**Chapter 4**

**4.1 Splash Screen:**

Live working Splash Screen for the application, it can show:

* Total user
* Identify user as Administrator
* Database connectivity



Fig 4.1: Splash Screen

**Coding:**

namespace Data\_and\_PC\_Securer

{

public partial class Splash : Form

{

int t = 0;

public Splash()

{

InitializeComponent();

timer1.Interval = 3000;

timer1.Start();

timer2.Interval = 300;

timer2.Start();

}

private void timer1\_Tick(object sender, EventArgs e)

{

timer1.Stop();

timer2.Stop();

if (t == 4)

{

this.DialogResult = DialogResult.No;

}

else

{

this.DialogResult = DialogResult.OK;

this.Close();

}

}

private void timer2\_Tick(object sender, EventArgs e)

{

if (t == 0)

{

label3.Text = "Initializing .";

t++;

SqlConnection con = new SqlConnection("Data Source=SHREYKUMARJAIN\\SHREYKUMARJAIN; Initial Catalog=DataSecurer; Integrated Security=TRUE");

con.Open();

label4.Text = "Database " + con.State.ToString();

con.Close();

}

else if (t == 1)

{

label3.Text = "Initializing . .";

t++;

}

else if (t == 2)

{

label3.Text = "Initializing . . .";

WindowsIdentity identity = WindowsIdentity.GetCurrent();

WindowsPrincipal principal = new WindowsPrincipal(identity);

if (principal.IsInRole(WindowsBuiltInRole.Administrator) == true)

{

label4.Text = "User : Admin";

}

else

{

label4.Text = "User : General";

}

t++;

}

else if (t == 3)

{

label3.Text = "Initializing . . .";

t = 0;

SqlConnection con = new SqlConnection("Data Source=SHREYKUMARJAIN\\SHREYKUMARJAIN; Initial Catalog=DataSecurer; Integrated Security=TRUE");

SqlCommand cmd;

con.Open();

SqlDataReader rd;

cmd = new SqlCommand("select count(\*) from passing ", con);

rd = cmd.ExecuteReader();

if (rd.Read() == true)

{

if (Convert.ToInt32(rd[0]) <= 0)

{

label4.Text = "No User Found";

t = 4;

}

else

{

label4.Text = "User Found" + Convert.ToInt32(rd[0]);

}

}}}}

}

**4.2 Password:**

Password module is used to verify the user for his key management and also for saving the best output folder that he wants. Also it provides only administrator of the computer the right to add or remove any user, also can provide new password for any user in case he/she forgets password.

Password module also having links to:

* Help
* About

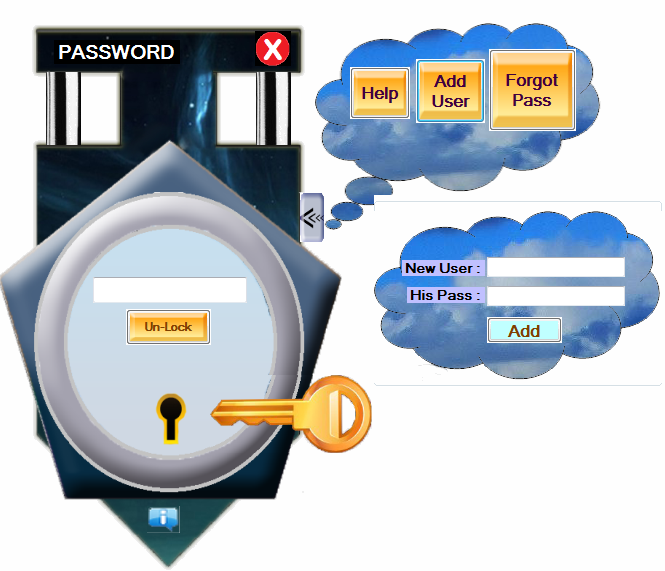
****

Fig 4.2: Password Module

**Coding:**

namespace Data\_and\_PC\_Securer

{

public partial class Password : Form

{

string name;

public Password()

{

InitializeComponent();

}

private void unlock\_Click(object sender, EventArgs e)

{

if (textBox1.Text == null)

{

MessageBox.Show("Enter Password First");

textBox1.Focus();

}

else

{

if (name != null)

{

try

{

SqlConnection con = new SqlConnection("Data Source=SHREYKUMARJAIN\\SHREYKUMARJAIN; Initial Catalog=DataSecurer; Integrated Security=TRUE");

SqlCommand cmd;

con.Open();

SqlDataReader rd;

cmd = new SqlCommand("select \* from passing where uname='" + name + "'", con);

rd = cmd.ExecuteReader();

if (rd.Read() == true)

{

if (textBox1.Text == rd[1].ToString())

{

con.Close();

try

{

SqlCommand cmd1;

con.Open();

cmd1 = new SqlCommand("insert into online values('" + name + "')", con);

int tempo = cmd.ExecuteNonQuery();

if (tempo > 0)

{

}

else

{

MessageBox.Show("Must Work Offline");

}

con.Close();

}

catch (Exception e2)

{

MessageBox.Show(e2.ToString());

}

timer1.Start();

timer1.Interval = 50;

}

else

{

MessageBox.Show("Invalid Password");

textBox1.Clear();

textBox1.Focus();

}

}

}

catch (Exception e1)

{

MessageBox.Show(e1.ToString());

}

}

else

{

Verify\_Name vn = new Verify\_Name();

vn.Show();

vn.get\_pass(textBox1.Text);

this.Hide();

}

}

}

private void timer1\_Tick(object sender, EventArgs e)

{

Size tempsize = pictureBox2.Size;

tempsize.Height--;

pictureBox2.Size = tempsize;

pictureBox3.Size = tempsize;

if (tempsize.Height < 0)

{

timer1.Stop();

Data\_and\_PC\_Securer dp = new Data\_and\_PC\_Securer();

this.Hide();

dp.Show();

}

}

private void pictureBox5\_Click(object sender, EventArgs e)

{

Application.Exit();

}

private void pictureBox6\_Click(object sender, EventArgs e)

{

if (pictureBox7.Visible == false)

{

pictureBox6.ImageLocation = @"E:\Main Project\Data and PC Securer\Data and PC Securer\Resources\SlArrow2.png";

timer2.Start();

timer2.Interval = 1;

}

else

{

pictureBox6.ImageLocation = @"E:\Main Project\Data and PC Securer\Data and PC Securer\Resources\SlArrow.png";

button1.Visible = false;

button2.Visible = false;

button3.Visible = false;

pictureBox7.Visible = false;

groupBox1.Visible = false;

}

}

private void timer2\_Tick(object sender, EventArgs e)

{

Size tempsize = pictureBox7.Size;

tempsize.Width++;

pictureBox7.Visible = true;

pictureBox7.Size = tempsize;

if (tempsize.Width >= 286)

{

timer2.Stop();

button1.Visible = true;

button2.Visible = true;

button3.Visible = true;

}

}

private void button1\_Click(object sender, EventArgs e)

{

WindowsIdentity identity = WindowsIdentity.GetCurrent();

WindowsPrincipal principal = new WindowsPrincipal(identity);

if (principal.IsInRole(WindowsBuiltInRole.Administrator) == true)

{

if (groupBox1.Visible == false)

{

groupBox1.Visible = true;

}

}

else

{

MessageBox.Show("Only Admin of this Computer can Add User");

}

}

private void pictureBox9\_Click(object sender, EventArgs e)

{

About ab = new About();

ab.Show();

}

private void button4\_Click(object sender, EventArgs e)

{

if (textBox2.Text != "" | textBox3.Text != "")

{

try

{

SqlConnection con = new SqlConnection("Data Source=SHREYKUMARJAIN\\SHREYKUMARJAIN; Initial Catalog=DataSecurer; Integrated Security=TRUE");

SqlCommand cmd;

con.Open();

cmd = new SqlCommand("insert into passing values('" + textBox2.Text + "','" + textBox3.Text + "')", con);

int tempo = cmd.ExecuteNonQuery();

if (tempo > 0)

{

MessageBox.Show("User Added");

}

else

{

MessageBox.Show("Unknown error during Adding User");

}

}

catch (Exception e7)

{

MessageBox.Show(e7.ToString());

}

}

}

}

}

**4.3 Settings:**

Settings provide the Administrator to edit profile of any user and also provide any user to personalize settings like:

* New user name
* New password
* New key
* Output folder

Fig 4.3: User Settings page

**Coding:**

namespace Data\_and\_PC\_Securer

{

public partial class Settings : Form

{

public Settings()

{

InitializeComponent();

}

private void checkBox1\_CheckedChanged(object sender, EventArgs e)

{

if (checkBox1.Checked == false)

{

label4.Visible = true;

textBox4.Visible = true;

}

else

{

label4.Visible = false;

textBox4.Visible = false;

}

}

private void Settings\_Load(object sender, EventArgs e)

{

try

{

SqlConnection con = new SqlConnection("Data Source=SHREYKUMARJAIN\\SHREYKUMARJAIN; Initial Catalog=DataSecurer; Integrated Security=TRUE");

SqlCommand cmd;

con.Open();

SqlDataReader rd;

cmd = new SqlCommand("select \* from online", con);

rd = cmd.ExecuteReader();

if (rd.Read() == true)

{

textBox1.Text = rd[0].ToString();

}

}

catch (Exception e3)

{

MessageBox.Show(e3.ToString());

}

try

{

SqlConnection con = new SqlConnection("Data Source=SHREYKUMARJAIN\\SHREYKUMARJAIN; Initial Catalog=DataSecurer; Integrated Security=TRUE");

SqlCommand cmd;

con.Open();

SqlDataReader rd;

cmd = new SqlCommand("select \* from passing where uname='" + textBox1.Text + "'", con);

rd = cmd.ExecuteReader();

if (rd.Read() == true)

{

textBox5.Text = rd[2].ToString();

textBox4.Text = rd[3].ToString();

//command code hided

}

}

catch (Exception e4)

{

MessageBox.Show(e4.ToString());

}

}

private void button1\_Click(object sender, EventArgs e)

{

try

{

SqlConnection con = new SqlConnection("Data Source=SHREYKUMARJAIN\\SHREYKUMARJAIN; Initial Catalog=DataSecurer; Integrated Security=TRUE");

SqlCommand cmd;

con.Open();

SqlDataReader rd;

cmd = new SqlCommand("select \* from passing where uname='" + textBox1.Text + "'", con);

rd = cmd.ExecuteReader();

if (rd.Read() == true)

{

if (textBox2.Text == rd[1].ToString())

{

con.Close();

if (textBox3.Text != null)

{

SqlCommand cmd1;

try

{

con.Open();

cmd1 = new SqlCommand("update passing set password= '" + textBox3.Text + "'where uname= '" + textBox1.Text + "'", con);

int temp = cmd1.ExecuteNonQuery();

if (temp > 0)

{

MessageBox.Show("Password updated");

}

else

{

MessageBox.Show("password not updated");

}

con.Close();

}

catch (Exception e5)

{

MessageBox.Show(e5.ToString());

}

}

}

}

}

catch (Exception e4)

{

MessageBox.Show(e4.ToString());

}

}

private void button3\_Click(object sender, EventArgs e)

{

if (folderBrowserDialog1.ShowDialog() == DialogResult.OK)

{

textBox5.Text = folderBrowserDialog1.SelectedPath;

}

}

private void textBox5\_TextChanged(object sender, EventArgs e)

{

try

{

SqlConnection con = new SqlConnection("Data Source=SHREYKUMARJAIN\\SHREYKUMARJAIN; Initial Catalog=DataSecurer; Integrated Security=TRUE");

SqlCommand cmd;

con.Open();

cmd = new SqlCommand("update passing set outputf= '" + textBox5.Text + "'where uname= '" + textBox1.Text + "'", con);

int temp = cmd.ExecuteNonQuery();

if (temp > 0)

{

MessageBox.Show("Output Folder updated");

}

con.Close();

}

catch (Exception e6)

{}

} } }

**4.4 Data & PC Securer:**

It is the main program or user interaction application window by which user can use any desirable activity or can run any function of the software for his/her use. Such as:

* Encrypt file
* Decrypt file
* Folder compression
* Hide folder
* Intelligent human readable text
* Hide message
* Email encryption
* Wifi sending of encrypted file

Fig 4.4: Data & PC Securer

**Coding:**

namespace Data\_and\_PC\_Securer

{

public partial class Data\_and\_PC\_Securer : Form

{

public Data\_and\_PC\_Securer()

{

InitializeComponent();

}

private void label12\_Click(object sender, EventArgs e)

{

SqlConnection con = new SqlConnection("Data Source=SHREYKUMARJAIN\\SHREYKUMARJAIN; Initial Catalog=DataSecurer; Integrated Security=TRUE");

SqlCommand cmd;

con.Open();

cmd = new SqlCommand("delete from online", con);

try

{

cmd.ExecuteNonQuery();

}

catch (Exception exp)

{

MessageBox.Show(exp.ToString());

}

con.Close();

Password p = new Password();

p.Show();

this.Close();

}

private void pictureBox1\_Click(object sender, EventArgs e)

{

Encrypt\_File ef = new Encrypt\_File();

ef.Show();

}

private void label1\_Click(object sender, EventArgs e)

{

Encrypt\_File ef = new Encrypt\_File();

ef.Show();

}

private void pictureBox3\_Click(object sender, EventArgs e)

{

Decrypt\_file df = new Decrypt\_file();

df.Show();

}

private void label2\_Click(object sender, EventArgs e)

{

Decrypt\_file df = new Decrypt\_file();

df.Show();

}

private void label3\_Click(object sender, EventArgs e)

{

Compression c = new Compression();

c.Show();

}

private void pictureBox4\_Click(object sender, EventArgs e)

{

Hide\_Folder hf = new Hide\_Folder();

hf.Show();

}

private void label4\_Click(object sender, EventArgs e)

{

Hide\_Folder hf = new Hide\_Folder();

hf.Show();

}

private void pictureBox6\_Click(object sender, EventArgs e)

{

Intelligent\_Mind\_Text imt = new Intelligent\_Mind\_Text();

imt.Show();

}

private void label5\_Click(object sender, EventArgs e)

{

Intelligent\_Mind\_Text imt = new Intelligent\_Mind\_Text();

imt.Show();

}

private void pictureBox5\_Click(object sender, EventArgs e)

{

Hide\_Message hm = new Hide\_Message();

hm.Show();

}

private void pictureBox8\_Click(object sender, EventArgs e)

{

WiFi\_Encryption wf = new WiFi\_Encryption();

wf.Show();

}

private void label8\_Click(object sender, EventArgs e)

{

WiFi\_Encryption wf = new WiFi\_Encryption();

wf.Show();

}

private void Data\_and\_PC\_Securer\_Load(object sender, EventArgs e)

{

try

{

SqlConnection con = new SqlConnection("Data Source=SHREYKUMARJAIN\\SHREYKUMARJAIN; Initial Catalog=DataSecurer; Integrated Security=TRUE");

SqlCommand cmd;

con.Open();

SqlDataReader rd;

cmd = new SqlCommand("select \* from online", con);

rd = cmd.ExecuteReader();

if (rd.Read() == true)

{

label10.Text = rd[0].ToString();

}

}

catch (Exception e3)

{

MessageBox.Show(e3.ToString());

}

}

private void label11\_Click(object sender, EventArgs e)

{

Settings s = new Settings();

s.Show();

}

}

}

**4.5 Encrypt File:**

You can encrypt any type of file by this module. User can:

* Take any number of files to encrypt
* Encryption can be done in many ways like:
  + DES
  + AES
  + UNI-CIPHER
* Working:
  + Password or the key as per user can be used as key
  + Inbuilt as well as self developed cryptographic techniques are used

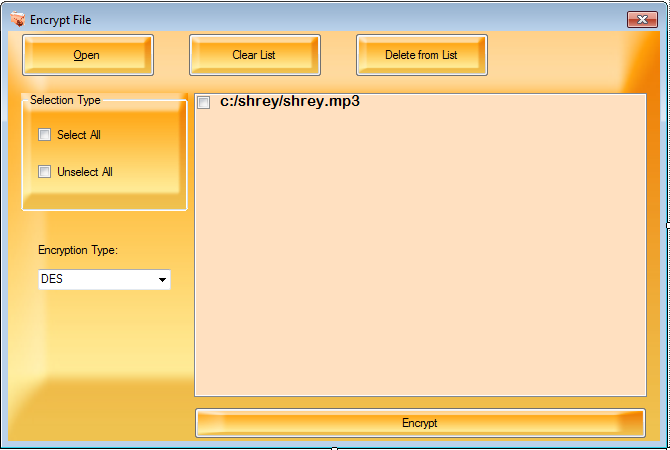


Fig 4.5: Encrypt File

**Coding:**

namespace Data\_and\_PC\_Securer

{

public partial class Encrypt\_File : Form

{

string passkey;

string ofolder;

string Sname;

public Encrypt\_File()

{

InitializeComponent();

//key selection

try

{

SqlConnection con = new SqlConnection("Data Source=SHREYKUMARJAIN\\SHREYKUMARJAIN; Initial Catalog=DataSecurer; Integrated Security=TRUE");

SqlCommand cmd;

con.Open();

SqlDataReader rd;

cmd = new SqlCommand("select \* from online", con);

rd = cmd.ExecuteReader();

if (rd.Read() == true)

{

con.Close();

try

{

SqlConnection con1 = new SqlConnection("Data Source=SHREYKUMARJAIN\\SHREYKUMARJAIN; Initial Catalog=DataSecurer; Integrated Security=TRUE");

SqlCommand cmd1;

con1.Open();

SqlDataReader rd1;

cmd1 = new SqlCommand("select \* from passing where uname='" + rd[0].ToString() + "'", con1);

rd1 = cmd.ExecuteReader();

if (rd1.Read() == true)

{

passkey = rd1[1].ToString();

ofolder = rd1[2].ToString();

if (rd1[3].ToString() != "")

{

passkey = rd1[3].ToString();

}

}

con1.Close();

}

catch (Exception e1)

{

MessageBox.Show(e1.ToString());

}

}

}

catch (Exception e1)

{

MessageBox.Show(e1.ToString());

}

}

private void btnOpen\_Click(object sender, EventArgs e)

{

try

{

openFileDialog1.InitialDirectory = @"c:\";

if (openFileDialog1.ShowDialog() == DialogResult.OK)

{

foreach (string fileName in openFileDialog1.FileNames)

{

mycheckedListBox.Items.Add(fileName, CheckState.Unchecked);

}

}

}

catch (Exception ex)

{

MessageBox.Show(ex.ToString());

}

}

private void btnChange\_Click(object sender, EventArgs e)

{

try

{

mycheckedListBox.Items.Clear();

}

catch (Exception ex)

{

MessageBox.Show(ex.ToString());

}

}

private void selectall\_CheckedChanged(object sender, EventArgs e)

{

try

{

if (selectall.Checked == true)

{

unselectall.Checked = false;

for (int i = 0; i < mycheckedListBox.Items.Count; i++)

{

mycheckedListBox.SetItemChecked(i, true);

}

}

}

catch (Exception ex)

{

MessageBox.Show(ex.ToString());

}

}

private void unselectall\_CheckedChanged(object sender, EventArgs e)

{

try

{

if (unselectall.Checked == true)

{

selectall.Checked = false;

for (int i = 0; i < mycheckedListBox.Items.Count; i++)

{

mycheckedListBox.SetItemChecked(i, false);

}

}

}

catch (Exception ex)

{

MessageBox.Show(ex.ToString());

}

}

private void button2\_Click(object sender, EventArgs e)

{

if (comboBox1.Text == "DES")

{

DES();

}

else if (comboBox1.Text == "AES")

{

AES();

}

else if (comboBox1.Text == "Uni Cipher")

{

Unicipher();

}

}

public void DES()

{

try

{

for (int i = 0; i < mycheckedListBox.Items.Count; i++)

{

if (mycheckedListBox.GetItemChecked(i) == true)

{

try

{

///normal encryption

try

{

UnicodeEncoding UE = new UnicodeEncoding();

byte[] key = UE.GetBytes(passkey);

string cryptFile = mycheckedListBox.Items[i].ToString();

FileStream fsCrypt = new FileStream(cryptFile, FileMode.Create);

RijndaelManaged RMCrypto = new RijndaelManaged();

CryptoStream cs = new CryptoStream(fsCrypt, RMCrypto.CreateEncryptor(key, key), CryptoStreamMode.Write);

FileStream fsIn = new FileStream(ofolder + "\\" + Path.GetFileName(mycheckedListBox.Items[i].ToString()), FileMode.Open);

int data;

while ((data = fsIn.ReadByte()) != -1) cs.WriteByte((byte)data);

fsIn.Close();

cs.Close();

fsCrypt.Close();

}

catch

{

MessageBox.Show("Encryption failed!", "Error");

}

}

catch (Exception e11)

{

MessageBox.Show(e11.ToString());

}

}

}

}

catch (Exception ex)

{

MessageBox.Show("Check a Renaming Style First... See on lect Round Selections");

MessageBox.Show(ex.ToString());

}

}

public void Unicipher()

{

try

{

for (int i = 0; i < mycheckedListBox.Items.Count; i++)

{

if (mycheckedListBox.GetItemChecked(i) == true)

{

try

{

UnicodeEncoding UE = new UnicodeEncoding();

byte[] key = UE.GetBytes(passkey);

string cryptFile = mycheckedListBox.Items[i].ToString();

FileStream fsCrypt = new FileStream(cryptFile, FileMode.Create);

RijndaelManaged RMCrypto = new RijndaelManaged();

CryptoStream cs = new CryptoStream(fsCrypt, RMCrypto.CreateEncryptor(key, key), CryptoStreamMode.Write);

FileStream fsIn = new FileStream(ofolder + "\\" + Path.GetFileName(mycheckedListBox.Items[i].ToString()), FileMode.Open);

Sname = ofolder + "\\" + Path.GetFileName(mycheckedListBox.Items[i].ToString());

int data;

while ((data = fsIn.ReadByte()) != -1) cs.WriteByte((byte)data);

fsIn.Close();

cs.Close();

fsCrypt.Close();

}

catch (Exception e11)

{

MessageBox.Show(e11.ToString());

}

try

{

UnicodeEncoding UE = new UnicodeEncoding();

byte[] key = UE.GetBytes(passkey);

string cryptFile = Sname;

FileStream fsCrypt = new FileStream(cryptFile, FileMode.Create);

RijndaelManaged RMCrypto = new RijndaelManaged();

CryptoStream cs = new CryptoStream(fsCrypt, RMCrypto.CreateEncryptor(key, key), CryptoStreamMode.Write);

FileStream fsIn = new FileStream(Sname, FileMode.Open);

int data;

while ((data = fsIn.ReadByte()) != -1) cs.WriteByte((byte)data);

fsIn.Close();

cs.Close();

fsCrypt.Close();

}

catch (Exception e11)

{

MessageBox.Show(e11.ToString());

}

}

}

}

catch (Exception ex)

{

MessageBox.Show("Check a Renaming Style First... See on lect Round Selections");

MessageBox.Show(ex.ToString());

}

}

public void AES()

{

try

{

for (int i = 0; i < mycheckedListBox.Items.Count; i++)

{

if (mycheckedListBox.GetItemChecked(i) == true)

{

try

{

for (int j = 0; j < 3; j++)

{

UnicodeEncoding UE = new UnicodeEncoding();

byte[] key = UE.GetBytes(passkey);

string cryptFile = mycheckedListBox.Items[i].ToString();

FileStream fsCrypt = new FileStream(cryptFile, FileMode.Create);

RijndaelManaged RMCrypto = new RijndaelManaged();

CryptoStream cs = new CryptoStream(fsCrypt, RMCrypto.CreateEncryptor(key, key), CryptoStreamMode.Write);

FileStream fsIn = new FileStream(ofolder + "\\" + Path.GetFileName(mycheckedListBox.Items[i].ToString()), FileMode.Open);

int data;

while ((data = fsIn.ReadByte()) != -1) cs.WriteByte((byte)data);

fsIn.Close();

cs.Close();

fsCrypt.Close();

}

}

catch (Exception e11)

{

MessageBox.Show(e11.ToString());

}

for (int j = 0; j < 2; j++)

{

try

{

UnicodeEncoding UE = new UnicodeEncoding();

byte[] key = UE.GetBytes(passkey);

string cryptFile = Sname;

FileStream fsCrypt = new FileStream(cryptFile, FileMode.Create);

RijndaelManaged RMCrypto = new RijndaelManaged();

CryptoStream cs = new CryptoStream(fsCrypt, RMCrypto.CreateEncryptor(key, key), CryptoStreamMode.Write);

FileStream fsIn = new FileStream(Sname, FileMode.Open);

int data;

while ((data = fsIn.ReadByte()) != -1) cs.WriteByte((byte)data);

fsIn.Close();

cs.Close();

fsCrypt.Close();

}

catch (Exception e11)

{

MessageBox.Show(e11.ToString());

}

}

}

}

}

catch (Exception ex)

{

MessageBox.Show("Check a Renaming Style First... See on lect Round Selections");

MessageBox.Show(ex.ToString());

}

}

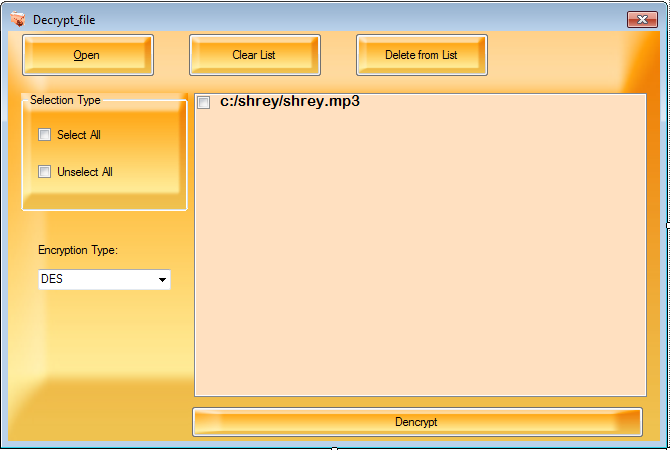
}

}

**4.6 Decrypt Files:**

You can decrypt any type of file by this module. User can:

* Take any number of files to decrypt
* Decryption can be done in many ways like:
  + DES
  + AES
  + UNI-CIPHER
* Working:
  + User can only decrypt the file that is encrypted by him only or case he/she wants to decrypt others file, must need his password
  + Password or the key as per user can be used as key
  + Inbuilt as well as self developed cryptographic techniques are used

Fig 4.6: Decrypt Files

**Coding:**

namespace Data\_and\_PC\_Securer

{

public partial class Decrypt\_file : Form

{

string passkey;

string ofolder;

string Sname;

public Decrypt\_file()

{

InitializeComponent();

//key selection

try

{

SqlConnection con = new SqlConnection("Data Source=SHREYKUMARJAIN\\SHREYKUMARJAIN; Initial Catalog=DataSecurer; Integrated Security=TRUE");

SqlCommand cmd;

con.Open();

SqlDataReader rd;

cmd = new SqlCommand("select \* from online", con);

rd = cmd.ExecuteReader();

if (rd.Read() == true)

{

con.Close();

try

{

SqlConnection con1 = new SqlConnection("Data Source=SHREYKUMARJAIN\\SHREYKUMARJAIN; Initial Catalog=DataSecurer; Integrated Security=TRUE");

SqlCommand cmd1;

con1.Open();

SqlDataReader rd1;

cmd1 = new SqlCommand("select \* from passing where uname='" + rd[0].ToString() + "'", con1);

rd1 = cmd.ExecuteReader();

if (rd1.Read() == true)

{

passkey = rd1[1].ToString();

ofolder = rd1[2].ToString();

if (rd1[3].ToString() != "")

{

passkey = rd1[3].ToString();

}

}

con1.Close();

}

catch (Exception e1)

{

MessageBox.Show(e1.ToString());

}

}

}

catch (Exception e1)

{

MessageBox.Show(e1.ToString());

}

}

private void btnOpen\_Click(object sender, EventArgs e)

{

try

{

openFileDialog1.InitialDirectory = @"c:\";

openFileDialog1.Filter = "Songs|\*.mp3";

if (openFileDialog1.ShowDialog() == DialogResult.OK)

{

foreach (string fileName in openFileDialog1.FileNames)

{

mycheckedListBox.Items.Add(fileName, CheckState.Unchecked);

}

}

}

catch (Exception ex)

{

MessageBox.Show(ex.ToString());

}

}

private void btnChange\_Click(object sender, EventArgs e)

{

try

{

mycheckedListBox.Items.Clear();

}

catch (Exception ex)

{

MessageBox.Show(ex.ToString());

}

}

private void btnDel\_Click(object sender, EventArgs e)

{

try

{

for (int i = 0; i < mycheckedListBox.Items.Count; i++)

{

if (mycheckedListBox.GetItemChecked(i) == true)

{

mycheckedListBox.Items.RemoveAt(i);

}

}

}

catch (Exception ex)

{

MessageBox.Show(ex.ToString());

}

}

private void unselectall\_CheckedChanged(object sender, EventArgs e)

{

try

{

if (unselectall.Checked == true)

{

selectall.Checked = false;

for (int i = 0; i < mycheckedListBox.Items.Count; i++)

{

mycheckedListBox.SetItemChecked(i, false);

}

}

}

catch (Exception ex)

{

MessageBox.Show(ex.ToString());

}

}

private void button2\_Click(object sender, EventArgs e)

{

if (comboBox1.Text == "DES")

{

DES();

}

else if (comboBox1.Text == "AES")

{

AES();

}

else if (comboBox1.Text == "Uni Cipher")

{

Unicipher();

}

}

public void DES()

{

try

{

for (int i = 0; i < mycheckedListBox.Items.Count; i++)

{

if (mycheckedListBox.GetItemChecked(i) == true)

{

try

{

///normal encryption

try

{

UnicodeEncoding UE = new UnicodeEncoding();

byte[] key = UE.GetBytes(passkey);

string cryptFile = mycheckedListBox.Items[i].ToString();

FileStream fsCrypt = new FileStream(cryptFile, FileMode.Create);

RijndaelManaged RMCrypto = new RijndaelManaged();

CryptoStream cs = new CryptoStream(fsCrypt, RMCrypto.CreateEncryptor(key, key), CryptoStreamMode.Write);

FileStream fsIn = new FileStream(ofolder + "\\" + Path.GetFileName(mycheckedListBox.Items[i].ToString()), FileMode.Open);

int data;

while ((data = cs.ReadByte()) != -1) cs.WriteByte((byte)data);

fsIn.Close();

cs.Close();

fsCrypt.Close();

}

catch

{

MessageBox.Show("Encryption failed!", "Error");

}

}

catch (Exception e11)

{

MessageBox.Show(e11.ToString());

}

}

}

}

catch (Exception ex)

{

MessageBox.Show("Check a Renaming Style First... See on lect Round Selections");

MessageBox.Show(ex.ToString());

}

}

public void Unicipher()

{

try

{

for (int i = 0; i < mycheckedListBox.Items.Count; i++)

{

if (mycheckedListBox.GetItemChecked(i) == true)

{

try

{

for (int j = 0; j < 2; j++)

{

UnicodeEncoding UE = new UnicodeEncoding();

byte[] key = UE.GetBytes(passkey);

string cryptFile = mycheckedListBox.Items[i].ToString();

FileStream fsCrypt = new FileStream(cryptFile, FileMode.Create);

RijndaelManaged RMCrypto = new RijndaelManaged();

CryptoStream cs = new CryptoStream(fsCrypt, RMCrypto.CreateEncryptor(key, key), CryptoStreamMode.Write);

FileStream fsIn = new FileStream(ofolder + "\\" + Path.GetFileName(mycheckedListBox.Items[i].ToString()), FileMode.Open);

int data;

while ((data = fsIn.ReadByte()) != -1) cs.WriteByte((byte)data);

fsIn.Close();

cs.Close();

fsCrypt.Close();

}

}

catch (Exception e11)

{

MessageBox.Show(e11.ToString());

}

try

{

UnicodeEncoding UE = new UnicodeEncoding();

byte[] key = UE.GetBytes(passkey);

string cryptFile = Sname;

FileStream fsCrypt = new FileStream(cryptFile, FileMode.Create);

RijndaelManaged RMCrypto = new RijndaelManaged();

CryptoStream cs = new CryptoStream(fsCrypt, RMCrypto.CreateEncryptor(key, key), CryptoStreamMode.Write);

FileStream fsIn = new FileStream(Sname, FileMode.Open);

int data;

while ((data = cs.ReadByte()) != -1) cs.WriteByte((byte)data);

fsIn.Close();

cs.Close();

fsCrypt.Close();

}

catch (Exception e11)

{

MessageBox.Show(e11.ToString());

}

}

}

}

catch (Exception ex)

{

MessageBox.Show("Check a Renaming Style First... See on lect Round Selections");

MessageBox.Show(ex.ToString());

}

}

public void AES()

{

try

{

for (int i = 0; i < mycheckedListBox.Items.Count; i++)

{

if (mycheckedListBox.GetItemChecked(i) == true)

{

try

{

for (int j = 0; j < 3; j++)

{

UnicodeEncoding UE = new UnicodeEncoding();

byte[] key = UE.GetBytes(passkey);

string cryptFile = mycheckedListBox.Items[i].ToString();

FileStream fsCrypt = new FileStream(cryptFile, FileMode.Create);

RijndaelManaged RMCrypto = new RijndaelManaged();

CryptoStream cs = new CryptoStream(fsCrypt, RMCrypto.CreateEncryptor(key, key), CryptoStreamMode.Write);

FileStream fsIn = new FileStream(ofolder + "\\" + Path.GetFileName(mycheckedListBox.Items[i].ToString()), FileMode.Open);

int data;

while ((data = fsIn.ReadByte()) != -1) cs.WriteByte((byte)data);

fsIn.Close();

cs.Close();

fsCrypt.Close();

}

}

catch (Exception e11)

{

MessageBox.Show(e11.ToString());

}

for (int j = 0; j < 2; j++)

{

try

{

UnicodeEncoding UE = new UnicodeEncoding();

byte[] key = UE.GetBytes(passkey);

string cryptFile = Sname;

FileStream fsCrypt = new FileStream(cryptFile, FileMode.Create);

RijndaelManaged RMCrypto = new RijndaelManaged();

CryptoStream cs = new CryptoStream(fsCrypt, RMCrypto.CreateEncryptor(key, key), CryptoStreamMode.Write);

FileStream fsIn = new FileStream(Sname, FileMode.Open);

int data;

while ((data = fsIn.ReadByte()) != -1) cs.WriteByte((byte)data);

fsIn.Close();

cs.Close();

fsCrypt.Close();

}

catch (Exception e11)

{

MessageBox.Show(e11.ToString());

}

}

}

}

}

catch (Exception ex)

{

MessageBox.Show("Check a Renaming Style First... See on lect Round Selections");

MessageBox.Show(ex.ToString());

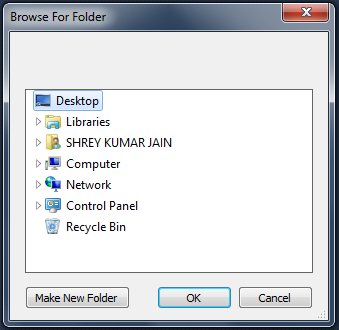
}

}

}

}

**4.7 Folder Compression:**

Folder compression compresses all files in the selected folder. The compression technique used is GZ compression.

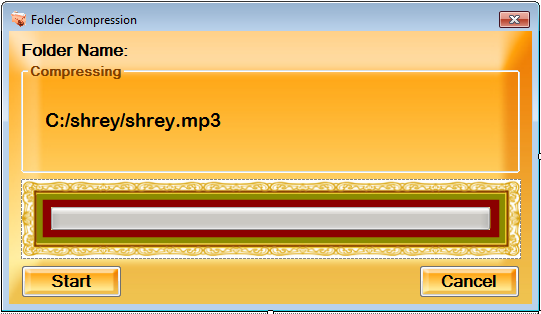
****

Fig 4.7: Folder Compression

**Coding:**

namespace Folder\_Compression

{

public partial class Compression : Form

{

string recieve;

int count = 0;

float prog;

public Compression()

{

InitializeComponent();

}

private void Folder\_Compression\_Load(object sender, EventArgs e)

{

try

{

if (folderBrowserDialog1.ShowDialog() == DialogResult.OK)

{

recieve = folderBrowserDialog1.SelectedPath;

label1.Text = recieve;

DirectoryInfo di = new DirectoryInfo(recieve);

//counting files

foreach (FileInfo fi in di.GetFiles())

{

count++;

}

prog = 100 / count;

}

else

{

this.Close();

}

}

Catch{}

}

private void button1\_Click(object sender, EventArgs e)

{ this.Close(); }

public void Compress(FileInfo fi)

{

using (FileStream inFile = fi.OpenRead())

{

if ((File.GetAttributes(fi.FullName) & FileAttributes.Hidden) != FileAttributes.Hidden & fi.Extension != ".gz")

{

using (FileStream outFile = File.Create(fi.FullName + ".gz"))

{

using (GZipStream Compress = new GZipStream(outFile, CompressionMode.Compress))

{

byte[] buffer = new byte[4096];

int numRead;

while ((numRead = inFile.Read(buffer, 0, buffer.Length)) != 0)

{

Compress.Write(buffer, 0, numRead);

} }}}} File.Delete(fi.FullName); }

private void button2\_Click(object sender, EventArgs e)

{

this.UseWaitCursor = true;

// Compress the directory's files.

DirectoryInfo di = new DirectoryInfo(recieve);

foreach (FileInfo fi in di.GetFiles())

{

label2.Text = fi.Name;

Compress(fi);

//file\_count++;

prog += prog;

progressBar1.Increment((int)prog);

}

MessageBox.Show("Compression Completed");

this.UseWaitCursor = false;

}}}

**4.8 Hide Folder:**

Hide folder module can be used to hide any folder by applying attributes to the folder as hidden, non readable, unlink, etc.

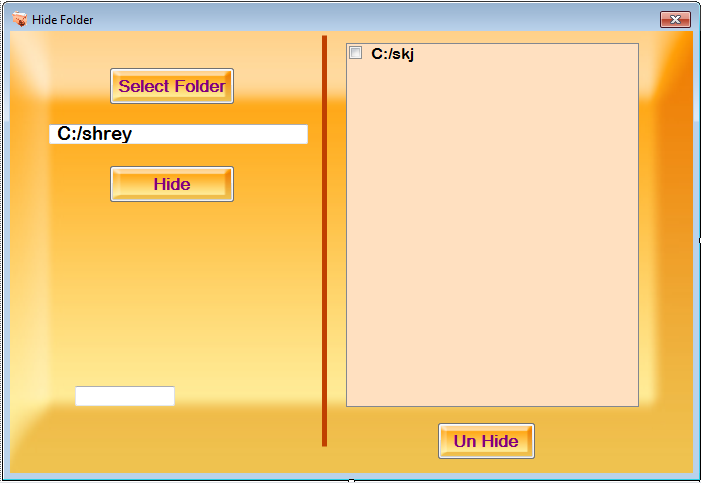
* Working:
  + Select Folder
  + Hide by applying attributes
  + Putting Names in Database

Fig 4.8: Hide Folder

**Coding:**

namespace Data\_and\_PC\_Securer

{

public partial class Hide\_Folder : Form

{

public Hide\_Folder()

{

InitializeComponent();

}

private void button2\_Click(object sender, EventArgs e)

{

if (folderBrowserDialog1.ShowDialog() == DialogResult.OK)

{

textBox1.Text = folderBrowserDialog1.SelectedPath;

}

}

private void button3\_Click(object sender, EventArgs e)

{

textBox2.Text = System.IO.Path.GetFileName(textBox1.Text);

// encrypt the only folder name

foreach (char c in textBox2.Text)

{

char enc = (char)(c + 1);

textBox2.Text = enc.ToString();

}

// rename folder with encrypted name

File.Move(folderBrowserDialog1.SelectedPath.ToString(), System.IO.Path.GetDirectoryName(folderBrowserDialog1.SelectedPath).ToString() + "\\" + textBox2.Text);

// save encrypt name with original folder name in DB // put attributes over it by DOS commands

if (textBox1.Text != "" | textBox2.Text != "")

{

try

{

SqlConnection con = new SqlConnection("Data Source=SHREYKUMARJAIN\\SHREYKUMARJAIN; Initial Catalog=DataSecurer; Integrated Security=TRUE");

SqlCommand cmd;

con.Open();

cmd = new SqlCommand("insert into hide\_folder values('" + textBox2.Text + "','" + System.IO.Path.GetFileName(textBox1.Text) + "','" + System.IO.Path.GetDirectoryName(folderBrowserDialog1.SelectedPath) + "')", con);

int tempo = cmd.ExecuteNonQuery();

if (tempo > 0)

{

StreamWriter sw1 = new StreamWriter(@"c:\shrey.bat");

sw1.WriteLine("@ECHO OFF");

sw1.WriteLine("cd\\");

sw1.WriteLine("attrin +r +a +s +h +i " + System.IO.Path.GetDirectoryName(folderBrowserDialog1.SelectedPath) + "\\" + textBox2.Text + " /s /d /l ");

sw1.Close();

con.Close();

}

else

{

MessageBox.Show("Unknown error");

}

}

catch (Exception e7)

{

MessageBox.Show(e7.ToString());

}

try

{

System.Diagnostics.Process proc = new System.Diagnostics.Process();

proc.StartInfo.FileName = @"c:\shrey.bat";

proc.StartInfo.RedirectStandardError = false;

proc.StartInfo.RedirectStandardOutput = false;

proc.StartInfo.UseShellExecute = false;

proc.Start();

proc.WaitForExit();

}

catch (Exception)

{

}

File.Delete(@"c:\shrey.bat");

}

else

{

MessageBox.Show("No Folder Selected");

}

// show the original name in hidden folder box

refreshlist();

}

private void button1\_Click(object sender, EventArgs e)

{

// select encrypted name from the original folder name from DB

try

{

SqlConnection con = new SqlConnection("Data Source=SHREYKUMARJAIN\\SHREYKUMARJAIN; Initial Catalog=DataSecurer; Integrated Security=TRUE");

SqlCommand cmd;

con.Open();

SqlDataReader rd;

cmd = new SqlCommand("select \* from hide\_folder where Oname='" + mycheckedListBox.SelectedItem.ToString() + "'", con);

rd = cmd.ExecuteReader();

if (rd.Read() == true)

{

File.Delete(@"c:\shrey.bat");

StreamWriter sw1 = new StreamWriter(@"c:\shrey.bat");

sw1.WriteLine("@ECHO OFF");

sw1.WriteLine("cd\\");

sw1.WriteLine("attrin -r -a -s -h -i " + rd[2].ToString() + "\\" + rd[0].ToString() + " /s /d /l ");

sw1.Close();

// rename folder by original name

File.Move(rd[2].ToString() + "//" + rd[0].ToString(), rd[2].ToString() + "//" + rd[1].ToString());

con.Close();

}

}

catch (Exception e1)

{

MessageBox.Show(e1.ToString());

}

// remove attrib from encrypted name folder

try

{

System.Diagnostics.Process proc = new System.Diagnostics.Process();

proc.StartInfo.FileName = @"c:\shrey.bat";

proc.StartInfo.RedirectStandardError = false;

proc.StartInfo.RedirectStandardOutput = false;

proc.StartInfo.UseShellExecute = false;

proc.Start();

proc.WaitForExit();

}

catch (Exception)

{

}

File.Delete(@"c:\shrey.bat");

// delete row

try

{

SqlConnection con = new SqlConnection("Data Source=SHREYKUMARJAIN\\SHREYKUMARJAIN; Initial Catalog=DataSecurer; Integrated Security=TRUE");

SqlCommand cmd;

con.Open();

cmd = new SqlCommand("delete from hide\_folder where Oname='" + mycheckedListBox.SelectedItem.ToString() + "'", con);

con.Close();

}

catch (Exception e10)

{

MessageBox.Show(e10.ToString());

}

//refreshing list

refreshlist();

}

private void refreshlist()

{

mycheckedListBox.Items.Clear();

try

{

SqlConnection con = new SqlConnection("Data Source=SHREYKUMARJAIN\\SHREYKUMARJAIN; Initial Catalog=DataSecurer; Integrated Security=TRUE");

SqlCommand cmd;

con.Open();

SqlDataReader rd;

cmd = new SqlCommand("select Oname from hide\_folder'" , con);

rd = cmd.ExecuteReader();

if (rd.Read() == true)

{

foreach (string a in rd)

{

mycheckedListBox.Items.Add(a);

}

}

}

catch (Exception e1)

{

MessageBox.Show(e1.ToString());

}

}

}

}

**4.8 Intelligent Human Readable Text:**

This module converts text into such format that only human mind can judge what letter or word it is. User can save the text and also can send it to others.

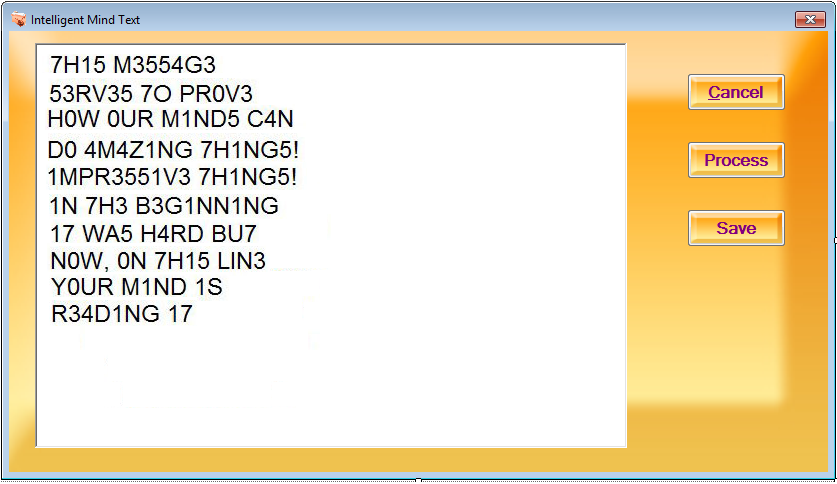


Fig 4.8: Intelligent Human Readable Text:

**Coding:**

private void button2\_Click(object sender, EventArgs e){

button3.Visible = true;

richTextBox1.Text = richTextBox1.Text.Replace("A", "4");

richTextBox1.Text = richTextBox1.Text.Replace("E", "3");

richTextBox1.Text = richTextBox1.Text.Replace("I", "1");

richTextBox1.Text = richTextBox1.Text.Replace("O", "0");

richTextBox1.Text = richTextBox1.Text.Replace("S", "5");

richTextBox1.Text = richTextBox1.Text.Replace("T", "7");}

private void button1\_Click(object sender, EventArgs e)

{ this.Close(); }

private void richTextBox1\_KeyPress(object sender, KeyPressEventArgs e)

{ e.KeyChar = Char.ToUpper(e.KeyChar);}

private void button3\_Click(object sender, EventArgs e)

{

saveFileDialog1.Filter = "Text|\*.txt";

DialogResult res = saveFileDialog1.ShowDialog();

if (res == DialogResult.Cancel)

{

return;

}

string file\_name = saveFileDialog1.FileName;

MessageBox.Show(file\_name);

StreamWriter sw = new StreamWriter(file\_name);

sw.WriteLine(richTextBox1.Text);

sw.Flush();

sw.Close();}

**4.9 Hide Message:**

Hide Message module can hide any text message in any format of image, basically we can implement such hiding technique for any file but in this version of project I’ve only used for Images.

Fig 4.9 Hide Message:

**Coding:**

namespace Data\_and\_PC\_Securer

{

public partial class Hide\_Message : Form

{

public Hide\_Message()

{

InitializeComponent();

}

private void button2\_Click(object sender, EventArgs e)

{

if (openFileDialog1.ShowDialog() == DialogResult.OK)

{

pictureBox1.ImageLocation = openFileDialog1.FileName;

textBox1.Text = openFileDialog1.FileName;

}

}

private void button1\_Click(object sender, EventArgs e)

{

try

{

//save text file

StreamWriter sw = new StreamWriter(Path.GetDirectoryName(openFileDialog1.FileName) + "\\shrey.txt");

sw.WriteLine(richTextBox1.Text);

sw.Flush();

sw.Close();

//savebat file

StreamWriter sw1 = new StreamWriter(Path.GetDirectoryName(openFileDialog1.FileName) + "\\shrey.bat");

sw1.WriteLine("echo off");

sw1.WriteLine("cd " + Path.GetDirectoryName(openFileDialog1.FileName));

sw1.WriteLine("copy /b " + openFileDialog1.SafeFileName + "+shrey.txt " + Path.GetFileNameWithoutExtension(openFileDialog1.FileName) + "1" + Path.GetExtension(openFileDialog1.FileName));

sw1.Flush();

sw1.Close();

}

catch

{

}

//running bat file

try

{

System.Diagnostics.Process proc = new System.Diagnostics.Process();

proc.StartInfo.FileName = Path.GetDirectoryName(openFileDialog1.FileName) + "\\shrey.bat";

proc.StartInfo.RedirectStandardError = false;

proc.StartInfo.RedirectStandardOutput = false;

proc.StartInfo.UseShellExecute = false;

proc.Start();

proc.WaitForExit();

MessageBox.Show("Message Hided");

}

catch (Exception)

{

MessageBox.Show("There exist a file named as :" + Path.GetFileNameWithoutExtension(openFileDialog1.FileName) + "1" + Path.GetExtension(openFileDialog1.FileName) + " please remove it first\nfrom the folder from image");

File.Delete(Path.GetDirectoryName(openFileDialog1.FileName) + "\\shrey.txt");

File.Delete(Path.GetDirectoryName(openFileDialog1.FileName) + "\\shrey.bat");

}

}

private void radioButton1\_CheckedChanged(object sender, EventArgs e)

{

if (radioButton1.Checked == true)

{

radioButton2.Checked = true;

MessageBox.Show("To Retrive Message: 1.Open Image in Notepad \n2.Move to the End of File... You Can See The Message");

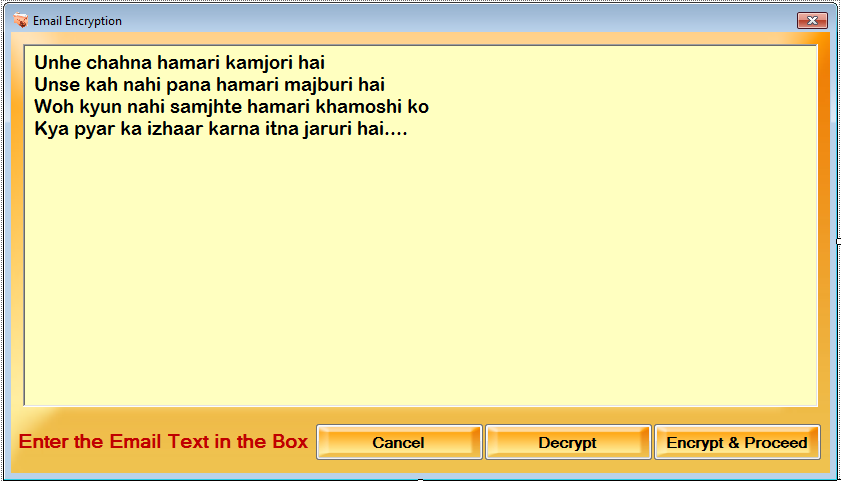
}

}

}

}

**4.10 Email Encryption:**

Email encryption provides user to first encrypt the text he/she wants to send and then move that text to the email client application by which user can send email with attachments to anyone in the world without opening the actual email web client. Emails are sent by SMTP server with authentication.

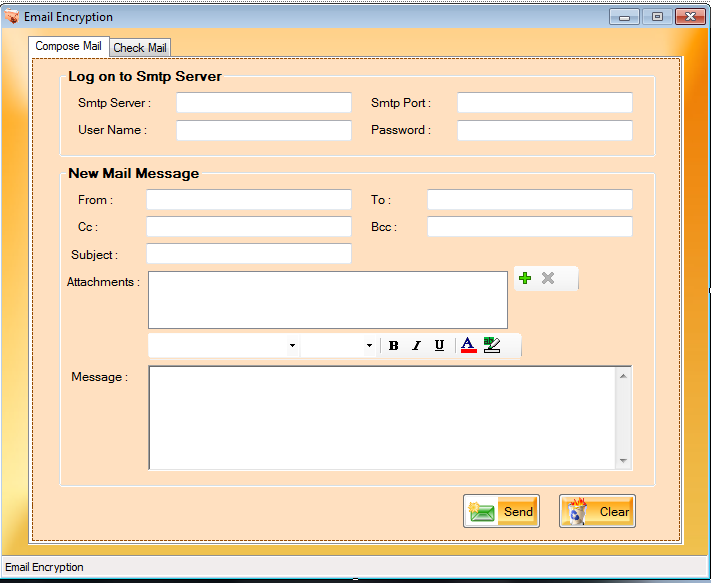
****

Fig 4.10: Email Encryption

**Coding:**

**Email Encryption:**

namespace Data\_and\_PC\_Securer

{

public partial class Email\_Encryption : Form

{

public Email\_Encryption()

{

InitializeComponent();

}

private void button2\_Click(object sender, EventArgs e)

{

this.Close();

}

private void button1\_Click(object sender, EventArgs e)

{

foreach (char c in richTextBox1.Text)

{

char enc = (char)(c + 11);

richTextBox1.Text = enc.ToString();

}

EmailClient ec = new EmailClient();

ec.Show();

ec.get\_text(richTextBox1.Text);

}

private void button3\_Click(object sender, EventArgs e)

{

foreach (char c in richTextBox1.Text)

{

char enc = (char)(c - 11);

richTextBox1.Text = enc.ToString();

}

}

}

}

**Email Client:**

namespace Email\_Client

{

public partial class EmailClient : Form

{

ArrayList attachments;

ArrayList popAttachmentsIndex;

bool \_lock = false;

Pop3Client pop = null;

string email = "";

int msg\_id = 0;

ArrayList attached\_file\_names;

public EmailClient()

{

InitializeComponent();

this.SetSelectionFont();

this.attachments = new ArrayList();

this.popAttachmentsIndex = new ArrayList();

this.attached\_file\_names = new ArrayList();

}

private void EmailTimer\_Tick(object sender, EventArgs e)

{

if (SmtpServer.ContainsFocus)

this.SmtpServer.BackColor = Color.Ivory;

else

this.SmtpServer.BackColor = SystemColors.Window;

if (SmtpPort.ContainsFocus)

this.SmtpPort.BackColor = Color.Ivory;

else

this.SmtpPort.BackColor = SystemColors.Window;

if (UserName.ContainsFocus)

this.UserName.BackColor = Color.Ivory;

else

this.UserName.BackColor = SystemColors.Window;

if (Password.ContainsFocus)

this.Password.BackColor = Color.Ivory;

else

this.Password.BackColor = SystemColors.Window;

if (PopServer.ContainsFocus)

this.PopServer.BackColor = Color.Ivory;

else

this.PopServer.BackColor = SystemColors.Window;

if (PopPort.ContainsFocus)

this.PopPort.BackColor = Color.Ivory;

else

this.PopPort.BackColor = SystemColors.Window;

if (PopUserName.ContainsFocus)

this.PopUserName.BackColor = Color.Ivory;

else

this.PopUserName.BackColor = SystemColors.Window;

if (PopPassword.ContainsFocus)

this.PopPassword.BackColor = Color.Ivory;

else

this.PopPassword.BackColor = SystemColors.Window;

if (From.ContainsFocus)

this.From.BackColor = Color.Ivory;

else

this.From.BackColor = SystemColors.Window;

if (To.ContainsFocus)

this.To.BackColor = Color.Ivory;

else

this.To.BackColor = SystemColors.Window;

if (Cc.ContainsFocus)

this.Cc.BackColor = Color.Ivory;

else

this.Cc.BackColor = SystemColors.Window;

if (Bcc.ContainsFocus)

this.Bcc.BackColor = Color.Ivory;

else

this.Bcc.BackColor = SystemColors.Window;

if (Subject.ContainsFocus)

this.Subject.BackColor = Color.Ivory;

else

this.Subject.BackColor = SystemColors.Window;

if (FromPopHeader.ContainsFocus)

this.FromPopHeader.BackColor = Color.Ivory;

else

this.FromPopHeader.BackColor = SystemColors.Window;

if (ToPopHeader.ContainsFocus)

this.ToPopHeader.BackColor = Color.Ivory;

else

this.ToPopHeader.BackColor = SystemColors.Window;

if (SubjectPopHeader.ContainsFocus)

this.SubjectPopHeader.BackColor = Color.Ivory;

else

this.SubjectPopHeader.BackColor = SystemColors.Window;

}

private void SmtpPort\_KeyPress(object sender, KeyPressEventArgs e)

{

if (!Char.IsDigit(e.KeyChar) && e.KeyChar != (char)8)

{

e.Handled = true;

}

}

private void Send\_Click(object sender, EventArgs e)

{

if (this.CheckInputValidation(SmtpServer.Text, SmtpPort.Text, UserName.Text, Password.Text, From.Text, To.Text,Cc.Text,Bcc.Text))

{

if (this.EmailValidation(this.From.Text))

{

bool isRecipient = false;

if (this.To.Text.Length > 0)

{

if (this.RecipientsEmailValidation(this.To.Text))

{

isRecipient = true;

}

else

{

MessageBox.Show(this, "Recipients' email address is not in the correct format, in \"To: \" field.", "Email Client", MessageBoxButtons.OK, MessageBoxIcon.Error);

return;

}

}

if (this.Cc.Text.Length > 0)

{

if (!(this.RecipientsEmailValidation(this.Cc.Text)))

{

MessageBox.Show(this, "Recipients' email address is not in the correct format, in \"Cc: \" field.", "Email Client", MessageBoxButtons.OK, MessageBoxIcon.Error);

return;

}

else

{

isRecipient = true;

}

}

if (this.Bcc.Text.Length > 0)

{

if (!(this.RecipientsEmailValidation(this.Bcc.Text)))

{

MessageBox.Show(this, "Recipients' email address is not in the correct format, in \"Bcc: \" field.", "Email Client", MessageBoxButtons.OK, MessageBoxIcon.Error);

return;

}

else

{

isRecipient = true;

}

}

if (Internet.IsConnectedToInternet())

{

if (isRecipient == true)

{

Rtf2Html rtf = new Rtf2Html();

string Html = rtf.ConvertRtfToHtml(this.MailMessage);

MailMessage mail\_message = new MailMessage();

mail\_message.From = this.From.Text;

mail\_message.To = this.To.Text;

mail\_message.CC = this.Cc.Text;

mail\_message.BCC = this.Bcc.Text;

mail\_message.Subject = this.Subject.Text;

mail\_message.MailType = MailEncodingType.HTML;

mail\_message.MailPriority = MailSendPriority.NORMAL;

mail\_message.Message = Html;

mail\_message.Attachments = this.attachments;

Thread thread = new Thread(new ParameterizedThreadStart(this.SendEmail));

thread.Start(mail\_message);

}

else

{

return;

}

}

else

{

MessageBox.Show(this, "You must connect to the internet.", "Email Client", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

}

else

{

MessageBox.Show(this, "Sender email address is not in the correct format.", "Email Client", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

}

private void SmtpClear\_Click(object sender, EventArgs e)

{

this.SmtpServer.Text = "";

this.SmtpPort.Text = "";

this.UserName.Text = "";

this.Password.Text = "";

this.From.Text = "";

this.To.Text = "";

this.Cc.Text = "";

this.Bcc.Text = "";

this.Subject.Text = "";

this.MailMessage.Text = "";

this.Attachments.Clear();

}

void smtp\_Disconnected(object sender, string Server)

{

if (this.statusStrip.InvokeRequired)

{

DisconnectEventHandler discon = new DisconnectEventHandler(this.smtp\_Disconnected);

this.Invoke(discon, new object[] { sender,Server });

}

else

{

this.ProgressLabel.Text = "Disconnected with the smtp server \"" + Server + "\"";

Thread.Sleep(500);

this.Send.Enabled = true;

this.ProgressLabel.Text = "Email Client";

}

}

void smtp\_EndedDataTransfer(object sender)

{

if (this.statusStrip.InvokeRequired)

{

DataTransferEventHandler data = new DataTransferEventHandler(this.smtp\_EndedDataTransfer);

this.Invoke(data, new object[] { sender });

}

else

{

this.ProgressLabel.Text = "Email message has sent";

}

}

void smtp\_StartedDataTransfer(object sender)

{

if (this.statusStrip.InvokeRequired)

{

DataTransferEventHandler data = new DataTransferEventHandler(this.smtp\_StartedDataTransfer);

this.Invoke(data, new object[] { sender});

}

else

{

this.ProgressLabel.Text = "Sending email message";

}

}

void smtp\_AuthenticationFinished(object sender, string userName)

{

if (this.statusStrip.InvokeRequired)

{

AuthenticateEventHandler auth = new AuthenticateEventHandler(this.smtp\_AuthenticationFinished);

this.Invoke(auth, new object[] { sender, userName });

}

else

{

this.ProgressLabel.Text = "Verification is completed";

}

}

void smtp\_AuthenticationBegan(object sender, string userName)

{

if (this.statusStrip.InvokeRequired)

{

AuthenticateEventHandler auth = new AuthenticateEventHandler(this.smtp\_AuthenticationBegan);

this.Invoke(auth, new object[] { sender, userName});

}

else

{

this.ProgressLabel.Text = "Verifying user name and password";

}

}

void smtp\_ConnectionEstablishing(object sender, string Server, int Port)

{

if (this.statusStrip.InvokeRequired)

{

ConnectEventHandler con = new ConnectEventHandler(this.smtp\_ConnectionEstablishing);

this.Invoke(con, new object[] { sender, Server, Port });

}

else

{

this.ProgressLabel.Text = "Connecting to smtp server \"" + Server + "\" on port " + Port;

}

}

void smtp\_ConnectionEstablished(object sender, string Server, int Port)

{

if (this.statusStrip.InvokeRequired)

{

ConnectEventHandler con = new ConnectEventHandler(this.smtp\_ConnectionEstablished);

this.Invoke(con, new object[] { sender, Server, Port });

}

else

{

this.ProgressLabel.Text = "Connection is established with the smtp server \"" + Server + "\"";

}

}

private void AttachmentToolStrip\_ItemClicked(object sender, ToolStripItemClickedEventArgs e)

{

if (e.ClickedItem.Tag == null)

return;

else if(e.ClickedItem.Tag.ToString() == "Add")

{

OpenFileDialog dialog = new OpenFileDialog();

if (dialog.ShowDialog() == DialogResult.OK)

{

ListViewItem item = new ListViewItem(Path.GetFileName(dialog.FileName) + " (" + ((double)(new FileInfo(dialog.FileName).Length / 1000)).ToString("f2") + " KB)");

item.ImageIndex = 0;

item.Tag = dialog.FileName;

this.attachments.Add(dialog.FileName);

this.Attachments.Items.Add(item);

}

}

else if (e.ClickedItem.Tag.ToString() == "Delete")

{

if (this.Attachments.SelectedItems.Count > 0)

{

this.attachments.Remove(this.Attachments.SelectedItems[0].Tag.ToString());

this.Attachments.SelectedItems[0].Remove();

}

}

}

private void Attachments\_SelectedIndexChanged(object sender, EventArgs e)

{

if (this.Attachments.SelectedItems.Count > 0)

{

this.AttachmentToolStrip.Items[1].Enabled = true;

}

else

{

this.AttachmentToolStrip.Items[1].Enabled = false;

}

}

private void FontStyle\_SelectedIndexChanged(object sender, EventArgs e)

{

this.SetSelectionFont();

}

private void FontSize\_SelectedIndexChanged(object sender, EventArgs e)

{

this.SetSelectionFont();

}

private void Bold\_Click(object sender, EventArgs e)

{

if (((ToolStripButton)this.FormattingToolStrip.Items[3]).Checked)

{

((ToolStripButton)this.FormattingToolStrip.Items[3]).Checked = false;

}

else

{

((ToolStripButton)this.FormattingToolStrip.Items[3]).Checked = true;

}

this.SetSelectionFont();

}

private void Italic\_Click(object sender, EventArgs e)

{

if (((ToolStripButton)this.FormattingToolStrip.Items[4]).Checked)

{

((ToolStripButton)this.FormattingToolStrip.Items[4]).Checked = false;

}

else

{

((ToolStripButton)this.FormattingToolStrip.Items[4]).Checked = true;

}

this.SetSelectionFont();

}

private void Underline\_Click(object sender, EventArgs e)

{

if (((ToolStripButton)this.FormattingToolStrip.Items[5]).Checked)

{

((ToolStripButton)this.FormattingToolStrip.Items[5]).Checked = false;

}

else

{

((ToolStripButton)this.FormattingToolStrip.Items[5]).Checked = true;

}

this.SetSelectionFont();

}

private void FontColor\_Click(object sender, EventArgs e)

{

ColorDialog dialog = new ColorDialog();

if (dialog.ShowDialog(this) == DialogResult.OK)

{

this.MailMessage.SelectionColor = dialog.Color;

((ToolStripButton)this.FormattingToolStrip.Items[7]).Image = CreateFontColorIcon(dialog.Color);

}

}

private void FontBackgroundColor\_Click(object sender, EventArgs e)

{

ColorDialog dialog = new ColorDialog();

if (dialog.ShowDialog(this) == DialogResult.OK)

{

this.MailMessage.SelectionBackColor = dialog.Color;

((ToolStripButton)this.FormattingToolStrip.Items[8]).Image = CreateFontBackColorIcon(dialog.Color);

}

}

private void MailMessage\_SelectionChanged(object sender, EventArgs e)

{

if (this.MailMessage.SelectionFont == null)

return;

else

{

this.\_lock = true;

((ToolStripComboBox)this.FormattingToolStrip.Items[0]).Text = this.MailMessage.SelectionFont.Name;

((ToolStripComboBox)this.FormattingToolStrip.Items[1]).Text = this.MailMessage.SelectionFont.Size.ToString();

((ToolStripButton)this.FormattingToolStrip.Items[3]).Checked = this.MailMessage.SelectionFont.Bold;

((ToolStripButton)this.FormattingToolStrip.Items[4]).Checked = this.MailMessage.SelectionFont.Italic;

((ToolStripButton)this.FormattingToolStrip.Items[5]).Checked = this.MailMessage.SelectionFont.Underline;

((ToolStripButton)this.FormattingToolStrip.Items[7]).Image = this.CreateFontColorIcon(this.MailMessage.SelectionColor);

((ToolStripButton)this.FormattingToolStrip.Items[8]).Image = this.CreateFontBackColorIcon(this.MailMessage.SelectionBackColor);

this.\_lock = false;

}

}

private void PopPort\_KeyPress(object sender, KeyPressEventArgs e)

{

if (!Char.IsDigit(e.KeyChar) && e.KeyChar != (char)8)

{

e.Handled = true;

}

}

private void Connect\_Click(object sender, EventArgs e)

{

if (this.CheckInputValidationForPop(this.PopServer.Text, this.PopPort.Text, this.PopUserName.Text, this.PopPassword.Text))

{

if (Internet.IsConnectedToInternet())

{

Thread th = new Thread(new ThreadStart(this.ReceiveEmails));

th.Start();

}

else

{

MessageBox.Show(this, "You must connect to the internet.", "Email Client", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

}

}

private void pop\_Disconnected(object sender, string Server)

{

if (this.statusStrip.InvokeRequired)

{

DisconnectEventHandler discon = new DisconnectEventHandler(this.pop\_Disconnected);

this.Invoke(discon, new object[] { sender, Server });

}

else

{

this.ProgressLabel.Text = "Disconnected with the pop server \"" + Server + "\"";

Thread.Sleep(500);

this.Send.Enabled = true;

this.ProgressLabel.Text = "Email Client";

}

}

private void pop\_EndedDataReceiving(object sender)

{

if (this.statusStrip.InvokeRequired)

{

DataReceivingEventHandler data = new DataReceivingEventHandler(this.pop\_EndedDataReceiving);

this.Invoke(data, new object[] { sender });

}

else

{

this.ProgressLabel.Text = "Email message has received";

}

}

private void pop\_StartedDataReceiving(object sender)

{

if (this.statusStrip.InvokeRequired)

{

DataReceivingEventHandler data = new DataReceivingEventHandler(this.pop\_StartedDataReceiving);

this.Invoke(data, new object[] { sender });

}

else

{

this.ProgressLabel.Text = "Receiving email message";

}

}

private void pop\_AuthenticationFinished(object sender, string userName)

{

if (this.statusStrip.InvokeRequired)

{

AuthenticateEventHandler auth = new AuthenticateEventHandler(this.pop\_AuthenticationFinished);

this.Invoke(auth, new object[] { sender, userName });

}

else

{

this.ProgressLabel.Text = "Verification is completed";

}

}

private void pop\_AuthenticationBegan(object sender, string userName)

{

if (this.statusStrip.InvokeRequired)

{

AuthenticateEventHandler auth = new AuthenticateEventHandler(this.pop\_AuthenticationBegan);

this.Invoke(auth, new object[] { sender, userName });

}

else

{

this.ProgressLabel.Text = "Verifying user name and password";

}

}

private void pop\_ConnectionEstablished(object sender, string Server, int Port)

{

if (this.statusStrip.InvokeRequired)

{

ConnectEventHandler con = new ConnectEventHandler(this.pop\_ConnectionEstablished);

this.Invoke(con, new object[] { sender, Server, Port });

}

else

{

this.ProgressLabel.Text = "Connection is established with the pop server \"" + Server + "\"";

}

}

private void pop\_ConnectionEstablishing(object sender, string Server, int Port)

{

if (this.statusStrip.InvokeRequired)

{

ConnectEventHandler con = new ConnectEventHandler(this.pop\_ConnectionEstablishing);

this.Invoke(con, new object[] { sender, Server, Port });

}

else

{

this.ProgressLabel.Text = "Connecting to pop server \"" + Server + "\" on port " + Port;

}

}

private void Disconnect\_Click(object sender, EventArgs e)

{

if (this.pop != null)

{

DialogResult result = MessageBox.Show(this, "Do you want to disconnect with the pop server \"" + this.pop.Pop3Server + "\" ?", "Email Client", MessageBoxButtons.YesNo, MessageBoxIcon.Question);

if (result == DialogResult.Yes)

{

try

{

this.pop.Disconnect();

this.MailMessages.Items.Clear();

this.FromPopHeader.Text = "";

this.ToPopHeader.Text = "";

this.SubjectPopHeader.Text = "";

this.PopAttachments.Items.Clear();

this.PopMessage.DocumentText = "<html></html>";

MessageBox.Show(this, "You are disconnected with the pop server " + this.pop.Pop3Server + ".", "Email Client", MessageBoxButtons.OK, MessageBoxIcon.Information);

this.EnableDisableConnectButton(true);

this.EnableDisableDisconnectButton(false);

}

catch (Pop3ClientException err)

{

MessageBox.Show(this, err.ErrorMessage, "Email Client", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

}

}

private void MailMessages\_MouseClick(object sender, MouseEventArgs e)

{

if (e.Button == MouseButtons.Right && this.MailMessages.SelectedItems.Count > 0)

{

ContextMenuStrip menu = new ContextMenuStrip();

Image image = (Image) Email\_Client.Properties.Resources.delete;

menu.Items.Add("Delete",image);

menu.ItemClicked += new ToolStripItemClickedEventHandler(menu\_ItemClicked);

menu.Show(Control.MousePosition);

}

else if (e.Button == MouseButtons.Left && this.MailMessages.SelectedItems.Count > 0)

{

int index = this.MailMessages.SelectedItems[0].Index;

index = index + 1;

this.email = "";

this.msg\_id = index;

this.FromPopHeader.Text = "";

this.ToPopHeader.Text = "";

this.SubjectPopHeader.Text = "";

this.PopMessage.DocumentText = "<html></html>";

this.PopAttachments.Items.Clear();

this.attached\_file\_names.Clear();

this.popAttachmentsIndex.Clear();

Thread th = new Thread(new ThreadStart(this.FetchEmailCallBack));

th.Start();

}

}

private void menu\_ItemClicked(object sender, ToolStripItemClickedEventArgs e)

{

DialogResult result = MessageBox.Show(this, "Do you want to delete the selected message?", "Email Client", MessageBoxButtons.YesNo, MessageBoxIcon.Question);

if (result == DialogResult.Yes)

{

int index = this.MailMessages.SelectedItems[0].Index;

index = index + 1;

try

{

this.pop.DeleteEmail(index);

this.MailMessages.SelectedItems[0].Remove();

if (this.msg\_id == index)

{

this.FromPopHeader.Text = "";

this.ToPopHeader.Text = "";

this.SubjectPopHeader.Text = "";

this.PopAttachments.Items.Clear();

this.PopMessage.DocumentText = "<html></html>";

}

MessageBox.Show(this, "Message having id " + index + " is deleted.", "Email Client", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

catch (Pop3ClientException err)

{

MessageBox.Show(this, err.ErrorMessage, "Email Client", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

}

private void DownloadAttachment\_Click(object sender, EventArgs e)

{

int selected\_index = this.PopAttachments.SelectedItems[0].Index;

int index = Convert.ToInt32(this.popAttachmentsIndex[selected\_index].ToString());

string selectedFile = this.PopAttachments.SelectedItems[0].Text;

selectedFile = selectedFile.Trim(new char[] { '"' });

string file\_name = Path.GetFileNameWithoutExtension(selectedFile);

string file\_extension = Path.GetExtension(selectedFile);

SaveFileDialog obj = new SaveFileDialog();

obj.Title = "Save a File";

obj.FileName = file\_name;

obj.Filter = "Document (\*" + file\_extension +")|\*" + file\_extension + "|All Files|\*.\*";

obj.FilterIndex = 1;

if (obj.ShowDialog(this) == DialogResult.OK)

{

string content = "";

string content\_type = "";

string attached\_file\_name = "";

this.pop.GetMailSection(index,ref content,ref content\_type,ref attached\_file\_name);

byte[] decoded\_data = MailDecoder.ConvertFromBase64String(content);

File.WriteAllBytes(obj.FileName, decoded\_data);

MessageBox.Show(this, "File \"" + Path.GetFileName(obj.FileName) + "\" has saved.", "Email Client", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

}

private void PopAttachments\_SelectedIndexChanged(object sender, EventArgs e)

{

if (this.PopAttachments.SelectedItems.Count > 0)

{

this.SaveAttachmentToolStrip.Items[0].Enabled = true;

}

else

{

this.SaveAttachmentToolStrip.Items[0].Enabled = false;

}

}

private void MailMessages\_SelectedIndexChanged(object sender, EventArgs e)

{

if (!(this.MailMessages.SelectedItems.Count > 0))

{

this.FromPopHeader.Text = "";

this.ToPopHeader.Text = "";

this.SubjectPopHeader.Text = "";

this.PopAttachments.Items.Clear();

this.PopMessage.DocumentText = "<html></html>";

}

}

// The delegate used to enable or disable a button control, in another thread

private delegate void EnableDisableEventHandler(bool enable);

private void EnableDisableSendButton(bool enable)

{

if (this.Connect.InvokeRequired)

{

EnableDisableEventHandler obj = new EnableDisableEventHandler(this.EnableDisableSendButton);

this.Invoke(obj, new object[] { enable });

}

else

{

this.Send.Enabled = enable;

}

}

private void EnableDisableConnectButton(bool enable)

{

if (this.Connect.InvokeRequired)

{

EnableDisableEventHandler obj = new EnableDisableEventHandler(this.EnableDisableConnectButton);

this.Invoke(obj, new object[] { enable });

}

else

{

this.Connect.Enabled = enable;

}

}

private void EnableDisableDisconnectButton(bool enable)

{

if (this.Connect.InvokeRequired)

{

EnableDisableEventHandler obj = new EnableDisableEventHandler(this.EnableDisableDisconnectButton);

this.Invoke(obj, new object[] { enable });

}

else

{

this.Disconnect.Enabled = enable;

}

}

// The delegate used to insert an item in the Inbox ListView

private delegate void InboxItemEventHandler(ListViewItem item);

private void InsertItem(ListViewItem item)

{

if (this.MailMessages.InvokeRequired)

{

InboxItemEventHandler obj = new InboxItemEventHandler(this.InsertItem);

this.Invoke(obj, new object[] { item });

}

else

{

this.MailMessages.Items.Add(item);

}

}

// The delegate used to Add attached file names in Pop Attachments ListView

private delegate void InsertPopAttachedFileNamesEventHandler(ArrayList attached\_file\_names);

private void InsertPopAttachedFileNames(ArrayList attached\_file\_names)

{

if (this.MailMessages.InvokeRequired)

{

InsertPopAttachedFileNamesEventHandler obj = new InsertPopAttachedFileNamesEventHandler(this.InsertPopAttachedFileNames);

this.Invoke(obj, new object[] { attached\_file\_names });

}

else

{

this.PopAttachments.Items.Clear();

for (int i = 0; i < attached\_file\_names.Count; i++)

{

this.PopAttachments.Items.Add((ListViewItem)attached\_file\_names[i]);

}

}

}

// The delegate used to clear the items in Pop Attachments ListView

private delegate void ClearPopAttachmentsEventHandler();

private void ClearPopAttachments()

{

if (this.PopAttachments.InvokeRequired)

{

ClearPopAttachmentsEventHandler obj = new ClearPopAttachmentsEventHandler(this.ClearPopAttachments);

this.Invoke(obj, new object[] { });

}

else

{

this.PopAttachments.Items.Clear();

}

}

// The delegate used to write Pop Message in browser

private delegate void WritePopMessageEventHandler(string content);

private void WritePopMessage(string content)

{

if (this.PopMessage.InvokeRequired)

{

WritePopMessageEventHandler obj = new WritePopMessageEventHandler(this.WritePopMessage);

this.Invoke(obj, new object[] { content });

}

else

{

this.PopMessage.DocumentText = content.Trim();

}

}

// The delegate used to update Status Bar

private delegate void UpdateStatusBarEventHandler(string text);

private void UpdateStatusBar(string text)

{

if (this.statusStrip.InvokeRequired)

{

UpdateStatusBarEventHandler obj = new UpdateStatusBarEventHandler(this.UpdateStatusBar);

this.Invoke(obj, new object[] { text });

}

else

{

this.ProgressLabel.Text = text;

}

}

// The delegate used to update Pop message header

private delegate void UpdatePopMessageHeaderEventHandler(string from, string to, string subject);

private void UpdatePopMessageHeader(string from, string to, string subject)

{

if (this.FromPopHeader.InvokeRequired || this.ToPopHeader.InvokeRequired || this.SubjectPopHeader.InvokeRequired)

{

UpdatePopMessageHeaderEventHandler obj = new UpdatePopMessageHeaderEventHandler(this.UpdatePopMessageHeader);

this.Invoke(obj, new object[] {from,to,subject });

}

else

{

this.FromPopHeader.Text = from.Trim();

this.ToPopHeader.Text = to.Trim();

this.SubjectPopHeader.Text = subject.Trim();

}

}

private void ReceiveEmails()

{

try

{

Pop3Client pop\_client = new Pop3Client();

pop\_client.Pop3Server = this.PopServer.Text;

pop\_client.Pop3Port = Convert.ToInt32(this.PopPort.Text);

pop\_client.UserName = this.PopUserName.Text;

pop\_client.Password = this.PopPassword.Text;

this.EnableDisableConnectButton(false);

pop\_client.ConnectionEstablishing += new ConnectEventHandler(this.pop\_ConnectionEstablishing);

pop\_client.ConnectionEstablished += new ConnectEventHandler(this.pop\_ConnectionEstablished);

pop\_client.AuthenticationBegan += new AuthenticateEventHandler(this.pop\_AuthenticationBegan);

pop\_client.AuthenticationFinished += new AuthenticateEventHandler(this.pop\_AuthenticationFinished);

pop\_client.StartedDataReceiving += new DataReceivingEventHandler(this.pop\_StartedDataReceiving);

pop\_client.EndedDataReceiving += new DataReceivingEventHandler(this.pop\_EndedDataReceiving);

pop\_client.Disconnected += new DisconnectEventHandler(this.pop\_Disconnected);

this.pop = pop\_client;

pop\_client.Connect();

this.EnableDisableDisconnectButton(true);

pop\_client.GetMailBoxDetails(); //it sets the TotalEmails and TotalEmailSize properties

if (pop\_client.TotalEmails >= 1)

{

this.UpdateStatusBar("Retrieving Emails");

this.FillInboxListView(pop\_client);

this.UpdateStatusBar("Email Client");

}

else

{

this.UpdateStatusBar("Email Client");

MessageBox.Show(this, "No email message exists in the inbox!.", "Email Client", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

}

catch (SmtpClientException err)

{

MessageBox.Show(this, err.ErrorMessage, "Email Client", MessageBoxButtons.OK, MessageBoxIcon.Error);

this.UpdateStatusBar("Email Client");

this.EnableDisableConnectButton(true);

this.EnableDisableDisconnectButton(false);

}

}

private void FetchEmailCallBack()

{

try

{

this.email = this.pop.FetchEmail(this.msg\_id);

this.UpdatePopMessageHeader(this.pop.From, this.pop.To, this.pop.Subject);

string content = "";

string content\_type = "";

string attached\_file\_name = "";

bool isHtmlIncluded = false;

int plain\_text\_message\_section = -1;

for (int i = 1; i <= this.pop.MailSections; i++)

{

this.pop.GetMailSection(i, ref content, ref content\_type, ref attached\_file\_name);

if (content\_type.ToLower().Equals("text/html"))

{

this.WritePopMessage(content);

isHtmlIncluded = true;

}

else if (content\_type.ToLower().Equals("base64"))

{

ListViewItem item = new ListViewItem(attached\_file\_name);

item.ImageIndex = 0;

attached\_file\_names.Add(item);

this.popAttachmentsIndex.Add(i);

}

else if (content\_type.ToLower().Equals("text/plain"))

{

plain\_text\_message\_section = i;

}

}

this.InsertPopAttachedFileNames(attached\_file\_names);

if (isHtmlIncluded == false && plain\_text\_message\_section != -1)

{

this.pop.GetMailSection(plain\_text\_message\_section, ref content, ref content\_type, ref attached\_file\_name);

this.WritePopMessage(content);

}

this.UpdateStatusBar("Email Client");

}

catch (Pop3ClientException err)

{

MessageBox.Show(this, err.ErrorMessage, "Email Client", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

private bool CheckInputValidationForPop(string pop\_server, string pop\_port, string user\_name, string password)

{

if (pop\_server.Equals(""))

{

MessageBox.Show(this, "You must provide pop server address.", "Email Client", MessageBoxButtons.OK, MessageBoxIcon.Information);

return false;

}

else if (pop\_port.Equals(""))

{

MessageBox.Show(this, "You must provide pop port number.", "Email Client", MessageBoxButtons.OK, MessageBoxIcon.Information);

return false;

}

else if (user\_name.Equals(""))

{

MessageBox.Show(this, "You must provide username.", "Email Client", MessageBoxButtons.OK, MessageBoxIcon.Information);

return false;

}

else if (password.Equals(""))

{

MessageBox.Show(this, "You must provide password.", "Email Client", MessageBoxButtons.OK, MessageBoxIcon.Information);

return false;

}

return true;

}

private void FillInboxListView(Pop3Client obj)

{

ArrayList sender = new ArrayList();

ArrayList subject = new ArrayList();

ArrayList date = new ArrayList();

ArrayList size = new ArrayList();

DateTime date\_time; string temp = "";

this.MailMessages.Items.Clear();

for (int i = 1; i <= obj.TotalEmails; i++)

{

string emailHeader = obj.FetchEmailTop(i, 0);

sender.Add(obj.From);

subject.Add(obj.Subject);

try

{

date\_time = DateTime.Parse(obj.Date);

temp = date\_time.ToString("D");

date.Add(temp);

}

catch(Exception)

{

date.Add(obj.Date);

}

size.Add(obj.GetMailSize(i));

}

for (int j = 0; j < sender.Count; j++)

{

ListViewItem item = new ListViewItem();

item.Text = sender[j].ToString();

item.SubItems.Add(subject[j].ToString());

item.SubItems.Add(date[j].ToString());

item.SubItems.Add(size[j].ToString());

this.InsertItem(item);

}

}

private void SendEmail(object mail\_msg)

{

try

{

MailMessage mail\_message = (MailMessage)mail\_msg;

SmtpClient smtp = new SmtpClient(this.SmtpServer.Text, Convert.ToInt32(this.SmtpPort.Text));

smtp.UserName = this.UserName.Text;

smtp.Password = this.Password.Text;

this.EnableDisableSendButton(false);

smtp.ConnectionEstablishing += new ConnectEventHandler(smtp\_ConnectionEstablishing);

smtp.ConnectionEstablished += new ConnectEventHandler(smtp\_ConnectionEstablished);

smtp.AuthenticationBegan += new AuthenticateEventHandler(smtp\_AuthenticationBegan);

smtp.AuthenticationFinished += new AuthenticateEventHandler(smtp\_AuthenticationFinished);

smtp.StartedDataTransfer += new DataTransferEventHandler(smtp\_StartedDataTransfer);

smtp.EndedDataTransfer += new DataTransferEventHandler(smtp\_EndedDataTransfer);

smtp.Disconnected += new DisconnectEventHandler(smtp\_Disconnected);

smtp.SendMail(mail\_message);

MessageBox.Show(this,"Email message has sent.", "Email Client", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

catch (SmtpClientException obj)

{

MessageBox.Show(this, obj.ErrorMessage, "Email Client", MessageBoxButtons.OK, MessageBoxIcon.Error);

this.EnableDisableSendButton(true);

this.ProgressLabel.Text = "Email Client";

}

}

private bool CheckInputValidation(string smtp\_server, string smtp\_port, string user\_name, string password, string from, string to,string cc,string bcc)

{

if (smtp\_server.Equals(""))

{

MessageBox.Show(this, "You must provide smtp server address.", "Email Client", MessageBoxButtons.OK, MessageBoxIcon.Information);

return false;

}

else if (smtp\_port.Equals(""))

{

MessageBox.Show(this, "You must provide smtp port number.", "Email Client", MessageBoxButtons.OK, MessageBoxIcon.Information);

return false;

}

else if (user\_name.Equals(""))

{

MessageBox.Show(this, "You must provide username.", "Email Client", MessageBoxButtons.OK, MessageBoxIcon.Information);

return false;

}

else if (password.Equals(""))

{

MessageBox.Show(this, "You must provide password.", "Email Client", MessageBoxButtons.OK, MessageBoxIcon.Information);

return false;

}

else if (from.Equals(""))

{

MessageBox.Show(this, "You must provide sender email address.", "Email Client", MessageBoxButtons.OK, MessageBoxIcon.Information);

return false;

}

else if ((!(to.Equals(""))) || (!(cc.Equals(""))) || (!(bcc.Equals(""))))

{

return true;

}

else if (to.Equals("") && cc.Equals("") && bcc.Equals(""))

{

MessageBox.Show(this, "You must provide recipient email address.", "Email Client", MessageBoxButtons.OK, MessageBoxIcon.Information);

return false;

}

return true;

}

private bool RecipientsEmailValidation(string recipient)

{

string[] splits = recipient.Split(new char[] { ',', ';' });

for (int i = 0; i < splits.Length; i++)

{

if (this.EmailValidation(splits[i]))

{

continue;

}

else

{

return false;

}

}

return true;

}

private bool EmailValidation(string email)

{

Regex regx = new Regex(@"([a-zA-Z\_0-9.-]+\@[a-zA-Z\_0-9.-]+\.\w+)", RegexOptions.IgnoreCase);

if (regx.IsMatch(email))

{

return true;

}

return false;

}

private void SetSelectionFont()

{

if (this.\_lock == true)

return;

else

{

System.Drawing.FontStyle style = System.Drawing.FontStyle.Regular;

if (((ToolStripButton)this.FormattingToolStrip.Items[3]).Checked)

{

style |= System.Drawing.FontStyle.Bold;

}

if (((ToolStripButton)this.FormattingToolStrip.Items[4]).Checked)

{

style |= System.Drawing.FontStyle.Italic;

}

if (((ToolStripButton)this.FormattingToolStrip.Items[5]).Checked)

{

style |= System.Drawing.FontStyle.Underline;

}

this.MailMessage.SelectionFont = new Font(((ToolStripComboBox)this.FormattingToolStrip.Items[0]).Text, Convert.ToSingle(((ToolStripComboBox)this.FormattingToolStrip.Items[1]).Text), style);

this.MailMessage.Focus();

}

}

private Bitmap CreateFontColorIcon(Color color)

{

Bitmap bmp = Email\_Client.Properties.Resources.fontcolor;

for (int x = 0; x < bmp.Width; x++)

{

for (int y = 12; y < bmp.Height; y++)

{

bmp.SetPixel(x, y, color);

}

}

return bmp;

}

private Bitmap CreateFontBackColorIcon(Color color)

{

Bitmap bmp = Email\_Client.Properties.Resources.fontbackcolor;

for (int x = 0; x < bmp.Width; x++)

{

for (int y = 12; y < bmp.Height; y++)

{

bmp.SetPixel(x, y, color);

}

}

return bmp;

}

public void get\_text(string gettext)

{

MailMessage.Text = gettext;

}

}

}

**4.11 Name Verification:**

As in password module you had seen that only password is only taken for authentication, this was because I’d developed this project mainly for administrator. So, now for user other than administrator must also verify his/her name for further authentication after entering password.

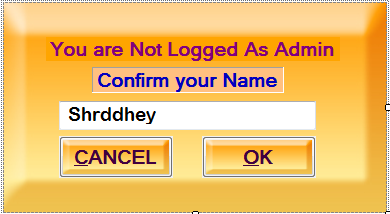
****

Fig 4.11: Name Verification

**Coding:**

namespace Data\_and\_PC\_Securer

{

public partial class Verify\_Name : Form

{

string pass;

public Verify\_Name()

{

InitializeComponent();

}

public void get\_pass(string p)

{

pass = p;

}

private void button1\_Click(object sender, EventArgs e)

{

if (textBox1.Text == null)

{

MessageBox.Show("Enter Name First");

textBox1.Focus();

}

else

{

try

{

SqlConnection con = new SqlConnection("Data Source=SHREYKUMARJAIN\\SHREYKUMARJAIN; Initial Catalog=DataSecurer; Integrated Security=TRUE");

SqlCommand cmd;

con.Open();

SqlDataReader rd;

cmd = new SqlCommand("select \* from passing where uname='" + textBox1.Text + "'", con);

rd = cmd.ExecuteReader();

if (rd.Read() == true)

{

if (pass == rd[1].ToString())

{

con.Close();

try

{

SqlCommand cmd1;

con.Open();

cmd1 = new SqlCommand("insert into online values('" + textBox1.Text + "')", con);

int tempo = cmd.ExecuteNonQuery();

if (tempo > 0)

{

}

else

{

MessageBox.Show("Must Work Offline");

}

con.Close();

}

catch (Exception e2)

{

MessageBox.Show(e2.ToString());

}

Data\_and\_PC\_Securer dp = new Data\_and\_PC\_Securer();

this.Close();

dp.Show();

}

else

{

MessageBox.Show("Invalid Password or invalid User Name");

Password pw = new Password();

pw.Show();

this.Close();

}

}

}

catch (Exception e1)

{

MessageBox.Show(e1.ToString());

}

}

}

}

}

**4.12 About:**

Information about Developer:

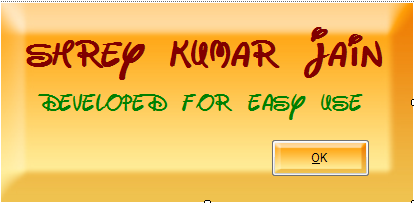


Fig 4.12: About

**Coding:**

namespace Data\_and\_PC\_Securer

{

public partial class About : Form

{

public About()

{

InitializeComponent();

}

private void button1\_Click(object sender, EventArgs e)

{

this.Close();

}

private void About\_Load(object sender, EventArgs e)

{

timer1.Interval = 15000;

timer1.Start();

}

private void timer1\_Tick(object sender, EventArgs e)

{

timer1.Stop();

button1\_Click(sender, e);

}

}

}

**4.13 C# Classes used:**

All the Namespaces and classes or project references used in the different modules of this project are as follows:

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Windows.Forms;

using System.Data.SqlClient;

using Email\_Client;

using Folder\_Compression;

using System.IO;

using System.Security.Cryptography;

using System.Security.Principal;

using System.Threading;

using System.Text.RegularExpressions;

using System.Net.Security;

using System.Net.Sockets;

using System.Net;

using System.Reflection;

These classes are used for different purpose like:

* Drawing of tools and windows forms
* Connect to SQL
* Execute SQL commands
* File handling
* Tool customizing
* SMTP Server connection
* Threading
* For checking regular expressions.
* Encryption
* Decryption
* Data compression