**WEB BASED CLAIM PROCESSING SYSTEM**

**WEB BASED CLAIM PROCESSING SYSTEM**

**PROJECT REPORT**

***Submitted by***

**Dinesh.S (11505210)**

**Koushik Chatterjee (11505216)**

**Mani Kumar. RM.N (11505219)**

**Mani Vannan.N (11505220)**

**Naveen Kumar.S (11505223)**

**Vinoth.R (11505254)**

***In partial fulfillment for the award of the diploma***

***Of***

**ENGINEERING AND TECHNOLOGY**

***In***

**COMPUTER ENGINEERING**

**HINDUSTAN INSTITUTE OF ENGINEERING TECHNOLOGY**

**POLYTECHNIC COLLEGE,**

**SEREPANANCHERY - 601 301.**

**APRIL 2013**

**Table of content**

|  |  |  |
| --- | --- | --- |
| **S. No.** | **Description** | **Page No.** |
| 1. | Abstract | 4 |
| 2 | Introduction | 5 |
| 3 | Hardware and Software Requirements | 6 |
| 4 | List of Modules | 11 |
| 5 | Existing System& Proposed System | 12 |
| 6 | Data Flow Diagrams | 13 |
| 7 | ER Diagram | 15 |
| 8 | Source Code | 19-31 |
| 9 | Snap Shots | 32-43 |
| 10 | Conclusion | 44 |
| 11 | Bibliography and References | 45 |

**Abstract**

The objective of the project is to apply claim (insurance) through online.

The Project has been designed in Visual studio.Net as front end and SQL server which acts as a back end.

The people's who work in office or corporate, they don't have time to go to their respective office’s allocated to them by their office management to submit the documents and apply for a claim.

In existing system it is a semi automated system. Here the policy holder needs to apply for any claim he can able to approach the policy office manually. Here the policy holder applies for his insurance claims through paper and submit necessary documents in Xerox copies. The insurer can receive the claim application from the policy holder and stores the information in local system database like ms word

and excel etc. It is a time delay process to solve more number of claim requests

To overcome this problem, we are going to implement this as a fully automated system. Claimant’s can directly apply to the offices through online.

This project is based on the ESI (Employee Security Insurance) company which is for medical insurance. An acknowledgement is provided by management to the customer or claimants to apply for a claim.

A reference ID is provided by the management to the customer and a file is send to the Admin about the customer the user has to login and provide remarks. Admin will checks the documents related to the reference ID and takes decision whether to grant\revoke the claims.

Web based claim processing system is a project developed for ESI (Employee Security Insurance) to enhanced the security and quick time apply for claim.

**INTRODUCTION**

A claim is a legal action to obtain money, property, or the enforcement of a right against another party. The legal document which carries a claim is called a Statement of Claim.

An insurance claim is the actual application for benefits provided by an insurance company. Policy holders must first file an insurance claim before any money can be disbursed to the hospital.

The insurance company may or may not approve the claim, based on their own assessment of the circumstances. Individuals who take out home, life, health, insurance policies must maintain regular payments called premiums to the insurers. In general, the insurance claim is filed with a local representative of the insurance company.

This agent becomes responsible for investigating the specific details of the insurance claim and negotiating the payment from the main insurers. Many times a recognized authority can file the necessary insurance claim forms directly with the insurance company.

After an insurance claim is field the insurance company may called the customer and verify all the details to accept the insurance. After then finally, they decide the actual amount to provide for the customer.

Hardware & Software Requirements

This section gives the details about the hardware and software’s environments that are required to successfully execute the Student Data Base System project.

**Hardware Requirements**

Processor : Pentium IV 2GHZ and above

Motherboard : supported for above processor

RAM : 1GB RAM and above

Monitor : 15” colors monitor (TFT or CRT)

HDD : SATA 40GB and above

Keyboard and mouse : ps/2 or USB connector

**Software Requirements**

Operating system : windows XP or higher

Developing tool : Microsoft visual studio V2010

Database : SQL server

Browser : Chrome or IE

**SOFTWARE SUMMARY**

**ASP.NET**

A Microsoft server-side Web technology. ASP.NET takes an object-oriented programming approach to Web page execution. Every element in an ASP.NET page is treated as an object and run on the server. An ASP.NET page gets compiled into an intermediate language by a .NET Common Language Runtime-compliant compiler. Then a JIT compiler turns the intermediate code to native machine code, and that machine code is eventually run on the processor. Because the code is run straight from the processor, pages load much faster than classic ASP pages, where embedded VBScript or JScrip that to be continuously interpreted and cached.

ASP.NET is used to create Web pages and Web services and is an integral part of Microsoft's .NET vision.

[**What ASP.NET Web Pages Help You Accomplish**](javascript:void(0))

Web application programming presents challenges that do not typically arise when programming traditional client-based applications. Among the challenges are:

1. **Implementing a rich Web user interface**   It can be difficult and tedious to design and implement a user interface using basic HTML facilities, especially if the page has a complex layout, a large amount of dynamic content, and full-featured user-interactive objects.
2. **Separation of client and server**   In a Web application, the client (browser) and server are different programs often running on different computers (and even on different operating systems). Consequently, the two halves of the application share very little information; they can communicate, but typically exchange only small chunks of simple information.
3. **Stateless execution**   When a Web server receives a request for a page, it finds the page, processes it, sends it to the browser, and then discards all page information. If the user requests the same page again, the server repeats the entire sequence, reprocessing the page from scratch. Put another way, a server has no memory of pages that it has processed—page are stateless. Therefore, if an application needs to maintain information about a page, its stateless nature can become a problem.
4. **Unknown client capabilities**    In many cases, Web applications are accessible to many users using different browsers. Browsers have different capabilities, making it difficult to create an application that will run equally well on all of them.
5. **Complications with data access**   Reading from and writing to a data source in traditional Web applications can be complicated and resource-intensive.
6. **Complications with scalability**   In many cases Web applications designed with existing methods fail to meet scalability goals due to the lack of compatibility between the various components of the application. This is often a common failure point for applications under a heavy growth cycle.

Meeting these challenges for Web applications can require substantial time and effort.

ASP.NET Web pages and the ASP.NET page framework address these challenges in the following ways:

1. **Intuitive, consistent object model**   The ASP.NET page framework presents an object model that enables you to think of your forms as a unit, not as separate client and server pieces. In this model, you can program the page in a more intuitive way than in traditional Web applications, including the ability to set properties for page elements and respond to events. In addition, ASP.NET server controls are an abstraction from the physical contents of an HTML page and from the direct interaction between browser and server. In general, you can use server controls the way you might work with controls in a client application and not have to think about how to create the HTML to present and process the controls and their contents.
2. **Event-driven programming model**   ASP.NET Web pages bring to Web applications the familiar model of writing event handlers for events that occur on either the client or server. The ASP.NET page framework abstracts this model in such a way that the underlying mechanism of capturing an event on the client, transmitting it to the server, and calling the appropriate method is all automatic and invisible to you. The result is a clear, easily written code structure that supports event-driven development.
3. **Intuitive state management**   The ASP.NET page framework automatically handles the task of maintaining the state of your page and its controls, and it provides you with explicit ways to maintain the state of application-specific information. This is accomplished without heavy use of server resources and can be implemented with or without sending cookies to the browser.
4. **Browser-independent applications**   The ASP.NET page framework enables you to create all application logic on the server, eliminating the need to explicitly code for differences in browsers. However, it still enables you to take advantage of browser-specific features by writing client-side code to provide improved performance and a richer client experience.
5. **.NET Framework common language runtime support**   The ASP.NET page framework is built on the .NET Framework, so the entire framework is available to any ASP.NET application. Your applications can be written in any language that is compatible that is with the runtime. In addition, data access is simplified using the data access infrastructure provided by the .NET Framework, including ADO.NET.

**C#**

C# is designed to be a platform-independent language in the tradition of Java (although it is implemented primarily on Windows). It's syntax is similar to C and C++ syntax, and C# is designed to be an object-oriented language. There are, for the most part, minor variations in syntax between C++ and C#. Main has no return type, there are no semicolons after class names, there are some (to C++ programmers) strange decisions regarding capitalization - such as the capitalization of Main. Other a few differences, the syntax is often the same. This decision is reasonable, in light of the fact that C syntax has been used with several other languages - notably Java.

Similar to Java, C# does not support multiple inheritances; instead it provides Java's solution: interfaces. Interfaces implemented by a class specify certain functions that the class is guaranteed to implement. Interfaces avoid the messy dangers of multiple inheritances while maintaining the ability to let several classes implement the same set of methods.

Another helpful feature of C# is garbage collection. Therefore, it is unnecessary to include a destructor for each class unless a class handles unmanaged resources; if so, it's necessary to release control those resources from within the class (The Finalize function is used to clear up these unmanaged resources; it can even be abbreviated with the same syntax as a C++ destructor). Of course, C# also provides direct access to memory through C++ style pointers, but these pointers are not garbage collected until specifically released by the programmer.

C#, as part of the .NET framework, is compiled to Microsoft Intermediate Language (MSIL), which is a language similar to Java's byte code. MSIL allows C# to be platform independent and runs using just in time compiling. Therefore programs running under .NET gain speed with repeated use. Furthermore, because the other languages that make up the .NET platform (including VB and COBOL) compile to MSIL

The potential for C# is great if the .NET platform succeeds. C# is designed to take advantage of the design of .NET, and Microsoft has poured a great deal of money into .NET. Do you need to learn C#? If you know C++, you'll probably be able to pick it up quickly, and yes, you can still use C++ with .NET. It's important to keep an eye on C# to see how it will affect you.

**SQL**

SQL Server is a relational database management system ([RDBMS](http://searchsqlserver.techtarget.com/definition/relational-database-management-system)) from [Microsoft](http://searchwinit.techtarget.com/definition/Microsoft) that's designed for the [enterprise](http://searchwinit.techtarget.com/definition/enterprise) environment. SQL Server runs on [T-SQL](http://searchsqlserver.techtarget.com/definition/T-SQL) (Transact -[SQL](http://searchsqlserver.techtarget.com/definition/SQL)), a set of programming [extension](http://searchcio-midmarket.techtarget.com/definition/extension)s from [Sybase](http://searchenterpriselinux.techtarget.com/definition/Sybase) and Microsoft that add several features to standard SQL, including transaction control, exception and [error handling](http://searchsoftwarequality.techtarget.com/definition/error-handling), row processing, and declared [variable](http://whatis.techtarget.com/definition/variable)s.

Code named [Yukon](http://searchsqlserver.techtarget.com/definition/Yukon) in development, SQL Server 2005 was released in November 2005. The 2005 product is said to provide enhanced flexibility, [scalability](http://searchdatacenter.techtarget.com/definition/scalability), reliability, and [security](http://searchsecurity.techtarget.com/definition/security) to database applications, and to make them easier to create and deploy, thus reducing the complexity and tedium involved in [database management](http://searchsqlserver.techtarget.com/definition/database-management-system). SQL Server 2005 also includes more administrative support.

.**List of Modules**

The lists of modules that are developed for Web Based Claim Processing System

1. Client
2. Sign up\Log in
3. Apply claim
4. Profile
5. Status
6. Download Documents
7. Forgot password
8. Admin
9. All permission
10. Approve/reject claim
11. Unique ID provider
12. Management
13. Provide Reference ID for client
14. Upload documents

**EXISTING SYSTEM**

1. Existing system is a semi automated system. Here the policy holder needs to apply for any claims he can able to approach the policy office manually.
2. Here the policy holder applies for his insurance claims through paper and submit necessary documents in Xerox copies. The insurer can receive the claim application from the policy holder and stores the information in local system database like ms word, excel etc.
3. To investigate a claim, the surveyor needs to collect the claim request through manually, and submit the claim process details also manually. It is a time delay process to solve more number of claim requests.

**PROPOSED SYSTEM**

1. The proposed system is fully automated system. Here the system needs to maintain the centralized database to store the information. Existing system maintain the data in local database only.
2. Our web-based technology empowers your Workers Compensation bill review process with up-to-the-minute information on claims, providers, payments, databases, tables and schedules ensures your claim re-pricing will net the absolute maximum amount allowable.

USER/

CUSTOMER

MANAGEMENT

ADMIN

DATA FLOW DIAGRAM

SEPARATE MODULES DFD

DB

DB

USER/

CUSTOMER

MANAGEMENT

Client

Details

Referance Id

User Id

Details

MANAGEMENT :

USER/CUSTOMER :

ADMIN :

DB

ADMIN

Controls

the Whole

Process

ER - DIAGRAM

CUSTOMER

MANAGEMENT

DB

ADMIN

DB

CUSTOMER

CUSTOMER :

DB

MANAGEMENT

MANAGEMENT :

DB

ADMIN

ADMIN :

**Figure Description**

**WEB BASED CLAIM PROCESSING SYSTEM – Main Page**

This is the Main page of web based claim processing system

It includes three types of functions, login for client, Admin and Management.

**SOURCE CODE**

**Login page**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.UI;

using System.Web.UI.WebControls;

using System.Data.SqlClient;

using System.Data.Odbc;

using System.Web.Configuration;

using System.Net.Mail;

using System.Collections;

using System.ComponentModel;

using System.Drawing;

using System.Text;

namespace koushik

{

public partial class WebForm2 : System.Web.UI.Page

{

DBAccess n = new DBAccess();

public string eno, name, job, comp, email, dob, add, mob, pwd, amt, d1, d2, d3, a, q,get;

// ArrayList a5 = new ArrayList();

DBAccess aa = new DBAccess();

protected void Page\_Load(object sender, EventArgs e)

{

}

protected void LoginButton\_Click1(object sender, EventArgs e)

{

string slog = aa.login(Login1.UserName, Login1.Password);

if (slog == "sucess")

{

Session["a"] = aa.empid;

Session["b"] = aa.cname;

Response.Redirect("clienthome.aspx");

}

}

protected void BtnSubmit\_Click(object sender, EventArgs e)

{

//eno = TxtEmpNo.Text;

pwd = TextBox1.Text;

name = TxtName.Text;

job = DdlDept.SelectedItem.Text;

comp = DdlDesignation.SelectedItem.Text;

email = TxtEmailID.Text;

d3 = DropDownList3.SelectedItem.Text;

add = TxtAddress.Text;

mob = TxtMobile.Text;

//amt = " 0";

d1 = DropDownList1.SelectedItem.Text;

d2 = DropDownList2.SelectedItem.Text;

dob = d1 + d2 + d3;

a = TextBox3.Text;

q = DropDownList4.SelectedItem.Text;

eno = n.newemp(pwd, name, job, comp, email, dob, add, mob,q,a);

Response.Write("<script>alert('use this employee id '+'" + eno + "' + 'for login')</script>");

TextBox1.Text = "";

TxtName.Text = "";

TxtEmailID.Text = "";

TxtAddress.Text = "";

TxtMobile.Text = "";

TextBox3.Text = "";

//Response.Redirect("clienthome.aspx");

}

}

}

**Client home**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.UI;

using System.Web.UI.WebControls;

namespace koushik

{

public partial class WebForm1 : System.Web.UI.Page

{

public string type, refid, remark, status, empid, reason,amt="0",rr;

//public static string empno;

DBAccess s = new DBAccess();

protected void Page\_Load(object sender, EventArgs e)

{

empid = Session["a"].ToString();

}

protected void BtnSubmit\_Click(object sender, EventArgs e)

{

//type = DdlType.SelectedItem.Text;

//amt = TxtAmount.Text;

refid = TextBox1.Text;

remark = TxtDescription.Text;

status="waiting";

reason = "";

rr= s.claim(empid,refid,remark,status,reason,amt);

if (rr == "success")

{

Response.Write("<script>alert('Request Sent Sucessfully')</script>");

}

else

Response.Write("<script>alert('Reference id is already exist')</script>");

TxtDescription.Text = "";

TextBox1.Text = "";

// TxtAmount.Text = "";

}

}

}

**Profile page**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.UI;

using System.Web.UI.WebControls;

namespace koushik

{

public partial class WebForm3 : System.Web.UI.Page

{

DBAccess p = new DBAccess();

public string empid,eno, name, job, comp, email, dob, add, mob, pwd, amt, d1, d2, d3;

protected void Page\_Load(object sender, EventArgs e)

{

empid = Session["a"].ToString();

p. cmd.Connection = p.con;

p.con.Open();

p.cmd.CommandText = "select \* from emp where empid='" + empid + "'";

p.dr = p.cmd.ExecuteReader();

while (p.dr.Read())

{

p.a1 = p.dr[2].ToString();

p.b1 = p.dr[5].ToString();

//p.c1 = p.dr[6].ToString();

p.d1 = p.dr[7].ToString();

p.e1 = p.dr[8].ToString();

}

p.con.Close();

TxtName.Text = p.a1;

TxtEmailID.Text = p.b1;

TxtAddress.Text = p.d1;

TxtMobile.Text = p.e1;

}

protected void BtnSubmit\_Click(object sender, EventArgs e)

{

pwd = TextBox1.Text;

name = TxtName.Text;

job = DdlDept.SelectedItem.Text;

comp = DdlDesignation.SelectedItem.Text;

email = TxtEmailID.Text;

d3 = DropDownList3.SelectedItem.Text;

add = TxtAddress.Text;

mob = TxtMobile.Text;

//amt = " 0";

d1 = DropDownList1.SelectedItem.Text;

d2 = DropDownList2.SelectedItem.Text;

dob = d1 + d2 + d3;

p.updateemp(empid,pwd, name, job, comp, email, dob, add, mob);

Response.Write("<script>alert('updated Sucessfully')</script>");

Response.Redirect("clienthome.aspx");

}

}

}

**Forgot password**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.UI;

using System.Web.UI.WebControls;

namespace koushik

{

public partial class forgot : System.Web.UI.Page

{

DBAccess n = new DBAccess();

string get;

protected void Page\_Load(object sender, EventArgs e)

{

}

protected void Button1\_Click(object sender, EventArgs e)

{

get = n.getpass(DropDownList5.SelectedItem.Text, TextBox4.Text, TextBox5.Text);

if (get == "fail")

{

Response.Write("<script>alert('Incorrect answer try again')</script>");

}

else

{

Response.Write("<script>alert('your password is '+'" + get + "')</script>");

Response.Write("<script>window.location.href='login.aspx';</script>");

}

}

}

}

**Status page**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.UI;

using System.Web.UI.WebControls;

using System.Data;

using System.Data.SqlClient;

using System.Data.Odbc;

using System.Web.Configuration;

using System.Net.Mail;

using System.Collections;

using System.ComponentModel;

using System.Drawing;

using System.Text;

namespace koushik

{

public partial class status : System.Web.UI.Page

{

public SqlConnection con;

public SqlCommand cmd;

public SqlDataReader dr;

public SqlDataAdapter da;

public DataSet ds;

public string x = "accepted", y = "rejected",empid;

protected void Page\_Load(object sender, EventArgs e)

{

ds = new DataSet();

con = new SqlConnection("Data Source=KILLER-PC\\SQLEXPRESS;Initial Catalog=insurance;Integrated Security=True");

cmd = new SqlCommand();

da = new SqlDataAdapter();

if (!Page.IsPostBack)

{

Bind();

}

}

public void Bind()

{

empid = Session["a"].ToString();

cmd.CommandText = "Select \* from claim where empid='"+empid+"' ";

cmd.Connection = con;

da = new SqlDataAdapter(cmd);

da.Fill(ds);

con.Open();

cmd.ExecuteNonQuery();

DataGrid1.DataSource = ds;

DataGrid1.DataBind();

}

protected void \_PageIndexChanged(object source, DataGridPageChangedEventArgs e)

{

DataGrid1.CurrentPageIndex = e.NewPageIndex;

Bind();

}

}

}

**ESI login page**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.UI;

using System.Web.UI.WebControls;

namespace koushik

{

public partial class hospitallogin : System.Web.UI.Page

{

protected void Page\_Load(object sender, EventArgs e)

{

}

protected void Login1\_Authenticate(object sender, AuthenticateEventArgs e)

{

if (Login1.UserName == "esi" && Login1.Password == "esi")

{

Response.Redirect("hospital.aspx");

}

else

{

Response.Write("<script>alert('Incorrect username and password')</script>");

}

}

}

}

**ESI main page**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.UI;

using System.Web.UI.WebControls;

using System.Data.SqlClient;

using System.Data.Odbc;

using System.Web.Configuration;

using System.Net.Mail;

using System.Collections;

using System.Collections.Generic;

using System.ComponentModel;

using System.Drawing;

using System.Text;

namespace koushik

{

public partial class hospital : System.Web.UI.Page

{

public SqlConnection con;

public SqlCommand cmd;

public SqlDataReader dr;

public SqlDataAdapter da;

public string name,rr;

protected void Page\_Load(object sender, EventArgs e)

{

con = new SqlConnection("Data Source=KILLER-PC\\SQLEXPRESS;Initial Catalog=insurance;Integrated Security=True");

cmd = new SqlCommand();

da = new SqlDataAdapter();

}

protected void Button2\_Click(object sender, EventArgs e)

{

name = TextBox1.Text;

cmd.Connection = con;

con.Open();

cmd.CommandText = "select \* from esi ";

dr = cmd.ExecuteReader();

while (dr.Read())

{

if (name == (dr[0].ToString()))

{

rr = "exist";

break;

}

else

{

rr = "success";

}

}

con.Close();

if (rr == "success")

{

cmd.Connection = con;

con.Open();

//Convert pdf in Binary formate

int lenght = FileUpload1.PostedFile.ContentLength;

byte[] data = new byte[lenght];

FileUpload1.PostedFile.InputStream.Read(data, 0, lenght);

using (cmd = new SqlCommand("insert into esi (refid,data)" + " values(@name,@data)", con))

{

cmd.Parameters.Add("@data", data);

cmd.Parameters.Add("@name", name);

cmd.ExecuteNonQuery();

TextBox1.Text = "";

Response.Write("<script>alert('file is Sucessfully sent to admin')</script>");

}

}

else

Response.Write("<script>alert('already exists')</script>");

}

}

}

**Admin login page**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.UI;

using System.Web.UI.WebControls;

namespace koushik

{

public partial class adminlogin : System.Web.UI.Page

{

protected void Page\_Load(object sender, EventArgs e)

{

}

protected void Login1\_Authenticate(object sender, AuthenticateEventArgs e)

{

if (Login1.UserName == "admin" && Login1.Password == "admin")

{

Response.Redirect("admin.aspx");

}

else

{

Response.Write("<script>alert('Incorrect username and password')</script>");

}

}

}

}

**Admin main page**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.UI;

using System.Web.UI.WebControls;

using System.Data;

using System.Data.SqlClient;

using System.Data.Odbc;

using System.Web.Configuration;

using System.Net.Mail;

using System.Collections;

using System.ComponentModel;

using System.Drawing;

using System.Text;

namespace koushik

{

public partial class admin1 : System.Web.UI.Page

{

public SqlConnection con;

public SqlCommand cmd;

public SqlDataReader dr;

public SqlDataAdapter da;

public DataSet ds;

public string x = "accepted", y = "rejected";

protected void Page\_Load(object sender, EventArgs e)

{

ds = new DataSet();

con = new SqlConnection("Data Source=KILLER-PC\\SQLEXPRESS;Initial Catalog=insurance;Integrated Security=True");

cmd = new SqlCommand();

da = new SqlDataAdapter();

if (!Page.IsPostBack)

{

BindData();

}

}

public void BindData()

{

cmd.CommandText = "Select \* from claim where status='waiting'";

cmd.Connection = con;

da = new SqlDataAdapter(cmd);

da.Fill(ds);

con.Open();

cmd.ExecuteNonQuery();

DataGrid1.DataSource = ds;

DataGrid1.DataBind();

}

protected void Button1\_Click(object sender, EventArgs e)

{

con.Open();

SqlCommand com = new SqlCommand("select data from esi where refid='"+TextBox1.Text+"'", con);

//com.Parameters.AddWithValue("id",TextBox1.Text);

SqlDataReader dr = com.ExecuteReader();

if (dr.Read())

{

Response.Clear();

Response.Buffer = true;

Response.ContentType = "application/pdf";

Response.AddHeader("content-disposition", "attachment;filename=" + dr[0].ToString() + "." + "pdf");

// to open file prompt Box open or Save file

Response.Charset = "";

Response.Cache.SetCacheability(HttpCacheability.NoCache);

Response.BinaryWrite((byte[])dr["data"]);

Response.End();

con.Close();

TextBox1.Text = "";

}

}

protected void \_PageIndexChanged(object source, DataGridPageChangedEventArgs e)

{

DataGrid1.CurrentPageIndex = e.NewPageIndex;

BindData();

}

protected void DataGrid1\_UpdateCommand(object source, DataGridCommandEventArgs e)

{

cmd.Parameters.Add("@refid", SqlDbType.Char).Value = ((TextBox)e.Item.Cells[1].Controls[0]).Text;

cmd.Parameters.Add("@reason", SqlDbType.Char).Value = ((TextBox)e.Item.Cells[3].Controls[0]).Text;

cmd.Parameters.Add("@amt", SqlDbType.Char).Value = ((TextBox)e.Item.Cells[4].Controls[0]).Text;

cmd.CommandText = "Update claim set status='"+x+"',reason=@reason,amt=@amt where refid=@refid";

cmd.Connection = con;

cmd.Connection.Open();

cmd.ExecuteNonQuery();

cmd.Connection.Close();

DataGrid1.EditItemIndex = -1;

BindData();

}

protected void DataGrid1\_DeleteCommand(object source, DataGridCommandEventArgs e)

{

cmd.Parameters.Add("@refid", SqlDbType.Char).Value = ((TextBox)e.Item.Cells[1].Controls[0]).Text;

cmd.Parameters.Add("@reason", SqlDbType.Char).Value = ((TextBox)e.Item.Cells[3].Controls[0]).Text;

//cmd.Parameters.Add("@amt", SqlDbType.Char).Value = ((TextBox)e.Item.Cells[5].Controls[0]).Text;

cmd.CommandText = "Update claim set status='" + y + "',reason=@reason where refid=@refid";

cmd.Connection = con;

cmd.Connection.Open();

cmd.ExecuteNonQuery();

cmd.Connection.Close();

DataGrid1.EditItemIndex = -1;

BindData();

}

protected void DataGrid1\_EditCommand(object source, DataGridCommandEventArgs e)

{

DataGrid1.EditItemIndex = e.Item.ItemIndex;

BindData();

}

protected void DataGrid1\_CancelCommand(object source, DataGridCommandEventArgs e)

{

DataGrid1.EditItemIndex = -1;

BindData();

}

}

}

SNAP SHOTS

Main Page

Sign up

Unique ID provided by Admin for login to User

Login Page

Client Home Page

Applying for Claim

Request sends successfully to Admin

Profile of User

Status view

Forgot Password

ESI (management) login

Uploading files

Browsing file(pdf)

File Browsed

File sent successfully

Admin Login

Viewing unchecked files

Downloading files of user

Downloaded files

Accept user request claim

Reject user request

Others Request

Login Page

**CONCLUSION**

The web based claim processing system is fully automated system and thus it will decrease the human effort. Web based claim processing system is very useful and it will ease the working of the people. No need of submission of documents in the respective office instead they can claim it online. The web based claim processing system is robust and reliable.

Bibliography & References

These are the links which assisted us at each and every step in completing this project, without them we would have not been be able to finish this important project.

References

1. www.projecttopics.info
2. msdn.microsoft.com
3. Feedbacks from different sites
4. google.com