CSSE DOCUMENT MANAGEMENT SYSTEM: IMPLEMENTATION AND

USABILITY EVALUATION

Except where reference is made to the work of others, the work described in this thesis is my own or was done in collaboration with my advisory committee. This thesis does not include proprietary or classified information.

_____

Chao Wang

Certificate of Approval:

_____	_____
Gerry V. Dozier, Ph.D.	Juan E. Gilbert, Ph.D. Chair
Associate Professor	Associate Professor
Computer Science & Software Engineering	Computer Science & Software Engineering
_____	_____
Cheryl Seals, Ph.D.	Stephen L. McFarland, Ph.D.
Assistant Professor	Acting Dean
Computer Science & Software Engineering	Graduate School

CSSE DOCUMENT MANAGEMENT SYSTEM: IMPLEMENTATION AND USABILITY EVALUATION

Chao Wang

A Thesis

Submitted to

the Graduate Faculty of

Auburn University

in Partial Fulfillment of

the Requirements for the

Degree of

Master of Science

Auburn, Alabama
August 7, 2006

CSSE DOCUMENT MANAGEMENT SYSTEM: IMPLEMENTATION AND USABILITY EVALUATION

Chao Wang

_____
Signature of Author

_____
Date of Graduation

VITA

Chao Wang, son of Fuchu Wang and Nanying Li, was born on December 31, 1969, in Hunan province, P. R. China. He graduated from Beijing University of Aeronautics and Astronautics with a Bachelor of Engineering degree in Aeroengine Engineering 1992. He worked as an Assistant Engineer and Engineer at Beijing University of Aeronautics and Astronautics until the end of 1999. From year 2000 on, he began to work as a programmer at Aerospace Network at the University of North Dakota. He was certified as an Oracle Certified Professional in year 2004. Chao Wang began his study at Auburn University in 2004. His major research interest is in Database Management Systems and software architecture.

THESIS ABSTRACT

CSSE DOCUMENT MANAGEMENT SYSTEM: IMPLEMENTATION AND USABILITY EVALUATION

Chao Wang

Master of Science, August 7, 2006
(B.S., Beijing University of Aeronautics and Astronautics, 1992)

92 Typed Pages

Directed by Juan E. Gilbert

The design and implementation of an application like the CSSE Document Management System (CSSEDMS) involves many considerations. Among all these considerations, usability is the key for success. An application is useless if it is not usable to its users. Further more, if users don't like your application, your project fails. "If you could build a system that resulted in world peace, but no one could use it … it would be useless. Usability matters." [3] The advanced algorithms and the most efficient calculation used in your application are just not as important as they seem to be. Usable, maintainable and flexible applications are what users are seeking.

Computer Science and Software Engineering Document Management System (CSSEDMS) is a web application designed for all CSSE faculty and students. It acts as a central repository for all CSSE digital reports, theses and dissertations. Through

CSSEDMS, CSSE faculty and students can upload their documents, search and browse documents in the repository. Certainly, the search and browse functionality is available to all Internet users. In general, CSSEDMS is a typical web application targeted to CSSE faculty and students. Guided by usability evaluation and modern software architecture ideas, this thesis work shows the practices to design and implement such a typical web application to meet user requirements and satisfaction.

Web application usability is an evolving issue. As we learn more and more about how our users interact with web application, how our users retrieve information and use it, what our users anticipate within their realm of experiences, we can iteratively improve our design and implementation. This is the best practice to meet user requirements and satisfaction. Specifically, for web application usability evaluation, we need to measure users' anticipation to interact with web applications and how users actually interact with web applications. It is this anticipation (logic and intuition of users) related to web experiences that decides the usability of a web application. This thesis details the process of usability evaluation and the techniques used to quantify users' response and behavior.

From an architectural point of view, CSSEDMS is a multi-tier web application. As for most modern applications, the emphasis of the architectural design revolves around the business and data layers. For architectural considerations, there are not right or wrong answers, only the tradeoffs. Based on requirements of the CSSE Document Management System, CSSEDMS design is a data-centric procedural design. All programming logic centers around data manipulation: retrieving data from the data store, displaying data to users and updating data back to the data store. Compared with full object-oriented design, data-centric procedural design is a dirty and quick solution. Full

object-oriented design demands much more investment. Data-centric procedural design is more suitable for CSSEDMS specifications and requirements.

ACKNOWLEDGEMENTS

All the work completed in this thesis benefits from the discussion with Dr. Gilbert. The author would like to express his special gratitude to Dr. Gilbert, associate professor, Department of Computer Science and Software Engineering, for his instruction, guidance, encouragement and patience in the completion of the thesis. In particular, his suggestions, criticism and general exchange of ideas contributed much to this thesis.

He would like to express his thanks to his advisory committee members, Dr. Seals and Dr. Dozier, and the professors and staff members in the Department of Computer Science and Software Engineering at Auburn University for their kindness and help throughout these two years. Thanks to Priyanka Gupta, Kenneth Rouse, Idongesit Mkpong-Ruffin and David Thornton for their help to improve my thesis.

Thanks to his wife, Linxiang Zhu, who gives her biggest support and encouragement to help the author finish whole thesis work. Also the author wants to thank his family members, who provide the inspiration and support during the past ten years.

Style manual or journal used: Guide to preparation and Submission of Thesis and Dissertation

Computer software used: MS Word 2003, TECPLOT 9.0

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1 INTRODUCTION AND BACKGROUND

1.1 Introduction and Background

The design and implementation goal of the Computer Science Software and Engineering Document Management System (CSSEDMS) is to develop a usable, maintainable and flexible application. In order to achieve this goal, usability and application architectural ideas are used through out the whole design and implementation process.

CSSEDMS will be used to manage all reports, theses and dissertations in the Computer Science and Software Engineering department. It is going to act as a central repository for all documents from the Computer Science and Software Engineering Department. According to the requirements of such a central repository, CSSEDMS should have following the functionalities:

- CSSE students and faculty can upload their digital documents into the repository. The uploading function will only be available to users at Auburn University who are associated in some way with the Computer Science and Software Engineering department. All other Internet users can browse/search documents in the repository that are available to the public, but can not upload any documents.
- Administrative personnel can manage uploaded documents from CSSEDMS. They can approve an uploaded document by making it available to all Internet users.

Before approving a document, administrative personnel can review the information about the document and the digital document itself. They can also delete/remove pending documents from the repository/database if the documents fail to meet CSSE departmental standard. However, once a document is approved and made public to all Internet users, the administrative personnel will not be able to make further changes. In addition, several administrative personnel should be able to share the same username and password.

- All Internet users can search/browse documents available to the public in the repository. Internet users can search by keywords, title, and authors. Internet users can also browse documents by publication year(s). The search functionality should support *free text* Search, which means that users can input any set of words, any set of phrase or even a sentence for searching. The search functionality should return brief information about documents, such as authors, title, publication date, etc. It also should return ranking information for each record (document) returned. After reviewing the information about a document, users should be able to view or download the document itself.

- The Document Management System should be easy to use and maintain. Additionally, the Document Management System should merge seamlessly with current CSSE departmental Web site. That is, the colors and styles of the CSSE Document Management System should match current CSSE Web site pages.

Based on above functionalities, several key design issues must be solved. First, which kind of authentication method should be used? Should CSSEDMS use only one

authentication method or more? In other words, is one authentication method enough for the requirements? Secondly, how should the search functionality that supports inflectional forms of verbs and nouns be implemented? Finally, how should usability evaluation be performed to make CSSEDMS usable? What are the steps and process for usability evaluation? The following sections will analyze these issues respectively and give corresponding background information.

1.2 User Authentication Issues

1.2.1 Authenticate Uploading Users

Broadly speaking, there are two kinds of authentication methods. One kind must be associated with computer system user accounts. The other kind may or may not be associated with computer system user accounts. In the Windows world, Windows based authentication belongs to the first kind. Form based authentication belongs to the later – it doesn't depend on computer system accounts.

In order to make uploading functionality only available to users at the Computer Science and Software Engineering department at Auburn University, CSSEDMS system should authenticate users through Auburn University user information database. In other words, CSSEDMS should use Auburn University user account information to authenticate uploading users. Auburn University user account information is available through database SCTPROD that resides at the SQL Server cluster at Auburn University. It is also available through Auburn University Active Directory. If CSSEDMS uses database SCTPROD to authenticate uploading users, Form based authentication will be

used. If CSSEDMS uses Auburn University Active Directory to authenticate uploading users, Windows based authentication is used.

According to Auburn University Office of Information Technology (OIT), user information kept in SCTPROD database in Auburn University SQL Server cluster is updated once a week by data from Auburn University mainframes. Updating is performed automatically through services offered by Auburn University OIT office. Auburn University Active Directory includes all user groups at Auburn University. As long as CSSEDMS has access to Auburn University Active Directory database, access restriction to CSSEDMS can be implemented straightforwardly.

Because of security considerations, OIT office of Auburn University only supports CSSEDMS's access to Auburn University Active Directory, not the database SCTPROD in Auburn University SQL Server cluster. Consequently, Windows based authentication is the choice available for CSSEDMS to authenticate users who want to upload documents.

1.2.2 Authenticate Administrative Personnel

In order to allow several CSSE administrative personnel the ability to share same username and password, the username and password for administrative purposes should not utilize personal Auburn University user account information. This is because Auburn University username and password are related to private information such as Auburn University email account, etc. Consequently, the authentication method used for authenticating administrative personnel should be Form based, which is independent of

Auburn University user account information. This authentication method is different from the authentication method used to authenticate uploading users, which is Windows based authentication, as discussed above.

In summery, two authentication methods are used to meet the requirements: Form based authentication method and Windows based authentication method. Form based authentication method is be used to authenticate administrative personnel. Windows based authentication method is used to authenticate users who need to upload documents into CSSEDMS.

1.3 Database Supported Search Functionality

1.3.1 Introduction

In our society digital data is exploding rapidly. Databases and digital libraries are storing more and more digital information in different formats than ever before. Databases with storage capacity of terabytes are not unusual at all. Terabytes are growing into petabytes [19]. In order to deal with large amounts of digital information, digital information searching engines have emerged as digital information prevails in our society. The first searching tool on the Internet was called “Archie”. It was created in 1990 by Alan Emtage, a student at McGill University in Montreal [21]. Ever since then the searching industry has built many big companies such as Yahoo and Google, etc. Besides the search engines used for searching the Internet, searching functionality within a web application is now a necessity in almost every single web application.

As stated in section 1.1, CSSEDMS web application needs the searching functionality to search a document by title, author and keywords. Because all information about the documents is saved into the database system, this searching functionality can be performed by a database management system. The major concern is the support for *free text* searches. *Free text* search must support a kind of linguistic search. That is, it should support inflectional forms of verbs and nouns. Database management systems do support this kind of search functionality.

1.3.2 Rich Text Search Functionality in Database Management Systems

Linguistic and proximity searches are kinds of rich text search functionality. Linguistic search is a search type that indexes and retrieves a document with terms that are reduced to linguistically basic form. For example, word “computing” is indexed as “compute”. Proximity search is used for searching two or more words that occur within a specified number of words. Proximity searching is normally used with a keyword or Boolean search. In general, rich text search needs a kind of fuzzy pattern matching, not the exact pattern matching.

Relational database management systems traditionally have had limited capabilities for finding fuzzy patterns in textual data. The traditional method is to use SQL SELECT clause to retrieve text based on exact pattern matching. The SQL-92 standard defines only basic string-search capabilities:

- For a string value equal to, less than, or greater than another string.

- For a string containing an exact string pattern.

Apparently, this string search capabilities can not handle fuzzy searches that need to look up words and phrases in inflectional forms or linguistic similarity to one another.

Computer digital data exists in two forms: structured and unstructured. Structured data has an enforced composition to the atomic data types. Structured data is normally managed by database systems that allow query against predetermined data types and relationships. In unstructured data, there is no conceptual definition or data type definition. Textual document is a good example of unstructured data. In textual documents, a word is simply a word. There is not any data type or relationship meaning attached to it. SQL-92 SQL SELECT clause is used for structured data saved in database management system. It can not effectively deal with unstructured data. In reality, data in structured form only represents a fraction of corporation data. The majority of digital data is stored as unstructured data primarily as text in database columns (as char, varchar or text type) and in the file systems such as web pages, manuals, reports, email, faxes etc. James Hamilton (Architect, SQL Server Programming) believes that only about five percent of data in this world is structured [5]. Pat Selinger (IBM Fellow & VP Data Management Architecture & Technology) believes that less than 15 percent of world's data is structured [8]. In order to deal with large amount of unstructured data, most modern popular database management systems support rich text search by offering indexing and querying support for plain text contents. Indexes of plain text content are different from normal database table indexes. While database table indexes are normally implemented using B+ tree data structure, indexes of plain text content need an indexing

component. It normally includes Filters, Sectioners, Lexes, and Indexing Engine. The overall architecture is shown as in Figure 1-1.

Oracle database management system offers a set of searching functionalities for unstructured data. Oracle Text is a complete text search solution. Previously called InterMedia Text, Oracle Text is a tool that enables you to build text query applications and document classification applications. Oracle Text provides indexing, word and theme searching, and viewing capabilities for text. Oracle Ultra Search includes web Search User Interface, remote crawler and can search multiple indexes. Oracle InterMedia offers content management and can search multimedia content [9].

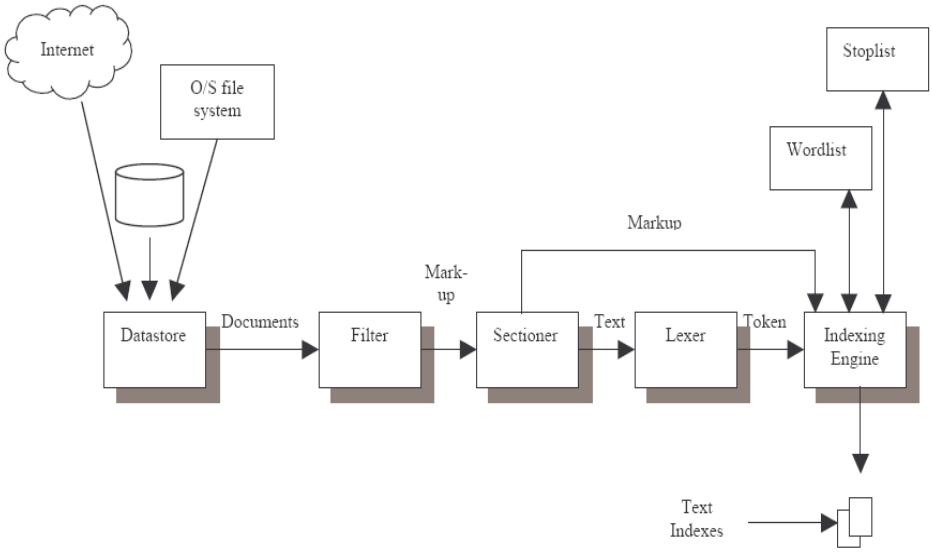

Figure 1-1 Indexing Architecture (www.oracle.com)

SQL Server 2000 offers text search functionality through Full Text Search support. Different from Oracle, SQL Server Full Text Search stores indexes and catalog

information in the file system, not in the database itself. Although full text indexes are stored in file systems, they are administered by SQL Server databases. Updating of full-text indexes can be requested through a schedule or can occur automatically when change happens.

The plain text searching functionality supported by Oracle and SQL Server and other database management systems includes two parts: the indexing part and the querying part. While the indexing part creates indexes out of plain text, the querying part offers text search constructs such as CONTAINS or FREETEXT predicates and the FREETEXTTABLE row set functions. Both Oracle and SQL Server support SQL similar queries against plain text and the creation and maintenance of underlying indexes that facilitates these kinds of queries.

SQL Server Full Text search allows full-text queries to be issued against plain text data in SQL Server tables, including words and phrases. It also supports queries against binary data type in SQL Server tables. When data in a column is a binary type, SQL Server Full-Text search uses filters to interpret the binary data. The filter extracts text out of the binary document and submits it for subsequent indexing processing and querying. Microsoft® SQL Server™ 2000 originally includes filters for these file types: .doc, .xls, .ppt, and .htm. Custom filters can be created for full-text indexing of other file types such as PDF. There is a limitation for full-text indexing a binary type file: binary contents must be less than 16 megabytes (MB) in size and returned filtered text must less than 256 kilobytes [10].

Besides the searching functionality against plain text in SQL Server tables, SQL Server also supports textual queries against data residing in the file system. Combined with Microsoft Indexing Service and SQL Server distributed queries, queries against information both in the database and in the file system can be executed. Since most unstructured information in file system is always related to structured information in database, this feature is especially useful. For example, you can query the names, sizes, and authors of all Microsoft Word files on the C drive that contains the phrase "Oracle 9i" in close proximity to text. The returned result can then be joined with the Authors table to obtain the authors' ages and addresses, etc. Oracle Ultra Search has similar search functionality.

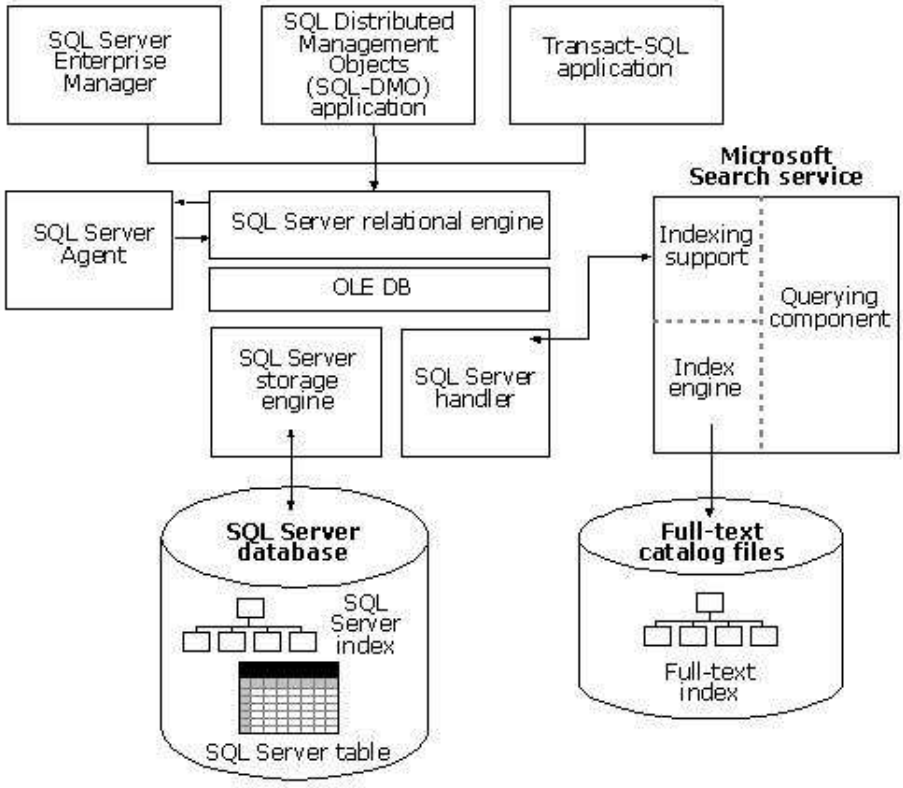

Figure 1-2 SQL Full Text Support Architecture (SQL Server Books Online)

CSSEDMS system uses Full Text search functionality from SQL Sever 2000 cluster at Auburn University. Supported by Full Text search, CSSEDMS can perform complex *free text search* – users can input any set of words, any set of phrases or even a full sentence for searching. In administration level, stored procedures are used to administrate SQL Server full text search. This includes creating catalog and full text indexes, populating the catalog and updating the full text indexes.

1.4 Usability Evaluation for CSSE Document Management System

1.4.1 Usability Evaluation Introduction

Usability is a quality attribute that assesses how easy users can use an application. Usability is normally defined by five qualities [11]:

- **Learnability**: How easy is it for users to perform basic tasks the first time they encounter the application interface?
- **Efficiency**: Once users have learned the application interface, how quickly can they perform tasks?
- **Memorability**: When users return to the application after a period of not using it, how easily can they reestablish their former proficiency?
- **Errors**: How many errors do users make, how severe are these errors, and how easily can they recover from the errors?
- **Satisfaction**: How pleasant is it to use the application?

1.4.2 Improving an Application's Usability

Improving the usability of an application/system is not easy. Best practices call for spending about 10% of an application's budget on usability evaluation. Iterative approach – usability test, make changes, and usability retest – is the best practice to improve an application's usability. It is the only effective approved way to improve the usability of an application [27].

Usability evaluation can be used in different development stages. Usability evaluation can be used for diagnosing problems, comparing alternatives and verifying if the application has met usability goals. Diagnostic usability evaluation is mainly used for finding out what is working well and what is not working well. With diagnostic usability evaluation, the development team can continue with what is working well and improve what is not working well. Because diagnostic usability evaluation can be used to diagnose usability problems, it is used more and more in the development phrases. Comparative usability evaluation is used to test alternative interface designs. The result of comparative usability evaluation can be used to determine which design is better. Best practices suggest comparing web application design with similar web application. Verification usability testing is normally used at the last stage of application development. The purpose is to make sure the design meets the preset usability goals and guarantee users' satisfaction.

The usability evaluation of the CSSE Document Management System measures three aspects of the application: effectiveness, efficiency and user satisfaction. Effectiveness – percent of correctness – is measured by calculating how many scenarios,

tasks or other types of activities that participants complete correctly. Efficiency – time to complete each scenario – is measured by recording the time needed for participants to complete scenarios and tasks. User satisfaction is a user's subjective responses to using a system. It represents users' anticipation (logic and intuition of users) of web applications.

The major part of a questionnaire used in CSSEDMS usability evaluation is used for measuring user satisfaction. Thomas S. Tullis and Jacqueline N. Stetson [12] compared different questionnaires for assessing web application usability. Five questionnaires have been compared: Questionnaire for User Interface Satisfaction (QUIS) 1988, Computer System Usability Questionnaire (CSUQ) 1995, System Usability Scale (SUS) 1996, Microsoft's "Words" and the questionnaire of their own. Their results shows that SUS – with only 10 rating scales yielded the most results across different sample sizes. SUS was also the only questionnaire whose questions address all different aspects of the user's reaction to the web application as a whole.

1.5 The Scope of This Study

Usability is essential for the design and implementation of a web application like CSSE Document Management System. It is the key for success. Architectural considerations play an important role for an application's maintainability and flexibility. The purpose of this thesis is to implement a usable, maintainable document management system using architectural ideas and usability evaluation. Because performance is not a big issue for CSSEDMS, the goal for architectural consideration is to guarantee

maintenance and flexibility, sacrificing calculation efficiency. Using the measurement of effectiveness, efficiency and user satisfaction, this thesis shows usability study practices for the fulfillment of CSSEDMS's usability requirements.

1.6 The Structure of the Thesis

The thesis consists of six chapters.

Chapter 1 introduces the CSSE Document Management System and related background information. Chapter 2 discusses design considerations of CSSE Document Management System. Following the discussion of design, Chapter 3 shows the implementation of CSSE Document Management System. Usability study of the CSSE Document Management System is introduced in Chapter 4 and Chapter 5. Chapter 6 concludes by summarizing the thesis and discussing further related work that will be carried out.

CHAPTER 2 DOCUMENT MANAGEMENT SYSTEM DESIGN

2.1 Introduction

Modern applications are normally based on three-tier (N-tier) architecture. Two tier (client/server) architecture is good for distributed computing when connections are less than two to three hundreds – on today's hardware [15]. When connections exceed two to three hundreds, the performance begins to deteriorate. It's relatively difficult for both Oracle and SQL Server to support more than three hundred concurrent users [15]. This is because the database sever needs to maintain keep-alive messages with each client. Three tier (N-tier) architecture is emerged in the 1990s to overcome the limitations of two tier architecture. The middle tier server sits between the user interface and the data source tier, running business logic and rules. Through queuing and database staging, three-tier architecture can accommodate thousands of connections. Tests carried out by Rocky Lhotka show that for up to three hundred concurrent users pounding against an application server, it just needs three to four database connections [16]. Middle tier application server improves performance, flexibility, reusability and scalability by centralizing processing logic. Middle tier application servers can also manage distributed database integrity by using two phase commit process. Besides improvements in performance, flexibility, reusability and scalability, three-tier architecture provides other advantages. First, there is a clear separation between user interface, business logic and

data access layers. This isolation promotes code re-use and makes maintaining and changing code easier. It is also easier to divide development tasks. Different developers or teams can work on different tiers (parts) of the application during implementation phase. Secondly, business logics are centralized into the middle tier, it is relatively easier to maintain at a central point. Centralization makes deployment and maintenance much easier.

Three-tier architecture functions as following. The user interface tier takes care of user input, dialog and display management. The middle tier runs the business logic and provides process management services. The middle tier also controls transactions and asynchronous queuing to guarantee successful transaction completion. Database management tier (data store) is dedicated to manage concurrent data accesses and keep data integrity.

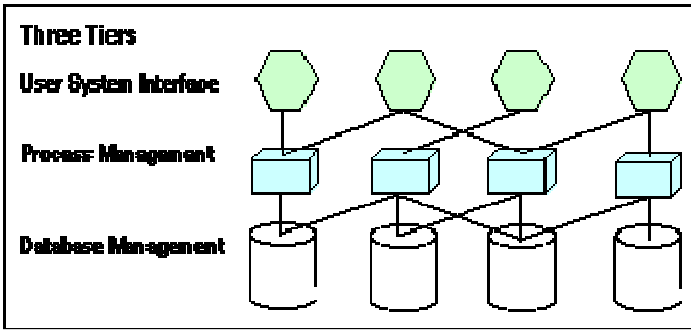

Figure 2-1 Three Tier Architecture Depiction ([13])

Business logic and data access layer reside in the middle tier. Inside the data access layer of most modern applications, there might be a further division: a high-level (close to business layer) data access level and a low level (create/read/update/delete SQL statement) data access layer. This division is based on the acknowledgement of domain

specific and non domain specific data. The high-level data access layer builds SQL statements according to specific Business Objects. It then executes the SQL statements via the low-level data access method and return Business Objects back to the Business tier. In other words, the high level data access level hide details from business object. The low level data access level takes SQL statements as input and sends them to database. Because the low level data access level deals with plain SQL statements, it is reusable across different domains.

Traditional two-tier client server architecture is also called rich client architecture. Browser based web applications are named thin client architecture. For rich client architecture, performance and deployment are a big headache. The notorious “DLL Hell” problem inflicts many servers, where one application could break another application by deploying an incompatible shared DLL. Thin client applications allow central deployment and updating. This significantly reduces the deployment and maintenance problems. However, compared with rich client applications, thin client applications have much reduced features, such as missing drag-and-drop, context sensitive help, etc. The most recent trend is smart client applications. Basically, smart client combines both rich client and thin client advantages. It can work both connected and offline. It has intelligent installation and updating. Using client side processing, smart client utilizes local resources and offers better responsiveness. Smart clients are emerging as next generation applications [20].

2.2 CSSE Document Management System Architecture

The CSSE Document Management System is an ASP.NET web application that uses a logical, multi-tier architecture. See Fig 2-2. Multi-tier architecture normally refers to the logical architecture of applications. Physically, all three tiers can be in one .exe file or .dll file. They can also be in different .exe file or .dll files. The CSSE Document Management System physically resides in one .dll file.

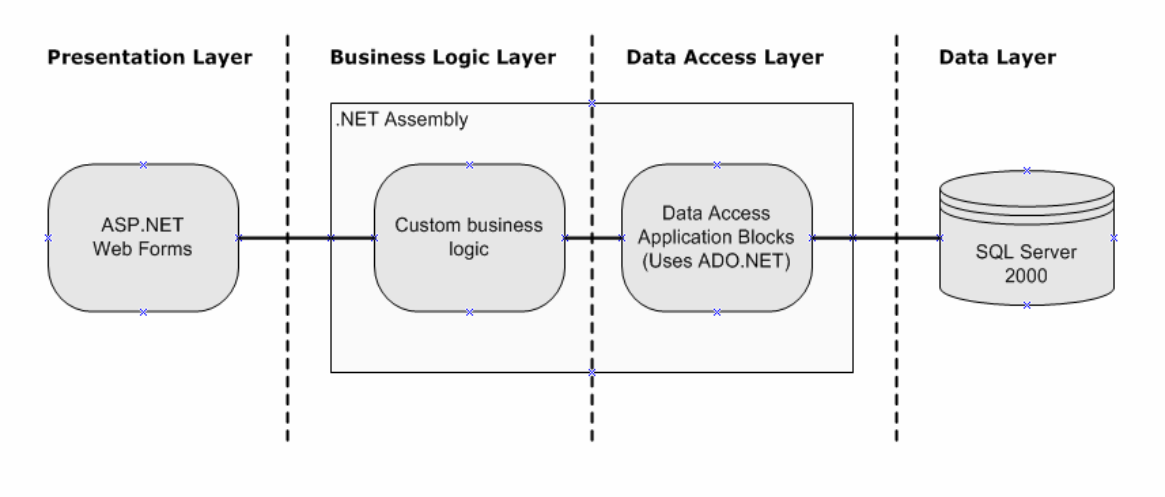

Figure 2-2 CSSE Document Management System Multi Tier Architecture

Because of expected low work load on the CSSE Document Management System, performance should not be a big issue. Maintenance and flexibility weight more than performance. Consequently, emphasis is placed on maintenance and flexibility instead of performance.

The presentation tier consists of web Forms. Web forms consist of web controls, which are dragged and dropped onto web forms. The Business tier deals with security, business rules and data processing. The CSSE Document Management System uses

stored procedures to encapsulate all the database queries and perform all data access functionalities. Both Business logic layer and Data access layer for the CSSE Document Management System reside on Auburn University fp.auburn.edu server. The data store tier consists of the CSSE database resides on Auburn University SQL Server Cluster.

At presentation layer, all web controls on web forms are wired up with events. The advantage is that the design is easy and intuitive. However, web forms contain both presentation and control logic. It is hard to maintain and extend as application becomes more complicated. As application becomes more complicated, redundant code begin to appear in different event handlers. Redundant code also appears in different web forms and hard coded logic is used to implement updates across web forms. In general, this design only offers one-way control from web forms to business tier. There is no way for Business tier to update web forms.

In order to overcome these disadvantages, some programming techniques are used to reduce redundant code – increasing reuse of code. Run time event handler adding and removing are used to propagate updates. One of the programming techniques used is to hook up different similar events to one single event handler. In this way, several events are handled by one event handler – reduces redundant code. Another programming techniques used is dynamically adding and removing multicast *delegates.* In this way, several event handlers can be wired up to one event.

Further consideration should be the separation of presentation and web form controls. Ideally, presentation only deals with display. Input processing and interface controlling are handled by web form controls. The result is an additional tier in the

system and even looser coupling. Model-View-Controller GUI design is the most popular method used in industries [20].

In CSSEDMS, the design of Business logic layer and Data access layer is a data - centric procedural design. All workflows revolve around data manipulation. For instance, the presentation layer (web forms) accepts input data. It then passes the data to Business logic for validation and other processing – according to business rules. After that, Business logic layer passes data to Data access layer, which in turn will save and update data into database.

2.3 CSSE Document Management System Database Design

The major goal of the CSSE Document Management System is to manage documents. According to the requirement analysis, there are Documents, People, DocumentTypes and Roles entities. Documents entity represents all the documents in CSSEDMS. It has DocumentID, DocumentFileType, DocumentTypeID, Title, Abstract, Document Finished Date, Document Submission Date, Approval Status, and Document – the digital file itself – properties. DocumentType represents document types in the CSSE Document Management System. It has two properties: DocumentTypeID and DocumentType. There are only three document types in the CSSE Document Management System: Report, Thesis and Dissertation. Roles entity represents all roles people play in the documents. It has RoleID and Role properties. There are eight roles people play in the CSSE documents: Major Professor, Committee Member, First Author, Second Author, Third Author, Fourth Author, Fifth Author and Sixth Author. In addition to above entities, there is a relationship table: DOCUMENTPEOPLE. It relates the

Documents entity with the People entity – the implementation of many to many relationship.

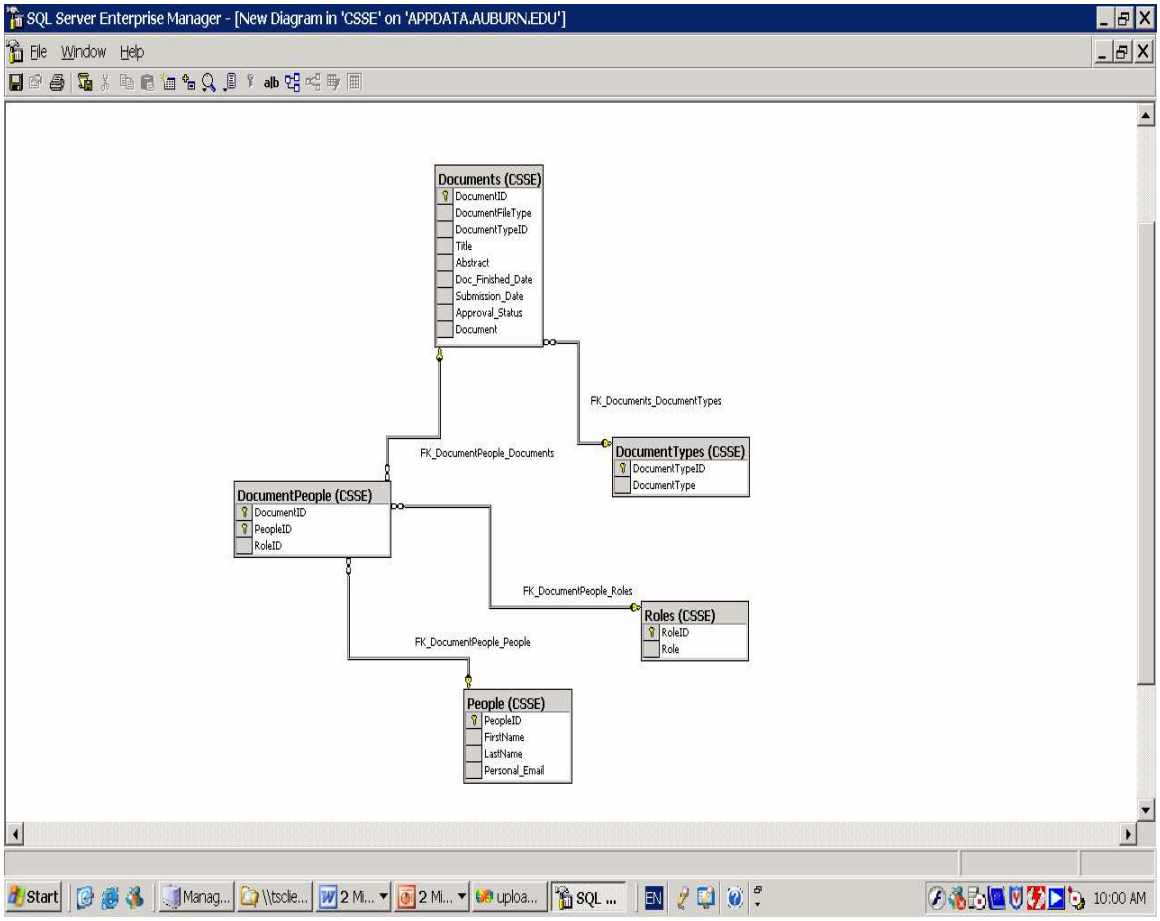

Figure 2-3 CSSE Document Management System Database Schema

A database named CSSE was created on Auburn University SQL Server Cluster to store the CSSE documents and related information. According to the database schema, DOCUMENTS, PEOPLE, ROLES, DOCUMENTTYPE and DOCUMENTPEOPLE tables are created in the CSSE database. Views are used for combining information from different tables. ALLINONE view joins DOCUMENTS, PEOPLE, DOCUMENTTYPE and ROLES table. vwSUBMISSION is a subset of ALLINONE view which returns the

documents that have been submitted and have not yet been approved by administrative personnel. ALLINONE_AUTHORS view return all authors related with one DocumentID.

2.4 CSSE Document Management System Authentication Methods

As discussed in Chapter 1, two authentication methods are used in the CSSE Document Management System: Form based authentication and Windows based authentication. Form based authentication is used to authenticate administrative users. Windows based authentication is used to authenticate users who need to upload documents.

2.4.1 Form Based Authentication

Form based authentication relies on browser cookies to carry user identity information. By default, the cookie information is encrypted when it is passed back and forth between clients (browsers) and servers. Username information and passwords are passed through HTML form to server. With form based authentication, usernames and passwords can be saved in whatever storage mechanism. Put it another way, usernames and passwords are not related with computer system username and passwords. Usernames and passwords can be retrieved from XML file, text file, binary file or database table. One of the problems with form base authentication is that the authentication ticket cookie can not be easily shared across multiple servers – server farms. This is because different servers use different validation and decryption key. In order to force multiple servers to use the same validation and decryption key, all the

servers in a farm should be configured in either web.config file or machine.config file with same validationKey and same decryptionKey. For example:

<machineKey

validationKey=”the same validation key”

decryptionKey=”the same decryption key”

validation=”SHA1” />

Form authentication in ASP.NET supports both session and persistent cookies. Session cookies are only valid for a live session. Persistent cookies reside on client machine’s hard drive. Persistent cookies can be valid for months or years. The CSSE Document Management System only uses session cookies, which is valid only for current session.

Authorization is used to determine which users can access which web pages within a directory. There are two elements in *web.config* file for restricting user’s access to web pages within a directory. <*allow*> element is used to enable user’s access. <*deny*> element is used to deny user’s access. CSSEDMS web application uses combined <*allow*> and <*deny*> elements to implement specific authorization.

2.4.2 Windows Based Authentication

Windows based authentication method is used for authenticating users who want to upload documents into CSSEDMS. Windows authentication is normally used in corporate intranet where there are already existing user accounts and groups in computer systems. The CSSE Document Management System uses Auburn University Active Directory user accounts and groups information to determine if the user logged in is

associated with Computer Science and Software Engineering department. Because Windows based authentication relates to internal computer system users and groups, Windows based authentication is not a good option for public registration system.

There are three kinds of Windows based authentication: Basic authentication, Integrated Windows authentication and Digest authentication. Both Integrated Windows authentication and Digest authentication are not compatible with non-IE browsers. In order to be compatible with Netscape browsers, the CSSE Document Management System uses Basic authentication. Basic authentication method passes usernames and passwords in plain text. Combined with SSL, Basic authentication method can guarantee required security requirements.

Auburn University Active Directory has defined different groups for use by college/school/department on campus. There are groups representing all students who are enrolled in at least one course offered by a specific department in a specific term. There is also a group for each major. COMP_Majors contains all students taking Computer Science and Software Engineering as their major. Each department has a Gid and a facultyGids group which contains all employees and faculty of that department. 00635_Comp_Sci_Faculty_Gids group contains all users in Computer Science and Software Engineering department. All these groups are maintained on a daily basis and always reflect the most updated Auburn University user information.

Based on group information in Auburn University Active Directory database, the configuration for Windows based authentication is like following:

```
<authentication mode="Windows" />
<authorization>
            <allow roles="auburn\COMP_Majors"/>
            <allow roles="auburn\00635_Comp_Sci_Faculty_Gids"/>
```

```
            <allow roles="auburn\00635_Comp_Sci_Gids"/>
            <deny users="*"/>
    </authorization>
```

The authentication section specifies that the authentication method is Windows based. The authorization section specifies that users from Computer Science and Software Engineering major, faculty and staff can access all web pages in current directory.

There is a small issue with Windows based authentication. After the authentication, Internet Explorer (IE) caches the credentials and continues sending same credentials for each subsequent request to the server until user closes browser. As long as the browser is open, the credentials are valid. Invalidating cookies will not explicit sign off a user. This is annoying because the only way to explicitly sign off a user is to clearing the IE caches. This might also cause some security problems where computers are publicly shared by many users. One of the solutions to this problem is to force users to re-authenticate after a certain period, particularly after a period of inactivity. Another way is to clear IE cached credentials after user sign off the web application. Whenever a user wants to sign off, the browser will be forced to execute:

```
// Clear current credentials
   document.execCommand(ClearAuthenticationCache, false)
```

This will clear the credentials cache for the entire iexplore.exe process. Certainly this solution has a side effect: users will lost all the credentials cached in IE and users will need to re-authenticate to any site being opened at that time.

2.5 CSSE Document Management System Design Features

From the analysis of the CSSE Document Management System, several key design features are used to meet the requirements.

Fist, the overall user interface layout is adopted from the current CSSE departmental Web site. The color and style are very similar to the departmental Web site. The CSSE Document Management System keeps all the links and javascripts rotating functions from the CSSE departmental Web site. The result is smooth switching and transferring from the CSSE departmental Web site to the CSSE Document Management System and vice versa. This layout and style similarity should also be able to increase CSSEDMS usability given that users are already familiar with the departmental Web site, especially for CSSE users.

Secondly, all the information and data is stored into Auburn University SQL Server cluster. One of the most popular designs for digital repositories is to save digital files in the web server file system. Digital file path information is saved in database tables. Whenever users perform searching function, returned result is the path of the digital file, not the digital file itself. It is the application code that uses the path information to retrieve the digital file from the server file system. The CSSE Document Management System saves all information and the digital file into the CSSE database. The searching functions directly return digital files in the CSSE database. There is no need for CSSEDMS to retrieve digital files according to path information. Additionally, as stated in Chapter 1, using filters from SQL Server, the searching functions can be performed on binary files themselves.

Compared with saving digital files on the server file system, saving files into the database system has advantages. First, saving digital documents into the database system is straight forward. Secondly, backup and restore of files can be carried out with all other database tables, that is, using database backup and restore functionalities. There is no need to backup files in the server file system.

CHAPTER3

CSSE DOCUMENT MANAGEMENT SYSTEM IMPLEMENTATION

3.1 Introduction

Developed using Visual Studio 2003, the CSSE Document Management System consists of three projects. *Admin* project contains functionalities for administrative users. *Upload* project contains functionalities for uploading users. *Cssedms* project contains functionalities for searching and browsing documents. These three projects have their own individual configuration files, which specify different authentication and authorization methods. The combination of these three projects forms one whole solution to the CSSE Document Management System requirements. The CSSE Document Management System is implemented in C# language.

3.2 *Upload* Project Implementation

The *upload* project implements functionality for users to upload documents into the CSSE database. As mentioned in Chapter1 and Chapter2, the first consideration is to restrict access only to users from the CSSE department. Windows based authentication method combined with Auburn University Active Directory user groups information prevents users not associated with CSSE department accessing the uploading functionality.

Whenever a user wants to upload a document, s/he needs to provide information about the document and the digital file itself. Information about the document includes authors, author email (for notification), document title, document abstract, document type and document finished date. See Figure3-1 for the uploading web page.

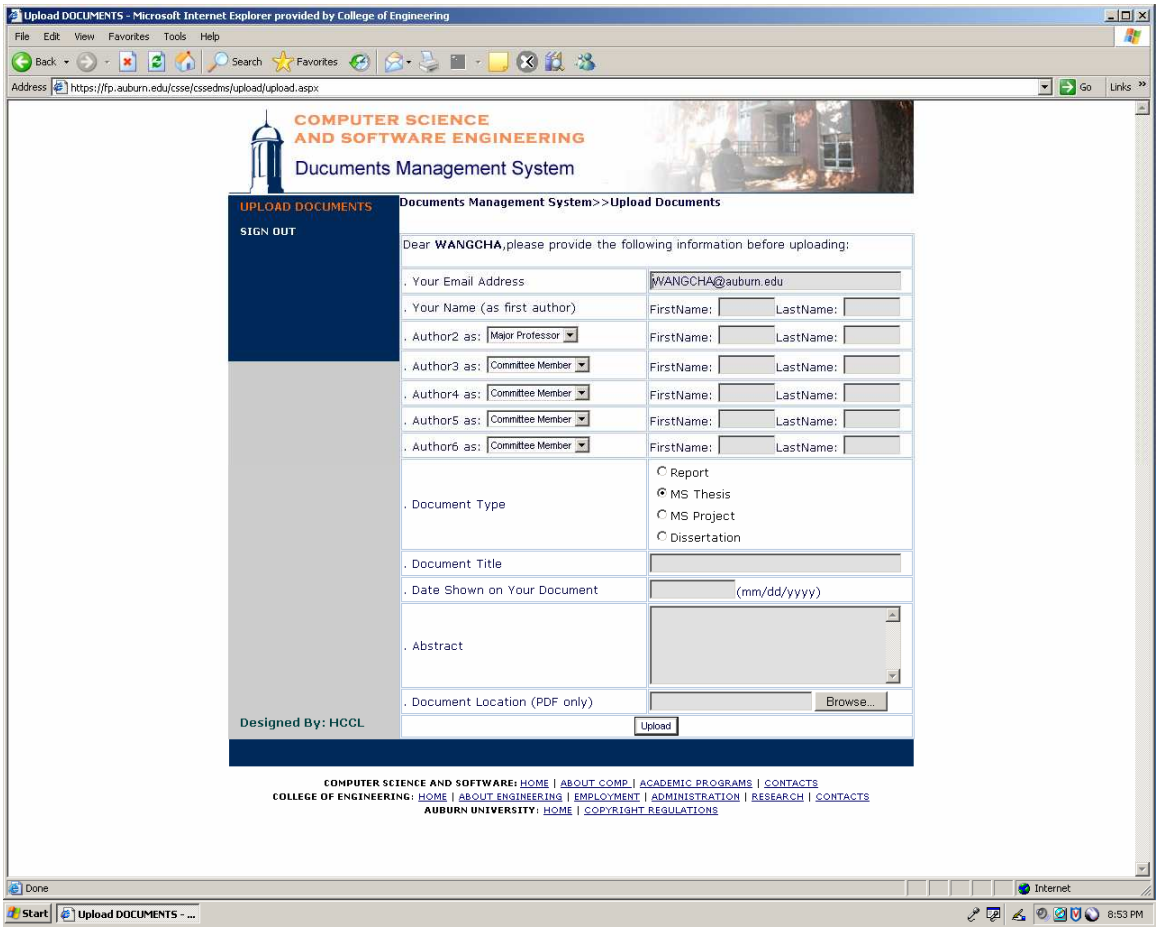

Figure 3-1 Uploading Page

All this information is actually saved into different tables in the CSSE database. Consequently the second consideration is about the insertion of data into different tables. Insertion actions to different tables should be all successful or all failed. It is not

advisable for some of the insertion to be successful and others failed, which will leave useless data inside the database. In other words, the insertion actions to different tables should be atomic. They should not be separated as different actions.

3.2.1 Inserting Different Tables in One Transaction

Executing insertion actions to different tables in one transaction is a way to make these actions atomic. A transaction is a series of operations performed as a single unit of work. Put all insertion actions into one transaction will ensure the consistency and reliability of the system despite any errors that might occur. All insertion actions in one transaction must complete successfully in order to make the transaction successful.

Microsoft .NET framework distinguishes local and distributed transactions. Local transactions have a single transaction-aware data resource. For example, SQL Server database system is a single transaction-aware resource enforces ACID property for a transaction. It can commit a transaction in the case of success or rollback a transaction in the case of failure. Distributed transactions can span different data resources. Microsoft Distributed Transaction Coordinator is used to coordinates commit and rollback actions across different data resources. A two-phase commit protocol is used to ensure the distributed transaction consistency.

In CSSEDMS, there is only one data resource, all transactions are local. There are several ways to implement these local transactions needed for insertion actions. One way is to use SQL Server database transaction. SQL Server database has BEGIN TRANSACTION and COMMIT/ROLLBACK TRANSACTION statements to wrap up different operations into one transaction. The disadvantage is that all the controls must be

written in T-SQL, which is basically a SQL language expanded with some primitive programming controls. Another way is to use automatic transactions from .NET framework. COM+ provides a programming model based on declarative transactions. ASP.NET pages, Web Service and any classes defined in the program can be declared to be transactional by using a declarative transaction attribute. For example, ASP.NET web pages can be declared to be transactional by using page attribute:

<@ Page Transaction="Required">

Once an ASP.NET page is declared as transactional, all actions within this page belongs to one transaction. ADO.NET also provides transaction support. ADO.NET provides the way to create a connection to the database, begin a transaction, commit or rollback a transaction and then close the connection. All the steps are carried out by a set of ADO.NET objects. The procedure is as following: first create a *SQLTransaction* object, begin the transaction using a *SQLConnection* object. Then execute all the operations inside this transaction. If all operations are successful, you can commit changes to the database by using the Commit method. Otherwise, you can use the Rollback method of the *SQLTransation* object to roll back the transaction.

The CSSE Document Management System uses the ADO.NET supported transaction to insert data into different tables. The reason for taking this method is that all codes can be written in C#, there is no need for writing T-SQL supported transaction. Transactions supported by ADO.NET use more system resources than transactions supported by database but less system resources than page level transactions.

The actual implementation in C# looks like following:

```
SqlConnection Conn = new
SqlConnection(ConfigurationSettings.AppSettings["ConnectionString"
]);
Conn.Open();
SqlCommand cmdInsert1 = Conn.CreateCommand();
SqlCommand cmdInsert2 = Conn.CreateCommand();
SqlCommand cmdInsert3 = Conn.CreateCommand();
SqlTransaction InsertTrans;

// Start a local transaction
InsertTrans = Conn.BeginTransaction();

try
{
      cmdInsert1.CommandText = "insertDocuments";

      …

      InsertTrans.Commit();

}
catch(Exception e1)
{

      try
      {
            InsertTrans.Rollback();
      }
      catch (SqlException ex)
      {
            if (InsertTrans.Connection != null)
            {
                  InsertTrans.Close();
            }
      }
      …

}
```

After a document is uploaded into the CSSE Document Management System, the system should notify the user of his/her uploading status. This is implemented by sending email to the user.

ASP.NET offers strong support for sending emails. To use ASP.NET supported email functionality, the server should have enabled SMTP service. SMTP Service offers

basic email sending and retrieving functionality according to the Simple Mail Transport Protocol (SMTP). SMTP service doesn't support the Post Office Protocol. It doesn't support multiple mailboxes for multiple users. It is impossible to retrieve email using an email client like Outlook from SMTP service. On a Windows server, the SMTP service uses two directories to process email: Pickup and Drop. SMTP service works by constantly polling the Pickup directory for new files written into the directory. If it finds a new file, it attempts to send it. On the other hand, when new emails come into the SMTP service from network, they will be saved in the Drop directory. Compared with full email system such as Exchange, the functionality offered by SMTP service is limited. However, it is sufficient for CSSEDMS. Obviously, one way to send email is to write email message as a file directly into the Pickup directory on server. If this method is used, Network Service (Named ASP.NET work process in .NET framework 1.0) should be assigned writing privilege. From the security point of view, this should not happen.

After setting up SMTP service, ASP.NET can send email through a couple of classes: the *SmtpMail* class and the *MailMessage* class. Both of them belong to *System.Web.Mail* namespace. *MailMessage* represents email messages that can be sent out using the *SmtpMail* class. The *SmtpMail* class actually performs the sending action. The *MailMessage* has many properties for specifying the email itself. For example, the *Attachments* property is a collection representing all files attached to the email message.

After a user uploads a document, the CSSE Document Management System uses *SmtpMail* and *MailMessage* to send out email to the user. The implementation is similar to the following:

```
MailMessage mailObj=new MailMessage();
mailObj.From="CSSEDMS@eng.auburn.edu";
mailObj.Subject="Thank you”;
…
mailObj.BodyFormat = MailFormat.Html;
mailObj.To = emailAddress;
try
{
        SmtpMail.Send(mailObj);
}catch (Exception ex){
        Label2.Text="Email sending error.";
}
```

3.2.2 Retaining Form Data

If a user wants to upload additional documents, the Document management system should be able to go back to original uploading page with the data in the form unchanged. There are two ways to keep original data in the uploading page. One way is to use *Server.Transfer()* to redirect http request. Different from *Response.Redirect()*, *Server.Transfer()* keeps the original http request – the URL will not change in the browser – the program can then retrieve all the form and query string values. Because the program can retrieve all the form and query string values, it is easy to redisplay all the original form data in the uploading web page. The second way is to use Panel control. Using the Panel control, the program can display part of a page at one time and display another part of the page at another time. The CSSE Document Management System uses Panel control to keep original uploading data in the web page.

3.3 *Admin* Project Implementation

The *Admin* project offers all the functionalities for reviewing the documents uploaded by users and approving the uploaded documents – make them available for Internet users. It also includes the functionality for removing unqualified documents in the system – delete unqualified document from CSSEDMS. The administrative functionalities are only available to administrative personnel. As discussed in Chapter 2, Form based authentication method is used to authenticate administrative personnel.

3.3.1 Paging Implementation

Whenever an administrative user logs into CSSE Document management system, the system will show all the pending documents as you can see in Fig 3.2. The listing of pending documents is implemented using a *DataGrid* control. *DataGrid* control offers the functionalities for displaying data in a table. In addition, *DataGrid* control also allows for sorting, paging, editing of its data and write the data back into the database. All these functionalities are implemented via event handlers which response to *DataGrid*'s events.

To make a *DataGrid* sortable, the *AllowSorting* property must be set to true. After setting this property to true, *DataGrid* will render the headers of each column as a hyperlink. If users click on the hyperlinks, web page will be posted back to the server and the *DataGrid*'s *SortCommand* event fires. It is programmer's responsibility to wire up this event with an event handler and write the event handler to determine the header clicked and make the data sorted by accordingly.

Similarly, to make a *DataGrid* pageable, the *AllowPaging* property must be set to true. After setting this property to true, *DataGrid* will display navigational interface at the

top or bottom of the *DataGrid.* Users can use the navigational interface to step through pages of data displayed in the *DataGrid*. The navigational interface is rendered also as hyperlinks. Clicking on these links will post the web page back and fire *DataGrid*'s *PageIndexChanged* event. It is also programmer's responsibility to hook up the event with event handler and write the corresponding code to implement the paging.

Figure 3-2 Pending Documents

The "Approve" and "Delete" functions are implemented through command buttons. Each command button can be combined to a field in data source. Clicking on the command button raises *ItemCommand* event. The event handler will

distinguish which command button is clicked by retrieving the command name passed with *DataGridCommandEventArgs*. The implementation looks like the following:

```
if (e.CommandName == "Approve")
{
      Try
      {
      sqlConnection1.Open();
      SqlCommand sqlApproveCommand=new
      sqlCommand("approveDocument",sqlConnection1);
            …

      } catch (Exception ex1)

      {

      }

Else

{}
```

3.3.2 Retrieving Binary Data From Database

The document itself (digital file) is stored in the Auburn University SQL Server cluster database. To view the digital document, CSSEDMS must retrieve the document itself directly from database table (DOCUMENTS). After retrieving the digital file from the CSSE database, the Business logic layer should write the contents of the document back to a user's browser and display it inside the browser.

In ADO.NET, the *SqlDataReader* class treats binary data type inside SQL Server the same way with any other data types. If the data field inside a SQL Server table is binary or image – image type is used to save much larger binary type data – the *SqlDataReader* will return a collection which must be casted into *byte[]* array. After the casting, *Response* class can be used to write this *byte[]* array back to the browser.

Before writing the *byte[]* array back to the browser, two more steps need to be carried out. First, the *Response* class should clear the http response headers. *Response.ClearHeaders()* method is used to clear http response headers. Secondly, *Response* class should add content type information for browsers to process the document. *Response.AddHeader()* method is used to specify the content type and the way the contents should be disposed.

Response.AddHeader() method adds a new HTTP header to the http response sent to user browsers. *Response.AddHeader()* method will not replace an existing http header. After a header has been sent, it cannot be removed. *Response.AddHeader()* method takes two parameters, one for *HeaderName*, one for *HeaderValue*. For user browsers to properly process PDF binary data, the header added must specify the way to dispose the subsequent data. The actual *Response.AddHeader()* is used as:

```
Response.AddHeader("Content-Disposition", "inlne;filename=temp.pdf");
```

The “inline” part of the *HeaderValue* is essential. Without it browsers will open the PDF reader in another window, leaving current window blank. The whole implementation for view PDF file in browsers is similar to following:

```
SqlDataReader dr=cmdSelect.ExecuteReader();
if (dr.Read())
{
      Response.ClearHeaders();
      switch(dr["DocumentFileType"].ToString())
      {
      case "doc":
            Response.ContentType="application/msword";
            break;
      case "ppt":
            Response.ContentType="application/ppt";
            break;
```

```
                case "txt":
                      Response.ContentType="text/plain";
                      break;
                case "pdf":
                      Response.ContentType="application/pdf";//text/plain";
                      break;
                case "jpg":
                      Response.ContentType="image/JPEG";
                      break;
                default:
                      Response.ContentType="text/plain";
                      break;
                }
                Response.AddHeader("Content-Disposition",
"inline;filename=temp.pdf");

                Response.BinaryWrite((byte[]) dr["Document"]);
                Response.End();
```

The above code considered the situations for other digital file types. The CSSE Document management system can process Microsoft Word document, .jpg documents, PDF documents, PowerPoint documents, text documents. Using the *Response.ContentType()* method, the server can specify whichever content type for the browsers to dispose.

3.4 *CSSEDMS* Project Implementation

3.4.1 Introduction

CSSEDMS project includes functionality for searching a document. Users can search documents by author name, document title and key words. While searching by author name is a kind of pattern matching, searching by title and key words are implemented as *free text* search. You can search any phrases, any words and even a sentence. Clicking on the returned search results, users can review information about every document: authors, document finished date, abstract, etc. Users can also browser actual contents of a document.

The *free text* searching functionality is implemented through SQL Server Full Text search support. As discussed in Chapter 1, SQL Server Full Text search support includes two parts: full-text indexing support and full-text querying support. The indexing part creates indexes out of plain text, the querying part offers text search constructs such as CONTAINS or FREETEXT predicates and the FREETEXTTABLE row set functions.

Inside full-text indexes, all full-text words and their locations are saved. Full-text catalog stores all full-text indexes of a database. Compared with normal indexes of a table, full-text indexes are different in a number of ways. First, full-text indexes are saved in file system, only administrated through database. Secondly, while normal indexes of a table are updated automatically, full-text indexes updating need explicit operation and this operation must be scheduled or triggered by the changes of table data.

3.4.2 Setting up SQL Server for Full-Text Search Support

In order to implement the searching functionality of CSSEDMS, the first step is to setup the SQL Sever database for supporting Full Text search. By default, SQL Server 2000 doesn't install Full Text search support. If Full Text search support is not installed, SQL Server 2000 must be shut down and the Full Text search must be installed. Auburn University SQL Server 2000 cluster already has Full Text Search support installed. The following procedure is used to set up Full Text search support in SQL Server 2000:

- Enable database to support Full Text index.
- Identify tables and columns that will need Full Text search.

- Create a Full Text catalog.
- Register the tables for Full Text search.
- Add columns that will need Full Text search.
- Create Full Text indexes
- Populate the Full Text catalog.
- Setup change tracking (addition of new data, or modification of data), schedule Full Text indexes updating in background.

Using SQL Server Enterprise Manager, the setting up procedure is straight forward. Unfortunately, the SQL Server database used for CSSEDMS resides on Auburn University SQL Server cluster and Full Text search cannot be administered from SQL Server Enterprise Manager on a client computer [18]. The reason is that the Full Text search engine runs as a service named Microsoft Search (MSSEARCH service) and a client computer – where you run Enterprise Manager locally – cannot determine the state of MSSEARCH service on the server. The only choice available is to use stored procedures and Query Analyzer to setup Full Text search on Auburn University SQL Server cluster. The following stored procedures are used to set up Full Text search for the CSSE database on Auburn University SQL Server 2000 cluster.

```
use CSSE
EXEC sp_fulltext_database 'enable'
exec sp_fulltext_catalog 'CSSECatalog','create'
exec sp_fulltext_table 'Documents','create','CSSECatalog','PK_Documents'
exec sp_fulltext_column 'Documents','Title','add'
```

exec sp_fulltext_column 'Documents','Abstract','add'

exec sp_fulltext_table 'Documents', 'Start_change_tracking'

exec sp_fulltext_table 'Documents', 'Start_background_updateindex'

Stored procedures carried out the setting up steps mentioned above. First, *sp_fulltext_database* stored procedure enables the CSSE database to support Full Text search. Then *sp_fulltext_catalog* stored procedure creates a catalog named *CSSECatalog*. After that, *sp_fulltext_table* stored procedure enables DOUCMENTS table to use Full Text search. The *sp_fulltext_column* stored procedure adds Title and Abstract column for Full Text search. The last two statements are used to start change tracking and setup Full Text indexes updating as background updating.

3.4.3 CSSEDMS *Free Text* Search Implementation

From Chapter 1, *free text search* support format free input. Users can input any set of words, any set of phrases or even a sentence for searching. CSSEDMS can identify significant phrases in the input and return a matching set of records.

After setting up SQL Server for Full Text search, creating the Full Text indexes, and populating the indexes, Full Text search queries can then be executed against the data in SQL Server tables. Recall from Chapter 1, Full text search constructs include CONTAINS and FREETEXT predicates and the FREETEXTTABLE row set functions. CONTAINS and FREETEXT predicates are used as conditions in a normal SQL statement. FREETEXTTABLE returns a table of matching records. The advantage of using FREETEXTTABLE is that there is ranking information for each returned record.

This is essential for implementing a search engine, which must return how well each record matches search criteria.

The result set from FREETEXTTABLE function only contains two columns: KEY and RANK. KEY is the primary key of the table to be set up for Full Text search. RANK is a value from 0 to 1000. It represents how well a record matches the search criteria. 1000 means best match and 0 means the worst match. RANK value is a relative value (see next section). In order to get other information about records returned, FREETEXTTABLE function must be joined with the table itself. The stored procedure named *searchDocByTitle* is used for searching by TITLE in CSSEDMS:

```
CREATE PROCEDURE searchDOCbyTitle
        @searchPhrase varchar(255)
AS
        SELECT  Documents.DocumentID,convert(decimal(3,1),convert(float,RANK)/10)  RANK,
        Author, Documents.Title, Type, [DATE]
        FROM
        csse.AllinOne, Documents, freetexttable(Documents, title, @searchPhrase)
        WHERE
        [KEY]=Documents.DocumentID
        AND csse.AllinOne.DocumentID=Documents.DocumentID
        AND Documents.Approval_Status=1
        ORDER BY RANK DESC
```

The results returned from stored procedure *searchByDocTitle* is a row set. In CSSEDMS search page, this row set is then passed to a *DataGrid* for displaying to end users.

3.4.4 RANK Information Explanation

After users submit a search, CSSEDMS will process the inputs and finally call a stored procedure in SQL Server 2000, which in turn will perform the full text search on all the records stored in DOCUMENTS table. When the search performed is searched by Title or Key words, the stored procedures called by CSSEDMS will return related records with the ranking information. When the search performed is search by Authors, the stored procedure will not return ranking information. The search result page looks like Figure 3-3 Searching Results.

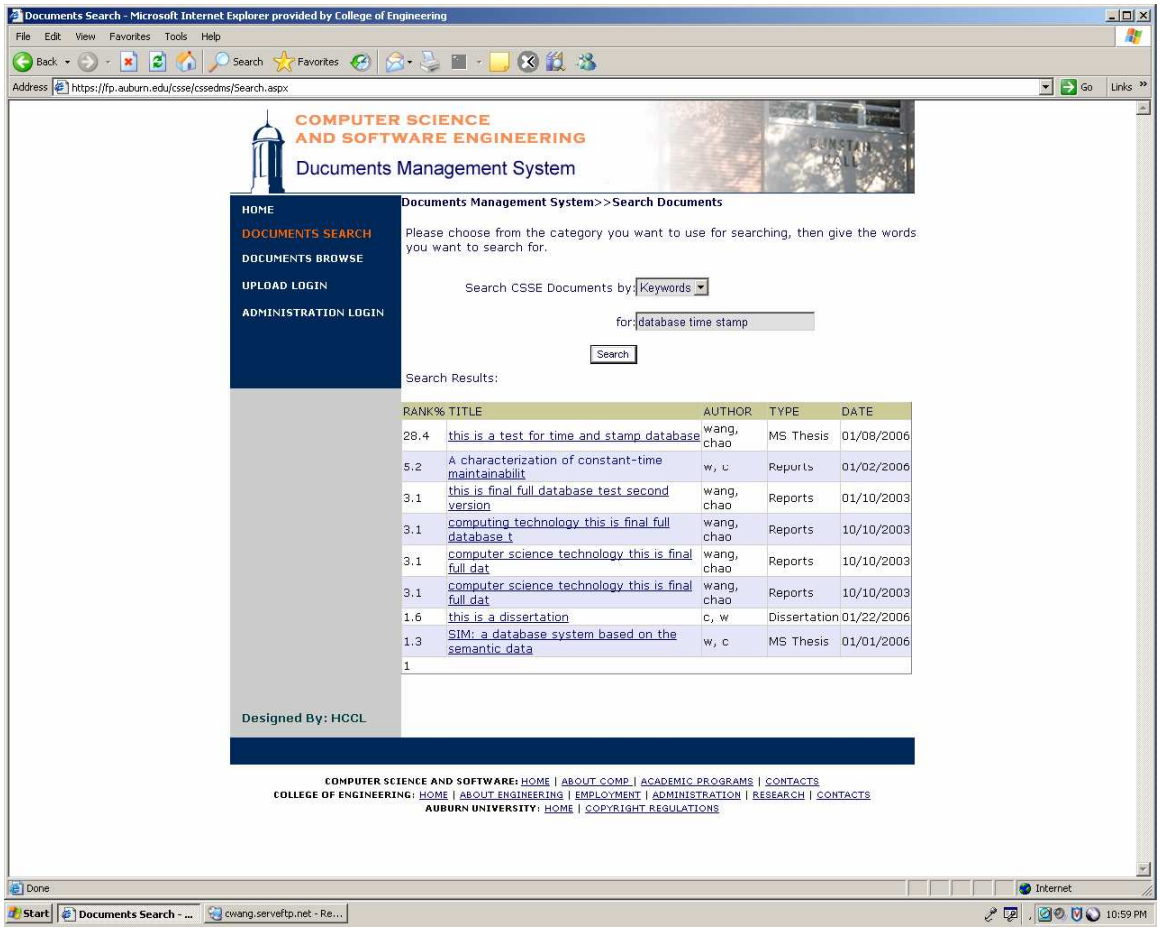

Figure 3-3 Searching Results

In the result set, the first column shows the ranking information. As discussed in section 3.4.1, the RANK information returned from FREETEXTTABLE is a value between 0 and 1000. CSSEDMS processes RANK value, turns it into a percentage value (divided by 10) and passes it back to the user interface.

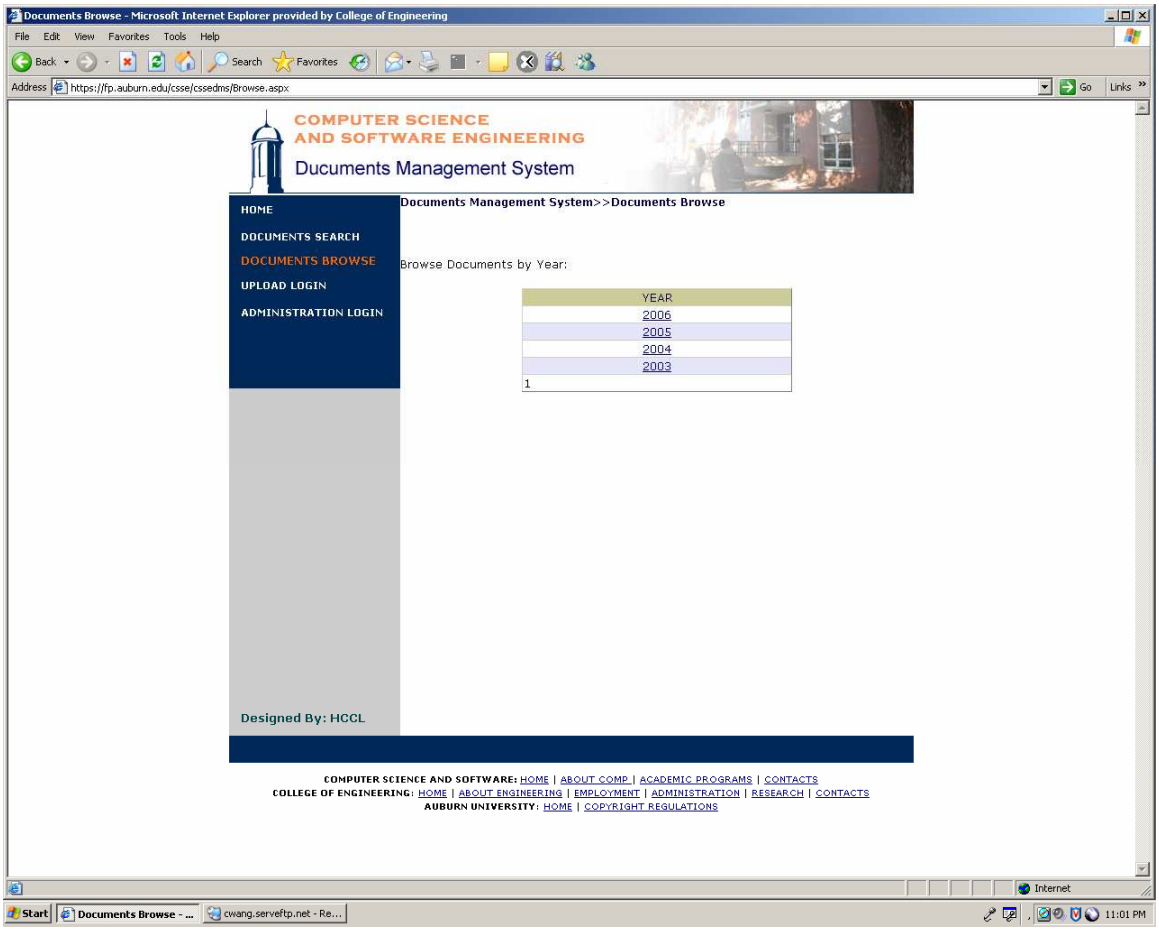

Figure 3-4 Document Browse

The RANK value returned from CONTAINSTABLE and FREETEXTTABLE is a relative value. This means that RANK values returned in the record set are relative to each other. The process used to calculate RANK values is very complicated. Statistics and computer artificial intelligence are used. Roughly speaking, the RANK value is based

on the frequency of rows that contain the key words searched. A major factor for determining the RANK value of the returned record is the frequency of the key word in the Full Text indexed field for that record. Another factor is the total number of key word occurrences in the table, which is used for normalizing the probabilities.

3.4.5 Browse Documents

Compared with document search functionality, document browsing is relatively simple. The purpose for document browsing is for the convenience of the users. With the searching functionality, users should be able to locate whichever documents are needed. Document browsing list all documents published by year. After clicking on the DOCUMENTS BROWSE menu, CSSEDMS looks like Fig 3-4.

The implementation of Document Browse is relatively simple. One stored procedure named *BrowseByYear* is used to retrieve records from the SQL Server database.

3.4.6 Retrieve all Authors of One Document

The CSSE database in the SQL Server 2000 cluster was normalized to third normal form. The normalization can reduce inconsistency and redundant data. However, it also incurs table joins and calculations when retrieving data from the database. Such a situation occurs when there is a need to retrieve all the authors of one document. At the database layer, the program must join PEOPLE table and DOCUMENTPEOPLE table to get all authors of a single document. By using a cursor, the following T-SQL code returns all authors of one document:

```
Create Procedure AllAuthors
@DocumentID int
As
Begin
DECLARE @authors varchar(300)  -- save all authors for one documentID
DECLARE @temauthor varchar(50)
SET @authors=''
DECLARE authors_cursor CURSOR FOR
SELECT firstName+' '+lastname
FROM people,documentPeople
WHERE Documentpeople.peopleID=people.peopleID
AND documentID= @DocumentID   -- get all authors for documentID=@DocumentID
ORDER BY DocumentPeople.roleID desc
OPEN authors_cursor
FETCH NEXT FROM authors_cursor
INTO @temauthor
WHILE @@FETCH_STATUS = 0
BEGIN
  SET @authors=@temauthor+','+@authors
    -- Get the next author.
  FETCH NEXT FROM authors_cursor
  INTO @temauthor
END
CLOSE authors_cursor
DEALLOCATE authors_cursor
End
```

Certainly retrieving all authors for one document can be implemented at the Business logic layer. The design of CSSDMS tried to keep all the data retrieving logic at the stored procedure level.

3.5 Exception Handling in CSSEDMS

Exception handling is very important for an application. If an exception is not handled at code level, it will propagate to next level, which is page level in CSSEDMS. If at the page level, the exception is still not handled, the exception will propagate further to application level. At the application level, if the exception is not handled, it can not propagate more and ASP.NET run time is forced to handle it. The way ASP.NET handles exceptions depends upon the settings specified in the *customErrors* section in *web.config*. If nothing is specified, ASP.NET will use the defaults and display the infamous 'yellow' error page. Because users will never know what's happening in the code, it is user unfriendly if ASP.NET handles the exception with default settings.

Almost every single line of code can throw an exception. Specifically, there is a possibility for every line of code to cause *OutOfMemoryException* and *StackOverflowException*, which are thrown by the common language runtime itself. When an exception occurs, control is given to the closest exception handler. The most inside exception handler is at the code level. The *try-catch-finally* clause is used to catch and handle code level exceptions. If an exception occurs outside of the *try-catch-finally* block and propagates to the next level, the *Page.Error* event fires. In the page level exception handler, the first action is to obtain the exception thrown, by using the *Server.GetLastError* method. After obtaining the *Exception* object, the handler will do

something like exception logging. It will then redirect the control to an error page. At the application level exception handler, the processing is similar with the page level. However, at the end of processing, *Server.ClearError()* should be called. *Server.ClearError()* clears the exception and inform ASP.NET that the exception has been handled. If the exception is not cleared, ASP.NET will handle the exception again and generate the infamous ‘yellow’ error page.

To prevent ASP.NET from using default settings to handle exceptions, CSSEDMS provides three levels of exception handling – code level, page level and application level. CSSEDMS also configures a default redirection in *customErrors* section in *web.config*. With three levels of exception handling, end users will see the customized error page, not the infamous ‘yellow’ page.

CHAPTER 4 USABILITY EVALUATION

4.1 Introduction

In design and implementation, the application developers and designers use their expertise to determine future user use of an application. They use their knowledge gathered in their career to make judgments about potential users. However, the application developers and designers are typically too close to the application to truly understand the user's needs. What seems extremely straightforward to developers and designers might be an obstacle to real users. Users always have problems that the developers and designers don't expect. Usability evaluation is used to validate the assumptions made by application developers and designers and make sure all the elements of the interface are meaningful for real users.

Planning is the key for successful usability evaluation. Before conducting the actual usability evaluation, a usability evaluation strategy should be developed. Some of the questions that should be asked are: What does the evaluation test for? What are the goals of this evaluation? What concerns do you have about the application that you want to evaluate? Who will be your participants? Where will you conduct the test? etc.

Based on the usability evaluation goals, an evaluation test plan is developed using scenarios. Scenarios define the tasks that participants will go through in the testing session. Scenarios must be prepared, tried out and refined before actual participants test on them. They must be clearly written and not demand too much time to understand. In

the scenario development stage, the usability evaluation designers must decide which kind of data will be collected when participants are taking the test. Will the data be subjective or objective? Will the data be quantitative or qualitative?

After developing the scenarios and the data to be collected, the next step is to recruit participants. At which time more questions need to be answered: How to recruit participants who can accurately represent potential users? What incentives can participants get? Are there logistics considerations to be made?

At the beginning of the actual testing, the usability evaluators need to help the participants understand that they are helping validate assumptions made by application developers and designers. The usability evaluators should tell participants that the evaluation is not a test for the participants. It is a test for the application. The usability evaluators should also ask participants to perform tasks as if they were in their own work environments. They should also encourage participants to think aloud as they are working on the tasks and make sure participants express their reactions as freely as possible. During the testing session, the usability evaluators should listen, don't lead and keep neutral in their language. In the mean time, they should make sure the data is collected correctly and their presence will not skew participants' original response.

The last step is to process the data collected, identify usability issues, list problems that participants had, and develop solutions for the problems identified. In other words, this step will identify what is working well in the application and what needs to be fixed.

4.2 CSSEDMS Usability Evaluation

4.2.1 The Scenarios/Tasks of CSSEDMS Usability Evaluation

CSSEDMS usability evaluation is a usability verification test. The goal is to verify if the design and implementation meet the preset usability goals and users' satisfaction. The major concerns are: Is CSSEDMS easy to use? Is the interface straightforward? Does CSSEDMS merge well with current CSSE departmental Web site? Because potential users of CSSEDMS are largely from the CSSE department, all participants have registered for at least one course at the CSSE department.

Based on CSSEDMS usability evaluation goals, three testing scenarios were developed. These three scenarios cover the major functionalities of CSSEDMS. At the beginning of the testing session, the Web browser's address was set to the home page of CSSEDMS, that is, https://fp.auburn.edu/csse/cssedms/.

Scenario/Task 1: upload a document into CSSEDMS.

Expected steps:

- Participants click on UPLOAD LOGIN menu.
- Participants input their Auburn University username and password, prefixing Auburn University username with "auburn\", and click "login" button or hit "Enter" key.
- Participants fill in the uploading form and click "upload" button.
- Participants click the SIGN OUT menu to sign out the system.

Scenario/Task 2: approve a document just being uploaded by making it available to Internet users.

Expected steps:

- Participants click the ADMINSTRATION LOGIN menu to sign in the system.
- Participants input administrative username (“admin”) and password (“adminpass”) and click on “login” button or hit “Enter” key.
- (optional) Participants click on the title of the document just uploaded, view the information about this document. Participants might view the digital document itself by clicking on the “view document contents” link.
- Participants identify the document just uploaded, click on “Approve” link to approve the document just being uploaded. This will make it available to Internet users.
- Participants click on “SIGN OUT” menu to sign out the system.

Scenario/Task 3: Search a document that was uploaded and approved in the last two steps. After finding the document they are searching for, view and save it to local disk.

Expected steps:

- Participants click on “SEARCH DOCUMENTS” menu to go to the Search page.
- Participants click on the dropdown list to choose one of the search methods (Search by Title, Search by Author and Search by Keywords). Participants input the words they want to search for and hit the “Enter” key or click on “Search” button.

- Participants click on the document title returned in a list.
- Participants view the information about the document and click on the document icon or link to see the digital document itself.
- Participants repeat last three steps for other two search methods. Participants are asked to use all three search methods to find the document uploaded.
- Participants click “File/Save as” to save the document to local disk.

4.2.2 CSSEDMS Usability Evaluation Data Collection

Recall from Chapter 1, the usability evaluation of the CSSE Document Management System measures three aspects of the application: effectiveness, efficiency and user satisfaction. To measure effectiveness, whether or not a participant completes a task successfully was recorded. To measure efficiency, time needed for a participant to finish all three tasks was recorded. User satisfaction measurement was carried out through a questionnaire which was filled out after participants finished three tasks.

The Usability evaluation for CSSEDMS largely depends on System Usability Scale (SUS) to measure users’ satisfaction. SUS score represents overall users’ satisfaction. SUS is developed by Digital Equipment Co Ltd, United Kingdom. It only has ten questions. See Appendix A – SUS Scale. To use SUS properly, the participants should answer SUS questions right after the participants have used the application. Participants should also be asked to give their instant response to each question, rather

than thinking about them for some time. Participant should not discuss the application with others before answering SUS questions. All the questions should be answered, if a participant is not sure about her/his response, s/he should mark choice 3 – the middle of the scale [27].

The usability evaluation for CSSEDMS modified the first question of SUS to accommodate the actual situation. The original question is:

I think that I would like to use this system frequently:

strongly disagree ***strongly agree***

1 2 3 4 5

This question is only suitable for web applications that the participants might want to use frequently. A search engine Web site is an example. For CSSEDMS, it is mostly unlikely that a participant will use it frequently. So the original question is modified to:

I think that I would like to use this system:

strongly disagree ***strongly agree***

1 2 3 4 5

In addition to SUS questions, the questionnaire for CSSEDMS usability evaluation has additional questions designed to measure user's anticipation, terms and language used. These questions reflect the look/feel of the system. The user anticipation measurement is used to confirm that the developer's assumptions about labeling and navigation are meaningful to users. This means, for example, when users click a link, they get exactly what they want to get. One way to do this measurement is to use a two-column table. The first column's header is "Link/Menu" and the second column is

"Expectation". See Appendix B – Labeling and Navigation Menu Expectation. Participants are asked to fill out this table. Such a table can be used to collect user's anticipation and intuition. However, responses from the participants will be text. It is difficult to score such textual responses. Ideally, all the responses or answers from the participants should be quantified in some way. In CSSEDMS evaluation questionnaire, questions were asked like:

> *You feel that the navigation menu ADMINISTRATIVE LOGIN means administrative personnel can login here by clicking on it*
>
> ***strongly disagree*** ***strongly agree***
>
> *1 2 3 4 5*

This question form is very similar to SUS questions. Using such question form, the responses from the participants can be easily quantified.

The question used for terms and language used looks like this:

> *I feel that I encountered terms that I didn't understand*
>
> ***Strongly disagree*** ***strongly agree***
>
> *1 2 3 4 5*

One question is used to measure whether or not CSSEDMS merges well with current CSSE departmental Web site:

> *I feel that the style/color of this system is similar with CSSE departmental Web site.*
>
> ***Strongly disagree*** ***strongly agree***

1 *2* *3* *4* *5*

Refer to Appendix C – Questionnaire for all the questions asked. Sum of the score of these questions generates an overall score regarding the system's look and feel. Consequently, this score is called "Look/Feel" score in CSSEDMS usability evaluation report.

CHAPTER 5 USABILITY EVALUATION REPORT

5.1 Usability Evaluation Data Processing

Three kinds of data are collected in the usability evaluation testing. They are: whether or not a participant successfully finished a task (the measurement of system effectiveness), the time that a participant used to finish three tasks (the measurement of system efficiency), and the data from questionnaire (the measurement of user satisfaction). Data processing for the first two is straight forward. Questionnaire data processing is more time consuming.

As stated in Chapter 4, the major part of the questionnaire is System Usability Scale (SUS). Consequently, scoring SUS was the major data processing work. Data collected in other questions in the questionnaire were processed similarly with SUS.

In System Usability Scale, each question has a score range from 0 to 4. For question 1, 3, 5, 7 and 9, the score is the position selected minus 1. For questions 2, 4, 6, 8, and 10, the score is 5 minus the position selected. To calculate overall SUS score, first sum the scores from each individual question; then multiply the sum of the scores by 2.5. The result is the overall value of SUS score. SUS score is a single value of a range between 0 and 100. It represents an overall view of users' satisfaction. The higher the SUS score, the better the users' satisfaction. Score from other questions (Look/Feel) in

the questionnaire represents the overall view of user's anticipation, proper terminology usage and if CSSEDMS merges well with the CSSE departmental Web site.

5.2 CSSEDMS Usability Evaluation Report

5.2.1 CSSDMS Usability Evaluation Data

Thirty seven students participated in CSSEDMS evaluation testing. Among these thirty seven students, fourteen are female, twenty-three are male. During the usability evaluation testing session, the conductor recorded the time used for each participant to complete all three tasks. If a participant failed to complete one of the tasks – get stuck in the system or asked for help, the time is noted as 0.

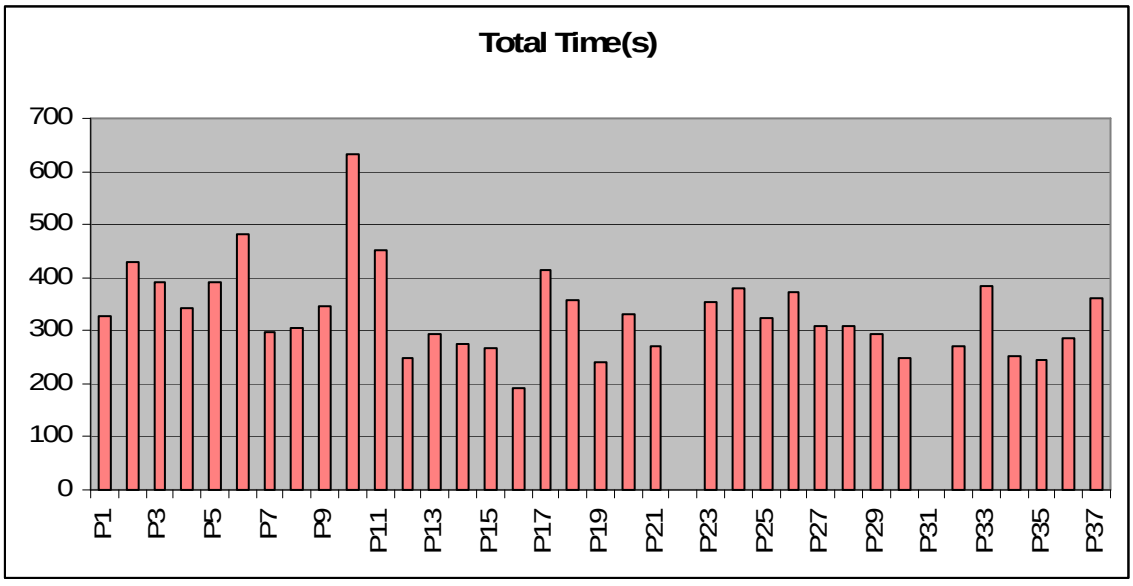

Figure 5-1 Total Time Used to Complete Three Tasks

Note: P22 and P31 Failed Task2 & Task3.

	SUS	LOOK/FEEL	TOTAL TIME(s)	
P1	92.5	87.5	327	
P2	72.5	75	429	
P3	60	68.75	393	
P4	70	87.5	342	
P5	87.5	81.25	390	
P6	97.5	93.75	480	
P7	82.5	87.5	299	
P8	85	81.25	306	
P9	70	87.5	347	
P10	82.5	87.5	634	
P11	70	87.5	453	
P12	82.5	81.25	248	
P13	92.5	87.5	293	
P14	87.5	75	275	
P15	90	87.5	268	
P16	100	93.75	191	
P17	85	100	414	
P18	77.5	87.5	357	
P19	92.5	93.75	242	
P20	95	87.5	331	
P21	75	68.75	271	
P22	60	68.75	0	failed task2 & task3
P23	90	87.5	353	
P24	77.5	81.25	379	
P25	85	93.75	325	
P26	70	68.75	373	
P27	92.5	81.25	310	
P28	90	81.25	309	
P29	77.5	81.25	294	
P30	92.5	100	247	
P31	67.5	56.25	0	failed task2 & task3
P32	97.5	93.75	270	
P33	77.5	75	383	
P34	75	87.5	254	
P35	80	68.75	245	
P36	92.5	100	287	
P37	100	81.25	362	
AVERAGE	80.94	82.81	368.11	
STDEV	10.84	10.03	111.68	

Table 5-1 CSSEDMS Usability Evaluation Data

Table 5-1 shows SUS score, Look/Feel score and the time used to finish three tasks for all thirty-seven participants. Two participants failed to finish task2 & task3, consequently

their time were marked as zero. Figure 5-1, Figure 5-2 and Figure 5-3 show chart representation of the data from Table 5-1.

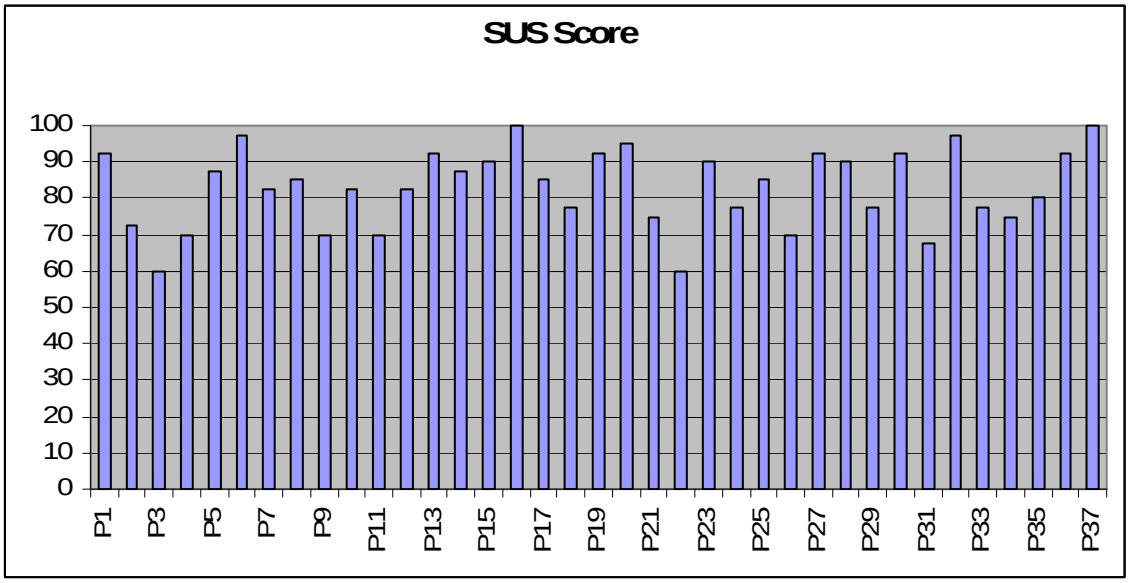

Figure 5-2 CSSEDMS Usability Evaluation SUS Score

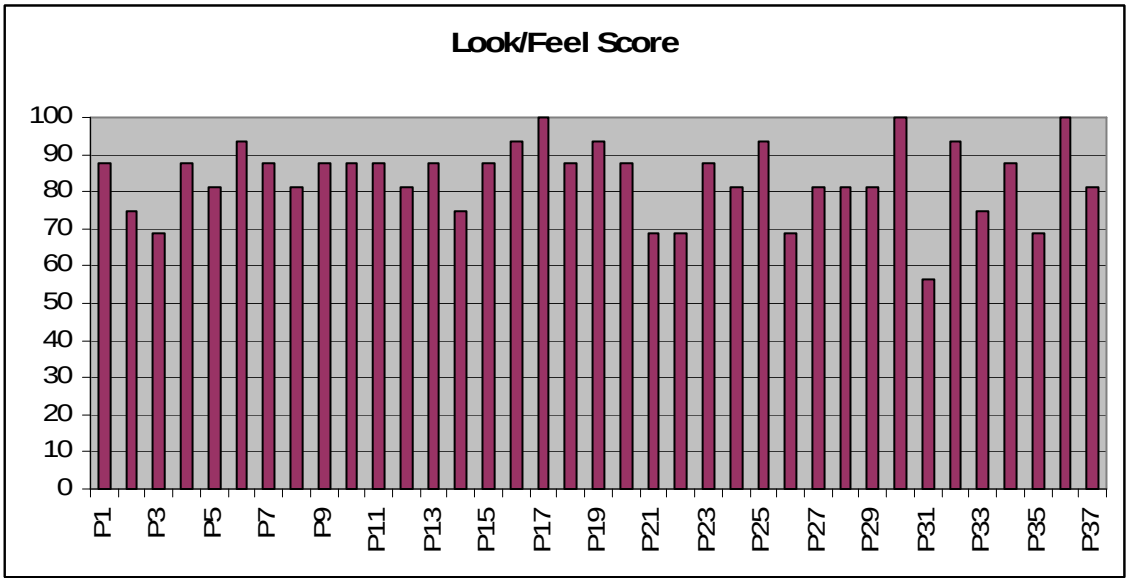

Figure 5-3 CSSEDMS Usability Evaluation Look/Feel Score

5.2.2 CSSDMS Usability Evaluation Report

CSSEDMS Usability Evaluation Report

Executive Summary

The usability evaluation of CSSEDMS revealed that most participants (95%) were able to complete all three tasks: upload a document into CSSEDMS, approve the document uploaded and search the document uploaded. The average overall subjective assessment (System Usability Score) of CSSDMS is 80.94 out of 100, which leaves low margin for further usability improvement [27]. Most participants claimed CSSEDMS merges well with current CSSE departmental Web site. The most important problem found was related with the authentication methods. Two authentication methods are used in CSSEDMS, Windows based authentication and Form based authentication. The style and feel of these two authentication methods are different, which leads to inconsistency to participants.

Three most visible problems were identified:

- When logging in to upload, participants were not used to prefix an auburn domain name before his/her username. This is because there isn't such a need when they log into a computer.
- Five participants were not sure if the document were already approved. They didn't see the message at the top of pending document list.
- One participant pointed out that if a document gets approved, the CSSDMS should email authors about the approval of the document.

Introduction

CSSEDMS usability evaluation was focused on measuring CSSEDMS system effectiveness, efficiency and user's satisfaction. Thirty seven participants were asked to perform three tasks using CSSEDMS system. Major participant experiences were observed when participants were using CSSEDMS system.

Participants

Thirty seven students participated in CSSEDMS evaluation testing. Among these thirty seven students, fourteen were female, twenty three were male. All participants are associated with the CSSE department. They must have registered for at least one course at the CSSE department.

Scenarios/Tasks Summary

Three scenarios/tasks were developed:

- Scenario/Task 1: Upload a document into CSSEDMS
- Scenario/Task 2: Approve the document uploaded in scenario 1.
- Scenario/Task 3: Search the document approved in scenario 2.

Detailed description of scenarios can be found in Chapter 4.

Scenario/Task Analysis

Scenario/Task 1: Upload a Document into CSSEDMS

Objectives

- Evaluate the usability of CSSEDMS's uploading functionality.

Goal

- 100% of participants would successful upload a document.

Notes

- At the beginning of the evaluation testing, all participants were given instructions on how to login to upload a document. Many of them were not comfortable with the way to prefix Auburn University username with auburn's domain. They asked for confirmation when they actually logged in to CSSEDMS system.

Results

- All participants succeeded in uploading a document.

Scenario/Task 2: Approve the Document Uploaded in Task 1.

Objectives

- Evaluate the usability of administrative "Approve" functionality.

Goal

- 100% of participants would successfully use the "Approve" functionality.

Notes

- One participant asked why CSSEDMS needed another password. Although this participant didn't fully understand the "Approve" functionality, s/he did successfully complete the task.
- Five participants were not sure if they succeed in approving a document. The message that appeared at the top of the pending document list didn't catch their eyes.

- Two participants failed task2. After logging in as an administrator, they quickly clicked on the title link and didn't notice the "Approve" link. This leaded them to the information page about the document. On the information page, there is not a link or button for approving the document. They didn't figure out that they should close the information page to go back to the "Approve" page.

Results

- 95% of participants succeeded.

Scenario/Task 3: Search the Document Uploaded in Task 1.

Objectives

- Evaluate the search functionality of CSSEDMS.

Goal

- 100% of participants can find the document in CSSEDMS uploaded in scenario/task 1

Notes

- Although two participants failed task2, they continued to the task 3. They tried to use the search functionality. They didn't find the document they uploaded in task1. However, they seemed to know how to use the search functionality.

Results

- 95% of participants succeeded in using searching functionality

Findings and Recommendations

Problem 1:

Logging into CSSEDMS needs to prefix Auburn University username with "auburn\", this caused confusion for participants. There is no need for a participant to prefix Auburn University username with "auburn\" when s/he logs into a computer. Participants are used to logging into an Auburn system in the manner.

Severity

Because of this problem:

- Many participants asked for confirmation before they logged into CSSEDMS.

Recommendation

- Use Form based authentication. If using Form based authentication, then there is no need to prefix Auburn University username with "auburn\". However, Form based authentication needs access to user information table in SQL Server cluster, which currently is not supported by Auburn University OIT office.
- Setting up Active Directory default domain name might be another solution to this problem.

Problem2:

Two participants quickly clicked on title link after being logged in as an administrator and didn't see the "Approve" link. This brought them to the document information page. They could not figure out the way to go back to the "Approve" page.

Severity

Because of this problem:

- Participants were not able to approve the uploaded document.

Recommendation

- A very brief tutoring about approving a document would be very useful.
- Add an “Approve” link on the document information page should solve the problem and also bring additional way (convenience) for approving a document.

Problem3:

Five participants were not sure that they already approved the document – already made it available to Internet users. The feedback at the top of the pending document list didn’t catch their attention.

Severity

Because of this problem:

- Participants were not sure about what the “Approve” link did.

Recommendation

- Set the feedback message in larger font (maybe in red color).
- Another solution is to change the display of the pending list. Add a column called status, which will change from “pending” to “approved” after a user clicks on “Approve” link.

CHAPTER 6 CONCLUSIONS AND FUTURE WORK

6.1 Conclusions

According to the document management needs at the CSSE department, this thesis work designs and implements a system that facilitates document management. Guided by usability and modern software architectural ideas, this thesis work developed a usable, maintainable web application (CSSEDMS) which merges well with the current CSSE departmental Web site.

This study emphasizes the practices used to build a usable and maintainable Web application. Improving the usability and maintainability of a web application like CSSEDMS is the theme of this thesis. The practices of this thesis work show that carefully designed usability evaluation is an effective way to locate the usability problems of an application and could consequently improve the application's usability.

This thesis work also shows that data - centric procedural design is an effective and simple architecture for applications that are not very complex. In CSSEDMS design, everything is about data manipulation. All workflows are related to retrieving data from the CSSE database, making changes and writing data back into the CSSE database etc. Since everything revolves around data flow, the design and implementation is straightforward.

6.2 Future Work

Although the result of CSSEDMS usability evaluation is satisfactory, in that the average System Usability Scale score reached 80.94 out of 100, the usability evaluation conducted should be the first iteration of the iterative approach for developing CSSEDMS system. Further usability evaluations will bring more clues about improving CSSEDMS system.

The CSSE Document Management System should include some reporting services. Such reporting services could generate reports based on year, subject, etc. The reporting services could offer an overall view about documents in the CSSE department.

Based on SQL Server full text support, this thesis implements *free text* search on the Title and Abstract of a document. However, the most direct way is to implement *free text* search on the contents of the document itself. Consequently, using filters against binary documents in the database is the next step to go. Filters are used by SQL Server Full Text support to interpret binary data. They are DLLs that include an implementation of the IFilter interface for a specific class of files. For different document types, different filters need to be developed, which will add a new challenge to over come in the future work of this thesis.

REFERENCES

1. Eva M. Thury, "Analysis of Student Web Browsing Behavior: Implications for Designing and Evaluating Web sites," in *Proceedings of the 16th Annual International Conference on Computer Document,* 1998.
2. Ruth Wilson, Julie Shortreed, and Monica Landoni, "A Study into the Usability of E-encyclopaedias," in *Proceedings of the 2004 ACM symposium on Applied Computing*, 2004**.**
3. Juan E. Gilbert, http://www.juanegilbert.com/.
4. Spool, Jared M. et al, "Web site Usability: A Designer's Guide," in *User Interface Engineering*, North Andover, Mass., 1997.
5. Ann Blandford, Suzette Keith, Iain Connell and Helen Edwards, "Evaluation: Analytical Usability Evaluation for Digital Libraries: a Case Study," in *Proceedings of the 4th ACM/IEEE-CS Joint Conference on Digital Libraries*, June 2004.
6. R. Anderson, B. Francis, A. Homer, R.Howard, D.sussman, K. Watson. "Professional ASP.NET." Birmingham, Wrox Press, ISBN 1-861004-88-5, 2001.
7. Laurie Kantner and Stephanie Rosenbaum, "Usability Studies of WWW Sites: Heuristic Evaluation vs. Laboratory Testing," *Proceedings of the 15th Annual International Conference on Computer Documentation*, Oct. 1997.

8. Valerie Mendoza and David G. Novick, “Usability: Usability Over Time,” *Proceedings of 23rd Annual International Conference on Design of Communication: Documenting & Designing for Pervasive Information SIGDOC ’05*, 2005.
9. “A Conversation with Pat Selinger – Leading the Way to Manage the World’s Information,” *ACM queue, Vol. 3, No. 3* - April 2005, http://acmqueue.com/.
10. www.oracle.com/technology.
11. SQL Server 2000 books online.
12. Jenny Preece, Yvonne Rogers, Helen Sharp, “Interaction Design: Beyond Human-Computer Interaction,” *ISBN: 0471492787*.
13. Thomas S. Tullis, Jacqueline N. Stetson, “A Cmparison of Questionnaires for Assessing Website Usability,” *UPA 2004 Conference*.
14. Darleen Sadoski, Santiago Comella-DordaDarlee, “Three Tier Software Architectures,” http://www.sei.cmu.edu/str/descriptions/threetier.html
15. Rocky Lhotka “Expert C# Business Objects,” *ISBN: 1590596323*.
16. Dickman, A. "Two-Tier Versus Three-Tier Apps." *Informationweek 553 (November 13, 1995): 74-80*.
17. http://www.dotnetrocks.com, *Archive #8*.
18. Stephen Walther, “ASP.NET Unleashed,” *ISBN: 0-672-32068-1*.
19. Microsoft Support, “PRB: Full Text Search Menus Are Not Enabled for Local Windows NT Accounts,” http://support.microsoft.com/default.aspx?scid=kb;en-us;270671

20. “Terabyte Grows to Petabyte,” http://www.sigex.com/pgs/todaysanalysis/traffic8P.php
21. Joe Hummel, “Architecting Modern Desktop Apps in .NET ,” *MSDN webcast,* http://www.microsoft.com/events/series/modernsoftdev.mspx
22. “A Brief History of Search Engines,” http://www.webreference.com/authoring/search_history/
23. Jesper Kjeldskov, Mikael B. Skov, Jan Stage, “Does Time Heal?: A Longitudinal Study of Usability.” *ACM Internatinal Conference Proceeding Series; Vol. 122,* 2005.
24. Laurie Kantner, Stephanie Rosenbaum, “Usability Studies of WWW Sites: Heuristic Evaluation vs. Laboratory Testing,” *ACM Special Interest Group for Design of Communications*. *Proceedings of the 15th Annual International Conference on Computer Documentation*. Pages: 153-160, 1997.
25. Melody Y. Ivory, Marti A Hearst, “The Sate of the Art in Automating Usability Evaluation of User Interfaces.” *ACM Computing Surveys (CSUR),* Volume 33 Issue 4.
26. Dennis de Champeaux, Doug Lea, Penelope Faure, “The process of Object-oriented design.” *Conference on Object Oriented Programming Systems Languages and Applications.* 1992.
27. U.S. Department of Health and Human Services. http://www.usability.gov/pubs/030106news.html
28. John Brooke, “SUS – A quick and dirty usability scale.” *Redhatch Consulting Ltd.*, http://www.usabilitynet.org/trump/documents/Suschapt.doc

APPENDIX A – SYSTEM USABILITY SCALE

1. I think that I would like to use this system frequently:

   strongly disagree **strongly agree**

   1 2 3 4 5

2. I found the system unnecessarily complex

   strongly disagree **strongly agree**

   1 2 3 4 5

3. I thought the system was easy to use:

   strongly disagree **strongly agree**

   1 2 3 4 5

4. I think that I would need the support of a technical person to be able to use this system

   strongly disagree **strongly agree**

   1 2 3 4

5. I found the various functions in this system were well integrated:

   strongly disagree **strongly agree**

   1 2 3 4 5

6. I thought there was too much inconsistency in this system

   strongly disagree **strongly agree**

   1 2 3 4 5

7. I would imagine that most people would learn to use this system very quickly

   strongly disagree **strongly agree**

   1 2 3 4 5

8. I found the system very cumbersome to use

strongly disagree				**strongly agree**
1	2	3	4	5

9. I felt very confident using the system

strongly disagree				**strongly agree**
1	2	3	4	5

10. I needed to learn a lot of things before I could get going with this system

strongly disagree				**strongly agree**
1	2	3	4	5

APPENDIX B – EXPECTATIONS TABLE

Labeling and Navigating Menu Expectations

Menu	Expectation
1. SEARCH DOCUMENTS	
2. BROWSE DOCUMENTS	
3. UPLOAD LOGIN	
4. ADMINISTRATIVE LOGIN	

APPENDIX C – QUESTIONNAIRE

Please try to answer all of the following questions. If you are not sure about one question, choose middle choice.

1. What is your gender?

   (1) Female (2) Male

2. I think that I would like to use this system:

   Strongly disagree strongly agree

   1 2 3 4 5

3. I found the system unnecessarily complex

   Strongly disagree strongly agree

   1 2 3 4

4. I thought the system was easy to use:

   Strongly disagree strongly agree

   1 2 3 4 5

5. I think that I would need the support of a technical person to be able to use this system

   Strongly disagree strongly agree

1 2 3 4 5

6. I found the various functions this system were well integrated:

Strongly disagree strongly agree

1 2 3 4 5

7. I thought there was too much inconsistency in this system

Strongly disagree strongly agree

1 2 3 4 5

8. I would imagine that most people would learn to use this system very quickly

Strongly disagree strongly agree

1 2 3 4 5

9. I found the system very cumbersome to use

Strongly disagree strongly agree

1 2 3 4 5

10. I felt very confident using the system

Strongly disagree strongly agree

1 2 3 4 5

11. I needed to learn a lot of things before I could get going with this system

Strongly disagree strongly agree

1 2 3 4 5

12. I feel that I encountered terms that I didn’t understand

Strongly disagree strongly agree

1 2 3 4 5

13. I feel that the navigation menu ADMINISTRATIVE LOGIN means administrative personnel can login here by clicking on it

Strongly disagree strongly agree

1 2 3 4 5

14. I feel that the navigation menu UPLOAD LOGIN means users can login here by clicking on it and upload a document

Strongly disagree strongly agree

1 2 3 4 5

15. I feel that the style/color of this system is similar with CSSE departmental Web site.

Strongly disagree strongly agree

1 2 3 4 5

16. Please provide any additional comments that you have about the system:

Note: After the experiment, please don't mention this experiment to any other students who will do this experiment later! Thank you very much!