

FILE SYSTEM REPOSITORY DESIGN and IMPLEMENTATION for ABET

Melis Oner

A thesis submitted in partial fulfillment of
the requirements for the degree of
Master of Science

Department of Computer Science

Central Michigan University
Mount Pleasant, Michigan
May 2014

Accepted by the Faculty of the College of Graduate Studies,
Central Michigan University, in partial fulfillment of
the requirements for the master's degree

Thesis Committee:

_____	Committee Chair
_____	Faculty Member
_____	Faculty Member

Date: _____

_____	Dean College of Graduate Studies
-------	-------------------------------------

Date: _____

Committee:

Patrick Kinnicutt, Ph.D., Chair

Gognzu Hu, Prof.

Ahmet Ugur, Ph.D.

ACKNOWLEDGMENTS

I would like to convey my absolute gratitude to my Thesis Committee: Dr. Patrick Kinnicutt, Prof. Gongzu Hu, and Dr. Ahmet Ugur. Dr. Kinnicutt has been a tremendous support throughout this entire process and was always available when I had questions or needed assistance. Prof. Hu was always willing to answer questions or point me in the right direction when I needed it, providing valuable resources. Additionally Dr. Ugur was always encouraging and interested in my work. I am also exceedingly grateful to Central Michigan University for giving me this opportunity to complete this work.

ABSTRACT

FILE SYSTEM REPOSITORY DESIGN and IMPLEMENTATION for ABET

by Melis Oner

Repositories are important for universities and colleges in helping to manage and capture intellectual assets. These assets may include datasets, research work and learning material such as question papers, class lecture notes and syllabi. A digital repository can hold a wide range of materials for a variety of purposes and users. It can support research, learning, and administrative processes. Repositories can ensure the availability of these assets for long periods of time and this might be very useful for the future strategies. The advantage of repositories is that they help institutions to assess and improve their existing research and learning experience of students. Higher education institutions also have to manage their educational assets effectively and transparently. Digital repositories are useful to have a coherent and coordinated system to capture, identify, store and retrieve their assets.

Many universities and colleges have been using Course Management Systems such as Blackboard. Blackboard facilitates sharing course content such as class notes, assignments, syllabi, and grades for professors and students. However, a course management system doesn't substitute for a repository system in the long term. Central Michigan University's Computer Science Department would like to use a digital repository for ABET accreditation. A repository system will not only satisfy the ABET criteria but also improve the functionality and efficiency within the department by decreasing the time spent for managing hard copies. This project develops a digital repository for CMU's Computer Science Department. The goal of the system is to provide an easy environment for users to submit documents and keep track of the new versions of the documents. Users will also easily display, update and modify the documents and information. The primary focus of this work is to design an

extensible data model for future development. Another important objective is to lessen the response time of user queries to search for data by using a database back end.

TABLE OF CONTENTS

LIST OF FIGURES	viii
CHAPTER	
I. INTRODUCTION	1
II. LITERATURE SEARCH	2
History of Document Management Systems	2
Components of document management systems	3
Types of repositories	4
Benefits of repositories	5
Existing Repositories	6
III. TECHNOLOGY SELECTION	8
Language Selection	8
<i>Java</i>	8
Integrated Development Environment	9
<i>Eclipse</i>	9
<i>NetBeans</i>	10
Object-Oriented Analysis and Design	10
<i>UML</i>	10
Object-Oriented Development	11
<i>Software Architecture Pattern: Three-Tier</i>	11
<i>Software Design Pattern: Model-View-Controller (MVC)</i>	12
<i>Relational Database Design with MySQL</i>	12
<i>JDBC</i>	13
IV. PROJECT DESIGN	14
Software Architecture	14
<i>Problem Statement</i>	14
<i>Functional Requirements</i>	14
<i>Quality Attributes</i>	15
System Interaction	16
<i>Administer Operations</i>	16

	<i>Document Operations</i>	17
	System Structure	19
V.	IMPLEMENTATION	23
	MYSQL	23
	<i>Schema Design</i>	23
	<i>Administer privileges</i>	25
	Java	25
	<i>Entering and General Use of the System</i>	25
	<i>Operations on the User, Class and Section Classes</i>	26
	<i>Operations on the Document Classes</i>	28
	JDBC	30
	<i>Data Access Classes</i>	30
	<i>SQL Statements</i>	31
	<i>Result set</i>	32
	<i>Transactions</i>	33
VI.	CONCLUSION	35
	<i>Future Work</i>	35
	REFERENCES	36

LIST OF FIGURES

FIGURE	PAGE
1. Use Case Diagram For User Operations	16
2. Use Case Diagram For Document Operations	18
3. Database Tables	24
4. Class viewer, controller and Class Model structure	28
5. Displaying document version information	33
6. Example of data inconsistency in the database before using transactions	34

CHAPTER I

INTRODUCTION

Repositories are important for universities and colleges in helping to manage and capture intellectual assets. These assets may include datasets, research work and learning material such as question papers, class lecture notes and syllabi. A digital repository can hold a wide range of materials for a variety of purposes and users. It can support research, learning, and administrative processes. Repositories can ensure the availability of these assets for long periods of time and this might be very useful for the future strategies. The advantage of repositories is that they help institutions to assess and improve their existing research and learning experience of students. Higher education institutions also have to manage their educational assets effectively and transparently. Digital repositories are useful to have a coherent and coordinated system to capture, identify, store and retrieve their assets.

Many universities and colleges have been using Course Management Systems such as Blackboard. Blackboard facilitates sharing course content such as class notes, assignments, syllabi, and grades for professors and students. However, a course management system doesn't substitute for a repository system in the long term. CMU's Computer Science Department would like to use a digital repository for the ABET accreditation. This digital repository system will not only satisfy the ABET criteria but also improve the functionality and efficiency within the department by decreasing the time and space requirements for managing hard copies.

This project develops a digital repository for the CMU Computer Science Department. The goal of the system is to provide an easy environment for users to submit documents and keep track of multiple versions of the documents. Users will also easily display, update and modify the documents and information. The primary focus is to design an extensible data model for future development. Another important objective is to lessen the response time for user queries.

CHAPTER II

LITERATURE SEARCH

Before beginning the development of a new digital repository system, one must understand the roles and functionalities of existing systems. There are several repository types. Repositories have been developed as a solution to the document management problem. One of the biggest challenges facing organizations and institutions is the management of exponentially expanding number of documents. Today's applications can involve dozens of different components such as research results and data collections, but frequently they are not managed in a consistent way in the manner of the application's life cycle. In this chapter, we will talk about the history and development of document management systems, components of a document management system, institutional repositories, and existing systems.

History of Document Management Systems

In the early 1980s, a number of vendors began developing software systems to manage paper-based documents. These systems dealt with paper documents, which included printed and published documents, prints, and photographs [?]. Later, developers began developing another type of system which managed electronic documents created on computers and stored locally on users' local file systems. Early electronic document management systems could deal with a limited number of file formats. These systems enabled organizations to capture forms and save copies of the documents and store the files in the repository for security and quick retrieval [?].

Document management has become a regulatory requirement for different lines of businesses such as accounting, health services, and food safety. Businesses have to meet legal requirements to avoid loss of business and damage to their reputation. Procedures, work instructions, and policy

statements are a part of the evidence that they meet the regulations' and having the documents under control is important in this manner. This is why- document management has been standardized by ISO 35.240.30 catalog by International Standards Organization. This is the standard catalog for IT applications in information, documentation and publishing [?]. It regulates different aspects of document management such as vocabulary in databases, format of information change, font and character sets etc.

Components of document management systems

At a minimum level a document management system must provide security control, addition and version control, metadata capture and use.

Metadata

Metadata is typically stored for each document. For example, metadata include the date the document was stored and identity of the user storing it. Advanced document management systems may use character recognition to extract the metadata from the document [?].

Storage

Store electronic documents. Storage of the documents often includes management of the same documents; where they are stored, for how long, and eventual document destruction[?].

Versioning

Versioning is the process of assigning either unique version names or unique version numbers to unique states of document. Documents are checked in or out of the document management system, allowing users to retrieve different versions and to continue work from a selected point [?]. Versioning is useful for documents that change over time and require updating such as

Master Course Syllabi or Course Syllabi.

Security

Document security might be very important depending on the document type or business branch.

Some document management systems allows an administrator to give access to documents based on type to only certain people or groups of people [?].

Indexing

Indexing tracks electronic documents. It might be as simple as keeping track of unique document identifiers; but often it takes a more complex form, providing classification through the documents' metadata. Indexing supports retrieval [?].

Retrieval

Retrieval is to retrieve the electronic documents from the storage. Document retrieval is defined as the matching of some stated user query against a set of text records [?].

Types of repositories

Jess Thompson, a Gartner Inc. researcher explains repositories as a platform designed to manage a multitude of assets throughout their life cycles [1]. On the other hand, a digital repository is a mechanism for managing and storing digital content. Putting content into an institutional repository enables staff and institutions to manage and preserve it, and therefore derives maximum value from it [?].

Digital repositories may include a wide range of content for a variety of purposes and users. What goes into a repository is currently less an issue of technological or software capability, and more a policy decision by each institution or administrator [?]. Typically content can include administrative

data and research outputs such as journal articles, e-learning objects and teaching materials. Putting digital content into a repository enables institutions to manage and preserve it, and therefore derive maximum value from it.

An institutional repository is an on-line archive for collecting, preserving and disseminating digital copies of the intellectual output of an institution or particularly a research institution. For a university, this includes materials such as academic journal articles, and digital versions of theses and dissertations. It might include other digital assests generated by academics, such as administrative documents, course notes, or learning materials.

A repository established for the use of a particular academic department or laboratory is properly called a departmental repository, though the term institutional repository is also used. We are developing a digital repository but it is specifically a departmental repository. We will use digital, institutional, and departmental repository interchangeably throughout the chapter.

Benefits of repositories

As businesses grow and change, the volume of information grows. Without an effective way to manage all the documents, it is very likely to face higher costs, lower productivity, and increased compliance risk [?]. Case studies show that employees think a growing volume of papers increase their workload [?], and most of them find working with an electronic document system is easier than dealing with a big pile of paper [?]. Universities and educational institutions benefit from repositories in various ways:

- Inter operates with other university systems and maximize efficiencies between them by sharing information.

- Increases the prestige of institution, depending on the content contained.
- Shares expertise efficiently within institutions.
- Supports modular course development.
- Encourages review of curriculum, pedagogy and assessment.
- Protects unpublished or easily lost (gray) literature such as thesis or technical reports.

Existing Repositories

There are many vendors who provide document management services. According to Open Access Repositories, there are currently almost 1300 repositories around the world [?]. Over the past three years the number has been growing at an average of one per day. We will talk about a several important departmental repositories.

EPrints is a free and open-source software package for building open access repositories that are compliant with the Open Archives Initiative Protocol for Metadata Harvesting [?]. It shares many features seen in document management systems, but it is primarily used for institutional repositories and scientific journals. EPrints has been developed at the University of Southampton School of Electronics and Computer Science and released under GPL license. Eprints was created in 2000 and it became the first and one of the most widely used free open access institutional repository software. Eprints is a Web and command-line application based on LAMP architecture. It has written in Perl and works cross-platform.

Invenio is an open source repository software contains tools for management of digital assets. Invenio is developed by CERN (European Organization for Nuclear Research) Document Server Software Consortium, and it is freely available to download. Free and paid support is available [?]. It is

written in Python and works on Unix-like operating systems. It is released under the GPL License. It provides some powerful features. Documents are organized in collections. There are virtual collection trees and the collections are customizable. It is capable of handling articles, books, theses, photos, videos and museum objects. It also allows creating user-defined document baskets, basket sharing within user groups and Amazon-like user comments for documents in the repository and shared baskets.

SimpleDL is digital collection management software that allows for the upload, description, management and access of digital collections. SimpleDL is capable of handling documents, PDFs, images, audio files, and data only objects [?]. SimpleDL is mostly used by libraries, archives, government agencies, universities, and other organizations that wish to host a digital collection. Searching abilities are provided by Apache Solr which is an open source enterprise search platform from the Apache Lucene project [?]. Apache Lucene is a free information retrieval software library used for searching and indexing [?].

CHAPTER III

TECHNOLOGY SELECTION

An object-oriented approach to business/information systems means a system is defined as a collection of objects that work together to accomplish tasks. The objects can carry out actions when asked and each object maintains its own data.

Language Selection

Object-oriented development is relatively new but the concept dates back several decades. It started in the 1960s in Norway with the development of the Simula programming language. A major milestone in the history of OO was the development of the SmallTalk, a programming language by Alan Kay and his associates in Xerox PARC in the early 1970s. Additional object-oriented programming languages have been developed, including Objective-C and C++. In 1995, Sun Microsystems introduced Java as a pure OO language. Microsoft immediately released J++ and recently C# as alternatives to Java. Because Java is a pure OO language and it is platform-independent, it is an excellent choice for developing OO systems.

Java

Java is designed to be a powerful, full-featured, object-oriented development language that is easy to learn and use on any computing platform. It is also designed to support the development of applications for networked environments.

Java's power comes from its large, useful library of classes containing hundreds of predefined classes. These classes provide methods to accomplish tasks from simple number formatting to establishing network connections and accessing relational database classes.

Simplicity in Java is achieved by using a set of keywords that is smaller than in most other languages such as COBOL and Visual Basic. These languages have hundreds of keywords where Java has only 48. Portability in Java means that programs can be written and compiled only once and then run in many different processors and operating systems. This feature is achieved by using byte code and an interpreter. Byte code is produced when you compile a Java program and then executed under the control of an interpreter. The interpreter is called Java Virtual Machine (JVM). The combination of byte code and JVM gives a developer the ability to write a program on a Microsoft Windows system, and have it execute on a UNIX operating system.

Integrated Development Environment (IDE)

An Integrated Development Environment (IDE) is a set of software tools that helps you code, test and document the program you write. IDEs are helpful but not always necessary in software development projects. Basic text editors such as Pico, and Notepad or more advanced text editors such as Emacs, and vi can be used to generate source code files. Then source code files are compiled through the command line. The Java Development Kit (JDK) consists of the Java compiler, hundreds of prewritten classes and the JVM. Java source code files can be compiled through the command line if JDK is installed in the computer. On the other hand, IDEs improve development productivity by providing sophisticated tools with interacting with JDK. Editors provide color-coding, code completion, debugging and graphical development tools.

Eclipse

Eclipse is an IDE. It contains a workspace and extensible plug-in system for customizing the development environment. With various plug-ins, Eclipse can be used to develop applications in many

different languages. Eclipse is written in Java and works cross-platform. It also has plug-ins for documentation, revision control, and GUI editor. In our project, we used Eclipse Standard, version Kepler release 1.

NetBeans

NetBeans is another IDE for developing primarily with Java but also with other languages. It works cross-platform as well. NetBeans platform simplifies the Graphical User Interface development compared to Eclipse.

Object-Oriented Analysis and Design

In approach to development, system analysis means to study, understand, and define the requirements for the system. System requirements define what the system needs to accomplish for the users in the business terms. These requirements are usually described using diagrams, or models. A model depicts one aspect of the real world project. We need a set of models for different aspects of the system requirements. The models created during the analysis are called logical models and the models created during the system design are called physical models. Object-Oriented development requires models that define classes of objects and that depict object interactions. OO models and notation are based on Unified Modeling Language (UML).

UML

The acronym UML stands for Unified Modeling Language. UML is a visual modeling language used to design object oriented software applications. It provides a set of graphic notations to represent relationships and attributes in object-oriented systems. It is a convenient tool to model the different stages of the software development life cycle. UML is a standard way to visualize systems'

architectural elements such as activities, actors, business processes, database schemas, reusable software components, and programming language statements. With UML editors, it is easy to export the UML design into Java or other object oriented programming languages.

UML has two categories. The first category, Structure Diagrams consists of seven diagrams. Class, object, component, profile, composite structure, deployment and package diagrams represent the structural information. The second category, Behavioral Diagrams also consists seven diagrams. Activity, communication, interaction, sequence, state, timing and use case diagrams represent the system behavior. There are many UML creation tools. In this project we use Dia to model system structure and behavior.

Object-Oriented Development

Object-oriented development has three parts: Object-oriented analysis, object-oriented design and object-oriented programming. We discussed OOA in the previous sections. In this chapter we will talk about the architecture design and the programming requirements.

Software Architecture Pattern: Three-Tier

Multi-tier (or n-tier) is a client-server architecture in which presentation, application processing, and data management functions are physically separated. A three-tier architecture is typically composed of a presentation tier, a domain tier, and a data storage tier. For the sake of three-tier architecture pattern we divide our classes into three categories: problem-domain classes, interface classes, and data access classes. Problem-domain classes are the classes of objects specific to the application, such as the UserModel class, CourseAssignment class and MasterCourseSyllabus class. Interface classes are usually Graphical User Interface (GUI) classes and the objects in these classes are buttons,

labels, lists, and text boxes. In our system, we use a console-based, menu system as the interface. This could be changed to drop-down menu structure easily in the future. Finally, data access classes, work with the database management system.

Software Design Pattern: Model-View-Controller (MVC)

A design pattern is a general reusable solution to a commonly occurring problem. A design pattern is not source code nor a complete design. It is a description or template for how to solve a problem. Object-oriented design patterns show relationships and interactions between classes or objects.

MVC is a design pattern for implementing user interfaces. It divides the software applications into three components; model, view, controller and it defines the interactions between them. Controller-view and controller-model interaction might be different in different interpretations of MVC but model-view interaction (no interaction) remains the same. Models represent the data. All visual representation of the system are made through the view classes. Controller has the algorithm or program flow and it is the communication bridge both between view and model, and the system and the user. MVC is originally developed for desktop applications but it has been widely used in Web applications.

Relational Database Design with MySQL

A relational database provides tools to organize data into tables. Each column represents a field (attribute) and each row represents a record (object). Objects are identified by the primary keys, which means the primary key for each object is unique. In this system, we use MySQL database management system (DBMS). MySQL is one of the most popular DBMS. It is compatible with many

different platforms such as Linux, Windows, Mac OS and Solaris.

Java Database Connectivity (JDBC) and java.sql Package

There is one protocol required to access a MySQL database from Java: Java Database Connectivity (JDBC). JDBC is the Sun Microsystems protocol for database connectivity. In addition, MySQL connector, mysql-connector-java, is the official JDBC driver and it serves as a bridge between JDK and the DBMS.

The Structured Query Language (SQL) is a popular, standardized language used to access relational databases. We use SQL statements in the methods of data access classes to perform basic tasks. Java.sql package contains the classes and methods that we need to work with a database in Java.

CHAPTER IV

PROJECT DESIGN

When developing new software and technical systems it is important to begin with a detailed design. Software requirements draw the skeleton of the design. This section explores the architecture of the software and design of the system.

Software Architecture

Software systems are constructed to satisfy organizations business goals. The architecture is a bridge between the abstract-business goals and the concrete-resulting system. One of the definitions of software architecture is this one: The software architecture of a system is the set of structures needed to reason about the system, which comprise software elements, relations among them, and properties of both.

Problem Statement

Central Michigan University's Computer Science Department would like to use a digital repository system to archive the course-related documents. Currently, they don't have a digital system: All documents are stored manually and physically, which is both time and space consuming. They would like to change this to increase efficiency and satisfy ABET criteria.

Functional Requirements

A digital Repository system must categorize different type of documents, such as Master Course Syllabus, Course Syllabus, Course Assignments (homework, mid-terms etc.), and Student Records (best, average, and worst case examples). One important feature of the digital repository system is the version-control and back-up. The system must store the old versions of a file when it is

updated. Users must be able to store and retrieve files to/from the digital repository system. A variety of file types (such as .doc, .docx, .pdf, .png, etc.) must be supported.

Quality Attributes

A digital repository system must be *extensible* and *modifiable*. Modifiability is about change. Change can occur to any aspect of the system. Our software will be used by the department. The department may change the computers and the platforms in the future. So, carrying our system from one platform to another must not cost too much. This is why all the technologies we use are free and platform independent or cross platform. Departments' needs may change, and the digital repository system might function differently in the future. For example, the current system is designed to be used by only one person, the department secretary, and the host machine will be the computer in the secretary office. If department wants faculty members to upload their own documents, digital repository system will be operational with no modification: All main user operations are designed and implemented for the system administrator. Separating data access, representation and logic layers will also help to extend the system with minimal effort. If department wants to add more advanced GUI features or wants to carry the system from a local machine to the Web, the system will not require a completely different structure and design, thanks to the three-tier and MVC patterns. In addition, MySQL is the most popular database system used with PHP, which is a powerful server scripting language. If the application will be carried to the web, the database will not require any maintenance.

Security is a measure of the system's ability to protect data and information from unauthorized access. Currently, we do not anticipate any harmful attacks to the system since it is currently not on the Web. Since the number of users is very limited and the data content is not crucial, the system does not require high security precautions. We use Java's md5 hashing method for the passwords and we

won't store the raw password in the database as a security precaution.

Availability refers to a property of software that it is there and ready to carry out its task when you need it to be. According to our primary requirements, system will be available during the work hours, in a particular machine.

System Interaction

The base system is split into two sections: one for administrator, the other for general users. General users will have the ability to perform only the document operations. Administer user will have the ability to perform document operations as well as administrative operations.

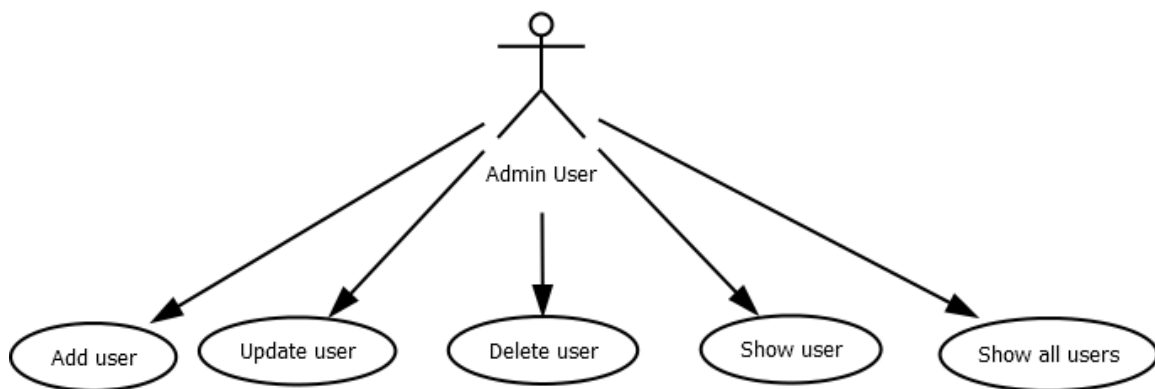


Figure 1. Use Case Diagram For User Operations

Administrative operations include User.java, Class.java, and Section.java classes. Each of these classes have the same functionality: add, delete, update, show, show all. In order to avoid data inconsistency, only the administrator will have the ability to perform these tasks. Document operations include slightly different functions on four other classes, which are MasterCourseSyllabus.java, CourseSyllabus.java, CourseAssignment.java and StudentRecord.java. Users will have the ability to upload a new document, display document information and retrieve a document's latest or an older version. Users will also have the ability to update or delete a document. The system will create a backup for the

updated/deleted document in the database. An overview of administrative operations and document operations can be seen in Figures 1, and 2.

Administrative Operations

To avoid data inconsistency, only the administrator will have the ability to perform operations on the User.java, Class.java and Section.java classes. Operations in these classes are straightforward. Administrators will have the ability to add new users to the system with their names, global ids. Administrators will be responsible for assigning administrator privileges and passwords for the new users. Classes and different sections of the classes will be saved by the administrator. Course names and all the sections of classes can be displayed. Classes and sections can be deleted and updated under some conditions. Course names can be updated any time. But course ids and course reference numbers can be deleted or updated if there is no document related to these courses or sections.

Document Operations

Document operations will be available to all users. Users will have the ability to add, delete, update documents and display document information. The primary purpose of this repository is to satisfy departmental needs. This is why documents are grouped in 4 categories: Master Course Syllabus, Course Syllabus, Student Records, and Course Assignments.

Master Course Syllabus indicates the general rules of a class independently from the instructor or the section. Course Syllabus is specific for different sections. There will be one course syllabus for one section, but there might be many course syllabi for a class. Course Assignment is used to store all kinds of course work that students are responsible for. It includes homeworks, project and lab assignments, quizzes, midterm and final exams. Student Records is used to store the best, average,

and worst examples of course assignments. First, users will choose a document type from the system menu. Each type of document requires different numbers of inputs. While course id is unique for a master course syllabus; course reference number (crn) will be unique for a course syllabus. Once the user enters the required inputs (such as course id and/or crn), the system will make sure that this class id/crn is already in the database. If not, then the user will have to contact the administrator and ask to add his/her section to the database.

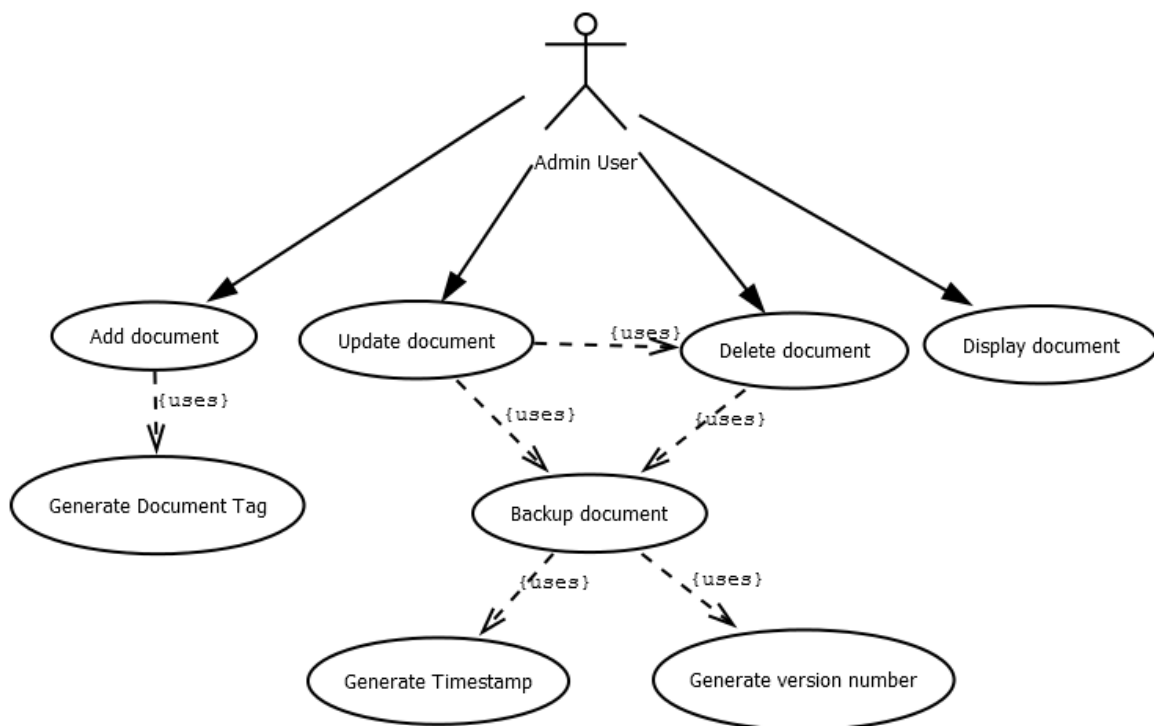


Figure 2. Use Case Diagram For Document Operations

Users will have the ability to delete and update an existing file. Once the user enters the required input, the system will search the database. Delete and update functions will be performed on document and back up documents tables in the database. If the document is in the database, system will create a back up document for the one that user wants to delete/update. The system will create a new version number and a time stamp, and save the existing file in the back up documents table.

After, if delete option is selected, document will be deleted from the documents table; if update option is selected document table will be updated with the new document, and time stamp.

When user wants to display information of a document, system will display the document type, the latest version's time stamp, and all of its versions and their timestamps. Latest version or a particular version of the document can be downloaded from the database to the C:\Users\globalId directory in user's computer.

System Structure

Once system's designer has a basic overview of the interaction between the user and the system as well as the interaction between different components of the system, they can begin to design more detailed pieces of the system such as classes and objects.

For this system, an MVC(Model-View-Controller) framework is developed with using factory and abstract factory design patterns. All operation classes have a model, viewer, model interface, viewer interface and a data access(DA) class. Every different functionality in the system is grouped as a Java package. There are eleven packages including user, class, and section packages; master course syllabus, course syllabus, assignment, student record, and document packages; menu, system, and logic packages.

A console-based menu structure is used to display the system functionalities. Menu is an important part of the system. Since there are more than 40 operations in total, not to print out 40 lines every time user wants to display the menu, it is divided into modules. There are 7 modules in total; userItems, classItems, sectionItems, mcsItems for master course syllabus operations, csItems for course syllabus operations, caItems for course assignment operations, and srItems for student record operations. Module class has three attributes; operation number, name and a list of items. Each

module can be displayed separately. Item class has two attributes; operation number and name. Name attributes are used to print the menu, and the number attributes are used to call the operation functions. Menus are composed from modules and modules are composed from Items. In the MenuModel class initializes all Items, Modules and Menus. There are two menus in the system; one is for administrator, the other is for the general user. The difference is some of the modules are not available for general user menu. When user logs in the system, depending on user's administrative privilege, system selects which menu to display. Using modules, gives great flexibility to insert and remove functionalities for different type of users. MenuViewer class has functions which interacts with the user. *printMenu()* prints the name of the modules of the menu. *askModule()* asks user to which operation set they want to go into. *askItem()* asks user which operation they want to perform.

System package has SystemModel, SystemViewer, SystemController, DatabaseConnectivity classes, and Model and Viewer interfaces. DatabaseConnectivity class has two functions; initialize and terminate. *initialize()* connects to the database with using JDBC driver and returns a connection object which is used in all of the data access classes. *terminate()* closes the connection. SystemViewer interacts with the user at the first step; takes user globalId and passes along SystemModel class in the SystemController's *login()* method. System controller has also *runMenu()* method, which operates the operation flow.

Login package has the AbstractMVCFactory, Factory and the main controller classes. AbstractMVCFactory class includes *makeModel()* and *makeViewer()* method signatures for every operation in the system. Factory class has all the *makeModel()* and *makeViewer()* methods. These methods create concrete models and viewers from the interfaces.

User, classOperations and section packages have exactly the same structure; Model and viewer classes, model and viewer interfaces, a data access class and map function interface. Viewer inter-

faces have three methods; *runInput(int functionNo)*, *passFunctionCode(int functionNo)*, and *printConsoleMessage(String message)*. Since this system is an MVC framework, user interactions and data operations are designed separate layers which communicates only through the controllers. For every operation in the menu, user will be asked to enter some input. But the required input will be different for every operation in a single module. Every viewer class has its own private methods such as *askClassId()*, *askCrn()* and it has a *setMap()* method. *setMap()* method, sets that class' operation map with filling a *userInputs* map object with using the class specific methods. Model classes have the same *setMap()* method and operation map object in addition to their constructor, and accessor methods. When an operation needs to use a class id or crn, the value is extracted from the *userInputs* map object with using its key. With this way, the order of the arguments don't matter. In the system controller's *runMenu()* method, Viewer passes along the *userInputs* object to the model class. Model class' *setMap()* function calls the associated data access class method. User, classOperations and section data access classes have the same functions; Find, add, delete, update, show, and show all. Data access class interacts with the database. Model class passes the required object to the DA function. In the DA function, sql queries are generated from the passed arguments and operation is performed on the database. Viewer displays the result of the query through the *printConsoleMessage()* method.

Document packages have the same structure with slightly different functionality. DA classes of Assignment, courseSyllabus, masterCourseSyllabus, and studentRecord packages have *retrive()* method, which is used to download a document from the database, addition to find, add, delete, update and show methods. In the database, all documents are stored in the *reposys.documents* table. But all different types have their own meta data tables. Data access methods in these packages are used to fill these meta data tables. However, DA methods of DocumentModelDA and BackupDocumentModelDA in the document.document package, interacts with the *reposys.document* and

reposys.backupdocuments table.

CHAPTER V

IMPLEMENTATION

This chapter explains how the project design is implemented to create the digital repository system. The uses of MySQL, Java and JDBC, and database schema used in MySQL are explained.

MYSQL

MySQL is a strong Database Management System (DBMS) which supports desktop and web applications. MySQL Server and Workbench were installed for database development purposes. MySQL offers database connectivity for using MySQL with applications and tools that are compatible with ODBC and JDBC. Connector/J is a standard database driver for Java platforms and development.

Schema Design

A relational database provides tools for developers to organize data into tables. Each column represents a field and each row represents a record. The database developed for the digital repository system includes nine tables: *usertable*, *course*, *section*, *assignment*, *coursesyllabus*, *mastercoursesyllabus*, *studentrecord*, *document* and *backup_documents* (Figure 3). These tables will be used to store User, Course, Section records, document meta data and the documents. The *usertable* contains users' unique ids which are named as *globalId*, *password*, *name* and *administer privilege* information. This table will be used for authenticating user and deciding which menu to display to the user.

The *course* table includes ids and names of the courses offered by CMU's Computer Science department. Course id, a six character-long variable (such as CPS100 or ITC341), is used as the primary key. As we explained in the Document Operations section, master course syllabi will be unique for every class id. One class might be offered by many different instructors.

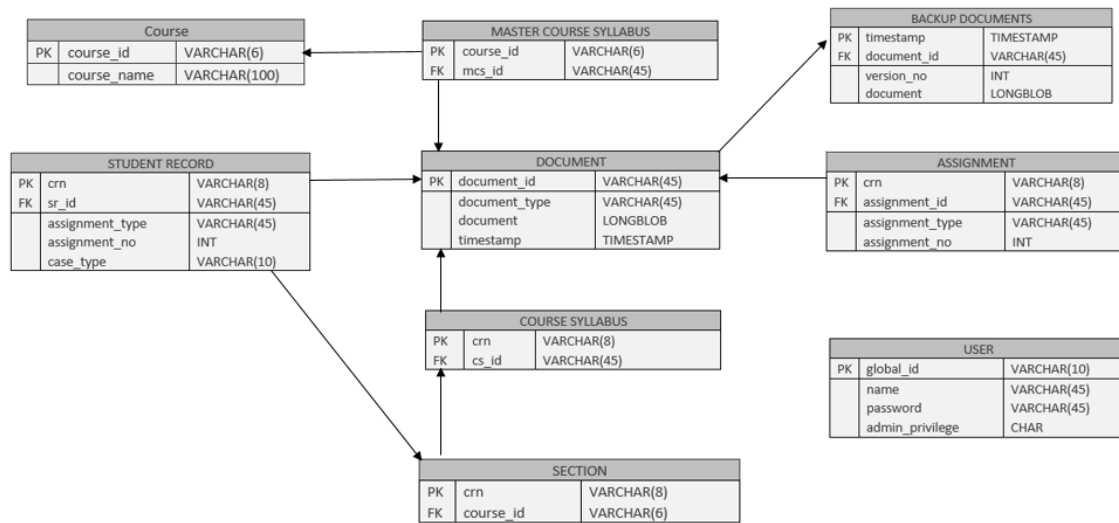


Figure 3. Database Tables

In this case, every section will have their own assignments and in-class syllabi (which are called course syllabi in this system). For every assignment in a section, best, average, and worst papers will be stored. In order to store different sections of a class separate *section* table is created. The section table has course id and crn variables. A crn is composed from eight digits. Since no numeric operations will be performed on crn, it is stored as a VARCHAR. Crn is the primary key of section table and foreign key course_id refers to the course table.

The *mastercoursesyllabus* table does not include the document itself, but it includes the course_id which the document belongs to and the unique document id (mcs_id) which will be assigned to the document. Similarly, *coursesyllabus* table includes crn and cs_id.

The *assignment* table has crn, assignment_type, assignment_no and assignment_id. Crn, assignment_type and assignment_no together is the primary key, and the assignment_id is a reference to the document table. Student Records table uses crn, assignment_type, assignment_no and the case_type as the primary key. Sr_id is the unique document id which will be assigned to the document.

The *document* table contains the document, document_id, document_type (student records, course syllabus etc.),and the timestamp. Document_id is the primary key. When a user uploads a new document, the system will generate a new time stamp and an id. The *backup_documents* table contains documents with their document_ids and version numbers. Every time a user wants to perform an update or delete operation, the system will store the document in the *backup_documents* table with a new version number and a time stamp. Even though the document_ids are unique in the meta data and the *document* tables, they are not unique in *backup_documents* table since there will be many versions of a particular document_id. This is why the time stamp is chosen to be the primary key.

Administrative privileges

This system deals with storing and versioning departmental documents in which data integrity is very important. Some operations can be performed only by the administrator. This is why the system must authenticate the user. Administrator can give administrative privilege to the other users when a new user is created, or this privilege can be assigned later with *update user* functionality.

Java

This system is developed using Java. The classes discussed in the System Interaction section have been developed using Java classes and interfaces to create MVC framework. Factory and Abstract Factory design patterns are implemented.

Entering and General Use of the System

When user starts the system, a database connection is established with local database server and a root user account. If the connection is successful, the DatabaseConnectivity.java class returns a Connection object and a welcome message is displayed. A connection object is used in every data

access class, and how it is used will be explained later in this chapter. In the `MainController.java` class, `login()` method is called and user is asked to enter his/her `global_id`. `isUserInDB(String global_id)` method in the `SystemModel.java` class, calls `find()` method from the `UserDA.java` class. This method returns a `User` object which includes administrative privilege attribute. If login is successful, in `MainController.java`, the user object is passed into the `setMenu()` method of the `MenuModel.java` class. If the user has admin privileges, the system will display the admin menu (see Figure ??); if user doesn't have admin privileges, then the system will display the regular user menu (see Chapter ??).

After successful login, the system displays the menu. The menu contains the module names and module numbers. Modules are the set of operations related to a class. For example, the *Class Operations* module, includes add class, delete class, update class and show class items. Class operations module number is 20, and the function numbers of the items in this module are consecutive (21, 22, 23 etc.). `askModule()` method will be called from the `menuViewer` object, and user will be asked to choose a module number. `MenuPrinter.runUserInput(int moduleNumber)` method will print the functions attached to the chosen module. After this, `menuViewer.askItem()` method will be called. This method returns the function number, entered by the user. If user enters -1 for module number or function number at any point, the system will call `DatabaseConnectivity.terminate()` method. The connection will be closed and program will be terminated.

Operations on the User, Class and Section Classes

The items in modules, represent the database interaction of the system for that class. For example, in the *Class Operations* module, user has add new class, delete or update a class and show class information items. Database interaction is happened only through the data access (DA) classes. Some of the items require to run couple DA methods. For the sake of creating good software architec-

ture, we considered SOLID design paradigm's single responsibility principle while creating the DA methods. When user wants to add a new class to the database, the system will ask the `class_id` and the `class_name` with the `askClassId()` and `askClassName()` methods of the `ClassViewer` class. Before adding the class, the system will make sure that, that class is not in the database. To do so, it will use `ClassDA.find(String class_id)` method. If this method turns false, then `ClassDA.add(ClassModel aModel)` method will be called in order to add the new class to the database. Otherwise, a `String` object in the `Model` class will be set to an error message, and `returnMessage()` method will pass this object to viewer, to display on the console.

At the early stages of the design, it seemed like only a small number of functions would be performed on couple different classes over and over. But soon after we start, we realized that not functions but the functionalities are the same. *Add new* functionality takes different and different numbers of parameters in different classes. One of our design objectives, keeping the controller short and clean, enforced us to build a good structure. To avoid long control statement blocks in the controller, we stored functions in the hashmaps in the models and viewers (see Chapter 4). There is one `operationsMap<Integer, MapFunctionInterface>` map object and a `userInputs` map object in `classViewer`. `askClassId()`, `askClassName()` and `setMap()` are the private methods of the `ClassViewer.java`. `setMap()` method, sets a hash map which stores functions with using the function numbers. Every object in the map is stored as a key-value pair in the `userInputs`. `userInputs` stores the values returned from the private methods. Using a hash map for this object is a good idea, because when it is passed to the model, we don't have to follow the order of elements since every element is called with its key.

In the controller, instead of calling an extensive control block, we call this line only once for every operation set:

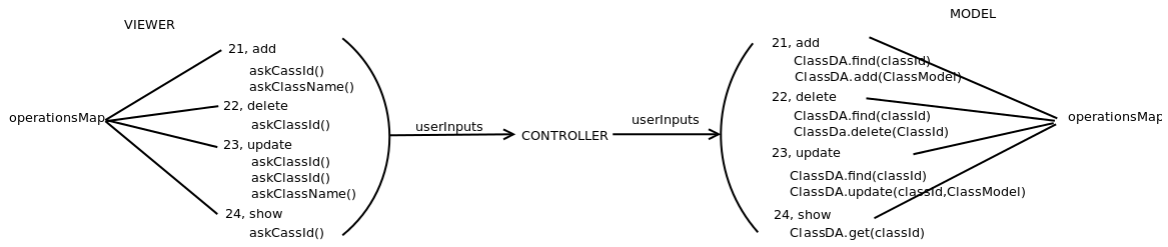


Figure 4. Class viewer, controller and Class Model structure

```
classModel.runInput(functionNo, classViewer.inputs)
```

This model is the backbone of the whole system.

Operations on the Document Classes

The most important thing about the documents in the system is the document key generation. The system generates a key, every time a user uploads a file. The key generator must generate a unique value, but it also must generate the same value for a document with the same attributes. We utilized Java's *hashCode()* function. In order to generate a unique and a consistent key, we applied *hashCode()* with the primary keys of the document classes. For example, *class_id.hashCode()* is used to generate master course syllabi keys and *crn.hashCode()* is used to generate course syllabi keys. As we stated before, there are meta-data tables (*assignment*, *studentrecord*, *coursesyllabus* and *mastercoursesyllabus*) and document tables (*document*, *backup_documents*) in the database. Meta-data tables are used to store the information about a document and the generated key, but not the document. When a user wants to add/upload a course assignment, he/she is asked to enter:

- crn
- assignment type (Homework, Project assignment, Lab assignment, Quiz, Midterm exam or Final exam)

- assignment number
- document name

The first three attributes are used to generate the *assignment_id*. Crn, assignment type, assignment number and assignment_id are stored in the *assignment* table. The document name is used to create a document instance from the DocumentModel.java class. *setFis()* method uses file name to create a FileOutputStream object, and this object is used to set a BLOB (Binary Large Object) object. The document object is stored in the *document* table.

Files are stored as BLOB objects in the database. A BLOB object can hold a variable amount of data. The four BLOB types are TINYBLOB, BLOB, MEDIUMBLOB and LONGBLOB. We store documents as LONGBLOB objects. *setBlob()* method takes a FileOutputStream as a parameter. We use the FileOutputStream object that is created by the *setFis()* method.

For every other document operation, the system will use the document key. To update or delete a document, the system will ask user to enter primary key values. If the update operation is selected; user will be asked to enter a document name unlike the delete operation. Then, a backup document object is created from the BackupDocumentModel.java class. This object has the same four attributes with the backup_documents table in the database. The system searches the document table with the backup document's id, and retrieves the BLOB object in order to create the backup document. After this, the system searches the backup_documents table with the document id to find the latest version number of that document. Eventually, the backup document object is stored in the backup_documents table. If delete operation is selected, the tuple with the document_id is dropped; If update operation is selected, only the document and timestamp columns are updated. Show function displays the document type, and the time stamp from the document table - which only consists the

latest version; and if there are any versions in the *backupdocuments* table, version numbers with the time stamps are displayed. For download document function, just like any other document function, the system generates the document_id with the user entries. The document_id, is used to get the BLOB object from the document or backup_documents tables. As we stated above in this chapter, BLOB is a binary object. To create a file from this large binary object; first, we create a binary stream then use this binary stream (with *blob.getBinaryStream()*) to create a *FileOutputStream*.

JDBC

When the user logs into the system, *DatabaseConnectivity.initialize()* method returns a *Connection* object. This object is passed into the every DA class via their constructors.

Data Access Classes

The fundamental purpose of a DA class is to provide methods that store and retrieve data and make instances of problem domain class persistent. As we spoke in Object-Oriented Development chapter, three-tier OO design model three categories of classes: interface classes for data input and display, project domain classes that model the essential entities, and DA classes that provide data storage and retrieval. Data input and output code is isolated from the other classes, which reduces the maintenance. DA methods are invoked only by the model classes. There are two types of methods in Java: static and non-static. Static methods are invoked using the class name. Since there will be no instances of the DA classes, all DA methods will be static. Constructors of DA classes are used only to pass along a *Connection* object. Five basic tasks are provided by a DA class: find, add, change, delete, get. In order to perform each task, we will generate SQL queries with using the objects passed from the model classes.

SQL Statements

The Structured Query Language (SQL) is a standardized language used to access relational databases. The five methods mentioned in Data Access Classes, invoke methods in the `Statement.java` class (from `java.sql` package) to execute the SQL statements. *SectionDA.find()* method uses the SQL `SELECT` statement to retrieve a specific section's record from the database; or *DocumentDA.retrieve()* method uses SQL `SELECT` statement to download a document from the database.

```
String sqlQuery = "SELECT * FROM section where crn = '" + crn + "'";  
  
String getFromDocument = "SELECT document FROM document WHERE d_id = '" +  
documentId + "'";
```

The add method uses SQL `INSERT` statement.

```
String sqlInsert = "INSERT INTO section (c_id, crn) VALUES ('" + c_id + "',  
'" + crn + "')";
```

`c_id` and `crn` variables are passed in the `Model` class, when the add method is invoked. Similarly to find and add, delete and update methods use SQL `DELETE` and `UPDATE` statements. *createStatement()* method in `Connection.java` class returns a statement instance, which is used to execute SQL statements. `Statement.java` class has *executeQuery()* method for SQL `INSERT` statement, and *executeUpdate()* for SQL `INSERT`, `UPDATE`, and `DELETE` statements. Sometimes it is more convenient to use a *PreparedStatement* object for sending SQL statements to the database. This type of statement is derived from the more general class, `Statement.java`. *PreparedStatement* objects can be used for SQL statements with no parameters, especially for SQL statements that take parameters. The advantage of using SQL statements that take parameters is that we can use the same statement and supply it with

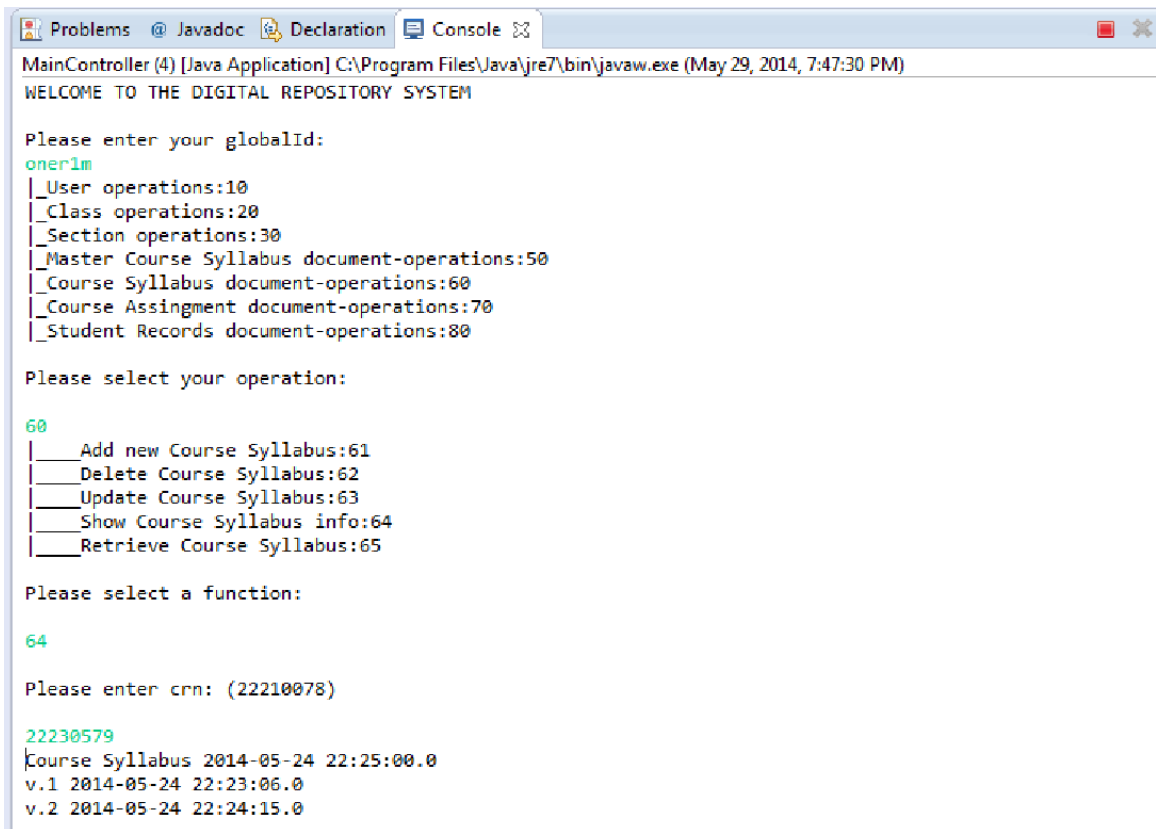
different values each time we execute it. Examples of this are in the meta data and document DA classes. Let's look at update() method in AssignmentDA.java class:

```
String updateDocumentTable = "UPDATE document SET document = ?, timestamp  
= ? WHERE d_id = ?";  
  
PreparedStatement pstmt2 = aConnection.prepareStatement(updateDocumentTable);  
  
pstmt2.setBlob(1, doc.getFis(), doc.getDocumentLength());  
  
pstmt2.setTimestamp(2, doc.getTimestamp());  
  
pstmt2.setString(3, doc.getDocumentId());
```

Result set

A *ResultSet* consists of records. Each records contains a set of columns. Each record contain the same amount of columns, although not all columns may have a value. *ResultSet* is created by executing the *Statement* or *PreparedStatement* objects we described in the previous subsection. The *DocumentDA.retrieve()* method and *showInfo()* methods in different DA classes use *ResultSet*. When user wants to find all the versions of a course syllabus document (see Figure 5), since there will be more than one records in the result set, we iterate the result set. To iterate the result set we use the *next()* method, which returns true/false.

```
ResultSet rs = aStatement.executeQuery(sqlQuery);  
  
boolean hasMore = rs.next();  
  
while (hasMore)  
{  
    crn = rs.getString(2); hasMore = rs.next();  
}
```

```
MainController (4) [Java Application] C:\Program Files\Java\jre7\bin\javaw.exe (May 29, 2014, 7:47:30 PM)
WELCOME TO THE DIGITAL REPOSITORY SYSTEM

Please enter your globalId:
oner1m
|_User operations:10
|_Class operations:20
|_Section operations:30
|_Master Course Syllabus document-operations:50
|_Course Syllabus document-operations:60
|_Course Assingment document-operations:70
|_Student Records document-operations:80

Please select your operation:
60
|_Add new Course Syllabus:61
|_Delete Course Syllabus:62
|_Update Course Syllabus:63
|_Show Course Syllabus info:64
|_Retrieve Course Syllabus:65

Please select a function:
64

Please enter crn: (22210078)
22230579
Course Syllabus 2014-05-24 22:25:00.0
v.1 2014-05-24 22:23:06.0
v.2 2014-05-24 22:24:15.0
```

Figure 5. Displaying document version information

Transactions

The most important thing about document operations methods was to use transactions. Transactions are used when you do not want one statement to take effect unless another one completes. As we described earlier, when we invoke *add()* method from a meta data class (for example *ClassSyllabusDA.java*), the system perfoms the addition task on two different tables in the database; *coursesyllabus* and *document*. The records must be added into *coursesyllabus* and the *document* tables at the same time; otherwise, the data will be inconsistent (see Figure 6). The *coursesyllabus* table will be updated, it will show a record for a document which does not exist in the document table. The way to be sure that either both actions occur or neither action occurs is to use a transaction. A transaction is a set of one or more statements that is executed as a unit, so either all of the statements are executed,

or none of the statements is executed.

```
MainController (4) [Java Application] C:\Program Files\Java\jre7\bin\javaw.exe (May 29, 2014, 8:31:10 PM)
WELCOME TO THE DIGITAL REPOSITORY SYSTEM

Please enter your globalId:
oner1m
|_User operations:10
|_Class operations:20
|_Section operations:30
|_Master Course Syllabus document-operations:50
|_Course Syllabus document-operations:60
|_Course Assingment document-operations:70
|_Student Records document-operations:80

Please select your operation:

50
|_Add new Master Course Syllabus:51
|_Delete Master Course Syllabus:52
|_Update Master Course Syllabus:53
|_Show Master Course Syllabus info:54
|_Retrieve Master Course Syllabus:55

Please select a function:

54

Please enter class Id: (CPS100)

cps100
Master Course Syllabus 2014-05-24 18:25:54.0
v.0 2014-05-21 21:48:14.0
v.1 2014-05-21 21:54:41.0
v.1 2014-05-22 21:18:09.0
v.1 2014-05-22 21:26:38.0
v.2 2014-05-22 21:33:29.0
```

Figure 6. Example of data inconsistency in the database before using transactions

When a connection is created, it is in auto-commit mode. This means that each individual SQL statement is treated as a transaction and is automatically committed right after executed. To allow two or more statements to be grouped into a transaction is to disable the auto-commit mode with:

```
aConnection.setAutoCommit (false);
```

After the auto-commit is disabled, no SQL statements are committed until the *commit()* method is called explicitly. We use a try-catch-finally block for applying transactions. In the catch block, we call *rollback()* method. This means that if a problem occurs and transactions can not be completed, system goes back to its previous state. It is important to remember to set auto-commit to true in the finally block.

CHAPTER VI

CONCLUSION

The Digital Repository System provides a convenient way to handle documents. Using this system will save CMU's Computer Science Department from spending a lot of time and effort to organize, manage and keep track of the documents. This way, instructors and staff will be able to focus on their jobs. Another advantage of the digital repository system is that it will minimize the physical storage space required to store the growing physical volumes of paper, and enable one to access data remotely. It will also reduce the risk of losing documents and the extra effort caused by organizing all the paper documents.

Future Work

The Digital Repository System is an initial prototype. Even though it is very easy to use, it is lacking a nice graphical user interface. A GUI addition will make the user experience much better. Even though the system is ready to be used by our department, it has not been tested for storing an extensive amount of documents or running the system for a long period of time. Also, currently it is not open to use of faculty members.

If the department would like to carry the system into Web in the future, using a high-level Web framework, such as Django, would be very convenient.

REFERENCES

- [1] J. Thompson. Q & a: What is a registry/repository, and who should consider one? http://www.gartner.com/it/content/754400/754413/qa_what_is_a_registry.pdf, 9 November 2007.