DESIGN AND DEVELOPMENT

OF A

DIGITAL LIBRARY SYSTEM

SOFTWARE

BY

MU'AZU ABDULMAJEED

(2005/22052EE)

PARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING,

HOOL OF ENGINEERING AND ENGINEERING TECHNOLOGY,

FEDERAL UNIVERSITY OF TECHNOLOGY MINNA

November 2010

DESIGN AND DEVELOPMENT OF A DIGITAL LIBRARY SYSTEM SOFTWARE

BY

MU'AZU ABDULMAJEED

(2005/22052EE)

A PROJECT SUBMITTED IN PARTIAL FUFILMENT OF THE REQUIREMENTS FOR THE AWARD OF BARCHELOR OF ENGINEERING (B.ENG) DEGREE IN THE DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING, SCHOOL OF ENGINEERING AND ENGINEERING TECHNOLOGY, FEDERAL UNIVERSITY OF TECHNOLOGY MINNA

November 2010

DEDICATION

I dedicate this work to God, the Most Glorious and to my Father, Mother and Sister who have all been of great support to me.

DECLARATION

I Muazu Abdulmajeed, declare that this work was done by me and has never been presented elsewhere for the award of a degree. I also hereby relinquish the copyright to the Federal University of Technology, Minna.

MUAZU ABDULMAJEED
(Name Of Student)

10/11/2010
(Signature and Date)

Paul Abraham-Attah
(Name Of Supervisor)

10/11/2010
(Signature and Date)

ENGR A.G. RAJI
(Name Of H.O.D)

(Jan. 11, 2011)
(Signature and Date)

Engr Dr. G.E. Igbalo
(Name Of External Examiner)

1/12/10
(Signature and Date)

ACKNOWLEDGMENT

I wish to acknowledge the support the programmers at Code Project whose works were of great benefit to me and to my supervisor who guided me throughout the development of the project.

ABSTRACT

Electronic sources of information and low-cost microcomputers have introduced unprecedented changes to the services and operations of modern libraries. With large-scale digitization projects underway at Google, in the near future, most of today's libraries will be required to provide digitized library services. This project designed and implemented the software system of a digital library system whose aim would be to provide digital library services.

The project work consist of designing a model and creating two (2) applications, one of which is to be used by the library or organization to provide digital library services, while the other, to be used by digital library patrons to access these digital library services. The project uses C# programming language with the .Net Framework and the Windows Presentation Foundation model in implementing the software and hence the implementation of the applications, are limited to operating under Microsoft's Operating Systems with the .Net Framework.

At the end of the project, all the necessary functionalities of the system successfully passed their requirement tests and the system's operating requirements were obtained.

TABLE OF CONTENT

LIST OF FIGURES

LIST OF TABLES

LIST OF PLATE

CHAPTER ONE
INTRODUCTION

1.1: Preamble

Libraries are increasingly called upon not only to collect information in electronic form but also to organize it into digital libraries. Digital Libraries are collections of organized sources of information and their preservation in electronic format. The content (source of information available electronically) may be created by libraries and held locally, or may be created and made available in a distributed fashion through a Digital Library Service.

A Digital Library Service refers to the systematic process of providing organized and controlled access to a digital library. Implicitly computers are used due to the basic requirements such as storage, organization and provision of secured access to the digital contents. A Digital Library Service Provider (DLSP) is hence, an organization or institution that provides Digital Library Service(s).

A system whose aim is to provide digital library services and a means of consuming such services is referred to as a Digital Library System. This consists of interconnected computers that house the software system. The software system consists of a servicing system and service consuming systems.

1.2: The Project

The project is to develop the software system of a digital library system that organizations such as a university may utilize to provide Digital Library Services and allow users to consume such services.

1.2: The Project

The project is to develop the software system of a digital library system that organizations such as a university may utilize to provide Digital Library Services and allow users to consume such services.

The design of the system's software was made with the following basic requirements in mind:

- Creation and management of a digital library (This includes addition, removal or editing of contents in a digital library).
- Provision of secured access to contents contained in the digital library to prevent infringement/violation of copyrighted contents.
- Maintenance of a digital library (This includes backup/restoration and cleaning of a digital library database).
- Provision of a client application to consume the digital library service.

In addition to the above basic requirements, the software sub-systems are to be designed in such a way that they will be open to varying implementations of other subsystems.

1.3: Benefits of the Project

The merits of developing a digital library System are too numerous to be mention in an introduction. The following is a list containing five (5) major benefits:

- Provides faster and more efficient means of easily storing and rapidly accessing of information in various formats

- Minimizes the cost of library and information management and in some instances, do away with such cost
- Provides increased accessibility of information to users especially those that may not be traditional patrons of a library, due to geographic location or organizational affiliation
- Allows a multiple of institutions and organizations use the same resources simultaneously
- Provides faster and more advanced methods of finding information
- Eliminates the problem of material deterioration that occurs in traditional library systems due to usage over a long period

1.4: Methodology

The design used for the system is based on hybrid of a Client-Server architecture, a 3-tier system and a Service based Oriented systems. The Database (serving as a Digital Library) will form the first level. An application responsible for control and logic will operate at the middle and will be hosting the Digital Library Service while client applications that consume the Digital Library Service shall represent the final tier. For economical reasons, the Database Tier and Application Server Tier were designed to reside on the same physical system while the Client Tier was developed to be available on client stations.

The implementation/programming of the system were carried out using the following tools:

- Visual Studio 2010 as the Integrated Development Environment used for design, modeling and coding.

- C# programming language (version 4.0) as the main coding language.

- The .Net Framework (version 4.0) as a hosting framework and source of base classes.
- .Net Remoting for sub-system communication.
- Windows Presentation Foundation (WPF) for implementing the user interface.

1.5: Scope and Limitations

The scope of this project is to design and develop the software applications of a digital library system. The project will not provide or create a digital library, however, for presentation purposes; a sample digital library will be included.

1.6: Report Summary

The main chapters in this report are:

2): Background Information and Concepts: This chapter puts forward the outcome of the research work carried out before the design process began.

3): Design and Implementation: This chapter describes the processes taken in building the software system.

Part A (Design): this part describes the process of creating the blueprint of the system.

Part B (Development): this part uses the blueprint created in Part A to implement system components and assemble them.

4): Testing and Analysis: this chapter tests the functionality implementation of the applications and analyses them to generate a minimum computing system requirement.

5): Conclusion: examines the project specification and decides whether the completed project fulfills the criteria. Any possible future work were also discussed in this chapter.

CHAPTER TWO
BACKGROUND INFORMATION & CONCEPTS

2.1: Libraries

The term Library denotes a collection of sources, resources, and services, and the structure in which they are housed; Libraries are organized for use and maintained by public bodies, institutions, or a private individuals [2]. In the more traditional sense, a library is a collection of books, the building or room that houses such a collection, or both.

The central mission of a library is to collect, organize, preserve, and provide access to knowledge and information. In fulfilling this mission, libraries preserve a valuable record of culture that can be passed down to succeeding generations. They form an essential link in this communication between the past, present, and future. The first libraries date as far back as 3000 BC [3] and were composed for the most part, of published records, a particular type of library called archives. These archives were made up almost completely of the records of commercial transactions or inventories, with only a few documents touching theological matters, historical records or legends.

Modern Libraries contain materials on a wide range of subjects; they are repositories and access points for not only printed materials such as manuscripts, books, newspapers, and magazines, but also art reproductions, films, sound/video recordings, maps, photographs, computer software, online databases, and other media. Libraries are now providing public facilities to allow access to their electronic resources and the Internet.

Libraries have always struggled against the physical destruction of their collections. Fires, floods, earthquakes, and wars have damaged the holdings of countless libraries,

Libraries have always struggled against the physical destruction of their collections. Fires, floods, earthquakes, and wars have damaged the holdings of countless libraries, destroying forever much of the recorded history of human civilization. Library materials also fall victim to slow decay caused by acid content in paper, insect infestation, improper storage or handling, and excessive heat, mildew, humidity, and air pollution. The slow decomposition of library materials is a universal problem, occurring on a massive scale in both developing and industrialized countries [2].

Electronic sources of information and low-cost microcomputers have introduced unprecedented changes to the services and operations of modern libraries. Computing trends that began few decades ago have enabled low-cost digital storage of information, rapid transmission of data across computer networks, and sophisticated retrieval and processing of electronic documents and information [3]. These changes—especially the rapid spread of the Internet—have reshaped the feasibility and economics of information distribution so radically that they have permanently altered the ways in which librarians perform their work. Against this background of increased information availability and technological innovation, libraries are developing new, at times revolutionary, methods of providing users with access to an ever-expanding amount of information.

2.2: Digitization and Digital Libraries

To ensure that library materials remain available to present and future generations of library users, libraries have begun digitizing their contents. Additionally, most of today's authors and producers provide their works in digital format in form of e-books, electronic documents and digital media. The collecting, organizing and storing of these digital contents

implies the creation of digital libraries [2]. Making available to users these digital contents in a controlled manner requires a form of digital library service.

2.2.1: ON-GOING DIGITIZATION PROJECTS

In the past few years, procedures for digitizing books at high speed and comparatively low cost have improved considerably with the result that it is now possible to plan the digitization of millions of books per year for creating digital libraries [4].

Large-scale digitization projects are underway at Google, the Million Book Project, and Internet Archive. With continued improvements in book handling and presentation technologies such as optical character recognition and e-books, and development of alternative depositories and business models, digital libraries are rapidly growing in popularity as demonstrated by Google, Yahoo!, and MSN's efforts. Just as libraries have ventured into audio and video collections, so have digital libraries such as the Internet Archive [4].

2.2.2: THE FUTURE OF LIBRARIES AND INFORMATION IN GENERAL

Daniel Akst, author of The Webster Chronicle, proposes that "the future of libraries—and (that) of information—is digital." [5] Peter Lyman and Hal Varian, information scientists at the University of California, Berkeley, estimate that "the world's total yearly production of print, film, optical, and magnetic content would require roughly 1,5 billion gigabytes of storage" [4]. Therefore, they believe that "soon it will be technologically possible for an average person to access virtually all recorded information."

However, archiving remains a problem. According to Larry Lannom, Director of Information Management Technology at the nonprofit Corporation for National Research

Initiatives, "all the problems associated with digital libraries are wrapped up in archiving." He goes on to state, "If in 100 years people can still read your article, we'll have solved the problem." [5]

2.2.3 Advantages in Providing Digital Library Service

- Increased accessibility to users. They also increase availability to individuals who may not be traditional patrons of a library, due to geographic location or organizational affiliation.
- The advantages of digital libraries as a means of easily storing and rapidly accessing of information in various formats are now widely recognized.
- A traditional library must spend large sums of money paying for staff, book maintenance, rent, and additional books. Digital libraries may reduce or, in some instances, do away with these fees.
- A multiple of institutions and patrons can use the same resources simultaneously. This may not be the case for copyrighted material: a library may have a license for "lending out" only one copy at a time; this is achieved with a system of digital rights management where a resource can become inaccessible after expiration of the lending period or after the lender chooses to make it inaccessible (equivalent to returning the resource).
- The user is able to use any search term (word, phrase, title, name or subject) to search the entire collection. Digital libraries can provide very user-friendly interfaces, giving clickable access to its resources.

2.2.4 Setbacks in Providing Digital Library Services

Providing Digital library Services has some challenges; the major of which are Copyright and licensing, Digital Preservation, Metadata creation and (in developing countries) adequate system infrastructure.

- **Copyright and licensing**: Some people have criticized that digital library systems are hampered by copyright law, because works cannot be shared over different periods in the manner of a traditional library. The republication of material on the Web by libraries may require permission from rights holders, who have conflict of interest with publishers who may wish to create online versions of their acquired content for commercial purposes. Some digital library projects, such as Project Gutenberg, work to digitize out-of-copyright works and make them freely available to the public. Some digital library service providers acquire a license to "lend out" their resources. This may involve the restriction of lending out only one copy at a time for each license, and applying a system of digital rights management for this purpose.
- **Digital preservation**: Digital preservation aims to ensure that digital content and information systems are still interpretable into the indefinite future. Each necessary component often must be migrated, preserved or emulated. Only where the meaning and content of digital media and information systems are well understood is migration possible, as is the case for office documents.
- **Metadata creation**: In traditional libraries, the ability to find works of interest is directly related to how well they were catalogued. While cataloguing electronic works digitized from a library's existing holding may be as simple as copying a record from the print to the electronic item, with complex and born-digital works requiring substantially more effort. To handle the growing volume of electronic publications, new tools and technologies have to be designed to allow effective automated semantic classification and searching. While full text search can be used

for some searches, there are many common catalog searches, which cannot be performed using full text.

- **Infrastructure in developing communities**: In developing communities, lack of adequate computing and networking infrastructure greatly hampers the availability and popularity of Digital Library Systems. The fact is that the concept of a Digital Library System still seems futuristic to most Institutions. However, as more and more potential consumers of Digital Library Services are becoming acquainted with computing technologies coupled with the fact that high- speed networking is becoming popular, it will soon become unavoidable for potential Digital Library Service Providers to invest in the system.

2.3: Cataloging Systems

A library catalog is an index to the library's collection that enables a user to find materials. Library users can determine whether the library owns the material that they need by searching through catalog records. Conventional catalog records typically list the item's author, its title, its subjects, the date it was published, the name of its publisher, and other information.

Cataloging codes set standards for the types of information that a catalog should include and for the format in which that information should be presented. Establishing consistency in the content and format of catalog descriptions simplifies the user's search for library materials. Additionally, Standardizing catalog descriptions will allow separate Digital Library Service Providers to share information about their collections with one another [3].

Part of the work in this project is to define a cataloging system that shall be used in designing the database structure. Although there are different cataloging standards already

available for conventional libraries that could easily be ported to the Digital Library System, the author has chosen to create a cataloging system whose aim is to be simple yet expandable.

2.4: Designing Software Systems

In this section, the author's research into Software engineering principles to be considered and used in implementing the software system is presented. It starts by analyzing different software engineering pattern/styles at the end of which a hybrid of these would be chosen in the design process in Part A of Chapter 3. The second part of this section explains the tools and concepts used to implement the project in Part B of Chapter 3.

2.4.1: Software Architectural Patterns and Types of Applications

Software engineering encompasses the set of significant decisions about the organization of a software system including the selection of the structural elements and their interfaces by which the system is composed; behavior as specified in collaboration among those elements; composition of these structural and behavioral elements into larger subsystems; and an architectural style that guides this organization.

2.4.1.1: Software Architectural Styles

In Software engineering, architectural styles, sometimes called software architectural patterns, are set of principles— coarse grained pattern that provides an abstract framework for a family of systems. In terms of pattern, software architectural style is a family of systems with a structured organization. In the following sub-sections, I shall discuss the various architectural styles I considered before the design process began.

A Client/Server Architectural Style

The client/server architectural style describes distributed systems that involve a separate client and server system, and a connecting network. The simplest form of client/server system involves a server application that is accessed directly by multiple clients, referred to as a 2-Tier architectural style.

Historically, client/server architecture indicated a graphical desktop UI application that communicated with a database server containing much of the business logic in the form of stored procedures, or with a dedicated file server. More generally, however, the client/server architectural style describes the relationship between a client and one or more servers, where the client initiates one or more requests (perhaps using a graphical UI), waits for replies, and processes the replies on receipt. The server typically authorizes the user and then carries out the processing required to generate the result. The server may send responses using a range of protocols and data formats to communicate information to the client [1].

N-Tier/3-Tier Architectural Style

N-tier and 3-tier are architectural deployment styles that describe the separation of functionality into segments with each segment being a tier that can be located on a physically separate computer. Generally, they make use of platform specific methods for communication instead of a message-based approach.

N-tier application architecture is characterized by the functional decomposition of applications, service components, and their distributed deployment, providing improved scalability, availability, manageability, and resource utilization. Each tier is completely independent from all other tiers, except for those immediately above and below it. The nth tier only has to know how to handle a request from the n+1th tier, how to forward that

request on to the n-1th tier (if there is one), and how to handle the results of the request. Communication between tiers is typically asynchronous in order to support better scalability. [1]

Object-Oriented Architectural Style

Object-oriented architecture is a design paradigm based on the division of responsibilities for an application or system into individual reusable and self-sufficient objects, each containing the data and the behavior relevant to the object. An object- oriented design views a system as a series of cooperating objects, instead of a set of routines or procedural instructions. Objects are discrete, independent, and loosely coupled; they communicate through interfaces, by calling methods or accessing properties in other objects, and by sending and receiving messages. [1]

Service Oriented Architectural Style

Service-oriented architecture (SOA) enables application functionality to be provided as a set of services, and the creation of applications that make use of software services. Services are loosely coupled because they use standards-based interfaces that can be invoked, published, and discovered. Services in SOA are focused on providing a schema and message-based interaction with an application through interfaces that are application scoped, and not component or object-based.

2.4.1.2 Application Architectures and Deployment Strategies

After going through different materials to help me in determining the appropriate application type for the applications in the Digital Library System, I came to realize that the system's requirements, technological constraints and the type of user experience that is to be delivered are what determine the application type and deployment strategy to employ. I shall

briefly discuss the two application architectures. The first is the service application architecture and the deployment strategy. The second is The Rich Client Application deployment strategy. I believe that these two best fit the scenario. [1]

Service Application Architecture

A service is a public interface that provides access to a unit of functionality. Services literally provide some programmatic service to the caller, who consumes the service. Services are loosely coupled and can be combined within a client, or combined within other services, to provide functionality that is more complex. Services are distributable and can be accessed from a remote machine as well as from the machine on which they are running. Services are message-oriented, meaning that service interfaces are defined by a Web Services Description Language (WSDL) file, and operations are called using Extensible Markup Language (XML)-based message schemas that are passed over a transport channel. Services support a heterogeneous environment by focusing interoperability at the message/interface definition. If components can understand the message and interface definition, they can use the service regardless of their base technology. Figure 2.1 shows an overall view of a typical service application architecture. [1]

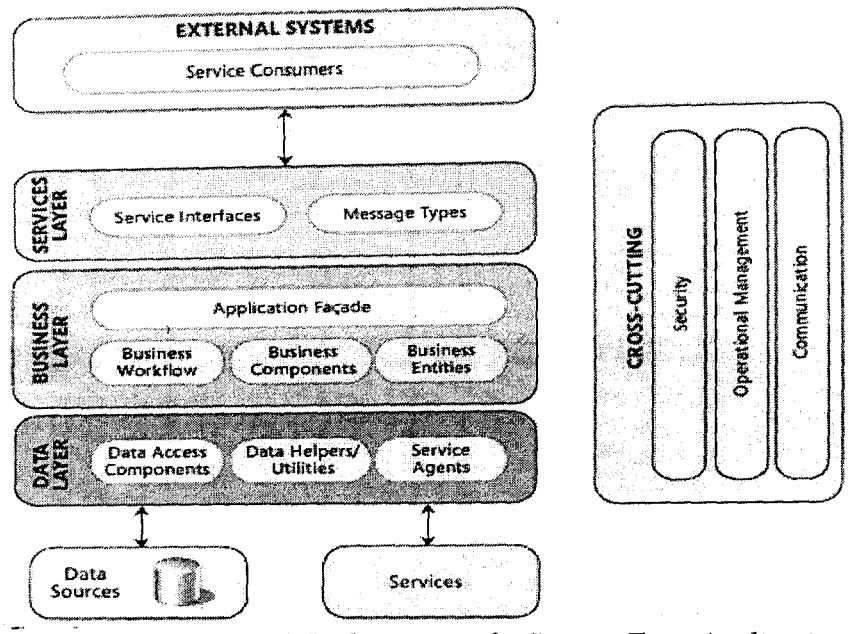


Fig 2.1 Architecture of a Service Type Application

A typical services application is composed of three layers: the service layer, business layer, and data layer. The service layer may include service interfaces, message types, and data types components; the business layer may include business logic, business workflow, and business entity components; and the data layer may include data access and service agent components.

Services are flexible by nature and can be used in a wide variety of scenarios and combinations. The following are typical scenarios:

- A Service exposed over the Internet.
- Service exposed over an intranet..
- Service exposed on the local machine.
- Mixed scenario.

[1]

Rich Client Application Architecture

Rich client UIs can provide high performance, interactive, and rich user experiences for applications that must operate in stand-alone, connected, occasionally connected, and disconnected scenarios. Windows Forms, Windows Presentation Foundation (WPF), and Microsoft Office Business Application (OBA) development environments can be used to quickly and easily build rich client applications.

While these technologies can be used to create stand-alone applications, they can also be used to create applications that run on the client machine but communicate with services exposed by other layers (both logical and physical) and other applications that expose operations the client requires. These operations may include data access, information retrieval, searching, sending information to other systems, back up, and related activities. Fig 2.2 shows an overall view of typical rich client architecture, and identifies the components usually found in each layer.

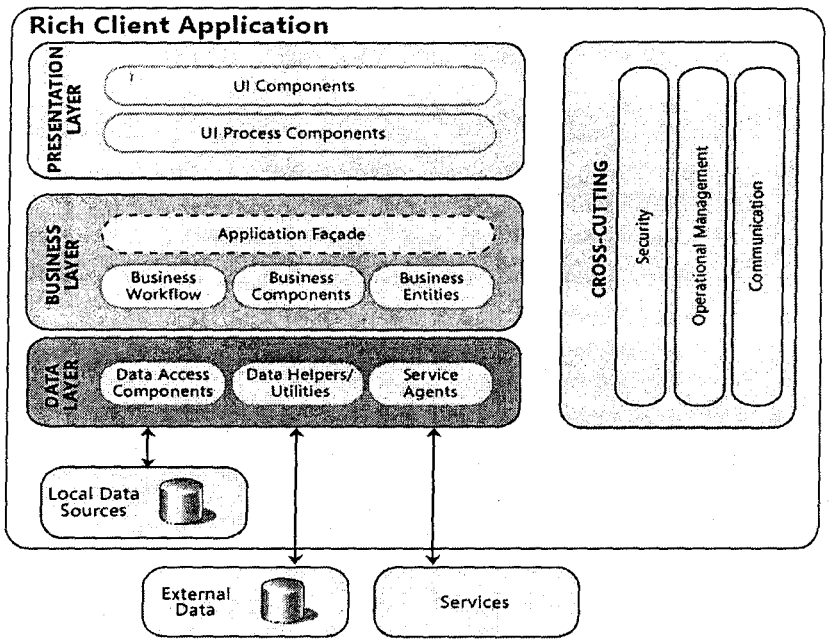


Fig 2.2 Architecture of Rich Client Applications

A typical rich client application is decomposed into three layers: the presentation layer, business layer, and data layer. The presentation layer usually contains UI and presentation logic components; the business layer usually contains business logic, business workflow and business entity components; and the data layer usually contains data access and service agent components.

Rich client applications may be thin applications consisting of mainly a presentation layer, which access a remote business layer hosted on server machines through services. An example of this is a data entry application that sends all of the data to the server for processing and storage.

At the other end of the scale, they may be complex applications that perform most of the processing themselves and only communicate with other services and data stores to consume or send back information. [1]

2.5 DEVELOPMEN TOOLS AND CONCEPTS

This section shall be describing the tools chosen and the possible alternatives based on the research work carried out. I shall also portray core concepts used in either design, development or during testing of the system.

2.5.1: CHOOSING THE DEVELOPMENT PLATFORM AND FRAMEWORK

Initial research in the project focused on identifying which platform to use in developing the application. It was necessary to identify and define the key goals of the project and then choose the platform that would enable those goals to be achieved.

To attempt to develop all the components of the system from scratch would be time consuming and unnecessary since there are frameworks already available that contain most

basic components. Two major frameworks were considered, Sun's Java Framework and Microsoft's .Net Framework. Both of this have very similar base components but slightly differ in implementation. The Java Framework, although being renowned for its portability across platforms, lacked in speed, efficiency and quality in user interface presentation as compared with the .net Framework. The .Net framework is however known to be Microsoft Operating System (OS) specific (although other OS's have light versions of the .net framework). The .Net framework was finally chosen as it performs excellently on about 95% of client systems (i.e. Microsoft's operating systems) as compared with the java framework that works fairly well on every platform. [1]

The .Net Framework

The .NET framework is a software framework that is readily available on most Microsoft Windows operating systems. It includes a large library of pre-coded solutions to common programming problems. It also contains a virtual machine that manages the execution of programs written specifically for the framework. The pre-coded solutions that form the framework's Base Class Library cover a large range of programming needs in a number of areas, including user interface, data access, database connectivity, cryptography, web application development, numeric algorithms, and network communications. The class library will be used in combination with the my code to produce the applications. The main languages that use the framework are Visual basic, Visual C++, Visual C# and F#. [8]

Programs written for the .NET Framework execute in a software environment that manages the program's runtime requirements. Also part of the .NET Framework, this runtime environment is known as the Common Language Runtime (CLR). The CLR provides the appearance of an application virtual machine so that programmers need not

consider the capabilities of the specific CPU that will execute the program. The CLR also provides other important services such as security, memory management, and exception handling. The class library and the CLR together constitute the .NET Framework. [8]

2.5.2: CHOOSING THE PROGRAMMING LANGUAGE

After choosing to use the .net framework, the choice of a programming language was another issue. After little research, it was found that Visual C# is the programming language that most directly reflects the underlying Common Language Infrastructure (CLI) of the .net framework. Most of its intrinsic types correspond to value-types implemented by the CLI framework. [1]

C# Programming Language

C# is a multi-paradigm programming language encompassing imperative, declarative, functional, generic, object-oriented (class-based), and component-oriented programming disciplines. It was developed by Microsoft within the .NET initiative and later approved as a standard by Ecma (ECMA-334) and ISO (ISO/IEC 23270). C# is one of the programming languages designed for the Common Language Infrastructure. C# is intended to be a simple, modern, general-purpose, object-oriented programming language. The most recent version is C# 4.0, which was released on April 12, 2010 .[6]

However, the language specification does not state the code generation requirements of a C# compiler: that is, it does not state that a C# compiler must target a Common Language Runtime, or generate Common Intermediate Language (CIL), or generate any other specific format. Therefore, a C# compiler could generate machine code like traditional compilers of C++ or FORTRAN.

2.5.3: CHOOSING THE PRESENTATION TECHNOLOGY

The .net framework offers two distinct presentation models. One is the Forms Presentation model that is based on the Graphics Device Interface (GDI) systems and the other is the Windows Presentation Foundation (WPF). After researching both technologies, I outlined the following differences:

- Windowing: Applications built using the GDI API use many windows, and reside under a parent window (MDI). Applications built in WPF have one window.
- Control Positioning: Applications built with GDI+ use absolute positioning. When the parent is resized, the child elements are not resized along with it. Applications built in WPF can use absolute, dynamic or data-bound positioning. With either absolute or dynamic positioning, the controls are positioned relative to the parent element. This allows the UI to adapt itself to different screen sizes
- Rendering Mode: With GDI and GDI+, rendering uses immediate rendering: the application repaints the area that becomes invalidated. In WPF, rendering uses retained rendering: the application keeps track of the drawing information but the system performs the painting.
- Event handling: GDI and GDI+ use events and eventhandler delegates using subscription/notification. WPF uses bubbling, tunneling, and direct event notification, and the event travels up and down the VisualTree.
- Video and Audio: Direct support for video and audio is not provided by GDI+ or Windows Forms, but must be obtained through a media player like Windows Media Player. WPF supports video and audio directly.

In addition to the above WPF provides developers with a unified programming model for building rich Windows smart client user experiences that incorporate UI, media, and documents. The core of WPF is a resolution-independent and vector-based rendering engine that is built to take advantage of modern graphics hardware. WPF extends the core with a comprehensive set of application-development features that include Extensible Application Markup Language (XAML), controls, data binding, layout, 2-D and 3-D graphics, animation, styles, templates, documents, media, text, and typography. WPF is included in the Microsoft .NET Framework, therefore building applications that incorporate other elements of the .NET Framework class library becomes easy.

Hence, the Windows Presentation Foundation model was chosen based on the above assessment. [8]

2.5.4: THE .NET REMOTING COMMUNICATION FRAMEWORK

.NET Remoting is a Microsoft Application Programming Interface (API) for interprocess communication.[8] .NET Remoting allows an application to make an object (termed remotable object) available across remoting boundaries, which includes different appdomains, processes or even different computers connected by a network. The .NET Remoting runtime hosts the listener for requests to the object in the appdomain of the server application. At the client end, any requests to the remotable object are proxied by the .NET Remoting runtime over Channel objects that encapsulate the actual transport mode, including TCP streams, HTTP streams and named pipes. As a result, by instantiating proper Channel objects, a .NET Remoting application can be made to support different communication protocols without recompiling the application. The runtime itself manages the act of serialization and marshalling of objects across the client and server appdomains.

CHAPTER THREE
DESIGN AND IMPLEMENTATION

3.0: Introduction

The background research that was undertaken to determine the platform to develop on, which language to code in, the presentation model to use and the communication system to adapt; was crucial in determining how to approach the design. This chapter describes the processes taken in building the software system.

3.1: System Design Requirements

Before any design process begins, it is crucial to know the requirements of the system. The requirements of the system, deduced from the objectives of the project, are here stated.

1. Create a Digital Library Service Application with the following requirements:
   - Provide security system to allow only managing personnel access to the digital library.
   - Allow Digital Library Service managers to edit, manage and search trough a database of digital contents. Editing includes creating and removing contents, and editing their information. Managing involves backing up and restoring of the database.
   - Allow Digital Library Service Providers to maintain and configure their digital libraries services network availability options.
   - Allow digital contents and their information/description to be accessed via a network and provide means of securing access to the Digital Library Service from the client stations.

2. Create a Digital Library Client Application with the following requirements:
   - Provide security system to allow only registered users login to a digital library service.
   - Retrieve content information and allow users to search the database for desired content.
   - Internally render the contents in the application. This will include both books and multimedia contents such as video, audio and still images.
   - Allow users to perform activities such as storing preferences, selecting favorites, rating contents and creating bookmarks that allow users to continue where they left off.

3.2: Proof of concept

Rather than begin to develop the application immediately, the decision was made to write a small version of the software. The purpose of this was to test the three basic concepts of the system:

1. Storing of digital contents and their information and accessing them uniformly: This was achieved by creating a table in a Database to hold the content metadata and a Content Store to store the contents on disk. A class was then created to unify access to both metadata and content as if they were from the same location.
2. Transferring digital contents and information between two processes of different application domain: This was achieved by transferring a file using I/O streams and composing the file at the receiving end.

3. Displaying of received digital content and its associated information: Books in the .xps format were displayed using WPFs document viewer while multimedia content were rendered using WPFs Media element control.

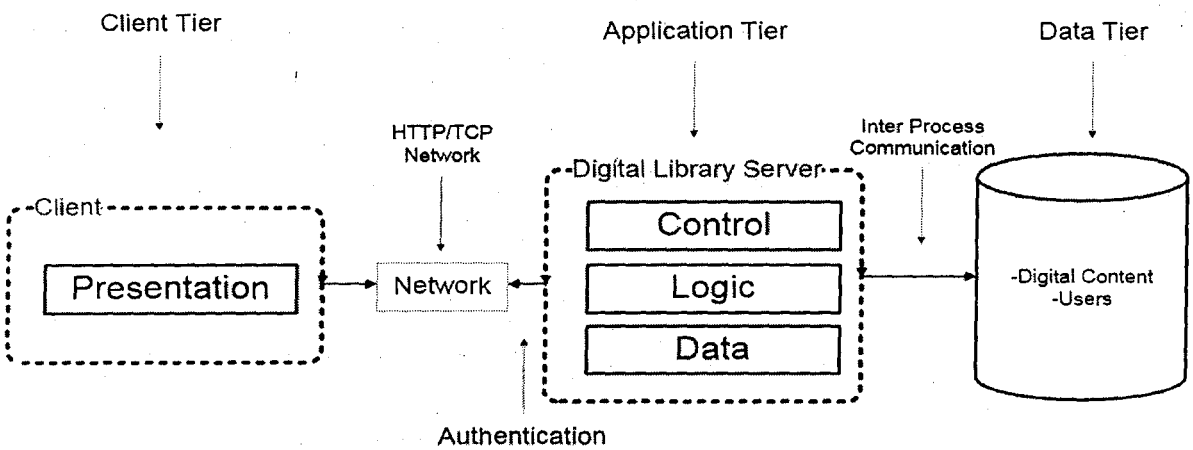
After succeeding in achieving the above three simulations, a decision was then taken on how to approach the design of the system.

3.3: Part A: DESIGN

Based on the above system requirements, the design process was conducted in a top-down fashion. It started with a conceptual design of the system; the system blueprint was then deduced after which each of the sub-components was designed. Finally, these sub-components were linked together to make the system perform the tasks required.

3.3.1: CONCEPTUAL DESIGN

The System's Design Concept is based on a 3-Tier Architecture shown in fig.3.1. The system consists of three (3) tiers, the Data Tier, Application Tier and the Client Tier. The Data Tier is responsible for data storage and retrieval. It consists of the Data itself and the Service used to access the data. In this case, the Data service is a database service. The



function of this layer is to provide data to the system.

Fig 3.1 System Conceptual Design

The Digital Library Server hosting the Digital Library Service forms the Application Tier. It is at this layer that the Data Service's data schema is converted to client consumable data schema. This Tier provides necessary logic and control of data flow and is the location

where most processing is carried out. The Application Tier and Data Tier will reside on the same physical system. The Application Tier is connected to the Client Tier through a network. This network connection could be HTTP-based or TCP-based. A firewall would exist between the Application Tier and Client Tier but was neglected for simplicity. Client stations collectively represent the Client Tier. This tier will be mainly responsible for data presentation and logic.

3.3.2: SYSTEM DESIGN

Based on the concept described in the previous section, the layout of the system was designed as shown in fig.3.2.

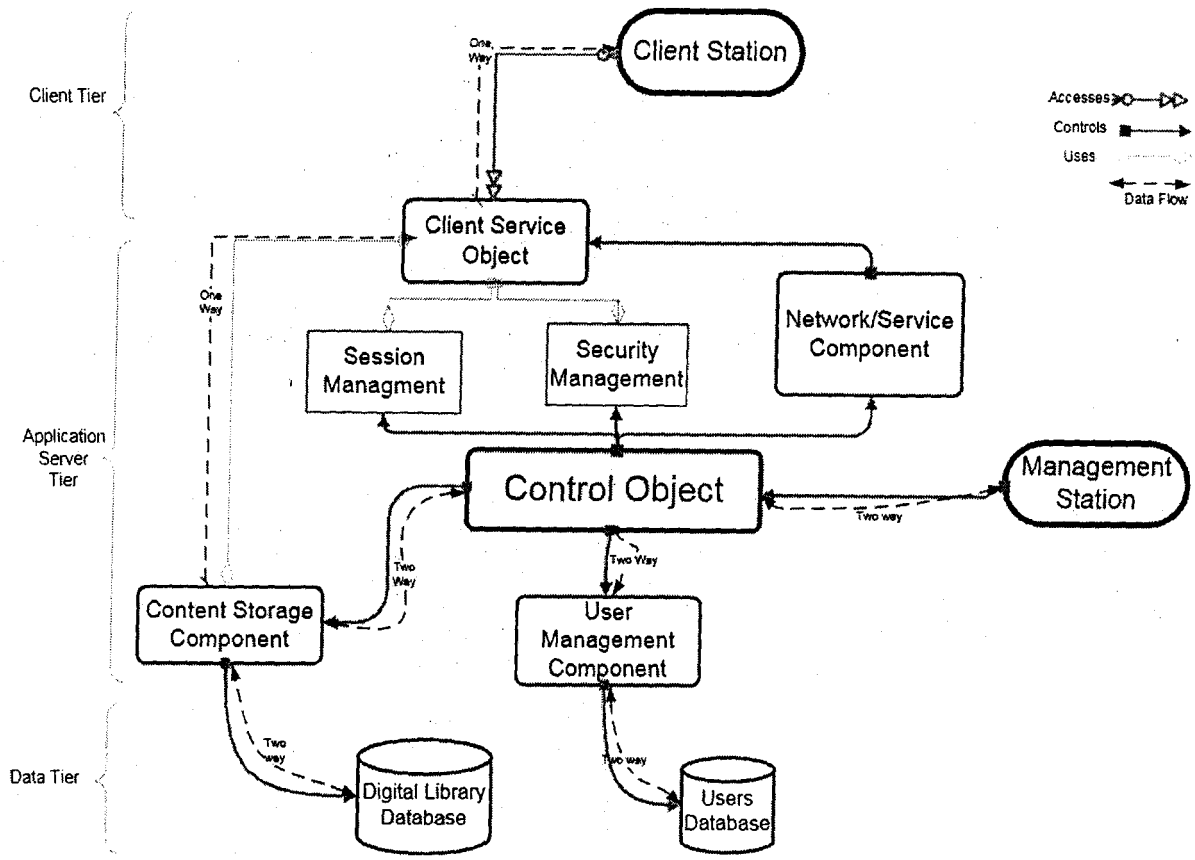


Fig 3.2 Digital Library System Design

The 3-tier structure is maintained while additional details were added. The Data Tier holds both the digital content collection (i.e. the Digital Library) and user registration information. The Users Database is managed via the User Management Component while the Digital Library Database is handled via the Content Storage Component.

The Control object controls and hosts the Digital Library Service through a Client Service Object via the Service/Network Component. Security and Session Management is handled and configured from the Control Object and are implemented using the Client Service Object. The Client Object has read only data reference to the Digital Library via the Content Storage System. The Hosted Client Object is proxied in a SingleCall fashion by Client Stations. This method of sharing is tantamount to a one-to-one communication between the server and client.

The Management station will be used by librarians to control the system. The design specifies that the Management station be at the server. This was done to simplify the system because only one management location is required.

3.3.3: COMPONENT DESIGN

When designing components of a system, the two important criteria to use are functionality and interoperability. This section will explain the design of components of the system from the Data Tier up to the Client Tier.

Digital Library Database Design

Design of the Digital Library Database was carried out in such a way that it will be Database System independent. This was done to allow different organizations use the database systems of they prefer. Storing the contents, which are Large Objects Binary (LOB)

files, in a database present some challenges and is not well supported by all database systems. For this reason, the content information and content were separated from each other. This was archived by storing the information in the database while the contents were stored in packages. Each digital content will be identified by a Global Unique Identifier (GUID). This Unique Identification (UID) method uses a 16-bit, 32 digit code to identify an object and makes the probability of two contents having the same ID less than a trillionth. Therefore identifying which row of information belonged to a particular content became less complicated.

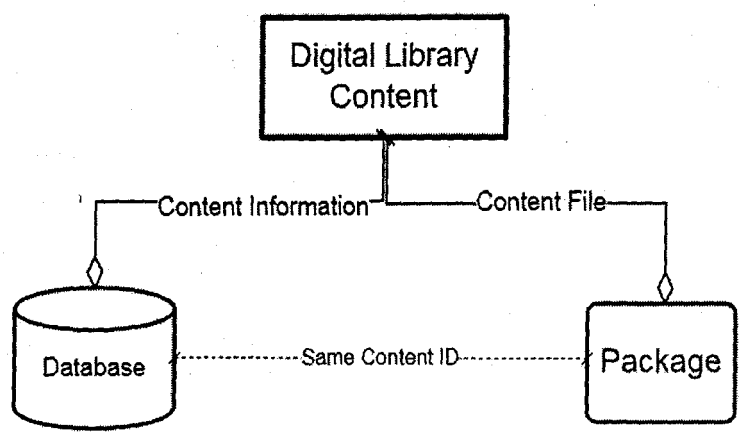


Fig 3.3 Digital Library Content Database Design

Datasets were used to map the database that contained the information. Datasets are objects that contain data tables where you can temporarily store the data for use by the application. If the application requires working with data, the data is loaded into the dataset, which provides the application with a local in-memory cache of the data to work with. This allows the application to work with the data even if the application becomes disconnected from the database. The dataset maintains information about changes to its data so updates can be tracked and sent back to the database when the application reconnects to it.

The structure of a Dataset is similar to that of a relational database; it exposes a hierarchical object model of tables, rows, columns, constraints, and relationships. This allows the application logic to be deigned irrespective of the database system. Fig.3.4 shows the dataset design meant to cache the content information data.

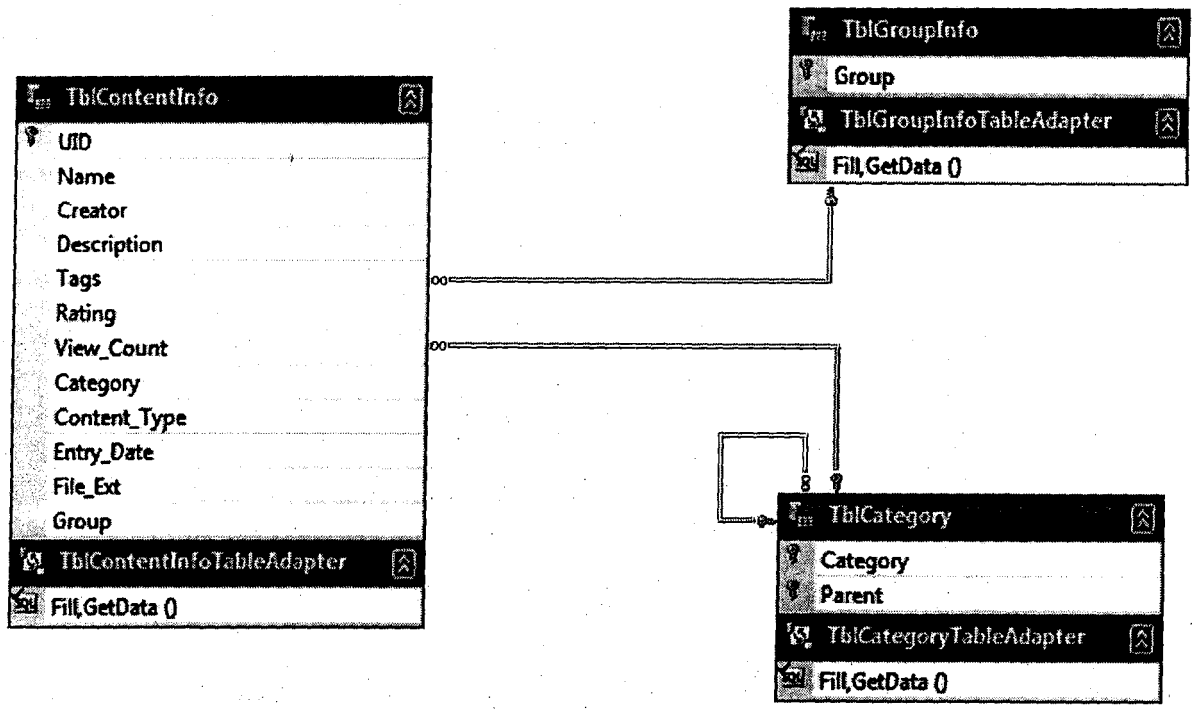


Fig 3.4 Content Dataset Structure

The design consists of three (3) tables. The Content Information Table (TblContentInfo), the Group Information Table (TblGroupInfo) and the Category Table (TblCategory) each with its table adapter used to connect to the database. Below is a detailed explanation of each table and its associated columns.

TblContentInfo (Content Information Table): This table holds all the information about content. It has a foreign key constraint associated with the Group Information Table and the Category Table to control content grouping and categorization. The columns in this table are tabulated below.

Table 3.1 Content Information Table Structure

COLUMN	DATA TYPE	FUNCTION
UID	Guid	Uniquely identifies each content.
Name	String	Represents the name or title of the content.
Creator	String	Represents the name of the author, producer or publisher.
Description	String	Represents a description of the content.
Tags	String	Represent keywords associated with the content.
Rating	Int	To represent the rating of a content.
View_Count	Int	Indicates how many times the content was viewed.
Category	String	Indicate to which category the content belongs.
Content_Type	String	To indicate the type of digital content (i.e. book, video, etc).
Entry_Date	DateTime	Indicates the Date and time the content was included.
File_Ext	String	Store the original File type information of the content.
Group	String	Indicates which group of contents the content belongs.

TblGroupInfo (Group Information Table): this table was created with one unique column (Group with a String data type) to allow multiple contents be in the same group but prevent two groups to have the same name. It has a foreign key constraint from the Content information Table.

TblCategory (Category Information Table): this table has one primary column (Category of String data type) and another column (Parent of string data type) data has a same table foreign key constraint to the -Category column. This was done to allow a category to have other categories within it.

The Class responsible for handling the contents in the package, i.e. the ContentStorage Class, was designed to allow storing, extracting and replacing of contents. In addition, it converts contents to I/O stream or byte array meant for transfer. Its design is discussed in the following section.

Content Storage System

The Content Storage System handles access to the database for the Control Object and Client Objects. These Objects communicate with Content Storage System via methods/function calls. Fig.0 shows the structure of the Content Storage System and its methods. This system has three groups of functions; the group containing the Get-prefixes is used by client service objects while the other two groups are used by the management station via the control object.

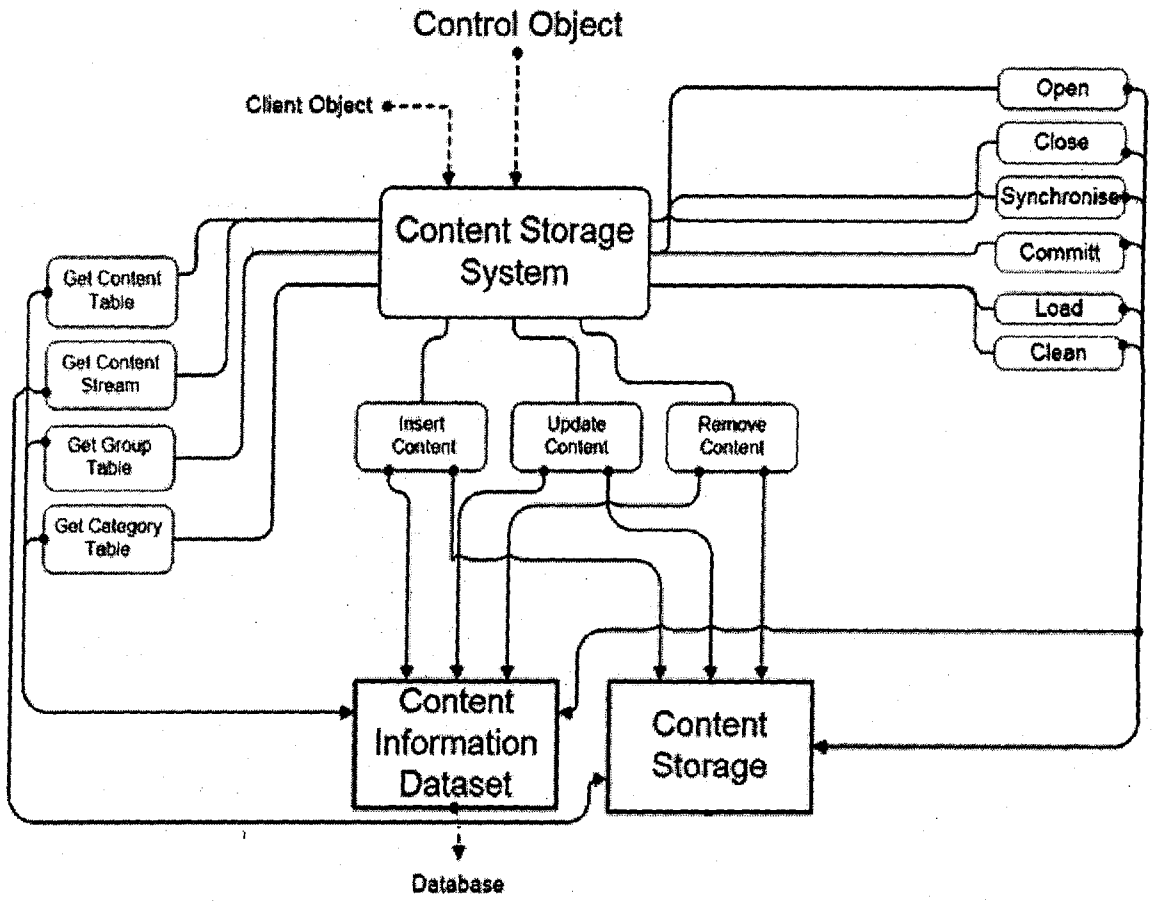


Fig 3.5 Content Storage System Component Design

User Management Component

This component allows the management stations to manage library users' service registration information via the control object. It consists of five methods that make use of the User Database via a dataset. Fig 3.6 illustrates the design.

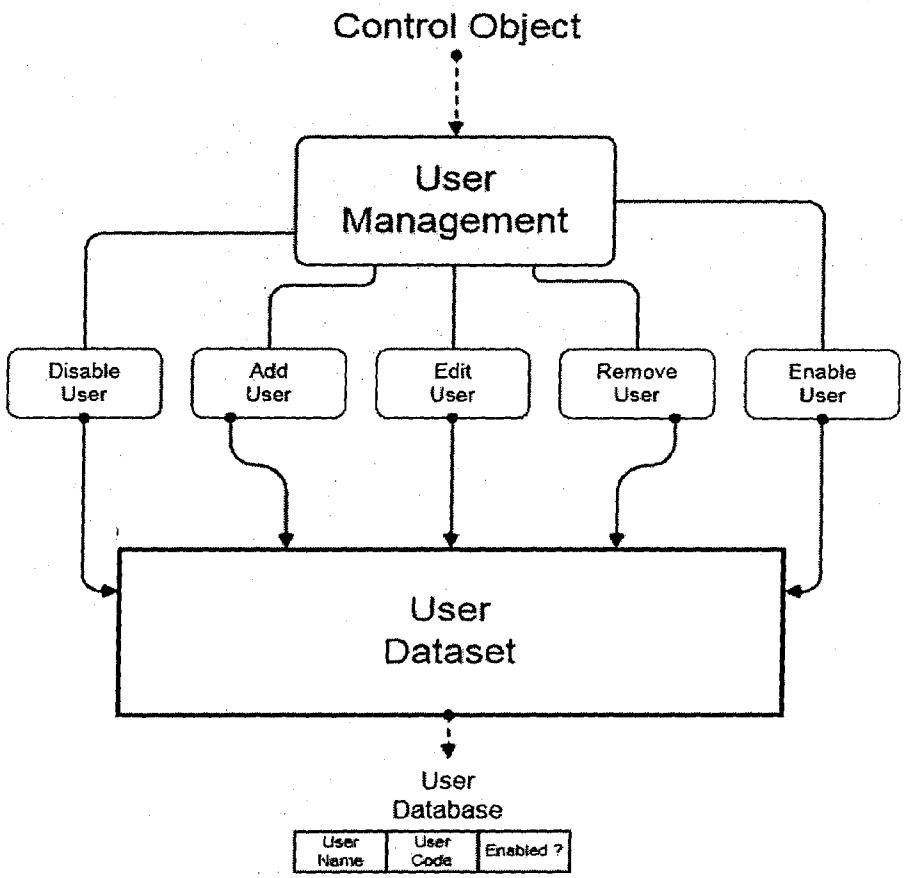


Fig.3.6 User Management Component Design

A user row consists of a User Name column, User Code column of Guid data type and an Enabled column of Boolean data type to control user activation.

Session Management Component

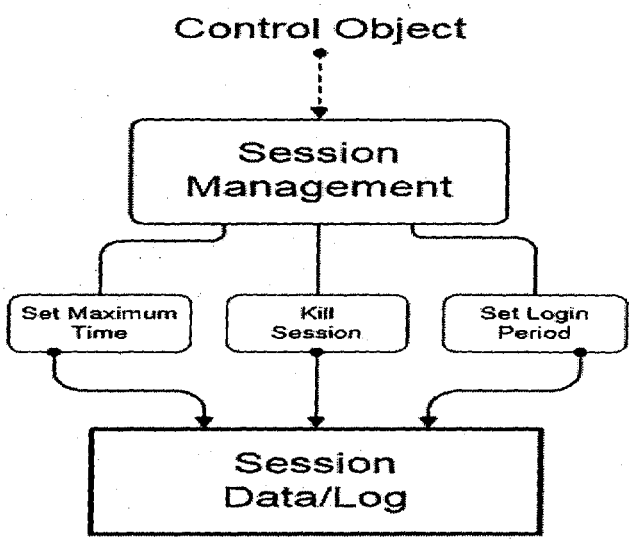


Fig 3.7 Session Management Component Design

The Session Management Component provides administrators with session control tools. It allows administrators to view active session and kill them when necessary. It is also allows administrators to configure maximum session time and login time-frame options.

Security Management Component

The Management station processes client authentication through the security management component and handles authentication throughout the lifetime of a Client object. The Security component was also designed to handle data and communication encryption. Fig.3.8 shows the structure of this component.

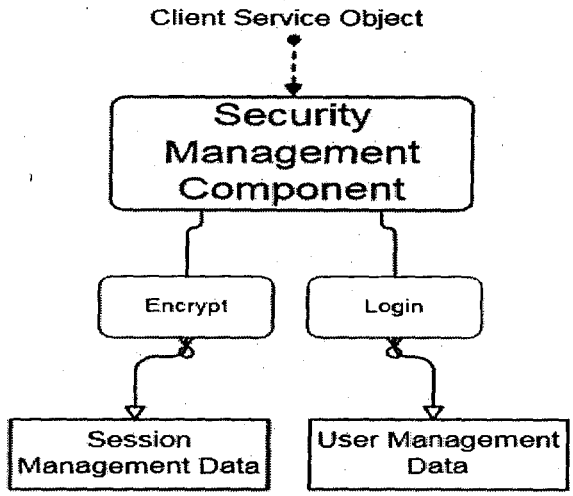


Fig 3.8 Security Management Component Design

Network Service Component

The Network Service Configuration component permits the management station to control Client Service activation via the BeginBroadcast (that accepts a broadcast URL as a parameter) and StopBroadcast. It also gives administrators control over the network protocol to use via the UseHTTPChannel and UseTCPChannel methods. This methods accept a port number that administrators wish to broadcast the service on.

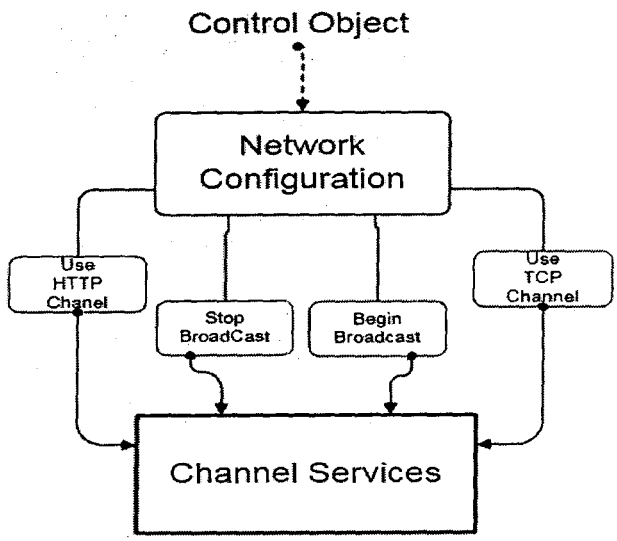


Fig 3.9 Network Configuration Component Design

Client Service Object

The Client Service Object is the component used for communication between the server and

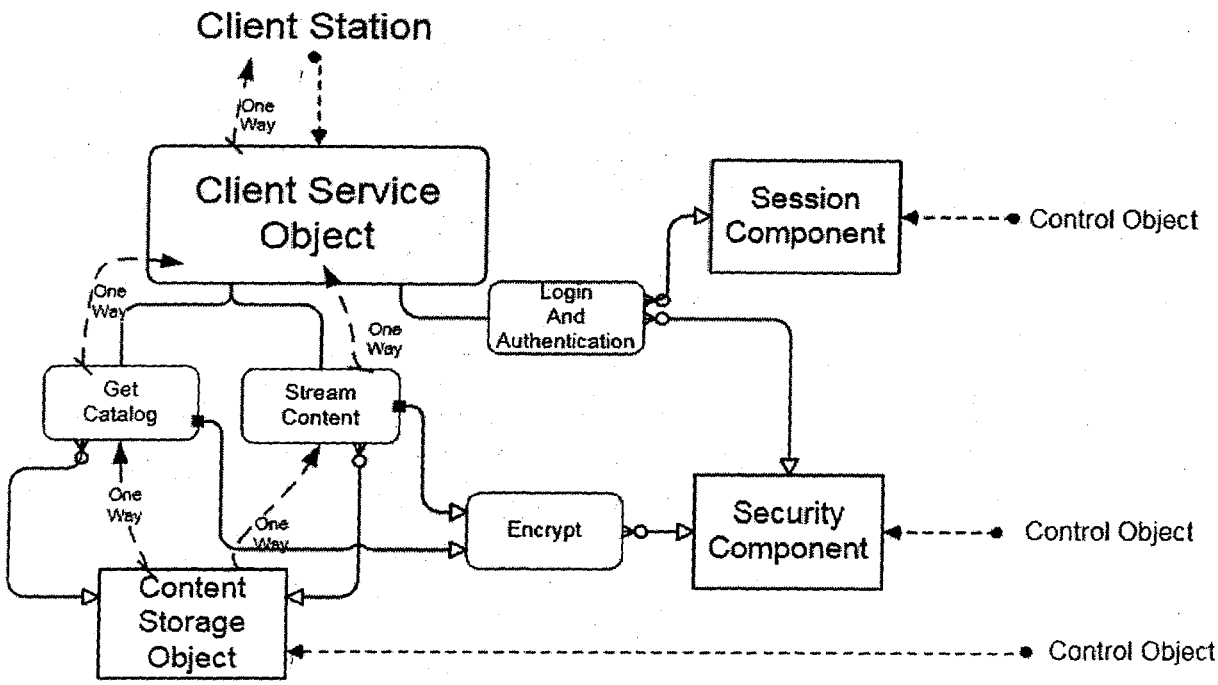


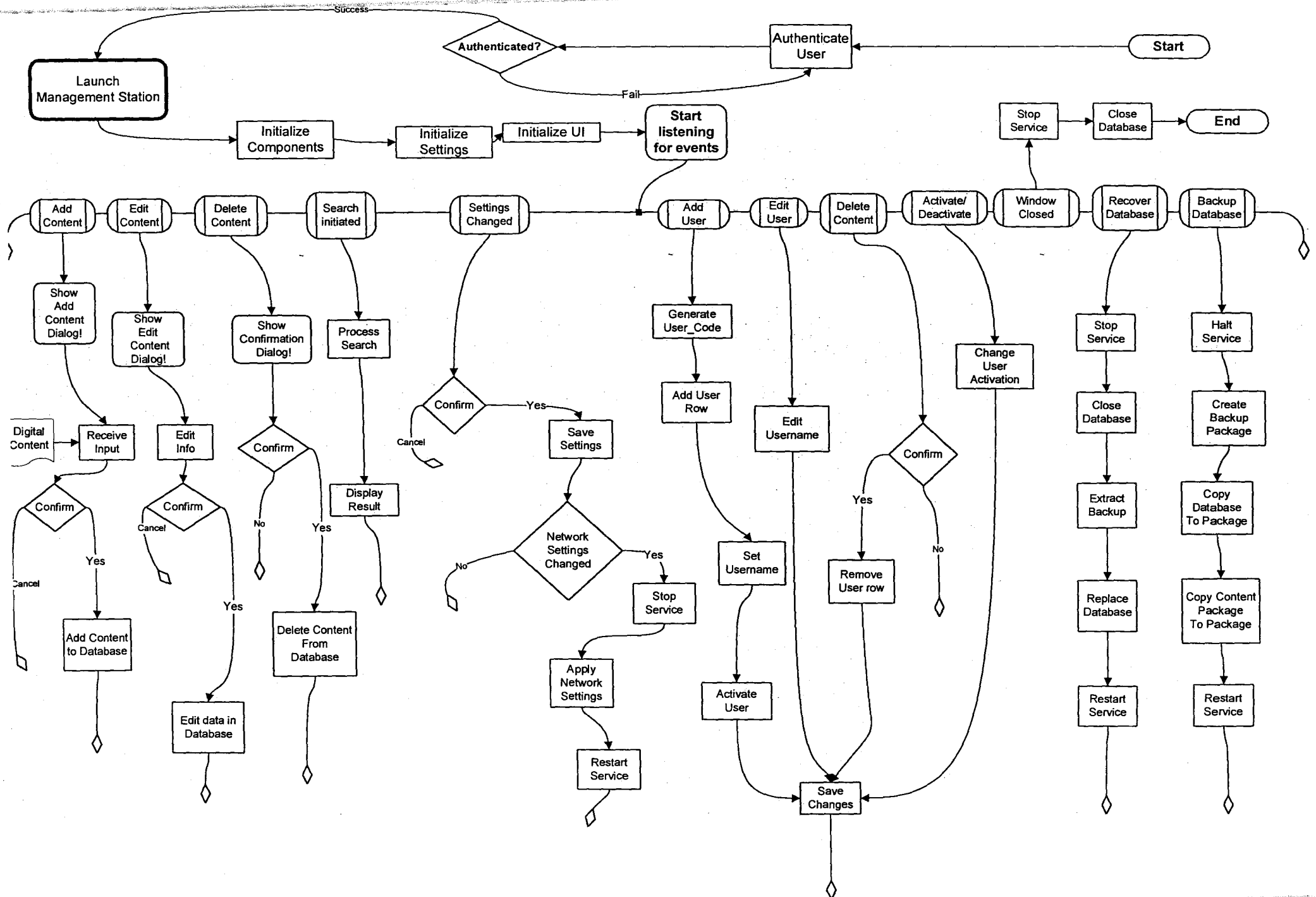
Fig 3.10 Client Service Object Design

the client. It is a marshal-by-reference object created on client stations and whose function calls are executed on the server. Fig.3.10 shows the structure of the object.

3.3.4: APPLICATIONS DESIGN

The System consists of two (2) applications; the Digital Library Management Station and the Digital Library Client Station.

The following pages show the flow chat of the Digital Library Management Station and the Digital Client Station.


Start
Authenticate User
Authenticated?
Success
Fail
Launch Management Station
Initialize Components
Initialize Settings
Initialize UI
Start listening for events
Stop Service
Close Database
End
Add Content
Edit Content
Delete Content
Search initiated
Settings Changed
Add User
Edit User
Delete Content
Activate/ Deactivate
Window Closed
Recover Database
Backup Database
Show Add Content Dialog!
Digital Content
Receive Input
Confirm
Cancel
Yes
Add Content to Database
Show Edit Content Dialog!
Edit Info
Confirm
Cancel
Yes
Edit data in Database
Show Confirmation Dialog!
Confirm
No
Yes
Delete Content From Database
Process Search
Display Result
Confirm
Cancel
Yes
Save Settings
Network Settings Changed
No
Yes
Stop Service
Apply Network Settings
Restart Service
Generate User_Code
Add User Row
Set Username
Activate User
Edit Username
Save Changes
Confirm
Yes
No
Remove User row
Change User Activation
Stop Service
Close Database
Extract Backup
Replace Database
Restart Service
Halt Service
Create Backup Package
Copy Database To Package
Copy Content Package To Package
Restart Service

3.4 Part B: DEVELOPMENT

Development of the system began after each component was designed. The tools used to carry out the development are listed in the table below.

Table 3.2 List of Development Tools

Development Tool	Tool used
Integrated Development Environment	Microsoft's Visual Studio 2010
Programming Language	C# 4.0
Code Framework	.Net Framework (3.5)
Presentation Framework	Windows Presentation Foundation (WPF)

The development process describes the code used to implement the system components which appears in the Code Appendix.

CLIENT SERVICE OBJECT IMPLEMENTATION

- Component Type: **class**
- Access level: **public**
- Inherits: **IClientServiceInterface**
- Variables and Fields:

Name	*Type*	*Access Level*	*Function*
Contents	Content_Storage_System	private	References the ContentStorageSystem Object of the Control component.
Session_Mgr	Session_Managment Component	private	References the Session_Managment Object of the Control component.
Content_Stream	IO.Memory Stream	private	Temporary file stream meant for file streaming

- Methods:

Name	*Access*	*Returns*	*Parameters*	*Function*

Login	public	Session_id	User_code, user_name	Client login
Get_Catalog	public	List<ContentInfo>	Session_id	Retrieve catalog
Begin_StreamContent	public	void	Content_id, Session_id	Copy the digital content to Content_stream
Get_Next_Stream_Packet	public	Stream_Packet	size	Gets the next sequence of stream
End_StreamContent	public	void	void	Nullifies Content_stream
Rate Content	public	void	Rating(int)	Submit content rating

SESSION MANAGEMENT COMPONENT IMPLEMENTATION

- Component Type: **class**
- Access level: **public**
- Inherits: n/a
- Variables and Fields:

Name	*Type*	*Access Level*	*Function*
Active_Sessions	List<Session>	public	Holds current list of sessions
Max_Session_time	int	private	Holds the maximum duration of a session in minutes
Logon_Start_Time	DateTime	private	Represents the start-time of the logon period
Logon_End_Time	DateTime	private	Represents the end-time of the logon period
Max_Users	int	public	Holds the maximum no of user sessions

- Methods:

Name	*Returns*	*Access*	*Parameters*	*Function*
Kill_Session	void	public	session_id(int)	Removes the given session from the session list
Set_Max_Session_time	void	public	Minutes(int)	Sets Max_Session_time
Set_Login_Period	void	public	StartTime, Endtime(Datetime)	Sets login period

Register_Session	session_id	public	User_code(Guid)	Creates a new session for given user
Check	void	public		Check for login expiration

NETWORK CONFIGURATION COMPONENT IMPLEMENTATION

The Network Configuration class was implemented as follows.

- Component Type: **class**
- Access level: **public**
- Inherits: **n/a**
- Variables and Fields:

Name	*Type*	*Access Level*	*Function*
Tcp_Port	int	public	Stores tcp port number
Http_Port	int	public	Stores http port number
Service_Name	string	public	Stores service name for network service addressing
Enable_Http	boolean	public	Indicates if service can be available over the internet

- Methods:

Name	*Access*	*Returns*	*Parameters*	*Function*
Broadcast	public	void	-	Registers channels and marshals the object
StopBroadcast	public	void	-	Unregister channels
Refresh	public	void	-	Re marshals the service object

SECURITY MANAGEMENT COMPONENT IMPLEMENTATION

- Component Type: **class**
- Access level: **public**
- Inherits: **n/a**
- Methods:

Name	*Returns*	*Access*	*Parameters*	*Function*
Login	Session_id	public	User_code, role	Authenticates user and provide session_id
Encrypt_Stream	void	public	Stream reference, session_id	Encrypts a file stream using the session_id

Client Station and Management Station

The Client Station Application and the Management Station implementations have been represented using class diagrams as shown in the pages that follow.

CHAPTER FOUR

TESTING AND ANALYSIS

4.1: Functionality Test

This testing procedure tests the applications to see if it performs each of its basic function. The test is carried out by running the application and performing the tasks one after the other. At the end of each test, the output is compared with expected results. Below is a table of functionality tests carried out and the result of each.

Table 4.1 Digital Library Management Station Functionality Test Result

TEST	RESULT	REMARK
Content Management		
Content Addition	Passed	Ok!
Content Editing	Passed	Ok!
Content Removal	Passed	Ok!
Database Operations		
Database Backup	Passed	Takes longer than desired.
Database Recovery	Passed	Ok!
User Management		
Register User	Passed	Ok!
Unregister User	Passed	Ok!
Activate User	Passed	Ok!
Deactivate User	Passed	Ok!
Security		
Login Authentication	Passed	Ok!
Encrypt Data	Failed	No longer used
Session Management		
Log Session	Passed	Ok!
Kill Session	Passed	Ok!
Reset Max Session Time	Passed	Ok!
Network and Client Services		
Activate Client Service	Passed	Ok!
Close Client Service	Passed	Ok!
Use TCP Channel	Passed	Ok!
Use HTTP Channel	Passed	Ok!

Table 4.2 Digital Library Client Station Functionality Test Result

TEST	RESULT	REMARK
Login		
Login Authentication	Passed	Ok!
Networking		
Connecting via TCP	Passed	Ok!
Connecting via HTTP	Passed	Ok!
Service Login	Passed	Ok!
Service Consumption		
Catalog Retrieval	Passed	Ok!
Content Streaming	Passed	Ok!
Rating Content	?	Not Implemented
Presentation		
Catalog Listing	Passed	Ok!
Categorizing	Passed	Ok!
Searching	Passed	Ok!
Filtering	Passed	Ok!
Content Display		
Book Rendering	Passed	(Only XPS format is supported)
Video Playback	Passed	Ok!
Audio Playback	Passed	Ok!
Image Rendering	Passed	Ok!
User Preferences		
Bookmarking	?	Not Yet Implemented
Favorite Management	?	Not Yet Implemented
Control Panel		
Save Settings	Passed	Ok!
Load Settings	Passed	Ok!
Change Settings	Passed	Ok!

4.2: System Requirement Analysis

The system requirement analysis, were conducted to test the applications minimum and recommended hardware and software requirements.

HARDWARE REQUIREMENTS

The hardware requirement was measured by running the applications in a normal state and taking the record of its footprint via the Performance Counter application that was set to monitor the applications. Below is a sample of data obtained from the performance counter.

Process Performance Report

Process	DigitalLibraryServer	DigitalLibraryx35
% Privileged Time	56.101	37.401
% Processor Time	60.776	46.751
% User Time	4.675	9.350
Creating Process ID	7,872.000	7,464.000
Elapsed Time	25,825.169	862.508
Handle Count	541.000	815.000
ID Process	9,212.000	4,500.000
IO Other Bytes/sec	172,680.059	4,421,045.839
IO Other Operations/sec	46.951	1,006.939
IO Write Bytes/sec	0.000	6,284,834.090
IO Write Operations/sec	0.000	0.999
Page Faults/sec	1,789.115	1,588.326
Page File Bytes	172,068,864.000	140,496,896.000
Page File Bytes Peak	380,518,400.000	222,228,480.000
Pool Nonpaged Bytes	74,848.000	31,992.000
Pool Paged Bytes	452,644.000	480,204.000
Priority Base	8.000	8.000
Private Bytes	172,068,864.000	140,496,896.000
Thread Count	22.000	38.000
Virtual Bytes	460,546,048.000	464,314,368.000
Virtual Bytes Peak	680,747,008.000	531,345,408.000
Working Set	144,670,720.000	126,541,824.000
Working Set - Private	101,412,864.000	76,419,072.000
Working Set Peak	300,027,904.000	165,711,872.000

25/10/2010 9:32AM

Plate 4.1 Sample Digital Library Applications Process Performance Report

After many samples were taking and compared, the required information was deduced based on my experience working with applications.

Table 4.3 Minimum System Applications Requirements

	Digital Library Server	Digital Library Client
Processing Capability	1.5GHz	1GHz
Memory Requirement	300MB	100MB
Disk Space Requirement	10MB+	500MB
Network bandwidth requirements	54Mbps	54Mbps

Table 4.4 Recommended System Applications Requirements

	Digital Library Server	Digital Library Client
Processing Capability	2.0GHz x 2	1GHz
Memory Requirement	500MB	300MB
Disk Space Requirement	10MB+	500MB
Network bandwidth requirements	100Mbps	100Mbps

SOFTWARE REQUIREMENTS

Software requirements of an application indicates the dependency of the application on other software components. In deducing the software requirements, I considered the software and library dependencies. The following is table of the requirements.

Table 4.5 Application Software Requirements

	Digital Library Server	Digital Library Client
Supported Operating Systems	-Windows XP (SP3) -Windows Vista (SP2), -Windows 7	-Windows XP (SP3) -Windows Vista (SP2), -Windows 7
Minimum Framework version	.Net Framework 3.5(SP1)	.Net Framework 3.5
Required Libraries:	-WPFToolkit.dll -PackageManager.dll	

CHAPTER FIVE

CONCLUSION

After reviewing the result of the tests conducted in chapter 4, it can be concluded that the aims of the project were achieved. Therefore, the software system of a Digital Library System designed and implemented in this project can successfully be used to provide and consume digital library services.

Hence, it will now be possible for patrons of libraries to search and gain access to information faster and more efficiently regardless of their geographical location. For Libraries, this means minimization of cost and possibilities of sharing their resources with other libraries.

5.1 Problems Encountered

From the result obtained in section 4.1, the major problem encountered was that of encrypting content streams. The reason was that the initial implementation was discarded as a result of the fact that encrypted streams became much larger than their parent streams which caused the streaming of contents to take much longer.

5.2 Possible Improvements or Additions

The client station developed in this project is not platform independent, for this reason, possible improvements to the project would be creating platform independent client stations using SUN's Java Framework.

REFERENCES

[1] **Software Engineering Concepts:** Microsoft Corporation "Microsoft Application Architecture Guide 2nd Edition" 2009, ISBN: 9780735627109

[2] **Libraries:** Wikipedia "http://en.wikipedia.org/wiki/library"

[3] **Libraries:** Microsoft Students With Encarta 2008 "Library (institution)"

[4] **Digital Libraries:** Wikipedia "http://en.wikipedia.org/wiki/Digital_library"

[5] **Digitization:** Daniel Akst "The Webster Chronicle" 2005

[6] **C# Programming:** MSDN "http://msdn.microsoft.com/en-us/vcsharp/aa33809.aspx"

[7] **.Net Framework:** MSDN [local 2008]

"ms-help://MS.MSDNQTR.v90.en/dv_fxnetstart/html/f61f02f2-2f20-483d-8f56-a9c8f3a54986.htm"

[8] **Windows Presentation Foundation :** MSDN [local 2008]

"ms-help://MS.MSDNQTR.v90.en/wpf_conceptual/html/f667bd15-2134-41e9-b4af-5ced6fafab5d.htm"

APPENDIX A

CODE

A.1 System Components Code

LBRARY COBTROL OBJECT

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.IO;

namespace DigitalLibraryServerX35
{
    class LibraryControlObject
    {
        /// <summary>
        ///
        /// </summary>
        public UsersManagementComponent Users_Manager;

        /// <summary>
        ///
        /// </summary>
        public NetworkServiceComponent NetworkService_Manager;

        /// <summary>
        ///
        /// </summary>
        public ClientServiceObject Client_Service;

        /// <summary>
        ///
        /// </summary>
        public SessionControlComponent Session_Manager;

        /// <summary>
        ///
        /// </summary>
        public ContentDatabaseManagmentComponent Content_Database_Manager;

        public ApplicationConfiguration configuration;


        public LibraryControlObject()
        {
            Prepare_Application_Paths();
            configuration = new ApplicationConfiguration();
            Content_Database_Manager = new ContentDatabaseManagmentComponent();
            Session_Manager = new SessionControlComponent { Max_Session_Time =
configuration.ClientConnectionTimeOut };
            Users_Manager = new UsersManagementComponent ();
            Client_Service = new ClientServiceObject(ref Content_Database_Manager);
            NetworkService_Manager = new NetworkServiceComponent { HTTP_Port = 8900, Use_Http =
true, ClientService = Client_Service };
```

```
        }

        private void Prepare_Application_Paths()
        {
            if (Directory.Exists(ApplicationPaths.Backup_Directory) == false)
                Directory.CreateDirectory(ApplicationPaths.Backup_Directory);

            if (File.Exists(ApplicationPaths.Users_DB_File) == false)
                File.Create(ApplicationPaths.Users_DB_File);

            if (File.Exists(ApplicationPaths.Config_File) == false)
                File.Create(ApplicationPaths.Config_File);
        }

        public string BackupDatabase(string location)
        {
            Content_Database_Manager.Close();
            try
            {
                DateTime stamp = DateTime.Now;
                PackageManagementComponent store = new PackageManagementComponent(location +
"\\DL_Backup[" +
                    stamp.Year + "_" +
                    stamp.Month + "_" +
                    stamp.Day + "_" +
                    stamp.Hour + "_" +
                    stamp.Minute + "_" +
                    stamp.Second + "]" +
                    ".DLBackup", false);

                store.StoreContent(ApplicationPaths.Data_Directory+"\\DBContentInfo.sdf",
"DBContentInfo.sdf", "", true);
                store.StoreContent(ApplicationPaths.Data_Directory + "\\DBContentInfo.store",
"DBContentInfo.store", "zip", true);
                store.Close();

                return store.path;
            }
            finally
            {
                Content_Database_Manager.LoadData();
            }
        }

        public void RestoreDatabase(string location)
        {
            Content_Database_Manager.Close();

            try
            {
                PackageManagementComponent store = new PackageManagementComponent(location,
false);
                store.ExtractContent(ApplicationPaths.Data_Directory+"\\DBContentInfo.sdf",
"DBContentInfo.sdf");
                store.ExtractContent(ApplicationPaths.Data_Directory+"\\DBContentInfo.store",
"DBContentInfo.store");
                store.Close();
            }
            finally
            {
                Content_Database_Manager.LoadData();
                try
```

```
                {
                    File.Delete(ApplicationPaths.Data_Directory+"\\Old_DB.DLBackup");
                }
                catch (Exception) { }
            }
        }

        public void Start_Service_Broadcast()
        { NetworkService_Manager.Broadcast_Service(); }

        public void Halt_Service_Broadcast()
        { NetworkService_Manager.StopBroadcast(); }

        public void Save_Configuration()
        { configuration.Save(); }

        public void Load_Configuration()
        { configuration.Reload(); }

        public void Close_Library()
        {
            Users_Manager.Save_To_Disk();
            Content_Database_Manager.Close();
        }

    }
}
```

CLIENT SERVICE OBJECT

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using DLCommunicator;
using System.ComponentModel;
using System.IO;

namespace DigitalLibraryServerX35
{
    [Serializable]
    public class ClientServiceObject : MarshalByRefObject, IClientService
    {
        #region Variables Section

        Client_Session client_Session;

        ContentDatabaseManagmentComponent ContentDB;
        UsersManagementComponent user_manager;
        SessionControlComponent Session_Manager;

        Stream dataBuffer;

        #endregion

        public ClientServiceObject(ref ContentDatabaseManagmentComponent cdb)
        {
            ContentDB = cdb; // CDB serves as a refrence to the main dataset
        }

        public void Register_ContentRating(Guid uid, int rating)
        {
            //throw new NotImplementedException();
        }

        public void Register_ContentViewed(Guid uid)
        {
            try
            {
                ContentDB.contenInfoTable.FindByUID(uid).View_Count++;
            }
            catch (Exception)
            {
            }
        }

        public List<ContentInfo> RequestAllContentInfo()
        {
            List<ContentInfo> contents = new List<ContentInfo>();
            foreach (var row in ContentDB.contenInfoTable)
            {
                contents.Add(Code.ConvertRow(row));
            }
            return contents;
        }

        public int Begin_RequestContentData(Guid uid)
```

```
        {
            dataBuffer = ContentDB.getContentStream(uid.ToString());
            dataBuffer.Position = 0;
            //bytesRead = 0;
            return (int)dataBuffer.Length;
        }

        public DataPacket Request_ContentData(int offset, int size)
        {
            DataPacket packet = new DataPacket { data = new byte[size], bytesRead = 0 };

            if ((packet.bytesRead = dataBuffer.Read(packet.data, 0, size)) > 0)
            {
                return packet;
            }
            else
                return null;

        }

        public void End_RequestContentData()
        {
            dataBuffer.Dispose();
            //dataBuffer = null;
        }

        public bool IsConnected
        {
            get
            {
                return true;
                //if (client_Session != null &&
Session_Manager.Active_ClientSession.Contains(client_Session))
                //    return true;
                //else
                //    return false;
            }
        }

        public bool Login(string user_name, Guid user_code)
        {
            client_Session = SecurityComponent.ClientLogin(user_name, user_code, ref
user_manager, ref Session_Manager);
            if (client_Session != null)
                return true;
            else
                return false;
        }

        public void Logout()
        {
            client_Sessipn = null;
        }

        #region code

        private ContentInfo convertRow(Guid uid)
        {
            var row = ContentDB.contenInfoTable.FindByUID(uid);
            if (uid != null)
            {
```

```
                return new ContentInfo { UID = row.UID, Category = row.Category, Creator =
row.Creator,        Description        =        row.Description,        LibraryContentType        =
Code.GetTypeOfContent(row.Content_Type), Name = row.Name, SearchTags = row.Tags };
            }
            return null;
        }

        #endregion


    }
}
```

USER MANAGEMENT COMPONENT

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Xml.Serialization;
using System.IO;
using System.ComponentModel;

namespace DigitalLibraryServerX35
{
    /// <summary>
    /// Stores and manage User Registration and Activation
    /// </summary>
    class UsersManagementComponent
    {
        BindingList<User> _Users;
        /// <summary>
        /// Stores the List of Registered Users
        /// </summary>
        public BindingList<User> Users
        {
            get { return _Users; }
            set { _Users = value; }
        }

        public UsersManagementComponent()
        {
            try
            {
                Load_From_Disk();
            }
            catch (Exception)
            {
                Users = new BindingList<User>();
                Users.Add(User.NEW_USER("new user"));
                try
                {
                    Save_To_Disk();
                }
                catch (Exception)
                {
                }
            }
        }

        /// <summary>
        /// Registers a new user.
        /// </summary>
        /// <param name="name"> the users name</param>
        public void Register_User(string name)
        {
            Users.Add(User.NEW_USER(name));
        }

        /// <summary>
        /// Renames a user, bur=t retains the user_code.
        /// </summary>
        /// <param name="user_code">ID of user</param>
```

```
/// <param name="new_name">the new name used to represent the user</param>
public void Edit_Uer(Guid user_code, string new_name)
{
    var user = Find_User(user_code);

    if (user != null)
        user.Name = new_name;
}

/// <summary>
/// Unregisters the user from the database
/// </summary>
/// <param name="user_code">ID of user</param>
public void Unregister_User(Guid user_code)
{
    var user = Find_User(user_code);

    if (user != null)
        Users.Remove(user);
}

/// <summary>
/// Activates a registered user
/// </summary>
/// <param name="user_code">ID of user</param>
public void Activate_User(Guid user_code)
{
    var user = Find_User(user_code);

    if (user != null)
        user.Enabled = true;
}

/// <summary>
/// Deactivates a registered user
/// </summary>
/// <param name="user_code">ID of user</param>
public void Deactivate_User(Guid user_code)
{
    var user = Find_User(user_code);

    if (user != null)
        user.Enabled = false;
}


public void Save_To_Disk()
{
    XmlSerializer xser = new XmlSerializer(Users.GetType());
    TextWriter txtwr = new StreamWriter(ApplicationPaths.Users_DB_File);
    xser.Serialize(txtwr, Users);
}

public void Load_From_Disk()
{
    Users = new BindingList<User>();
    XmlSerializer xser = new XmlSerializer(Users.GetType());
    TextReader txtrd = new StreamReader(ApplicationPaths.Users_DB_File);
    Users = xser.Deserialize(txtrd) as BindingList<User>;
}

/// <summary>
```

```
        /// Finds an registered user
        /// </summary>
        /// <param name="user_code"></param>
        /// <returns></returns>
        public User Find_User(Guid user_code)
        {
            var u = (from user in Users
                     where user.User_Code == user_code
                     select user).ToArray();
            if (u != null)
                return u[0];
            else
                return null;
        }
    }

    public class User
    {
        string _Name;
        /// <summary>
        /// The name used to identify the user
        /// </summary>
        public string Name
        {
            get { return _Name; }
            set { _Name = value; }
        }

        Guid _User_Code;
        /// <summary>
        /// The uid code unique to each user
        /// </summary>
        public Guid User_Code
        {
            get { return _User_Code; }
            set { _User_Code = value; }
        }

        bool _enabled;
        /// <summary>
        /// Indicates whether the user is activated
        /// </summary>
        public bool Enabled
        {
            get { return _enabled; }
            set { _enabled = value; }
        }

        public static User NEW_USER(string name)
        {
            return new User { _Name = name, _User_Code = Guid.NewGuid(), _enabled = false };
        }
    }
}
```

NETWORK SERVICE COMPONENT \

```
using System.Linq;
using System.Runtime.Remoting.Channels.Http;
using System.Runtime.Remoting.Channels.Tcp;
using System.Runtime.Remoting.Channels;
using System.Collections;
using System.Runtime.Remoting;
using System.Timers;
using System.ComponentModel;

namespace DigitalLibraryServerX35
{
    /// <summary>
    /// Network and Service Component
    /// </summary>
    public class NetworkServiceComponent : INotifyPropertyChanged
    {
        #region INotifyPropertyChanged Interface Implementation

        /// <summary>
        /// Handles Property change notifications
        /// </summary>
        public event PropertyChangedEventHandler PropertyChanged;
        protected void OnPropertyChanged(string name)
        {
            if (PropertyChanged != null)
            {
                PropertyChanged(this, new PropertyChangedEventArgs(name));
            }
        }

        #endregion

        /// <summary>
        /// Stores the service name of the Digital library service
        /// </summary>
        public string Service_Name;

        /// <summary>
        /// Stores the TCP port number to use for TCP broadcasting
        /// </summary>
        public int TCP_Port;

        /// <summary>
        /// Stores the HTTP port number to use for TCP broadcasting
        /// </summary>
        public int HTTP_Port;

        /// <summary>
        /// Used to indicate whether the service should also be broadcated over the Http channel.
        /// This should be set to true if the service is to be available over the internet.
        /// </summary>
        public bool Use_Http;

        /// <summary>
        /// Privately holds an Http Channel to be registered during broadcast.
        /// </summary>
        private HttpChannel HTTP_Channel;
```

```
/// <summary>
/// Stores the HTTP channel properties configuration
/// </summary>
private IDictionary http_Channel_Properties;

/// <summary>
/// Privately holds a Tcp Channel to be registered during broadcast.
/// </summary>
private TcpChannel TCP_Channel;

/// <summary>
/// Client service to be broadcasted over the network
/// </summary>
public ClientServiceObject ClientService;

/// <summary>
/// Refresh timing object
/// </summary>
private Timer Refresh_Timer;


bool _ServiceActivated;
/// <summary>
/// Indicates whether the service is currently activated
/// </summary>
public bool ServiceActivated
{
    get { return _ServiceActivated; }
    set
    {
        _ServiceActivated = value;
        OnPropertyChanged("ServiceStatusIndicator");
    }
}

public System.Windows.Media.Brush ServiceStatusIndicator
{
    get
    {
        if (ServiceActivated)
            return System.Windows.Media.Brushes.OliveDrab;
        else
            return System.Windows.Media.Brushes.DarkOrange;
    }
    set
    {
    }
}

/// <summary>
/// Network Configuration and Service Managment Component
/// </summary>
public NetworkServiceComponent()
{
    Service_Name = string.Empty;
    Refresh_Timer = new Timer(52000); //refresh every 52 seconds
    Refresh_Timer.Elapsed += new ElapsedEventHandler(Refresh_Timer_Elapsed);
}

/// <summary>
/// Refreshes broadcast when 52 seconds elapses
```

```
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        void Refresh_Timer_Elapsed(object sender, ElapsedEventArgs e)
        {
            RefreshBroadcast();
        }

        /// <summary>
        /// Starts Broadcasting the service.
        /// This method should be called first before any subsequent Refresh calls.
        /// </summary>
        public void Broadcast_Service()
        {
            if (Service_Name == string.Empty)
                Service_Name = "digitallibraryservice.rem";

            if (Use_Http)
            {
                if (HTTP_Port < 10)
                    HTTP_Port = 8900;   // Default Http port Number
                http_Channel_Properties = new Hashtable();
                http_Channel_Properties["name"] = "httpchnl";
                http_Channel_Properties["port"] = HTTP_Port;
                HTTP_Channel = new HttpChannel(http_Channel_Properties,
                   new BinaryClientFormatterSinkProvider(),
                   new BinaryServerFormatterSinkProvider());              //Make use of binary
serialization for the http channel
                ChannelServices.RegisterChannel(HTTP_Channel, false); //Registers the application
with the HTTP channel
                RemotingServices.Marshal(ClientService,  Service_Name);  //Marshals the Client
Service through the channel
            }
            else
            {
                if (TCP_Port < 10)
                    TCP_Port = 5200;   // Default Tcp port Number
                TCP_Channel = new TcpChannel(TCP_Port);
                ChannelServices.RegisterChannel(TCP_Channel, true); //Registers the application
with the TCP channel
                RemotingServices.Marshal(ClientService,  Service_Name);  //Marshals the Client
Service through the channel
            }


            ServiceActivated = true;

            Refresh_Timer.Start();
        }

        /// <summary>
        /// Stops Broadcasting the service.
        /// </summary>
        public void StopBroadcast()
        {
            Refresh_Timer.Stop();

            if (ChannelServices.RegisteredChannels.Contains(HTTP_Channel))
                ChannelServices.UnregisterChannel(HTTP_Channel); //Unregisters the Http Channel

            if (ChannelServices.RegisteredChannels.Contains(TCP_Channel))
                ChannelServices.UnregisterChannel(TCP_Channel); //Unregisters the Tcp Channel
```

```
            RemotingServices.Disconnect(ClientService);
            ServiceActivated = false;
        }

        /// <summary>
        /// Re-Broadcast the service after service broadcast time-out
        /// </summary>
        public void RefreshBroadcast()
        {
            RemotingServices.Marshal(ClientService,  Service_Name);  //Re-Marshals  the  Client
Service through the channels
            ServiceActivated = true;
        }


    }
}
```

SESSION CONTROL COMPONENT

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.ComponentModel;

namespace DigitalLibraryServerX35
{
    /// <summary>
    /// Session Control and Managment Component
    /// </summary>
    public class SessionControlComponent
    {
        BindingList<Client_Session> _Active_ClientSession;

        /// <summary>
        /// Hold the list of active sessions to be bound to the UI
        /// </summary>
        public BindingList<Client_Session> Active_ClientSession
        {
            get { return _Active_ClientSession; }
            set { _Active_ClientSession = value; }
        }

        /// <summary>
        /// Maximum Session time in min
        /// </summary>
        public int Max_Session_Time;

        /// <summary>
        /// Logon period start time
        /// </summary>
        private DateTime _Start_Time;

        /// <summary>
        /// Logon period end time
        /// </summary>
        private DateTime _End_Time;

        /// <summary>
        ///
        /// </summary>
        /// <param name="user"></param>
        public Client_Session Create_Session(User user)
        {
            if (DateTime.Now <= _End_Time && DateTime.Now >= _Start_Time)
            {
                var new_session = Client_Session.New_Client_Session(user);
                Active_ClientSession.Add(new_session);
                return new_session;
            }
            else
                return null;
        }

        /// <summary>
        ///
        /// </summary>
```

```
        /// <param name="session"></param>
        public void Kill_Session(Client_Session session)
        {
            Active_ClientSession.Remove(session);
        }

        /// <summary>
        ///
        /// </summary>
        /// <param name="starts"></param>
        /// <param name="ends"></param>
        public void Set_Login_TimeFrame(DateTime starts, DateTime ends)
        {
            _Start_Time = starts;
            _End_Time = ends;
        }

        /// <summary>
        ///
        /// </summary>
        private void Refresh()
        {
            if (!(DateTime.Now <= _End_Time && DateTime.Now >= _Start_Time))
            {
                Active_ClientSession.Clear();
            }

            else
            {
                var Expired_Sessions = from session in Active_ClientSession
                                       where (DateTime.Now - session._LoginTime).TotalMinutes >=
Max_Session_Time
                                       select session;

                foreach (var session in Expired_Sessions)
                {
                    Active_ClientSession.Remove(session);
                }
            }
        }
    }

    /// <summary>
    ///
    /// </summary>
    public class Client_Session
    {
        Guid _Session_ID;
        /// <summary>
        ///
        /// </summary>
        public Guid Session_ID
        {
            get { return _Session_ID; }
            set { _Session_ID = value; }
        }

        User _Session_User;
        /// <summary>
        ///
        /// </summary>
        public User Session_User
```

```
        {
          get { return _Session_User; }
          set { _Session_User = value; }
        }

        /// <summary>
        ///
        /// </summary>
        public DateTime _LoginTime;

        /// <summary>
        ///
        /// </summary>
        public string Session_Duration
        {
            get
            {
                var duration = (DateTime.Now - _LoginTime);
                string span = "";
                span += ((duration.Hours < 10) ? "0"+duration.Hours: duration.Hours.ToString());
                span += ":"+ ((duration.Minutes < 10) ? "0"+duration.Minutes:
duration.Minutes.ToString());
                span += ":"+((duration.Seconds < 10) ? "0"+duration.Seconds:
duration.Seconds.ToString());
                return span;
            }
            set { }
        }

        public static Client_Session New_Client_Session(User user)
        {
            return new Client_Session { Session_ID = Guid.NewGuid(), _Session_User = user,
_LoginTime = DateTime.Now };
        }

    }
}
```

Security Component

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.IO;

namespace DigitalLibraryServerX35
{
    static class SecurityComponent
    {
        public static Client_Session ClientLogin(string username, Guid usercode, ref UsersManagementComponent users_manager, ref
SessionControlComponent session_manager)
        {
            User user;
            Client_Session new_session = null;
            Client_Session old_session = null;
            if ((user = users_manager.Find_User(usercode)) != null)
            {
                if (user.Name == username)
                {
                    var sessions = (from session in session_manager.Active_ClientSession where session.Session_User == user
select session).ToArray();
                    if(sessions != null)
                        old_session = sessions[0];

                    if (old_session != null)
                    { session_manager.Kill_Session(old_session); }
                    return session_manager.Create_Session(user);
                }
            }

            return null;
        }

        public static void EncryptStream(Guid session_ID, ref Stream stream)
        {
            throw new NotImplementedException("Stream encryption not yet implemented!");
        }
    }

}
```

[C#] MainWindow Code

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Navigation;
using System.Windows.Shapes;
using System.Runtime.Remoting;
using System.Runtime.Remoting.Channels;
using System.Runtime.Remoting.Channels.Tcp;
using System.Runtime.Remoting.Channels.Http;
using DigitalLibraryServerX35.DSContentInfoTableAdapters;
using System.IO;
using System.Timers;
using Microsoft.Windows.Controls;
using System.Windows.Controls.Primitives;
using System.Xml.Serialization;
using System.Collections;
using System.IO.Packaging;
using System.Windows.Media.Animation;
using System.Threading;
using System.ComponentModel;

namespace DigitalLibraryServerX35
{
    /// <summary>
    /// Interaction logic for MainWindow.xaml
    /// </summary>
    public partial class MainWindow : Window
    {
        LibraryControlObject DigitalLibrary;

        public MainWindow()
        {
            Cursor = Cursors.Wait;

            InitializeComponent();

            DigitalLibrary = new LibraryControlObject();

            InitializeNetworkService();

            BindContolsToData();

            InitializeEffects();

            Cursor = Cursors.Arrow;

        }

        private void Window_Loaded(object sender, RoutedEventArgs e)
        {
            tblContent.ItemsSource = DigitalLibrary.Content_Database_Manager.contenInfoTable;
            tblContent.CanUserDeleteRows = false;
        }

        private void Window_Closing(object sender, System.ComponentModel.CancelEventArgs e)
        {
            DigitalLibrary.Close_Library();
        }

        private void InitializeEffects()
        {
            InitStryBdFadeViewerOff(TabCntrlMain.Name);
            InitStryBdFadeViewerOn(TabCntrlMain.Name);
        }

        private void BindContolsToData()
        {
```

```
            tblContent.ItemsSource = DigitalLibrary.Content_Database_Manager.contenInfoTable;
            tblUsers.ItemsSource = DigitalLibrary.Users_Manager.Users;
            tblUserSessions.ItemsSource = DigitalLibrary.Session_Manager.Active_ClientSession;
            this.DataContext = DigitalLibrary.configuration;
        }

        void InitializeNetworkService()
        {
            connectionIndicator.SetBinding(Rectangle.FillProperty, new Binding { Source = DigitalLibrary.NetworkService_Manager,
 Path = new PropertyPath("ServiceStatusIndicator") });
            DigitalLibrary.Start_Service_Broadcast();
        }

        private void ServiceCntrlBtn_Click(object sender, RoutedEventArgs e)
        {
            if (ServiceCntrlBtn.IsChecked.Value)
            {
                DigitalLibrary.Halt_Service_Broadcast();
                ServiceCntrlBtn.Content = "4";
                ServiceCntrlBtn.ToolTip = "Restart";
            }

            else
            {
                DigitalLibrary.Start_Service_Broadcast();
                ServiceCntrlBtn.Content = ";";
                ServiceCntrlBtn.ToolTip = "Halt";
            }
        }

        private void BtnTabPage_Click(object sender, RoutedEventArgs e)
        {
            if (clickedBtn == BtnPageHome || clickedBtn == BtnPageContent || clickedBtn == BtnPageCPSettings || clickedBtn ==
BtnPageCPDatabase || clickedBtn == BtnPageCPUsers)
            {
                InitStryBdFadeViewerOff(TabCntrlMain.Name);
                InitStryBdFadeViewerOn(TabCntrlMain.Name);
            }

            stryBdFadeTabMainOff.Begin(this);
            clickedBtn = (e.Source as Button);
        }

        Button clickedBtn;
        private void ChangePage()
        {

            if (clickedBtn == BtnPageHome)
                pageHome.IsSelected = true;
            else if (clickedBtn == BtnPageContent)
                pageContent.IsSelected = true;
            else if (clickedBtn == BtnPageCPSettings)
                pageCPSettings.IsSelected = true;
            else if (clickedBtn == BtnPageCPDatabase)
                pageCPDatabase.IsSelected = true;
            else if (clickedBtn == BtnPageCPUsers)
                pageCPUsers.IsSelected = true;
        }

        private void btnContentAdd_Click(object sender, RoutedEventArgs e)
        {
            WinAddContent addWizard = new WinAddContent(ref DigitalLibrary.Content_Database_Manager.GroupInfoTable);
            addWizard.ShowDialog();
            if (addWizard.Done)
            {
                var groups = from row in DigitalLibrary.Content_Database_Manager.GroupInfoTable
                             where row.Group == addWizard.contentRow.Group
                             select row;
                if (groups.Count() < 1)
                    DigitalLibrary.Content_Database_Manager.GroupInfoTable.AddTblGroupInfoRow(addWizard.contentRow.Group);

                DigitalLibrary.Content_Database_Manager.AddContent(addWizard.contentRow, addWizard.path);
            }
        }
        private void btnContentEdit_Click(object sender, RoutedEventArgs e)
        {
            if (tblContent.SelectedIndex > -1)
            {
                WinAddContent addWizard = new
WinAddContent(DigitalLibrary.Content_Database_Manager.contenInfoTable[tblContent.SelectedIndex]);
                addWizard.ShowDialog();
                if (addWizard.Done)
```

```
        {
            var groups = from row in DigitalLibrary.Content_Database_Manager.GroupInfoTable
                         where row.Group == addWizard.contentRow.Group
                         select row;
            if (groups.Count() < 1)
                //CDB.GroupInfoTable.AddTblGroupInfoRow(addWizard.contentRow.Group);
                DigitalLibrary.Content_Database_Manager.GroupInfoTable.AddTblGroupInfoRow(addWizard.contentRow.Group);

            //CDB.UpdateContent(addWizard.contentRow, addWizard.path);
            DigitalLibrary.Content_Database_Manager.UpdateContent(addWizard.contentRow, addWizard.path);
        }
    }
}

ApplicationConfiguration Config;
private void BtnSaveSettings_Click(object sender, RoutedEventArgs e)
{
    DigitalLibrary.Save_Configuration();
}

private void BtnRevertSettings_Click(object sender, RoutedEventArgs e)
{
    DigitalLibrary.Load_Configuration();
}

private void LoadConfig()
{
    pageCPSettings.DataContext = Config;
}

private string BackupDatabase(string location)
{
    return DigitalLibrary.BackupDatabase(location);
}

private void RestoreDatabase(string location)
{
    DigitalLibrary.RestoreDatabase(location);
    tblContent.ItemsSource = DigitalLibrary.Content_Database_Manager.contenInfoTable;
}

private void txtBxContentSearch_TextChanged(object sender, TextChangedEventArgs e)
{
    try
    {
        if (txtBxContentSearch.Text.Length > 0 && txtBxContentSearch.Text != "search")
        {
            tblSearch.Visibility = Visibility.Visible;
            string search = txtBxContentSearch.Text;
            tblSearch.ItemsSource = from content in DigitalLibrary.Content_Database_Manager.contenInfoTable
                                    where content.Name.ToLower().Contains(search)
                                       || content.Tags.ToLower().Contains(search)
                                       || content.Description.ToLower().Contains(search)
                                       || content.Category.ToLower().Contains(search)
                                       || content.Group.ToLower().Contains(search)
                                    select content;
        }
        else
        {
            tblSearch.Visibility = Visibility.Collapsed;
            tblSearch.ItemsSource = null;
        }
    }
    catch (Exception) { }
}

private void txtBxContentSearch_GotKeyboardFocus(object sender, KeyboardFocusChangedEventArgs e)
{
    if (txtBxContentSearch.Text == "search")
        txtBxContentSearch.Text = "";
}

private void txtBxContentSearch_LostKeyboardFocus(object sender, KeyboardFocusChangedEventArgs e)
{
    if (txtBxContentSearch.Text == "")
    {
        txtBxContentSearch.Text = "search";
    }
}

private void BtnBackupDB_Click(object sender, RoutedEventArgs e)
{
    try
```

```
            {
                BackupDatabase(ApplicationPaths.Backup_Directory);
                tblAvailableBackups.ItemsSource = Config.AvailableBackups;

            }
            catch (Exception)
            {

            }
        }

        private void BtnRestoreDB_Click(object sender, RoutedEventArgs e)
        {
            if(tblAvailableBackups.SelectedIndex > -1)
                try
                {
                    RestoreDatabase(Config.AvailableBackups[tblAvailableBackups.SelectedIndex]);
                }
                catch (Exception) { }
        }

        #region Effects

        Storyboard stryBdFadeTabMainOff;
        Storyboard stryBdFadeTabMainOn;

        private void InitStryBdFadeViewerOff(string ControlName)
        {
            DoubleAnimation animFadeOff = new DoubleAnimation(0.0, new Duration(new TimeSpan(0, 0, 0, 0, 500)));
            stryBdFadeTabMainOff = new Storyboard();
            stryBdFadeTabMainOff.Children.Add(animFadeOff);
            Storyboard.SetTargetName(animFadeOff, ControlName);
            Storyboard.SetTargetProperty(animFadeOff, new PropertyPath(Grid.OpacityProperty));
            stryBdFadeTabMainOff.Completed += new EventHandler(stryBdFadeTabMainOff_Completed);
        }

        void stryBdFadeTabMainOff_Completed(object sender, EventArgs e)
        {
            ChangePage();
            stryBdFadeTabMainOn.Begin(this);
        }

        private void InitStryBdFadeViewerOn(string ControlName)
        {
            DoubleAnimation animFadeOn = new DoubleAnimation(1.0, new Duration(new TimeSpan(0, 0, 0, 0, 800)));
            stryBdFadeTabMainOn = new Storyboard();
            stryBdFadeTabMainOn.Children.Add(animFadeOn);
            Storyboard.SetTargetName(animFadeOn, ControlName);
            Storyboard.SetTargetProperty(animFadeOn, new PropertyPath(Grid.OpacityProperty));
        }
        #endregion



        private void btnUserAdd_Click(object sender, RoutedEventArgs e)
        {
            DigitalLibrary.Users_Manager.Register_User("new user");
            tblUsers.SelectedIndex = tblUsers.Items.Count - 1;
        }

        private void btnUserDelete_Click(object sender, RoutedEventArgs e)
        {
            if (tblUsers.SelectedIndex > -1)
            {
                if(MessageBox.Show("Are you sure you want to delete this user?", "Delete Confirmation", MessageBoxButton.YesNo)
== MessageBoxResult.Yes)

DigitalLibrary.Users_Manager.Unregister_User(DigitalLibrary.Users_Manager.Users[tblUsers.SelectedIndex].User_Code);
            }
        }

        private void btnUsertEdit_Click(object sender, RoutedEventArgs e)
        {

        }


    }


}
```

[XML] Application Configuration

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
    <configSections>
        <sectionGroup name="userSettings" type="System.Configuration.UserSettingsGroup, System, Version=2.0.0.0, Culture=neutral,
PublicKeyToken=b77a5c561934e089" >
            <section name="DigitalLibraryServerX35.Properties.Settings" type="System.Configuration.ClientSettingsSection, System,
Version=2.0.0.0,         Culture=neutral,         PublicKeyToken=b77a5c561934e089"         allowExeDefinition="MachineToLocalUser"
requirePermission="false" />
        </sectionGroup>
    </configSections>
    <connectionStrings>
        <add name="DigitalLibraryServerX35.Properties.Settings.ContentInfoDBConnectionString"
            connectionString="Data Source=|DataDirectory|\ContentInfoDB.sdf"
            providerName="Microsoft.SqlServerCe.Client.3.5" />
        <add name="DigitalLibraryServerX35.Properties.Settings.DBContentInfoConnectionString"
            connectionString="Data Source=|DataDirectory|\DBContentInfo.sdf"
            providerName="Microsoft.SqlServerCe.Client.3.5" />
    </connectionStrings>
    <userSettings>
        <DigitalLibraryServerX35.Properties.Settings>
            <setting name="ConfigPath" serializeAs="String">
                <value>|DataDirectory|\config.ucf</value>
            </setting>
            <setting name="Username" serializeAs="String">
                <value>server</value>
            </setting>
            <setting name="Password" serializeAs="String">
                <value />
            </setting>
            <setting name="ClientServiceName" serializeAs="String">
                <value>nig.futmx.seet.ece.dls</value>
            </setting>
            <setting name="ClientServicePort" serializeAs="String">
                <value>0</value>
            </setting>
            <setting name="Client_Session_Maximum_Clients" serializeAs="String">
                <value>0</value>
            </setting>
            <setting name="Client_Session_Start_Time" serializeAs="String">
                <value>10/29/2010 08:00:00</value>
            </setting>
            <setting name="Client_Session_End_Time" serializeAs="String">
                <value>10/29/2010 18:00:00</value>
            </setting>
            <setting name="Client_Session_Max_Session_Time" serializeAs="String">
                <value>0</value>
            </setting>
            <setting name="Database_Committ_Interval" serializeAs="String">
                <value>0</value>
            </setting>
            <setting name="Client_Session_UseHttp" serializeAs="String">
                <value>True</value>
            </setting>
            <setting name="Client_Session_Max_Sessions" serializeAs="String">
                <value>50</value>
            </setting>
        </DigitalLibraryServerX35.Properties.Settings>
    </userSettings>
</configuration>
```

[C#] MainWindow Code

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Navigation;
using System.Windows.Shapes;
using System.Runtime.Remoting;
using System.Runtime.Remoting.Channels;
using System.Runtime.Remoting.Channels.Tcp;
using DLCommunicator;
using System.IO;
using System.Threading;
using System.Windows.Threading;
using System.Windows.Media.Animation;
using System.Deployment.Application;
using System.Runtime.Serialization;
using System.Runtime.Serialization.Formatters.Binary;
using DigitalLibraryx35;
using System.Runtime.Remoting.Channels.Http;
using System.Collections;

namespace DigitalLibrary
{
    /// <summary>
    /// Interaction logic for MainWindow.xaml
    /// </summary>
    public partial class MainWindow : Window
    {
        AppConfig config;
        public MainWindow()
        {
            Cursor = Cursors.AppStarting;
            InitializeComponent();
            config = new AppConfig();
            pageControlPanel.DataContext = config;
            InitializeFader();
            InitializeViewers();
            InitializeFileOperations();

            InitializeNetwork();
            InitializeHomeContents();
            Cursor = Cursors.Arrow;

        }

        private void InitializeFileOperations()
        {
            LoadedDir = new DirectoryInfo(Code.GetLoadedFile(string.Empty));

            LoadedDir.Create();
            LoadedDir.Attributes = FileAttributes.Encrypted;
        }


        #region EventHandlers: Window
        private void Window_Loaded(object sender, RoutedEventArgs e)
        {
            //RefreshContentList();
        }
        private void Window_Closing(object sender, System.ComponentModel.CancelEventArgs e)
        {
        }
        #endregion


        #region Fading Effect
        private Grid FaderGrid;
        private void InitializeFader()
        {
            FaderGrid = new Grid { HorizontalAlignment = HorizontalAlignment.Stretch, VerticalAlignment =
VerticalAlignment.Stretch, Background = Brushes.White, Opacity = 0.9};
```

```
            InitContentListFadeOff();
            InitContentListOn();
        }
        private void Fade()
        {
            try
            {
                grdBound.Children.Add(FaderGrid); //this.Topmost = false;
            }
            catch (Exception){}
        }
        private void UnFade()
        {
            try
            {
                grdBound.Children.Remove(FaderGrid); //this.Topmost = true;
            }
            catch (Exception) { }
        }
        #endregion


        #region Viewers
        BookViewerXPS BookViewer;
        ViewerPlayer MediaViewer;
        private void InitializeViewers()
        {
            BookViewer = new BookViewerXPS { HorizontalAlignment = HorizontalAlignment.Stretch, VerticalAlignment =
VerticalAlignment.Stretch };
            MediaViewer = new ViewerPlayer { HorizontalAlignment = HorizontalAlignment.Stretch, VerticalAlignment =
VerticalAlignment.Stretch };

        }
        #endregion


        #region Networking
        IClientService service;
        private System.Timers.Timer timerCheckServer;
        private void InitializeNetwork()
        {
            TcpChannel channel = new TcpChannel();
            IDictionary props = new Hashtable();
            props["name"] = "httpchnl";
            HttpChannel httpChnnl = new HttpChannel(props,
               new BinaryClientFormatterSinkProvider(),
               new BinaryServerFormatterSinkProvider());
            ChannelServices.RegisterChannel(channel, false);
            ChannelServices.RegisterChannel(httpChnnl, false);
            UpdateServiceIndicator(ConnectToService());
            RefreshContentList();
            timerCheckServer = new System.Timers.Timer(1000);
            timerCheckServer.AutoReset = true;
            timerCheckServer.Elapsed += new System.Timers.ElapsedEventHandler(timerCheckServer_Elapsed);
            timerCheckServer.Start();
        }

        private bool ConnectToService()
        {
            try
            {
                service = (IClientService)Activator.GetObject(typeof(IClientService), config.Service_Address);
                try
                {
                    var test = service.IsConnected;
                    return true;
                }
                catch (Exception)
                {
                    return false;
                }
            }
            catch (Exception)
            {

            }
            return false;
        }
        private void RefreshContentList()
        {
            bool connected = false;
            try
```

```
            {
                connected = service.IsConnected;
            }
            catch (Exception ex){}

            if (connected)
            {
                this.Dispatcher.Invoke(System.Windows.Threading.DispatcherPriority.Normal, new Action(delegate()
                {
                    Fade();
                    lBxContents.ItemsSource = ActiveContentList = ContentList = service.RequestAllContentInfo();
                    lBxHomeContentPreview.ItemsSource = ActiveContentList;
                    SetPage(0);
                    BuildCategoryList();
                    UnFade();
                }));
            }


        }

        private void timerCheckServer_Elapsed(object sender, System.Timers.ElapsedEventArgs e)
        {
            bool connected = false;
            try
            {
                connected = service.IsConnected;
            }
            catch (Exception)
            {
                connected = ConnectToService();
                RefreshContentList();
            }


            if (connected)
            {
                UpdateServiceIndicator(true);
            }

            else
            {
                this.Dispatcher.Invoke(System.Windows.Threading.DispatcherPriority.Normal, new Action(delegate()
                {
                    UpdateServiceIndicator(false);
                    ActiveContentList = ContentList = null;
                    lBxContents.ItemsSource = null;
                    CategoryLBx.ItemsSource = null;
                    HomeContents = null;
                    lBxHomeContentPreview.ItemsSource = null;
                    SetPage(0);
                    try{MediaViewer.player.Stop();}
                    catch (Exception){}

                    try{BookViewer.CloseFile();}
                    catch (Exception){}

                    Viewer.Content = null;
                    grdLibrary.SetBinding(Grid.WidthProperty, new Binding { Path = new PropertyPath("ActualWidth"), ElementName =
"grdViewer" });

                } ));
            }
        }


        private void UpdateServiceIndicator(bool connected)
        {
            if (connected)
            {
                connectionIndicator.Dispatcher.Invoke(System.Windows.Threading.DispatcherPriority.Send,  new  Action(delegate()  {
connectionIndicator.Fill = Brushes.YellowGreen; }));
                lblNetworkStatus.Dispatcher.Invoke(System.Windows.Threading.DispatcherPriority.Normal,  new  Action(delegate()  {
lblNetworkStatus.Content = "Online"; }));
            }

            else
            {
                connectionIndicator.Dispatcher.Invoke(System.Windows.Threading.DispatcherPriority.Send,  new  Action(delegate()  {
connectionIndicator.Fill = Brushes.Red; }));
                lblNetworkStatus.Dispatcher.Invoke(System.Windows.Threading.DispatcherPriority.Normal,  new  Action(delegate()  {
lblNetworkStatus.Content = "Offline"; }));
            }
```

```
}
#endregion

private void BtnTabPage_Click(object sender, RoutedEventArgs e)
{
    Button clickedBtn = (Button)e.Source;

    grdContent.Width = 0;

    if (clickedBtn == BtnPageHome)
        pageHome.IsSelected = true;
    else if (clickedBtn == BtnPageContent)
    {
        pageContent.IsSelected = true;
        grdLibrary.Visibility = System.Windows.Visibility.Visible;
        grdContent.Visibility = System.Windows.Visibility.Visible;
    }
    else if (clickedBtn == BtnTabControlPanel)
        pageControlPanel.IsSelected = true;
}

#region Content Activity
List<ContentInfo> ContentList;  //holds the list from the service provider
List<ContentInfo> ActiveContentList; //holds group of content to be displayed

WinDownloader downloader;
DirectoryInfo LoadedDir;

private void BtnRetrieveContent_Click(object sender, RoutedEventArgs e)
{
    RefreshContentList();
}
private void lBxContents_KeyUp(object sender, KeyEventArgs e)
{
    if (e.Key == Key.Enter)
    {
        ContentSelected();
    }
}
private void lBxContents_MouseDoubleClick(object sender, MouseButtonEventArgs e)
{
    currentContent = lBxContents.SelectedItem as ContentInfo;
    ContentSelected();
}
private void ContentSelected()
{
    InitStryBdFadeViewerOff();
    NextGrid = grdContent;
    PrevGrid = grdLibrary;
    stryBdFadeViewerOff.Begin(this);
    if (lBxContents.Items.Count > 0)
    {
        lBxContents.SelectedIndex = 0;
        OpenContent();
    }
}

private ContentInfo currentContent;
private void OpenContent()
{
    CloseContent();
    Fade();
    string path = Code.GetLoadedFile(currentContent.UID.ToString() + "." + currentContent.Extension);

    if (!(File.Exists(path)))
    {
        downloader = new WinDownloader(ref service, currentContent);
        downloader.ShowDialog();
    }

        switch (currentContent.LibraryContentType)
        {
            case TypeOfContent.Audio:
                Viewer.Content = MediaViewer;
                MediaViewer.loadMedia(path, currentContent.Name);
                break;
            case TypeOfContent.Book:
                Viewer.Content = BookViewer;
                BookViewer.LoadFile(path, currentContent.Name);
                break;
            case TypeOfContent.Content:
                break;
            case TypeOfContent.Picture:
```

```
                Viewer.Content = MediaViewer;
                MediaViewer.loadMedia(path, currentContent.Name);
                break;
            case TypeOfContent.Video:
                Viewer.Content = MediaViewer;
                MediaViewer.loadMedia(path, currentContent.Name);
                break;
            default:
                break;
        }
        service.Register_ContentViewed(currentContent.UID);

        UnFade();
    }

    private void CloseContent()
    {
        BookViewer.CloseFile();
        MediaViewer.unloadMedia();
        Viewer.Content = null;
        //InitializeViewers();
    }

    private void BtnCloseContent_Click(object sender, RoutedEventArgs e)
    {
        CloseContent();

        InitStryBdFadeViewerOff();
        NextGrid = grdLibrary;
        PrevGrid = grdContent;
        stryBdFadeViewerOff.Begin(this);

        string[] loadedFiles = Directory.GetFiles(Code.GetLoadedFile(string.Empty));
        for (int i = 0; i < loadedFiles.Length; i++)
        {
            try
            {
                File.Delete(loadedFiles[i]);
            }
            catch (Exception){}
        }
    }

    private void BtnOpenContent_Click(object sender, RoutedEventArgs e)
    {
        InitStryBdFadeViewerOff();
        NextGrid = grdContent;
        PrevGrid = grdLibrary;
        stryBdFadeViewerOff.Begin(this);

    }
    private void BtnSwitchToLibrary_Click(object sender, RoutedEventArgs e)
    {
        InitStryBdFadeViewerOff();
        NextGrid = grdLibrary;
        PrevGrid = grdContent;
        stryBdFadeViewerOff.Begin(this);
    }
    private void txtBxContentSearch_TextChanged(object sender, TextChangedEventArgs e)
    {
        string search = txtBxContentSearch.Text;
        lBxContents.ItemsSource = (from item in ContentList
                                   where item.Name.ToLower().Contains(search)
                                        || item.SearchTags.ToLower().Contains(search)
                                        || item.Creator.ToLower().Contains(search)
                                        || item.Description.ToLower().Contains(search)
                                        || item.Category.ToLower().Contains(search)
                                   select item).ToList();

    }

    #endregion

    #region Animations

    Grid NextGrid;
    Grid PrevGrid;
    Storyboard stryBdFadeViewerOff;
    Storyboard stryBdFadeViewerOn;

    private void InitStryBdFadeViewerOff()
    {
        DoubleAnimation animFadeOff = new DoubleAnimation(0.0, new Duration(new TimeSpan(0, 0, 0, 0, 500)));
```

```
            stryBdFadeViewerOff = new Storyboard();
            stryBdFadeViewerOff.Children.Add(animFadeOff);
            Storyboard.SetTargetName(animFadeOff, grdViewer.Name);
            Storyboard.SetTargetProperty(animFadeOff, new PropertyPath(Grid.OpacityProperty));
            stryBdFadeViewerOff.Completed += new EventHandler(stryBdFadeViewerOff_Completed);
        }

        void stryBdFadeViewerOff_Completed(object sender, EventArgs e)
        {

            NextGrid.SetBinding(Grid.WidthProperty, new Binding { Path = new PropertyPath("ActualWidth"), ElementName =
"grdViewer" });
            PrevGrid.Width = 0;
            InitStryBdFadeViewerOn();
            stryBdFadeViewerOn.Begin(this);
        }

        private void InitStryBdFadeViewerOn()
        {
            DoubleAnimation animFadeOn = new DoubleAnimation(1.0, new Duration(new TimeSpan(0, 0, 0, 0, 500)));
            stryBdFadeViewerOn = new Storyboard();
            stryBdFadeViewerOn.Children.Add(animFadeOn);
            Storyboard.SetTargetName(animFadeOn, grdViewer.Name);
            Storyboard.SetTargetProperty(animFadeOn, new PropertyPath(Grid.OpacityProperty));
            stryBdFadeViewerOn.Completed += new EventHandler(stryBdFadeViewerOn_Completed);
        }

        void stryBdFadeViewerOn_Completed(object sender, EventArgs e)
        {
            //throw new NotImplementedException();
        }


        #region Effects

        Storyboard stryBdFadeContentListOff;
        Storyboard stryBdFadeControlOn;

        private void InitContentListFadeOff()
        {
            DoubleAnimation animFadeOff = new DoubleAnimation(0.0, new Duration(new TimeSpan(0, 0, 0, 0, 400)));
            stryBdFadeContentListOff = new Storyboard();
            stryBdFadeContentListOff.Children.Add(animFadeOff);
            Storyboard.SetTargetName(animFadeOff, lBxContents.Name);
            Storyboard.SetTargetProperty(animFadeOff, new PropertyPath(Grid.OpacityProperty));
            stryBdFadeContentListOff.Completed += new EventHandler(stryBdFadeContentListOff_Completed);
        }

        void stryBdFadeContentListOff_Completed(object sender, EventArgs e)
        {
            SetPage(CurrentPage);
            stryBdFadeControlOn.Begin(this);
        }

        private void InitContentListOn()
        {
            DoubleAnimation animFadeOn = new DoubleAnimation(1.0, new Duration(new TimeSpan(0, 0, 0, 0, 600)));
            stryBdFadeControlOn = new Storyboard();
            stryBdFadeControlOn.Children.Add(animFadeOn);
            Storyboard.SetTargetName(animFadeOn, lBxContents.Name);
            Storyboard.SetTargetProperty(animFadeOn, new PropertyPath(Grid.OpacityProperty));
        }
        #endregion

        #endregion


        #region ControlPanel

        private void BtnRevertSettings_Click(object sender, RoutedEventArgs e)
        {
            config.Load();
            rdBtnHomeIsStartScreen.SetBinding(RadioButton.IsCheckedProperty, "HomeIsStartScreen");
            rdBtnLibraryIsStartScreen.SetBinding(RadioButton.IsCheckedProperty, "LibraryIsStartScreen");
        }
        private void BtnSaveSettings_Click(object sender, RoutedEventArgs e)
        {
            config.Save();
            try
            {
                var test = service.IsConnected;
            }
```

```
            catch (Exception ex)
            {
                ConnectToService();
            }
        }
        #endregion

        #region Home
        System.Timers.Timer HomePreviewContentTimer;

        List<ContentInfo> HomeContents;
        private void InitializeHomeContents()
        {
            try
            {
                HomeContents = service.RequestAllContentInfo();
                grdHomePreview.DataContext = HomeContents;
                var sorted = (from content in HomeContents
                              orderby content.Rating descending
                              select content);
                var toprated = sorted.ToList();
                try
                {
                    toprated.RemoveRange(10, toprated.Count() - 10);
                    //toprated.RemoveRange(10, toprated.Count-10);
                }
                catch (Exception) { }

                lbxHomeTopRated.ItemsSource = toprated;
            }
            catch (Exception) { }

            HomePreviewContentTimer = new System.Timers.Timer(4000);
            HomePreviewContentTimer.Elapsed += new System.Timers.ElapsedEventHandler(HomePreviewContentTimer_Elapsed);
            HomePreviewContentTimer.Start();
        }

        void HomePreviewContentTimer_Elapsed(object sender, System.Timers.ElapsedEventArgs e)
        {
            BtnNextHomeContent_Click(BtnNextHomeContent, new RoutedEventArgs());
        }

        private void BtnNextHomeContent_Click(object sender, RoutedEventArgs e)
        {
            try
            {
                if (lBxHomeContentPreview.SelectedIndex < (lBxHomeContentPreview.Items.Count - 1))
                    lBxHomeContentPreview.SelectedIndex++;
                else
                    lBxHomeContentPreview.SelectedIndex = 0;
            }
            catch (Exception)
            {

                ReflectedVisual.Dispatcher.Invoke(System.Windows.Threading.DispatcherPriority.Send,    new    Action(delegate()    {
ReflectedVisual.Opacity = 0.8; }));
                Thread.Sleep(100);
                ReflectedVisual.Dispatcher.Invoke(System.Windows.Threading.DispatcherPriority.Send,    new    Action(delegate()    {
ReflectedVisual.Opacity = 0.6; }));
                Thread.Sleep(100);
                ReflectedVisual.Dispatcher.Invoke(System.Windows.Threading.DispatcherPriority.Send,    new    Action(delegate()    {
ReflectedVisual.Opacity = 0.5; }));
                Thread.Sleep(200);
                lBxHomeContentPreview.Dispatcher.Invoke(System.Windows.Threading.DispatcherPriority.Send, new Action(delegate()
                    {
                        if (lBxHomeContentPreview.SelectedIndex < (lBxHomeContentPreview.Items.Count - 1))
                            lBxHomeContentPreview.SelectedIndex++;
                        else
                            lBxHomeContentPreview.SelectedIndex = 0;
                    }));
                Thread.Sleep(100);
                ReflectedVisual.Dispatcher.Invoke(System.Windows.Threading.DispatcherPriority.Send,    new    Action(delegate()    {
ReflectedVisual.Opacity = 0.6; }));
                Thread.Sleep(100);
                ReflectedVisual.Dispatcher.Invoke(System.Windows.Threading.DispatcherPriority.Send,    new    Action(delegate()    {
ReflectedVisual.Opacity =0.85; }));
                Thread.Sleep(100);
                ReflectedVisual.Dispatcher.Invoke(System.Windows.Threading.DispatcherPriority.Send,    new    Action(delegate()    {
ReflectedVisual.Opacity = 1; }));
            }

        }
        private void BtnPrevHomeContent_Click(object sender, RoutedEventArgs e)
```

```
        {
            if (lBxHomeContentPreview.SelectedIndex > 0)
                lBxHomeContentPreview.SelectedIndex--;
            else
                lBxHomeContentPreview.SelectedIndex = (lBxHomeContentPreview.Items.Count - 1);
        }
        #endregion

        private void lBxHomeContentPreview_SelectionChanged(object sender, SelectionChangedEventArgs e)
        {
            lblCurrentHomeItemIndicator.Content = "" + (lBxHomeContentPreview.SelectedIndex + 1) + " / " +
(lBxHomeContentPreview.Items.Count);
        }

        private void btnFilterByType_Click(object sender, RoutedEventArgs e)
        {
            string content = (sender as Button).Content.ToString();
            switch (content)
            {
                case "All":
                    lBxContents.ItemsSource = ActiveContentList;
                    break;
                case "audios":
                    lBxContents.ItemsSource = from item in ActiveContentList
                                              where item.LibraryContentType == TypeOfContent.Audio
                                              select item;
                    break;
                case "videos":
                    lBxContents.ItemsSource = from item in ActiveContentList
                                              where item.LibraryContentType == TypeOfContent.Video
                                              select item;
                    break;
                case "pictures":
                    lBxContents.ItemsSource = from item in ActiveContentList
                                              where item.LibraryContentType == TypeOfContent.Picture
                                              select item;
                    break;
                case "books":
                    lBxContents.ItemsSource = from item in ActiveContentList
                                              where item.LibraryContentType == TypeOfContent.Book
                                              select item;
                    break;
                default:
                    lBxContents.ItemsSource = ActiveContentList;
                    break;
            }
        }

        List<string> Categories;
        public void BuildCategoryList()
        {
            Categories = new List<string>();
            Categories.Add("All");
            foreach (var content in ContentList)
            {
                if (!Categories.Contains(content.Category))
                    Categories.Add(content.Category);
            }
            CategoryLBx.ItemsSource = Categories;
        }

        private void CategoryLBx_SelectionChanged(object sender, SelectionChangedEventArgs e)
        {
            int index = -1;
            if ((index = CategoryLBx.SelectedIndex) > 0)
            {
                ActiveContentList = (from item in ContentList
                                     where item.Category == Categories[index]
                                     select item).ToList();

                CategoryName.Text = Categories[index];
            }
            else
            {
                ActiveContentList = ContentList;
                CategoryName.Text = "";
            }

            CurrentPage = 0;
            strtBdFadeContentListOff.Begin(this);

        }
```

```
        public List<ContentInfo> GetPagedContentList(int from, int to)
        {
            if (ActiveContentList == null || from < 0 || from >= ActiveContentList.Count)
                return null;
            else
            {
                if (to != -1 && to < ActiveContentList.Count)
                {
                    List<ContentInfo> PaginatedContents = new List<ContentInfo>();
                    for (int i = from; i < to; i++)
                    {
                        PaginatedContents.Add(ActiveContentList[i]);
                    }

                    return PaginatedContents;
                }

                else
                {
                    List<ContentInfo> PaginatedContents = new List<ContentInfo>();
                    for (int i = from; i < ActiveContentList.Count; i++)
                    {
                        PaginatedContents.Add(ActiveContentList[i]);
                    }

                    return PaginatedContents;
                }
            }
        }

        int itemsPerPage
        {
            get { return config.Content_Items_Per_Page; }
        }
        int CurrentPage;
        int TotalPages
        {
            get
            {
                if (ActiveContentList != null)
                {
                    int totalItems = ActiveContentList.Count;
                    if ((totalItems % itemsPerPage) > 0)
                        return (totalItems / itemsPerPage) + 1;
                    else
                        return (totalItems / itemsPerPage);
                }
                else
                    return 0;
            }
        }
        private void BtnPrevCatalogPage_Click(object sender, RoutedEventArgs e)
        {
            if (CurrentPage > 0)
            {
                --CurrentPage;
                stryBdFadeContentListOff.Begin(this);
            }
        }

        private void BtnNextCatalogPage_Click(object sender, RoutedEventArgs e)
        {
            if (CurrentPage < (TotalPages -1))
            {
                ++CurrentPage;
                stryBdFadeContentListOff.Begin(this);
            }
        }

        private void SetPage(int page)
        {
            CurrentPage = page;
            lBxContents.ItemsSource = GetPagedContentList(CurrentPage * itemsPerPage, (CurrentPage * itemsPerPage) +
itemsPerPage);
            if (TotalPages > 0)
                lblCurrentCatalogPageIndicator.Content = "Page " + (CurrentPage+1) + " of " + TotalPages;
            else
                lblCurrentCatalogPageIndicator.Content = "";
        }


    }
}
```

Application Configuration Class

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace DigitalLibraryx35
{
    public class AppConfig
    {
        public void Save()
        {
            Properties.Settings.Default.Save();
        }

        public void Load()
        {
            Properties.Settings.Default.Reload();
        }

        public void Reset()
        {
            Properties.Settings.Default.Reset();
        }


         public string Logon_UserName
        {
            get{return Properties.Settings.Default.Logon_Username ;}
            set { Properties.Settings.Default.Logon_Username = value; }
        }
         public string Logon_Password
        {
            get{return Properties.Settings.Default.Logon_Password ;}
            set { Properties.Settings.Default.Logon_Password = value; }
        }
         public string Service_Username
        {
            get{return Properties.Settings.Default.Service_Username ;}
            set { Properties.Settings.Default.Service_Username = value; }
        }
         public string Service_User_Code
        {
            get{return Properties.Settings.Default.Service_User_Code ;}
            set { Properties.Settings.Default.Service_User_Code = value; }
        }
         public string Service_Name
        {
            get{return Properties.Settings.Default.Service_Name ;}
            set { Properties.Settings.Default.Service_Name = value; }
        }
         public int Service_Port
        {
            get{return Properties.Settings.Default.Service_Port ;}
            set { Properties.Settings.Default.Service_Port = value; }
        }
         public string Server_IP
        {
            get{return Properties.Settings.Default.Server_IP ;}
            set { Properties.Settings.Default.Server_IP = value; }
        }
         public string Service_Protocol
         {
             get { return Properties.Settings.Default.Service_Type; }
             set { Properties.Settings.Default.Service_Type = value; }
         }
         public string Service_Address
         {
             get
             {
                 return Service_Protocol + "://" + Server_IP + ":" + Service_Port + "/" + Service_Name;
             }
         }
         public string UserInfo_Department
        {
            get{return Properties.Settings.Default.UserInfo_Department ;}
            set { Properties.Settings.Default.UserInfo_Department = value; }
        }
         public string UserInfo_Level
         {
```

```
            get { return Properties.Settings.Default.UserInfo_Level; }
            set { Properties.Settings.Default.UserInfo_Level = value; }
        }
        public string UserInfo_Preferences
        {
            get { return Properties.Settings.Default.UserInfo_Prefered; }
            set { Properties.Settings.Default.UserInfo_Prefered = value; }
        }
        public bool UserOptions_Auto_Connect
        {
            get { return Properties.Settings.Default.UserOptions_Auto_Connect; }
            set { Properties.Settings.Default.UserOptions_Auto_Connect = value; }
        }
        public bool UserOptions_Auto_Bookmark
        {
            get { return Properties.Settings.Default.UserOptions_Auto_Save_Bookmark; }
            set { Properties.Settings.Default.UserOptions_Auto_Save_Bookmark = value; }
        }
        public bool Application_Home_Isstartpage
        {
            get { return Properties.Settings.Default.Home_Is_Default_Page; }
            set { Properties.Settings.Default.Home_Is_Default_Page = value; }
        }
        public int Content_Items_Per_Page
        {
            get { return Properties.Settings.Default.Content_Items_Per_Page; }
            set { Properties.Settings.Default.Content_Items_Per_Page = value; }
        }


    }
}
```