messages_text", " (" +
unread_messages_count + " unread)");
       }
       else {
         session.setAttribute("unread_messages_text", "");
       }

       // Get message.
       Message message = MessageDB.selectAdmin(message_id,
user);
       request.setAttribute("msg", message);
     }
```

```
    else if
(user.getSystemRole().equals("upmir_community_moderator")) {
      MessageDB.readAdmin(message_id);

      // Update message number count.
      long unread_messages_count =
MessageDB.countMineUnreadModerator(user);
      if (unread_messages_count > 0) {
        session.setAttribute("unread_messages_text", " (" +
unread_messages_count + " unread)");
      }
      else {
        session.setAttribute("unread_messages_text", "");
      }

      // Get message.
      Message message = MessageDB.selectModerator(message_id,
user);
      request.setAttribute("msg", message);
    }
    else {
      MessageDB.read(message_id, user);

      // Update message number count.
      long unread_messages_count =
MessageDB.countMineUnread(user);
      if (unread_messages_count > 0) {
        session.setAttribute("unread_messages_text", " (" +
unread_messages_count + " unread)");
      }
      else {
        session.setAttribute("unread_messages_text", "");
      }

      // Get message.
      Message message = MessageDB.select(message_id, user);
      request.setAttribute("msg", message);
    }
   }

   String url = "/WEB-INF/user/read_message.jsp";
   RequestDispatcher dispatcher =
getServletContext().getRequestDispatcher(url);
   dispatcher.forward(request, response);
  }

  public void doPost(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
   doGet(request, response);
  }
}
```

DisplayMessageList.java

```
package user;

import java.io.IOException;
import java.util.ArrayList;

import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;

import com.google.gson.Gson;

import data.MessageDB;
import data.SQLUtil;
import domain.Message;
import domain.User;

public class DisplayMessageList extends HttpServlet {
  public void doGet(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
    HttpSession session = request.getSession();
    User user = (User)session.getAttribute("me");

    if (user != null) {
      String[] cols = {"type", "subject", "CONCAT(first_name, ' ',
last_name)", "message_timestamp", "`read`", "message_id", "sender",
"DATE_FORMAT(message_timestamp, '%M %e, %Y, %k:%i:%s')"};
      String filter = "";
      String order = "";
      String limit = "";

      // Filtering
      filter = SQLUtil.getFilter(request, cols);

      // Ordering
      order = SQLUtil.getOrder(request, cols);

      // Paging
      limit = SQLUtil.getLimit(request);

      MessageJson mjson = new MessageJson();

      if (user.getSystemRole().equals("upmir_system_administrator")) {
        mjson.setData(MessageDB.selectMineAdmin(filter, order, limit,
user));
        mjson.setEcho(Integer.parseInt(request.getParameter("sEcho")));
        mjson.setTotalRecords(MessageDB.countAdmin(user));

mjson.setTotalDisplayRecords(MessageDB.countFilterAdmin(filter,
user));
      }
      else if
(user.getSystemRole().equals("upmir_community_moderator")) {
        mjson.setData(MessageDB.selectMineModerator(filter, order,
limit, user));
        mjson.setEcho(Integer.parseInt(request.getParameter("sEcho")));
        mjson.setTotalRecords(MessageDB.countModerator(user));

mjson.setTotalDisplayRecords(MessageDB.countFilterModerator(filter,
user));
      }
      else {
        mjson.setData(MessageDB.selectMine(filter, order, limit, user));
        mjson.setEcho(Integer.parseInt(request.getParameter("sEcho")));
        mjson.setTotalRecords(MessageDB.count(user));
        mjson.setTotalDisplayRecords(MessageDB.countFilter(filter,
user));
      }
```

```
    response.setContentType("application/json");
    response.setCharacterEncoding("UTF-8");
    response.getWriter().write(new Gson().toJson(mjson));
   }
  }

  public void doPost(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
    doGet(request, response);
  }
}

class MessageJson {
  private int sEcho;
  private int iTotalRecords;
  private int iTotalDisplayRecords;
  private String[][] aaData;

  public MessageJson() {
    sEcho = 0;
    iTotalRecords = 0;
    iTotalDisplayRecords = 0;
    aaData = null;
  }

  public void setEcho(int sEcho) {
    this.sEcho = sEcho;
  }

  public int getEcho() {
    return sEcho;
  }

  public void setTotalRecords(int iTotalRecords) {
    this.iTotalRecords = iTotalRecords;
  }

  public int getTotalRecords() {
    return iTotalRecords;
  }

  public void setTotalDisplayRecords(int iTotalDisplayRecords) {
    this.iTotalDisplayRecords = iTotalDisplayRecords;
  }

  public int getTotalDisplayRecords() {
    return iTotalDisplayRecords;
  }

  public void setData(ArrayList<Message> message_list) {
    if (message_list != null) {
      aaData = new String[message_list.size()][8];

      for (int i = 0; i < message_list.size(); i++) {
        // Type
        aaData[i][0] = message_list.get(i).getType();
        // Subject
        aaData[i][1] = message_list.get(i).getSubject();
        // Sender
        aaData[i][2] = message_list.get(i).getSender().getFirstName() + " "
+ message_list.get(i).getSender().getLastName();
        // Date
        aaData[i][3] = message_list.get(i).getMessageTimestampLong();

        // Read
        if (message_list.get(i).getRead()) {
          aaData[i][4] = "y";
        }
        else {
          aaData[i][4] = "n";
        }

        // Message ID
        aaData[i][5] = message_list.get(i).getID().toString();
        // Full name of sender
        aaData[i][6] = message_list.get(i).getSender().getFirstName() + " "
+ message_list.get(i).getSender().getLastName();
        // Date (long format)
        aaData[i][7] = message_list.get(i).getMessageTimestampLong();
      }
    }
    else {
      aaData = null;
    }
  }

  public String[][] getData() {
    return aaData;
  }
}
```

DisplayUser.java

```
package user;

import java.io.IOException;

import javax.servlet.RequestDispatcher;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;

import data.MaterialDB;
import data.UserDB;
import domain.Material;
import domain.User;

public class DisplayUser extends HttpServlet {
  public void doGet(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
    if (request.getParameter("id") == null) {
      HttpSession session = request.getSession();
      User user = (User)session.getAttribute("me");

      response.sendRedirect(request.getContextPath() + "/profile?id=" +
user.getID());

      return;
    }

    if (UserDB.userExists(Long.parseLong(request.getParameter("id"))))
{
```

```
    User user =
UserDB.select(Long.parseLong(request.getParameter("id")));
    request.setAttribute("u", user);

    // Get latest submission.
    Material material = MaterialDB.selectLatest(user);
    request.setAttribute("m", material);
   }

   String url = "/WEB-INF/user/view.jsp";
   RequestDispatcher dispatcher =
getServletContext().getRequestDispatcher(url);
   dispatcher.forward(request, response);
 }

 public void doPost(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
   doGet(request, response);
 }
}
```

LoginUser.java

```
package user;

import java.io.IOException;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

public class LoginUser extends HttpServlet {
 public void doGet(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
   if (request.getHeader("Referer") == null) {
    response.sendRedirect(request.getContextPath());
   }
   else {
    response.sendRedirect(request.getHeader("Referer"));
   }
 }

 public void doPost(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
   doGet(request, response);
 }
}
```

LogoutUser.java

```
package user;

import java.io.IOException;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;

public class LogoutUser extends HttpServlet {
 public void doGet(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
   HttpSession session = request.getSession(false);

   if(request.isRequestedSessionIdValid() ) {
    session.invalidate();
   }

   response.sendRedirect(request.getContextPath());
 }

 public void doPost(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
   doGet(request, response);
 }
}
```

ReportMaterial.java

```
package user;

import java.io.IOException;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;

import data.MaterialDB;
import data.MessageDB;
import data.OperationDB;
import domain.Material;
import domain.Message;
import domain.User;

public class ReportMaterial extends HttpServlet {
 public void doPost(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
   validateMessage(request);

   HttpSession session = request.getSession();
   User sender = (User)session.getAttribute("me");

   long material_id =
Long.parseLong(request.getParameter("material_id"));
   Material material = MaterialDB.select("get:" + material_id);

   Message message = new Message();
   message.setSubject("A new report on a material has been filed.");

   String content = "Admin/Moderators,<br><br>" +
       "A new report has been filed by user <a href=\"profile?id=" +
sender.getID() + "\">" + sender.getFirstName() + " " +
sender.getLastName() + "</a> " +
       "on file titled <a href=\"file?id=" + material_id + "\">" +
material.getMetadata().getTitle() + "</a>. Stated below is the
<strong>reason</strong>:" +
       "<div class=\"read_message_report_description\">" +
request.getParameter("report_description") + "</div>" +
       "Click <a href=\"file?id=" + material_id + "\">here</a> to go
directly to the reported material. " +
       "Be sure to tag this message as resolved after reviewing and
making the appropriate action/s to this complaint.";
```

```
    message.setContent(content);
    message.setSender(sender);
    message.setMaterialID(material.getPID());
    message.setRecipient(new User());
    message.setType("flag");

    int i = 0;

    i = MessageDB.insert(message);

    if (i > 0) {
      String op = "<a href='admin/u/view?id=" + sender.getID() + "'>" +
sender.getFirstName() + " " + sender.getLastName() + "</a> reported
material <a href='admin/m/view?id=" + material.getID() + "'>" +
material.getMetadata().getTitle() + "</a>.";
       OperationDB.insert(op);

      session.setAttribute("submit_message", "true");
    }
    else {
      session.setAttribute("submit_message", "false");
    }

    response.sendRedirect(request.getContextPath() + "/file?id=" +
material_id);
  }

  public void doGet(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
    response.sendRedirect(request.getContextPath());
  }

  private void validateMessage(HttpServletRequest request) throws
ServletException {
    // Report Description
    if (request.getParameter("report_description") == null ||
request.getParameter("report_description").trim().length() > 3000) {
      if (request.getParameter("report_description").trim().length() < 1) {
        throw new ServletException("Validation: Please specify the reason
for your report.");
      }
      else {
        throw new ServletException("Validation: Your report exceeds
character limit of 3000.");
      }
    }
  }
}
```

ReportUser.java

```
package user;

import java.io.IOException;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;

import data.MessageDB;
import data.OperationDB;
import data.UserDB;
import domain.Message;
import domain.User;

public class ReportUser extends HttpServlet {
  public void doPost(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
    validateMessage(request);

    HttpSession session = request.getSession();
    User sender = (User)session.getAttribute("me");

    long user_id = Long.parseLong(request.getParameter("uid"));
    User user = UserDB.select(user_id);

    Message message = new Message();
    message.setSubject("A new report on a user has been filed.");

    String content = "Admin,<br><br>" +
          "A new report has been filed by user <a href=\"profile?id=" +
sender.getID() + "\">" + sender.getFirstName() + " " +
sender.getLastName() + "</a> " +
          "on user <a href=\"profile?id=" + user_id + "\">" +
user.getFirstName() + " " + user.getLastName() + "</a>. Stated below
is the <strong>reason</strong>:" +
          "<div class=\"read_message_report_description\">" +
request.getParameter("report_description") + "</div>" +
          "Click <a href=\"profile?id=" + user_id + "\">here</a> to go
directly to the reported user. " +
          "Be sure to tag this message as resolved after reviewing and
making the appropriate action/s to this complaint.";

    message.setContent(content);
    message.setSender(sender);
    message.setRecipient(new User());
    message.setType("flag");

    int i = 0;

    i = MessageDB.insert(message);

    if (i > 0) {
      String op = "<a href='admin/u/view?id=" + sender.getID() + "'>" +
sender.getFirstName() + " " + sender.getLastName() + "</a> reported
user <a href='admin/u/view?id=" + user.getID() + "'>" +
user.getFirstName() + " " + user.getLastName() + "</a>.";
       OperationDB.insert(op);

      session.setAttribute("submit_message", "true");
    }
    else {
      session.setAttribute("submit_message", "false");
    }

    response.sendRedirect(request.getContextPath() + "/profile?id=" +
user_id);
  }

  public void doGet(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
    response.sendRedirect(request.getContextPath());
  }
```

```
  private void validateMessage(HttpServletRequest request) throws
ServletException {
    // Report Description
    if (request.getParameter("report_description") == null ||
request.getParameter("report_description").trim().length() > 3000) {
      if (request.getParameter("report_description").trim().length() < 1) {
        throw new ServletException("Validation: Please specify the reason
for your report.");
      }
      else {
        throw new ServletException("Validation: Your report exceeds
character limit of 3000.");
      }
    }
  }
}
```

SubmitComment.java

```
package user;

import java.io.IOException;

import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;

import data.CommentDB;
import data.MaterialDB;
import data.UserDB;
import domain.Comment;
import domain.User;

public class SubmitComment extends HttpServlet {
  public void doPost(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
    validateComment(request);

    HttpSession session = request.getSession();

    Comment comment = new Comment();

    User user = (User)session.getAttribute("me");


comment.setContent(request.getParameter("comment_content").trim())
;
    comment.setUser(UserDB.select(user.getEmailAddress()));
    comment.setMaterial(MaterialDB.select("get:" +
request.getParameter("material_id")));

    CommentDB.insert(comment);

    response.sendRedirect(request.getContextPath() + "/file?id=" +
request.getParameter("material_id"));
  }

  private void validateComment(HttpServletRequest request) throws
ServletException {
    // Content
    if (request.getParameter("comment_content") == null ||
request.getParameter("comment_content").trim().length() < 1 ||
request.getParameter("comment_content").trim().length() > 500) {
      throw new ServletException("Validation: Invalid comment.");
    }

    // Material
    if (request.getParameter("material_id") == null) {
      throw new ServletException("Validation: Invalid material.");
    }
    else {
      String pid = "get:" + request.getParameter("material_id");
      // Check material if it exists and if is available for viewing.
      if (!MaterialDB.materialExistsAndApproved(pid)) {
        throw new ServletException("Validation: Invalid material.");
      }
    }
  }
}
```

SubmitFavorite.java

```
package user;

import java.io.IOException;

import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;

import data.CommentDB;
import data.MaterialDB;
import data.UserDB;
import domain.Comment;
import domain.User;

public class SubmitComment extends HttpServlet {
  public void doPost(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
    validateComment(request);

    HttpSession session = request.getSession();

    Comment comment = new Comment();

    User user = (User)session.getAttribute("me");


comment.setContent(request.getParameter("comment_content").trim())
;
    comment.setUser(UserDB.select(user.getEmailAddress()));
    comment.setMaterial(MaterialDB.select("get:" +
request.getParameter("material_id")));

    CommentDB.insert(comment);

    response.sendRedirect(request.getContextPath() + "/file?id=" +
request.getParameter("material_id"));
  }
```

```
  private void validateComment(HttpServletRequest request) throws
ServletException {
    // Content
    if (request.getParameter("comment_content") == null ||
request.getParameter("comment_content").trim().length() < 1 ||
request.getParameter("comment_content").trim().length() > 500) {
      throw new ServletException("Validation: Invalid comment.");
    }

    // Material
    if (request.getParameter("material_id") == null) {
      throw new ServletException("Validation: Invalid material.");
    }
    else {
      String pid = "get:" + request.getParameter("material_id");
      // Check material if it exists and if is available for viewing.
      if (!MaterialDB.materialExistsAndApproved(pid)) {
        throw new ServletException("Validation: Invalid material.");
      }
    }
  }
}
```

SubmitMessage.java

```
package user;

import java.io.IOException;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;

import org.apache.commons.validator.routines.EmailValidator;

import data.MessageDB;
import data.UserDB;
import domain.Message;
import domain.User;

public class SubmitMessage extends HttpServlet {
  public void doPost(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
    validateMessage(request);

    HttpSession session = request.getSession();
    User sender = (User)session.getAttribute("me");

    Message message = new Message();


message.setSubject(request.getParameter("message_subject").trim());

message.setContent(request.getParameter("message_content").trim());
    message.setSender(sender);

    if (request.getParameter("email_address") == null) {

message.setRecipient(UserDB.select(Long.parseLong(request.getPara
meter("user_id"))));
    }
    else {

message.setRecipient(UserDB.select(request.getParameter("email_ad
dress")));
    }

    message.setType(request.getParameter("message_type"));

    int i = 0;

    i = MessageDB.insert(message);

    if (i > 0) {
      session.setAttribute("submit_message", "true");
    }
    else {
      session.setAttribute("submit_message", "false");
    }

    response.sendRedirect(request.getContextPath() + "/user/inbox");
  }

  public void doGet(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
    response.sendRedirect(request.getContextPath() + "/user/inbox");
  }

  private void validateMessage(HttpServletRequest request) throws
ServletException {
    // User is already determined.
    if (request.getParameter("email_address") == null) {
      // Check if user id was properly passed to this servlet.
      if (request.getParameter("user_id") == null) {
        throw new ServletException("Validation: Invalid recipient.");
      }
    }
    // Form requires an address (email).
    else {
      // Email Address
      if
(!EmailValidator.getInstance().isValid(request.getParameter("email_add
ress").trim())) {
        throw new ServletException("Validation: Invalid email address.");
      }
      else {
        if
(!UserDB.userExists(request.getParameter("email_address").trim())) {
          throw new ServletException("Validation: Invalid email address.");
        }
      }
    }

    // Subject
    if (request.getParameter("message_subject") == null ||
request.getParameter("message_subject").trim().length() < 1 ||
request.getParameter("message_subject").trim().length() > 200) {
      throw new ServletException("Validation: Invalid message subject.");
    }

    // Content
```

```
    if (request.getParameter("message_content") == null ||
request.getParameter("message_content").trim().length() > 5000) {
      if (request.getParameter("message_content").trim().length() < 1) {
        throw new ServletException("Validation: Message (field) is
required.");
      }
      else {
        throw new ServletException("Validation: Message (field) exceeds
character limit of 5000.");
      }
    }

    // Extra checks
    if (request.getParameter("message_type") == null ||
(!request.getParameter("message_type").equals("plain") &&
!request.getParameter("message_type").equals("support") &&
!request.getParameter("message_type").equals("flag") &&
!request.getParameter("message_type").equals("attention")) &&
!request.getParameter("message_type").equals("annotation")) {
      throw new ServletException("Validation: Invalid message type.");
    }
  }
}
```

SubmitSupport.java

```
package user;

import java.io.IOException;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;

import data.MessageDB;
import data.OperationDB;
import domain.Message;
import domain.User;

public class SubmitSupport extends HttpServlet {
  public void doPost(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
    validateMessage(request);

    HttpSession session = request.getSession();
    User sender = (User)session.getAttribute("me");

    Message message = new Message();

    message.setSubject("New message created using the 'Contact Us'
form");

message.setContent(request.getParameter("message_content").trim());
    message.setSender(sender);
    message.setRecipient(new User());
    message.setType("support");

    int i = 0;

    i = MessageDB.insert(message);

    if (i > 0) {
      String op = "<a href='profile?id=" + sender.getID() + "'>" +
sender.getFirstName() + " " + sender.getLastName() + "</a> sent a <a
href='user/inbox/read?id=" + i + "'>message</a> using the 'Contact Us'
form.";
      OperationDB.insert(op);

      session.setAttribute("submit_message", "true");
    }
    else {
      session.setAttribute("submit_message", "false");
    }

    response.sendRedirect(request.getContextPath() + "/help");
  }

  public void doGet(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
    response.sendRedirect(request.getContextPath());
  }

  private void validateMessage(HttpServletRequest request) throws
ServletException {
    if (request.getParameter("message_content") == null ||
request.getParameter("message_content").trim().length() > 5000) {
      if (request.getParameter("message_content").trim().length() < 1) {
        throw new ServletException("Validation: Message is required.");
      }
      else {
        throw new ServletException("Validation: Your message exceeds
character limit of 5000.");
      }
    }
  }
}
```

SubmitUserSettings.java

```
package user;

import java.io.IOException;
import java.text.DateFormat;
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.Date;
import java.util.TimeZone;

import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;

import org.apache.commons.codec.digest.DigestUtils;
import org.apache.commons.validator.routines.EmailValidator;

import data.OperationDB;
import data.UserDB;
import domain.Community;
import domain.User;

public class SubmitUserSettings extends HttpServlet {
```

```
  public void doPost(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
    // Validate input. (server-side)
    validateUser(request);

    HttpSession session = request.getSession();

    User user = (User)session.getAttribute("me");
    long user_id = UserDB.selectID(user.getEmailAddress());


user.setEmailAddress(request.getParameter("email_address").trim());

user.setPassword(DigestUtils.md5Hex(request.getParameter("passwor
d")));
    user.setFirstName(request.getParameter("first_name").trim());
    user.setLastName(request.getParameter("last_name").trim());
    user.setDescription(request.getParameter("description").trim());
    user.setOccupation(request.getParameter("occupation").trim());
    user.setOrganization(request.getParameter("organization").trim());

user.setContactNumber(request.getParameter("contact_number").trim(
));

    user.setStudentNumber("");

    if (user.getSystemRole().equals("upmir_community_moderator")) {

//user.setCommunity(CommunityDB.select(request.getParameter("com
munity").trim()));
    }
    else {
      user.setCommunity(new Community());
    }

    DateFormat df = new SimpleDateFormat("yyyy-MM-dd");
    df.setTimeZone(TimeZone.getTimeZone("Asia/Manila"));
    df.setLenient(false);

    Date birthday = new Date();

    try {
      // Birthday
      birthday = df.parse(request.getParameter("birthday_year") + "-" +
request.getParameter("birthday_month") + "-" +
request.getParameter("birthday_day"));
    }
    catch (ParseException e) {
      e.printStackTrace();
    }

    user.setBirthday(birthday);

    int i = 0;
    String action_redirect = "/user/settings";

    if (UserDB.userExists(user_id)) {
      i = UserDB.update(user_id, user);
    }
    else {
      action_redirect = "";
    }

    if (i > 0) {
      String op = "<a href='profile?id=" + user.getID() + "'>" +
user.getFirstName() + " " + user.getLastName() + "</a> updated his/her
account.";
      OperationDB.insert(op);

      session.setAttribute("submit_user", "true");
    }
    else {
      session.setAttribute("submit_user", "false");
    }

    response.sendRedirect(request.getContextPath() + action_redirect);
  }

  public void doGet(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
    response.sendRedirect(request.getContextPath() + "/user/settings");
  }

  private void validateUser(HttpServletRequest request) throws
ServletException {
    // Email Address
    if
(!EmailValidator.getInstance().isValid(request.getParameter("email_add
ress").trim())) {
      throw new ServletException("Validation: Invalid email address.");
    }

    // Password
    if (request.getParameter("password").length() > 0 ||
request.getParameter("password2").length() > 0) {
      if
(!request.getParameter("password").equals(request.getParameter("pas
sword2"))) {
        throw new ServletException("Validation: Passwords don't
match.");
      }
      else {
        if (request.getParameter("password").length() < 5) {
          throw new ServletException("Validation: Invalid password.");
        }
      }
    }

    // First Name
    if (request.getParameter("first_name").trim().length() < 1) {
      throw new ServletException("Validation: First name (field) is
blank.");
    }

    // Last Name
    if (request.getParameter("last_name").trim().length() < 1) {
      throw new ServletException("Validation: Last name (field) is
blank.");
    }

    DateFormat df = new SimpleDateFormat("yyyy-MM-dd");
    df.setLenient(false);
    Date current_date = new Date();
    Date test_date;
```

```
    // Birthday
    try {
      test_date = df.parse(request.getParameter("birthday_year") + "-" +
request.getParameter("birthday_month") + "-" +
request.getParameter("birthday_day"));

      if (test_date.after(current_date)) {
        throw new ServletException("Validation: Invalid date for birthday
(field).");
      }
    }
    catch (ParseException e) {
      throw new ServletException("Validation: Invalid date for birthday
(field).");
    }

    // Description
    if (request.getParameter("description").trim().length() > 500) {
      throw new ServletException("Validation: Description (field) exceeds
character limit of 500.");
    }

    // Occupation
    if (request.getParameter("occupation").trim().length() > 100) {
      throw new ServletException("Validation: Occupation (field) exceeds
character limit of 100.");
    }

    // Organization
    if (request.getParameter("organization").trim().length() > 100) {
      throw new ServletException("Validation: Organizatoin (field)
exceeds character limit of 100.");
    }

    // Contact Number
    if (request.getParameter("contact_number").trim().length() > 25) {
      throw new ServletException("Validation: Contact number (field)
exceeds character limit of 25.");
    }
  }
}
```