P-3611

SECURED ACCESS OF HEALTHCARE DATA FROM MULTIPLE HOSPITALS

A PROJECT REPORT

Submitted by

S.ANANDHAKRISHNAN	0710108301
R.SANGETHARAJ	0710108304

In partial fulfillment for the award of the degree

of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE AND ENGINEERING

KUMARAGURU COLLEGE OF TECHNOLOGY

An Autonomous Institution

(Affiliated to Anna University of Technology, Coimbatore)

COIMBATORE – 641 049

APRIL 2011

KUMARAGURU COLLEGE OF TECHNOLOGY: COIMBATORE-641 049

BONAFIDE CERTIFICATE

Certified that this project report entitled "**SECURED ACCESS OF HEALTHCARE DATA FROM MULTIPLE HOSPITALS**" is the bonafide work of S.ANANDHAKRISHNAN and R.SANGETHARAJ who carried out the research under my supervision.

[illegible]

[Mr.Dinesh Ranganathan M.S.,]
Project Guide

[illegible]

[Mrs.S.Devaki M.E. (PhD),]
Head of the Department

The candidates with **university Register Nos. 0710108301 & 0710108304** was examined by us in project viva-voice examination held on 20.04.2011.

[illegible]

Internal Examiner

[illegible]

External Examiner

DECLARATION

We,

S.ANANDHAKRISHNAN	**Reg.No: 0710108301**
R.SANGETHARAJ	**Reg.No: 0710108304**

Hereby declare that the project entitled **"Secured Access of Healthcare data from Multiple Hospitals"**, submitted in partial fulfillment to Anna University as the project work of Bachelor of Engineering (Computer Science and Engineering) degree, is record of original work done by us under the supervision and guidance of Department of Computer Science and Engineering, Kumaraguru College of Technology, Coimbatore.

Place: Coimbatore
Date:

S.Anandhakrishnan
[S.Anandhakrishnan]

[illegible]
[R.Sangetharaj]

R.Dinesh

[Mr.Dinesh Ranganathan M.S.,]
Project Guide

ACKNOWLEDGEMENT

First and foremost, we would like to thank the Lord Almighty for enabling me to complete this project.

We express my profound gratitude to our Chairman **Padmabhusan Arutselvar Dr.N.Mahalingam, B.Sc., F.I.E.,** for giving this opportunity to pursue this course.

We would like to thank **Dr.S.Ramachandran, Ph.D.,** *Principal* for providing the necessary facilities to complete my thesis.

We take this opportunity to thank **Dr.S.Thangasamy Ph.D.,** *Dean, Research and Development,* for his precious suggestions. We also thank **Mrs.S.Devaki M.E. (PhD),** *HOD,* Department of Computer Science and Engineering, for her support and timely motivation.

We register my hearty appreciation to the Guide **Mr.Dinesh Ranganathan M.S.,** *Assistant Professor,* Department of Computer Science and Engineering, my Project advisor. We thank for him support, encouragement and ideas. We thank him for the countless hours he has spent with us, discussing everything from research to academic choices.

We would like to convey my honest thanks to all **Teaching** staff members and **Non Teaching** staffs of the department for their support. We would like to thank all my classmates who gave me a proper light moments and study breaks apart from extending some technical support whenever we needed them most.

TABLE OF CONTENTS

ABSTRACT

Intranet-based healthcare data exchange, which is particularly useful for the management of cooperative and life time healthcare records, requires the use of a common format to allow access to heterogeneous reservoirs of data scattered at different hospitals, as well as protection from intrusion and piracy. However, the unified management of multiple reservoirs is difficult to achieve, due to the different policies operated by different hospitals and the heterogeneous format of their information reservoirs.

We propose a software system to retrieve healthcare information in a common format through the intranet from multiple heterogeneous reservoirs. The proposed system consisting of the Healthcare Admin system, Hospital client system, common formatting system, user information system involving distributed processing with multiple matching agents connected to heterogeneous reservoirs and one flexible master controller to unify the different formats and different hospital policies, thus providing a secure common format and simplifying the problem of reservoir maintenance including the addition, removal and modification of reservoirs. The XML provides an efficient means of reservoir management, allowing a common format for information exchange, device independent display for diverse display resolutions of terminal devices, user identification for authentication, digital signature for data integrity, and selective encryption for protecting confidential health information.

1 .INTRODUCTION

1.1GENERAL:

Intranet-based healthcare data exchange, which is particularly useful for the management of cooperative healthcare and life time healthcare records, requires the use of a common format to allow access to heterogeneous reservoirs scattered at different hospitals, as well as protection from intrusion and piracy. However, the unified management of multiple reservoirs is difficult to achieve, due to the different policies operated by different hospitals and the heterogeneous format of their information reservoirs.

1.2EXISTING SYSTEM:

Healthcare organizations generate various documents for each patient including diagnosis, treatment advised, blood test reports, pathology results, X-ray reports, analysis reports etc.. Each of these documents is critical and may be referred to. Further, there are a number of medico-legal cases that are admitted in the hospital and there is a statutory obligation on the part of the hospital to provide the details of any specific case if such a request from competent authority (courts or police or investigation agencies). This implies that the hospital administration typically spends lot of time trying to retrieve these records and maintaining them.

DRAWBACKS

Often due to the deluge of documents making it impossible to locate the correct record which can cause many problems operationally and legally. Also since the hard copies cannot be shared there is unnecessary and tedious retyping of data or creation of copies of the document.

1.3 PROPOSED SYSTEM:

- A Healthcare software system to convert different format of patient details collected from different hospitals into common XML format and to store the patient details of common format as a XML document in a secured manner.

- Patient records can be retrieved easily and instantaneously

- The documents can be shared over the LAN/intranet thus preventing wasteful copies

- Documents retrieved can be emailed to the concerned doctor or anyone else

- Since the patient history is available instantaneously, the valuable time of specialists etc. is saved

- With documents in digital format and with backup features, ensures the safety of documents

1.4 OBJECTIVE OF THE PROJECT

- Basic objective of this project is to create a digital health care software system for managing the health documents from various hospitals in single digital signed system and provide access to the documents.

- Creates a central electronic repository of the patient records, which can be scanned into the system.

- Folders for each patient can be created and security can be applied which ensures confidentiality of the documents

- The documents can be indexed with data like patient name, patient number, date etc...

- Any documents related to the patient which are generated electronically (like TEXT, SQL, ACCESS files) can be directly imported into the system; digital signature is added and converted to XML then encrypted.

2. LITERATURE REVIEW

2.1 HARDWARE REQUIREMENTS

• Processor	:	Intel Pentium P4 1.7 GHZ
• RAM	:	512 MB
• Hard Disk Drive	:	80 GB
• Keyboard	:	101 Keys
• Mouse	:	Optical Mouse
• Monitor	:	SVGA/color

2.2 SOFTWARE REPQIREMENTS

• Operating System	:	Windows XP 3
• IDE used	:	Visual Studio .Net Framework 2010
• Web Technologies	:	XML.
• DATABASES	:	SQL Server 2008, MS ACCESS 2000.
• Language Used	:	Visual C#

2.3 SOFTWARE DESCRIPTION

- **VISUAL STUDIO .NET FRAME WORK 4.0**

The Microsoft .NET Framework is a software framework that can be installed on computers running Microsoft Windows operating systems. It includes a large library of coded solutions to common programming problems and a virtual machine that manages the execution of programs written specifically for the framework. The .NET Framework is a Microsoft offering and is intended to be used by most new applications created for the Windows platform. The framework's Base Class Library provides a large range of features including user interface, data access, database connectivity, cryptography, web application development, numeric algorithms, and network communications. The class library is used by programmers, who combine it with their own code to produce applications. 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.

- **WINDOWS PRESENTATION FOUNDATION:**

This subsystem is a part of .NET Framework 3.0.The Windows Presentation Foundation (or WPF) is a graphical subsystem for rendering user interfaces in Windows-based applications. WPF, previously known as "Avalon", was initially released as part of .NET Framework 3.0. Designed to remove dependencies on the aging GDI subsystem, WPF is built on DirectX, which provides hardware acceleration and enables modern UI features like transparency, gradients, and transforms. WPF provides a consistent programming model for building applications and provides a clear separation between the user interface and the business logic.

WPF also offers a new markup language, known as XAML, which is an alternative means for defining UI elements and relationships with other UI elements. A WPF application can be deployed on the desktop or hosted in a web browser. It also enables rich control, design, and development of the visual aspects of Windows programs.

It aims to unify a number of application services: user interface, 2D and 3D drawing, fixed and adaptive documents, advanced typography, vector graphics, raster graphics, animation, data binding, audio and video. Microsoft Silverlight is a web-based subset of WPF that enables Flash-like web and mobile applications with the same programming model as .NET applications.

- **GRAPHICAL SERVICES**

All graphics, including desktop items like windows, are rendered using Direct3D. This aims to provide a unified avenue for displaying graphics and is the enabling factor that allows 2D, 3D, media, and animation to be combined in a single window. Allows more advanced graphical features when compared to Windows Forms and its GDI underpinnings.

- **MEDIA SERVICES**

WPF provides an integrated system for building user interfaces with common media elements like vector and raster images, audio, and video. WPF also provides an animation system and a 2D/3D rendering system. WPF provides shape primitives for 2D graphics along with a built-in set of brushes, pens, geometries, and transforms. The 3D capabilities in WPF are a subset of the full-feature set provided by Direct3D. However, WPF provides tighter integration with other features like user interfaces, documents, and media. This makes it possible to have 3D user interfaces, 3D documents, or 3D media. There is support for most common image formats: BMP, JPEG, PNG, TIFF, Windows Media Photo, GIF, and ICON.

- **ANIMATIONS**

WPF supports time-based animations, in contrast to the frame-based approach. This decouples the speed of the animation from how the system is performing. WPF supports low level animation via timers and higher level abstractions of animations via the Animation classes. Any WPF element property can be animated as long as it is registered as a Dependency Property. Animation classes are based on the .NET type of property to be animated. For instance, changing the color of an element is done with the Color Animation class and animating the Width of an element is done with the Double Animation class.

- **EFFECTS**

WPF 3.0 provides for Bitmap Effects, which are raster effects applied to a Visual. These raster effects are written in unmanaged code and force rendering of the Visual to be performed on the CPU and not hardware accelerated by the GPU. Bitmap Effects were deprecated in .NET 3.5 SP 1.The .NET Framework 3.5 SP1 adds the Effect class, which is a Pixel-Shader 2.0 effect that can be applied to a visual, which allows all rendering to remain on the GPU.The Effect class is extensible allowing application to specify their own shader effects..NET 3.5 SP1 ships with two built-in effects, Blur Effect and DropShadowEffect.

- **ASP.NET**

ASP.NET is a unified Web development model that includes the services necessary for you to build enterprise-class Web applications with a minimum of coding. ASP.NET is part of the .NET Framework, and when coding ASP.NET applications you have access to classes in the .NET Framework. You can code your applications in any language compatible with the common language runtime (CLR), including Microsoft Visual Basic and C#. These languages enable you to develop ASP.NET applications that benefit from the common language runtime, type safety, inheritance, and so on.

Visual Web Developer is a full-featured development environment for creating ASP.NET Web applications. Visual Web Developer offers you the following features:

- **Web page designs** A powerful Web page editor that includes WYSIWYG editing and an HTML editing mode with IntelliSense and validation.

- **Page design features** Consistent site layout with master pages and consistent page appearance with themes and skins.

- **Code editing** A code editor that enables you to write code for your dynamic Web pages in Visual Basic or C#. The code editor includes syntax coloration and IntelliSense.

- **Testing and debugging** A local Web server for testing and a debugger that helps you find errors in your programs.

- **Deployment** Tools to automate typical tasks for deploying a Web application to a hosting server or a hosting provider.

- **VISUAL C#:**

  C# (pronounced "see sharp") is a multi-paradigm programming language encompassing imperative, 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. Its development team is led by Anders Hejlsberg. The most recent version is C# 3.0, which was released in conjunction with the .NET Framework 3.5 in 2007. The next proposed version, 4.0, is in development.

- **FEATURES**

  By design, C# is the programming language that most directly reflects the underlying Common Language Infrastructure (CLI). Most of its intrinsic types correspond to value-types implemented by the CLI framework. However, the language specification does not state the code generation requirements of the 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. Theoretically, a C# compiler could generate machine code like traditional compilers of C++ or FORTRAN.

Some notable distinguishing features of C# are:

- There are no global variables or functions. All methods and members must be declared within classes. Static members of public classes can substitute for global variables and functions.
- Local variables cannot shadow variables of the enclosing block, unlike C and C++. Variable shadowing is often considered confusing by C++ texts.
- C# supports a strict Boolean datatype, bool. Statements that take conditions, such as while and if, require an expression of a type that implements the true operator, such as the boolean type. While C++ also has a boolean type, it can be freely converted to and from integers, and expressions such as if(a) require only that a is convertible to bool, allowing a to be an int, or a pointer. C# disallows this "integer meaning true or false" approach on the grounds that forcing programmers to use expressions that return exactly bool can prevent certain types of common programming mistakes in C or C++ such as if (a = b) (use of assignment = instead of equality ==).
- In C#, memory address pointers can only be used within blocks specifically marked as unsafe, and programs with unsafe code need appropriate permissions to run. Most object access is done through safe object references, which always either point to a "live" object or have the well-defined null value; it is impossible to obtain a reference to a "dead" object (one which has been garbage collected), or to a random block of memory. An unsafe pointer can point to an instance of a value-type, array, string, or a block of memory allocated on a stack. Code that is not marked as unsafe can still store and manipulate pointers through the System.IntPtr type, but it cannot dereference them.
- Managed memory cannot be explicitly freed; instead, it is automatically garbage collected. Garbage collection addresses the problem of memory leaks by freeing the programmer of responsibility for releasing memory which is no longer needed.

- In addition to the try...catch construct to handle exceptions, C# has a try...finally construct to guarantee execution of the code in the finally block.
- Multiple inheritances is not supported, although a class can implement any number of interfaces. This was a design decision by the language's lead architect to avoid complication and simplify architectural requirements throughout CLI.
- C# is more type safe than C++. The only implicit conversions by default are those which are considered safe, such as widening of integers. This is enforced at compile-time, during JIT, and, in some cases, at runtime. There are no implicit conversions between Booleans and integers, nor between enumeration members and integers (except for literal 0, which can be implicitly converted to any enumerated type). Any user-defined conversion must be explicitly marked as explicit or implicit, unlike C++ copy constructors and conversion operators, which are both implicit by default.
- Enumeration members are placed in their own scope.
- C# provides properties as syntactic sugar for a common pattern in which a pair of methods, access or (getter) and mutator (setter) encapsulate operations on a single attribute of a class.
- C# currently (as of 3 June 2008) has 77 reserved words.
- C# is the programming language that most directly reflects the underlying Common Language Infrastructure (CLI). Most of its intrinsic types correspond to value-types implemented by the CLI framework. However, the language specification does not state the code generation requirements of the 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. Theoretically, a C# compiler could generate machine code like traditional compilers of C++ or FORTRAN. With Visual developers can build solutions for the broadest range of clients, including Windows, the Web, and mobile or embedded devices. Using this elegant programming language and tool, developers can leverage their existing C++ and Java-language skills and knowledge to be successful in the .NET environment.
- C# is more type safe than C++. The only implicit conversions by default are those which are considered safe, such as widening of integers. This is enforced at compile-time, during JIT, and, in some cases, at runtime. There are no implicit conversions between booleans and

integers, or between enumeration members and integers (except for literal 0, which can be implicitly converted to any enumerated type).

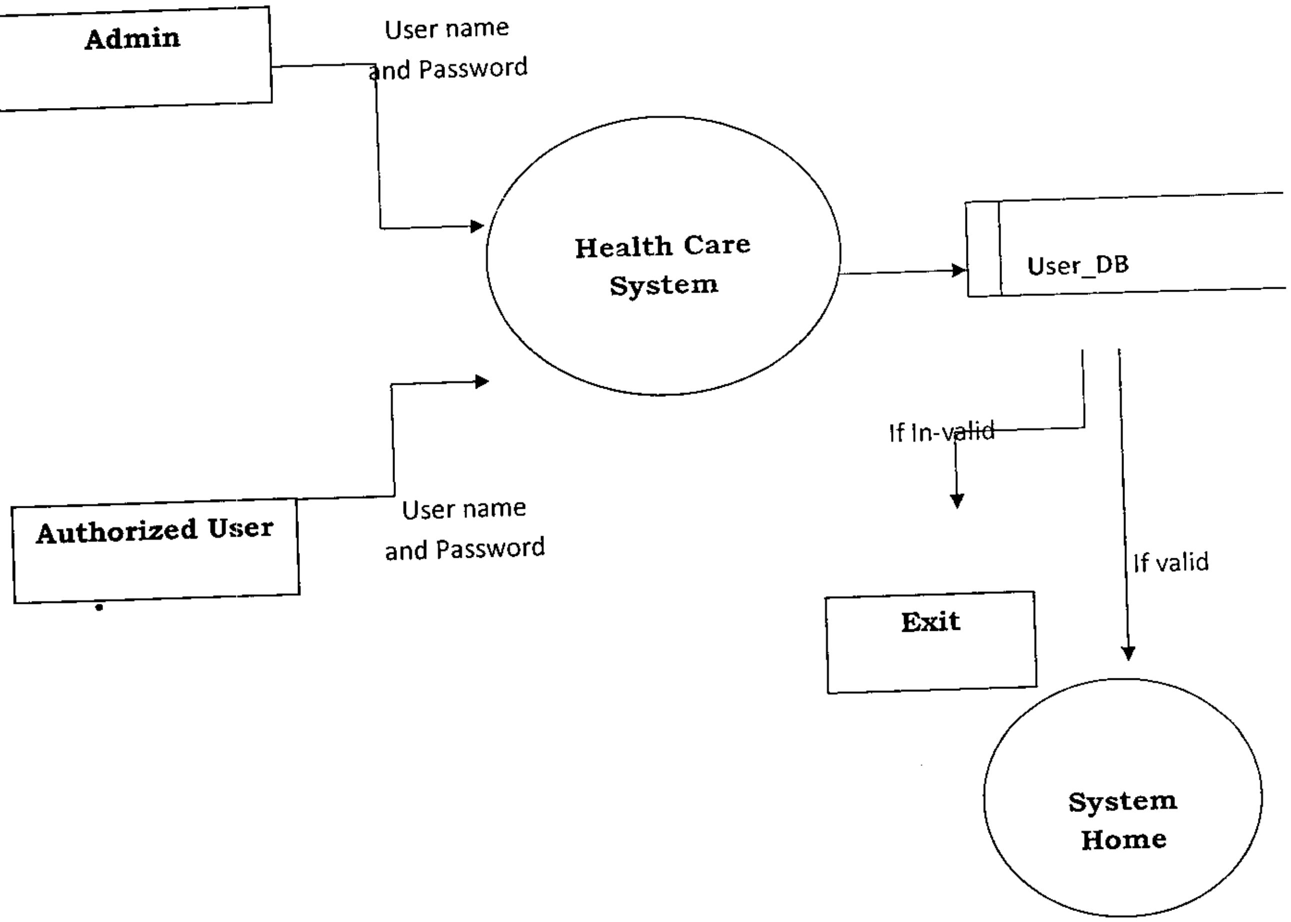
- Any user-defined conversion must be explicitly marked as explicit or implicit, unlike C++ copy constructors and conversion operators, which are both implicit by default. In C#, memory address pointers can only be used within blocks specifically marked as unsafe, and programs with unsafe code need appropriate permissions to run. Most object access is done through safe object references. Managed memory cannot be explicitly freed; instead, it is automatically garbage collected.

3. DETAILS OF METHODOLOGY EMPLOYED

3.1 IMPLEMENTATION OF CLIENT APPLICATION

This System allows the registered hospitals to login into the system using their Hospital ID and the hospitals can upload their patient details of own format.

DFD for User Login



3.2 IMPLEMENTATION OF HEALTHCARE ADMIN

- This System is used by the administrator to register the hospitals in the database and to view and delete the hospitals from the database.
- By registering, Hospitals can get access to the system with their Hospital ID provided by Administrator.

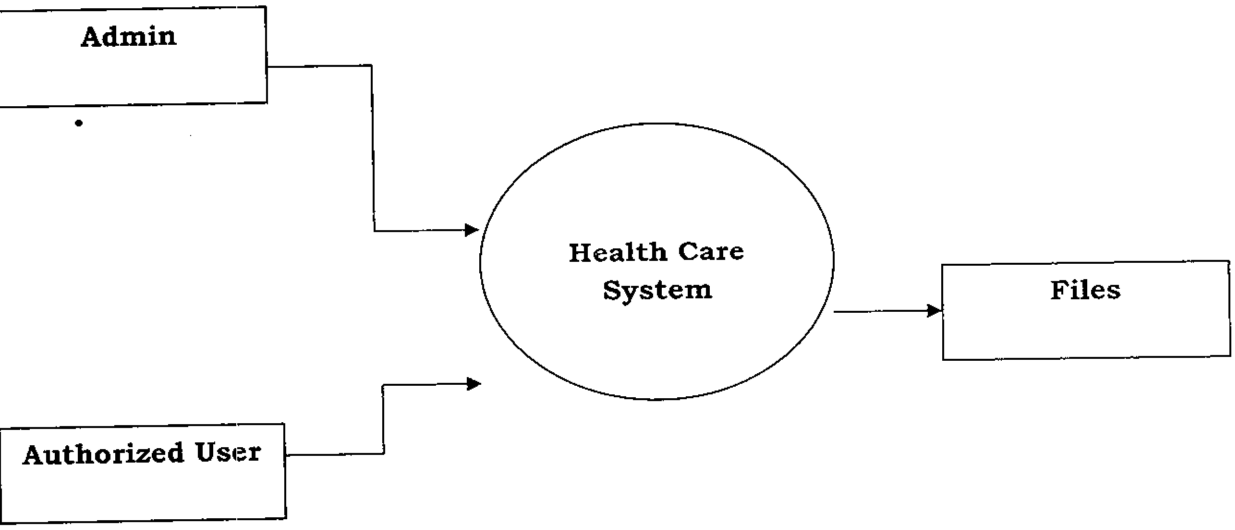
3.3 COMMON FORMATTING SYSTEM

- This System converts different format of patient details into common XML format and provides a User ID to the patients with respect to their hospitals.
- Different data format such as text, MS-Access and Sql server

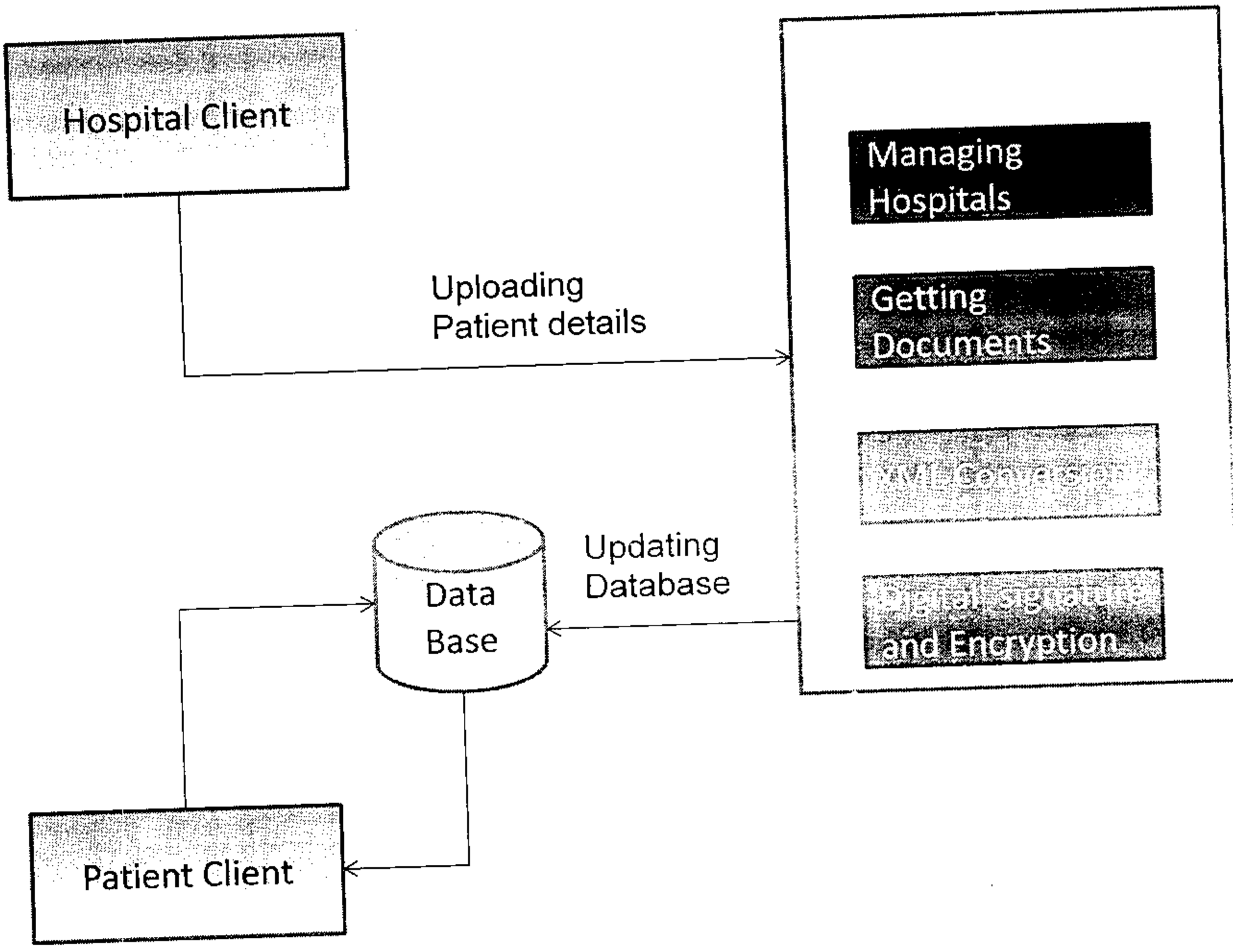
3.4 USER INFORMATION SYSTEM IMPLEMENTATION

This system allows the patients to login using their valid user ID and hospital name to get their health care report.

DATA FLOW DIAGRAM



4. SYSTEM FLOW DIAGRAM



5. PERFORMANCE EVALUATION

5.1 SYSTEM DESIGN

FUNDAMENTAL DESIGN CONCEPTS

System design sits in the technical kernel of software engineering and applied science regardless of the software process model that is used. Beginning once the software requirements have been analyzed and specified, tests that are required in the building and verifying the software is done. Each activity transforms information in a number that ultimately results in validated computer software.

There are mainly three characteristics that serve as guide for evaluation of good design,

- The design must be implement all of explicit requirements contained in the analysis model, and it must accommodate all of the implicit requirements desired by the customer.
- The design must be readable, understandable guide for those who generate code and for those who test and subsequently support the software.
- The design should provide a complete picture of software, addressing the data, its functional and behavioral domains from the implementation perspective.

System design is thus a process of planning a new system or to replace or the complement of the existing system. The design based on the limitations of the existing system and the requirements specification gathered in the phase of system analysis.

Input design is the process of converting the user-oriented description of the computer based business information into program-oriented specification. The goal of designing input data is to make the automation as easy and free from errors as possible.

Logical Design of the system is performed where its features are described, procedures that meet the system requirements are formed and a detailed specification of the new system is provided

Architectural Design of the system includes identification of software components, decoupling and decomposing them into processing modules, conceptual data structures and specifying relationship among the components.

Detailed Design is concerned with the methods involved in packaging of processing modules and implementation of processing algorithms, data structure and interconnection among modules and data structure.

External Design of software involves conceiving, planning and specifying the externally observable characteristics of the software product. The external design begins in the analysis phase and continues till the design phase.

As per the design phase the following designs had to be implemented, each of these design were processed separately keeping in mind all the requirements, constraints and conditions. A step-by-step process was required to perform the design.

Process Design is the design of the process to be done; it is the designing that leads to the coding. Here the conditions and the constraints given in the system are to be considered. Accordingly the designing is to be done and processed.

The **Output Design** is the most important and direct source of information to the user. The output design is an ongoing activity during study phase. The objectives of the output design define the contents and format of all documents and reports in an attractive and useful format.

The main output generated here is the reports. The reports were generated for selective reasons. The various reports generated are as follows,

- Reports producing the details of the customer.
- Reports producing the order status.
- Reports producing the date wise, month wise order details.
- Reports producing the current stock as on date.
- Report on the available products in the company.
- Report on the transaction of the products between company and the user.

5.2 SYSTEM TESTING

Software testing is a critical element of software quality assurance and represents the ultimate review of specification, design and code generation. Once the source code has been generated, software must be tested to uncover as many errors as possible before delivery to the customer. In order to find the highest possible number of errors, tests must be conducted systematically and test cases must be designed using disciplined techniques.

TYPES OF TESTING

- **White box Testing**

White box testing sometimes called as glass box testing is a test case design method that uses the control structures of the procedural design to derive test cases.

Using White Box testing methods, the software engineer can derive test case, that guarantee that all independent paths with in a module have been exercised at least once, exercise all logical decisions on their true and false sides, execute all loops at their boundaries and within their operational bounds, exercise internal data structures to ensure their validity. "Logic errors and incorrect assumptions are inversely proportional to the probability that a program path will be executed".

The logical flow of a program is sometimes counterintuitive, meaning that unconscious assumptions about flow of control and data may lead to make design errors that are uncovered only once path testing commences.

"Typographical errors are random"

When a program is translated into programming language source code, it is likely that some typing errors will occur. Many will be uncovered by syntax and typing checking mechanisms, but others may go undetected until testing begins. It is as likely that a type will exist on an obscure logical path as on a mainstream path.

- **Black box Testing**

Black box testing, also called as behavioral testing, focuses on the functional requirements of the software. That is, black box testing enables the software engineer to derive sets of input conditions that will fully exercise all functional requirements for a program. Black box testing attempts to find errors in the following categories:

1. Incorrect or missing functions
2. Interface errors
3. Errors in data structures or external data base access
4. Behavior or performance errors
5. Initialization and termination errors

By applying black box techniques, a set of test cases that satisfy the following criteria were been created: Test cases that reduce, by a count that is greater than one, the number of additional test cases that must be designed to achieve reasonable testing and test cases that tell something about the presence or absence of classes of errors, rather than an error associated only with the specific test at hand.

Black - box testing is not an alternative to white - box testing techniques. Rather it is complementary approach that is likely to uncover a different class of errors than white - box methods.

- ***Unit Testing***

Unit testing focuses verification effort on the smallest unit of the software design, the module. Using the procedural design description as a guide, important control paths are tested to uncover errors within the boundary of the module. This testing can be conducted in parallel for all the modules.

- **Integration Testing**

The objective of the integration testing is to take modules which are tested successfully during unit test and build a program structure that has been dictated by specification. All the three modules are combined in this testing. The entire program is tested as a whole.

- ***Validation Testing***

Validation testing tests "Are we building the product right?" It finds out the deviations from the specification and prepares a deviation list. It includes two type of testing namely alpha and beta testing.

Alpha tests are carried out at the control environment the customer tests the software in the presence of the developer. This is called looking over the shoulder.

In beta testing, the customer tests the software in the absence of the developer. This is called live testing. Before the release, the software is used for trail.

- ***Output Testing***

No system could be useful if it does not produce the required output in the specific format. Output testing is performed to ensure the correctness of the output and its format.

6. CONCLUSION

Thus, the proposed system, which provides a means of accessing healthcare information located in multiple heterogeneous reservoirs located at different hospitals, by retrieving it via the intranet and converting it to a common format. The proposed system consisting of Healthcare Admin system, Hospital client system, common formatting system, user information system, which involves distributed processing with multiple heterogeneous HIS (Human Information System). These multiple agents with one central controller can interconnect with heterogeneous reservoirs having different formats and different hospital policies, while providing the information in a secure common format, as well as simplifying the complex operations of reservoir maintenance, including the addition, removal and modification of reservoirs. The XML provides an efficient means of reservoir management, allowing a common format for information exchange, device independent display for the diverse display resolutions of terminal devices, user identification for authendification, digital signature for data integrity, and encryption for protecting confidential health information. The proposed system facilitates the establishment of a patient-centered framework in which to exchange health care data over the intranet, particularly in the case of multi-institutional multispecialist cooperative healthcare and life-time health care records. In future, auditing features could be added to judge the approval of this HIS by different hospitals enrolled in the usability of the software system.

7. APPENDICES

7.1 SOURCE CODE

ADMIN- HOSPITAL REGISTRATION AND VIEWING:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;

namespace HealthCareSys
{
public partial class AdminHome : Form
{
public AdminHome()
{
InitializeComponent();
}

private void button1_Click(object sender, EventArgs e)
{
HospitalDBDataContext Datas = new HospitalDBDataContext();
Hospital Data = new Hospital();
Data.ContactAddress = txtaddress.Text;
Data.Contactno = txtno.Text;
Data.HospitalID = txthosid.Text;
Data.Name = txtname.Text;
Data.Pass = txtpass.Text;
Data.Status = comboBox1.Text;
Datas.Hospitals.InsertOnSubmit(Data);
Datas.SubmitChanges();
MessageBox.Show("Hospital Registered");
}
private void button2_Click(object sender, EventArgs e)
{
HospitalDBDataContext Datas = new HospitalDBDataContext();
var dt = from k in Datas.Hospitals select k;
dataGridView1.DataSource = dt.ToList();
}
}
private void button3_Click(object sender, EventArgs e)
{
HospitalDBDataContext Datas = new HospitalDBDataContext();
var dy = (from k in Datas.Hospitals
where k.HospitalID == txthosid.Text
select k).First();
Datas.Hospitals.DeleteOnSubmit(dy);
Datas.SubmitChanges();
MessageBox.Show("Deleted ...");
}
}
}
```

XML CLASS:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Xml.Linq;
using System.Security.Cryptography;
using System.Xml;
using System.Security.Cryptography.Xml;

namespace CovertToXML
{
public class ConvertToXML
{
public XElement ConvertText(List<DocType> Docs)
{
XElement d = XElement.Load(@"D:\HealthCareSys\HealthCareSys\Test.xml");
foreach (var doc in Docs)
{
XElement child = new XElement("Report",
new XAttribute("AccessBy", "public"),
new XElement("Hospital", doc.HospitalName),
new XElement("Patient", doc.PatientName),
new XElement("OnDate", doc.OnDate),
new XElement("Phone", doc.PhoneNO),
new XElement("Subject", doc.Subject),
new XElement("Symptom", doc.Symptoms),
new XElement("Treatment", doc.Treatments),
new XElement("Address", doc.Address));
```

```
d.Add(child);
}
return d;
}
//we used the classes in the System.Security.Cryptography.Xml namespace
//to sign an XML document or part of an XML document with a digital signature.
//XML digital signatures (XMLDSIG) allow you to verify that data was not altered after it was
signed
public void SignDocument(string path)
{
try
{
// Create a new CspParameters object to specify
// a key container.
CspParameters cspParams = new CspParameters();
cspParams.KeyContainerName = "XML_DSIG_RSA_KEY";

// Create a new RSA signing key and save it in the container.
RSACryptoServiceProvider rsaKey = new RSACryptoServiceProvider(cspParams);

// Create a new XML document.
XmlDocument xmlDoc = new XmlDocument();

// Load an XML file into the XmlDocument object.
xmlDoc.PreserveWhitespace = true;
xmlDoc.Load(path + ".xml");

// Sign the XML document.
SignXml(xmlDoc, rsaKey);
```

```
// Save the document.
xmlDoc.Save(path + "signed.xml");
}
catch (Exception e)
{
Console.WriteLine(e.Message);
}
}
// Sign an XML file.
// This document cannot be verified unless the verifying
// code has the key with which it was signed.
public static void SignXml(XmlDocument xmlDoc, RSA Key)
{
// Check arguments.
if (xmlDoc == null)
throw new ArgumentException("xmlDoc");
if (Key == null)
throw new ArgumentException("Key");

// Create a SignedXml object.
SignedXml signedXml = new SignedXml(xmlDoc);

// Add the key to the SignedXml document.
signedXml.SigningKey = Key;

// Create a reference to be signed.
Reference reference = new Reference();
reference.Uri = "";

// Add an enveloped transformation to the reference.
XmlDsigEnvelopedSignatureTransformenv=new        XmlDsigEnvelopedSignatureTransform();
```

```
reference.AddTransform(env);
// Add the reference to the SignedXml object.
signedXml.AddReference(reference);

// Compute the signature.
signedXml.ComputeSignature();

// Get the XML representation of the signature and save
// it to an XmlElement object.
XmlElement xmlDigitalSignature = signedXml.GetXml();

// Append the element to the XML document.
xmlDoc.DocumentElement.AppendChild(xmlDoc.ImportNode(xmlDigitalSignature, true));
}
public string VerifySign(string path)
{
// Create a new CspParameters object to specify
// a key container.
CspParameters cspParams = new CspParameters();
cspParams.KeyContainerName = "XML_DSIG_RSA_KEY";

// Create a new RSA signing key and save it in the container.
RSACryptoServiceProvider rsaKey = new RSACryptoServiceProvider(cspParams);

// Create a new XML document.
XmlDocument xmlDoc = new XmlDocument();

// Load an XML file into the XmlDocument object.
xmlDoc.PreserveWhitespace = true;
xmlDoc.Load(path + "signed.xml");
```

```
// Verify the signature of the signed XML.

bool result = VerifyXml(xmlDoc, rsaKey);

// Display the results of the signature verification to
// the console.
if (result)
{
return ("The XML signature is valid.");
}
else
{
return ("The XML signature is not valid.");
}


}
// Verify the signature of an XML file against an asymmetric
// algorithm and return the result.
public static Boolean VerifyXml(XmlDocument Doc, RSA Key)
{
// Create a new SignedXml object and pass it
// the XML document class.
SignedXml signedXml = new SignedXml(Doc);

// Find the "Signature" node and create a new
// XmlNodeList object.
XmlNodeList nodeList = Doc.GetElementsByTagName("Signature");

// Throw an exception if no signature was found.
if (nodeList.Count <= 0)
{
```

```
throw new CryptographicException("Verification failed: No Signature was found in the
document.");
}
// This example only supports one signature for
// the entire XML document.  Throw an exception
// if more than one signature was found.
if (nodeList.Count >= 2)
{
throw new CryptographicException("Verification failed: More that one signature was found for
the document.");
}

// Load the first <signature> node.
signedXml.LoadXml((XmlElement)nodeList[0]);

// Check the signature and return the result.
return signedXml.CheckSignature(Key);
}
}
}
```

XML CONVERSION:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using CovertToXML;
using System.Xml.Linq;
using System.Xml;

namespace CareClient
{
public partial class Home : Form
{
public Home()
{
InitializeComponent();
}

private void Home_Load(object sender, EventArgs e)
{
this.Text = SessionData.HospitalSession + "  - Health Care System.";
}

private void loadDocumentsToolStripMenuItem_Click(object sender, EventArgs e)
{
```

```
if (SessionData.HospitalSession.ToLower() == "kg")
{
HDocuDataContext Datas = new HDocuDataContext();
var dt = from k in Datas.HosDocuments
select k;
foreach (var t in dt)
{
listBox1.Items.Add(t.documentid);
}
}
else if (SessionData.HospitalSession.ToLower() == "apollo")
{
System.Data.OleDb.OleDbConnection Con = new System.Data.OleDb.OleDbConnection(); ;
Con.ConnectionString          =          @"Provider=Microsoft.Jet.OLEDB.4.0;Data
Source=D:\HealthCareSys\HealthCareSys\Apollo.mdb;Persist Security Info=True";
Con.Open();

System.Data.OleDb.OleDbCommand Cmd = new System.Data.OleDb.OleDbCommand();
Cmd.CommandText = "select documentid from healthtb";
System.Data.DataSet Ds = new DataSet();
System.Data.OleDb.OleDbDataAdapter Da = new System.Data.OleDb.OleDbDataAdapter();
Cmd.Connection = Con;
Da.SelectCommand = Cmd;
Da.Fill(Ds, "HealthTB");

for (int i = 0; i < Ds.Tables["HealthTB"].Rows.Count; i++)
{
listBox1.Items.Add(Ds.Tables["HealthTB"].Rows[i]["documentid"].ToString());
}
}
else
```

```
{
string startFolder = @"D:\" + SessionData.HospitalSession + @"\";


// Take a snapshot of the file system.
System.IO.DirectoryInfo dir = new System.IO.DirectoryInfo(startFolder);

// This method assumes that the application has discovery permissions
// for all folders under the specified path.
IEnumerable<System.IO.FileInfo>          fileList          =          dir.GetFiles("*.*",
System.IO.SearchOption.AllDirectories);

//Create the query
IEnumerable<System.IO.FileInfo> fileQuery =
from file in fileList
where file.Extension == ".txt"
orderby file.Name
select file;
foreach (var f in fileQuery)
{
listBox1.Items.Add(f.FullName.ToString());
}
}
}

private void listBox1_SelectedIndexChanged(object sender, EventArgs e)
{
if (SessionData.HospitalSession.ToLower() == "kg")
{
HDocuDataContext Datas = new HDocuDataContext();
var dt = from k in Datas.HosDocuments
where k.documentid == listBox1.SelectedItem.ToString()
select k;
dataGridView1.DataSource = dt.ToList();
}
else if (SessionData.HospitalSession.ToLower() == "apollo")
{
System.Data.OleDb.OleDbConnection Con = new System.Data.OleDb.OleDbConnection(); ;
Con.ConnectionString = @"Provider=Microsoft.Jet.OLEDB.4.0;Data Source=D:\HealthCareSys\HealthCareSys\Apollo.mdb;Persist Security Info=True";
Con.Open();

System.Data.OleDb.OleDbCommand Cmd = new System.Data.OleDb.OleDbCommand();
Cmd.CommandText = "select * from healthtb where documentid ='" + listBox1.SelectedItem.ToString() + "'";
System.Data.DataSet Ds = new DataSet();
System.Data.OleDb.OleDbDataAdapter Da = new System.Data.OleDb.[illegible]();
Cmd.Connection = Con;
Da.SelectCommand = Cmd;
Da.Fill(Ds, "HealthTB");

dataGridView1.DataSource = Ds.Tables["HealthTB"];
}
else
{
richTextBox1.LoadFile(listBox1.SelectedItem.ToString(), RichTextBoxStreamType.PlainText);
}
}

private void button1_Click(object sender, EventArgs e)
{
}
```

```
private void uploadDocumentsToolStripMenuItem_Click(object sender, EventArgs e)
{
if (SessionData.HospitalSession.ToLower() == "kg")
{
List<DocType> Docs = new List<DocType>();
HDocuDataContext Datas = new HDocuDataContext();
var dt = from k in Datas.HosDocuments
select k;
foreach (var t in dt)
{
DocType SqlDoc = new DocType();
SqlDoc.Address = t.address;
SqlDoc.Content = "";
SqlDoc.HospitalName = t.hospitalid;
SqlDoc.OnDate = DateTime.Now.ToString();
SqlDoc.PatientName = t.PatientId;
SqlDoc.PhoneNO = t.phone;
SqlDoc.Subject = t.disease;
SqlDoc.Symptoms = t.Symptom;
SqlDoc.Treatments = t.doctortreated;
Docs.Add(SqlDoc);

}
ConvertToXML DocConvert = new ConvertToXML();

XElement tempfile = DocConvert.ConvertText(Docs);
tempfile.Save(@"D:\HealthCareSys\HealthCareSys\" + SessionData.HospitalSession + ".xml");
DocConvert.SignDocument(@"D:\HealthCareSys\HealthCareSys\"                              +
SessionData.HospitalSession);
```

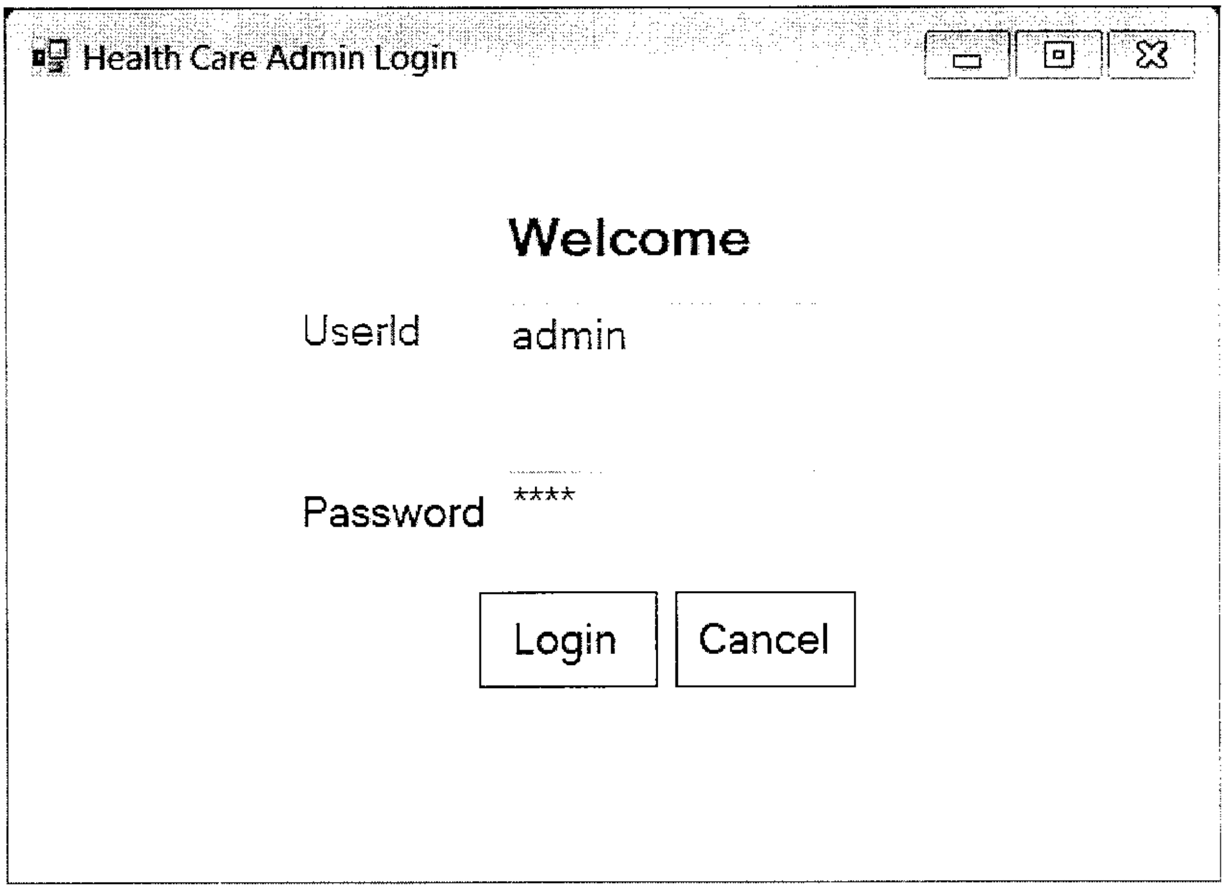
```
MessageBox.Show(DocConvert.VerifySign(@"D:\HealthCareSys\HealthCareSys\"                    +
SessionData.HospitalSession));
EncryptXML d = new EncryptXML();

MessageBox.Show(d.Encrypt(@"D:\HealthCareSys\HealthCareSys\"                                +
SessionData.HospitalSession, "p1").ToString());
}
else  if (SessionData.HospitalSession.ToLower() == "apollo")
{
List<DocType> Docs = new List<DocType>();

System.Data.OleDb.OleDbConnection Con = new System.Data.OleDb.OleDbConnection(); ;
Con.ConnectionString              =              @"Provider=Microsoft.Jet.OLEDB.4.0;Data
Source=D:\HealthCareSys\HealthCareSys\Apollo.mdb;Persist Security Info=True";
Con.Open();

System.Data.OleDb.OleDbCommand Cmd = new System.Data.OleDb.OleDbCommand();
Cmd.CommandText = "select * from healthtb";
System.Data.DataSet Ds = new DataSet();
System.Data.OleDb.OleDbDataAdapter Da = new System.Data.OleDb.OleDbDataAdapter();
Cmd.Connection = Con;
Da.SelectCommand = Cmd;
Da.Fill(Ds, "HealthTB");

for (int i = 0; i < Ds.Tables["HealthTB"].Rows.Count; i++)
{
DocType SqlDoc = new DocType();
SqlDoc.Address = Ds.Tables["HealthTB"].Rows[i]["Address"].ToString();
SqlDoc.Content = "";
SqlDoc.HospitalName = Ds.Tables["HealthTB"].Rows[i]["hospitalid"].ToString();
SqlDoc.OnDate = DateTime.Now.ToString();
SqlDoc.PatientName = Ds.Tables["HealthTB"].Rows[i]["patientid"].ToString();
SqlDoc.PhoneNO = Ds.Tables["HealthTB"].Rows[i]["phone"].ToString();
SqlDoc.Subject = Ds.Tables["HealthTB"].Rows[i]["disease"].ToString();
SqlDoc.Symptoms = Ds.Tables["HealthTB"].Rows[i]["symptom"].ToString();
SqlDoc.Treatments = Ds.Tables["HealthTB"].Rows[i]["doctortreated"].ToString();
Docs.Add(SqlDoc);

}
ConvertToXML DocConvert = new ConvertToXML();

XElement tempfile = DocConvert.ConvertText(Docs);
tempfile.Save(@"D:\HealthCareSys\HealthCareSys\" + SessionData.HospitalSession + ".xml");
DocConvert.SignDocument(@"D:\HealthCareSys\HealthCareSys\"                              +
SessionData.HospitalSession);

MessageBox.Show(DocConvert.VerifySign(@"D:\HealthCareSys\HealthCareSys\"                +
SessionData.HospitalSession));
EncryptXML d = new EncryptXML();

MessageBox.Show(d.Encrypt(@"D:\HealthCareSys\HealthCareSys\"                            +
SessionData.HospitalSession, "p1").ToString());
}
else
{

System.IO.File.Delete(@"D:\HealthCareSys\HealthCareSys\" + SessionData.HospitalSession +
".xml");
List<DocType> Docs = new List<DocType>();
char[] ch = { ':' };
for (int i = 0; i < listBox1.Items.Count; i++)
{
```

```
richTextBox1.LoadFile(listBox1.Items[i].ToString(), RichTextBoxStreamType.PlainText);
DocType TempDoc = new DocType();
foreach (var line in richTextBox1.Lines)
{

if (line.ToLower().Contains("hospital"))
{
string[] lines = line.Split(ch);
TempDoc.HospitalName = lines[1].ToString();


}
else if (line.ToLower().Contains("patient"))
{
string[] lines = line.Split(ch);
TempDoc.PatientName = lines[1].ToString();


}
else if (line.ToLower().Contains("address"))
{
string[] lines = line.Split(ch);
TempDoc.Address = lines[1].ToString();


}
else if (line.ToLower().Contains("phone"))
{
string[] lines = line.Split(ch);
TempDoc.PhoneNO = lines[1].ToString();


}
else if (line.ToLower().Contains("date"))
{
string[] lines = line.Split(ch);
TempDoc.OnDate = lines[1].ToString();


}
else if (line.ToLower().Contains("symptom"))
{
string[] lines = line.Split(ch);
TempDoc.Symptoms = lines[1].ToString();


}
else if (line.ToLower().Contains("treatment"))
{
string[] lines = line.Split(ch);
TempDoc.Treatments = lines[1].ToString();


}


}
Docs.Add(TempDoc);
}

ConvertToXML DocConvert = new ConvertToXML();
XElement tempfile = DocConvert.ConvertText(Docs);
tempfile.Save(@"D:\HealthCareSys\HealthCareSys\" + SessionData.HospitalSession + ".xml");
DocConvert.SignDocument(@"D:\HealthCareSys\HealthCareSys\"                    +
SessionData.HospitalSession);

MessageBox.Show(DocConvert.VerifySign(@"D:\HealthCareSys\HealthCareSys\"      +
SessionData.HospitalSession));
EncryptXML d = new EncryptXML();
```

```
MessageBox.Show(d.Encrypt(@"D:\HealthCareSys\HealthCareSys\"                    +
SessionData.HospitalSession, "p1").ToString());

}
}
}
}
```

7.2 SCREEN SHOT

IMPLEMENTATION OF HEALTH CARE ADMIN APPLICATION

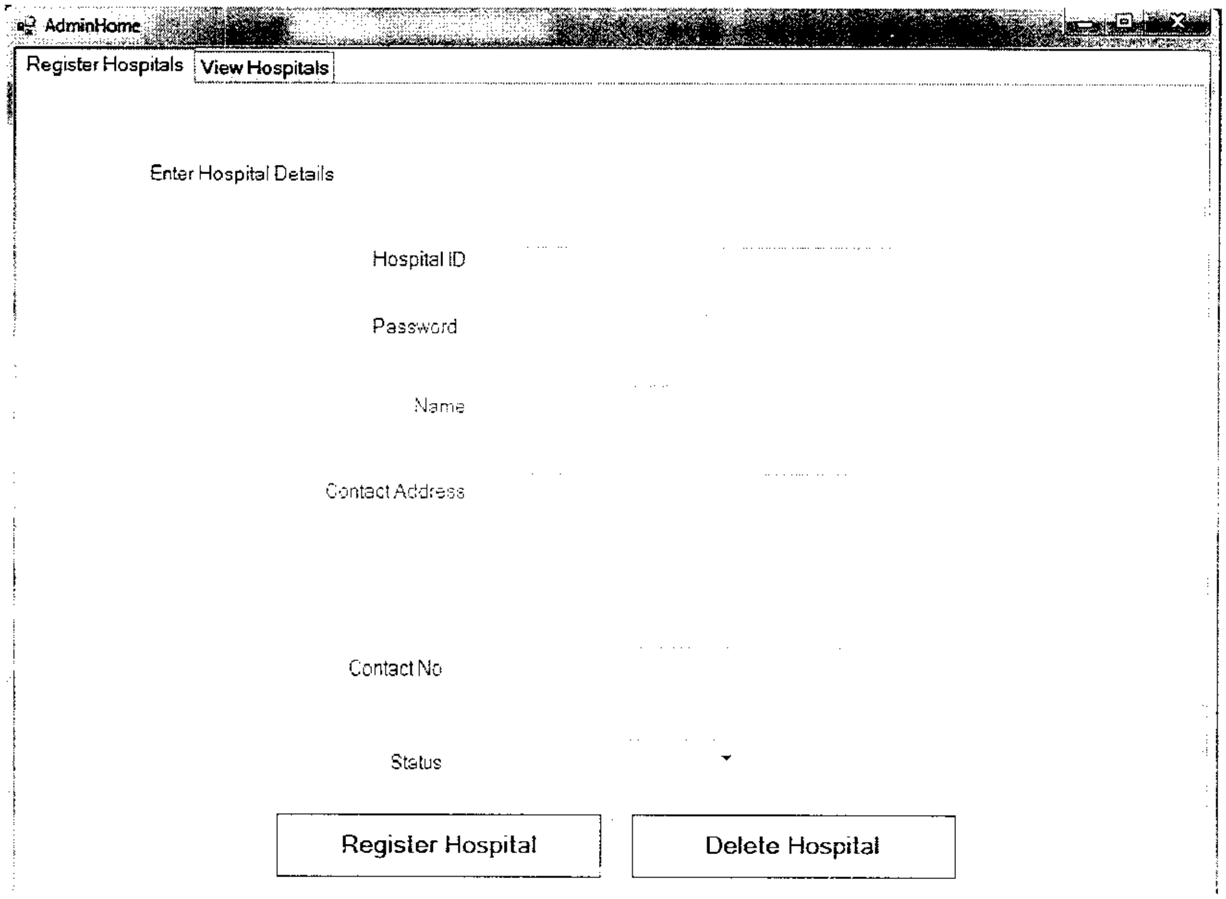

Health Care Admin Login

Welcome

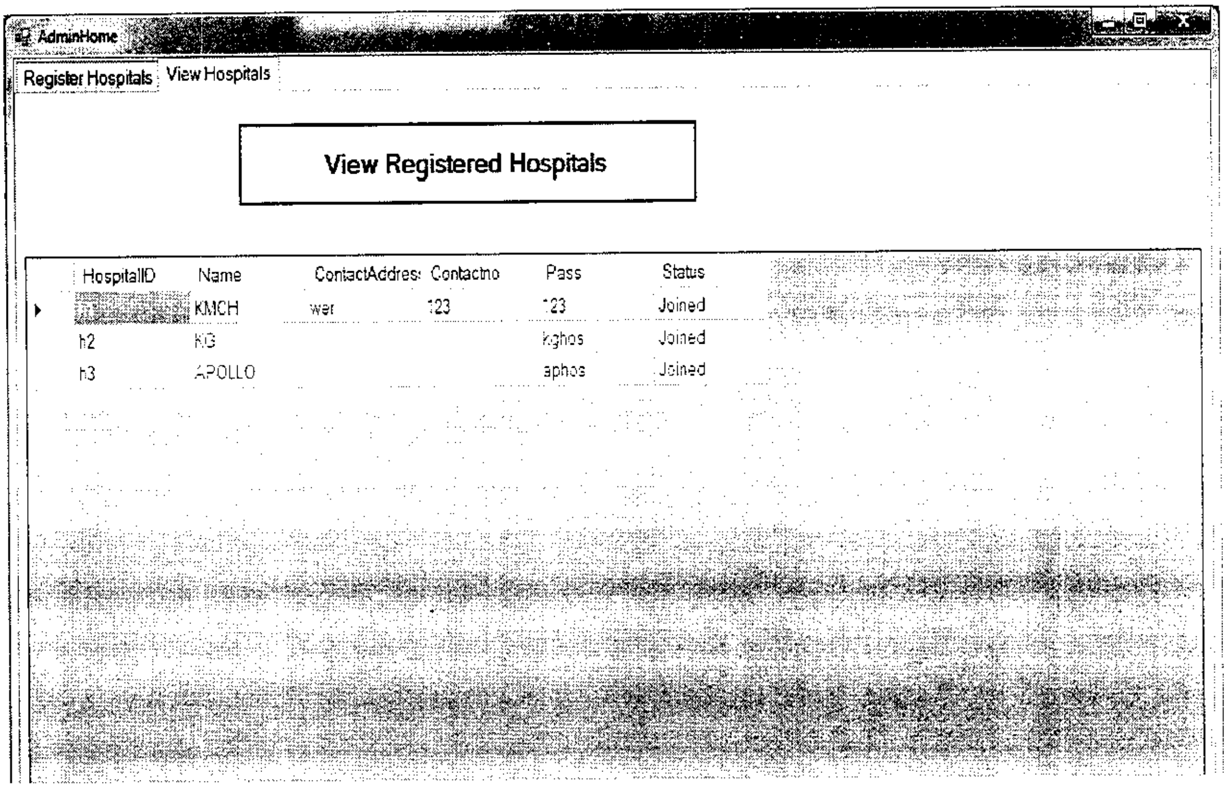
UserId admin

Password ****

HOSPITAL REGISTRATION:

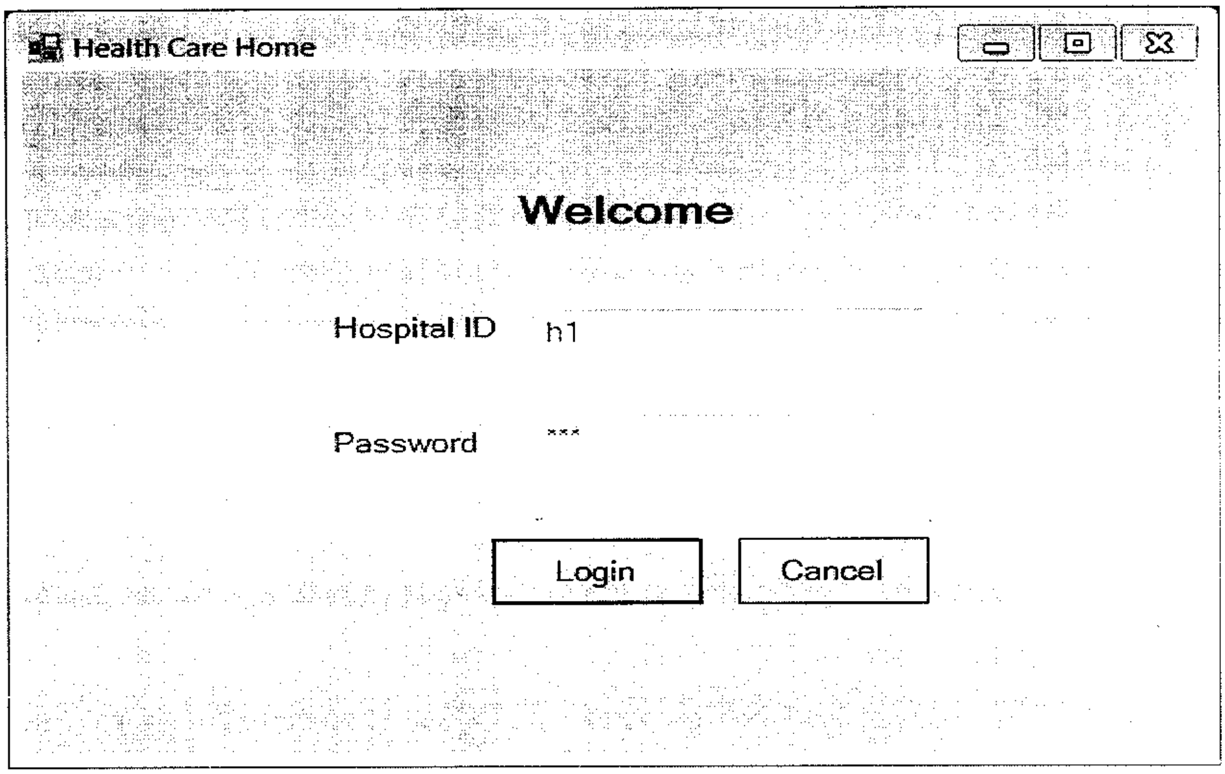


VIEW REGISTERED HOSPITALS:



HospitalID	Name	ContactAddres	Contactno	Pass	Status
[illegible]	KMCH	wer	123	123	Joined
h2	KG			kghos	Joined
h3	APOLLO			aphos	Joined

IMPLEMENTATION OF CLIENT APPLICATION:


Health Care Home

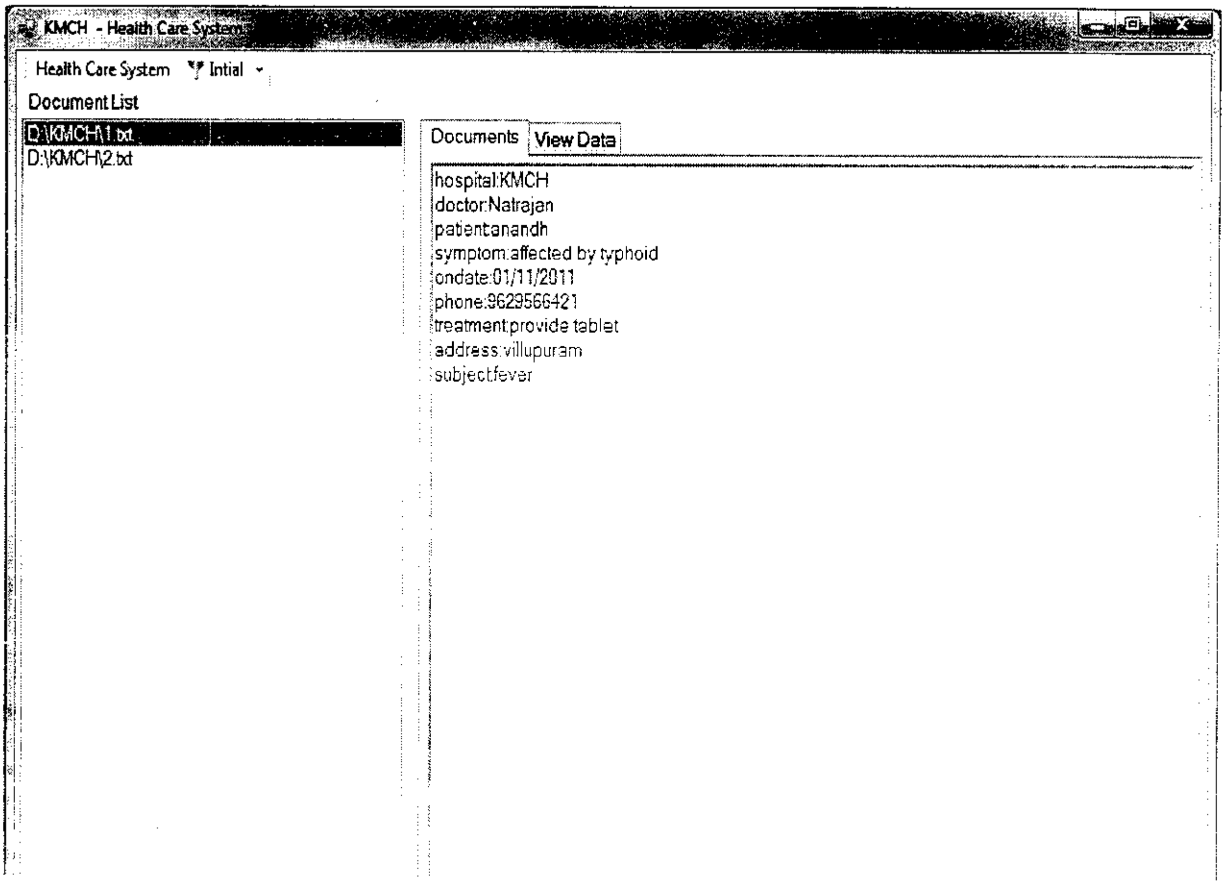
Welcome

Hospital ID h1

Password ***

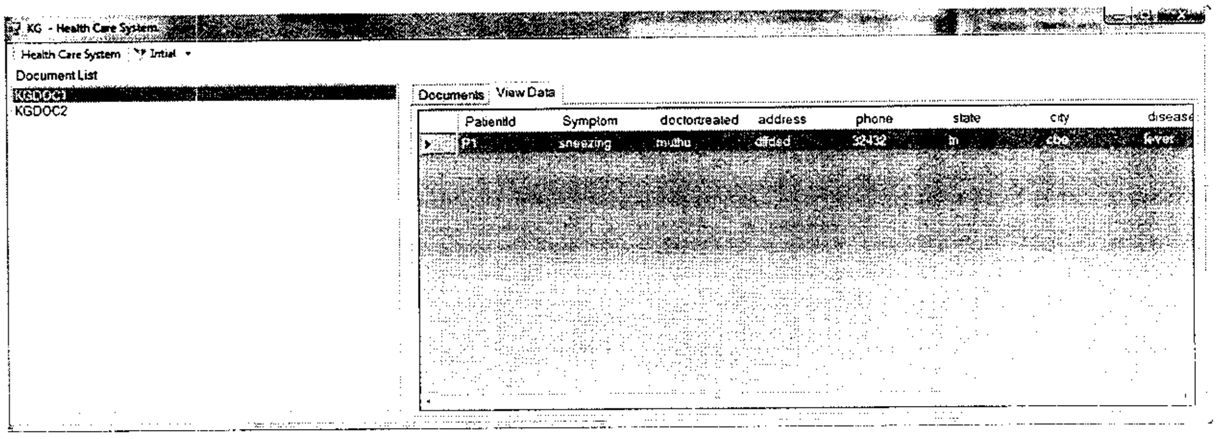
VIEWING AND UPLOADING DOCUMENTS:

INPUT: TEXT FORMAT



```
hospital:KMCH
doctor:Natrajan
patient:anandh
symptom:affected by typhoid
ondate:01/11/2011
phone:9629566421
treatment:provide tablet
address:villupuram
subject:fever
```

INPUT: SQL FORMAT

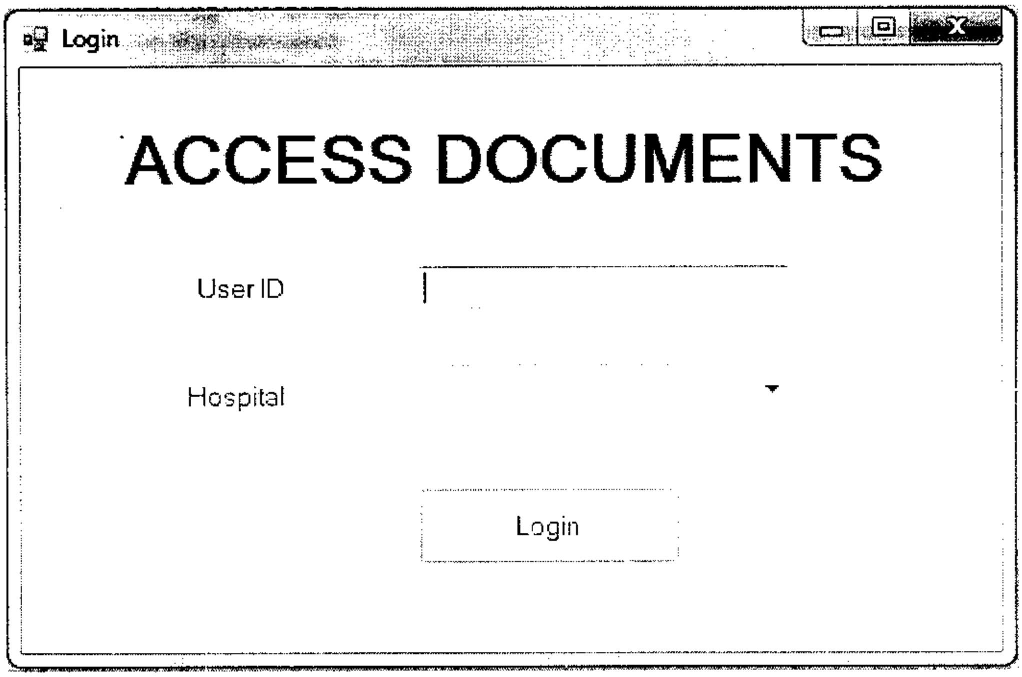


PatientId	Symptom	doctortreated	address	phone	state	city	disease
P1	sneezing	muthu	dfdsd	32432	tn	cbe	fever

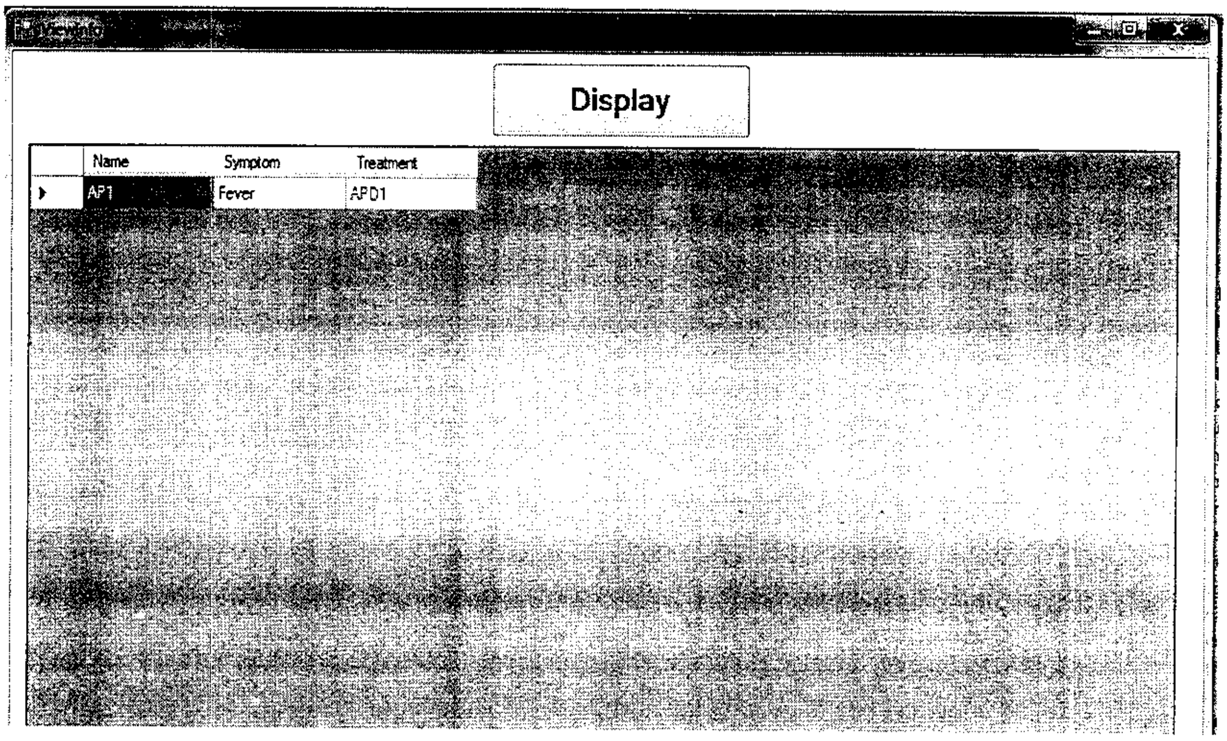
OUTPUT: XML FORMAT

```
<?xml version="1.0" encoding="utf-8"?>
<Reports>
  <Report AccessBy="public">
    <Hospital>KMCH</Hospital>
    <Patient>anandh</Patient>
    <OnDate>01/11/2011</OnDate>
    <Phone>9629566421</Phone>
    <Subject />
    <Symptom>affected by typhoid</Symptom>
    <Treatment>provide tablet</Treatment>
    <Address>villupuram</Address>
  </Report>
  <Report AccessBy="public">
    <Hospital>KMCH</Hospital>
    <Patient>sangeth</Patient>
    <OnDate>01/11/2011</OnDate>
    <Phone>9629566421</Phone>
    <Subject />
    <Symptom>affected by fever</Symptom>
    <Treatment>provide tablet</Treatment>
    <Address>salem</Address>
  </Report>
</Reports>
```

IMPLEMENTATION OF USER INFORMATION SYSTEM APPLICATION:



VIEWING PATIENT INFORMATIONS:



	Name	Symptom	Treatment
▶	AP1	Fever	APD1

8.REFERENCES

[1] S. Tzelepi, G. Pangalos, G. Nikolacopoulou, Security of medical multimedia, Med. Inform. Internet Med 73 (3) (2002) 169-184.

[2] B. Blobel, Authorisation and access control for electronic health record systems, int. J. Med. Inform. 73 (3) (2004) 251-257.

[3] R.E. Scott, P.Jennett, M. Yeo, Access and authorization in a Glocal e-Health Policy context, Int. J. Med. Inform. 73 (3) (2004) 259-266.

[4] V.N. Kallepalli, S.A. Ehikioya, S. Camorlinga, J.A. Rueda, Security middleware infrastructure for DICOM image in health information system, J. Digit. Imageing 16 (4) (2003) 356-364.

[5] D. Gritzalis, C. Lambrinoudakis, A security architecture for interconnecting health information systems, Int. J. Med. Inform. 73 (3) (2004) 305-309.

[6] D.F. Ferraiolo, D.R. Kuhn, R. Chandramouli, Role-Based Access Control, Artech House, 2003.

[7] G.K. Georgiadis, I.K. Mavridis, G. Nikolakopoulou, G.I. Pangalos, Implementing context and team based access control in healthcare intranets, Med, Inform. Internet Med. 27 (3) (2002) 185-201.

[8] J.Choe, SKYoo, Web-based secure access from multiple patient repositories, Int.J.Med.Inform, In Press, medinf.2007.06.001.

[9] D. Eastlake, J. Reagle, D. Solo, XML-Signature Syntax and Processing, W3C Recommendation, 12 February 2002, http://www.w3.org/TR/2002/REC -xmldsig-core-20020212, accessed 10 July 2004.

[10] D. Eastlake, J. Reagle, XML Encryption Syntax and Processing, W3C Recommendation, 10 December 2002, http://www.w3.org/TR/2002/REC -xmlenc-core-20021210, accessed 10 July 2004.

[11] B. Dournaee, XML Security, McGraw-Hill, 2002, pp. 107-278.

[12] S. Gritzalis, D. Gritzalis, C. Moulinos, J. Iliadis, An integrated architecture for deploying a virtual private medical network over the web, Med. Inform. Internet Med. 26 (1) (2001) 49-72.