Fall 11-1-2007

Securing Data through Encryption

Ravi Seshadri
Dakota State University

Securing Data through Encryption

(INFS 788 - Planning)

A graduate project submitted to Dakota State University in partial fulfillment of the

Requirements for the degree of

Master of Science

In

Information Systems

November, 2007

By

Ravi Seshadri

Project Committee:

Committee Chair: Dr. Sreekanth Malladi

Committee member: Dr. Xinwen Fu

Committee member: Dr. Surendra Sarnikar

DAKOTA STATE UNIVERSITY

PROJECT APPROVAL FORM

We certify that we have read this project and that, in our opinion, it is satisfactory in scope and quality as a project for the degree of Master of Science in Information Systems.

Student Name: Ravi Seshadri

Master's Project Title: Securing Data Through Encryption

Faculty supervisor: [illegible] Date: 01 Dec 2007

Committee member: [illegible] Date: 12/3/2007

Committee member: Surendra Sarnikar Date: 12/3/2007

ACKNOWLEDGMENT

I would like to thank Dr. Sreekanth Malladi for being my project supervisor and providing me ideas and valuable suggestions, without which this project would not been implemented. I would like to thank for Dr. Xinwen Fu and Dr. Surendra Sarnikar for being committee members for this project. I would like to express my gratitude to Dr. Omar El-Gayar for guiding me in right direction which was very helpful in designing this project. Finally, I would like to thank my wife and children for being patient and encouraging me to complete my Masters in Information Systems.

DECLARATION

I hereby certify that this project constitutes my own product, that where the language of others is set forth, quotation marks so indicate, and that appropriate credit is given where I have used the language, ideas, expressions or writings of another.

I declare that the project describes original work that has not previously been presented for the award of any other degree of any institution.

Signed,

Ravi Seshadri

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

Background of the Problem

The regulations and legislation for organizations are demanding strong data privacy and make the data secure. The organizations especially in banking and finance industry are under pressure to meet these standards. The government is trying to enact new tough privacy legislation to protect the consumer's data. The legislation that is currently being proposed in California, for example, describes not only who and under what conditions confidential information can be released but also how that information should be stored on the servers.

Statement of the problem

Data encryption must be done by all the organizations that maintain the secure data. Data encryption will encode the data so that it is un-readable for un-authorized persons and can be read only by intended recipient. Organizations should enforce strict policy on their employees to encrypt the data prior sending them to the recipients. The data maintained in the database must also be encrypted and should be viewed only by authorized personals. To address this issue, they must have an application that encrypts and decrypts the data using various algorithms that are available. The application must be user friendly, affordable and should support multiple users. The application should be used to encrypt and decrypt the data that is received, stored and transmitted. Health Insurance

Portability and Accountability (HIPPA) and Gramm-Leach-Bliley Act (GLBA) regulations impose standards for the privacy and protection of all electronic information.

Objectives of the project

The objective of this project is to create a windows based application that encrypts and decrypts a given string or a file. This application can be used to encrypt the sensitive corporate data that can be stored in the servers or sent by e-mail. The recipient will use the same application to decrypt the data that will be in a readable format. The intended recipient can decrypt the encrypted data by providing a pass phrase and a string value that was used by the encrypted person or program.

This application can be installed as a stand alone program in a desktop or a notebook computer. The presentation tier and business logic will be developed as a separate module. This will enable the users to use the business logic in their own application by including the reference in their solution. The application can be used to encrypt and decrypt simple text and files. Users will be provided with an option to encrypt or decrypt text or file. If the user selects to encrypt text, he/she has to enter the text that has to be encrypted, select the encrypting algorithm and provide password. The application will encrypt the data using the algorithm the user has selected and the password the user has entered. The user can now copy the encrypted string and send it to the recipient and let him/her know the encryption algorithm and the password. The recipient uses the same application to decrypt the data and provides the algorithm and password that he/she received from the sender. The above logic will be implemented to encrypt and decrypt file also. In this case, user will be prompted to select a file that is present in his local machine and prompted to enter the name of the destination. The application will not allow encrypting the file if name of the source file and name of the destination file are same.

Project Scope

Securing Data through Encryption is windows based application and uses client server technology. The application is designed and developed using Microsoft .Net Framework and the programming language used is C#. Microsoft SQL Server 2005 is used in backend to store the data. The database is designed in relational model and it is normalized. The application comes along with an installer program and it can be installed in any machine that has Microsoft Windows XP and Microsoft .Net framework 2.0. The SQL Server 2005 can be installed in the client's machine or in another machine that is in the same network where the client machine exists. The application connects to the database using ODBC.

CHAPTER 2

LITERATURE REVIEW

When the usage of internet started growing rapidly, allowing people to reach hundreds of millions of servers around the world, it also gave hackers to interrupt the information being passed from one person to another and steal the data. Even though software is present to secure the transmission of data, the best way to maintain the security is to transmit the data in encrypted format. There are two levels of control to secure the data. First the data should be encrypted when it goes outside the organization. Second level is there should be systems present to ensure employees abide by the policies of organization. The companies are increasingly adopting encryption method to store and transmit data to avoid loss of sensitive information and other important information stored and transmitted. According to latest internet security threat report from Symantec, firms are not doing enough to protect against data losses. "The study found that 54 percent of all data breaches that could facilitate identity theft were the result of the loss or theft of computers or data storage devices, while 28 percent could be put down to failures to implement security policies." (IT Week, 2007, p. 1). "Companies choosing not to protect their data in spite of the recent prevalence of data theft will find it almost impossible to defend their position to the public," said Stonewood IT security advisor Andrew Donaghue. "Many Asian, European, and North American governments mandate that companies in various fields—such as healthcare, finance, and education protect data." (Computer, 2007, p.13-15)

CHAPTER 3

SYSTEM DESIGN (RESEARCH METHODOLOGY)

“Today's presenters find themselves with far more options for interfacing with audiences, whether it be face to face or across time zones. Whether you succeed in your meeting, training or sales mission increasingly depends on how well you understand the best uses, limitations and ever-evolving features of these burgeoning presentation tools” (Dave Zielinski, 2002, pg. S1)

The application is designed in a very user-friendly interface, so that even a common person will be able to understand the menus and options that are available in the system. “People say, by the way, it's very, very complicated to become very granular, that out of the 150 resources within a corporation, I can only have access to ten of 'em. That's difficult to do. Well, it may have been at one point, but I think we're getting much, much better about more granular authorization within an IT system.” (Howard Schmidt, 2005, pg.10)

The application has given strong importance for authentication module, where various roles are defined in the application. The application allows the administrator to create users with different roles so that they will be able to perform certain operations only.

System Requirements

Securing Data through encryption is a windows application built using client server technology. The user interface is a windows form designed and programmed using C# and the backend is SQL Server 2005. The System requirements for developer are as follows:

- Windows XP Professional
- Microsoft .Net Framework 2.0
- Microsoft Visual Studio 2005
- SQL Server 2005

. The System requirements for installation that will be used by end user are as follows:

- Windows XP Home or Professional
- Microsoft .Net Framework 2.0

If the users are in a network, then they can be connected to a Server that has SQL Server 2005. If the users are present in different locations and are not in network, then SQL Server has to be installed in their local machine or another machine that is in the same network.

Methodology

Agile software development is used to develop this project because of the flexibility. Each feature can be developed independently and will be merged together at the end and tested.

Table 1: Access levels of users

Role	Permissions
Admin	Add, Edit & Delete Users. Encrypt & Decrypt Data
Manager	Encrypt & Decrypt Data
Encrypt	Encrypt Data
Decrypt	Decrypt Data

The application is designed to create users with various levels of permissions to access the application. The admin has all rights and can create, modify and remove the users present in the system. Apart from this, the admin can perform the encryption and decryption also. The Manager is one level below the Admin and he can encrypt or decrypt the data. If there is a need to give single permission, either to encrypt or decrypt the data, then User – E or User - D can be created.

Table 2: Application Modules

Module	Description
Login Form	To Log on to the application
Main Menu Form	Allows user to choose the action he/she wish to perform
Manage Users Form	Allows admin to view the users and select a user
Update User Form	Allows admin to add, edit or delete user
Encrypt String form	Allows users to Encrypt a given string
Decrypt String form	Allows users to Decrypt a given string
Encrypt File form	Allows users to Encrypt a file
Decrypt File form	Allows users to Decrypt a file

Login Form:

This is the starting point of the application. This form will display login form to the users, which has a textbox for entering the user name and a textbox to enter the password. When the user enters the information, it will be passed to a store procedure present in the database which validates against the data present in the system. If the entered information is valid, then it returns zero for Success, and if the entered information is not valid then it returns one for failure. The logon information along with date and time is logged into a table. If there is a need to see this, a report can be generated out of it.

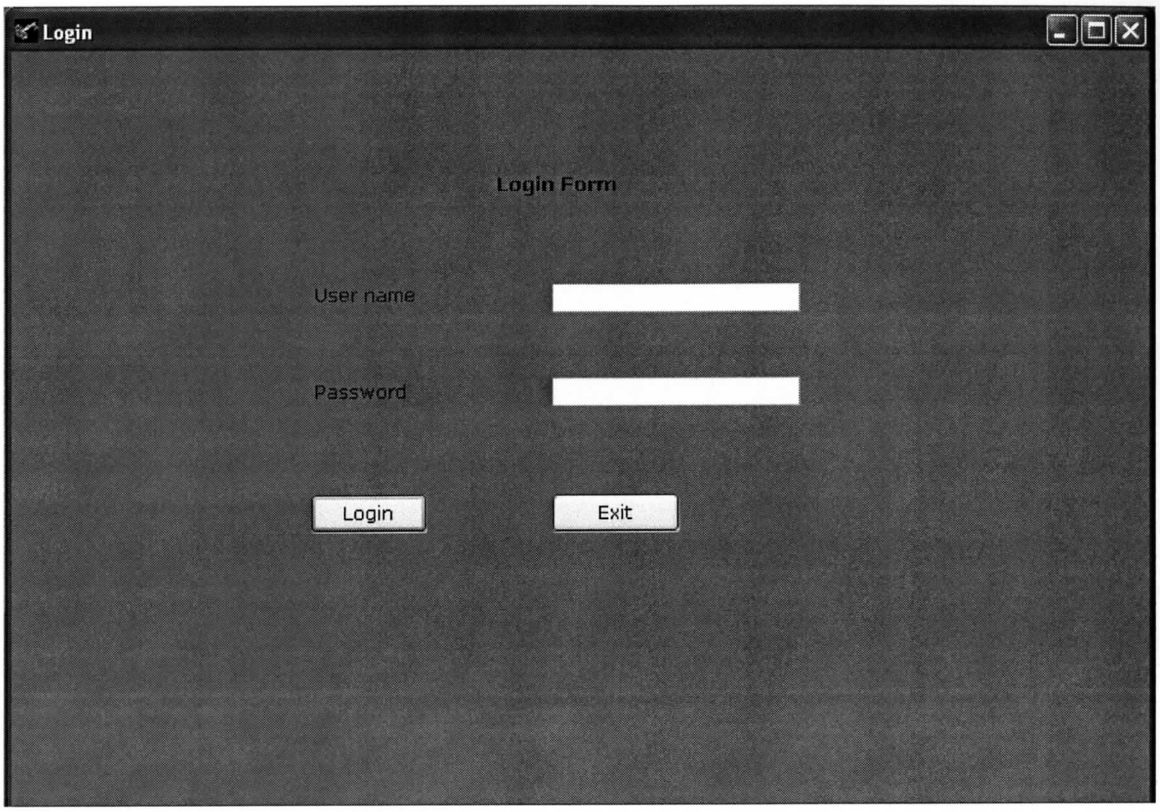


Fig 2.1 Login Form

Main Menu Form

This form displays the various options to choose for the user. If the logged on user is having administrative privileges, then all options that are available in the application will be displayed. If the logged on user is a Manager, then options to Encrypt String, Encrypt File, Decrypt String and Decrypt File will be displayed. Encrypt String and Encrypt File options will be displayed for users who have permissions to encrypt alone. Decrypt String and Decrypt File options will be displayed for users who have permissions to decrypt alone.

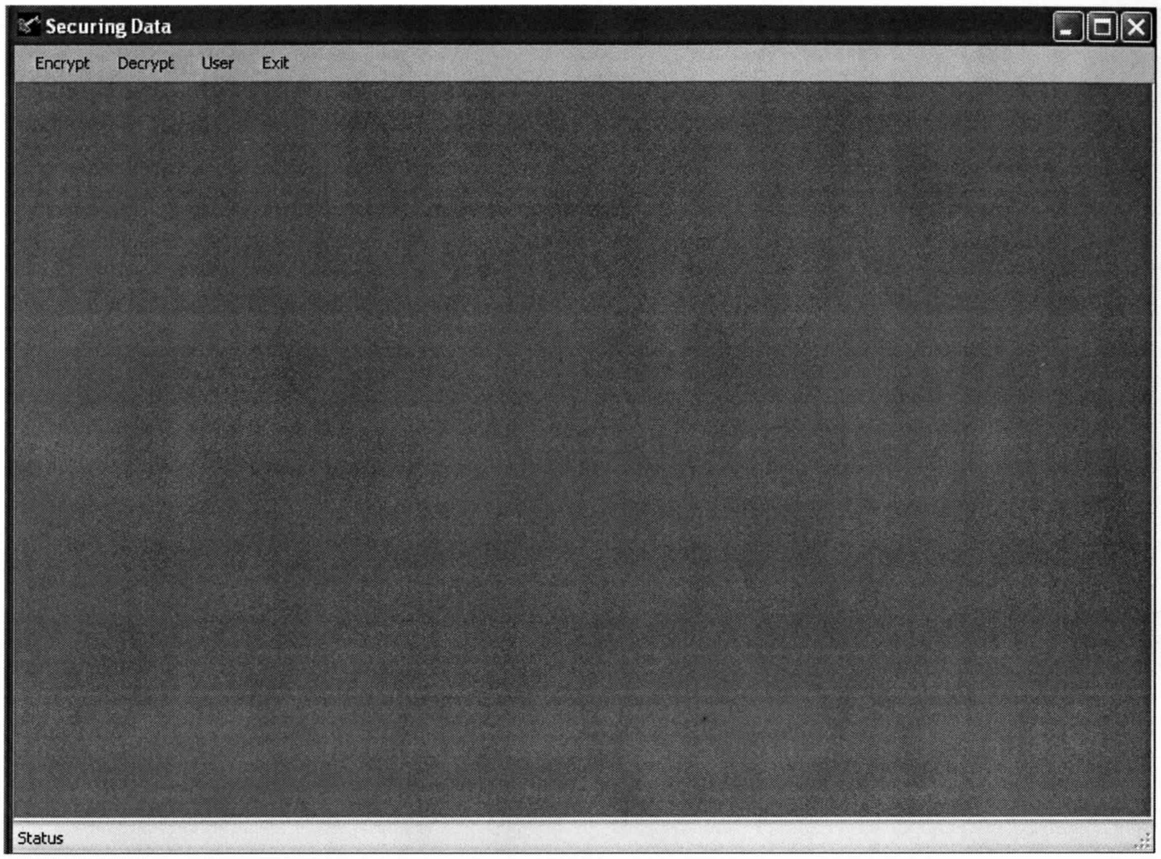
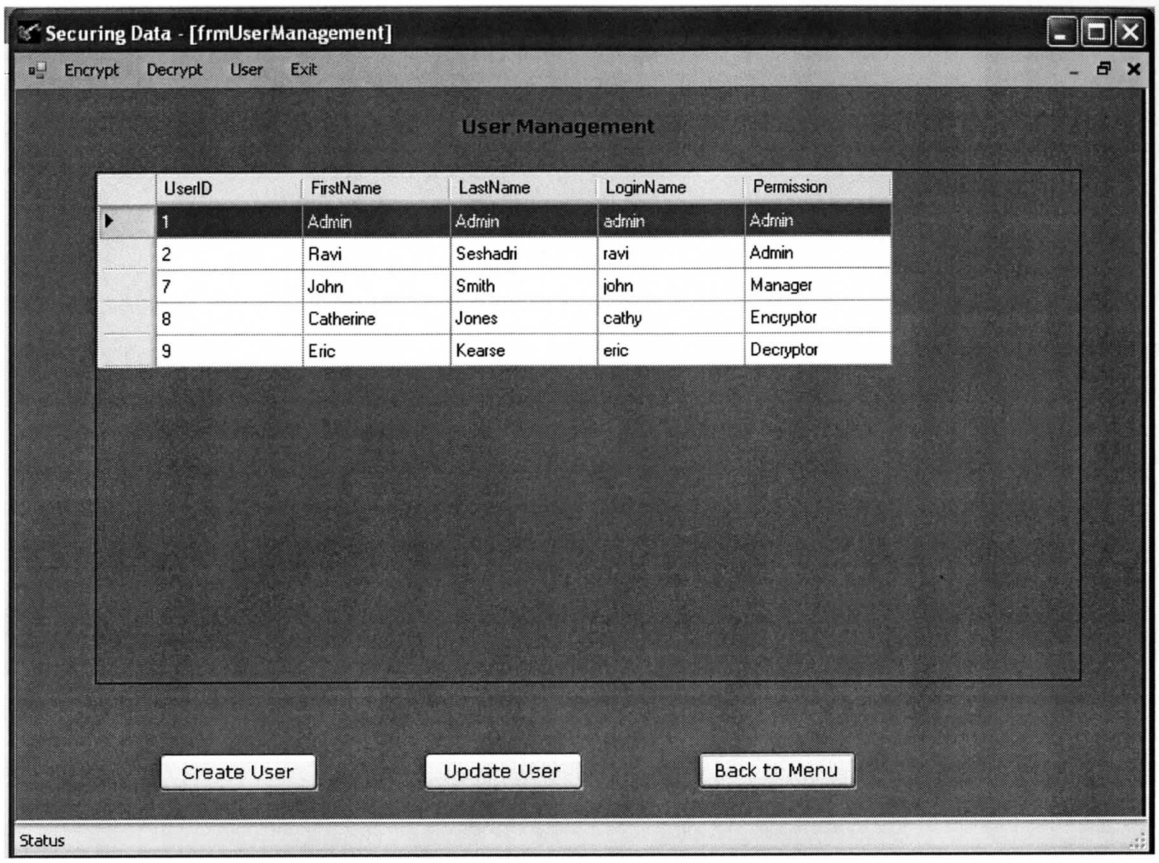


Fig 2.2 Main Menu Form

Manage Users Form

Administrators can use this form to view the users. A grid will be displayed with the list of users in the system. The admin can select a user to edit the information or delete from the system. A 'Create User' button will be present in the form, if the admin wishes to create a new user.

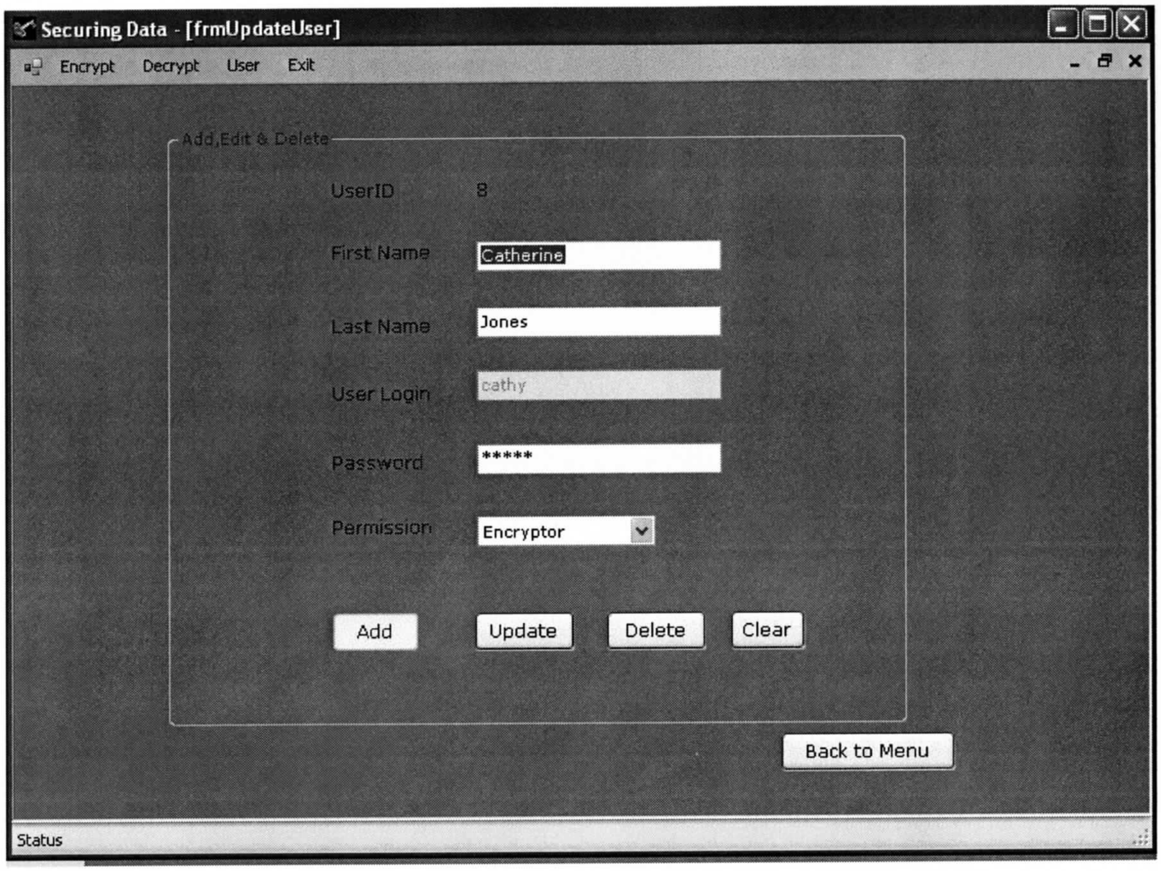


UserID	FirstName	LastName	LoginName	Permission
1	Admin	Admin	admin	Admin
2	Ravi	Seshadri	ravi	Admin
7	John	Smith	john	Manager
8	Catherine	Jones	cathy	Encryptor
9	Eric	Kearse	eric	Decryptor

Fig 2.3 Manage Users

Update User Form

This form will be displayed along with the user information, when the admin selects a user from the Manage Users Form. The fields will be pre populated with the user information and the admin can edit the fields. There will be four buttons in the bottom of the form. The first button is 'Create User' and will be enabled only if the user has clicked 'Create User' from the Manage Users Form. The second button is 'Update User' and the third button is 'Delete User'. These buttons are enabled if the user has clicked 'Edit User' from the Manage Users Form. The fourth button is 'Cancel' if the admin wishes to quit from this screen without making any changes. Proper validations will be made prior to submitting the form.


Securing Data - [frmUpdateUser]

Encrypt Decrypt User Exit

Add,Edit & Delete

UserID	8
First Name	Catherine
Last Name	Jones
User Login	cathy
Password	*****
Permission	Encryptor

Fig 2.4 Update Users

Encrypt String Form

This form has a text area where the user can enter the string he wishes to encrypt, a drop down which has various algorithms and a text field where the user will be entering the password. There will be an 'Encrypt Button' and when the user clicks this button, the entered string will be encrypted using the selected algorithm and the password entered by the user. The result will be in a text area that cannot be edited.

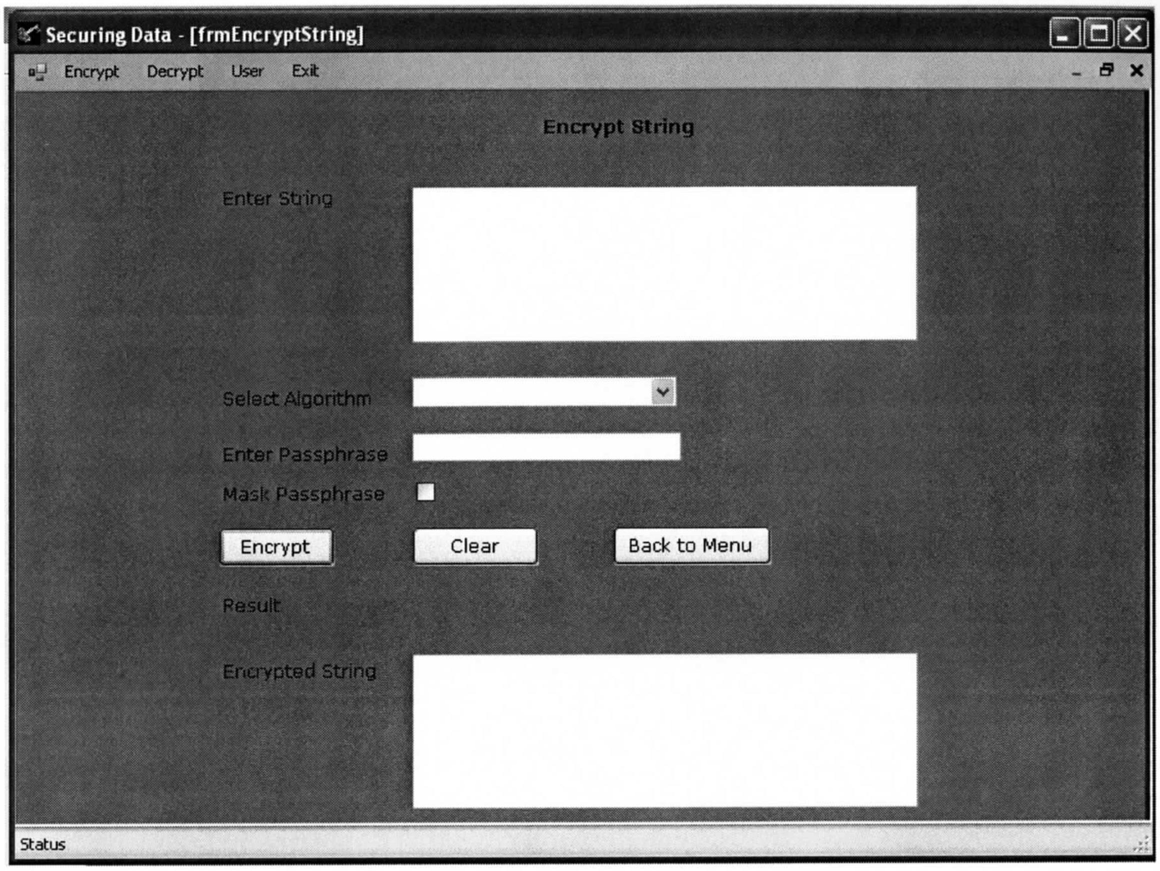


Fig 2.5 Encrypt String

Decrypt String Form

This form has a text area where the user can enter the encrypted data that he wishes to decrypt, a drop down which has various algorithms and a text field where the user will be entering the password. The algorithm and the password must match the values used while encrypting the string. There will be a 'Decrypt Button' and when the user clicks this button, the encrypted string will be decrypted using the selected algorithm and the password entered by the user. The result will be in a text area that cannot be edited.

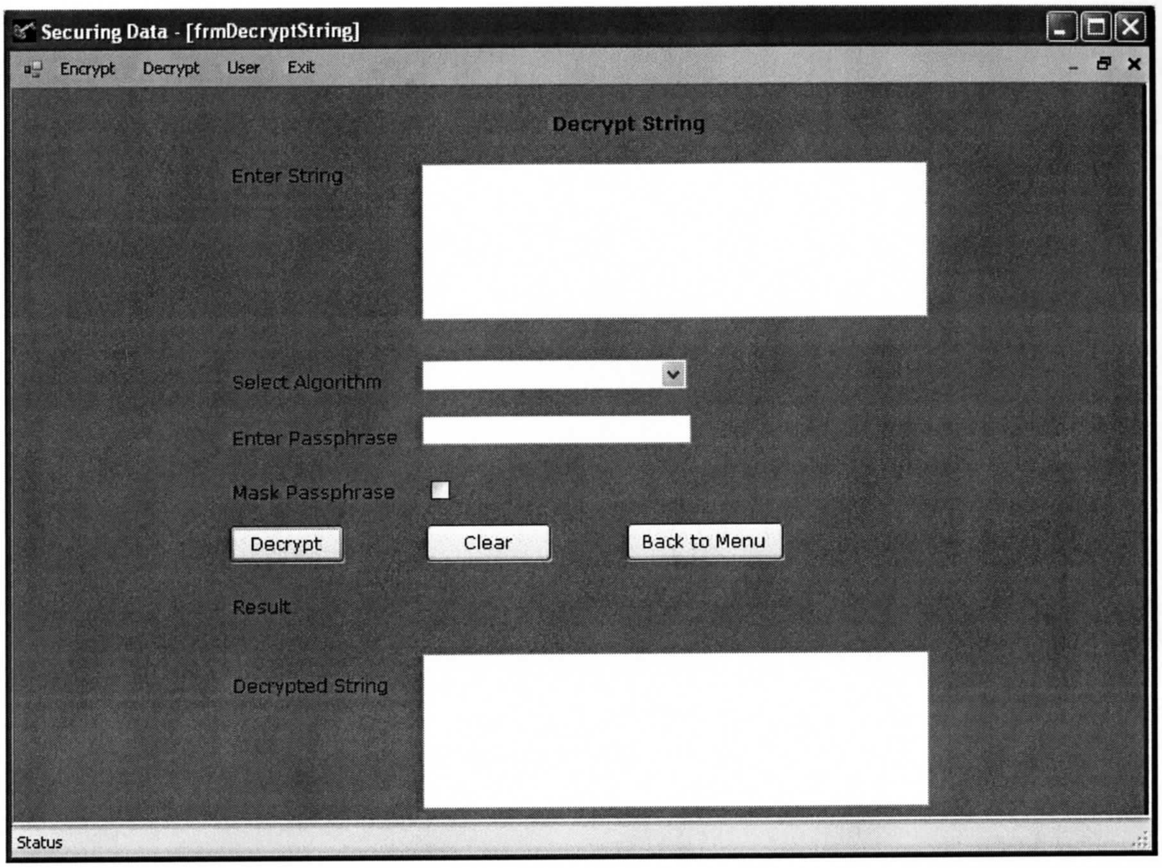


Fig 2.6 Decrypt String

Encrypt File Form

This form has a text box along with a 'Select File' button where the user can select a file that is present in the local machine that has to be encrypted. A drop down list that has various algorithms and a text field will be displayed, where the user will be entering the password. There will be another text box and 'Select Destination' button, where the user has to select the path and provide the file name in which the encrypted file will be saved. There will be an 'Encrypt Button' and when the user clicks this button, the selected file will be encrypted and saved in the destination entered by the user.

Validations will be performed to make sure the selected file exists and the selected destination folder exists.

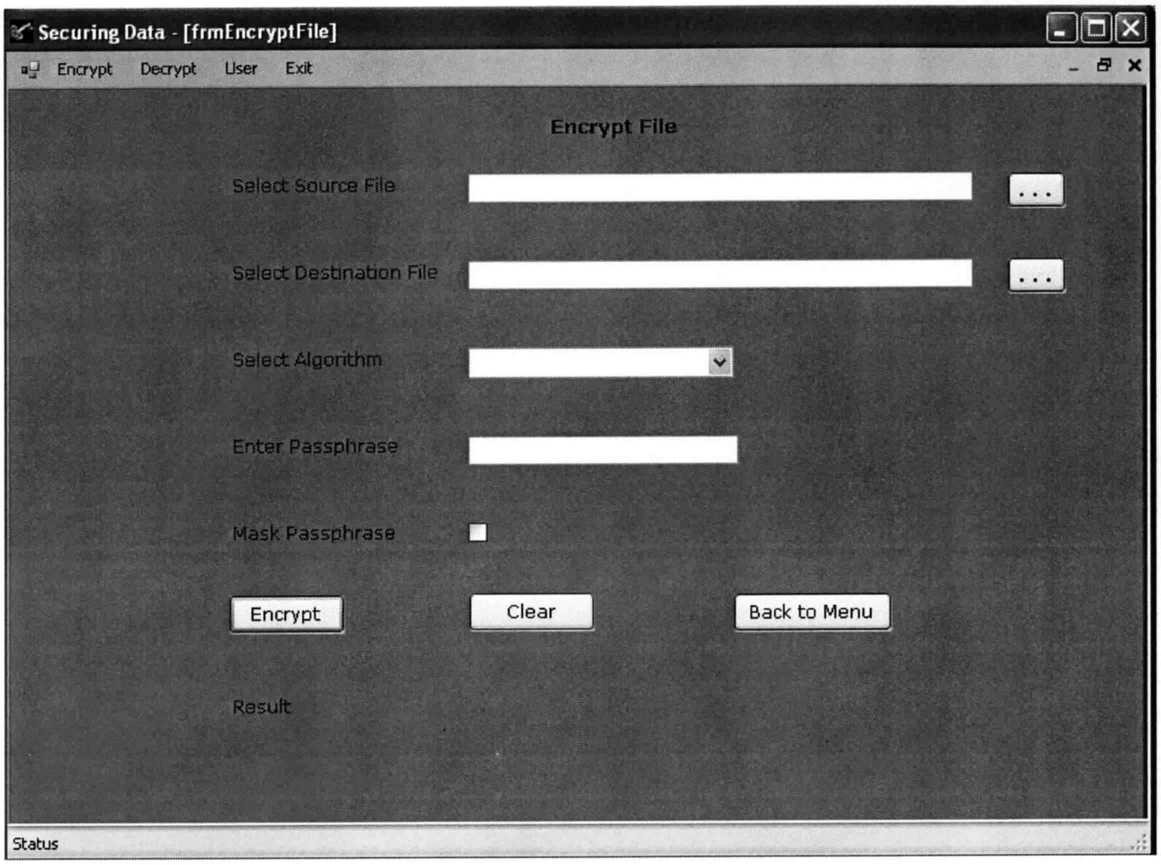


Fig 2.7 Encrypt File

Decrypt File Form

This form has a text box along with a 'Select File' button where the user can select a file that is present in the local machine that has to be decrypted. A drop down list that has various algorithms and a text field will be displayed, where the user will be entering the password. The algorithm and the password must match the values used while encrypting the file. There will be another text box and 'Select Destination' button, where the user has to select the path and provide the file name in which the decrypted file will be saved. There will be a 'Decrypt Button' and when the user clicks this button, the selected file will be decrypted and saved in the destination entered by the user. Validations will be performed to make sure the selected file exists and the selected destination folder exists.

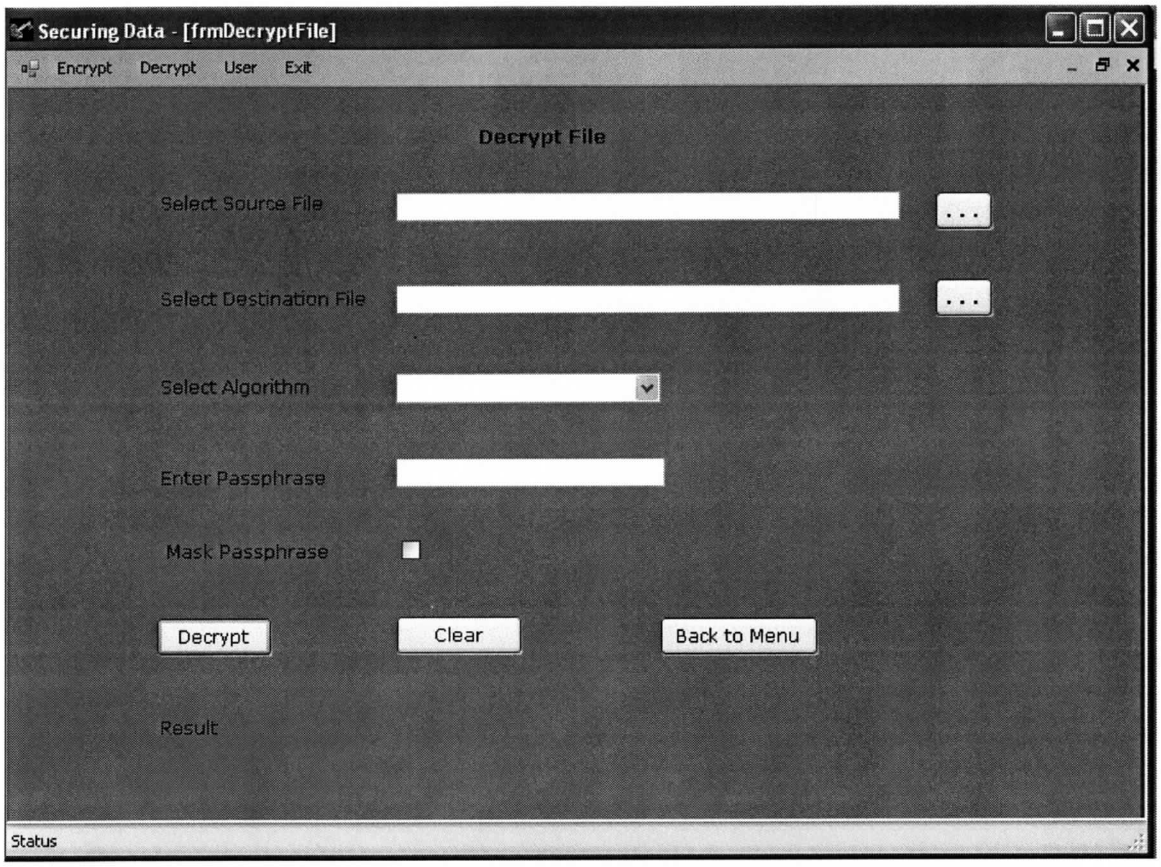


Fig 2.8 Decrypt File

RESULTS AND DISCUSSION

Securing data through encryption was tested thoroughly and installed in NCR e-Commerce, West Columbia, SC. NCR e-Commerce is a software development firm that has been delivering customized software solutions for over a decade. This application was installed to five business managers in NCR who frequently exchanges data and certain confidential information with the clients. At present they send the data through email with the data inside the email body or as an attachment. Since the clients were insisting on exchanging the data in more secured manner, it was decided to use this application to encrypt the data before sending.

In NCR, the application was installed in the five business managers' machine and they were connected to one SQL Server in the network. All the managers except one had the rights of 'Manager' where they can encrypt and decrypt the data. One manager has the 'Admin' rights so that he can create and edit user information. The application along with installation instructions and system requirements were provided two of clients of NCR. When client send confidential data, it was encrypted using a particular algorithm and a pass phrase. The clients sent the data to NCR and provided the algorithm and pass phrase over phone to NCR. One of the clients made arrangements to use the same algorithm and pass phrase until further notice.

Initial feedback from the clients to NCR indicated that they are satisfied with the application and the business managers are planning to provide this application to more clients.

CONCLUSIONS

Based on the feedback that I got from NCR, I feel that I have achieved what I have planned for this project in a timely manner and was finished ahead of schedule. The developers in NCR were also interested in this application and they wanted to the use the core component used in this project. The developers would like to use this component in their automated process where the clients will be sending the files to an FTP site and an application created by NCR will get the file from the FTP site and process. As an added security the clients can send the file in encrypted form and the NCR application can decrypt and process the file. According to Mary Kearse who works as e-Commerce Consultant in NCR, "Securing Data through Encryption is an user friendly application and innovative which can be installed in minutes"

REFERENCES

Phil Muncaster (2007, March). *Firms neglect mobile data security.* Retrieved June 17, 2007 from http://proquest.umi.com/pqdweb?did=1240390741&sid=2&Fmt=3&clientId=18865&RQT=309&VName=PQD

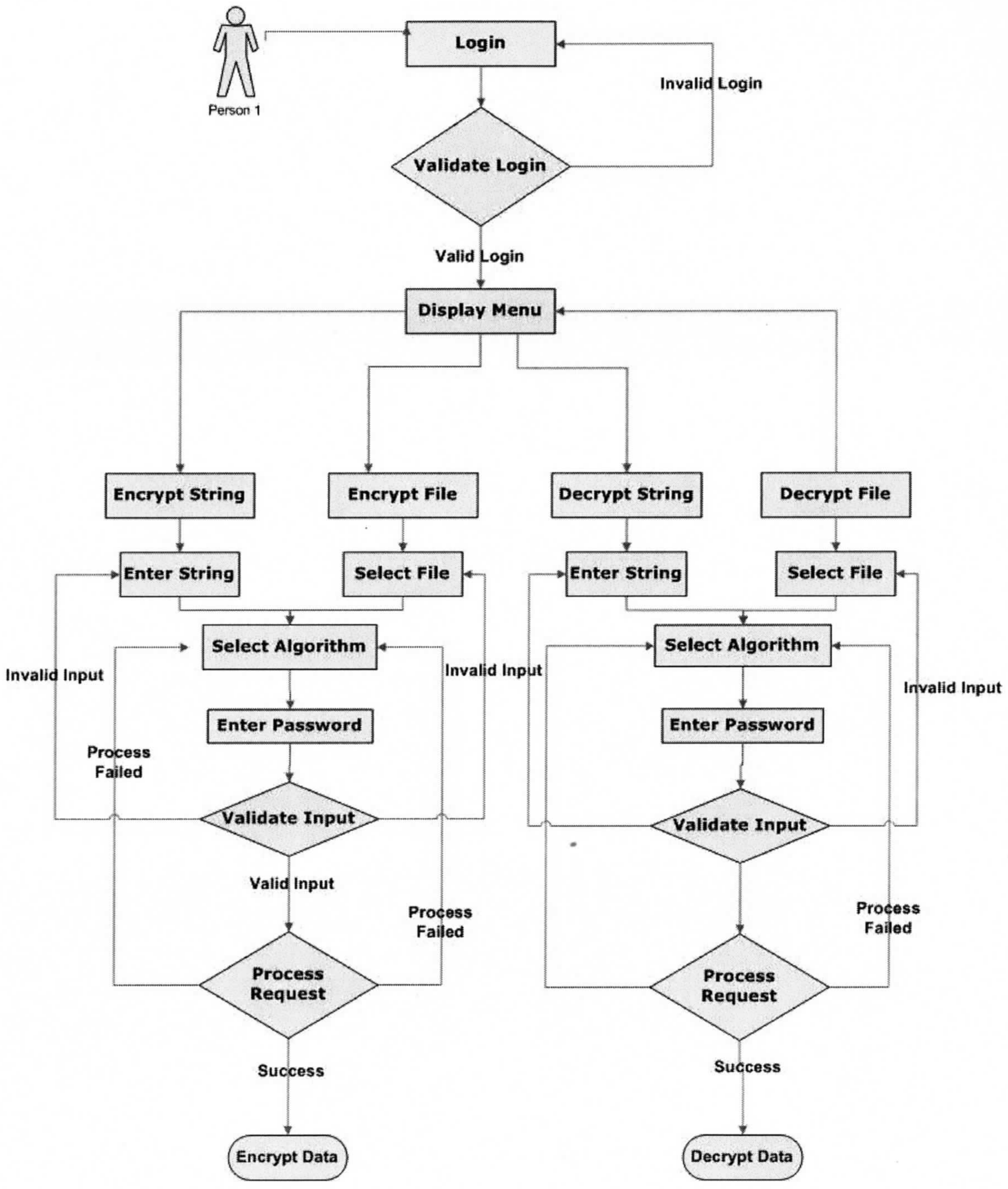
Cameron Laird, "Taking a Hard-Line Approach to Encryption," *Computer*, vol. 40, no. 3, pp. 13-15, Mar., 2007

Dave Zielinski (2002) *Choose your Presentation tools carefully Presentations.* Retrieved from http://proquest.umi.com/pqdweb?did=239880261&sid=12&Fmt=4&clientId=18865&RQT=309&VName=PQD on July 10, 2007

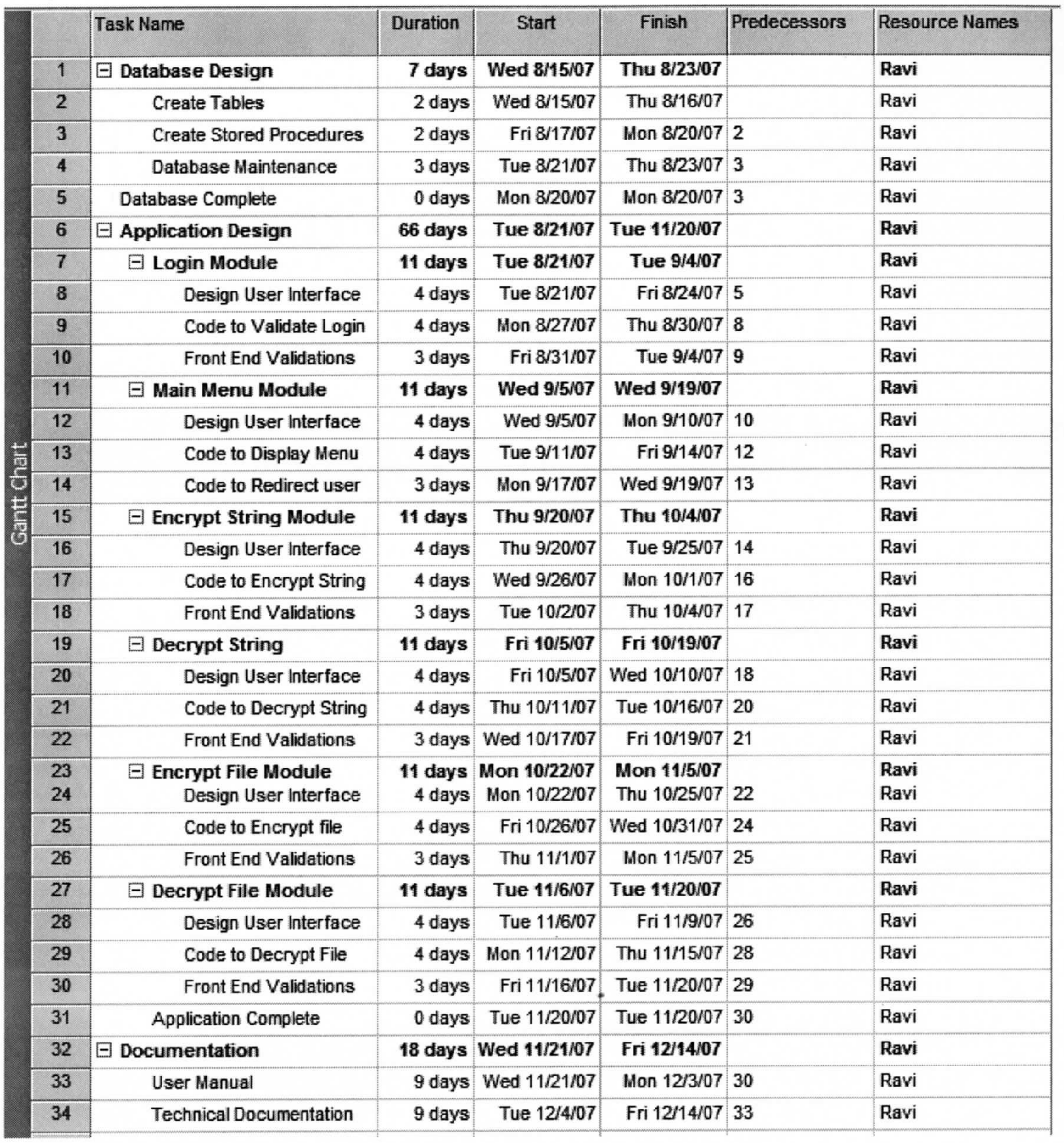
Poonam Khanna.(2005, June0. *Two-factor authentication is key to sound ID management: Schmidt* Retrieved July 12, 2007 from http://proquest.umi.com/pqdweb?did=859973911&sid=23&Fmt=4&clientId=18865&RQT=309&VName=PQD

APPENDIX A: USERS' MANUAL

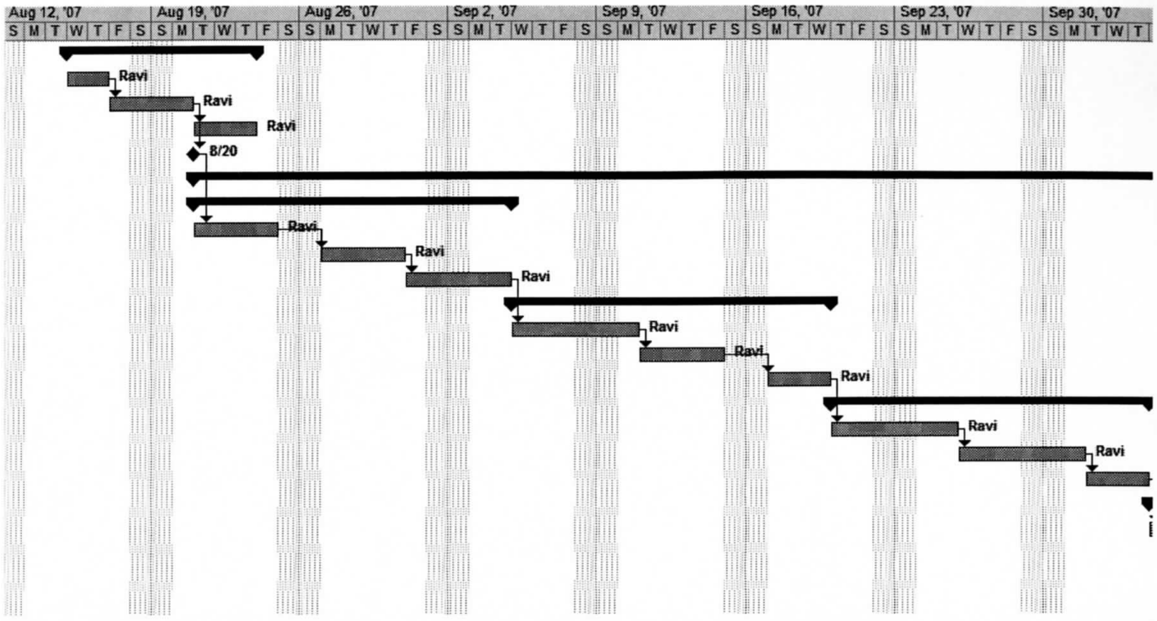
Application Flow

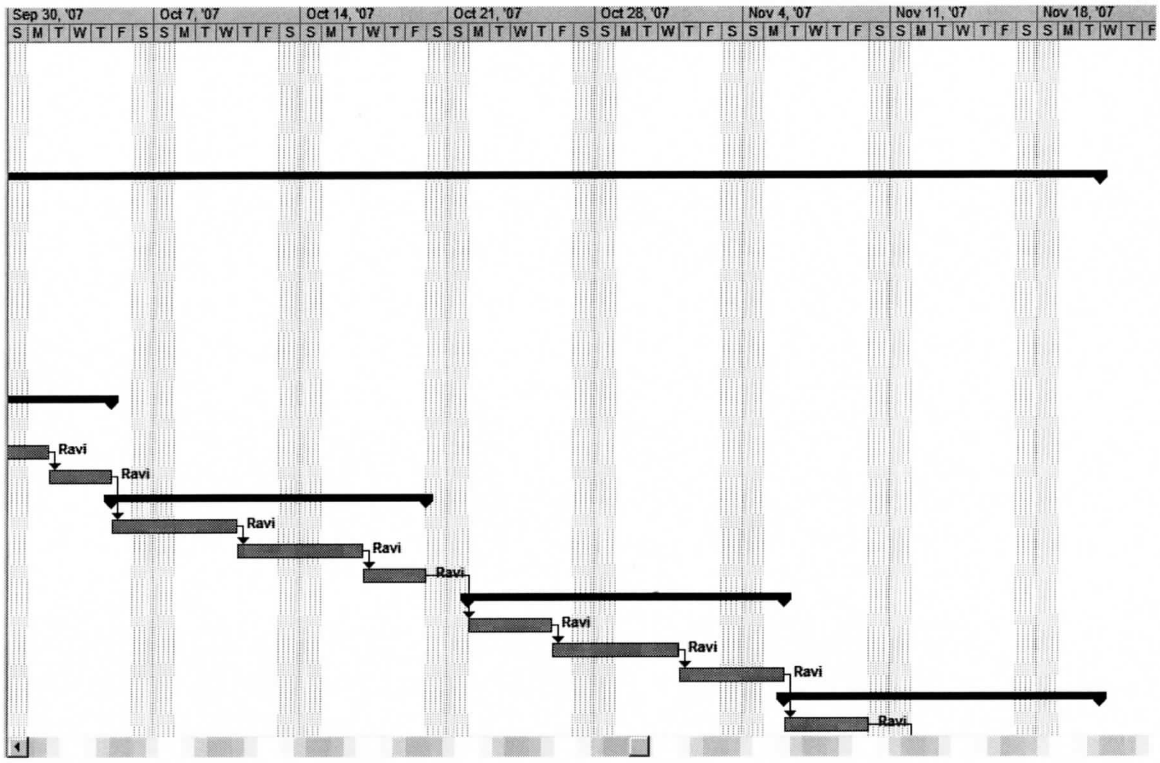


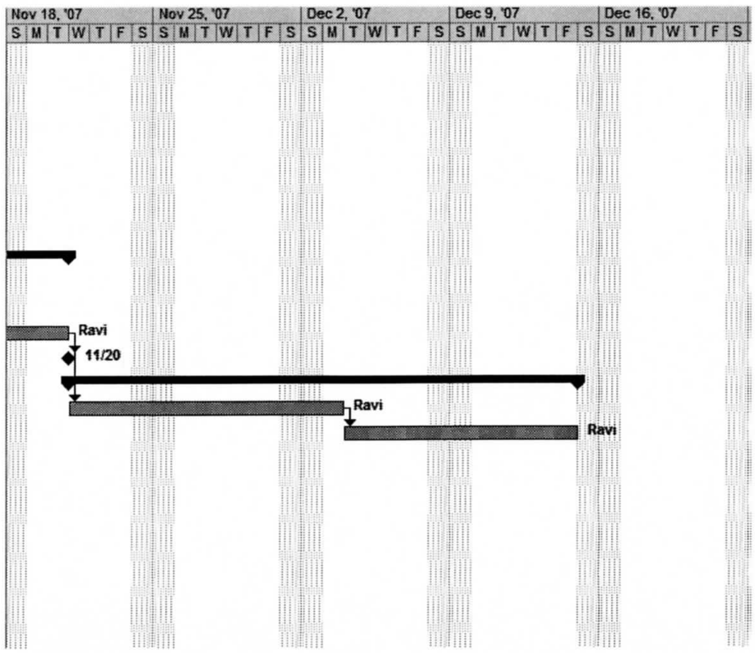
Project Plan


Gantt Chart

	Task Name	Duration	Start	Finish	Predecessors	Resource Names
1	⊟ **Database Design**	**7 days**	**Wed 8/15/07**	**Thu 8/23/07**		**Ravi**
2	Create Tables	2 days	Wed 8/15/07	Thu 8/16/07		Ravi
3	Create Stored Procedures	2 days	Fri 8/17/07	Mon 8/20/07	2	Ravi
4	Database Maintenance	3 days	Tue 8/21/07	Thu 8/23/07	3	Ravi
5	Database Complete	0 days	Mon 8/20/07	Mon 8/20/07	3	Ravi
6	⊟ **Application Design**	**66 days**	**Tue 8/21/07**	**Tue 11/20/07**		**Ravi**
7	⊟ **Login Module**	**11 days**	**Tue 8/21/07**	**Tue 9/4/07**		**Ravi**
8	Design User Interface	4 days	Tue 8/21/07	Fri 8/24/07	5	Ravi
9	Code to Validate Login	4 days	Mon 8/27/07	Thu 8/30/07	8	Ravi
10	Front End Validations	3 days	Fri 8/31/07	Tue 9/4/07	9	Ravi
11	⊟ **Main Menu Module**	**11 days**	**Wed 9/5/07**	**Wed 9/19/07**		**Ravi**
12	Design User Interface	4 days	Wed 9/5/07	Mon 9/10/07	10	Ravi
13	Code to Display Menu	4 days	Tue 9/11/07	Fri 9/14/07	12	Ravi
14	Code to Redirect user	3 days	Mon 9/17/07	Wed 9/19/07	13	Ravi
15	⊟ **Encrypt String Module**	**11 days**	**Thu 9/20/07**	**Thu 10/4/07**		**Ravi**
16	Design User Interface	4 days	Thu 9/20/07	Tue 9/25/07	14	Ravi
17	Code to Encrypt String	4 days	Wed 9/26/07	Mon 10/1/07	16	Ravi
18	Front End Validations	3 days	Tue 10/2/07	Thu 10/4/07	17	Ravi
19	⊟ **Decrypt String**	**11 days**	**Fri 10/5/07**	**Fri 10/19/07**		**Ravi**
20	Design User Interface	4 days	Fri 10/5/07	Wed 10/10/07	18	Ravi
21	Code to Decrypt String	4 days	Thu 10/11/07	Tue 10/16/07	20	Ravi
22	Front End Validations	3 days	Wed 10/17/07	Fri 10/19/07	21	Ravi
23	⊟ **Encrypt File Module**	**11 days**	**Mon 10/22/07**	**Mon 11/5/07**		**Ravi**
24	Design User Interface	4 days	Mon 10/22/07	Thu 10/25/07	22	Ravi
25	Code to Encrypt file	4 days	Fri 10/26/07	Wed 10/31/07	24	Ravi
26	Front End Validations	3 days	Thu 11/1/07	Mon 11/5/07	25	Ravi
27	⊟ **Decrypt File Module**	**11 days**	**Tue 11/6/07**	**Tue 11/20/07**		**Ravi**
28	Design User Interface	4 days	Tue 11/6/07	Fri 11/9/07	26	Ravi
29	Code to Decrypt File	4 days	Mon 11/12/07	Thu 11/15/07	28	Ravi
30	Front End Validations	3 days	Fri 11/16/07	Tue 11/20/07	29	Ravi
31	Application Complete	0 days	Tue 11/20/07	Tue 11/20/07	30	Ravi
32	⊟ **Documentation**	**18 days**	**Wed 11/21/07**	**Fri 12/14/07**		**Ravi**
33	User Manual	9 days	Wed 11/21/07	Mon 12/3/07	30	Ravi
34	Technical Documentation	9 days	Tue 12/4/07	Fri 12/14/07	33	Ravi


Aug 12, '07
Aug 19, '07
Aug 26, '07
Sep 2, '07
Sep 9, '07
Sep 16, '07
Sep 23, '07
Sep 30, '07
Ravi
8/20


Sep 30, '07
Oct 7, '07
Oct 14, '07
Oct 21, '07
Oct 28, '07
Nov 4, '07
Nov 11, '07
Nov 18, '07
Ravi


Nov 18, '07
Nov 25, '07
Dec 2, '07
Dec 9, '07
Dec 16, '07
S M T W T F S S M T W T F S S M T W T F S S M T W T F S S M T W T F S
Ravi
11/20
Ravi
Ravi

Using the application

1. Double Click the exe file 'SecuringData.exe' found under

SecuringData\bin

2. If you have not installed SQL Server, then click 'Login' button

3. **To Encrypt String**

a. Click the menu 'Encrypt' and Click sub menu 'Encrypt String'

b. Enter the string you want to encrypt in the text box.

c. Select the Algorithm you wish to apply.

d. Enter the Pass phrase that you want to use. This is needed while decrypting the string

e. Click the Check box 'Mask Pass phrase' if you do not want others to view the pass phrase while you type.

f. Click the button 'Encrypt' to encrypt the string.

g. The encrypted string will be displayed in the text box.

4. **To decrypt String**

a. Click the menu 'Decrypt' and Click sub menu 'Decrypt String'

b. Enter the string you want to decrypt in the text box.

c. Select the Algorithm that you used while encrypting.

d. Enter the Pass phrase that was used to encrypt the string.

e. Click the Check box 'Mask Pass phrase' if you do not want others to view the pass phrase while you type.

f. Click the button 'Decrypt' to decrypt the string.

g. The decrypted string will be displayed in the text box.

5. **To Encrypt file**

a. Click the menu 'Encrypt' and Click sub menu 'Encrypt file'

b. Select the source file you want to encrypt.

c. Select the destination file where you want to save the encrypted file.

d. Select the Algorithm you wish to apply.

e. Enter the Pass phrase that you want to use. This is needed while decrypting the file

f. Click the Check box 'Mask Pass phrase' if you do not want others to view the pass phrase while you type.

g. Click the button 'Encrypt' to encrypt the file.

h. The result of the process will be displayed.

6. **To Decrypt file**

a. Click the menu Decrypt' and Click sub menu 'Decrypt file'

b. Select the source file you want to decrypt.

c. Select the destination file where you want to save the decrypted file.

d. Select the Algorithm that you used while encrypting the file.

e. Enter the Pass phrase that was used to encrypt the file.

f. Click the Check box 'Mask Pass phrase' if you do not want others to view the pass phrase while you type.

g. Click the button 'Decrypt' to decrypt the file.

h. The result of the process will be displayed.

7. To Add user

a. Click the menu 'Users' and click the sub menu 'view user'

b. Click the button 'Create User'

c. Enter first name, last name, login name password and the permission level

d. Click Add

8. To Edit user

a. Click the menu 'Users' and click the sub menu 'view user'

b. Select the user you wish to edit from the grid displayed

c. Click the button 'Update User'

d. Update first name, last name, login name password and the permission level

e. Click 'Update User'

9. To Delete user

a. Click the menu 'Users' and click the sub menu 'view user'

b. Select the user you wish to delete from the grid displayed

c. Click the button 'Update User'

d. Click 'Delete User'

10. To Exit from the application

a. Click 'Exit' from the menu and click 'quit' from the sub menu

APPENDIX B: GLOSSARY

CryptoStream:

Defines a stream that links data streams to cryptographic transformations.

DES:

Represents the base class for the Data Encryption Standard (DES) algorithm from which all DES implementations must derive.

DESCryptoServiceProvider:

Defines a wrapper object to access the cryptographic service provider (CSP) version of the Data Encryption Standard (DES) algorithm.

FileStream:

Exposes a Stream around a file, supporting both synchronous and asynchronous read and write operations.

HMACSHA1:

Computes a Hash-based Message Authentication Code (HMAC) using the SHA1 hash function.

ICryptoTransform:

Defines the basic operations of cryptographic transformations.

MemoryStream:

Creates a stream whose backing store is memory.

PasswordDeriveBytes:

Derives a key from a password using an extension of the PBKDF1 algorithm.

Rfc2898DeriveBytes:

Implements password-based key derivation functionality, PBKDF2, by using a pseudo-random number generator based on HMACSHA1.

Rijndael:

Represents the base class from which all implementations of the Rijndael symmetric encryption algorithm must inherit.

RijndaelManaged:

Accesses the managed version of the Rijndael algorithm. This class cannot be inherited

SHA1:

Computes the SHA1 hash for the input data.

Stream:

Provides a generic view of a sequence of bytes.

StreamWriter:

Implements a TextWriter for writing characters to a stream in a particular encoding.

TextWriter:

Represents a writer that can write a sequential series of characters. This class is abstract.

TripleDES:

Represents the base class for Triple Data Encryption Standard algorithms from which all TripleDES implementations must derive.

TripleDESCryptoServiceProvider:

Defines a wrapper object to access the cryptographic service provider (CSP) version of the TripleDES algorithm. This class cannot be inherited.

APPENDIX C: CODE

Application Layer:

```
using System;
using System.Collections.Generic;
using System.Text;
using System.Security.Cryptography;
using System.IO;


namespace DSUComponents
{
    public class DSUComp
    {
        private static string hashAlgorithm = "SHA1"; // can be "MD5"
        private static int passwordIterations = 2; // can be any number
        private static string initVector = "@1B2c3D4e5F6g7H8"; // must
be 16 bytes
        private static int keySize = 256; // can be 192 or 128
        private static string saltValue = "s@1tValue";

        /*
        ----------------------------------------------------------------
-------------------------------
        Method :  EncryptUsingDESCrypto
        Desc : To Encrypt a string using DES alogorithm
        Returns : Encrypted string
        ----------------------------------------------------------------
-------------------------------
        */

        public static string EncryptUsingDESCrypto(string
originalString, string passPhrase)
        {
            try
            {
                DESCryptoServiceProvider cryptoProvider = new
DESCryptoServiceProvider();
                cryptoProvider.Padding = PaddingMode.PKCS7;
                byte[] saltValueBytes =
Encoding.ASCII.GetBytes(saltValue);
                Rfc2898DeriveBytes dBytes = new
Rfc2898DeriveBytes(passPhrase, saltValueBytes);
                cryptoProvider.KeySize = 64;
                cryptoProvider.Key = dBytes.GetBytes(8);
                cryptoProvider.IV = dBytes.GetBytes(8);
                ICryptoTransform ict =
cryptoProvider.CreateEncryptor();
                MemoryStream memoryStream = new MemoryStream();
                CryptoStream cryptoStream = new
CryptoStream(memoryStream, ict, CryptoStreamMode.Write);
                StreamWriter writer = new StreamWriter(cryptoStream);
                writer.Write(originalString);
                writer.Flush();
                cryptoStream.FlushFinalBlock();
                writer.Flush();
                return Convert.ToBase64String(memoryStream.GetBuffer(),
0, (int)memoryStream.Length);
            }
            catch
            {
                return "ERROR";
            }
        }

        /*
        -----------------------------------------------------------
-------------------------------
        Method :  DecryptUsingDESCrypto
        Desc   : To Decrypt a string using DES alogorithm
        Returns: Decrypted string
        -----------------------------------------------------------
-------------------------------
        */
        public static string DecryptUsingDESCrypto(string
cryptedString, string passPhrase)
        {
            try
            {
                DESCryptoServiceProvider cryptoProvider = new
DESCryptoServiceProvider();
                cryptoProvider.Padding = PaddingMode.PKCS7;
                byte[] saltValueBytes =
Encoding.ASCII.GetBytes(saltValue);
                Rfc2898DeriveBytes dBytes = new
Rfc2898DeriveBytes(passPhrase, saltValueBytes);
                cryptoProvider.KeySize = 64;
                cryptoProvider.Key = dBytes.GetBytes(8);
                cryptoProvider.IV = dBytes.GetBytes(8);
                ICryptoTransform ict =
cryptoProvider.CreateDecryptor();
                MemoryStream memoryStream = new
MemoryStream(Convert.FromBase64String(cryptedString));
                CryptoStream cryptoStream = new
CryptoStream(memoryStream, ict, CryptoStreamMode.Read);
                StreamReader reader = new StreamReader(cryptoStream);
                return reader.ReadToEnd();
            }
            catch
            {
                return "ERROR";
            }
        }
        /*
        ----------------------------------------------------------------
------------------------------
        Method :  EncryptUsingTripleDES
        Desc : To Encrypt a string using Triple DES alogorithm
        Returns : Encrypted string
        ----------------------------------------------------------------
------------------------------
        */

        public static string EncryptUsingTripleDES(string
originalString, string passPhrase)
        {
            try
            {
                TripleDESCryptoServiceProvider TripleDEScsp = new
TripleDESCryptoServiceProvider();
                TripleDEScsp.Padding = PaddingMode.PKCS7;
                byte[] saltValueBytes =
Encoding.ASCII.GetBytes(saltValue);
                Rfc2898DeriveBytes dBytes = new
Rfc2898DeriveBytes(passPhrase, saltValueBytes);
                TripleDEScsp.KeySize = 192;
                TripleDEScsp.Key = dBytes.GetBytes(24);
                TripleDEScsp.IV = dBytes.GetBytes(8);
                ICryptoTransform ict = TripleDEScsp.CreateEncryptor();
                MemoryStream memoryStream = new MemoryStream();
                CryptoStream cryptoStream = new
CryptoStream(memoryStream, ict, CryptoStreamMode.Write);
                StreamWriter writer = new StreamWriter(cryptoStream);
                writer.Write(originalString);
                writer.Flush();
                cryptoStream.FlushFinalBlock();
                writer.Flush();
                return Convert.ToBase64String(memoryStream.GetBuffer(),
0, (int)memoryStream.Length);
            }
            catch
            {
                return "ERROR";
            }
        }
        /*
        ----------------------------------------------------------------
------------------------------
        Method :  DecryptUsingTripleDES
        Desc : To Decrypt a string using Triple DES alogorithm
        Returns : Decrypted string
        ----------------------------------------------------------------
------------------------------
        */
        public static string DecryptUsingTripleDES(string
cryptedString, string passPhrase)
        {
            try
            {
                TripleDESCryptoServiceProvider TripleDEScsp = new
TripleDESCryptoServiceProvider();
                TripleDEScsp.Padding = PaddingMode.PKCS7;
                byte[] saltValueBytes =
Encoding.ASCII.GetBytes(saltValue);
                Rfc2898DeriveBytes dBytes = new
Rfc2898DeriveBytes(passPhrase, saltValueBytes);
                TripleDEScsp.KeySize = 192;
                TripleDEScsp.Key = dBytes.GetBytes(24);
                TripleDEScsp.IV = dBytes.GetBytes(8);
                ICryptoTransform ict = TripleDEScsp.CreateDecryptor();
                MemoryStream memoryStream = new
MemoryStream(Convert.FromBase64String(cryptedString));
                CryptoStream cryptoStream = new
CryptoStream(memoryStream, ict, CryptoStreamMode.Read);
                StreamReader reader = new StreamReader(cryptoStream);
                return reader.ReadToEnd();
            }
            catch
            {
                return "ERROR";
            }
        }
        /*
        ----------------------------------------------------------------
------------------------------
        Method :  EncryptUsingRijandel
        Desc : To Encrypt a string using Rijandel alogorithm
        Returns : Encrypted string
        ----------------------------------------------------------------
------------------------------
        */
        public static string EncryptUsingRijandel(string
originalString, string passPhrase)
        {
            try
            {
                byte[] initVectorBytes =
Encoding.ASCII.GetBytes(initVector);
                byte[] saltValueBytes =
Encoding.ASCII.GetBytes(saltValue);
                byte[] plainTextBytes =
Encoding.UTF8.GetBytes(originalString);
                PasswordDeriveBytes password = new
PasswordDeriveBytes(passPhrase, saltValueBytes, hashAlgorithm,
passwordIterations);
                byte[] keyBytes = password.GetBytes(keySize / 8);
                RijndaelManaged symmetricKey = new RijndaelManaged();
                symmetricKey.Mode = CipherMode.CBC;
                ICryptoTransform ict =
symmetricKey.CreateEncryptor(keyBytes, initVectorBytes);
                MemoryStream memoryStream = new MemoryStream();
                CryptoStream cryptoStream = new
CryptoStream(memoryStream, ict, CryptoStreamMode.Write);
                cryptoStream.Write(plainTextBytes, 0,
plainTextBytes.Length);
```

```
                cryptoStream.FlushFinalBlock();
                byte[] cipherTextBytes = memoryStream.ToArray();
                memoryStream.Close();
                cryptoStream.Close();
                string cipherText =
Convert.ToBase64String(cipherTextBytes);
                return cipherText;
            }
            catch
            {
                return "ERROR";
            }
        }
        /*
        ----------------------------------------------------------------
------------------------------
        Method :  DecryptUsingRijandel
        Desc : To Decrypt a string using Rijandel alogorithm
        Returns : Decrypted string
        ----------------------------------------------------------------
------------------------------
        */
        public static string DecryptUsingRijandel(string cipherText,
string passPhrase)
        {
            try
            {
                byte[] initVectorBytes =
Encoding.ASCII.GetBytes(initVector);
                byte[] saltValueBytes =
Encoding.ASCII.GetBytes(saltValue);
                byte[] cipherTextBytes =
Convert.FromBase64String(cipherText);
                PasswordDeriveBytes password = new
PasswordDeriveBytes(passPhrase, saltValueBytes, hashAlgorithm,
passwordIterations);
                byte[] keyBytes = password.GetBytes(keySize / 8);
                RijndaelManaged symmetricKey = new RijndaelManaged();
                symmetricKey.Mode = CipherMode.CBC;
                ICryptoTransform ict =
symmetricKey.CreateDecryptor(keyBytes, initVectorBytes);
                MemoryStream memoryStream = new
MemoryStream(cipherTextBytes);
                CryptoStream cryptoStream = new
CryptoStream(memoryStream, ict, CryptoStreamMode.Read);
                byte[] plainTextBytes = new
byte[cipherTextBytes.Length];
                int decryptedByteCount =
cryptoStream.Read(plainTextBytes, 0, plainTextBytes.Length);
                memoryStream.Close(); cryptoStream.Close();
                string plainText =
Encoding.UTF8.GetString(plainTextBytes, 0, decryptedByteCount);
                return plainText;
            }
            catch
            {
                return "ERROR";
            }
        }
        /*
        ---------------------------------------------------------------
------------------------------
        Method :  EncryptFileUsingDESCrypto
        Desc : To Encrypt a file using DES alogorithm
        Returns : Success or Error
        ---------------------------------------------------------------
------------------------------
        */

        public static string EncryptFileUsingDESCrypto(string
sInputFilename, string sOutputFilename, string passPhrase)
        {
            try
            {
                FileStream fsInput = new FileStream(sInputFilename,
FileMode.Open, FileAccess.Read);
                FileStream fsEncrypted = new
FileStream(sOutputFilename, FileMode.Create, FileAccess.Write);
                DESCryptoServiceProvider cryptoProvider = new
DESCryptoServiceProvider();

                cryptoProvider.Padding = PaddingMode.PKCS7;
                byte[] saltValueBytes =
Encoding.ASCII.GetBytes(saltValue);
                Rfc2898DeriveBytes dBytes = new
Rfc2898DeriveBytes(passPhrase, saltValueBytes);
                cryptoProvider.KeySize = 64;
                cryptoProvider.Key = dBytes.GetBytes(8);
                cryptoProvider.IV = dBytes.GetBytes(8);
                ICryptoTransform ict =
cryptoProvider.CreateEncryptor();

                CryptoStream cryptostream = new
CryptoStream(fsEncrypted, ict, CryptoStreamMode.Write);
                byte[] bytearrayinput = new byte[fsInput.Length];
                fsInput.Read(bytearrayinput, 0, bytearrayinput.Length);
                cryptostream.Write(bytearrayinput, 0,
bytearrayinput.Length);
                cryptostream.Close();
                fsInput.Close();
                fsEncrypted.Close();
                return "success";
            }
            catch
            {
                return ("ERROR");
            }
        }
        /*
        ---------------------------------------------------------------
------------------------------
        Method :  DecryptFileUsingDESCrypto
        Desc : To Decrypt a file using DES alogorithm
        Returns : Success or Error
        ---------------------------------------------------------------
------------------------------
        */

        public static string DecryptFileUsingDESCrypto(string
sInputFilename, string sOutputFilename, string passPhrase)
        {
            try
            {
                DESCryptoServiceProvider cryptoProvider = new
DESCryptoServiceProvider();
                cryptoProvider.Padding = PaddingMode.PKCS7;
                byte[] saltValueBytes =
Encoding.ASCII.GetBytes(saltValue);
                Rfc2898DeriveBytes dBytes = new
Rfc2898DeriveBytes(passPhrase, saltValueBytes);
                cryptoProvider.KeySize = 64;
                cryptoProvider.Key = dBytes.GetBytes(8);
                cryptoProvider.IV = dBytes.GetBytes(8);

                FileStream fsread = new FileStream(sInputFilename,
FileMode.Open, FileAccess.Read);
                ICryptoTransform ict =
cryptoProvider.CreateDecryptor();
                CryptoStream cryptostreamDecr = new
CryptoStream(fsread, ict, CryptoStreamMode.Read);
                StreamWriter fsDecrypted = new
StreamWriter(sOutputFilename);
                fsDecrypted.Write(new
StreamReader(cryptostreamDecr).ReadToEnd());
                fsDecrypted.Flush();
                fsDecrypted.Close();
                return "success";
            }
            catch
            {
                return ("ERROR");
            }
        }
        /*
        ---------------------------------------------------------------
------------------------------
        Method :  EncryptFileUsingTripleDES
        Desc : To Encrypt a file using Triple DES alogorithm
        Returns : Success or Error
        ---------------------------------------------------------------
------------------------------
        */

        public static string EncryptFileUsingTripleDES(string
sInputFilename, string sOutputFilename, string passPhrase)
        {
            try
            {
                FileStream fsInput = new FileStream(sInputFilename,
FileMode.Open, FileAccess.Read);
```

```
                FileStream fsEncrypted = new
FileStream(sOutputFilename, FileMode.Create, FileAccess.Write);
                TripleDESCryptoServiceProvider TripleDEScsp = new
TripleDESCryptoServiceProvider();
                TripleDEScsp.Padding = PaddingMode.PKCS7;
                byte[] saltValueBytes =
Encoding.ASCII.GetBytes(saltValue);
                Rfc2898DeriveBytes dBytes = new
Rfc2898DeriveBytes(passPhrase, saltValueBytes);
                TripleDEScsp.KeySize = 192;
                TripleDEScsp.Key = dBytes.GetBytes(24);
                TripleDEScsp.IV = dBytes.GetBytes(8);

                ICryptoTransform ict = TripleDEScsp.CreateEncryptor();
                CryptoStream cryptostream = new
CryptoStream(fsEncrypted, ict, CryptoStreamMode.Write);
                byte[] bytearrayinput = new byte[fsInput.Length];
                fsInput.Read(bytearrayinput, 0, bytearrayinput.Length);
                cryptostream.Write(bytearrayinput, 0,
bytearrayinput.Length);
                cryptostream.Close();
                fsInput.Close();
                fsEncrypted.Close();
                return "success";
            }
            catch
            {
                return ("ERROR");
            }

        }
        /*
        -----------------------------------------------------------------
------------------------------
        Method :  DecryptFileUsingTripleDES
        Desc : To Decrypt a file using Triple DES alogorithm
        Returns : Success or Error
        -----------------------------------------------------------------
------------------------------
        */

        public static string DecryptFileUsingTripleDES(string
sInputFilename, string sOutputFilename, string passPhrase)
        {
            try
            {
                TripleDESCryptoServiceProvider TripleDEScsp = new
TripleDESCryptoServiceProvider();
                TripleDEScsp.Padding = PaddingMode.PKCS7;
                byte[] saltValueBytes =
Encoding.ASCII.GetBytes(saltValue);
                Rfc2898DeriveBytes dBytes = new
Rfc2898DeriveBytes(passPhrase, saltValueBytes);
                TripleDEScsp.KeySize = 192;
                TripleDEScsp.Key = dBytes.GetBytes(24);
                TripleDEScsp.IV = dBytes.GetBytes(8);
                FileStream fsread = new FileStream(sInputFilename,
FileMode.Open, FileAccess.Read);
                ICryptoTransform ict = TripleDEScsp.CreateDecryptor();
                CryptoStream cryptostreamDecr = new
CryptoStream(fsread, ict, CryptoStreamMode.Read);
                StreamWriter fsDecrypted = new
StreamWriter(sOutputFilename);
                fsDecrypted.Write(new
StreamReader(cryptostreamDecr).ReadToEnd());
                fsDecrypted.Flush();
                fsDecrypted.Close();
                return "success";
            }
            catch
            {
                return ("ERROR");
            }
        }
        /*
        ---------------------------------------------------------------
-----------------------------
        Method :  EncryptFileUsingRijandel
        Desc : To Encrypt a file using Rijandel alogorithm
        Returns : Success or Error
        ---------------------------------------------------------------
-----------------------------
        */
        public static string EncryptFileUsingRijandel(string
sInputFilename, string sOutputFilename, string passPhrase)
        {
            try
            {
                FileStream fsInput = new FileStream(sInputFilename,
FileMode.Open, FileAccess.Read);
                FileStream fsEncrypted = new
FileStream(sOutputFilename, FileMode.Create, FileAccess.Write);
                byte[] initVectorBytes =
Encoding.ASCII.GetBytes(initVector);
                byte[] saltValueBytes =
Encoding.ASCII.GetBytes(saltValue);

                PasswordDeriveBytes password = new
PasswordDeriveBytes(passPhrase, saltValueBytes, hashAlgorithm,
passwordIterations);
                byte[] keyBytes = password.GetBytes(keySize / 8);
                RijndaelManaged symmetricKey = new RijndaelManaged();
                symmetricKey.Mode = CipherMode.CBC;
                ICryptoTransform ict =
symmetricKey.CreateEncryptor(keyBytes, initVectorBytes);

                CryptoStream cryptostream = new
CryptoStream(fsEncrypted, ict, CryptoStreamMode.Write);
                byte[] bytearrayinput = new byte[fsInput.Length];
                fsInput.Read(bytearrayinput, 0, bytearrayinput.Length);
                cryptostream.Write(bytearrayinput, 0,
bytearrayinput.Length);
                cryptostream.Close();
                fsInput.Close();
                fsEncrypted.Close();
                return "success";
            }
            catch
            {
                return ("ERROR");
            }

        }
        /*
        ---------------------------------------------------------------
-------------------------------
        Method :  DecryptFileUsingRijandel
        Desc : To Decrypt a file using Rijandel alogorithm
        Returns : Success or Error
        ---------------------------------------------------------------
-------------------------------
        */
        public static string DecryptFileUsingRijandel(string
sInputFilename, string sOutputFilename, string passPhrase)
        {
            try
            {
                byte[] initVectorBytes =
Encoding.ASCII.GetBytes(initVector);
                byte[] saltValueBytes =
Encoding.ASCII.GetBytes(saltValue);
                PasswordDeriveBytes password = new
PasswordDeriveBytes(passPhrase, saltValueBytes, hashAlgorithm,
passwordIterations);
                byte[] keyBytes = password.GetBytes(keySize / 8);
                RijndaelManaged symmetricKey = new RijndaelManaged();
                symmetricKey.Mode = CipherMode.CBC;
                ICryptoTransform ict =
symmetricKey.CreateDecryptor(keyBytes, initVectorBytes);
                FileStream fsread = new FileStream(sInputFilename,
FileMode.Open, FileAccess.Read);
                CryptoStream cryptostreamDecr = new
CryptoStream(fsread, ict, CryptoStreamMode.Read);
                StreamWriter fsDecrypted = new
StreamWriter(sOutputFilename);
                fsDecrypted.Write(new
StreamReader(cryptostreamDecr).ReadToEnd());
                fsDecrypted.Flush();
                fsDecrypted.Close();
                return "success";
            }
            catch
            {
                return ("ERROR");
            }

        }

    }
}
```

Presentation Layer:

LoginForm

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
using System.Configuration;
using System.Data.SqlClient;
using DSUComponents;


namespace SecuringData
{
    public partial class frmLogin : Form
    {
        string strSqlConn;
        SqlConnection objConn;
        string sPermission;

        public frmLogin()
        {
            InitializeComponent();
        }

        private void btnExit_Click(object sender, EventArgs e)
        {
            Application.Exit();

        }

        private void btnLogin_Click(object sender, EventArgs e)
        {

            //Form MDIParent = new MDIParent(iPermission);
            Form MDIParent;

            //Check config whether DB instllation exists
            string sCheckDBRequired = "";
            string sResult = "";
            try
            {
                sCheckDBRequired =
ConfigurationSettings.AppSettings["DBConnection"].ToString();
                sResult =
DSUComp.DecryptUsingRijandel(sCheckDBRequired, "ZSecurity");
            }
            catch
            {
                sResult = "";
            }

            if (sResult == "ByPass")
            {
                MDIParent = new MDIParent("Admin");
                MDIParent.Show();
                this.Hide();
                return;
            }

            if (txtUserName.Text.Trim() == "")
            {
                MessageBox.Show("Please enter the user name",
"UserName", MessageBoxButtons.OK, MessageBoxIcon.Stop);
                return;
            }

            if (txtPassword.Text.Trim() == "")
            {
                MessageBox.Show("Please enter the password",
"Password", MessageBoxButtons.OK, MessageBoxIcon.Stop);
                return;
            }
            if (!validateLogin())
            {
                MessageBox.Show("You have entered an invalid user name
or password", "Invalid login", MessageBoxButtons.OK,
MessageBoxIcon.Stop);
                return;
            }

            MDIParent = new MDIParent(sPermission);
            MDIParent.Show();
            this.Hide();
        }

        private bool validateLogin()
        {
            int intReturnVal = 1;

            OpenDatabase();
            try
            {
                DataSet dsUser = new DataSet();
                SqlDataAdapter objCmd;
                string sql = "Select userid,userpermission from
tblUsers where loginname = '" + txtUserName.Text.Trim() + "' and
password = '" + txtPassword.Text.Trim() + "'";
                objCmd = new SqlDataAdapter(sql, objConn);
                objCmd.Fill(dsUser, "UserInfo");
                if (dsUser.Tables[0].Rows.Count == 0)
                {
                    intReturnVal = 1;
                }
                else
                {
                        intReturnVal = 0;
                        sPermission =
dsUser.Tables[0].Rows[0]["userpermission"].ToString();

                    }
                }
                catch(Exception ex)
                {
                    MessageBox.Show(ex.ToString());
                }
                finally
                {
                    Closedatabase();
                }
                if (intReturnVal == 0)
                {
                    return true;
                }
                else
                {
                    return false;
                }
            }

            private void frmLogin_Load(object sender, EventArgs e)
            {

            }

            private bool OpenDatabase()
            {
                try
                {
                    strSqlConn =
ConfigurationSettings.AppSettings["conectionstring"].ToString();
                    objConn = new SqlConnection(strSqlConn);
                    objConn.Open();
                    return true;
                }
                catch (Exception ex)
                {
                    MessageBox.Show(ex.ToString());
                    return false;

                }
            }

            private void Closedatabase()
            {
                try
                {
                    if (objConn.State.ToString() == "Open")
                    {
                        objConn.Close();
                        objConn = null;
                    }
                }
            catch
            {
            }
        }
    }
     }
```

MDIForm

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

namespace SecuringData
{
    public partial class MDIParent : Form
    {
        private int childFormNumber = 0;
        string sUserPermission;
        public MDIParent(string uPerm)
        {
            InitializeComponent();
            uPermission = uPerm;
        }

        public string uPermission
        {
            set
            {
                sUserPermission = value;
            }

        }
        private void ShowNewForm(object sender, EventArgs e)
        {
            // Create a new instance of the child form.
            Form childForm = new Form();
            // Make it a child of this MDI form before showing it.
            childForm.MdiParent = this;
            childForm.Text = "Window " + childFormNumber++;
            childForm.Show();
        }


        private void ExitToolsStripMenuItem_Click(object sender,
EventArgs e)
        {
            Application.Exit();
        }
        private void CascadeToolStripMenuItem_Click(object sender,
EventArgs e)
        {
            LayoutMdi(MdiLayout.Cascade);
        }

        private void TileVerticleToolStripMenuItem_Click(object sender,
EventArgs e)
        {
            LayoutMdi(MdiLayout.TileVertical);
        }

        private void TileHorizontalToolStripMenuItem_Click(object
sender, EventArgs e)
        {
            LayoutMdi(MdiLayout.TileHorizontal);
        }

        private void ArrangeIconsToolStripMenuItem_Click(object sender,
EventArgs e)
        {
            LayoutMdi(MdiLayout.ArrangeIcons);
        }

        private void CloseAllToolStripMenuItem_Click(object sender,
EventArgs e)
        {
            foreach (Form childForm in MdiChildren)
            {
                childForm.Close();
            }
        }

        //private void fileMenu_Click(object sender, EventArgs e)
        //{

        //}

        private void wToolStripMenuItem_Click(object sender, EventArgs
e)
        {
            Form frmEncryptString = new frmEncryptString();
            frmEncryptString.MdiParent = this;
            frmEncryptString.Show();
        }

        private void encryptFileToolStripMenuItem_Click(object sender,
EventArgs e)
        {
            Form frmEncryptFile = new frmEncryptFile();
            frmEncryptFile.MdiParent = this;
            frmEncryptFile.Show();
        }
        private void viewUserToolStripMenuItem_Click(object sender,
EventArgs e)
        {
            Form frmUserManagement = new frmUserManagement();
            frmUserManagement.MdiParent = this;
            frmUserManagement.Show();
        }

        private void contentsToolStripMenuItem_Click(object sender,
EventArgs e)
        {
            Application.Exit();
        }



        private void decryptFileToolStripMenuItem_Click(object sender,
EventArgs e)
        {
            Form frmDecryptFile = new frmDecryptFile();
            frmDecryptFile.MdiParent = this;
            frmDecryptFile.Show();
        }


        private void decryptStringToolStripMenuItem1_Click(object
sender, EventArgs e)
        {
            Form frmDecryptString = new frmDecryptString();
            frmDecryptString.MdiParent = this;
            frmDecryptString.Show();
        }

        private void MDIParent_Load(object sender, EventArgs e)
        {
            mnuEncrypt.Enabled = false;
            mnuDecrypt.Enabled = false;
            mnuUser.Enabled = false;

            switch (sUserPermission.ToUpper())
            {
                case "ADMIN":
                        mnuEncrypt.Enabled = true;
                        mnuDecrypt.Enabled = true;
                        mnuUser.Enabled = true;
                        break;
                case "MANAGER":
                        mnuEncrypt.Enabled = true;
                        mnuDecrypt.Enabled = true;
                        break;
                case "ENCRYPTOR":
                        mnuEncrypt.Enabled = true;
                        break;
                case "DECRYPTOR":
                        mnuDecrypt.Enabled = true;
                        break;
            }
        }
    }
}
```

frmEncryptString

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

namespace SecuringData
{
    public partial class frmEncryptString : Form
    {
        public frmEncryptString()
        {
            InitializeComponent();
        }


        private void chkMaskPassphrase_CheckedChanged(object sender,
EventArgs e)
        {
            if (chkMaskPassphrase.Checked)
            {
                txtPassPhrase.Multiline = false;
                txtPassPhrase.PasswordChar = '*';
            }
            else
            {
                txtPassPhrase.Multiline = true;
            }
        }

        private void btnBack_Click(object sender, EventArgs e)
        {
            this.Dispose();
        }

        private void btnClear_Click(object sender, EventArgs e)
        {
            Clearfields();
        }

        private void btnEncrypt_Click(object sender, EventArgs e)
        {
            string sInputString = "";
            string sResult = "";
            string sPassPhrase = "";
            //Validate input
            if (txtInput.Text.Trim().ToString() == "")
            {
                MessageBox.Show("Please Enter the String you want to
Encrypt", "Error", MessageBoxButtons.OK, MessageBoxIcon.Warning);
                return;
            }

            //Make sure the user has selected the algorithm
            if (cboAlgorithm.Text.Trim() == "")
            {
                MessageBox.Show("Please select the algorithm to use for
Encrypting", "Error", MessageBoxButtons.OK, MessageBoxIcon.Warning);
                return;
            }
            //Make sure the user has entered the passphrase
            if (txtPassPhrase.Text.Trim().ToString() == "")
            {
                MessageBox.Show("Please enter the passphrase for
Encrypting", "Error", MessageBoxButtons.OK, MessageBoxIcon.Warning);
                return;
            }

            //Now Encrypt the string
            sInputString = txtInput.Text.Trim();
            sPassPhrase = txtPassPhrase.Text.Trim();
            lblResult.Text = "Encrypting text...";
            switch (cboAlgorithm.Text)
            {
                case "DES":
                    try
                    {
                        sResult =
DSUComp.EncryptUsingDESCrypto(sInputString, sPassPhrase);
                        if (sResult != "ERROR")
                        {
                            lblResult.Text = "Encrypted successfully";
                            txtOutput.Text = sResult;
                        }
                        else
                        {
                            txtOutput.Text = "Error while encrypting
the text";
                        }
                    }
                    catch(Exception ex)
                    {
                        lblResult.Text = "Error while encrypting the
text";

                        txtOutput.Text = ex.ToString();
                    }
                    break;

                case "TripleDES":
                    try
                    {
                        sResult =
DSUComp.EncryptUsingTripleDES(sInputString, sPassPhrase);
                        if (sResult != "ERROR")
                        {
                            lblResult.Text = "Encrypted successfully";
                            txtOutput.Text = sResult;
                        }
                        else
                        {
                            txtOutput.Text = "Error while encrypting
the text";
                        }
                    }
                    catch (Exception ex)
                    {
                        lblResult.Text = "Error while encrypting the
text";
                        txtOutput.Text = ex.ToString();
                    }
                    break;
                case "Rijandel":
                    try
                    {
                        sResult =
DSUComp.EncryptUsingRijandel(sInputString, sPassPhrase);
                        if (sResult != "ERROR")
                        {
                            lblResult.Text = "Encrypted successfully";
                            txtOutput.Text = sResult;
                        }
                        else
                        {
                            txtOutput.Text = "Error while encrypting
the text";
                        }
                    }
                    catch (Exception ex)
                    {
                        lblResult.Text = "Error while encrypting the
text";
                        txtOutput.Text = ex.ToString();
                    }
                 break;
            }


        }

        private void frmEncryptString_Load(object sender, EventArgs e)
        {
            Clearfields();
        }

        private void Clearfields()
        {
            txtInput.Text = "";
            txtOutput.Text = "";
            txtPassPhrase.Text = "";
            cboAlgorithm.SelectedIndex = -1;
            lblResult.Text = "";
        }

    }
      }
```

frmEncryptFile.cs

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

namespace SecuringData
{
    public partial class frmEncryptFile : Form
    {
        public frmEncryptFile()
        {
            InitializeComponent();
        }

        private void btnBack_Click(object sender, EventArgs e)
        {
            this.Dispose();
        }

        private void btnEncrypt_Click(object sender, EventArgs e)
        {
            string strInputFile;
            string strDestination;
            string sResult = "";
            string sPassPhrase = "";
            //Validate input file
            if (txtInputFile.Text.Trim().ToString() == "")
            {
                MessageBox.Show("Please select the file you want to
Encrypt", "Error", MessageBoxButtons.OK, MessageBoxIcon.Warning);
                return;
            }
            else
            {
                strInputFile = @txtInputFile.Text.Trim().ToString();
            }
            // Now make sure the file exists.  The user may change the
path manually, in the text box
            if (!File.Exists(strInputFile))
            {
                MessageBox.Show("The file you have selected does not
exists. Please select a valid file.", "Error", MessageBoxButtons.OK,
MessageBoxIcon.Warning);
                return;
            }

            //Validate destination file
            if (txtDestinationFile.Text.Trim().ToString() == "")
            {
                MessageBox.Show("Please select the destination file
where you want to save the encrypted file", "Error",
MessageBoxButtons.OK, MessageBoxIcon.Warning);
                return;
            }
            else
            {
                strDestination =
txtDestinationFile.Text.Trim().ToString();
            }
            //Make sure the directory exists.  The user may have
changed the directory manually

            FileInfo fi = new FileInfo(strDestination);
            DirectoryInfo df = new DirectoryInfo(fi.DirectoryName);
            if (!df.Exists)
            {
                MessageBox.Show("The directory you have selected does
not exists. Please select a valid directory.", "Error",
MessageBoxButtons.OK, MessageBoxIcon.Warning);
                return;
            }
            //Make sure the source and destination are different.  We
don't want to over write the souce file
            if (txtInputFile.Text.Trim().ToString() ==
txtDestinationFile.Text.Trim().ToString())
            {
                MessageBox.Show("Source file and Destination file
cannot be same", "Error", MessageBoxButtons.OK,
MessageBoxIcon.Warning);
                return;
            }
            //Make sure the user has selected the algorithm
            if (cboAlgorithm.Text.Trim() == "")
            {
                MessageBox.Show("Please select the algorithm that you
want to use for Encrypting", "Error", MessageBoxButtons.OK,
MessageBoxIcon.Warning);
                return;
            }
            //Make sure the user has entered the passphrase
            if (txtPassPhrase.Text.Trim().ToString() == "")
            {
                MessageBox.Show("Please enter the passphrase for
Encrypting", "Error", MessageBoxButtons.OK, MessageBoxIcon.Warning);
                return;
            }
            //Now Encrypt the file
            sPassPhrase = txtPassPhrase.Text.Trim();
            lblResult.Text = "Encrypting file...";
            switch (cboAlgorithm.Text)
            {
                case "DES":
                    try
                    {
                        sResult =
DSUComp.EncryptFileUsingDESCrypto(strInputFile, strDestination,
sPassPhrase);
                        if (sResult != "ERROR")
                        {
                            lblResult.Text = "Encrypted successfully";
                        }
                        else
                        {
                            lblResult.Text = "Error while encrypting
the file";
                        }
                    }
                    catch
                    {
                        lblResult.Text = "Error while encrypting the
file";
                    }
                    break;

                case "TripleDES":
                    try
                    {
                        sResult =
DSUComp.EncryptFileUsingTripleDES(strInputFile, strDestination,
sPassPhrase);
                        if (sResult != "ERROR")
                        {
                            lblResult.Text = "Encrypted successfully";
                        }
                        else
                        {
                            lblResult.Text = "Error while encrypting
the file";
                        }
                    }
                    catch
                    {
                        lblResult.Text = "Error while encrypting the
file";
                    }
                    break;
                case "Rijandel":
                    try
                    {
                        sResult =
DSUComp.EncryptFileUsingRijandel(strInputFile, strDestination,
sPassPhrase);
                        if (sResult != "ERROR")
                    {
                        lblResult.Text = "Encrypted successfully";
                    }
                    else
                    {
                        lblResult.Text = "Error while encrypting
the file";
                    }
                }
                catch
                {
                    lblResult.Text = "Error while encrypting the
file";
                }
                break;
            }
        }

        private void chkMaskPassphrase_CheckedChanged(object sender,
EventArgs e)
        {
            if (chkMaskPassphrase.Checked)
            {
                txtPassPhrase.Multiline = false;
                txtPassPhrase.PasswordChar = '*';
            }
            else
            {
                txtPassPhrase.Multiline = true;
            }
        }

        private void btnClear_Click(object sender, EventArgs e)
        {
            txtInputFile.Text = "";
            txtDestinationFile.Text = "";
            txtPassPhrase.Text = "";
            cboAlgorithm.SelectedIndex = -1;
        }

        private void btnSource_Click(object sender, EventArgs e)
        {
            OpenFileDialog ofd = new OpenFileDialog();
            if (ofd.ShowDialog() == DialogResult.OK)
            {
                txtInputFile.Text = ofd.FileName;
            }
            ofd.Dispose();
        }

        private void btnDestination_Click(object sender, EventArgs e)
        {
            OpenFileDialog ofd = new OpenFileDialog();
            if (ofd.ShowDialog() == DialogResult.OK)
            {
                txtDestinationFile.Text = ofd.FileName;
                }
                ofd.Dispose();
            }
        }
    }
```

frmDecryptString.cs

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

namespace SecuringData
{
    public partial class frmDecryptString : Form
    {
        public frmDecryptString()
        {
            InitializeComponent();
        }

        private void btnBack_Click(object sender, EventArgs e)
        {
            this.Dispose();
        }

        private void frmDecryptString_Load(object sender, EventArgs e)
        {
            Clearfields();
        }

        private void chkMaskPassphrase_CheckedChanged(object sender,
EventArgs e)
        {
            if (chkMaskPassphrase.Checked)
            {
                txtPassPhrase.Multiline = false;
                txtPassPhrase.PasswordChar = '*';
            }
            else
            {
                txtPassPhrase.Multiline = true;
            }
        }

        private void btnClear_Click(object sender, EventArgs e)
        {
            Clearfields();
        }

        private void btnDecrypt_Click(object sender, EventArgs e)
        {
            string sInputString = "";
            string sResult = "";
            string sPassPhrase = "";
            lblResult.ForeColor = Color.Black;
            //Validate input
            if (txtInput.Text.Trim().ToString() == "")
            {
                MessageBox.Show("Please Enter the String you want to
Decrypt", "Error", MessageBoxButtons.OK, MessageBoxIcon.Warning);
                return;
            }

            //Make sure the user has selected the algorithm
            if (cboAlgorithm.Text.Trim() == "")
            {
                MessageBox.Show("Please select the algorithm that you
have used while Encrypting", "Error", MessageBoxButtons.OK,
MessageBoxIcon.Warning);
                return;
            }
            //Make sure the user has entered the passphrase
            if (txtPassPhrase.Text.Trim().ToString() == "")
            {
                MessageBox.Show("Please enter the passphrase that you
have provided while Encrypting", "Error", MessageBoxButtons.OK,
MessageBoxIcon.Warning);
                return;
            }

            //Now Decrypt the string
            sInputString = txtInput.Text.Trim();
            sPassPhrase = txtPassPhrase.Text.Trim();
            lblResult.Text = "Decrypting text...";
            switch (cboAlgorithm.Text)
            {
                case "DES":
                    try
                    {
                        sResult =
DSUComp.DecryptUsingDESCrypto(sInputString, sPassPhrase);
                        if (sResult != "ERROR")
                        {
                            lblResult.ForeColor = Color.Red;
                            lblResult.Text = "Decrypted successfully";
                            txtOutput.Text = sResult;
                        }
                        else
                        {
                            lblResult.ForeColor = Color.Red;
                            lblResult.Text = "Error while decrypting
the text. Please select the same alogithm and passphrase that you used
to encrypt the string";
                            txtOutput.Text = "";
                        }
                    }
                    catch (Exception ex)
                    {
                            lblResult.ForeColor = Color.Red;
                            lblResult.Text = "Error while decrypting the
text";
                            txtOutput.Text = ex.ToString();
                        }
                        break;

                    case "TripleDES":
                        try
                        {
                            sResult =
DSUComp.DecryptUsingTripleDES(sInputString, sPassPhrase);
                            if (sResult != "ERROR")
                            {
                                lblResult.ForeColor = Color.Black;
                                lblResult.Text = "Decrypted successfully";
                                txtOutput.Text = sResult;
                            }
                            else
                            {
                                lblResult.ForeColor = Color.Red;
                                lblResult.Text = "Error while decrypting
the text. Please select the same alogithm and passphrase that you used
to encrypt the string";
                                txtOutput.Text = "";
                            }
                        }
                        catch (Exception ex)
                        {
                            lblResult.Text = ex.ToString();
                        }
                        break;
                    case "Rijandel":
                        try
                        {
                            sResult =
DSUComp.DecryptUsingRijandel(sInputString, sPassPhrase);
                            if (sResult != "ERROR")
                            {
                                lblResult.ForeColor = Color.Black;
                                lblResult.Text = "Decrypted successfully";
                                txtOutput.Text = sResult;
                            }
                            else
                            {
                                lblResult.ForeColor = Color.Red;
                                lblResult.Text = "Error while decrypting
the text. Please select the same alogithm and passphrase that you used
to encrypt the string";
                                txtOutput.Text = "";
                            }
                        }
                        catch (Exception ex)
                        {
                            lblResult.ForeColor = Color.Red;
                            lblResult.Text = "Error while decrypting the
text";
```

```
                        txtOutput.Text = ex.ToString();
                    }
                    break;
            }
        }

        private void Clearfields()
        {
            txtInput.Text = "";
            txtOutput.Text = "";
            txtPassPhrase.Text = "";
            cboAlgorithm.SelectedIndex = -1;
            lblResult.Text = "";
        }
    }
  }
```

frmDecryptFile.cs

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


namespace SecuringData
{
    public partial class frmDecryptFile : Form
    {

        public frmDecryptFile()
        {
            InitializeComponent();
        }

        private void btnBack_Click(object sender, EventArgs e)
        {
            this.Dispose();
        }

        private void frmDecryptFile_Load(object sender, EventArgs e)
        {
            //cboAlgorithm.Items.Add("Rijandel");
        }

        private void btnDecrypt_Click(object sender, EventArgs e)
        {
            string strInputFile;
            string strDestination;
            string sResult = "";
            string sPassPhrase = "";
            //Validate input file
            if (txtInputFile.Text.Trim().ToString() == "")
            {
                MessageBox.Show("Please select the file you want to
decrypt", "Error", MessageBoxButtons.OK, MessageBoxIcon.Warning);
                return;
            }
            else
            {
                strInputFile = @txtInputFile.Text.Trim().ToString();
            }
            // Now make sure the file exists.  The user may change the
path manually, in the text box
            if (!File.Exists(strInputFile))
            {
                MessageBox.Show("The file you have selected does not
exists. Please select a valid file.", "Error", MessageBoxButtons.OK,
MessageBoxIcon.Warning);
                return;
            }

            //Validate destination file
            if (txtDestinationFile.Text.Trim().ToString() == "")
            {
                MessageBox.Show("Please select the destination file
where you want to save the decrypted file", "Error",
MessageBoxButtons.OK, MessageBoxIcon.Warning);
                return;
            }
            else
            {
                strDestination =
txtDestinationFile.Text.Trim().ToString();
            }
            //Make sure the directory exists.  The user may have
changed the directory manually

            FileInfo fi = new FileInfo(strDestination);
            DirectoryInfo df = new DirectoryInfo(fi.DirectoryName);
            if (!df.Exists)
            {
                MessageBox.Show("The directory you have selected does
not exists. Please select a valid directory.", "Error",
MessageBoxButtons.OK, MessageBoxIcon.Warning);
                return;
            }
            //Make sure the source and destination are different.  We
don't want to over write the souce file
            if (txtInputFile.Text.Trim().ToString() ==
txtDestinationFile.Text.Trim().ToString())
            {
                MessageBox.Show("Source file and Destination file
cannot be same", "Error", MessageBoxButtons.OK,
MessageBoxIcon.Warning);
                return;
            }
            //Make sure the user has selected the algorithm
            if (cboAlgorithm.Text.Trim()  == "")
            {
                MessageBox.Show("Please select the algorithm that you
have used while Encrypting", "Error", MessageBoxButtons.OK,
MessageBoxIcon.Warning);
                return;
            }
            //Make sure the user has entered the passphrase
            if (txtPassPhrase.Text.Trim().ToString() == "")
            {
                MessageBox.Show("Please enter the passphrase that you
have provided while Encrypting", "Error", MessageBoxButtons.OK,
MessageBoxIcon.Warning);
                return;
            }

            //Now Decrypt the file
            sPassPhrase = txtPassPhrase.Text.Trim();
            lblResult.Text = "Encrypting file...";
            switch (cboAlgorithm.Text)
            {
                case "DES":
                    try
                    {
                        sResult =
DSUComp.DecryptFileUsingDESCrypto(strInputFile, strDestination,
sPassPhrase);
                        if (sResult != "ERROR")
                        {
                            lblResult.Text = "Decrypted successfully";
                        }
                        else
                        {
                            lblResult.Text = "Error while Decryptring
the file";
                        }
                    }
                    catch
                    {
                        lblResult.Text = "Error while Decryptring the
file";
                    }
                    break;

                case "TripleDES":
                    try
                    {
                        sResult =
DSUComp.DecryptFileUsingTripleDES(strInputFile, strDestination,
sPassPhrase);
                        if (sResult != "ERROR")
                        {
                            lblResult.Text = "Decrypted successfully";
                        }
                        else
                        {
                            lblResult.Text = "Error while Decryptring
the file";
```

```
                            }
                        }
                        catch
                        {
                            lblResult.Text = "Error while Decryptring the
file";
                        }
                        break;
                    case "Rijandel":
                        try
                        {
                            sResult =
DSUComp.DecryptFileUsingRijandel(strInputFile, strDestination,
sPassPhrase);
                            if (sResult != "ERROR")
                            {
                                lblResult.Text = "Decrypted successfully";
                            }
                            else
                            {
                                lblResult.Text = "Error while Decryptring
the file";
                            }
                        }
                        catch
                        {
                            lblResult.Text = "Error while Decryptring the
file";
                        }
                        break;
                }
        }

        private void chkMaskPassphrase_CheckedChanged(object sender,
EventArgs e)
        {
            if (chkMaskPassphrase.Checked)
            {
                txtPassPhrase.Multiline = false;
                txtPassPhrase.PasswordChar = '*';
            }
            else
            {
                txtPassPhrase.Multiline = true;
            }

        }

        private void btnSource_Click(object sender, EventArgs e)
        {
            OpenFileDialog ofd = new OpenFileDialog();
            if (ofd.ShowDialog() == DialogResult.OK)
            {
                txtInputFile.Text = ofd.FileName;
            }
            ofd.Dispose();
        }
        private void btnDestination_Click(object sender, EventArgs e)
        {
            OpenFileDialog ofd = new OpenFileDialog();
            if (ofd.ShowDialog() == DialogResult.OK)
            {
                txtDestinationFile.Text = ofd.FileName;

            }
            ofd.Dispose();
        }

        private void btnClear_Click(object sender, EventArgs e)
        {
            txtInputFile.Text = "";
            txtDestinationFile.Text = "";
            txtPassPhrase.Text = "";
            cboAlgorithm.SelectedIndex = -1;
        }
    }
      }
```

frmUserManagement.cs

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


namespace SecuringData
{
    public partial class frmUserManagement : Form
    {
        string strSqlConn;
        SqlConnection objConn;
        private frmUpdateUser frmUpdateUser;
        string strUserID;

        public frmUserManagement()
        {
            InitializeComponent();
        }

        private void btnBack_Click(object sender, EventArgs e)
        {
            this.Dispose();
        }

        private void frmUserManagement_Load(object sender, EventArgs e)
        {
            DisplayUsers();
        }



        private bool OpenDatabase()
        {
            try
            {
                strSqlConn =
ConfigurationSettings.AppSettings["conectionstring"].ToString();
                objConn = new SqlConnection(strSqlConn);
                objConn.Open();
                return true;
            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.ToString());
                return false;

            }
        }

        private void Closedatabase()
            {
                  try
                  {
                        if (objConn.State.ToString() == "Open")
                        {
                              objConn.Close();
                              objConn = null;
                        }
                  }
                  catch
                  {
                  }
            }

        private void DisplayUsers()
        {
            if (OpenDatabase())
            {
                try
                {
                    DataTable dt = new DataTable();
                    SqlDataAdapter da = new SqlDataAdapter("Select
userid as UserID, userfirstname as FirstName, userlastname as LastName,
loginname as LoginName, userpermission as Permission from tblUsers",
objConn);
                    da.Fill(dt);
                    this.dataGridView1.DataSource = dt;
                    this.dataGridView1.SelectionMode =
DataGridViewSelectionMode.FullRowSelect;
                    this.dataGridView1.MultiSelect = false;
                    dataGridView1.Refresh();
                }
                catch (Exception ex)
                {
                    MessageBox.Show(ex.ToString(), "Error",
MessageBoxButtons.OK, MessageBoxIcon.Warning);

                }
                finally
                {
                     Closedatabase();
                }
            }
        }

        private void dataGridView1_RowEnter(object sender,
DataGridViewCellEventArgs e)
        {
            strUserID =
dataGridView1.Rows[e.RowIndex].Cells[0].Value.ToString();

        }

         private void btnCreate_Click(object sender, EventArgs e)
        {
            string strOpMode = "A";
            this.frmUpdateUser = new frmUpdateUser(strOpMode, "");
            frmUpdateUser.MdiParent = MdiParent;
            frmUpdateUser.Show();
            this.Dispose();
        }

        private void btnUpdateUser_Click(object sender, EventArgs e)
        {
            string strOpMode = "E";
            this.frmUpdateUser = new frmUpdateUser(strOpMode,
strUserID);
            frmUpdateUser.MdiParent = MdiParent;
            frmUpdateUser.Show();
            this.Dispose();
        }


    }
      }
```

frmUpdateUser.cs

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


namespace SecuringData
{
    public partial class frmUpdateUser : Form
    {
        private string strOpMode;
        private string strUserValue;
        string strSqlConn;
        SqlConnection objConn;

        public frmUpdateUser(string sMode, string sUserID)
        {
            InitializeComponent();
            OpMode = sMode;
            UserToBeEdited = sUserID;
        }

        public string OpMode
        {
            set
            {
                strOpMode = value;
            }

        }

        public string UserToBeEdited
        {
            set
            {
                // This could be setting a field variable, or in this
case
                // it is setting the Text property of a control
directly.
                strUserValue = value;
            }
        }

        private void frmUpdateUser_Load(object sender, EventArgs e)
        {
            if (this.strOpMode == "A")
            {
                lblUserID.Text = "";
                btnDelete.Enabled = false ;
                btnUpdate.Enabled = false;
                btnAdd.Enabled = true;
                txtUserName.Enabled = true;
            }
            else if (this.strOpMode == "E")
            {
                lblUserID.Text = strUserValue;
                DisplayUserInfo();
                btnDelete.Enabled = true;
                btnUpdate.Enabled = true;
                btnAdd.Enabled = false;
                txtUserName.Enabled = false;
            }
        }
        private void DisplayUserInfo()
        {
            OpenDatabase();
            try
            {
                DataSet dsUser = new DataSet();
                SqlDataAdapter objCmd;
                string sPermission = "";
                string sql = "Select userfirstname , userlastname,
loginname, password,userpermission from tblUsers where userid = " +
strUserValue;
                objCmd = new SqlDataAdapter(sql, objConn);
                objCmd.Fill(dsUser, "UserInfo");
                if (dsUser != null)
                {
                    txtFirstName.Text =
dsUser.Tables["UserInfo"].Rows[0]["userfirstname"].ToString();
                    txtLastName.Text =
dsUser.Tables["UserInfo"].Rows[0]["userlastname"].ToString();
                    txtUserName.Text =
dsUser.Tables["UserInfo"].Rows[0]["loginname"].ToString();
                    txtPassword.Text =
dsUser.Tables["UserInfo"].Rows[0]["password"].ToString();
                    sPermission =
dsUser.Tables["UserInfo"].Rows[0]["userpermission"].ToString();
                    if (sPermission.ToUpper()  == "ADMIN")
                    {
                        cboPermission.SelectedIndex = 0;
                    }
                    else if (sPermission.ToUpper() == "MANAGER")
                    {
                        cboPermission.SelectedIndex = 1;
                    }
                    else if (sPermission.ToUpper()  == "ENCRYPTOR")
                    {
                        cboPermission.SelectedIndex = 2;
                    }
                    else if (sPermission.ToUpper()  == "DECRYPTOR")
                    {
                        cboPermission.SelectedIndex = 3;
                    }

                }
            }
            catch
            {
            }
            finally
            {
                Closedatabase();
            }
        }
        private void btnBack_Click(object sender, EventArgs e)
        {
            this.Dispose();
        }

        private void btnAdd_Click(object sender, EventArgs e)
        {
            int iResult = 1;
            if (ValidateForm())
            {
                //process data
                OpenDatabase();
                try
                {
                    SqlCommand MyCommand = new
SqlCommand("sp_create_user", objConn);
                    MyCommand.CommandType =
CommandType.StoredProcedure;
                    SqlParameter objParam = new SqlParameter();

                    objParam = MyCommand.Parameters.Add("@sFirstName",
SqlDbType.VarChar, 50);
                    objParam.Direction = ParameterDirection.Input;
                    objParam.Value = txtFirstName.Text.Trim();

                    objParam = MyCommand.Parameters.Add("@sLastName",
SqlDbType.VarChar, 50);
                    objParam.Direction = ParameterDirection.Input;
                    objParam.Value = txtLastName.Text.Trim();

                    objParam = MyCommand.Parameters.Add("@sLoginName",
SqlDbType.VarChar, 50);
                    objParam.Direction = ParameterDirection.Input;
                    objParam.Value = txtUserName.Text.Trim();

                    objParam = MyCommand.Parameters.Add("@sPassword",
SqlDbType.VarChar, 50);
                    objParam.Direction = ParameterDirection.Input;
                    objParam.Value = txtPassword.Text;

                    objParam = MyCommand.Parameters.Add("@permission",
SqlDbType.VarChar, 25);
                    objParam.Direction = ParameterDirection.Input;
                    objParam.Value = cboPermission.Text;

                    objParam = MyCommand.Parameters.Add("@iResult",
SqlDbType.Int);
                    objParam.Direction = ParameterDirection.Output;

                    MyCommand.ExecuteScalar();
                    iResult = (int)objParam.Value;

                    if (iResult == 0)
                    {
                        MessageBox.Show("User Added Succesfully",
"Success", MessageBoxButtons.OK, MessageBoxIcon.Information);
```

```
                    }
                    else
                    {
                        MessageBox.Show("User Name Exists. Please
choose another name", "Username Exists", MessageBoxButtons.OK,
MessageBoxIcon.Warning);
                    }

                }
                catch
                {
                    MessageBox.Show("There was an error while trying to
add the user.", "Add User Failed", MessageBoxButtons.OK,
MessageBoxIcon.Warning);
                }
                finally
                {
                    Closedatabase();
                }

            }
        }
        private bool ValidateForm()
        {
            if (txtFirstName.Text.Trim() == "")
            {
                MessageBox.Show("First name cannot be empty", "Invalid
data", MessageBoxButtons.OK, MessageBoxIcon.Warning);
                return false;
            }
            if (txtLastName.Text.Trim() == "")
            {
                MessageBox.Show("Last name cannot be empty", "Invalid
data", MessageBoxButtons.OK, MessageBoxIcon.Warning);
                return false;
            }
            if (txtUserName.Text.Trim() == "")
            {
                MessageBox.Show("User name cannot be empty", "Invalid
data", MessageBoxButtons.OK, MessageBoxIcon.Warning);
                return false;
            }
            if (txtPassword.Text.Trim() == "")
            {
                MessageBox.Show("Password cannot be empty", "Invalid
data", MessageBoxButtons.OK, MessageBoxIcon.Warning);
                return false;
            }
            if (cboPermission.Text.Trim() == "")
            {
                MessageBox.Show("Please select the permission level for
this user", "Error", MessageBoxButtons.OK, MessageBoxIcon.Warning);
                return false;
            }
            return true;
        }
        private void btnClear_Click(object sender, EventArgs e)
        {
            lblUserID.Text = "";
            txtFirstName.Text = "";
            txtLastName.Text = "";
            txtUserName.Text = "";
            txtPassword.Text = "";
            btnAdd.Enabled = true;
            cboPermission.SelectedIndex = -1;
        }
        private bool OpenDatabase()
        {
            try
            {
                strSqlConn =
ConfigurationSettings.AppSettings["conectionstring"].ToString();
                objConn = new SqlConnection(strSqlConn);
                objConn.Open();
                return true;
            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.ToString());
                return false;

            }
        }

        private void Closedatabase()
        {
            try
            {
                if (objConn.State.ToString() == "Open")
                {
                    objConn.Close();
                    objConn = null;
                }
            }
            catch
            {
            }
        }

        private void btnUpdate_Click(object sender, EventArgs e)
        {

            if (ValidateForm())
            {
                UpdateUser("E");


            }
        }

        private void UpdateUser(string sMode)
        {
            int iResult = 1;
            OpenDatabase();
            try
            {
                SqlCommand MyCommand = new SqlCommand("sp_edit_user",
objConn);
                MyCommand.CommandType = CommandType.StoredProcedure;
                SqlParameter objParam = new SqlParameter();

                objParam = MyCommand.Parameters.Add("@iUserid",
SqlDbType.Int, 4);
                objParam.Direction = ParameterDirection.Input;
                objParam.Value =
Convert.ToInt32(lblUserID.Text.ToString());

                objParam = MyCommand.Parameters.Add("@sFirstName",
SqlDbType.VarChar, 50);
                objParam.Direction = ParameterDirection.Input;
                objParam.Value = txtFirstName.Text.Trim();

                objParam = MyCommand.Parameters.Add("@sLastName",
SqlDbType.VarChar, 50);
                objParam.Direction = ParameterDirection.Input;
                objParam.Value = txtLastName.Text.Trim();

                objParam = MyCommand.Parameters.Add("@sPassword",
SqlDbType.VarChar, 50);
                objParam.Direction = ParameterDirection.Input;
                objParam.Value = txtPassword.Text;

                objParam = MyCommand.Parameters.Add("@permission",
SqlDbType.VarChar, 25);
                objParam.Direction = ParameterDirection.Input;
                objParam.Value = cboPermission.Text;

                objParam = MyCommand.Parameters.Add("@mode",
SqlDbType.Char, 1);
                objParam.Direction = ParameterDirection.Input;
                objParam.Value = sMode;

                MyCommand.ExecuteScalar();

                if (sMode == "E")
                {
                MessageBox.Show("User Updated Succesfully", "Success",
MessageBoxButtons.OK, MessageBoxIcon.Information);
                }
                else if (sMode == "D")
                {
                    MessageBox.Show("User Deleted Succesfully",
"Success", MessageBoxButtons.OK, MessageBoxIcon.Information);
                }

            }
            catch (Exception ex)
            {
                //MessageBox.Show("There was an error while trying to
edit the user.", "Add User Failed", MessageBoxButtons.OK,
MessageBoxIcon.Warning);
                MessageBox.Show(ex.ToString());
            }
            finally
            {
                Closedatabase();
            }
        }

        private void btnDelete_Click(object sender, EventArgs e)
        {
            UpdateUser("D");
        }
    }
      }
```

Data Link Layer

```
IF  EXISTS (SELECT * FROM dbo.sysobjects WHERE id =
 OBJECT_ID(N'[dbo].[sp_create_user]') AND
OBJECTPROPERTY(id,N'IsProcedure') = 1)
DROP PROCEDURE [dbo].[sp_create_user]
GO
IF  EXISTS (SELECT * FROM dbo.sysobjects WHERE id =
 OBJECT_ID(N'[dbo].[sp_validate_user]') AND
OBJECTPROPERTY(id,N'IsProcedure') = 1)
DROP PROCEDURE [dbo].[sp_validate_user]
GO
IF  EXISTS (SELECT * FROM dbo.sysobjects WHERE id =
 OBJECT_ID(N'[dbo].[sp_edit_user]') AND
OBJECTPROPERTY(id,N'IsProcedure') = 1)
DROP PROCEDURE [dbo].[sp_edit_user]
```

```
GO

IF  EXISTS (SELECT * FROM dbo.sysobjects WHERE id =
 OBJECT_ID(N'[dbo].[tblUsers]') AND OBJECTPROPERTY(id, N'IsUserTable')
= 1)

DROP TABLE [dbo].[tblUsers]

GO

IF  EXISTS (SELECT * FROM dbo.sysobjects WHERE id =
 OBJECT_ID(N'[dbo].[tblUserActivity]') AND OBJECTPROPERTY(id,
N'IsUserTable') = 1)

DROP TABLE [dbo].[tblUserActivity]



SET ANSI_NULLS ON

GO

SET QUOTED_IDENTIFIER ON

GO

CREATE TABLE [dbo].[tblUsers](

        [userid] [int] IDENTITY(1,1) NOT NULL,

        [userfirstname] [varchar](50) NULL,

        [userlastname] [varchar](50) NULL,

        [loginname] [varchar](50) NULL,

        [password] [varchar](50) NULL,

        [userpermission] [varchar](25) NULL,
        [datecreated] [datetime] NULL CONSTRAINT
[DF_tblUsers_datecreated]
  DEFAULT (getdate())
) ON [PRIMARY]

GO
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[tblUserActivity](
        [userid] [int] NULL,
        [logindate] [datetime] NULL CONSTRAINT
[DF_tblUserActivity_logindate]
  DEFAULT (getdate()),
        [logoutdate] [datetime] NULL
) ON [PRIMARY]

GO
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
```

```
CREATE Procedure [dbo].[sp_edit_user]
@iUserid int,
@sFirstName varchar(50),
@sLastName varchar(50),
@sPassword varchar(50),
@permission varchar(25),
@mode char(1)
as

Begin
    if @mode = 'E'
    Begin
        Update tblUsers set
        userfirstname = @sFirstName,
        userlastname = @sLastName,
        password = @sPassword,
        userpermission = @permission
        where userid = @iUserid
    End
    else if @mode = 'D'
    Begin
        Delete from tblUsers where userid = @iUserid
```

```
          End

End


GO

SET ANSI_NULLS ON

GO

SET QUOTED_IDENTIFIER ON

GO


CREATE Procedure [dbo].[sp_create_user]

@sFirstName varchar(50),

@sLastName varchar(50),

@sLoginName varchar(50),

@sPassword varchar(50),

@permission varchar(25),

@iResult int out

as


Begin


          If EXISTS (Select userid from tblUsers where loginname =

@sLoginName)

                    Begin --Login Name exists
```

```
                        Set @iResult = 1

                        Return @iResult

                End

        Else

                Begin

                        Insert into tblUsers (userfirstname, userlastname,

loginname, password,userpermission )

                        values (@sFirstName,@sLastName,

        @sLoginName,@sPassword,@permission )

                        Set @iResult = 0

                        Return @iResult

                End

End


GO

SET ANSI_NULLS ON

GO

SET QUOTED_IDENTIFIER ON

GO

CREATE Procedure [dbo].[sp_validate_user]

@sLoginName varchar(50),

@sPassword varchar(50),

@iResult int out
```

```
as


Begin

        Declare @iUserId int

        Set @iUserId = 0

        Select @iUserId = userid from tblUsers where loginname =

@sLoginName

 and password = @sPassword

        If @iUserId <> 0

                Begin --Valid username and password

                        Insert into tblUserActivity (userid) values (@iUserId)

                        Set @iResult = 0

                        Return @iResult

                End

        Else

                Begin

                        Set @iResult = 1

                        Return @iResult

                End

End


SET QUOTED_IDENTIFIER ON

GO
```

```
Insert into tblUsers (
 userfirstname,userlastname,loginname,password,userpermission) values
('Admin', 'Admin','admin','dsu','Admin')

SET ANSI_NULLS ON
GO
```