SYNOPSIS

The project "**DIGITAL LOCKER**" is designed using Microsoft ASP.Net as front end and MS-SQL Server 2000 as back end. The coding language used is VB.Net. The .Net framework used is 2.0

This project is helpful for providing two step security process for storing important document, images, videos and audio files etc. Here the steps of verifications are user id and password, OTP and Image selection.

Here the user will register their personal details such as id, name, address, mail id, phone number etc and he will also upload an image which he wants to set as password. After registration the user login process starts.

After the proper login the user can upload their important document and files in a secured manner. The password of the user will be stored in database in encrypted format, so there will be no possibility for password malfunctioning.

1.3 SYSTEM ENVIRONMENT

1.3.1 HARDWARE SPECIFICATION

This section gives the details and specification of the hardware on which the system is expected to work,

Processor	: Pentium IV 1.7 GHz
Hard Disk Capacity	: 80 GB
RAM	: 1 GB
Monitor	: 15'' Color
Keyboard	: 102 keys
Mouse	: 3 buttons

1.3.2 SOFTWARE SPECIFICATION

This section gives the details of the software that are used for the development,

Environment	: Visual Studio .Net 2005
Front-End	: ASP.Net
Back-End	: MS SQL Server 2008
Operating System	: Windows 10 Pro
Browser	: Firefox

1.3.3MODULE DESCRIPTION

USER

REGISTRATION

Here user will upload his details such as ID, name, address, mobile no, gender, email id, image type, image and password. Here the mail id is very important because during login the OTP will be sent to the mail that has been given during registration

LOGIN LEVEL1

This is first level of login here user have to give their user id and password. Then the will be text box for OTP, when the user clicks send OTP button an OTP will be sent through mail. By using that OTP user can pass through the first level login.\

LOGIN LEVEL2

This is next level of login. During registration user will upload a image which has been used as password. In this login user will upload a image.

The compiler will check both the images (i.e., image uploaded during registration and uploaded during login) pixel by pixel and allows the user to next form.

ADD DOCUMENT

Here user will upload a document which has to be stored in secured manner. This form include field such as user id, user name, description and document. The uploaded document can be viewed and downloaded by user.

1.3.4SOFTWARE DESCRIPTION

THE .NET FRAMEWORK

The .NET Framework is a new computing platform that simplifies application development in the highly distributed environment of the Internet.

OBJECTIVES OF. NET FRAMEWORK:

1. To provide a consistent object-oriented programming environment whether object code is stored and executed locally of Internet-distributed or executed remotely.

2. To provide a code-execution environment to minimize software deployment and guarantee safe execution of code.

3. Eliminates the performance problems.

There are different types of applications, such as Windows-based applications and Web-based applications. To make communication on distributed environment to ensure that code be accessed by the .NET Framework can integrate with any other code.

VISUAL BASIC .NET

- Visual Basic. NET, the latest version of Visual Basic, includes many new features. The Visual Basic supports interfaces but not implementation inheritance.

- Visual basic.net supports implementation inheritance, interfaces and overloading. In addition, Visual Basic .NET supports multithreading concept.

IMPLEMENTATION INHERITANCE:

Visual C#.NET supports implementation inheritance. This means that, while creating applications in Visual C#.NET, we can derive from another class, which is known as the base class, that derived class inherits all the methods and properties of the base class. In the derived class, we can either use the existing code of the base class or

override the existing code. Therefore, with help of the implementation inheritance, code can be reused.

CONSTRUCTORS AND DESTRUCTORS:

Constructors are used to initialize objects, whereas destructors are used to destroy them. In other words, destructors are used to release the resources allocated to the object. In Visual C#.NET, the sub finalize procedure is available. The sub finalize procedure is used to complete the tasks that must be performed when an object is destroyed. The sub finalize procedure is called automatically when an object is destroyed. In addition, the sub finalize procedure can be called only from the class it belongs to or from derived classes.

OVERLOADING:

Overloading is another feature in Visual C#.NET. Overloading enables us to define multiple procedures with the same name, where each procedure has a different set of arguments. Besides using overloading for procedures, we can use it for constructors and properties in a class

MULTITHREADING:

Visual C#.NET also supports multithreading. An application that supports multithreading can handle multiple tasks simultaneously, we can use multithreading to decrease the time taken by an application to respond to user interaction. To decrease the time taken by an application to respond to user interaction, we must ensure that a separate thread in the application handles user interaction.

STRUCTURED EXCEPTION HANDLING:

Visual C#.NET supports structured handling, which enables us to detect and remove errors at runtime. In Visual C#.NET, we need to use try…catch…finally statements to create exception handlers. Using try…catch…finally statements, we can

create robust and effective exception handlers to improve the performance of our application.

ASP.NET

ASP.NET is a web application framework developed and marketed by Microsoft to allow programmers to build dynamic web sites, web applications and web services. It was first released in January 2002 with version 1.0 of the .NET Framework, and is the successor to Microsoft's Active Server Pages (ASP) technology. ASP.NET is built on the Common Language Runtime (CLR), allowing programmers to write ASP.NET code using any supported .NET language. The ASP.NET SOAP extension framework allows ASP.NET components to process SOAP messages.

CODE-BEHIND MODEL

Microsoft recommends dealing with dynamic program code by using the code-behind model, which places this code in a separate file or in a specially designated script tag. Code-behind files typically have names like MyPage.aspx.cs or MyPage.aspx.vb while the page file is MyPage.aspx (same filename as the page file (ASPX), but with the final extension denoting the page language). This practice is automatic in Microsoft Visual Studio and other IDEs. When using this style of programming, the developer writes code to respond to different events, like the page being loaded, or a control being clicked, rather than a procedural walk through the document.

ASP.NET's code-behind model marks a departure from Classic ASP in that it encourages developers to build applications with separation of presentation and content in mind. In theory, this would allow a web designer, for example, to focus on the design markup with less potential for disturbing the programming code that drives it. This is similar to the separation of the controller from the view in model-view-controller frameworks.

SQL SERVER 2008

SQL Server 2008 (codenamed Yukon), released in October 2000, is the successor to SQL. It included native support for managing XML data, in addition to relational data. For this purpose, it defined an xml data type that could be used either as a data type in database columns or as literals in queries. XML columns can be associated with XSD schemas; XML data being stored is verified against the schema. XML is converted to an internal binary data type before being stored in the database. Specialized indexing methods were made available for XML data. XML data is queried using XQuery; SQL Server 2005 added some extensions to the T-SQL language to allow embedding XQuery queries in T-SQL. In addition, it also defines a new extension to XQuery, called XML DML, that allows query-based modifications to XML data. SQL Server 2008 also allows a database server to be exposed over web services using TDS packets encapsulated within SOAP (protocol) requests. When the data is accessed over web services, results are returned as XML.

For relational data, T-SQL has been augmented with error handling features (try/catch) and support for recursive queries (Common Table Expressions). SQL Server 2005 has also been enhanced with new indexing algorithms and better error recovery systems. Data pages are check summed for better error resiliency, and optimistic concurrency support has been added for better performance. Permissions and access control have been made more granular and the query processor handles concurrent execution of queries in a more efficient way. Partitions on tables and indexes are supported natively, so scaling out a database onto a cluster is easier. SQL CLR was introduced with SQL Server 2005 to let it integrate with the .NET Framework.

SQL Server 2008 introduced "MARS" (Multiple Active Results Sets), a method of allowing usage of database connections for multiple purposes.

SERVICES

SQL Server also includes an assortment of add-on services. While these are not essential for the operation of the database system, they provide value added services on top of the core database management system. These services either run as a part of some SQL Server component or out-of-process as Windows Service and presents their own API to control and interact with them.

SERVICE BROKER

The Service Broker, which runs as a part of the database engine, provides a reliable messaging and message queuing platform for SQL Server applications. Used inside an instance, it is used to provide an asynchronous programming environment. For cross instance applications, Service Broker communicates over TCP/IP and allows the different components to be synchronized together, via exchange of messages

REPLICATION SERVICES

SQL Server Replication Services are used by SQL Server to replicate and synchronize database objects, either in entirety or a subset of the objects present, across replication agents, which might be other database servers across the network, or database caches on the client side. Replication follows a publisher/subscriber model, i.e., the changes are sent out by one database server ("publisher") and are received by others ("subscribers"). SQL Server supports three different types of replication.

- Maintaining relationships between the data in the database
- Ensuring that data is stored correctly, and that the rules defining data relationships are not violated.

Recovering all data to appoint of known consistency in case of system failures.

2. SYSTEM ANALYSIS

2.1 EXISTING SYSTEM

1. Security involving communications and networks is not as simple as it might first appear to the novice. The requirements for security services a can be given self-explanatory one word labels: confidentiality, authentication, no repudiation, integrity. But the mechanisms used to meet those requirements can be quite complex and understanding them may involve rather subtle reasoning.
2. In developing a particular security mechanism or algorithm one must always consider potential countermeasures. In many cases, countermeasures are designed by looking at the problem in a completely different way.
3. Because of point 2, the procedures used to provide particular services are often counterintuitive: It is not obvious form the statement of a particular requirement that such elaborate measures are needed. It is only when the various countermeasures are considered that the measures used make sense.
4. Having designed various security mechanisms, it is necessary to decide where to use them. This is true both in terms of physical placement and in a logical sense.
5. Security mechanisms usually involve more than a particular algorithm or protocol. They usually also require that participants be in possession of some secret information (e.g. an encryption key), which raises questions about the creation, distribution, and protection of that secret information. There is also a reliance on communications protocols whose behaviors may complicate the task of developing the security mechanism. For example, if the proper functioning of the security mechanism requires setting time limits of the transit time of a message form sender to receiver, they may protocol or networks security service and mechanism can be seemed.

2.2 DRAWBACKS OF EXISTING SYSTEM

No security measures are provided in the existing voting system and manual database is not more effective. User found some difficulties to store their document and files in a secured manner.

2.3 PROPOSED SYSTEM

The project is to implement the digital locker security system using image selection and OTP. In this, image authentication will be used for substantiation and to monitor the correct person. The images that has been selected by user during registration and login will be compared pixel by pixel to get correct output .

2.4. ADVANTAGES OF PROPOSED SYSTEM:

Image selection are processed through the image data comparison system. The images are submitted by uploading and converted as byte array which is checked with user image already present in the database.

1. The security is more in the proposed system.
2. Consolidated reports are viewed whenever required.
3. Easy to use options are provided in the web page and giving input is fast.
4. Increased security due to image selection and OTP.

3. SYSTEM DESIGN

The design of the system is essentially a blue print, or plan for a solution of the system to be developed. A part of the system or subsystem of a whole system can itself be considered a system with its own complements.

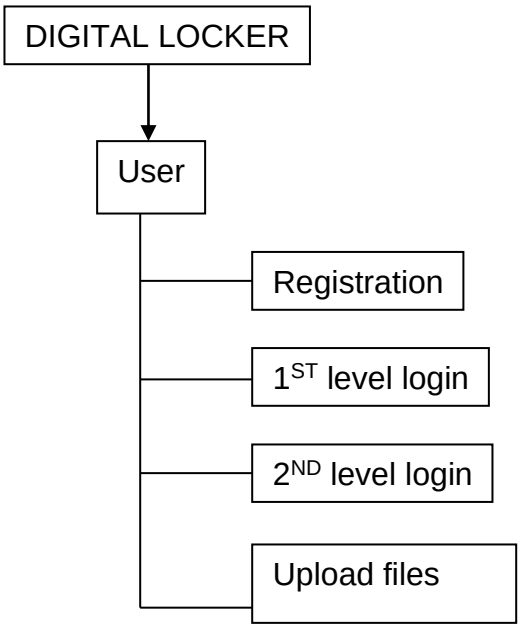
Here the focus is on detecting that which is all the modules method needed for the system, the specification of these design is the only way by which we can accurately translate the end-user requirements in to a finished software product or system.

The data flow oriented design in an architectural design method that allows a convenient transition from the analysis model to a design description of program structure the DFD presents a system overview depicting its overall purpose an its interactions with external objects it provides a general pictorial of data transformations in the system.

A DFD shows the flow of data through a system which may be a manual procedure software system, a hard ware system or any combination of these. A DFD shows the movements of the data through the different transformations, which are the process in the system. DFD are made up of a number of symbols, which represent system components, process, data store, and data flow and external entities.

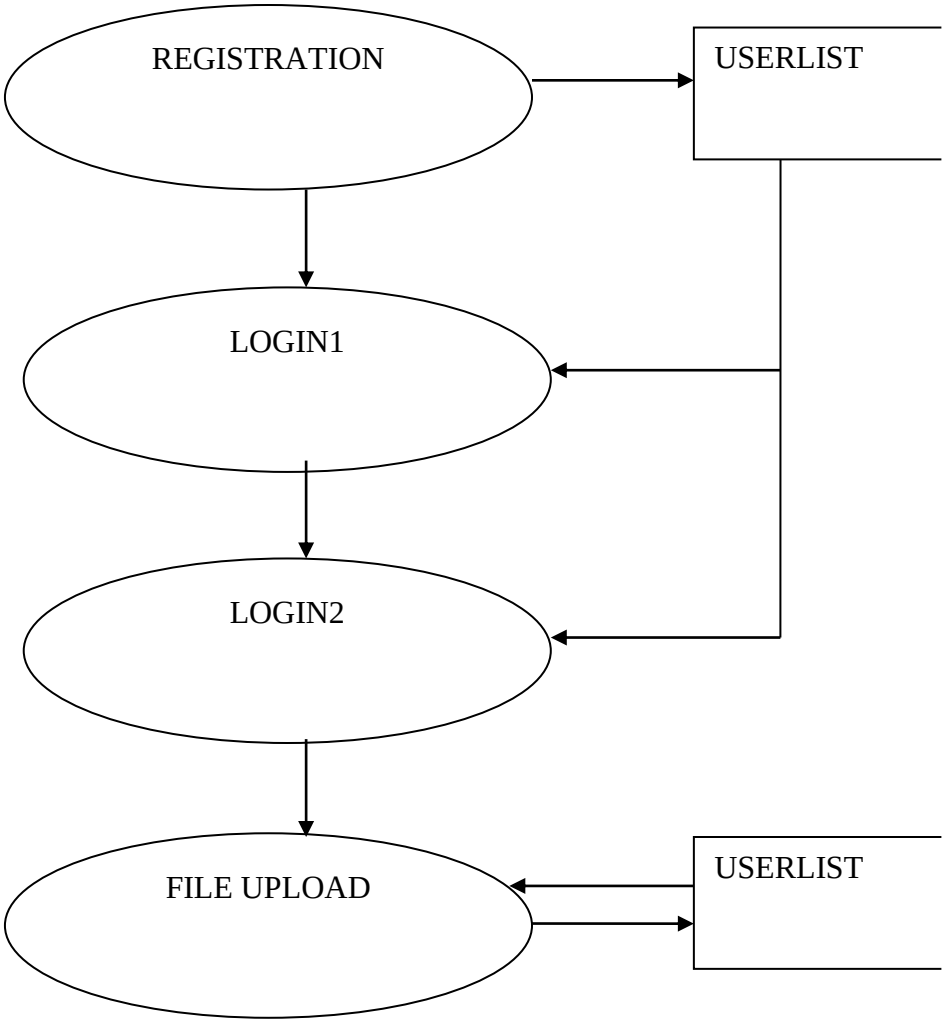
- The data store symbol represents the data that is not moving, that is the data at rest
- The process symbol represents an activity that transforms or manipulates the data
- The External entity symbol represents source of data to the system or destinations of data from system.
- The data flow symbol represents movement of the data.

3.1 SYSTEM FLOW DIAGRAM



3.2 DATA FLOW DIAGRAM

LEVEL - 1



3.3 DATABASE DESIGN

The most important consideration in designing the database is, how information will be used. The main objectives of designing a database are:

Data Integration:

In a database, information from several files are coordinated, accessed and operated upon as through it is in a single file. Logically, the information are centralized, physically, the data may be located on different devices, connected through data communication facilities.

Data Integrity:

Data integrity means storing all data in one place only and how each application to access it. This approach results in more consistent information, one update being sufficient to achieve a new record status for all applications. This leads to less data redundancy; data items need not be duplicated; a reduction in the direct access storage requirement.

Data Independence:

Data independence is the insulation of application programs from changing aspects of physical data organization. This objective seeks to allow changes in the content and organization of physical data without reprogramming of applications and to allow modifications of application programs without reorganizing the physical data.

The tables needed for each module were designed and the specification of each and every column based on the records is provided and details collected during record specification of the system study.

The major objectives for maintaining such a database are:

- The database must reduce redundancy
- Enforce standards
- Share data and maintain integrity
- The data must be independent and consistent
- We must be able to apply security restrictions

TABLE STRUCTURE

Table : USER REGISTRATION

FIELD NAME	TYPE	SIZE	DESCRIPTION
UserId	Int	4	PRIMARY KEY
UserName	Varchar	20	NOT NULL
Address	Varchar	50	NOT NULL
Mobile	Varchar	10	NOT NULL
Gender	Varchar	6	NOT NULL
Emailid	Varchar	30	NOT NULL
ImageType	Varchar	30	NOT NULL
Image	Varchar	50	NOT NULL
Password	Varchar	20	NOT NULL

Table : Logintable1

FIELD NAME	TYPE	SIZE	DESCRIPTION
UserName	Varchar	15	FOREIGN KEY
Password	Varchar	15	NOT NULL
OTP	Varchar	15	NOT NULL

Table : LoginTable2

FIELD NAME	TYPE	SIZE	DESCRIPTION
UserID	Int	4	FOREIGN KEY
Image	Varchar	50	NOT NULL

Table : DocumentUpload

FIELD NAME	TYPE	SIZE	DESCRIPTION
UserID	Int	4	FOREIGN KEY
UserName	Varchar	30	NOT NULL
Description	Varchar	50	NOT NULL
Document	Varchar	30	NOT NULL

3.4 INPUT DESIGN

Input Design is one of the most expensive phases of the operation of computerized system and it is often the major problem of a system. A large number of problems with a system can usually be tracked backs to fault input design and method. Needless to say, therefore, that the input data is the life blood of a system and have to be analyzed and designed with utmost case and consideration. The decisions made during the input design are,

- To provide cost effective method of input.
- To achieve the highest possible level of accuracy.
- To ensure that the input is understand by the user.

System analysis decide the following input design details like, what data to be as input, what medium to use, how the data should be arranged or coded, data items and transactions needing validations to detect errors and at last the dialogue to guide user in providing input.

Input data of a system may not be necessarily be raw data captured in the system from scratch. These can also be the output of another system or sub system. The design of input covers all phases of input from the creation of initial data to actual entering the data to the system for processing. The design of inputs involves identifying the data needed, specifying the characteristics of each data item, capturing and preparing data for computer processing and ensuring correctness of data.

3.5 OUTPUT DESIGN

Output Design generally refers to the results and information's that are generated by the system for many end-users, output is the main reason for developing the system and the basis on which they evaluate the usefulness of the application.

The objective of a system finds its shape in terms of the output. The analysis of the objective of a system leads to determination of outputs. Outputs of a system can face various forms. The most common are reports, screen displays, printed forms, graphical drawings etc.,

The output can also be varied in terms of their contents frequency, timing and format. The user of the output from a system are the justification for its existence. If the output are inadequate in any way, the system are itself is adequate. The basic requirements of output are that it should be accurate, timely and appropriate, in terms of content, medium and layout for its intended purpose.

4. SYSTEM TESTING AND IMPLEMENTATION

4.1 SYSTEM TESTING

After the source code has been completed, documented as related data structures. Completion of the project has to undergo testing and validation where there is subtitle and definite attempt to get errors. The project developer treats lightly, designing and execution of the project test that will demonstrates that the program works rather than uncovering errors, unfortunately errors will be present and if the project developer doesn't found errors, the user will find out.

The project developer is always responsible for testing the individual units i.e. modules of the program. In many cases, developer should conduct the integration testing i.e. the testing step that leads to the construction of the complete program structure.

This project has undergone the following testing procedures to ensure its correctness.

- **Unit testing**
- **Integration testing**
- **Validation testing**

UNIT TESTING

In unit testing, user has to test the programs making up the system. For this reason, Unit testing sometimes called as Program testing. The software units in a system are the modules and routines that are assembled and integrated to perform a specific function.

In this project, user details, login process could be tested individually like given all the fields and can be updated for all criteria.

INTEGRATION TESTING

Integration testing is a systematic technique for constructing the program structure. While at the same time conducting tests to uncover errors associated with interfacing. The objective is to take unit tested modules and build a program structure that has been dictated by design.

In this integration testing its done using the main module and based on the type of integration testing the subordinate tables and other criteria along with their path, is replaced one at a time with actual modules. In the project, after integrating the all modules, they are tested with their integration and that could integrated and manipulated with respect to the to and fro in between modules.

VALIDATION TESTING

It is said that validation is successful when the software functions in a systematic manner that can be reasonably accepted by the customers. This type of testing is very important because it is the only way to check whether the requirements given by user have been completely fulfilled.

The input given to various forms are validated effectively. The input is given for user registration, login process, document addition. Each module is tested independently. In this project, username and password is validated successfully before login. Iris print image is uploaded and validated through byte array comparison with database data for login verification.

4.2 SYSTEM IMPLEMENTATION

When the initial design was done for the system, the client was consulted for the acceptance of the design so that further proceedings of the system development can be carried on the project. After the development of the system a demonstration was given to them about the working system. The aim of the system illustration was to identify any malfunction of the system.

After the management of the system was approved the system implemented in the concern, initially the system was run parallel with existing manual system. The system has been tested with live data and has proved to be error free and user friendly.

Implementation is the process of converting a new or revised system design into an operational one when the initial design was done by the system; a demonstration was given to the end user about the working system. This process is uses to verify and identify any logical mess working of the system by feeding various combinations of test data. After the approval of the system by both end user and administrator the system was implemented.

5. CONCLUSION

The new system eliminates the difficulties in the existing system. It is developed in a user-friendly manner. The system is very fast and any transaction can be viewed or retaken at any level. Error messages are given at each level of input of individual stages. This software is very particular in reducing the work and achieving the accuracy. It will reduce time be avoids redundancy of data.

The user can easily understand the details available from the report. This software will support for the future development.

The software is menu driven.

- Very large date can be stored and also can be retrieved very easily.
- Data is entered in formatted manner.
- The report can be taken in any format.
- Modification and maintenance can be made very easily

6. SCOPE FOR FUTURE ENHANCEMENT

The system is very flexible and user-friendly, so the maintenance based on the changing environment and requirements can be incorporated easily. Any changes that are likely to cause failures are prevented with security and preventive measures could be taken. The coding is done in understandable and flexible method program which helps easy changing.

Since MS-SQL Server and C#.NET are very flexible tools. User can easily incorporate any modular program in the application.

- It facilitates the user to easily save by uploading the OTP and image password.
- Facilities fast data backup and restoration facility in case of data loss situations.
- If the modules are written as web services, then it can be used other web sites also.

7. BIBLIOGRAPHY

BOOKS

- Benolt Marchal, VB.NET by example 2003 – TataMcGraw- Hill.
- Grey Buczek, .NET developers guide 2002, Prentice-Hall India.
- Ingo Rammer and Mario Szpuszta, Advanced .NET Remoting 2nd Edition (March 2005)
- Matthew mac donald,Microsoft, Visual Basic.NET Programmer's Cookbook:-,Tata McGraw-Hill Edition
- Marchu, "ASP.NET With .Net 2.0 " Orelliy March 2005
- Pankaj Jalole, An Integral approach to software engineering,Nilesh 2nd edition

WEBSITES

http://www.dotnetspider.com

http://www.aspnetbegineers.com

http://www.quickstart.com

USER REGISTRATION

```
using System;
using System.Drawing;
using System.Data;
using System.Data.SqlClient;
using System.Configuration;
using System.Collections;
using System.Web;
using System.Web.Security;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Web.UI.WebControls.WebParts;
using System.Web.UI.HtmlControls;

public partial class adduser : System.Web.UI.Page
{
   protected void Page_Load(object sender, EventArgs e)
   {
      if ((Page.IsPostBack == false))
      {
         if (cls.con.State != ConnectionState.Open)
         {
            cls.con.Open();
         }
         cls.cmd.CommandText = "select coalesce(max(userid),0) from  userslist";
         if (cls.cmd.ExecuteScalar() == null)
         {
            TextBox1.Text = "1";
         }
         else
         {
            int sno = int.Parse(cls.cmd.ExecuteScalar().ToString()) + 1;
            TextBox1.Text = sno.ToString();
            TextBox1.Enabled = false;
         }} }
   protected void Button1_Click(object sender, EventArgs e)
   {
      try
      {
         if (cls.con.State != ConnectionState.Open)
         {
            cls.con.Open();
         }
```

```
        string fname = "";
          bool saved=false;
          string UserrID = "";
          string UserrName = "";
          string Address = "";
          string Phone = "";
          string Gender = "";

          string Password = "";


          string emailid = "";
          string imagetype = DropDownList2.SelectedItem.Text;
          UserrID = TextBox1.Text;
          UserrName = TextBox2.Text;
          Address = TextBox3.Text;
          Phone = TextBox4.Text;
          Gender = DropDownList1.SelectedItem.Text;
          Password = TextBox5.Text;
          emailid = TextBox12.Text;

         EncryptMessage(ref Password, TextBox5.Text);


   cls.cmd.CommandText = "Select count(*) from UsersList Where UserName='" +
UserrName.Replace("'","''") + "'";
   int cnt = int.Parse (cls.cmd.ExecuteScalar().ToString());
   if (cnt > 0)
   {
      lblMessage.Text = "User Name already found";
      return;
   }
   cls.cmd.CommandText = "insert into UsersList([UserID], [UserName], [Address],
[Phone], [Gender], [Password], [EmailId], [ImageType],FilePath) values('" +
UserrID.Replace("'", "''") + "','" + UserrName.Replace("'", "''") + "','" +
Address.Replace("'", "''") + "','" + Phone.Replace("'", "''") + "','" + Gender.Replace("'",
"''") + "','" + Password.Replace("'", "''") + "','" + emailid.Replace("'", "''") + "','" +
imagetype.Replace("'", "''") + "','" + UserrID + ".jpg')";
          cls.cmd.ExecuteNonQuery();

          if (FileUpload2.PostedFile != null)
          {
             string[] fn = FileUpload2.PostedFile.FileName.Split(new char[] { '\\' });
```

```
            fname = fn[fn.Length - 1];
            FileUpload2.PostedFile.SaveAs((Server.MapPath(".") +
("\\FaceFingerPrintImages\\" + UserrID.ToString() + ".jpg")));

            saved = true;
    }

          if ((saved == true))
          {

            try
            {
               clssendmail.MailSend(TextBox12.Text, "Confirmation", "You are added in
user list with id : " + TextBox1.Text);
            }
            catch (Exception ex)
            {
               lblMessage.Text = ex.Message;
               return;
            }
            lblMessage.Text = "User Details Saved";
          }
          else
          {
            lblMessage.Text = "User Details Not Saved";
          }
       }
       catch (Exception ex)
       {
          lblMessage.Text = ex.Message;
       }
       finally
       {
          cls.con.Close();
       }
       SqlDataSource1.SelectCommand = "SELECT [UserID], [UserName], [Address],
[Phone], [Gender], [Password], [EmailId], [ImageType] FROM [UsersList] Where
UserId='" + TextBox1.Text+ "'";
       SqlDataSource1.DataBind();
       GridView1.DataBind();
       //  GridView2.DataBind();
    }
    public void EncryptMessage(ref string encaccno, string accno)    {
```

```
        clsTripleDES des = new clsTripleDES();
        encaccno = des.Encrypt(accno);

    }
    public string DecryptMessage(string encaccno)
{
      string accno = "";
      clsTripleDES des = new clsTripleDES();
       accno = des.Decrypt(encaccno);
      return accno;
    }
    int i;

    protected void GridView1_RowDataBound(object sender, GridViewRowEventArgs e)
    {
       try
       {

          if (e.Row.RowType == DataControlRowType.DataRow)
          {

                if (e.Row.Cells[7].Text != "")
                {
                   e.Row.Cells[7].Text ="<a href='FaceFingerPrintImages/"
+e.Row.Cells[7].Text + "'><img src='FaceFingerPrintImages/" + e.Row.Cells[7].Text + "'
width='100' height='100'/></a>";
                }

          }
       }
       catch (Exception ex)
       {
       }
    }
    protected void GridView2_RowDataBound(object sender, GridViewRowEventArgs e)
    {

    }
}
```

ADD DOCUMENT

```
using System;
using System.Drawing;
using System.Data;
using System.Data.SqlClient;
using System.Configuration;
using System.Collections;
using System.Web;
using System.Web.Security;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Web.UI.WebControls.WebParts;
using System.Web.UI.HtmlControls;

public partial class adddocument : System.Web.UI.Page
{
   protected void Page_Load(object sender, EventArgs e)
   {
      if ((Page.IsPostBack == false))
      {
         if (cls.con.State != ConnectionState.Open)
         {
            cls.con.Open();
         }

         TextBox1.Text = Session["un"].ToString();
         TextBox2.Text = Session["username"].ToString();
            TextBox1.Enabled = false;
            TextBox2.Enabled = false;
            SqlDataSource1.SelectCommand = "SELECT [SNo],EntryDate,UserId,
[UserName], [Title], [FilePath] FROM [Documents] Where UserId='" + TextBox1.Text +
"'";
            SqlDataSource1.DataBind();
            GridView1.DataBind();


      }
   }
   protected void Button1_Click(object sender, EventArgs e)
   {
      try
      {
         if (cls.con.State != ConnectionState.Open)
         {
```

```
            cls.con.Open();
        }


        string fname = "";
        bool saved = false;
        string UserrID = "";
        string UserrName = "";
        string title = "";
        UserrID = TextBox1.Text;
        UserrName = TextBox2.Text;
        title = TextBox3.Text;
        cls.cmd.CommandText = "Select count(*) from Documents";
        int cnt = int.Parse(cls.cmd.ExecuteScalar().ToString())+1;
string ext="";
        if (FileUpload2.PostedFile != null)
        {
            string[] fn = FileUpload2.PostedFile.FileName.Split(new char[] { '\\' });
            fname = fn[fn.Length - 1];
            FileUpload2.PostedFile.SaveAs((Server.MapPath(".") + ("\\DocumentImages\\"
+ fname)));
            System.IO.FileInfo f = new System.IO.FileInfo(Server.MapPath(".") +
("\\DocumentImages\\" + fname));

       ext=     f.Extension;
            f.MoveTo (Server.MapPath(".") + "\\DocumentImages\\" + cnt.ToString() + ext);
            saved = true;
        }


        cls.cmd.CommandText = "insert into Documents ([SNo],
[EntryDate],UserId,UserName, [Title], [FilePath]) values('" + cnt.ToString() + "','" +
string.Format("{0:MM-dd-yyyy hh:mm:ss tt}",DateTime.Now) + "','" +
UserrID.Replace("'", "''") + "','" + UserrName.Replace("'", "''") + "','" + title.Replace("'",
"''") + "','" + cnt + "" + ext + "')";
        cls.cmd.ExecuteNonQuery();



        if ((saved == true))
        {
            lblMessage.Text = "User Details Saved";
        }
        else
```

```
        {
            lblMessage.Text = "User Details Not Saved";
        }
    }
    catch (Exception ex)
    {
        lblMessage.Text = ex.Message;
    }
    finally
    {
        cls.con.Close();
    }
    SqlDataSource1.SelectCommand = "SELECT [SNo],EntryDate,UserId, [UserName],
[Title], [FilePath] FROM [Documents] Where UserId='" + TextBox1.Text + "'";
    SqlDataSource1.DataBind();
    GridView1.DataBind();
    //  GridView2.DataBind();
  }

  int i;

  protected void GridView1_RowDataBound(object sender, GridViewRowEventArgs e)
  {
    try
    {

      if (e.Row.RowType == DataControlRowType.DataRow)
      {

        if (e.Row.Cells[5].Text != "")
        {
          e.Row.Cells[5].Text = "<a href='DocumentImages/" + e.Row.Cells[5].Text +
"'><img src='DocumentImages/" + e.Row.Cells[5].Text + "' width='100'
height='100'/></a>";
        }
}   }
    catch (Exception ex)
    {
    }  }
  protected void GridView2_RowDataBound(object sender, GridViewRowEventArgs e)
  {

  }
}
```