2005

A web application for Medasolution Healthcare Company customer service system

Hao Jia

A WEB APPLICATION FOR MEDASOLUTION HEALTHCARE COMPANY CUSTOMER SERVICES SYSTEM

A Project

Presented to the

Faculty of

California State University,

San Bernardino

In Partial Fulfillment

of the Requirements for the Degree

Master of Science

in

Computer Science

---

by

Hao Jia

March 2005

A WEB APPLICATION FOR MEDASOLUTION HEALTHCARE COMPANY CUSTOMER SERVICES SYSTEM

A Project

Presented to the

Faculty of

California State University,

San Bernardino

---

by

Hao Jia

March 2005

Approved by:

[redacted]

Dr. David Turner, Chair, Computer Science — Dec 10, 2004 Date

[redacted]

Dr. Richard John Botting

[redacted]

Dr. Yasha Karant

ABSTRACT

This project is a web application implemented using ASP.Net and SQL Server 2000. The purpose is to implement the communication between the Medasolution Health Care Company and its clients. Medasolution is a virtual company designed by the author to handle medicare insurance business, which has four categories of clients: members, employers, brokers and medicare providers. Each client can use the web interfaces as a tool to manage their data in the Medasolution database system. This project allows clients to enter the web pages based on their category level. Besides, Medasolution strictly follows the governments privacy rules and keeps its client information from being used outside the Medasolution.

This document not only describes the details of the project in both application mechanism and functionality, but also presents the source code and explanations.

ACKNOWLEDGEMENTS

I would first like to thank Dr. David Turner, my Project Advisor, for his patience by giving me the enough time to finish my project. I would like to appreciate Dr. Richard Botting and Dr. Yasha Karant. They gave me precious advice after I began to write my Software Requirements Specification, even when they were very busy. Their help made me believe that I have the ability to do my project. Although writing always seems like the hard part, the real work is done mostly because of the all help coming from these three professors.

I would like to thank my wife, Xueyuan Yu for giving me endless support and encouragement for my Project. Without her support I cannot imagine that I can finish this project. Besides, I would like to thank my friends, Yibin Jiang, Ray Yang, Carol Lau, Peng Tao, Lily Liu, Aileen Lo and Yang Yang for their help.

Last, and most important, without the support of my Mom and Dad and my sister Hai Jia, and brother Jiang Liang, I would never have been able to do this.

TABLE OF CONTENTS

LIST OF TABLE

LIST OF FIGURES

CHAPTER ONE

INTRODUCTION

Purpose of Project

The main purpose of this project is to provide friendly and simple web interfaces to Medasolution clients: members, employers, brokers and medicare providers. They can use web interfaces to communicate with Medasolution as well as fulfill their functionalities. This document not only gives a detailed look from the client perspective in order to achieve clarity for users, but also takes a look from the designer perspective.

Scope of Project

This project is a web application that integrates ASP.Net, Visual Basic.Net, and ADO.Net, and supported by the Internet Information Server 5.1 server engine. The web services depend on the ability of parties to communicate with each other, even if they use different system platforms or different data formats. In this Project a virtual publicly traded health care company -- Medasolution Health Care

Company -- is built up to implement this web services. This Project is designed for various clients: members, employer, brokers, and medical providers.

Privacy Rule

This project completely abides by the standards for Privacy of Individually Identifiable Health Information ("Privacy Rule") issued by The U.S. Department of Health and Human Services ("HHS"). The Privacy Rule implements the requirement of the Health Insurance Portability and Accountability Act of 1996 ("HIPAA"). The major goal of the Privacy Rule is to assure that that individuals' health information is properly protected while allowing the flow of health information needed to provide and promote high quality health care and to protect the public's health and well being.

According to the Privacy Rule, all Medasolution information is protected. Whatever the information is held or transmitted in any form of media, whether electronic, paper or oral.

This project as a research, also abides by the National

Institute of Health ("NIH") regulation. All protected health information used in this project will be used or dislosed under the Privacy Rule.

For more information on exercising clients rights set out in this notice, look at Medasolution web page. The clients may also call the free toll phone number listed on the web and ask for Medasolution privacy official for this purpose.

If the clients believe that Medasolution has violated their privacy rights set out in this notice, they may file a complaint with the Medasolution at the following address:

Privacy Complaint

United States Office of Personal Management

P.O. Box 707

Washington, DC 20004 - 0707

Filing a complaint will not affect client benefits under the Medasolution program. The client also may file a complaint with the secretary of the United States Department of Health and Human Services.

CHARPTER TWO

TECHNOLOGIES

A Web application is a dynamic extension of a Web server. There are two types of Web applications:

Presentation-oriented. A presentation-oriented Web application generates dynamic Web pages containing various types of markup language (HTML, XML, and so on) in response to requests.

Service-oriented. A service-oriented Web application implements the endpoint of a fine-grained Web service. Service-oriented Web applications are often invoked by presentation-oriented applications.

In this project we only use presentation-oriented type.

.Net Framework

The .Net Framework is a development and execution environment that allows different programming languages and libraries to work together seamlessly to create Windows-based applications that are easier to build, manage, deploy, and integrate with other networked systems.

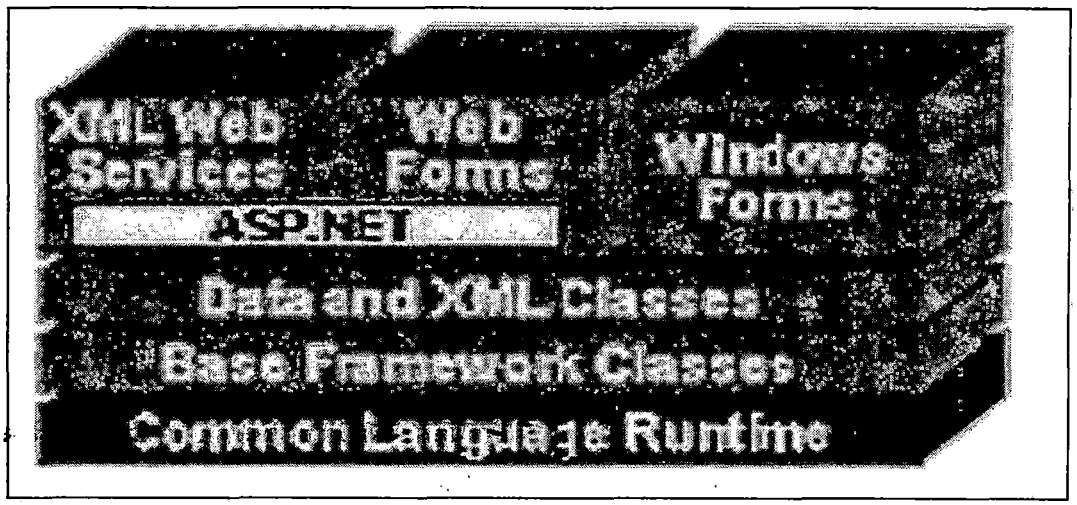

Figure 1. .Net Framework Infrastructure

The .Net Framework provides the basic infrastructure that Windows-based applications need to make Microsoft's .Net vision of connecting information, people, systems, and devices a reality. Figure 1 clearly shows the relationship between tiers.

Visual Basic.Net

Visual Basic.Net is one of the most popular programming languages in the world. Visual Basic .Net combines the power of the .Net Framework and the common language runtime with the productivity benefits that are the hallmark of Visual

Basic. Although the Visual Basic .Net language looks the same on the surface, the internal implementation of the language and the compiler has evolved significantly since Visual Basic 6.0.

Visual Basic.Net is not a language that stands in isolation—it is the latest offering in the Visual Basic product line. Visual Basic as a language and development tool has a strong history and a large following. Two major design goals for Visual Basic.Net were to preserve identical functionality for identical language keywords and to provide the same types of productivity enhancements that have made previous versions of Visual Basic so popular.

ADO.Net

In this project, we use Microsoft SQL Server 2000 as data source. ADO.Net not only provides consistent access to SQL Server, but also allows us to connect to these data sources and retrieve, manipulate and update data. ADO.Net includes .Net Framework data providers for connecting to a database, executing commands, and retrieving results. Those results are either processed directly, or placed in an ADO.Net dataset object.

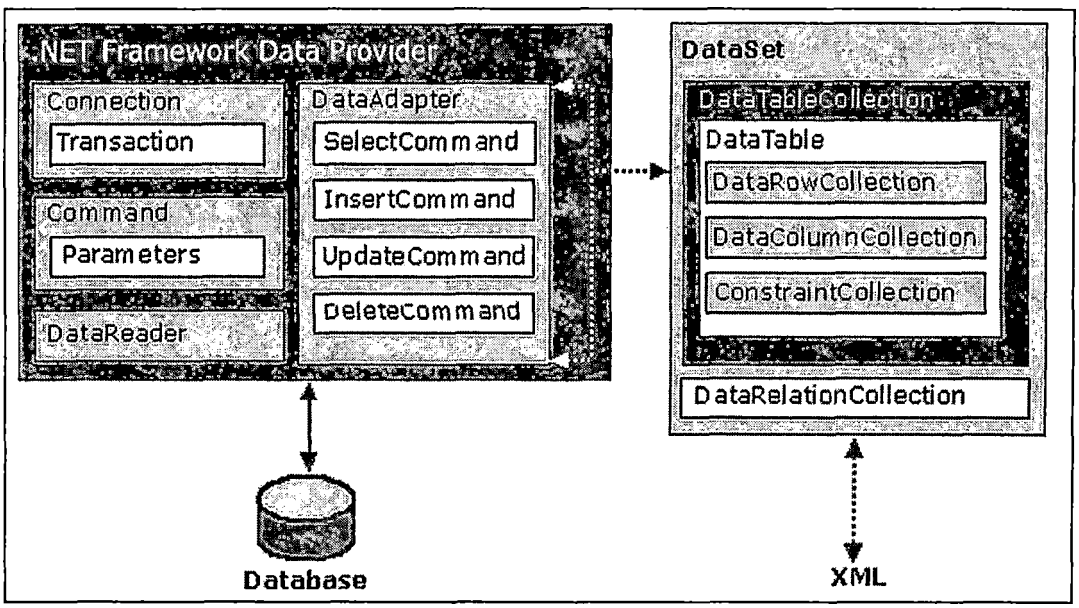

Figure 2. ADO.Net Architecture

The ADO.Net components have been designed to factor data access from data manipulation. There are two central components of ADO.Net that accomplish this: the DataSet, and the .Net Framework data provider, which is a set of components including the Connection, Command, DataReader, and DataAdapter object. Figure 2 shows the architecture. The ADO.Net DataSet is core component of disconnected architecture of ADO.Net. The DataSet contains a collection of one or more DataTable objects made up of rows and columns of data.

JavaScript

JavaScript is a script language which can be included within an HTML document, and are then executed by the Web browser when the document is loaded. A similar scripting language, known as VBScript, has been developed by Microsoft, however, it does not work in Netscape explorer. Therefore, we adopt Javascript to handle all client-side functionalities.

ASP.Net

ASP.Net is one of the most popular web development languages in the world; it provides a unified Web development model that includes the services necessary for developers to build enterprise-class Web applications. ASP.Net is built up based on the .Net Framework. Its architecture makes the ASP.Net to optimize the Windows operating system. As Figure 3 illustrates, all web clients communicate with ASP.Net applications through Microsoft Internet Information Services (IIS). It also provides a programming model and infrastructure for more scalable and stable applications that help provide greater protection.

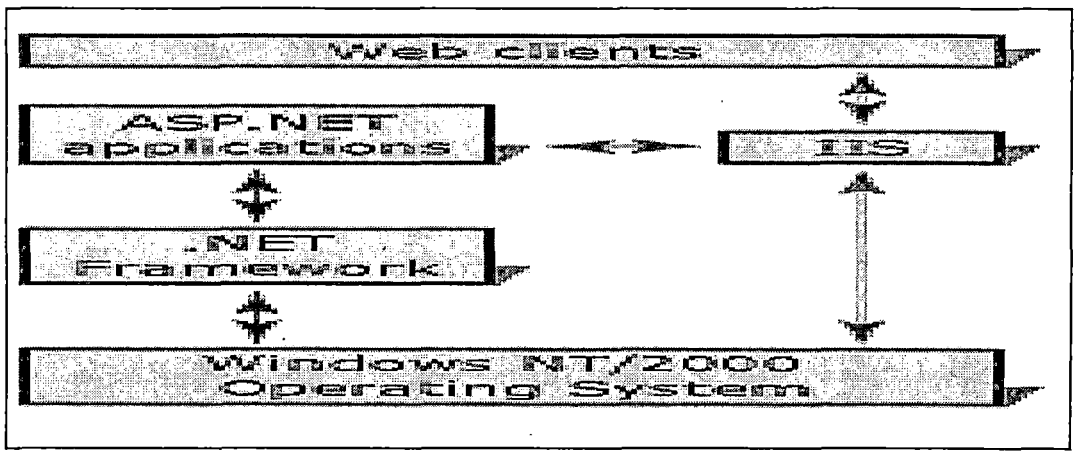

Figure 3. ASP.Net Architecture

ASP.Net is a compiled, .Net-based environment; it authors applications in any .Net compatible language, including Visual Basic.Net, C#, and JScript.Net. Additionally, the entire .Net Framework is available to any ASP.Net application.

CHAPTER THREE

PROJECT DESIGN

Overall Description

Background in Medicare Market

The healthcare program, created in 1965, provides health insurance to about one out of every seven Americans. In 1998, Medicare spent about $217 Billion for health care services, an average of about $5,500 per beneficiary. Over the past 30 years, medicare is playing more and more important roles in our life with the advance of industrial society.

There are several facts that illustrate the current situation in the medicare field. First, medicare is a federal health insurance program that provides 39 million elderly and disabled Americans access to high-quality health care. More than any other population group in this country, Medicare beneficiaries enjoy real health care security, because their coverage cannot be lost or taken way. Second, medicare has been a leader in cost containment. From the late 1960s, when Medicare was successfully implemented, to 1997, the most recent year for which data are available, the growth in

spending per beneficiary has been greater than in the private health insurance industry. Third, medical expending in the family has dramatically increased, especially for the beneficiaries age 65 and above who spend 19 percent of their income, on average, for health care and supplemental insurance policies to augment their medical care coverage.

To help pay for services not covered by medicare, most beneficiaries have some types of supplemental insurance. Most people either receive retiree coverage through a former employer or purchase a supplemental "medigap" policy. Under such circumstances, the medical insurance costs have greatly grown recently to satisfy the medicare beneficiaries' requirements.

Medasolution Introduction

Medasolution, designed as a virtual publicly traded managed health care company, is located in Riverside County of California State. Its mission is to help people be healthy, secure and comfortable. As a respected health care company in California, Medasolution has an extensive network for more than 4000 physicians, and serves up to 200,000 members. In order to extend its customer service and be convenient for

its customers, the Board of the company makes a new strategy that provides online services for customers.

The main clients of Medasolution can be classified into four categories: i) members, who are the end users of Medasolution company, as well as the medical insurance beneficiaries. ii) Employer or Business Group, whose sizes can range from small business to mid-market employer, even to the large companies which have demand in medical insurance for its employees. iii) Broker, who is the media between Medasolution and its end users. One role of the broker is to help clients and employees choose the carrier so they are virtually guaranteed to find a doctor they want and the coverage they need. The other role is to report feedback from clients regarding services. iv) Medicare Provider, who is the provider for members.

Product Perspective

This project is web-based, and will be integrated with .Net Framework. The hardware interface requirement is that it must run on the existing web server or local host. The software interface requirement is that it must support

current versions of Internet Explorer or Netscape. The communication interface requirement is that it must support HTTP. The system will be opened 24 hours a day, 7 days per week. All actions are user initiated. No separate backup and recovery or maintenance functions are required as that is handled by system administration on the hosting server machine.

Software Interface

Medasolution system's Software interfaces are provided by Microsoft operating system, Internet Information Services 5.1 for Active Server Page.Net (ASP.Net), Visual Basic.Net compiler version 7.00.9951 for compiling Visual Basic file, .Net Framework version 1.00.3705.6018 for supporting ASP.Net pages. Besides, the Medasolution system user needs to access the system by using JavaScript compatible web browser such as Microsoft IE and Netscape.

Table 1. Software Interfaces

Software	System
Operating System	Windows 2000(professional, Server, and Advanced Server), and Windows Server 2003 Family for both client and server application
Web Browser	Microsoft IE 5.0 or above Netscape
Web Server	IIS 5.11
VB Compiler	V7.00.9951
ASP.Net Platform	V1.00.3705.6018
Database System	Microsoft SQL Server 2000
JavaScript	JavaScript 1.3

Hardware Interface

Because the Windows operating system already handles the hardware interfaces for this project. Thus, there is no implementation related to hardware in this project. The Table 1 shows the summarization of software interfaces.

Communication Interface

The communication interfaces are HTTP for general information. Private information is encrypted using SSL protocols. Communication between application and database SQL Server 2000 is through ADO.net bound in the Visual Studio.Net software.

Memory Constraints

There is no specific memory requirement for the client computer. Although there is no explicit memory requirement for the web server and database server, enough memory is required to guarantee acceptable response time. 128 Mega bytes or higher are recommended.

Operations

Users should access Medasolution through the World Wide Web. The logon session can expire if the user is inactive for 40 minutes. It is set in the web server via configuration file.

Assumptions and Dependencies

In this project, there are following assumptions:

It is assumed that it has a medicare company named by Medasolution which has a web server running for its clients.

In the Medasolution databases, there are some data such as members, employers, brokers, and medicare providers.

It also supposed that the Medasolution allows its registered clients to modify their information stored in the Medasolution databases.

Project Architecture Design

Medasolution is a 3-Tier architecture. The entire process begins with a web browser making a request for an ASP.Net web page, and finishes with the HTTP handler generating the suitable response as output for the web requester. Figure 4 shows a typical 3-tier architecture scenario.

Data Tier. This tier is responsible for retrieving, storing and updating information, therefore this tier can be ideally implemented through a commercial database.

Logic Tier. This tier is the brain of application. It interacts with the business tier and data access tier. In this sub tier it containts classes to calculate aggregated values, such as total members per medicare provider, etc. This tier does not know about GUI controls and how to access database. Data access tier acts as an interface to the database. This tier knows how to retrieve and store information from the database.

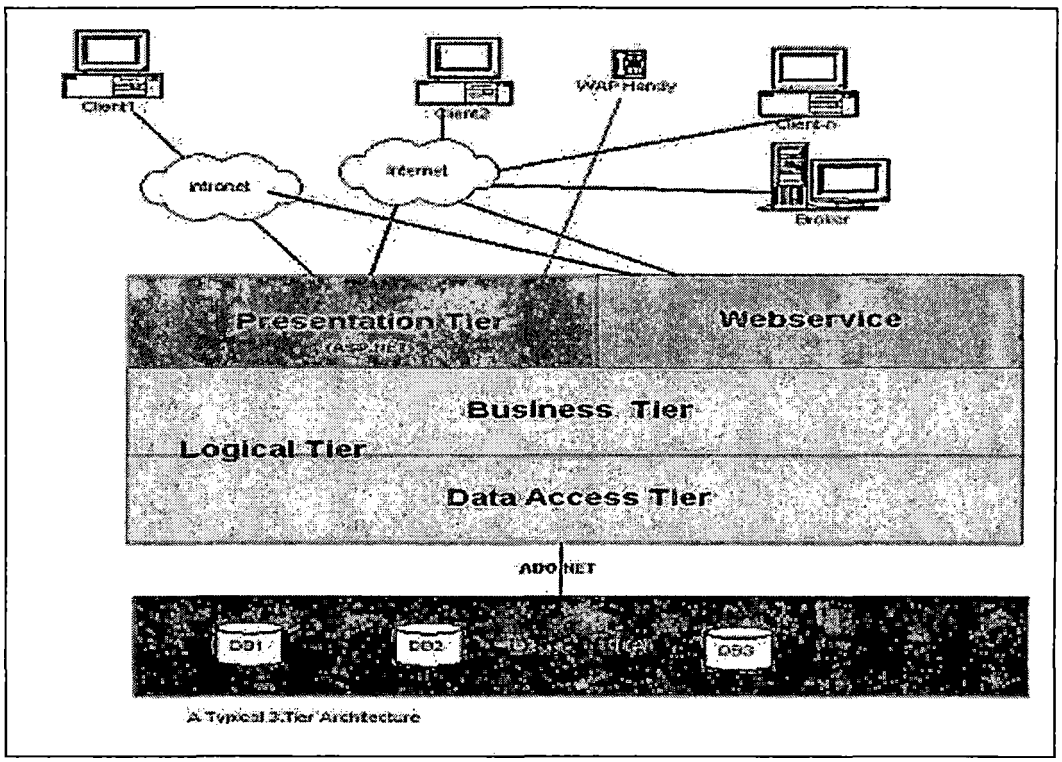

Figure 4. Medasolution Architecture Design

Presentation Tier. This tier is responsible for communication with users and web service consumers, and it uses objects from the business tier to respond to GUI raised events.

Medasolution System Design

Use Case Diagram

The whole system is separated into four parts based on the four client categories. Figure 5 shows the use case

1diagram that graphically depicts the users and principal functions of the Medasolution system.

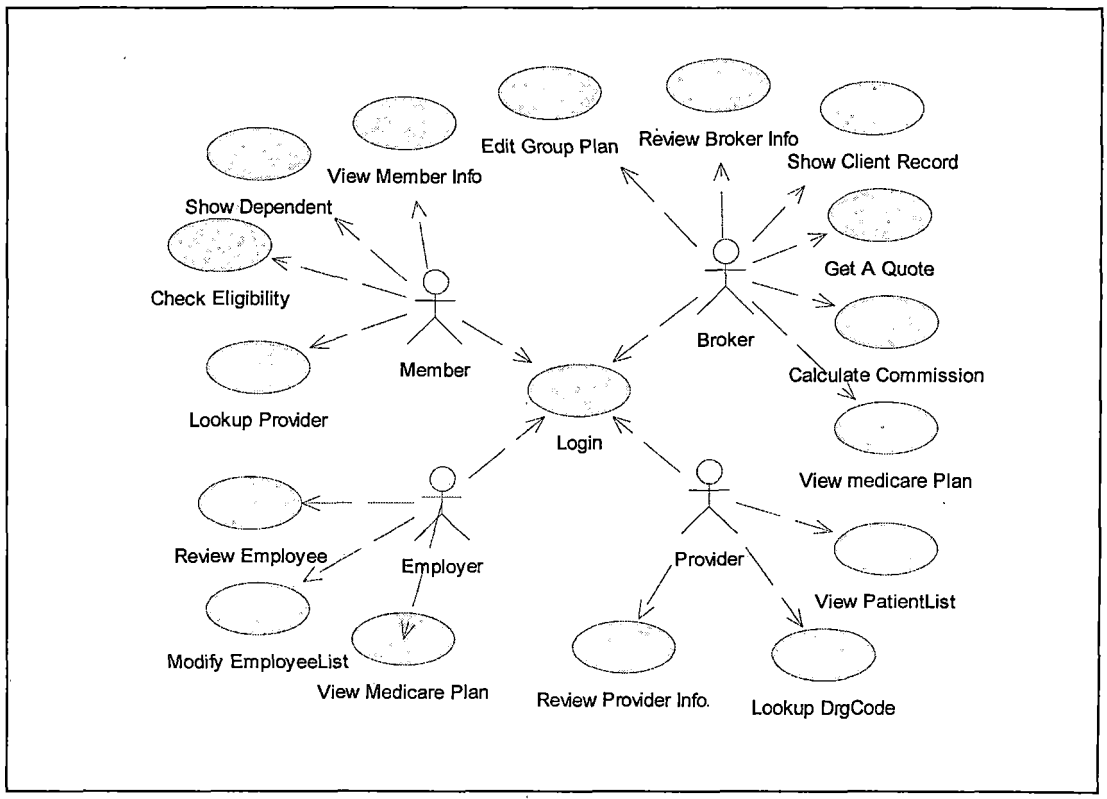

Figure 5. Medasolution Use Case Diagram

Web Page Design

In this Project the banner of Medasolution is designed using Macromedia Firework MX 2004, which is a flexible and powerful graphical environment in which both bitmapped and vector images may be used to generate artwork for web pages. Macromedia Firework generates the animation to promote the

Medasolution's name. The logo of Medasolution which is used in the banner is processed by Flash MX 2004. All Medasolution web pages are designed by Macromedia Dreamweaver MX 2004.

Graphical User Interfaces and Description

The user interfaces for Medasolution are designed as HTML pages. The contents are generated dynamically by ASP.Net or JavaScript in response to user's request. The following features incorporated to produce a more descriptive representation of the interface.

Home Page. This Page not only shows hyperlinks for registered users, but also provides a login section for members, employers, brokers, and providers. If the username and password are incorrect, the system will generate error message to remind the user to input again. Figure 6 shows the home page.

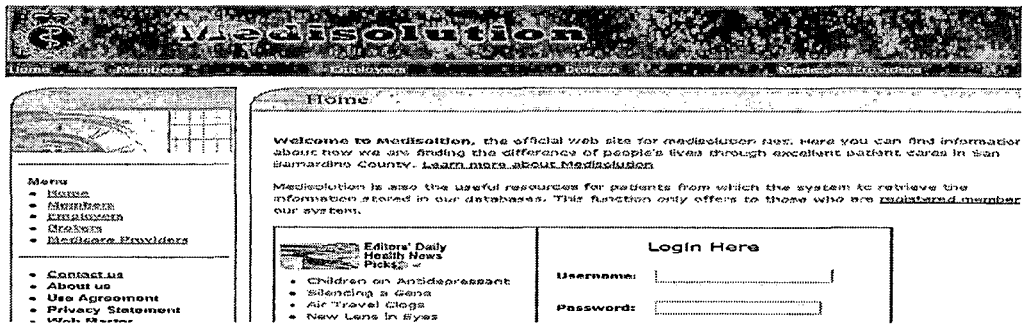

Figure 6. Home Page

Member Home Page. Only registered member users can reach this page. It allows members four hyperlinks to view specific information. This is the home page for the member users. Figure 7 shows the member home page.

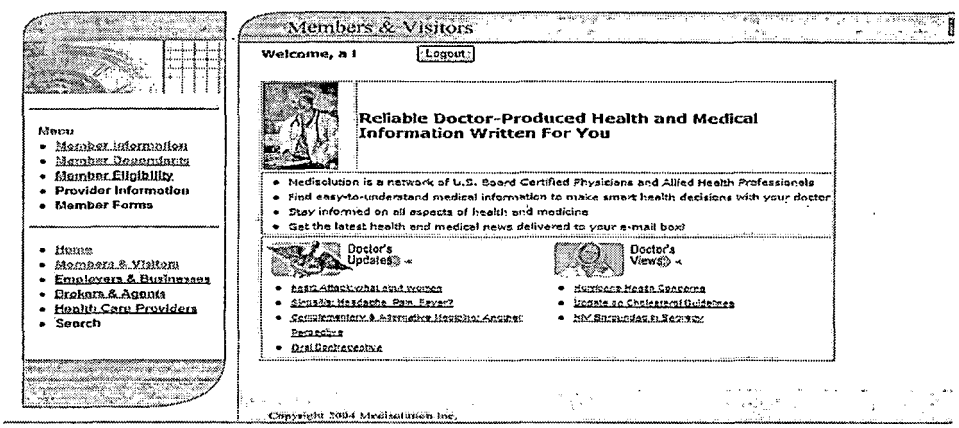

Figure 7. Member Home Page

Member Personal Information Page. It contains two functions: one is to provide the full contents of the member's information, so the member can view his/her information; the other is to allow the member to edit his/her information if needed and then save to the database. Figure 8 shows the member personal information page.

Member Dependent Information Page. It allows the member to view all his/her dependents or modify his/her dependents. Besides, the member can also add new dependent(s) into the

Medasolution database. Figure 9 shows the dependent information page.

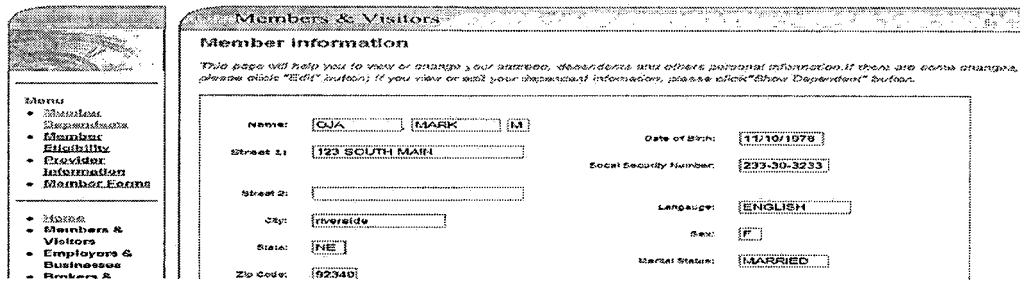

Figure 8. Personal Information Page

Member Provider Information Page. In this page it provides two functions: one is to show the current specialty doctor; the other is to provider the option of specialty category and speaking language for the member to select in the list. Figure 10 shows the two functions in the provider page.

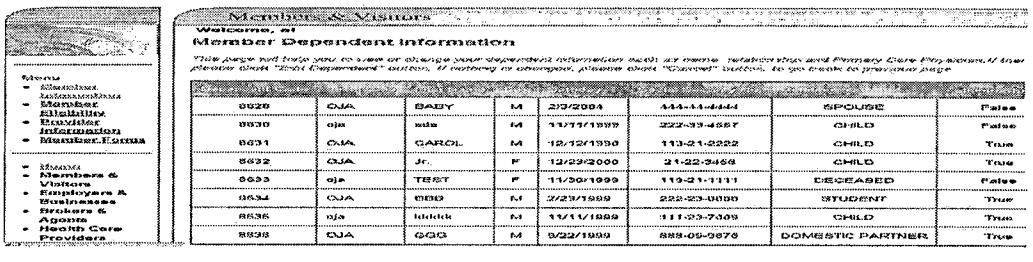

Figure 9. Dependent Information Page

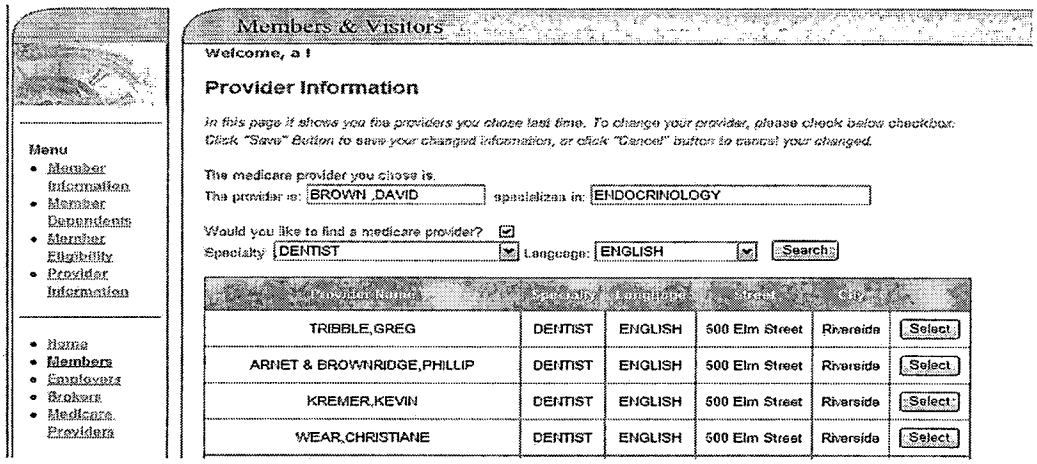

Figure 10. Provider Information Page

Member Eligibility Page. It shows member's basic and his/her eligibility information of the benefit plan and history records. Figure 11 shows the member eligibility page.

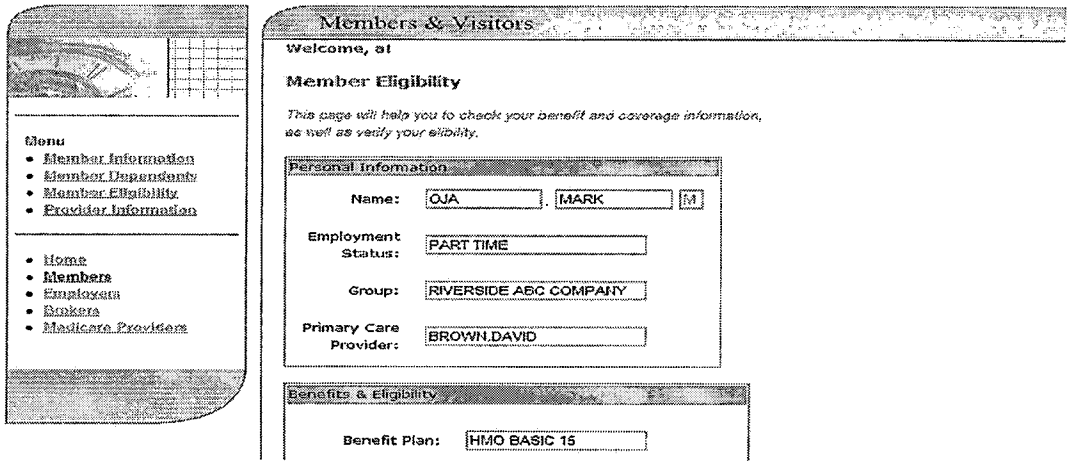

Figure 11. Eligibility Page

Employer Home Page. Only registered employer users can

reach this page. It allows employers to view other related information stored in the database. Figure 12 shows the employer home page. It also provides five hyperlinks for the employer.

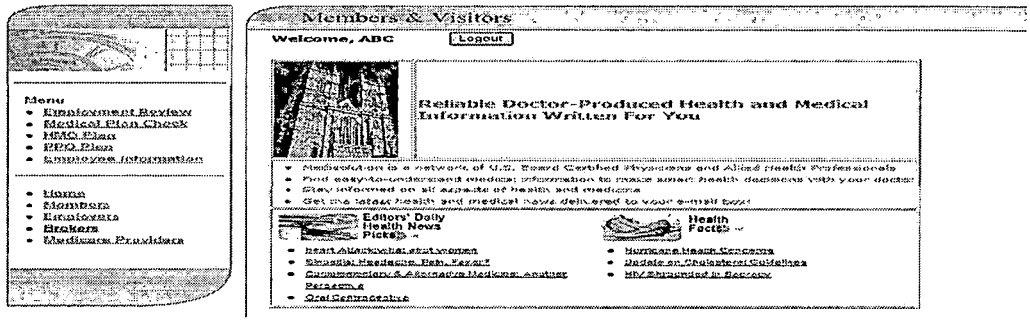

Figure 12. Employer Home Page

Employer Review Page. In this page it shows the employer information such primary contact such the employer can input new contact person. Figure 13 shows the detailed page.

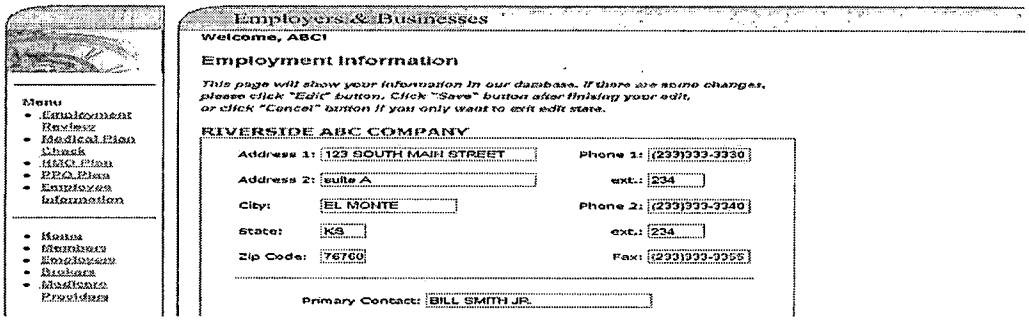

Figure 13. Employer Information Page

Employer Medicare Plan Page. It provides all detailed medicare plan offered by Medasolution including premiums and plan contents. Figure 14 shows the medicare plan page.

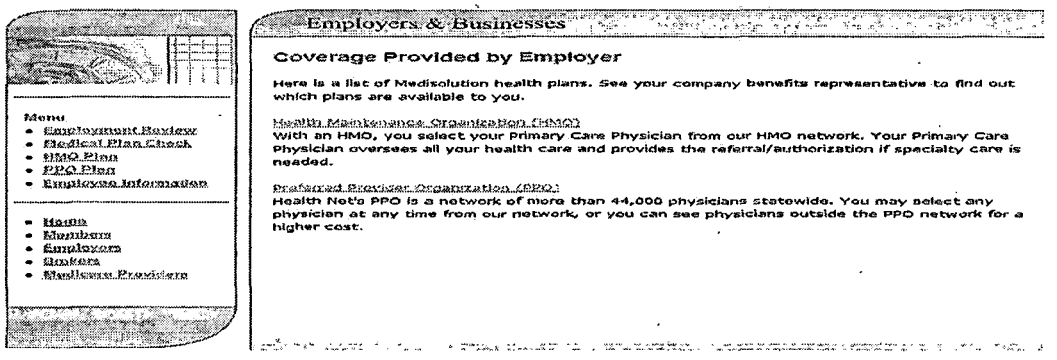

Figure 14. Medicare Plan Page

Health Maintenance Organization Plan Page. In this page, it illustrates who can participate this plan. Figure 15 shows the detailed.

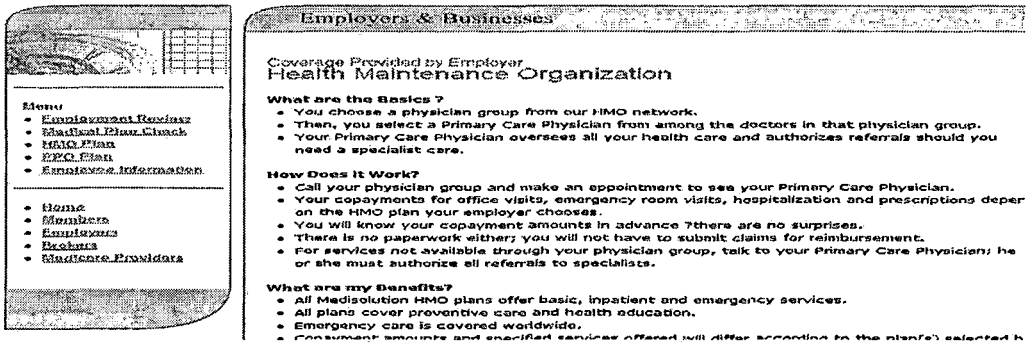

Figure 15. Health Maintenance Organization Plan Page

Preferred Provider Organization Plan Page. In this page, it illustrates that who can participate this plan. Figure 16 shows the detail page.

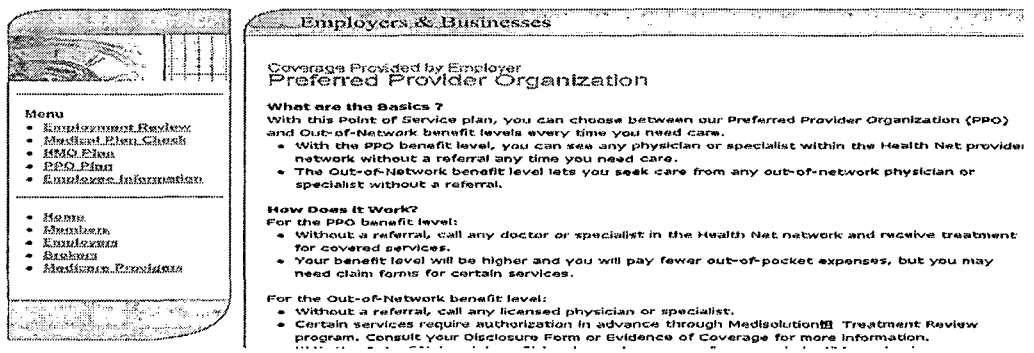

Figure 16. Preferred Provider Organization Plan Page

Broker Home Page. Only registered broker users can reach this page. It allows a broker to browse his/her information by clicking the links in the page. Figure 17 shows the broker home page.

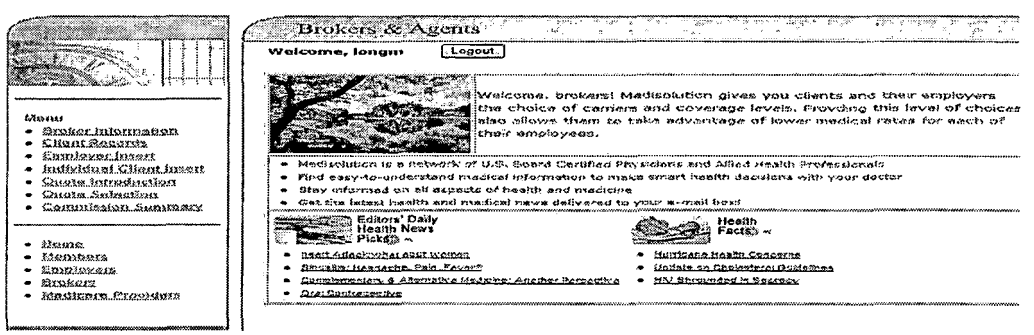

Figure 17. Broker Home Page

Broker Information Page. In this page, a broker can view all its information. It also allows the broker to modify the information by clicking the edit button on the page. Figure 18 shows the broker information page.

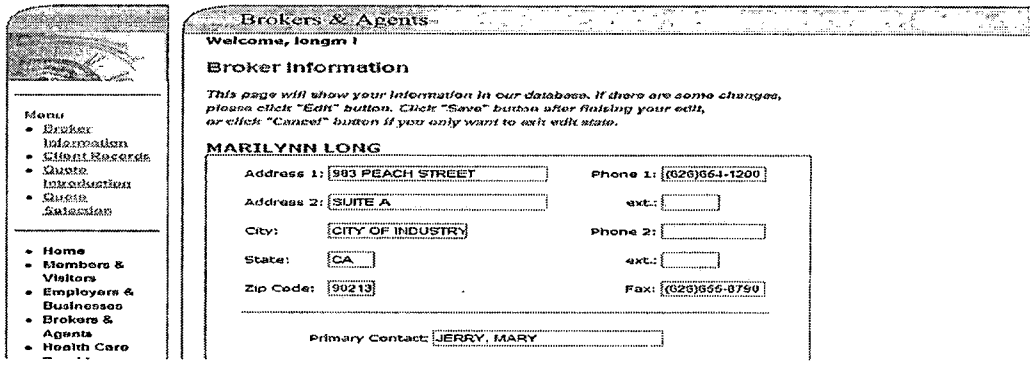

Figure 18. Broker Information Page

Broker Pre-Quote Page. When the broker enters this page, it has two hyperlinks and a button. Two links are for the detailed benefit plans Figure 19 shows the detailed page.

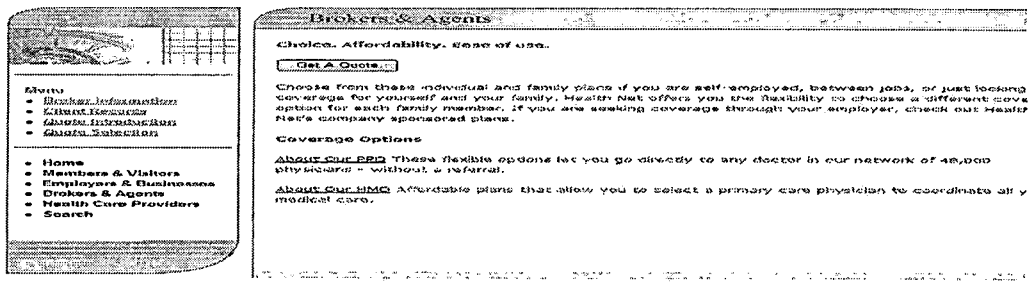

Figure 19. Pre-Quote Page

Broker Get Quote Page. It shows a broker a page which allows the broker to quote premium based on the client's requirement. The broker can input the client's birthday and benefit plan type for narrow down the selection. Figure 20 shows the get quote page.

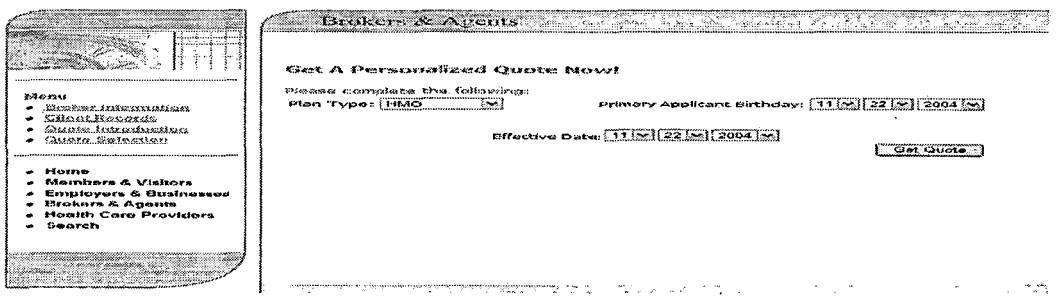

Figure 20. Get A Quote Page

Broker Calculate commission page. It shows the brokers how much commission he/she earns and earning source. Figure 21 shows the calculation page.

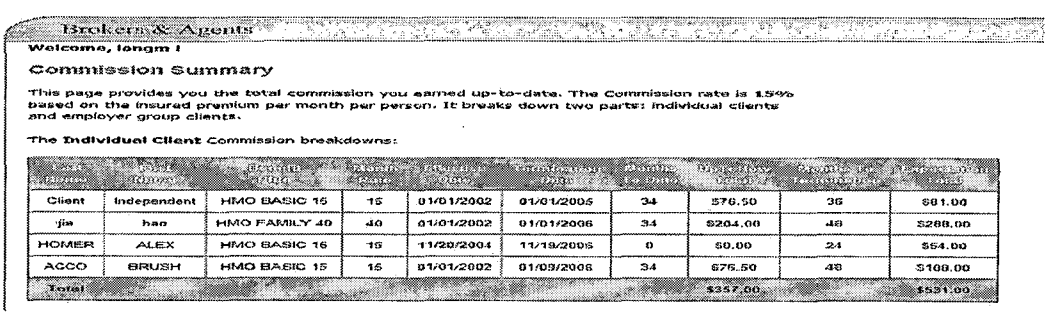

Figure 21. Calculate Commission Page

Broker Client Record Page. In this page it shows the broker all clients developed during a period of time. All clients are classified as individual and employer. Figure 22 shows the client record page.

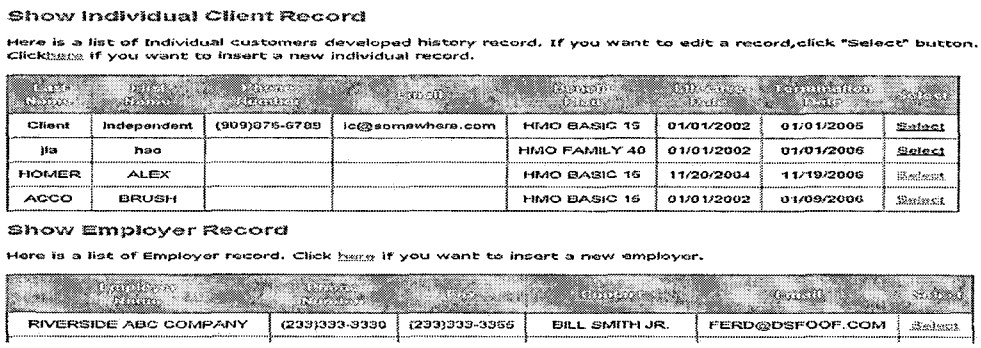

Figure 22. Client Record Page

Broker Insert New Employer Record Page. In this page, it allows a broker to input a new employer after this employer participates Medasolution medicare plan. Figure 23 shows the broker insert new employer page. This page just provides the broker to input employer's information. The information of employees who are working with this employer needs another page to enter.

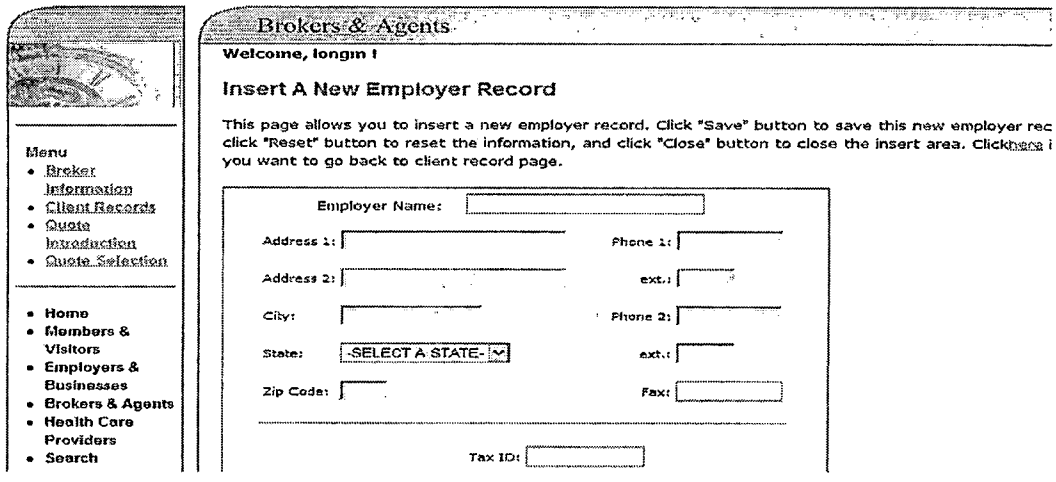

Figure 23. Insert New Employer Page

Broker Insert New Individual Record Page. In this page, it allows the broker enter a new individual information into the database via the page interface. Figure 24 shows the insert new individual client page.

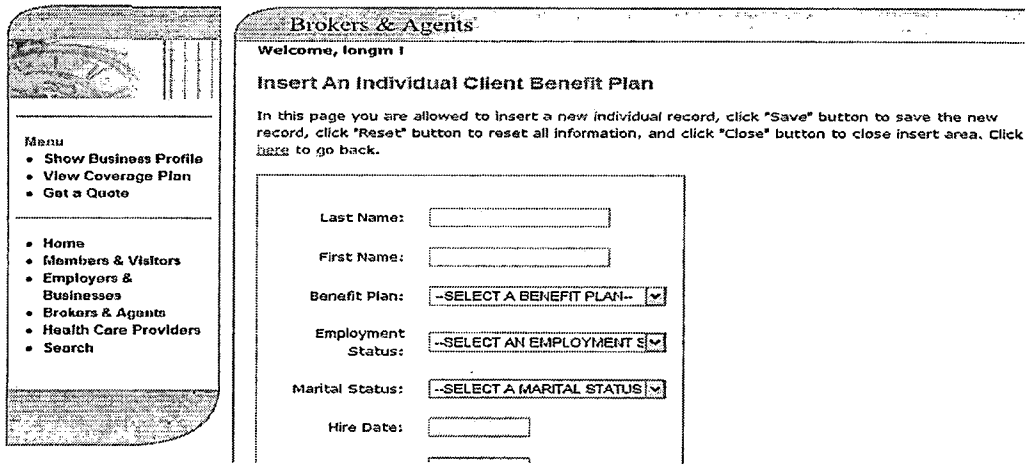

Figure 24. Insert New Individual Client Page

Broker Modify Individual Client Page. In this page a broker can not only modify an individual client developed by this broker, but also delete an individual client record. Figure 25 shows us the broker modify individual client page.

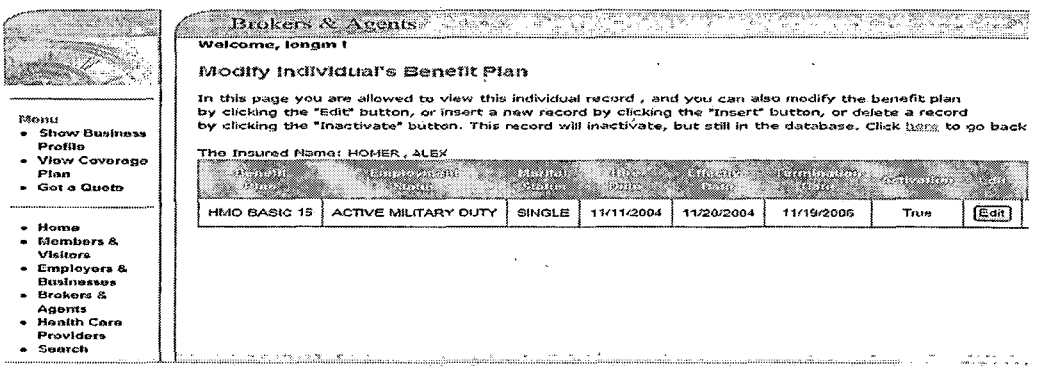

Figure 25. Modify Individual Client Page

Broker Modify Employer Client Page. In this page it not only allows a broker to modify an employer, but also to inactivate an employer. Figure 26 shows the detailed.

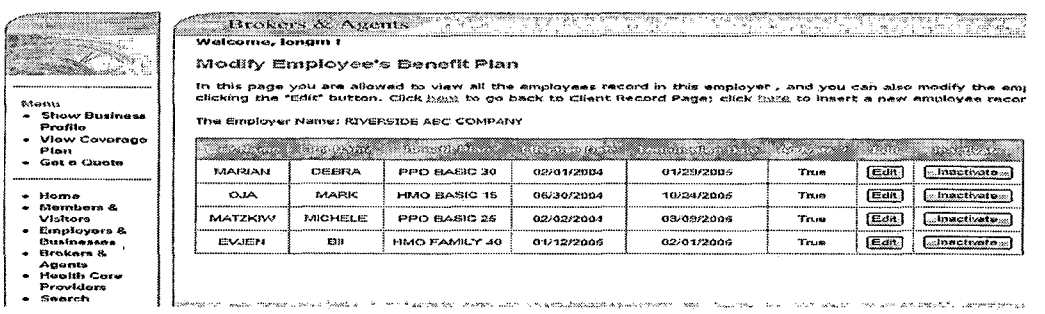

Figure 26. Modify Employer Client Page

Provider Home Page. Only registered medicare providers can reach this page. It allows providers to view other pages.

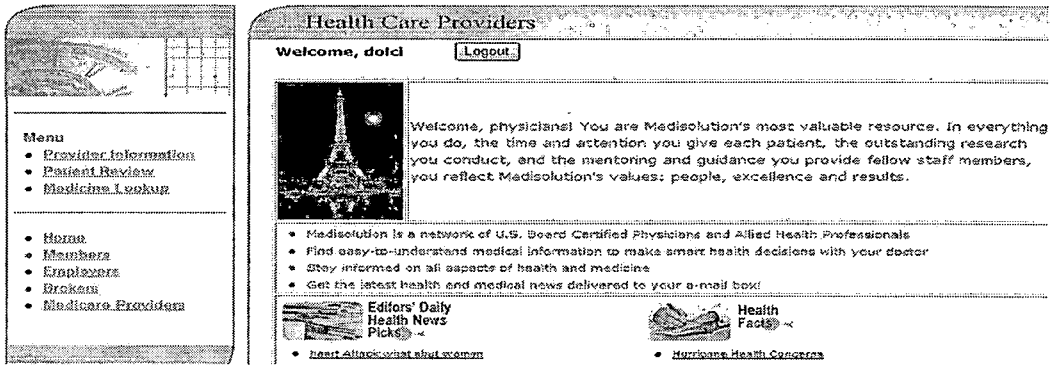

Figure 27. Medicare Provider Home Page

Provider View Patients Page. In this page it allows a medicare provider to modify his/her information stored in the Medasolution database. The provider can choose to cancel edit state by clicking the cancel button on the page. Figure 28 shows the detail.

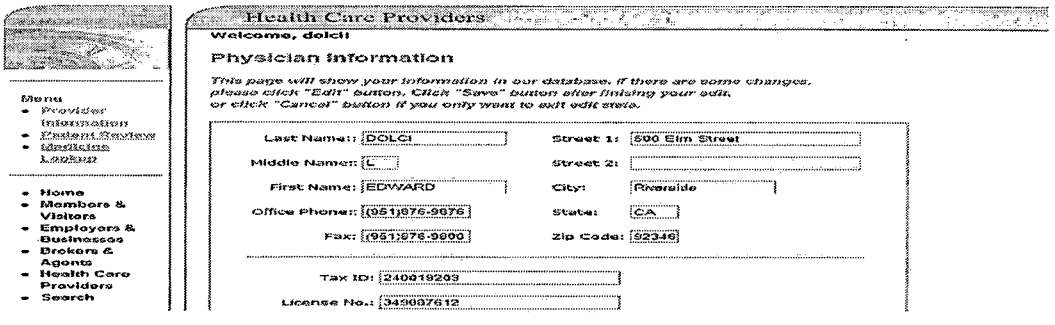

Figure 28. Medicare Provider Information Page

Provider Review Patient Page. In this page, it provides three functions for a medicare a provider. i) enter a patient's last name and search this patient's history record; ii) search all of the patients treated by the provider before; iii) search a patient treatment history. Figure 29 shows us the provider view patient page.

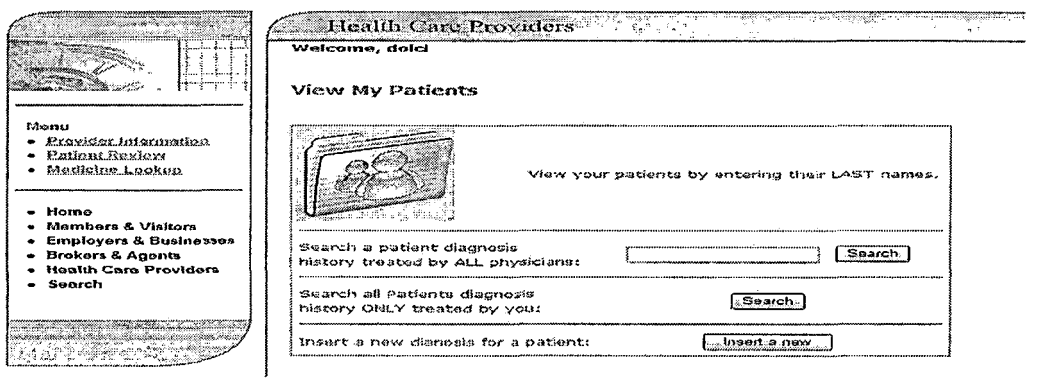

Figure 29. Review Patient Record Page

Provider Lookup a Drgcode Page. In this page, it allows a medicare provider to look up all drgcode in the Medasolution database. Figure 30 shows us the provider look up drgCode page.

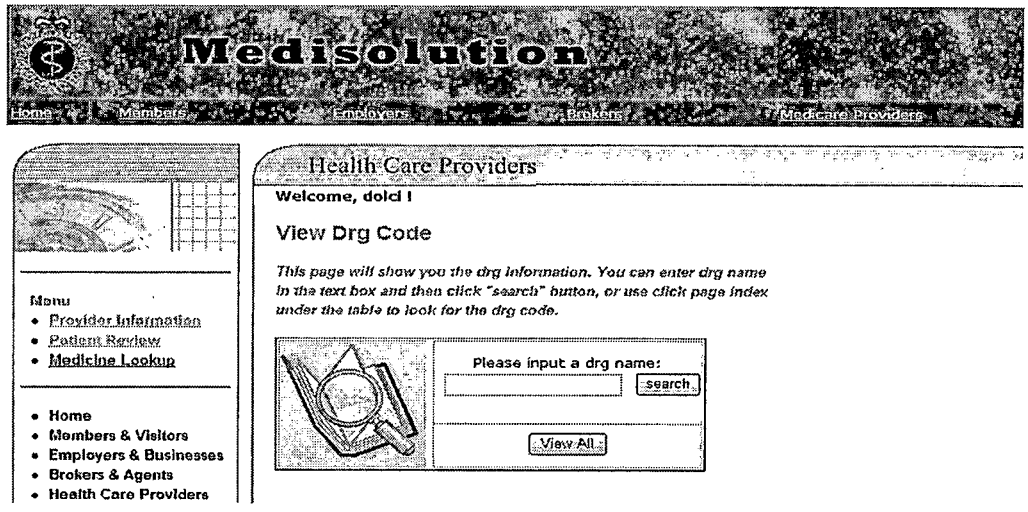

Figure 30. Lookup DrgCode Page

Database Design

We assume that Medasolution system must have at least four registered users represented four categories of users: member, employer, broker, and medicare provider. Also, each user must have a username and password in the user table in order to logon to database system. Besides, there are several other rules needed to be followed in the database:

i) An employer user must have at least one member;

ii) A broker user must have at least one individual user and one employer user;

iii) A medicare provider must have at least one patient user i.e. member user.

Figure 31 is the Medasolution ER diagram showing the

relations and the important attributes of the important entities.

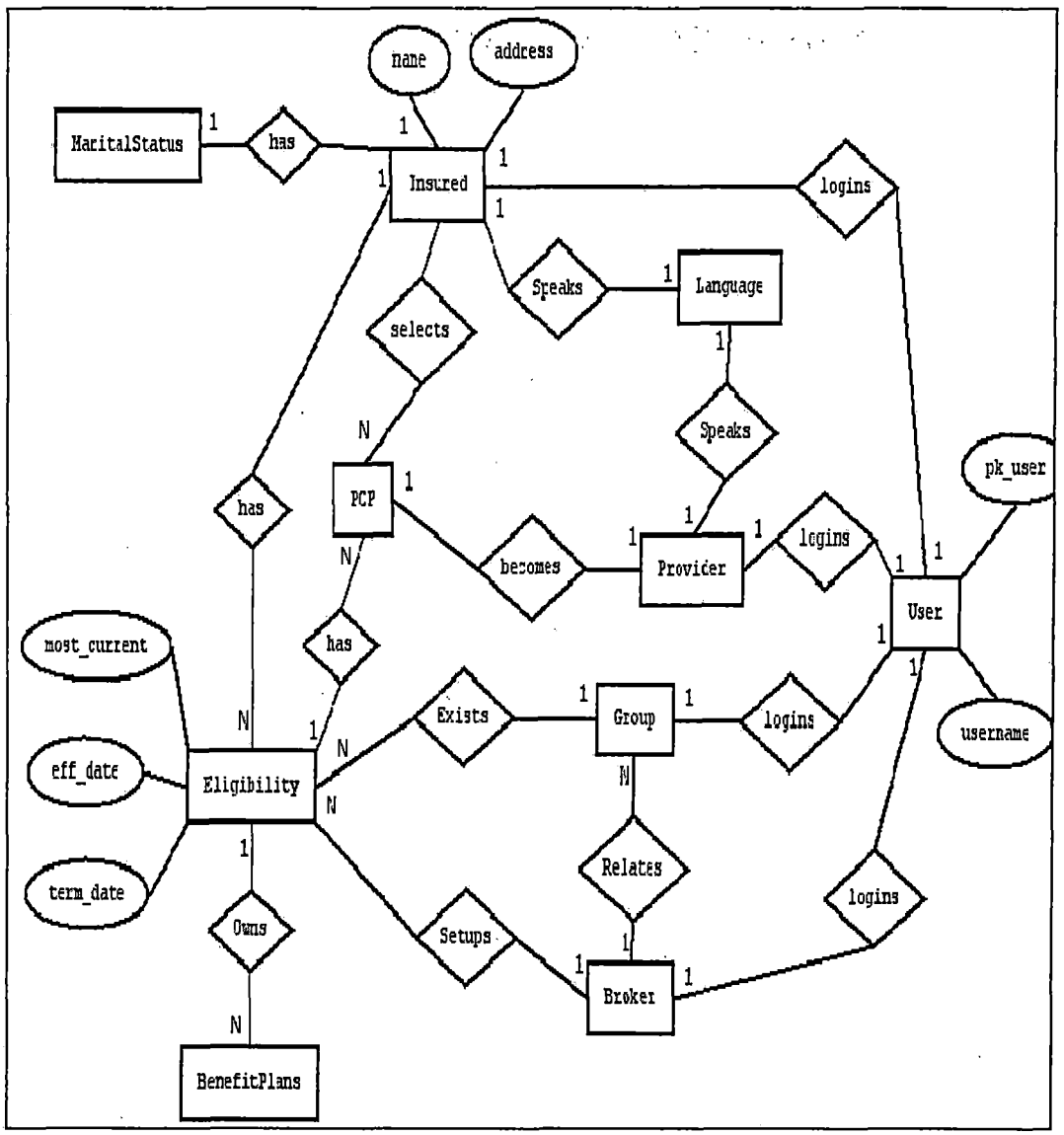

Figure 31. Medasolution Entity Relationship Diagram

CHAPTER FOUR

PROJECT IMPLEMENTATION

Project Class Design

Since the last chapter described the design of this project, the implementation is to be the next step. The Unified Modeling Language (UML) class diagram is a well known and commonly used diagram to analysis and design the relationships and functionalities of each class and its functions. Figure 32 shows us the class diagram. And then I use pseudo codes to illustrate each function. As I mentioned before, the whole project can be separated into four parts based on the user categories. The home page is functioned as logon page, which has all links to the four user home pages. The only function of this page is to provide two text boxes for users to enter username and password. Figure 33 shows the home page function.

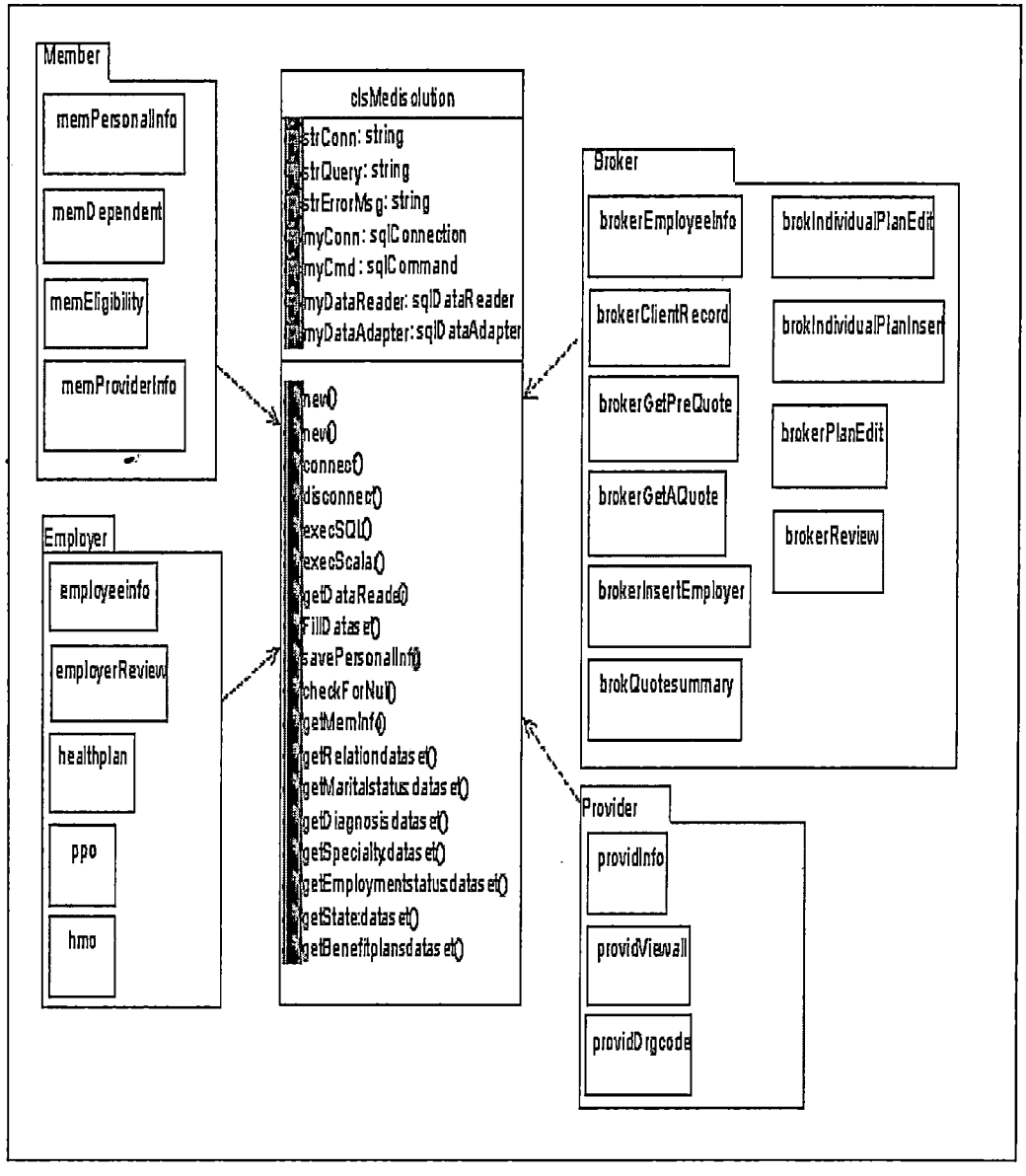

Figure 32. Medasolution Class Diagram

Member Functions Implementation

This section describes the logical algorithms used in this project and explanation for each method of member user. To make the class easier to read, all classes is written in

plain English language.

Member home page class.

Page: memHome.aspx
Attributes: strQuery: string Super Class: clsMedasolution
Methods: Page_load: retrieve session user from home page function. Logout: logout the current user and clear all sessions.

Figure 33. Member Home Page Class

Member personal information page class.

Page: memPersonalInfo.aspx
Attributes: strQuery: string clsdb: clsMedasolution
Methods: LoadLanguage: retrieve the language from database and load language name LoadState: retrive state from database and load state name LoadMaritalStatus: retrieve maritalStatus from database and load maritalstatus status Databind: binding data into the labels

Figure 34. Member Personal Information Page Class

Member dependent information page class

Page: MemDependent.aspx
Attributes: strQuery: string ds: dataset
Methods: LoadRelation: retrieve the relationship from database and load relationship name Databind: Load data into the datagrid getCharbit: change the Boolean value and return the char value saveChangedInfo: if edit state then edit dependents info else insert a new dependent info

Figure 35. Member Dependent Information Page Class

Member provider information page class.

Page: MemProvidInfo.aspx
Attributes: strQuery: string ds: dataset clsdb: clsMedasolution
Methods: Databind: Load data into the datagrid Search_Click: search the provider based on the input parameter. Return the result and output to datagrid Dg_select: retrieve the provider record selected by the member

Figure 36. Member Provider Information Page Class

Member eligibility page class.

Page: MemEligibility.aspx
Attributes: strQuery: string strSessionuser: string ds: dataset clsdb: clsMedasolution
Methods: Databind: Load eligibility data into the datagrid

Figure 37. Member Eligibility Page Class

Employer Functions Implementation

This section describes the logical algorithms used in this project and explanation for each method of employer user. To make the class easier to read, all classes is written in plain English language.

Employer review page class.

Page: EmployReview.aspx
Attributes: strQuery: string clsdb: clsMedasolution
Methods: DataFiller: Load data into the datagrid Save_Click: save data into the group table in the database Cancel_Click: cancel all info in the textbox ControlReadonlystate: interchange the readonly state

Figure 38. Employer Review Page Class

Employee information page class.

Page: MemmployeeInfo.aspx
Attributes: strQuery: string strErrormsg: string
Methods: DataFiller: Load employer data into the datagrid Save_Click: save the edited data into database Cancel_Click:cancel the data input

Figure 39. Employee Information Page Class

Preferred Provider Organization plan page class.

Page: PPO.aspx
Methods: PPO5_click: load ppo5 benefit plan PPO25_click: load ppo25 benefit plan PPO30_click: load ppo30 benefit plan PPO40_click: load ppo40 benefit plan PPO50_click: load ppo50 benefit plan

Figure 40. Preferred Provider Organization Plan Page Class

Health Maintenance Organization plan page class.

Page: hmo.aspx
Methods: HMO15_click: load ppo5 benefit plan HMO40_click: load ppo25 benefit plan

Figure 41. Health Maintenance Organization Plan Page Class

Broker Functions Implementation

This section describes the logical algorithms used in this project and explanation for each method of broker user. To make the class easier to read, all classes is written in plain English language.

Broker information page class.

Page: brokInfo.aspx
Attributes: strQuery: string clsdb: clsMedasolution
Methods: DataFiller: load broker data into datagrid Save_Click: save the edited data into database Cancel_Click: cancel the data input in the textbox Loadstate: load the state name into the dropdownlist

Figure 42. Broker Information Page Class

Broker client record page class.

Page: brokerClientRecord.aspx
Attributes: clsdb:clsMedasolution strSessionuser:string
Methods: loadIndividualRecord: load individual client record from database loadEmployerRecord: load employer client record from database

Figure 43. Broker Client Record Page Class

Broker get a quote page class.

Page:getAQuote.aspx
Attributes: clsdb:clsMedasolution
Methods: loadDateMonthYear: load the month,day, and year checkLeapyear: check the year whether it is leap year or not getAQuote: get a quote of benefit plan. BindDay: bind the day based on the month and year.

Figure 44. Broker Get A Quote Page Class

Broker individual plan insert page class.

Page: brokIndividualPlanInsert
Attributes: Clsdb:clsMedasolution
Methods: loadBenefitplans: load the benefitplan from database loadMaritalStatus: load maritalStatus from database loadEmploymentStatus: load employmentstatus from database save_click: save the input data into the database reset_click: reset the data in the textbox close_click: close the insert table panel clearTextbox: clear the input in the text box area

Figure 45. Broker Individual Plan Insert Page Class

Broker individual plan edit page class.

Page: brokIndividualPlanEdit
Attributes: strSessionuser: string clsdb:clsMedasolution
Methods: loadBenefitplans: load the benefitplan from database loadMaritalStatus: load maritalStatus from database loadEmploymentStatus: load employmentstatus from database Editthisrow: if editstate then edit else delete this row save_click: save the input data into the database reset_click: reset the data in the textbox close_click: close the insert table panel clearTextbox: clear the input in the text area

Figure 46. Broker Individual Plan Edit Page Class

Broker employer plan insert page class.

Page: brokEmployPlanInsert
Attributes: clsdb:clsMedasolution
Methods: loadBenefitplans: load the benefitplan from database loadMaritalStatus: load maritalStatus from database loadEmploymentStatus: load employmentstatus from database save_click: save the input data into the insured and eligibility table reset_click: reset the data in the textbox close_click: close the insert table panel

Figure 47. Broker Employer Plan Insert Page Class

provider user. To make the class easier to read, all classes is written in plain English language.

Provider information page class.

Page: providerInfo.aspx
Atrributes: clsdb:clsMedasolution
Methods: Datafiller: load data into the page Save_click: save the edit information into the database

Figure 50. Provider Information Page Class

Provider view patient page class.

Page: providViewall.aspx
Atrributes: clsdb:clsMedasolution
Methods: Databind: load data into the datagrid of the page Loaddiagnosiscode: load diagnosiscode into the page searchByPatient_click: search a patient by patient lastname searchByProvider_click: search patient list who is treated by the provider save_click: save all edited textbox information

Figure 51. Provider View Patient Page Class

Provider lookup drgCode page class.

Page:brokerdrgcode.aspx
Attributes: clsdb:clsMedasolution
Methods: Viewall: load all drgCode list Search_click: search a specific drgCode

Figure 52. Provider DrgCode Page Class

Home page class.

Page:home.aspx
Attributes: clsdb:clsMedasolution strQuery: string strError: string
Methods: Submit_click: submit the username and password into database to test authentication of user

Figure 53. Home Page Class

User Characteristics

Users in Medasolution Customer System include member, employer, broker, provider and visitors. All the users are assumed to know how to user web browser and speak English. They should also be able to follow manual written in plain English text.

Software System Attributes

Reliability

All contents and logs shall be generated automatically so no human effort is needed. The server shall be up twenty-four hours a day and seven days a week, with exception that periodical system maintenance needs to be conducted depending on the reliability of the web server. The system should be handle network packet loss smoothly. The system should not save inconsistent data or incomplete data into the database.

Security

The web server used to execute the Medasolution package will eventually be SSL-enabled for the authentication of users and submission of confidential information such as a personal profile.

Maintainability

The Medasolution consists of ASPX pages, VB class code, web configuration file and style sheets. They are put under different directories with a hierarchy. Other files including ASP source code and documents will be put in separate directories as well. ASPX files are organized using

packages. This structure will aid in maintaining all modules, and therefore maximizes ease of maintenance.

APPENDIX A

VISUAL BASIC.NET FILE

clsMedasolution.vb

```
Imports Microsoft.VisualBasic
Imports System
Imports System.Data
Imports System.Data.SqlClient
Imports System.Reflection
Imports system.windows.forms

Namespace nsMedasolution
Public Class clsMedasolution
    'Declare a connection string to database
    Private _strConn As String ="Persist Security Info=false;Data Source=(local);"
        _ "Initial Catalog=medasolution;User ID=phillip;Password="
    'Declare a dataset, connection and adapter
    private ds As DataSet
    'Declare database variables
   private myConn as SqlConnection
   private myCmd as   SqlCommand
   private myDataAdapter As SqlDataAdapter
   private myDataReader as sqlDataReader
   private strQuery As String
   Public Sub New()
        MyBase.new()
   End Sub
   'Declare a constructure with an argurement
   Public Sub New(ByVal strConn As String)
        MyBase.new()
        _strConn = strConn
   End Sub
   'Declare a property
   Public Property connectionString As String
        Get
            Return _strConn
        End Get
         Set(ByVal Value As String)
             _strConn = Value
         End Set
   End Property
   Public sub showMsgbox(strMsg as string)
        messagebox.show(strMsg, "Alert test" , MessageBoxButtons.OK,
```

```
        MessageBoxIcon.Exclamation, MessageBoxDefaultButton.Button1,
        MessageBoxOptions.ServiceNotification)
end sub

public Function doLogon(byval strQuery as string, byRef intClassID as integer,
    byRef strUsername as string) as boolean
    myConn = new sqlConnection(_strConn)
    myCmd = new sqlCommand(strQuery, myConn)
    Dim blnContinue as boolean
    try
        Connect()
        myDataReader = myCmd.ExecuteReader
        if myDataReader.read
            blnContinue = true
            doLogon = true
        else
            blnContinue = false
            doLogon = false
        end if
        if blnContinue then
            intClassID = myDataReader("classID")
            strUsername = myDataReader("username")
        end if
    catch ex as sqlexception
        exit function
    finally
        myDataReader.close
        Disconnect()
    end try
end Function

Public Function getReader(byval strQuery as string, byref dr as SqlDatareader,
        byref strErrorMsg as string) as boolean
    Dim dbReader as SqlDataReader
    myConn = new sqlConnection(_strConn)
    myCmd = new sqlCommand(strQuery, myConn)
    myCmd.commandType = Commandtype.text
    myCmd.CommandText = strQuery
    Try
        Connect()
        dbReader = myCmd.executeReader()
        dr = dbReader
```

```
    catch ex as exception
        getReader = false
        strErrorMsg = "Error in reader function<br>" & ex.tostring()
        exit function
    end try
    'disconnect()
    getReader =true
end Function

public Function FillDataset(byval strQuery as string, byref ds as dataset, byval
        tablename as string) as boolean
    if (ds is nothing ) then
        FillDataset = nothing
        exit function
    end if
    myConn = new sqlConnection(_strConn)
    try
        Connect()
        myDataAdapter = new SqlDataAdapter(strQuery, myConn)
        myDataAdapter.fill(ds, tablename)
    catch ex as exception
        FillDataset= false
        exit Function
    end try
    FillDataset = true
    Disconnect()
end function

Public Function ExecScalar(byval strSQL as string, byRef strmsg as string) as
        integer
    Dim intReturnResult as integer
    try
        Connect()
        myCmd = new SqlCommand(strSQL, myConn)
        intReturnResult = myCmd.ExecuteScalar()
    catch e as Exception
        ExecScalar = -1
        strmsg = e.stackTrace.ToString()
        exit Function
    end try
    ExecScalar = intReturnResult
    Disconnect()
```

```
end Function

Public function checkForNull( byval value as object) as string
    if Not IsDBNull(value) then
        checkForNull = value.tostring()
    else
        checkForNull = ""
    end if
end function

Public Function getMemberInfo(byval username as string) As DataRow
        Dim intGetGroup as integer = getGroupID(username)
        strQuery = "select i.last_name,i.first_name, i.middle_name, " & _
            "i.street1, "i.street2, i.city, " i.state, i.zipcode, i.phone, " & _
            "i.phone2, i.phone_ext, i.phone2_ext, i.email, " & _
            "i.dob, i.ssn, i.gender, lg.language, lo.location_name, " & _
            "g.group_name, (select p.last_name + ',' + p.first_name " & _
            "as Provider_Name from [user] u, insured i, provider p, " & _
            "pcp pcp where u.username= '" & username  & """  & _
            "and i.fk_usertable = u.pk_user and i.pk_insured = " & _
            "pcp.fk_insured and pcp.fk_provider = p.pk_provider " & _
            "and pcp.most_current =1) as Provider_name, " & _
            " ms.maritalstatus from Insured i, [user] u, eligibility e," & _
            "[group] g, location lo, language lg, maritalstatus ms " & _
            " where u.username= '" & username & """ & _
            " and u.pk_user = i.fk_usertable and i.pk_insured = " & _
            "e.fk_insured and g.pk_group = '" & intGetGroup & "' and " & _
            "lo.pk_location = g.fk_location " & _
            "and e.fk_group = g.pk_group and lg.pk_language = " & _
            "i.fk_language and i.fk_maritalstatus = ms.pk_maritalstatus " & _
            " and e.pk_eligibility in (select e.pk_eligibility from " & _
            "eligibility e where e.most_current=1)"
        try
            Connect()
            myDataAdapter = New SqlDataAdapter(strQuery, myConn)
            ds = New DataSet()
            myDataAdapter.Fill(ds, "Insured")
        catch ex as exception
            showMsgbox("error in this function in class")
        end try
        Return ds.Tables(0).rows(0)
        Disconnect()
```

```
    End Function

Public Function getGroupID(byval username as string) as integer
    'get medicalgroup name based on username
    'Dim strQuery as string
    strQuery = "select g.pk_group " & _
        " from [group] g, insured i, eligibility e, [user] u " & _
        " where u.username= '" & username & "'" & _
        " and i.fk_usertable = u.pk_user and e.fk_group = g.pk_group " & _
        " and e.pk_eligibility in " & _
            "(select pk_eligibility from [user] u, insured i, eligibility e " & _
            " where u.username = '" & username & "'" & _
            " and i.fk_usertable = u.pk_user and fk_insured = i.pk_insured " &
            " and e.most_current =1 ) "
    try
        'myConn = new SqlConnection(_strConn)
        Connect()
        myCmd = new SqlCommand(strQuery, myConn)
        Dim strResult as string
        'myConn.open()
        myDataReader = myCmd.executeReader()
        while myDataReader.read()
            strResult = myDataReader.item("pk_group")
        end while
        Dim intResult as integer
        if strResult <> "" then
            if  IsNumeric(strResult)  then
                getGroupID = cint(strResult)
            end if
        else
            getGroupID = -1
        end if
    catch ex as exception
        exit function
    finally
        myDatareader.close
        'myConn.close()
        Disconnect()
    end try
end Function

public Function getRelation() as DataTable
```

```
    strQuery = "select pk_relation, relationship from relation where " & _
            "relationship <> 'INSURED'"
    Dim ds as new Dataset
    try
    Connect()
    myDataAdapter = new SqlDataAdapter(strQuery, myConn)

    myDataAdapter.fill(ds, "relation")
    catch ex as sqlexception
        exit function
    end try
    return ds.tables("relation")
    Disconnect()
end Function

public Function getDiagnosiscode() as dataset
    strQuery ="select pk_diagnosiscode, diagnosiscode_name from " & _
        diagnosiscode order by diagnosiscode_name"
    Dim ds as new Dataset
    try
        connect()
        myDataAdapter = new SqlDataAdapter(strQuery, myConn)
        myDataAdapter.Fill(ds, "Diagnosiscode")
    catch ex as sqlexception
        exit function
    end try
    return ds
    Disconnect()
end function

public Function getLanguage() as Dataset
    strQuery = "select pk_language, language from language order by " & _
        pk_language asc"
    myConn = new SqlConnection(_strConn)
    myDataAdapter = new SqlDataAdapter(strQuery, myConn)
    Dim ds as new Dataset()
    myDataAdapter.Fill(ds, "language")
    return ds
end Function

public Function getMaritalStatus() as DataSet
    strQuery = "select * from maritalstatus order by pk_maritalstatus asc"
```

```
        myConn = new SqlConnection(_strConn)
        myDataAdapter = new SqlDataAdapter(strQuery, myConn)
        Dim ds as new DataSet()
        myDataAdapter.Fill(ds, "MaritalStatus")
        return ds
    end Function
    public Function getState() as DataSet
        strQuery = "select * from state order by abbreviation asc"
        myConn    = new SqlConnection(_strConn)
        myDataAdapter = new SqlDataAdapter(strQuery, myConn)
        Dim ds as new Dataset()
        myDataAdapter.Fill(ds, "State")
        return ds
    end Function

    public Function getSpecialty() as DataSet
        strQuery = "select pk_specialty, specialty_name from specialty order by " _
                "specialty_name asc"
        myConn = new SqlConnection(_strConn)
        myDataAdapter = new sqlDataAdapter(strQuery, myConn)
        Dim ds as new Dataset
        myDataAdapter.fill(ds,"specialty")
        return ds
    end Function

    public Function getEmploymentStatus() as dataset
        strQuery = "select pk_employmentstatus, status_name from " & _
            "employmentstatus order by status_name"
        myConn = new sqlConnection(_strConn)
        myDataAdapter = new sqlDataAdapter(strQuery, myConn)
        Dim ds as new DataSet
        myDataAdapter.Fill(ds, "employmentStatus")
        return ds
    end Function

    public Function getBenefitPlans() as DataSet
        strQuery = "select pk_benefitplans, benefitplan_name from benefitplans " _
            "order by benefitplan_name"
        myConn = new sqlConnection(_strConn)
        myDataAdapter = new sqlDataAdapter(strQuery, myConn)
        Dim ds as new Dataset
        myDataAdapter.Fill(ds, "benefitplans")
```

```
    return ds
end Function

public Function savePersonalInfoChanged(firstname, lastname,middlename, _
    street, street2, city, state1,zipcode, homephone, homephoneext, _
    cellphone, cellphoneext, email, dob, ssn, language,  sex, ms, username) _
    as integer

    Dim intResult as integer
    strQuery = "update Insured set first_name=" & firstname & "_ ,last_name="
        & lastname & "," middle_name=" & middlename & ",street1=" & _
        "street & ",street2=" & street2 & ",city=" & city & ",state=" & state1 &
        ",zipcode=" & zipcode & ",phone=" & homephone & ",phone_ext=" &
        "homephoneext & ",phone2=" & cellphone & "," & _
        "phone2_ext=" & cellphoneext & ",email=" & email & "," & _
        "dob=" & convert.todatetime(dob) & ",ssn=" & ssn & "," & _
        "fk_language=" & cint(language) & ",gender=" & sex & _
        ",fk_maritalstatus=" & cint(ms) & "," & _
        "date_modified =" & datetime.now & _
        " from insured i, [user] u " & _
        " where i.fk_usertable = u.pk_user and u.username='" & username & "'"
    dim str as string
    str ="update Insured set phone_ext=" & homephoneext & _
          " from insured i, [user] u " & _
        " where i.fk_usertable = u.pk_user and u.username='" & username & "'"
    try
        connect()
        myCmd = new SqlCommand(str, myConn)
        intResult = myCmd.executeNonQuery()
    catch ex as exception
        savePersonalInfoChanged = -1
        exit function
    end try
    Disconnect()
    savePersonalInfoChanged = intResult
end Function

public Function ExecSQL(byval strQuery as string, byVal blnsp as " & _
    boolean,byRef strErrorMsg as string) as integer
    Dim intReturnValue as integer
    myConn = new sqlConnection(_strConn)
    myCmd = new sqlCommand(strQuery, myConn)
```

```
        if blnsp then
            myCmd.commandType = CommandType.storedProcedure
        else
            myCmd.CommandType = CommandType.text
        end if
        myCmd.CommandText = strQuery
        try
            myConn.open()
            intReturnValue = myCmd.executeNonQuery()
        catch ex as sqlException
            ExecSQL = -1
            strErrorMsg = ex.stackTrace.tostring()
            exit function
        Finally
            myConn.close
            ExecSQL = intReturnValue
        end try
    end function

    public sub Connect()
        if myConn is Nothing then
            myConn = new SqlConnection(_strConn)
        end if
        if myConn.state = ConnectionState.closed then
            myConn.open()
        end if
    end sub

    public sub Disconnect()
        if Not (myConn.state = connectionstate.closed) then
            myConn.close()
        end if
        myConn.close()
    end sub
    End Class
End Namespace
```

APPENDIX B

ASP.NET FILES

Home.aspx

```
<script language="vb" runat="server">
public myConn as new sqlConnection()
public clsdb as new clsMedasolution
public strError as string
sub submit_click(a as object, e as eventargs)
    try
        Dim strQuery as string ="select classID, username from [user] where
            username='" & txtuser.text & "' and password='" & txtpass.text & "'"
        dim intClassID as integer
        Dim strSessionuser as string
        Dim blnUserAuthenticate as boolean = clsdb.doLogon _
                (strQuery, intClassID, strSessionuser)

        if blnUserAuthenticate = true then
            FormsAuthentication.redirectFromLoginPage(txtuser.text, false)
            session("classID")= intClassID
            session("username") = strSessionuser
            select case session("classID")
                case 1:
                    response.redirect("memHome.aspx?username=" & txtuser.text)
                case 2:
                    response.redirect("employHome.aspx")
                case 3:
                    response.redirect("brokHome.aspx")
                case 4:
                    response.redirect("providHome.aspx")
                case 99:
                    response.redirect("webmaster.aspx")
            end select
        else
            output.visible=true
            output.text="**Invalid Login"
        end if
        catch ex as sqlexception
            strError ="Error in data connection to database" & ex.tostring()
            response.Write(strError)
        finally
    end try
end sub
</script>
```

MemPersonalInfo.aspx

```
<script language="VB" runat="server" >
'Initialize class clsMedasolution
Dim clsdb as new clsMedasolution
Dim strSessionUser as string
Dim i as integer
'Declare query string
Dim strQuery as string
Dim strError as string
Dim myCmd as SqlCommand
Dim myDataAdapter as SqlDataAdapter
Dim ds as DataSet
sub Page_load(a as object, e as eventargs)
    strSessionUser = system.web.HttpContext.Current.session("username").tostring
    lblTop.text = "Welcome, " & strsessionUser & " !"
    if not Page.IsPostback then
        DataBinding()
    end if
end sub

sub cellphoneTextbox_changed(a as object, e as eventargs)
    'set cellphoneext textbox to be fillable only when cellphone textbox is ok
    p2txtcellphoneext.readonly = "false"
end sub

sub Load_Language( byVal strLanguagename as string)
    ds = clsdb.getLanguage()
    ddllanguage.datasource = ds
    ddllanguage.datasource = ds.tables("language")
    ddllanguage.datatextfield    = "language"
    ddllanguage.dataValueField = "pk_language"
    ddllanguage.databind()

    for i = 0 to ddllanguage.items.count - 1
    if ddllanguage.Items(i).Tostring() = strlanguageName then
        ddllanguage.selectedIndex = i
    end if
    next
End sub

sub Load_MaritalStatus(byVal strMS as string)
```

```
    ds = clsdb.getMaritalStatus()
    ddlms.datasource = ds
    ddlms.datasource = ds.tables("MaritalStatus")
    ddlms.dataTextField = "maritalstatus"
    ddlms.dataValueField = "pk_maritalstatus"
    ddlms.DataBind()

    For i = 0 To ddlms.items.count - 1
    if ddlms.Items(i).tostring() = strMS then
        ddlms.selectedIndex = i
    end if
    next
end sub

sub Load_sex(byval strsex as string)
    for i = 0 to ddlsex.items.count -1
        if ddlsex.items(i).tostring = strsex then
            ddlsex.selectedIndex = i
        end if
    next
end sub

sub Load_state(byVal strStateName as string)
    ds = clsdb.getState()
    ddlState.datasource = ds
    ddlState.DataSource = ds.Tables("State")
    ddlState.DataTextField = "name"
    ddlState.DataValueField = "abbreviation"
    ddlState.databind()
    For i = 0 to ddlState.items.count - 1
    if ddlstate.Items(i).value = strStateName then
        ddlstate.selectedIndex = i
        'ddlstate.items(i).value
    end if
    next
end sub

sub DataBinding()
try
    'Dim dr as DataRow
    'dr = clsdb.getMemberinfo(strSessionUser)
```

```
Dim intGroupID as integer = clsdb.getGroupID(strSessionUser)

strQuery = "select i.last_name,i.first_name, i.middle_name, i.street1, " & _
        "i.street2, i.city, i.state, i.zipcode, i.phone, i.phone2, " & _
        "i.phone_ext, i.phone2_ext, i.email, i.dob, i.ssn, i.gender," & _
        " lg.language, lo.location_name, g.group_name," & _
        " (select p.last_name + ',' + p.first_name as Provider_Name " & _
        " from [user] u, insured i, provider p, pcp pcp " & _
        " where u.username= '" & strSessionUser    & "'"    & _
        " and i.fk_usertable = u.pk_user and i.pk_insured = pcp.fk_insured _
        " and pcp.fk_provider = p.pk_provider and pcp.most_current =1) as "_
        "Provider_name, ms.maritalstatus from Insured i, [user] u, " & _
        "eligibility e, [group] g, location lo, language lg, " & _
        " maritalstatus ms where u.username= '" & strSessionuser & "'" & _
        " and u.pk_user = i.fk_usertable and i.pk_insured = e.fk_insured " & _
        " and g.pk_group = '" & intGroupID & "' and lo.pk_location = " & _
        "g.fk_location and e.fk_group = g.pk_group and " & _
        "lg.pk_language = i.fk_language " & _
        " and i.fk_maritalstatus = ms.pk_maritalstatus " & _
        " and e.pk_eligibility in " & _
            " (select e.pk_eligibility from eligibility e " & _
            " where e.most_current=1)"
Dim blnResult as boolean
ds = new Dataset()
blnResult = clsdb.FillDataset(strQuery, ds, "Insured")

if blnResult = true then
    'lblTop.text &= "<br>FillDataset works well"
else
    lblTop.text &= "<br>FillDataset does not work"
end if
Dim tblInsured as DataTable = ds.Tables("Insured")
if intGroupID <> -1 then
'lblTop.text &= "<br>GroupID:" & intGroupID.tostring()
else
lblTop.text &= "<br>Not groupID"
end if
'Data bind the first table block
txtfirstname.text = clsdb.checkfornull(tblInsured.rows(0)("first_name"))
txtlastname.text = clsdb.checkfornull(tblInsured.rows(0)("last_name"))
txtmiddlename.text    = clsdb.checkfornull(tblInsured.rows(0)("middle_name"))
txtstreet1.text = clsdb.checkfornull(tblInsured.rows(0)("street1"))
```

```
        txtstreet2.text = clsdb.checkForNull(tblInsured.rows(0)("street2"))
        txtcity.text = clsdb.checkForNull(tblInsured.rows(0)("city"))
        txtstate.text = clsdb.checkForNull(tblInsured.rows(0)("state"))
        txtzipcode.text = clsdb.checkForNull(tblInsured.rows(0)("zipcode"))
        txthomephone.text = clsdb.checkForNull(tblInsured.rows(0)("phone"))
        txthomephoneext.text = clsdb.checkfornull(tblInsured.rows(0)("phone_ext"))
        txtcellphone.text = clsdb.checkfornull(tblInsured.rows(0)("phone2"))
        txtcellphoneext.text = clsdb.checkfornull(tblInsured.rows(0)("phone2_ext"))
        txtemail.text = clsdb.checkForNull(tblInsured.rows(0)("email"))
        'Data bind the second table block
        txtdob.text = clsdb.checkfornull(tblInsured.rows(0)("dob"))
        txtssn.text = clsdb.checkfornull(tblInsured.rows(0)("ssn"))
        txtsex.text = clsdb.checkfornull(tblInsured.rows(0)("gender"))
        txtlocation.text = clsdb.checkfornull(tblInsured.rows(0)("location_name"))
        txtlanguage.text = clsdb.checkfornull(tblInsured.rows(0)("language"))
        txtmedicalgroup.text = clsdb.checkfornull(tblInsured.rows(0)("group_name"))
        txtms.text = clsdb.checkfornull(tblInsured.rows(0)("maritalstatus"))
        txtpcp.text = clsdb.checkfornull(tblInsured.rows(0)("provider_name"))
        catch ex as exception
            response.write(ex.tostring())
            exit sub
        end try
    end sub

    sub EditPersonalInfo_Click(sender as object, e as EventArgs)
        'convert the panel to edit condition
        panelPersonal.visible=false
        panelEditPersonal.visible=true
        'Dim dr as DataRow
        try
        'dr = clsdb.getMemberinfo(strSessionUser)
        'Data bind the first table block
        p2txtfirstname.text = txtfirstname.text
        p2txtlastname.text = txtlastname.text
        p2txtmiddlename.text    = txtmiddlename.text
        p2txtstreet1.text = txtstreet1.text
        p2txtstreet2.text = txtstreet2.text
        p2txtcity.text = txtcity.text
        Dim strStateName as string = txtstate.text
        load_state(strStateName)
        p2txtzipcode.text = txtzipcode.text
        p2txthomephone.text = txthomephone.text
```

```
        p2txthomephoneext.text = txthomephoneext.text
        p2txtcellphone.text = txtcellphone.text
        p2txtcellphoneext.text = txtcellphoneext.text
        p2txtemail.text = txtemail.text

        'Data bind the second table block
        p2txtdob.text = txtdob.text
        p2txtssn.text = txtssn.text
        'p2txtlanguage.text = cstr(dr("language"))
        Dim strlanguage as string = txtlanguage.text
        Load_Language(strlanguage)
        Dim strsex as string = txtsex.text
        Load_sex(strsex)
        p2txtlocation.text = txtlocation.text
        p2txtmedicalgroup.text = txtmedicalgroup.text
        Dim strMS as string = txtms.text
        Load_MaritalStatus(strMS)
        p2txtpcp.text = txtpcp.text
        catch ex as exception
            response.write(ex.tostring())
            exit sub
        end try
    end sub

    sub ShowDependent_Click(sender as object, e as EventArgs)
        'panelPersonal.visible=false
        'panelDependent.visible=true
    end sub

    sub savePersonalInfo_Click(sender as object, e as EventArgs)
        If page.isValid()
            'call savePersonalInfoChanged function
            Dim intResult as integer
            dim tblInsured as string 'datatable
            strQuery = " Update Insured set first_name = '" &    p2txtfirstname.text & _
                "',last_name = '" & p2txtlastname.text & "', middle_name = _
                '" & p2txtmiddlename.text & "street1 = '" & p2txtstreet1.text & _
                "', street2 = '" & p2txtstreet2.text & "',city = '" & _
                p2txtcity.text & "', state = '" & ddlstate.selectedItem.value & "'," & _
                "zipcode = '" & p2txtzipcode.text & "', phone = '" & _
                p2txthomephone.text & "',phone_ext = '" & _
                p2txthomephoneext.text & "', phone2 = '" & p2txtcellphone.text & "'
```

```
            "phone2_ext = '" & p2txtcellphoneext.text & "', email = '" & _
            p2txtemail.text & "',dob = '" & convert.ToDateTime(p2txtdob.text) &_
            "', ssn = '" & p2txtssn.text & "',fk_language = '" & _
            cint(ddllanguage.selectedItem.value) & "', gender = '" & _
            "ddlsex.selectedItem.value & "',fk_maritalstatus = '" & _
            cint(ddlms.selectedItem.value) & "', date_modified = '" & _
            "DateTime.Now & from insured i, [user] u " & _
            " where i.fk_usertable = u.pk_user and u.username='" & _
            strSessionuser & "'
        intResult = clsdb.execSQL(strQuery, false,strError)
        if intResult = -1 then
            lblTop.text &= "<br>No record had been updated" & strError
        else
            lblTop.text &= "<br>" & intResult.tostring & " records had been
modified."
        end if
    end if
    DataBinding()
    panelEditPersonal.visible=false
    panelPersonal.visible=true
end sub

sub cancelPersonalInfo_Click(sender as object, e as EventArgs)
    panelEditPersonal.visible=false
    panelPersonal.visible=true
end sub
</script>
```

MemEligibility.aspx

```
<script language="vb" runat="server">
Public strSessionUser as String     'username from the previous page
Public strQuery as String           'string value to hold all query to database
Public strBenefitplancode as string 'global value for benefitplan.ascx property
Public clsdb as new clsMedasolution
Public myConn as new SqlConnection(clsdb.connectionString)
Public myCmd as sqlCommand
Public myDataReader as sqlDataReader
Public myParameter as sqlParameter
sub Page_load(a as object, e as EventArgs)
    strSessionUser = httpContext.current.session("username").Tostring()
    lblTop.text = "Welcome, " & strSessionUser & "!"
```

```
    if Not Page.IsPostBack then
        DataBind()
    end if
end sub

sub DataBind()
    myCmd = new SqlCommand("sp_selectInsuredGroupProvider", myConn)
    myCmd.CommandType = CommandType.storedProcedure
    myParameter = new sqlParameter("@username", sqldbType.nvarchar, 50)
    myCmd.parameters.add(myParameter)
    myCmd.Parameters("@username").value = strSessionuser

    try
        myConn.open()
        myDataReader    = myCmd.executeReader()
        if myDataReader.read() then
            txtlastname.text = myDataReader("last_name")
            txtfirstname.text = myDataReader("first_name")
            txtmiddlename.text = myDataReader("middle_name")
            txtemploymentstatus.text = myDataReader("status_name")
            txtgroup.text = myDataReader("group_name")
            txtpcp.text = myDataReader("Provider_name")
            txtbenefitplan.text = myDataReader("benefitplan_name")
            txteffdate.text = myDataReader("eff_date")
            txttermdate.text = myDataReader("term_date")
            strBenefitplancode = myDataReader("benefitplan_code")
            viewstate("benefitplancode") = myDataReader("benefitplan_code")
        end if
    catch ex as sqlException
        lblfirst.text &= "<br> data reader is failed."
    Finally
        myDataReader.close()
        myConn.close()
        'deliver value to ascx property strValue
        benefitplanascx.strValue = strbenefitplancode
    end try
end sub
</script>
```

MemDependentInfo.aspx

```
<script language="vb" runat="server" >
```

```
public strSessionuser as string
public strQuery as string
public clsdb as new clsMedasolution
public myConn as new SqlConnection(clsdb.connectionString)
public myCmd as sqlCommand
public myDataAdapter as sqlDataAdapter
public myParameter as SqlParameter
public myDataReader as SqlDataReader
public ds as Dataset
public strDependentPK as string   'dependent table PK
public strProcess as string       'determine viewstate("InsertEdit")
sub Page_load()
    strSessionuser = httpContext.current.session("username").ToString()
    if Not Page.IsPostback then
        viewstate("InsertEdit") = 0
        lblTop.text = "Welcome, " & strSessionUser & "!"
        Databind()
        LoadRelationship()
    end if
end sub

sub DataBind()
'lblfirst.text &= "<BR>" & strSessionuser
    myCmd = new sqlCommand("sp_getDependents", myConn)
    myCmd.CommandType = CommandType.StoredProcedure
    myParameter = new sqlParameter("@username", sqlDBType.nvarchar, 50)
    myCmd.parameters.add(myParameter)
    myCmd.Parameters("@username").value = strSessionuser
    try
        myConn.open()
        dg_getDependent.datasource = myCmd.ExecuteReader()
        dg_getDependent.Databind
    catch ex as sqlexception
        response.write(ex.tostring())
    finally
        myConn.close()
    end try
end sub

sub checkbox_checked(a as object, e as eventargs)
    'Show the input table for users
    if cbInsertIndex.checked = true then
```

```
            viewstate("InsertEdit") = 1
            panelDependent.visible = true
        'lblfirst.text &= "<br>strProcess = " & viewstate("InsertEdit")
        end if
    end sub

    sub dg_EditItem(a as object, e as DataGridCommandEventArgs)
        Dim pkCell as TableCell = e.Item.cells(0)
        Dim lastnameCell as TableCell = e.Item.cells(1)
        Dim firstnameCell as TableCell = e.Item.cells(2)
        Dim sexCell as TableCell = e.Item.cells(3)
        Dim dobCell as TableCell = e.item.cells(4)
        Dim ssnCell as TableCell = e.Item.cells(5)
        Dim relationshipCell as TableCell = e.Item.cells(6)
        Dim fulltimestudentCell as TableCell = e.item.cells(7)
        if e.Commandsource.CommandName = "Edited_Row" then
            viewstate("InsertEdit") = 2
            panelDependent.visible = true
            viewstate("DependentPK") = pkCell.text
            txtdeplastname.text = lastnameCell.text
            txtdepfirstname.text = firstnameCell.text
            ddldepsex.selectedvalue = sexCell.text
            txtdepdob.text = dobCell.text
            txtdepssn.text = ssnCell.text
            Dim i as integer = 0
            Dim li as ListItem
            for each li in ddldeprelationship.items
                if li.text = relationshipCell.text then
                    ddldeprelationship.selectedindex = i
                end if
                i = i + 1
            next
            sldepfs.checked = boolean.parse(fulltimestudentCell.text)
        end if
    end sub

    sub LoadRelationship()
        ddldeprelationship.DataSource = (clsdb.getRelation()).defaultView
        ddldeprelationship.DataTextField = "relationship"
        ddldeprelationship.DataValueField = "pk_relation"
        ddldeprelationship.Databind
        'Insert a prompt at the first selection
```

```
    ddldeprelationship.Items.Insert(0,"select a relationship")
    ddldeprelationship.selectedIndex = 0
end sub

public Function getCharBit(byVal blnValue as boolean) as integer
    Dim intBit as integer
    if blnValue = true
        intBit = 1
        return intBit
    else
        intBit = 0
        return intBit
    end if
end Function
sub SaveDependent_Click(sender as object, e as EventArgs)
    """""""""""""""""ViewState 1 --Insert a new value"""""""""""""""""""
  if cint(viewstate("InsertEdit")) = 1 then
    myCmd = new SqlCommand("sp_InsertNewDependent", myConn)
    myCmd.CommandType = CommandType.storedProcedure
    myCmd.Parameters.add(new                    sqlParameter("@lastname",
sqlDbType.nvarchar,35))
    myCmd.Parameters("@lastname").value = txtdeplastname.text
    myCmd.Parameters.Add(new sqlparameter("@firstname", sqlDbType.nvarchar,
35))
    myCmd.Parameters("@firstname").value = txtdepfirstname.text
    myCmd.Parameters.Add(new SqlParameter("@gender", sqlDBType.nvarchar,
1))
    myCmd.Parameters("@gender").value = ddldepsex.selectedvalue
    myCmd.Parameters.Add(new                    Sqlparameter("@dob",
sqlDBType.smallDateTime,4))
    myCmd.Parameters("@dob").value = txtdepdob.text
    myCmd.Parameters.Add(new sqlparameter("@fk_relation", sqlDBType.int, 4))
    myCmd.Parameters("@fk_relation").value = ddldeprelationship.selectedvalue
    myCmd.Parameters.Add(new sqlParameter("@dependentcode", sqlDBtype.int,
4))
    myCmd.parameters("@dependentcode").value = dg_getDependent.items.count
+ 1
    myCmd.Parameters.Add(new SqlParameter("@ssn", sqlDBType.nvarchar,11))
    myCmd.Parameters("@ssn").value = txtdepssn.text
    myCmd.Parameters.Add(new SqlParameter("@date_created", & _
            sqlDBType.smallDateTime, 4))
    myCmd.Parameters("@date_created").value = dateTime.now
```

```
myCmd.Parameters.Add(new SqlParameter("@date_modified", & _
        sqlDBType.smallDateTime, 4))
myCmd.Parameters("@date_modified").value = dateTime.now
myCmd.Parameters.Add(new SqlParameter("@IsArchived", & _
        sqlDBType.nvarchar, 1))
myCmd.Parameters("@IsArchived").value = 0
myConn.open()
    myCmd.executeNonQuery()
    'lblfirst.text &= "<br>Insert into INSURED table successfully."
catch ex as sqlException
    lblfirst.text &= "ERROR: fail to insert a record into " & _
        "INSURED. <br> " & ex.tostring()
Finally
    myConn.close()
end try

'get PK of new entry for the dependent table
Dim strPKInsured as string
Dim intPKInsured as integer
strPKInsured = "select pk_insured from insured where last_name = '" &
        txtdeplastname.text & "' and first_name='" & txtdepfirstname.text & "'"
try
    myConn.open()
    myCmd = new sqlCommand(strPKInsured, myConn)
    myDataReader = myCmd.executeReader()
    if myDataReader.read() then
        intPKInsured = myDataReader.getInt32(0)
    else
        intPKInsured = 0
    end if
catch ex as sqlexception
    lblfirst.text &= "<br>error: fail to get PK of insured" & ex.tostring()
Finally
    myDataReader.close()
    myConn.close()
end try
'get insured primary key of the new entry
Dim strPKInsuredPrimary as string
Dim intPKInsuredPrimary as integer
strPKInsuredPrimary = "select pk_insured from insured i, [user] u " & _
    "where u.pk_user = i.fk_usertable and u.username='" & strSessionuser & "'"
try
```

```
            myConn.open()
            myCmd = new sqlCommand(strPKInsuredPrimary, myConn)
            myDataReader = myCmd.executeReader()
            if myDataReader.read() then
                intPKInsuredPrimary = myDataReader.getInt32(0)
            else
                intPKInsuredPrimary = 0
            end if
        catch ex as sqlexception
            lblfirst.text &= "<BR>ERROR: fail to get PK of insured Primary." & _
                ex.tostring()
        Finally
            myDataReader.close()
            myConn.close()
        end try
        myCmd = new SqlCommand("sp_InsertNewDependentToDependentTable",
            myConn)
        myCmd.CommandType = CommandType.storedProcedure
        myCmd.Parameters.Add(new SqlParameter("@fk_insureddependent", _
            sqlDBType.int, 4))
        myCmd.Parameters("@fk_insureddependent").value =    intPKInsured
        myCmd.Parameters.Add(new SqlParameter("@fk_insuredprimary", " & _
            sqlDBType.int, 4))
        myCmd.Parameters("@fk_insuredprimary").value =    intPKInsuredPrimary
        myCmd.Parameters.Add(new SqlParameter("@date_created", " & _
            sqlDBType.smalldatetime, 4))
        myCmd.Parameters("@date_created").value =    datetime.now
        myCmd.Parameters.Add(new SqlParameter("@date_modified", " & _
            sqlDBType.smalldatetime, 4))
        myCmd.Parameters("@date_modified").value =    datetime.now
        myCmd.Parameters.Add(new SqlParameter("@full_time_student", " & _
            sqlDBType.char, 1))
        myCmd.Parameters("@full_time_student").value                              =
getCharBit(sldepfs.checked)
        myCmd.Parameters.Add(new SqlParameter("@IsArchived", " & _
            sqlDBType.nvarchar, 1))
        myCmd.Parameters("@IsArchived").value = 0
        try
            if intPKInsuredPrimary = 0 or intPKInsured = 0 then
            exit sub
            else
            myConn.open()
```

```
        myCmd.executeNonQuery()
        end if
    catch ex as sqlException
        lblfirst.text &= "ERROR: fail to insert a dependent 2. <br> " & ex.tostring()
    Finally
        myConn.close()
    end try
    clearBoxValue()
    panelDependent.visible    = false
    """""""""""""ViewState 2"""""""""""""""""""""""""""""""""""""""""""""
    else if cint(viewstate("InsertEdit")) = 2 then
    strDependentPK = viewstate("DependentPK")
    'get the pk_insured PK in the Insured table
    strQuery = "select fk_insureddependent from dependents " & _
        "where pk_dependents =" & cint(strDependentPK)
        Dim intInsuredPK as integer
        try
            myConn.open()
            myCmd = new SqlCommand(strQuery, myConn)
            myDataReader    = myCmd.ExecuteReader()
            if myDataReader.read() then
                intInsuredPK = myDataReader.getInt32(0)
            end if
        catch ex as sqlexception
            lblfirst.text &= "<BR>get pk_insured error" & ex.tostring()
        end try
        myConn.close()
        'Update the Insured table first
        strQuery = "update Insured set last_name = @last_name, first_name" & _
            " = @first_name, ssn = @ssn, dob = @dob, gender =@gender, " & _
            "fk_relation = @fk_relation, date_modified = @date_modified " & _
            "where pk_insured= " & intInsuredPK
        myCmd = new SqlCommand(strQuery, myConn)
        myCmd.Parameters.add(new SqlParameter("@last_name", " & _
            "sqlDBType.nvarchar, 35))
        myCmd.parameters("@last_name").value = txtdeplastname.text
        myCmd.Parameters.add(new SqlParameter("@first_name",
             "sqlDBType.nvarchar, 35))
        myCmd.Parameters("@first_name").value = txtdepfirstname.text
        myCmd.Parameters.add(new   SqlParameter("@ssn",   sqlDBtype.nvarchar,
11))
        myCmd.Parameters("@ssn").value = txtdepssn.text
```

```
        myCmd.parameters.add(new SqlParameter("@dob",
            sqlDBType.smalldatetime,4))
        myCmd.Parameters("@dob").value = Convert.ToDateTime(txtdepdob.text)
        myCmd.Parameters.add(new SqlParameter("@gender", sqlDBtype.nvarchar,
1))
        myCmd.Parameters("@gender").value = ddldepsex.selectedvalue
        myCmd.Parameters.add(new  SqlParameter("@fk_relation",  sqlDBtype.int,
4))
        myCmd.Parameters("@fk_relation").value                              =
ddldeprelationship.selectedvalue
        myCmd.Parameters.add(new sqlParameter("@date_modified",
                sqlDBtype.smalldatetime, 4))
        myCmd.Parameters("@date_modified").value = DateTime.Now
        try
            myConn.open()
            myCmd.executeNonQuery()
            'lblfirst.text &= "<br>Update INSURED table successfully."
        catch ex as sqlException
            lblfirst.text &= "<BR>ERROR: fail to update INSURED table. "
                <br> " & ex.tostring()
        Finally
            myConn.close()
        end try
        'Update Dependent table
        strQuery = "update Dependents set full_time_student = @fulltime_student,
"
                "date_modified = @date_modified where pk_dependents = " &
                cint(strDependentPK)
        myCmd = new SqlCommand(strQuery, myConn)
        myCmd.Parameters.add(new SqlParameter("@fulltime_student",
            sqlDBtype.nvarchar,1))
        myCmd.Parameters("@fulltime_student").value                         =
getCharBit(sldepfs.checked)
        myCmd.Parameters.add(new sqlParameter("@date_modified",
            sqlDBType.smalldatetime, 4))
        myCmd.Parameters("@date_modified").value = DateTime.Now
        try
            myConn.open()
            myCmd.executeNonQuery()
            'lblfirst.text &= "<br>Update into DEPENDENT table successfully."
        catch ex as sqlException
            lblfirst.text &= "<BR>ERROR: fail to update DEPENDENT table.
```

```
                <br> " ex.tostring()
        Finally
            myConn.close()
        end try
        clearBoxValue()
        panelDependent.visible    = false
    else
        lblfirst.text &= "Error be here"
    end if

    DataBind()
end sub

sub clearBoxValue()
    txtdeplastname.text = ""
    txtdepfirstname.text =""
    txtdepssn.text =""
    txtdepdob.text = ""
    ddldeprelationship.selectedIndex = 0
    ddldepsex.selectedIndex = 0
    sldepfs.checked = false
end sub

sub CancelinsertDependent_Click(sende as object, e as EventArgs)
    clearBoxValue()
    panelDependent.visible    = false
end sub
</script>
```

MemProviderInfo.aspx

```
<script language=VB runat=server>
public strSessionUser as string          'username carrier from the previous page
public strQuery as string                'query srtring
public strErrorMsg as string             'error message string
public clsdb as new clsMedasolution              'declare the dll file
public myConn as new SqlConnection(clsdb.connectionString)
public myCmd as sqlCommand
public myDataReader as sqlDataReader
public myParameter as sqlparameter
sub Page_load(a as object, e as eventargs)
    strSessionuser = httpContext.current.session("username").ToString()
```

```
    lblTop.text = "Welcome, " & strSessionuser & " !"
    if Not Page.IsPostBack then
        if session("classID") <> 2 then
            response.write("../employHome.aspx")
            'response.end
        end if
        LoadEmploymentstatus()
        LoadBenefitplans()
    end if

end sub

sub searchEmployee_Click(a as object, e as Eventargs)
    Dim strNeedSearch as string
    if txtsearchname.text <> "" then
        strNeedSearch = "Need Search function"
        DataFiller(strNeedSearch)
        if dg_search.items.count <= 0 then
            'lblfirst.text &= "<Br>No Data is found!"
            panel_display.visible = false
            dg_search.visible = false
        else
            panel_display.visible = true
            dg_search.visible = true
        end if
    else
        lblResult.text = " Need input a name"
        panel_display.visible = true
        dg_search.visible= false
        lblResult.visible = true
        exit sub
    end if
end sub

sub DataFiller(optional byVal strWhere as string = "")
    if strSessionuser <> "" then
        if strWhere <> "" then
            strQuery = "sp_ShowAllEmployeeForSearch"
        else
            strQuery = "sp_ShowAllEmployeeInGroup"
        end if
        myCmd = new sqlCommand(strQuery, myConn)
```

```
        myCmd.CommandType = CommandType.storedProcedure
        myParameter = new sqlParameter("@username", sqlDBType.nvarchar,35)
        myCmd.Parameters.add(myParameter)
        myCmd.Parameters("@username").value = strSessionuser
        if strWhere <> "" then
            myCmd.Parameters.add(new sqlParameter("@lastname",
                sqldbtype.nvarchar, 35))
            myCmd.Parameters("@lastname").value = txtsearchName.text
        end if
        try
            myConn.open()
            dg_search.datasource = myCmd.executeReader()
            dg_search.dataBind()
        catch ex as sqlexception
            lblfirst.text &= "<br>Error: fail to show All Records<br>" &
ex.Tostring()     finally
            myConn.close
        end try
    else
        lblfirst.text &= "<BR>Need a login name."
        response.end
    end if
end sub

sub EditDeleteRow_Click(a as object, e as DataGridCommandEventArgs)
    if e.Item.ItemIndex > -1   and e.CommandSource.CommandName =
            "EditThis" then
        'set the viewstate
        viewstate("InsertEdit") = 2          'edit state
        dg_search.selectedIndex = e.Item.ItemIndex
        Dim dgItem as DataGridItem
        dgItem = dg_search.selectedItem
        'declare tablecell and load value from datagrid to tablecells
        Dim pkEligibilityCell as TableCell = dgItem.controls(0)
        Dim lastnameCell as TableCell = dgItem.controls(1)
        Dim firstnameCell as TableCell = dgItem.controls(2)
        Dim employmentstatusCell as TableCell = dgItem.controls(3)
        Dim benefitplanCell as TableCell = dgItem.controls(4)
        Dim hiredateCell as TableCell = dgItem.Controls(5)
        Dim effdateCell as TableCell = dgItem.Controls(6)
        Dim termdateCell as TableCell = dgItem.Controls(7)
        'bind tablecell value to table value
```

```
        viewstate("pk_eligibility") = pkEligibilityCell.text
        txtlastname.text = lastnameCell.text
        txtfirstname.text = firstnameCell.text
        Dim i as integer =   0
        Dim li as listitem
        txthiredate.text = hiredateCell.text
        txteffdate.text = effdateCell.text
        txttermdate.text = termdatecell.text
        panel_EditInsertTable.visible=true
    else if e.Item.itemindex > -1 and e.CommandSource.Commandname =
            "DeleteThis" then
        dg_search.selectedIndex = e.Item.ItemIndex
        Dim dgItem as DataGridItem
        dgItem = dg_search.selectedItem
        Dim PKEligibilityCell as TableCell = dgItem.Controls(0)
        Dim intPKEligibility as integer = Cint(PKEligibilityCell.text)
        Dim btnDelete as Button   = ctype(e.item.cells(9).controls(0), button)
        btnDelete.attributes.add("onClick", "return confirm_Delete();")

        'get the insured pk from eligibility table
        strQuery = "select fk_insured from eligibility where pk_eligibility ="
            & intPKEligibility
        Dim intPKInsuredValue as integer
        intPKInsuredValue = clsdb.ExecScalar(strQuery, strErrorMsg)
        if strErrorMsg <> "" then
            lblfirst.text &= "<br>Error: failed to get fk_insured from eligibility
table. "
        end if
        'Delete the record in Eligibility table
        Dim intDeleteValue as integer
        strQuery = "delete from Eligibility where pk_eligibility =" &
intPKEligibility
        intDeleteValue = clsdb.ExecSQL(strQuery, false, strErrorMsg)
        if intDeleteValue = -1 then
            lblfirst.text &= "<br>Error: failed to delete a record into
                ELIGIBILITY TABLE."
        end if
        'Delete the record in the insured table
        strQuery = "delete from insured where pk_insured=" & intPKInsuredValue
        Dim intDeleteInsuredRecord as integer= clsdb.execSQL(strQuery,
                false, strErrorMsg)
        if strErrorMsg <> "" then
```

```
            lblfirst.text &= "<BR>Error: failed to delete a record in
                INSURED TABLE."
        end if
        dataFiller()
    end if
end sub

sub BtnInsert_Click(a as object, e as Eventargs)
    'set the viewstate
    viewstate("InsertEdit") = 1                'Insert state
    'Show the display panel firs
    if panel_display.visible = false then
        DataFiller()
        panel_display.visible =true
    end if
    if dg_search.columns(8).visible = true then
            dg_search.columns(8).visible = false
    end if
    txtlastname.readonly=false
    txtfirstname.readonly =false
    panel_EditInsertTable.visible=true
end sub

sub BtnEdit_Click(a as object, e as Eventargs)
    'Show the display panel firs
    if panel_display.visible = false then
        DataFiller()
        panel_display.visible =true
    end if
    'show/hide the last column in the datagrid
    dg_search.columns(9).visible = false
    dg_search.columns(8).visible = true
end sub

sub BtnDelete_Click(a as object, e as Eventargs)
    if strSessionuser <> "" then
        DataFiller()
        panel_display.visible = true
        dg_search.visible = true
        dg_search.columns(8).visible = false
        dg_search.columns(9).visible = true
    end if
```

```
    end sub

sub Save_click(a as object, e as eventargs)
    """""""""""""In Edit state """""""""""""""""""""""""""""""
    if viewstate("InsertEdit") = 2 and Page.Isvalid then
        callEditSaveRecord()
    """""""""""""In Insert state""""""""""""""""""""""""""""""""""""""""
    else if viewState("InsertEdit") = 1    and Page.Isvalid then
        call save_reocrd()
        clearTextBoxValue()
        DataFiller()
        panel_EditInsertTable.visible= false
    end if
end sub
</script>
```

EmploerReview.aspx

```
<script language="VB" runat=server>
public strSessionuser as string              'declare sessionuser
public clsdb as new clsMedasolution          'declare clsMedasolution class object
public myConn as new sqlConnection(clsdb.connectionString)
public myCmd as sqlcommand
public myDataReader as sqlDataReader
public strQuery as string
public strErrorMsg as string                 'error message string
sub Edit_Click(a as object, e as Eventargs)
    ControlReadonlyState()
end sub

sub Save_Click(a as object, e as eventargs)
    call saveRecord()
    controlreadonlystate()
    dataFiller()
end sub

sub ControlReadonlyState()
    txtaddress1.readonly= not txtaddress1.readonly
    txtaddress2.readonly = not txtaddress2.readonly
    txtcity.readonly = not txtcity.readonly
    txtzipcode.readonly = not txtzipcode.readonly
    txtphone1.readonly = not txtphone1.readonly
```

```
    txtphone2.readonly = not txtphone2.readonly
    txtext1.readonly = not txtext1.readonly
    txtext2.readonly = not txtext2.readonly
    txtfax.readonly = not txtfax.readonly
    txtcontact1.readonly = not txtcontact1.readonly
    txtcontact1email.readonly = not txtcontact1email.readonly
    txtcontact2.readonly = not txtcontact2.readonly
    txtcontact2email.readonly = not txtcontact2email.readonly
    txtstate.readonly = not txtstate.readonly
    if txtstate.readonly = false then
        txtstate.visible = false
        ddlstate.visible = true
    else
        txtstate.visible = true
        ddlstate.visible = false
    end if
    btnSave.visible = not btnSave.visible
    btnCancel.visible = not btnCancel.visible
    btnEdit.visible = not btnEdit.visible
end sub
</script>
```

EmployeeInfo.aspx

```
<script language=VB runat=server>
public strSessionUser as string          'username carrier from the previous page
public strQuery as string                'query srtring
public strErrorMsg as string             'error message string
public clsdb as new clsMedasolution              'declare the dll file
public myConn as new SqlConnection(clsdb.connectionString)
public myCmd as sqlCommand
public myDataReader as sqlDataReader
public myParameter as sqlparameter
sub searchEmployee_Click(a as object, e as Eventargs)
    Dim strNeedSearch as string
    if txtsearchname.text <> "" then
        strNeedSearch = "Need Search function"
        DataFiller(strNeedSearch)
        if dg_search.items.count <= 0 then
            'lblfirst.text &= "<Br>No Data is found!"
            panel_display.visible = false
            dg_search.visible = false
        else
```

```
            panel_display.visible = true
            dg_search.visible = true
        end if
    else
        lblResult.text = " Need input a name"
        panel_display.visible = true
        dg_search.visible= false
        lblResult.visible = true
        exit sub
    end if
end sub

sub DataFiller(optional byVal strWhere as string = "")
    if strSessionuser <> "" then
        if strWhere <> "" then
            strQuery = "sp_ShowAllEmployeeForSearch"
        else
            strQuery = "sp_ShowAllEmployeeInGroup"
        end if
        myCmd = new sqlCommand(strQuery, myConn)
        myCmd.CommandType = CommandType.storedProcedure
        myParameter = new sqlParameter("@username", sqlDBType.nvarchar,35)
        myCmd.Parameters.add(myParameter)
        myCmd.Parameters("@username").value = strSessionuser

        if strWhere <> "" then
            myCmd.Parameters.add(new sqlParameter("@lastname",
                sqldbtype.nvarchar, 35))
            myCmd.Parameters("@lastname").value = txtsearchName.text
        end if
        try
            myConn.open()
            dg_search.datasource = myCmd.executeReader()
            dg_search.dataBind()
        catch ex as sqlexception
            lblfirst.text &= "<br>Error: fail to show All Records<br>" &
ex.Tostring()     finally
            myConn.close
        end try
    else
        lblfirst.text &= "<BR>Need a login name."
        response.end
```

```
    end if
end sub

sub BtnShowAll_Click(a as object, e as Eventargs)
    if panel_display.visible=false then
        DataFiller()
        panel_EditInsertTable.visible=false
        panel_display.visible=true
    else
        dataFiller()
        if dg_search.columns(8).visible = true then
            dg_search.columns(8).visible = false
        end if
    end if
end sub

sub EditDeleteRow_Click(a as object, e as DataGridCommandEventArgs)
    if e.Item.ItemIndex > -1    and e.CommandSource.CommandName =
            "EditThis" then
        'set the viewstate
        viewstate("InsertEdit") = 2          'edit state
        dg_search.selectedIndex = e.Item.ItemIndex
        Dim dgItem as DataGridItem
        dgItem = dg_search.selectedItem
        'declare tablecell and load value from datagrid to tablecells
        Dim pkEligibilityCell as TableCell = dgItem.controls(0)
        Dim lastnameCell as TableCell = dgItem.controls(1)
        Dim firstnameCell as TableCell = dgItem.controls(2)
        Dim employmentstatusCell as TableCell = dgItem.controls(3)
        Dim benefitplanCell as TableCell = dgItem.controls(4)
        Dim hiredateCell as TableCell = dgItem.Controls(5)
        Dim effdateCell as TableCell = dgItem.Controls(6)
        Dim termdateCell as TableCell = dgItem.Controls(7)
        'bind tablecell value to table value
        viewstate("pk_eligibility") = pkEligibilityCell.text
        txtlastname.text = lastnameCell.text
        txtfirstname.text = firstnameCell.text
        Dim i as integer =    0
        Dim li as listitem
        For each li in ddlemploymentstatus.items
            if li.text = employmentstatusCell.text
                ddlemploymentstatus.selectedindex = i
```

```
            end if
            i = i + 1
        next
        i = 0
        For each li in ddlbenefitplan.items
            if li.text = benefitplanCell.text then
                ddlbenefitplan.selectedIndex = i
            end if
            i = i + 1
        Next
        txthiredate.text = hiredateCell.text
        txteffdate.text = effdateCell.text
        txttermdate.text = termdatecell.text
        panel_EditInsertTable.visible=true
    else if e.Item.itemindex > -1 and e.CommandSource.Commandname =
            "DeleteThis" then
        dg_search.selectedIndex = e.Item.ItemIndex
        Dim dgItem as DataGridItem
        dgItem = dg_search.selectedItem
        Dim PKEligibilityCell as TableCell = dgItem.Controls(0)
        Dim intPKEligibility as integer = Cint(PKEligibilityCell.text)
        Dim btnDelete as Button    = ctype(e.item.cells(9).controls(0), button)
        btnDelete.attributes.add("onClick", "return confirm_Delete();")

        'get the insured pk from eligibility table
        strQuery = "select fk_insured from eligibility where pk_eligibility ="
            & intPKEligibility
        Dim intPKInsuredValue as integer
        intPKInsuredValue = clsdb.ExecScalar(strQuery, strErrorMsg)
        if strErrorMsg <> "" then
            lblfirst.text &= "<br>Error: failed to get fk_insured from eligibility
table. "
        end if

        'Delete the record in Eligibility table
        Dim intDeleteValue as integer
        strQuery = "delete from Eligibility where pk_eligibility =" &
intPKEligibility
        intDeleteValue = clsdb.ExecSQL(strQuery, false, strErrorMsg)
        if intDeleteValue = -1 then
            lblfirst.text &= "<br>Error: failed to delete a record into
                ELIGIBILITY TABLE."
```

```
            end if
            'Delete the record in the insured table
            strQuery = "delete from insured where pk_insured=" & intPKInsuredValue
            Dim intDeleteInsuredRecord as integer= clsdb.execSQL(strQuery,
                false, strErrorMsg)
            if strErrorMsg <> "" then
                lblfirst.text &= "<BR>Error: failed to delete a record in
                    INSURED TABLE."
            end if
            dataFiller()
        end if
    end sub

    sub BtnInsert_Click(a as object, e as Eventargs)
        'set the viewstate
        viewstate("InsertEdit") = 1                'Insert state
        'Show the display panel firs
        if panel_display.visible = false then
            DataFiller()
            panel_display.visible =true
        end if
        if dg_search.columns(8).visible = true then
                dg_search.columns(8).visible = false
        end if
        txtlastname.readonly=false
        txtfirstname.readonly =false
        panel_EditInsertTable.visible=true
    end sub

    sub BtnEdit_Click(a as object, e as Eventargs)
        'Show the display panel firs
        if panel_display.visible = false then
            DataFiller()
            panel_display.visible =true
        end if
        'show/hide the last column in the datagrid
        dg_search.columns(9).visible = false
        dg_search.columns(8).visible = true
    end sub

    sub BtnDelete_Click(a as object, e as Eventargs)
        if strSessionuser <> "" then
```

```
            DataFiller()
            panel_display.visible = true
            dg_search.visible = true
            dg_search.columns(8).visible = false
            dg_search.columns(9).visible = true
        end if
        'panel_EditInsertTable.visible=false
end sub

sub Save_click(a as object, e as eventargs)
        """""""""""""In Edit state """""""""""""""""""""""""""""""
        if viewstate("InsertEdit") = 2 and Page.Isvalid then
            myCmd = new sqlCommand("sp_updateEligibilityForGroup", myConn)
            myCmd.CommandType = CommandType.storedProcedure
            myParameter = new SqlParameter("@pk_eligibility", sqldbtype.int)
            myParameter.value = viewstate("pk_eligibility")
            myCmd.Parameters.add(myParameter)
            myParameter = new sqlParameter("@fk_employmentstatus", sqldbtype.int)
            myParameter.value = ddlemploymentstatus.selectedValue
            myCmd.Parameters.add(myParameter)
            myParameter = new sqlParameter("@fk_benefitplans", sqldbtype.int)
            myParameter.value = ddlbenefitplan.selectedValue
            myCmd.Parameters.add(myParameter)
            myParameter = new sqlParameter("@hiredate", sqldbtype.smalldatetime)
            myParameter.value = txthiredate.text
            myCmd.Parameters.add(myParameter)
            myParameter = new sqlParameter("@eff_date", sqldbtype.smalldatetime, 4)
            myParameter.value = txteffdate.text
            myCmd.Parameters.add(myParameter)
            myParameter = new sqlParameter("@term_date", sqldbtype.smalldatetime,
4)
            myParameter.value = txttermdate.text
            myCmd.Parameters.add(myParameter)
            myParameter = new sqlParameter("@date_modified", sqldbtype.
                smalldatetime, 4)
            myParameter.value = dateTime.now
            myCmd.Parameters.add(myParameter)
            try
                myConn.open
                lblfirst.text &= "<BR>record effects is " & myCmd.executeNonQuery()
            catch ex as sqlexception
                lblfirst.text &="Error: Fail to update eligibility table." & ex.Tostring()
```

```
        finally
            myConn.close
        end try
    """""""""""""In Insert state"""""""""""""""""""""""""""""""""""""""""""""
    else if viewState("InsertEdit") = 1    and Page.Isvalid then
    saveButton.attributes.add("onClick", "return btnsave_click();" )
        'insert value to insured table
        myCmd = new sqlCommand("sp_InsertInsuredForGroup", myConn)
        myCmd.CommandType = Commandtype.storedProcedure
        myCmd.parameters.add(new sqlParameter("@last_name",
            sqldbtype.nvarchar, 35))
        myCmd.Parameters("@last_name").value    = txtlastname.text
        myCmd.Parameters.add(new sqlParameter("@first_name",
                sqldbtype.nvarchar, 35))
        myCmd.Parameters("@first_name").value = txtfirstname.text
        myCmd.Parameters.add(new sqlParameter("@date_created",
                sqldbtype.smalldatetime,4))
        myCmd.Parameters("@date_created").value = datetime.now
        myCmd.Parameters.add(new sqlParameter("@date_modified",
                sqldbtype.smalldatetime,4))
        myCmd.Parameters("@date_modified").value = dateTime.now
        myCmd.Parameters.add(new                    sqlParameter("@fk_relation",
sqldbtype.int,4))
        myCmd.Parameters("@fk_relation").value = 1
        myCmd.Parameters.add(new sqlParameter("@IsArchived",
                sqldbtype.nvarchar, 1))
        myCmd.Parameters("@IsArchived").value = 0
        Dim intReturnValue as integer
        try
            myConn.open()
            intReturnValue = myCmd.executeNonQuery()
            'lblfirst.text &= "<BR>insert insured sucessfully = " & intReturnValue
        catch ex as sqlexception
            lblfirst.text &= "<BR>Error: Fail to insert to INSURED table."
        Finally
            myConn.close
        end try
        'get the pk_insured for the new insert insured
        strQuery = "select pk_insured from insured where last_name='"
            & txtlastname.text "' and first_name='" & txtfirstname.text & "'"
        Dim intPKInsured as integer = clsdb.ExecScalar(strQuery, strErrorMsg)
        'get the group pk
```

```
        strQuery = "select g.pk_group from [group] g, [user] u " & _
                   " where u.pk_user = g.fk_usertable " & _
                   " and u.username ='" & strSessionuser & "'"
        Dim intPKgroup as integer = clsdb.ExecScalar(strQuery, strErrorMsg)
        'get location pk
        strQuery = "select fk_location from [group] g, [user] u " & _
                " where u.pk_user = g.fk_usertable " & _
                " and u.username = '" & strsessionuser & "'"
        Dim intPKlocation as integer = clsdb.ExecScalar(strQuery, strErrorMsg)
        'INSERT INTO THE ELIGIBILITY TABLE
        if intPKInsured > 0 and intPKgroup <> 0 and intPKlocation <> 0 then
            myCmd = new sqlCommand("sp_InsertEligibilityForGroup", myConn)
            myCmd.CommandType = CommandType.storedProcedure
            myParameter = new sqlParameter("@fk_insured", sqldbtype.int)
            myParameter.value = intPKInsured
            myCmd.Parameters.add(myParameter)
            myParameter = new sqlParameter("@fk_benefitplans", sqldbtype.int)
            myParameter.value = ddlbenefitplan.selectedvalue
            myCmd.Parameters.add(myParameter)
            myParameter = new sqlParameter("@fk_group", sqldbtype.int)
            myParameter.value = intPKgroup
            myCmd.Parameters.add(myParameter)
            myParameter = new sqlParameter("@fk_location", sqldbtype.int)
            myParameter.value = intPKlocation
            myCmd.Parameters.add(myParameter)
            myParameter = new sqlParameter("@fk_employmentstatus",
sqldbtype.int)
            myParameter.value = ddlemploymentstatus.selectedValue
            myCmd.Parameters.add(myParameter)
            myParameter = new sqlParameter("@hiredate",
sqldbtype.smalldatetime)
            myParameter.value = txthiredate.text
            myCmd.Parameters.add(myParameter)
            myParameter = new sqlParameter("@eff_date",
sqldbtype.smalldatetime)
            myParameter.value = txteffdate.text
            myCmd.Parameters.add(myParameter)
            myParameter = new sqlParameter("@term_date",
sqldbtype.smalldatetime)
            myParameter.value = txttermdate.text
            myCmd.Parameters.add(myParameter)
            myParameter = new sqlParameter("@date_created",
```

```
            sqldbtype.smalldatetime)
        myParameter.value = Datetime.now
        myCmd.parameters.add(myParameter)
        myCmd.Parameters.add(new sqlParameter("@date_modified", _
            sqldbtype.smalldatetime))
        myCmd.Parameters("@date_modified").value = DateTime.now
        myParameter = new sqlParameter("@most_current",
sqldbtype.nvarchar,1)
        myParameter.value = 1
        myCmd.Parameters.add(myParameter)
        myParameter = new sqlParameter("@IsArchived", sqldbtype.nvarchar,
1)
        myParameter.value = 0
        myCmd.Parameters.add(myParameter)

        Dim intReturnResult as integer
        try
            myConn.open()
            intReturnResult = myCmd.executeNonQuery()
            'lblfirst.text &= "<BR>insert ELIGIBILITY sucessfully = "
                & intReturnResult
        catch ex as sqlexception
            lblfirst.text &= "<BR>Error: Fail to insert to
                ELIGIBILITY table.<BR>"
             & ex.Tostring()
        Finally
            myConn.close
        end try
        'INSERT A RECORD INTO WEBMASTER TABLE
        myCmd = new sqlCommand("sp_InsertWebMasterFromGroup",
myConn)
        myCmd.commandtype = commandtype.storedProcedure
        myCmd.parameters.add(new sqlParameter("@insuredID",
sqldbtype.int))
        myCmd.Parameters("@insuredID").value = intPKInsured
        myCmd.parameters.add(new sqlParameter("@requestBy",
sqldbtype.int))
        myCmd.Parameters("@requestBy").value = intPKgroup
        myCmd.parameters.add(new sqlParameter("@requestDate",
            sqldbtype.smalldatetime))
        myCmd.Parameters("@requestDate").value = datetime.now()
        Dim intWebMaster as integer
```

```
            try
                myConn.open()
                intWebMaster = myCmd.executeNonQuery()
            catch ex as sqlexception
                lblfirst.text &= "<BR>Error: Fail to insert to
                    ELIGIBILITY table.<BR>" & ex.Tostring()
            Finally
                myConn.close
            end try
            'if strErrorMsg <> "" then
                'lblfirst.text &="<BR>Error: failed to insert into webmaster."
            'end if
        else
            lblfirst.text   &= "<BR> intPKInsured = " & intPKInsured & "
                "cant insert eligibility table."
        end if
    else
        lblfirst.text &= "<br>Impossible to reach there"
    end if
    clearTextBoxValue()
    DataFiller()
    panel_EditInsertTable.visible= false
end sub
</script>
```

BrokReview.aspx

```
<script language="VB" runat=server>
public strSessionuser as string             'declare sessionuser
public clsdb as new clsMedasolution         'declare clsMedasolution class object
public myConn as new sqlConnection(clsdb.connectionString)
public myCmd as sqlcommand
public myDataReader as sqlDataReader
public strQuery as string
public strErrorMsg as string                'error message string
sub Page_Load(a as object, e as EventArgs)
    strSessionuser = httpcontext.current.session("username").Tostring()
    lblTop.text = "Welcome, " & strsessionuser & " !"
    if Not Page.IsPostBack then
        DataFiller()
        LoadState()
    else
```

```
        if strSessionuser = "" or session("classID") <> 3 then
            response.redirect("../brokHome.aspx")
            response.end
        end if
    end if
end sub
sub DataFiller()
strQuery ="select address1, address2,city,state,zip,phone1,phone2,phone1_ext,
phone2_ext,fax," & _
    "primary_contact,secondary_contact,primary_email,secondary_email,
broker_name " & _
    " from broker b, [user] u where b.fk_usertable = u.pk_user and u.username= '" &
strsessionuser & "
    try
    myConn.open
    myCmd = new sqlCommand(strQuery, myConn)
    myDataReader = myCmd.executeReader()
    if myDataReader.read then
        lblcompanyname.text = clsdb.checkfornull(mydataReader("broker_name"))
        txtaddress1.text = clsdb.checkfornull(myDataReader("address1"))
        txtaddress2.text = clsdb.checkForNull(myDataReader("address2"))
        txtcity.text = clsdb.checkForNull(myDataReader("city"))
        txtstate.text = clsdb.checkForNull(myDataReader("state"))
        txtzipcode.text = clsdb.checkForNull(myDataReader("zip"))
        txtphone1.text = clsdb.checkForNull(myDataReader("phone1"))
        txtphone2.text = clsdb.checkForNull(myDataReader("phone2"))
        txtext1.text = clsdb.checkForNull(myDataReader("phone1_ext"))
        txtext2.text = clsdb.checkForNull(myDataReader("phone2_ext"))
        txtfax.text = clsdb.checkForNull(myDataReader("fax"))
        txtcontact1.text = clsdb.checkForNull(myDataReader("primary_contact"))
        txtcontact1email.text                                              =
clsdb.checkForNull(myDataReader("primary_email"))
        txtcontact2.text                                                   =
clsdb.checkForNull(myDataReader("secondary_contact"))
        txtcontact2email.text =
            clsdb.checkForNull(myDataReader("secondary_email"))
    end if
    catch ex as sqlexception
    lblfirst.text &="<br> error: failed to read data."
    end try
    myDataReader.close
    myConn.close
```

```
End sub

sub Edit_Click(a as object, e as Eventargs)
    ControlReadonlyState()
end sub

sub Save_Click(a as object, e as eventargs)
    myCmd = new SqlCommand("sp_updateBroker", myConn)
    myCmd.CommandType = CommandType.storedProcedure
    myCmd.Parameters.add(new sqlParameter("@address1", sqlDBType.nvarchar,
40))
    myCmd.Parameters("@address1").value = txtaddress1.text
    myCmd.Parameters.add(new sqlParameter("@address2", sqlDBType.nvarchar,
40))
    myCmd.Parameters("@address2").value = txtaddress2.text
    myCmd.Parameters.add(new sqlParameter("@city", sqlDBType.nvarchar, 40))
    myCmd.Parameters("@city").value = txtcity.text
    myCmd.Parameters.add(new sqlParameter("@state", sqlDBType.nvarchar, 50))
    myCmd.Parameters("@state").value = ddlstate.selectedItem.value
    myCmd.Parameters.add(new sqlParameter("@zip", sqlDBType.nvarchar, 40))
    myCmd.Parameters("@zip").value = txtzipcode.text
    myCmd.Parameters.add(new sqlParameter("@phone1", sqlDBType.nvarchar,
15))
    myCmd.Parameters("@phone1").value = txtphone1.text
    myCmd.Parameters.add(new sqlParameter("@phone2", sqlDBType.nvarchar,
15))
    myCmd.Parameters("@phone2").value = txtphone2.text
    myCmd.Parameters.add(new sqlParameter("@phone1_ext",
            sqlDBType.nvarchar, 5))
    myCmd.Parameters("@phone1_ext").value = txtext1.text
    myCmd.Parameters.add(new sqlParameter("@phone2_ext",
            sqlDBType.nvarchar, 10))
    myCmd.Parameters("@phone2_ext").value = txtext2.text
    myCmd.Parameters.add(new sqlParameter("@fax", sqlDBType.nvarchar, 15))
    myCmd.Parameters("@fax").value = txtfax.text
    myCmd.Parameters.add(new sqlParameter("@primary_contact",
            sqlDBType.nvarchar, 40))
    myCmd.Parameters("@primary_contact").value = txtcontact1.text
    myCmd.Parameters.add(new sqlParameter("@secondary_contact",
            sqlDBType.nvarchar, 40))
    myCmd.Parameters("@secondary_contact").value = txtcontact2.text
    myCmd.Parameters.add(new sqlParameter("@secondary_email",
```

```
            sqlDBType.nvarchar, 40))
    myCmd.Parameters("@secondary_email").value = txtcontact2email.text
    myCmd.Parameters.add(new sqlParameter("@primary_email",
            sqlDBType.nvarchar, 40))
    myCmd.Parameters("@primary_email").value = txtcontact1email.text
    myCmd.Parameters.add(new sqlParameter("@date_modified",
            sqlDBType.nvarchar, 40))
    myCmd.Parameters("@date_modified").value = DateTime.Now
    myCmd.Parameters.add(new  sqlParameter("@username",  sqlDBType.nvarchar,
35))
    myCmd.Parameters("@username").value = strSessionuser
    Dim intReturnValue as integer
    try
    myConn.open
    intReturnValue = myCmd.ExecuteNonQuery()
    'lblfirst.text &= "<BR>" & intReturnValue.tostring() & " record is updated."
    catch ex as sqlException
    lblfirst.text &= "<BR>Error:failed to update to GROUP table."
    finally
    myconn.close
    end try
    controlreadonlystate()
    dataFiller()
end sub
</script>
```

BrokerCalculationSummary.aspx

```
<script language=VB runat=server>
public clsdb as new clsMedasolution
public strSessionuser as string
public strQuery, strErrorMsg as string              'query string
Dim sum1, sum2 as double 'public runningTotal as double
Dim sum3, sum4 as double
public myConn as new sqlConnection(clsdb.connectionstring)
public myCmd as sqlCommand
public myDataReader as sqlDataReader
public ds as new dataset
public dt as DataTable
public dr as DataRow

sub LoadIndividualCommission()
```

```
    strQuery = " select e.pk_eligibility, i.pk_insured, i.last_name,i.first_name ," & _
            "bp.benefitplan_name, bp.month_rate, e.eff_date, e.term_date , " & _
            "datediff(month,e.eff_date,getDate()) as months_todate, " & _
            "month_rate * datediff(month, e.eff_date, getDate())* 0.15 as " & _
            "upToNowTotal, datediff(month,e.eff_date,e.term_date) as " & _
            "months_to_end,month_rate * datediff(month, e.eff_date, " & _
            "term_date)*0.15 as expectationTotal " & _
            "from insured i, eligibility e, benefitplans bp where i.pk_insured = " &
_
            "e.fk_insured and e.fk_broker = " & _
            "( select pk_broker from broker b, [user] u where b.fk_usertable = " &
_
            "u.pk_user and u.username='" & strsessionuser & "') and e.fk_group
        " & _
            "is null and e.fk_benefitplans = bp.pk_benefitplans and " & _
            "e.most_current = 1 "and getdate() between e.eff_date and e.term_date
        "
    Dim blnReturnValue as boolean = clsdb.FillDataSet(strQuery, ds, " & _
        "IndividualInBroker")
    if blnReturnValue then
        dg_individual.datasource = ds.tables("IndividualInBroker").defaultView
        dg_individual.databind
        panel_individual.visible=true
    else
        lblindividual.text &= "<br><font color=red>You donot have " & _
            "individual client.</fonr>"
    end if
end sub

sub FillGroupDataSet(byval strCurrentuser as string)
    "1. GET DATASET WITH ROWS FROM MEDASOLUTION DATABASE
    Dim strQueryGroupid = "select _group,group_name,primary_contact, " & _
        primary_email, phone1,fax from [group] g where g.fk_broker = ( " & _
        "select pk_broker from broker b, [user] u " & _
       "where u.username ='" & strCurrentuser & "' and u.pk_user = b.fk_usertable)
"
    Dim strQueryGroupInsured = "select e.fk_group, i.last_name,i.first_name, " & _
        "bp.benefitplan_name, bp.month_rate, e.eff_date, e.term_date, " & _
        "datediff(month,e.eff_date,getDate()) as months_todate, " _
        "month_rate * datediff(month, e.eff_date, getDate())* 0.15 as
upToNowTotal,
        "datediff(month,e.eff_date,e.term_date) as months_to_end, " & _
```

```
        "month_rate * datediff(month, e.eff_date, term_date)*0.15 as
expectationTotal
        "from eligibility e, benefitplans bp, insured i " & _
        "where e.fk_insured = i.pk_insured and e.fk_benefitplans =
p.pk_benefitplans
        "and e.fk_group in ( select pk_group from [group] g where g.fk_broker = ( "
        "select pk_broker from broker b, [user] u " & _
       "where u.username ='" & strCurrentuser & "' and u.pk_user = b.fk_usertable)
"
        ") and e.most_current = 1 " & _
        "and getdate() between e.eff_date and e.term_date "
    "2.CONNECT DATABASE AND EXECUTE QUERY
    Dim blnGroupId as boolean = clsdb.FillDataSet(strQueryGroupid,
ds,"GroupID")
    Dim blnGroupInsured as boolean = clsdb.FillDataset(strQueryGroupInsured, ds,
"GroupInsured")
    "3.SETUP THE RELATIONSHIP BETWEEN TWO TABLES
    if blnGroupId and blnGroupInsured then
    Dim odatRel as new DataRelation("TwoTables",
            ds.Tables("GroupID").columns("pk_group"), _
            ds.Tables("GroupInsured").columns("fk_group"))
    ds.relations.add(odatRel)
    end if
end sub

sub LoadGroupCommission()
    "1.GET COUNTS OF GROUP DEVELOPED BY THIS BROKER
    strQuery = "select count(*) from [group] where fk_broker = ( " & _
        "select pk_broker from broker b, [user] u " & _
       "where u.username ='" & strSessionuser & "' and u.pk_user = b.fk_usertable)
"
    Dim intTotalCount as integer = clsdb.ExecScalar(strQuery, strErrorMsg)
    "2.INSERT THE PKGROUP AND GROUPNAME INTO A TABLE
    if intTotalCount > 0 then
        FillGroupDataSet(strSessionuser)
        dg_outer.datasource = ds.Tables("GroupID")
        dg_outer.dataBind
    end if
end sub

sub dgIndividual_ItemDataBound(a as object, e as DataGridItemEventArgs)
    if e.item.itemType = listitemtype.item or e.item.itemtype =
```

```
            listitemtype.alternatingitem then
        sum1 += Double.parse(e.item.cells(9).text)
        e.item.cells(9).text = string.Format("{0:c}",
            convert.Todouble(e.item.cells(9).text))
        sum2 += Double.parse(e.item.cells(11).text)
        e.item.cells(11).text = string.Format("{0:c}",
            convert.Todouble(e.item.cells(11).text))
    else if e.item.itemtype = ListItemType.footer then
        e.item.cells(2).text = "Total"
        e.item.cells(9).text = string.Format("{0:c}", sum1)
        e.item.cells(11).text = string.Format("{0:c}", sum2)
    end if
end sub

sub dggroup_itemDataBound(a as object, e as DataGridItemEventArgs)
    if e.item.itemType = listitemtype.item or e.item.itemtype =
            listitemtype.alternatingitem then
        sum3 += Double.parse(e.item.cells(7).text)
        e.item.cells(7).text = string.Format("{0:c}",
                convert.Todouble(e.item.cells(7).text))
        sum4 += Double.parse(e.item.cells(9).text)
        e.item.cells(9).text = string.Format("{0:c}",
                convert.Todouble(e.item.cells(9).text))
    else if e.item.itemtype = ListItemType.footer then
        e.item.cells(0).text = "Total"
        e.item.cells(7).text = string.Format("{0:c}", sum3)
        e.item.cells(9).text = string.Format("{0:c}", sum4)
    end if
end sub

sub dgouter_itemDateBound(a as object, e as DataGridItemEventargs)
    if e.item.itemtype = Listitemtype.footer then
        e.item.cells(0).text = "Total"
        e.item.cells(1).text =    " Up-To-Now Total: "
        e.item.cells(1).text &= string.Format("{0:c}", sum3) '&    &
            string.Format("{0:c}", sum4)
        e.item.cells(1).text &=    " Expect. Total: "
        e.item.cells(1).text &= string.Format("{0:c}", sum4)
    end if
end sub
</script>
```

brokerClientRecord.aspx

```
<script language=VB runat=server>
public clsdb as new clsMedasolution
public strSessionuser as string
public ds as new Dataset
public strQuery as string              'query string
sub Page_Load(a as object, e as eventargs)
    if httpcontext.current.session("username").tostring() <> "" then
        strSessionuser = session("username").ToString()
        lblTop.text = "Welcome, " & strSessionuser & " !"
    else
        response.redirect("../brokHome.aspx")
        response.end
    end if
    LoadIndividualRecord()
    LoadGroupRecord()
end sub

sub LoadIndividualRecord()
    strQuery = " select e.pk_eligibility, i.pk_insured, i.last_name,i.first_name ," & _
            "i.phone as Phone, i.email as Email, bp.benefitplan_name, " & _
            "e.eff_date, e.term_date from insured i, eligibility e, " & _
            "benefitplans bp where i.pk_insured = e.fk_insured " & _
            "and e.fk_broker = ( select pk_broker from broker b, [user] u
            "where b.fk_usertable = u.pk_user " & _
            "and u.username='" & strSessionuser & "') and e.fk_group is null " & _
            "and e.fk_benefitplans = bp.pk_benefitplans and e.most_current = 1"
    Dim blnReturnValue as boolean = clsdb.FillDataSet(strQuery, ds, " & _
        "IndividualInBroker")
    if blnReturnValue then
        dg_individual.datasource = ds.tables("IndividualInBroker").defaultView
        dg_individual.databind
        panel_individual.visible=true
    else
        lblindividual.text &= "<br><font color=red>You donot have " & _
            "individual client.</fonr>"
    end if
end sub

sub LoadGroupRecord()
    strQuery = "select g.pk_group, g.group_name, g.phone1, g.fax,
```

```
        g.primary_contact, g.primary_email          "from [group] g, broker b,
        [user] u where g.fk_broker = b.pk_broker and b.fk_usertable = u.pk_user "
        & "and u.username = '" & strSessionuser & "'"
    Dim blnreturn as boolean = clsdb.FillDataSet(strQuery, ds, "GroupInBroker")
    if blnreturn then
        dg_group.datasource = ds.tables("GroupInBroker").defaultview
        dg_group.databind
        panel_group.visible = true
    else
        lblgroup.text &= "<br><font color=red> You donot have
            GROUP client.</font>"
    end if
end sub
</script>
```

brokerInsertEmployee.aspx

```
<script language=VB runat=server>
public clsdb as new clsMedasolution
public strSessionuser as string
public ds as new Dataset
public strQuery, strErrorMsg as string          'query and errorMsg string
public myConn as new sqlConnection(clsdb.connectionString)
public myCmd as sqlCommand
sub Page_Load(a as object, e as eventargs)
    if httpcontext.current.session("username").tostring() <> "" then
        strSessionuser = session("username").ToString()
        lblTop.text = "Welcome, " & strSessionuser &" !"
    else
        response.redirect("../brokHome.aspx")
        response.end
    end if
    if Not page.IsPostback then
        loadState()
        panel_group.visible=false
        panel_insert.visible=true
    end if
end sub

sub LoadGroupRecord()
strQuery = "select g.pk_group, g.group_name, g.phone1, g.fax, g.primary_contact,
    g.primary_email " & "from [group] g where g.pk_broker="
```

```
    & viewstate("pk_broker")
Dim blnreturn as boolean = clsdb.FillDataSet(strQuery, ds, "GroupInBroker")
if blnreturn then
    dg_group.datasource = ds.tables("GroupInBroker").defaultview
    dg_group.databind
    panel_group.visible = true
else
    'lblgroup.text &= "<br><font color=red> You donot have GROUP
client.</font>"
end if
end sub

sub Save_Click(a as object, e as eventargs)
    "1.VALIDATE THE INPUT TEXTBOX
    "2.GET THE PK_BROKER
    strQuery = "select pk_broker from broker b, [user] u where b.fk_usertable =
        u.pk_user and u.username = '" & strSessionuser & "'"
    Dim intPKbroker as integer = clsdb.execScalar(strQuery, strErrorMsg)
    if strErrorMsg = "" then
        lblreminder.text &= "<br>pkbroker=" & intPKbroker
    end if
    "3.INSERT INTO GROUP TABLE
    strQuery = "sp_InsertGroupFromBroker"
    myCmd = new SqlCommand("sp_InsertGroupFromBroker", myConn)
    myCmd.CommandType = CommandType.storedProcedure
    myCmd.Parameters.add(new sqlParameter("@group_name",
        sqlDBType.nvarchar, 40))
    myCmd.Parameters("@group_name").value = txtemployername.text
    myCmd.Parameters.add(new sqlParameter("@address1",
        sqlDBType.nvarchar, 40))
    myCmd.Parameters("@address1").value = txtaddress1.text
    myCmd.Parameters.add(new sqlParameter("@address2",
        sqlDBType.nvarchar, 40))
    myCmd.Parameters("@address2").value = txtaddress2.text
    myCmd.Parameters.add(new sqlParameter("@city", sqlDBType.nvarchar, 40))
    myCmd.Parameters("@city").value = txtcity.text
    myCmd.Parameters.add(new sqlParameter("@state", sqlDBType.nvarchar, 50))
    myCmd.Parameters("@state").value = ddlstate.selectedItem.value
    myCmd.Parameters.add(new sqlParameter("@zip", sqlDBType.nvarchar, 40))
    myCmd.Parameters("@zip").value = txtzipcode.text
    myCmd.Parameters.add(new  sqlParameter("@phone1",  sqlDBType.nvarchar,
15))
```

```
    myCmd.Parameters("@phone1").value = txtphone1.text
    myCmd.Parameters.add(new  sqlParameter("@phone2",  sqlDBType.nvarchar,
15))
    myCmd.Parameters("@phone2").value = txtphone2.text
    myCmd.Parameters.add(new sqlParameter("@phone1_ext",
          sqlDBType.nvarchar, 5))
    myCmd.Parameters("@phone1_ext").value = txtext1.text
    myCmd.Parameters.add(new sqlParameter("@phone2_ext",
       sqlDBType.nvarchar, 10))
    myCmd.Parameters("@phone2_ext").value = txtext2.text
    myCmd.Parameters.add(new sqlParameter("@fax", sqlDBType.nvarchar, 15))
    myCmd.Parameters("@fax").value = txtfax.text
    myCmd.Parameters.add(new sqlParameter("@primary_contact",
       sqlDBType.nvarchar, 40))
    myCmd.Parameters("@primary_contact").value = txtcontact1.text
    myCmd.Parameters.add(new sqlParameter("@secondary_contact",
       sqlDBType.nvarchar, 40))
    myCmd.Parameters("@secondary_contact").value = txtcontact2.text
    myCmd.Parameters.add(new sqlParameter("@secondary_email",
       sqlDBType.nvarchar, 40))
    myCmd.Parameters("@secondary_email").value = txtcontact2email.text
    myCmd.Parameters.add(new sqlParameter("@primary_email",
       sqlDBType.nvarchar, 40))
    myCmd.Parameters("@primary_email").value = txtcontact1email.text
    myCmd.Parameters.add(new sqlParameter("@date_modified",
       sqlDBType.nvarchar, 40))
    myCmd.Parameters("@date_modified").value = DateTime.Now
    myCmd.Parameters.add(new sqlParameter("@date_created",
       sqlDBType.nvarchar, 40))
    myCmd.Parameters("@date_created").value = DateTime.Now
    myCmd.Parameters.add(new sqlParameter("@fk_broker", sqlDBType.int, 4))
    myCmd.Parameters("@fk_broker").value = intPKbroker
    myCmd.Parameters.add(new sqlParameter("@taxid", sqlDBType.nvarchar, 40))
    myCmd.Parameters("@taxid").value = txttaxid.text
    myCmd.Parameters.add(new sqlParameter("@eff_date",
       sqlDBType.smalldatetime, 4))
    myCmd.Parameters("@eff_date").value = txteffdate.text
    myCmd.Parameters.add(new sqlParameter("@term_date",
       sqlDBType.smalldatetime, 4))
    myCmd.Parameters("@term_date").value = txteffdate.text
    Dim intReturnValue as integer
    try
```

```
public clsdb as new clsMedasolution
public ds as new dataset

sub Page_load(a as object, e as EventArgs)
    strSessionuser = httpContext.current.session("username").Tostring()
    lbltop.text = "Welcome, " & strSessionuser
    if not Page.IsPostBack then
        if session("classID") <> 4  or (session("username") is nothing) then
            response.redirect("/dreamweaver/providhome.aspx")
        end if
        loadDiagnosiscode()
    end if
end sub

sub LoadDiagnosiscode()
    with ddldiagnosisname
        .Datasource = clsdb.getDiagnosiscode()
        .datatextfield = "diagnosiscode_name"
        .datavaluefield = "pk_diagnosiscode"
        .Databind()
        .items.insert(0, " -----select a diagnosis name-----")
        .selectedindex = 0
    end with
end sub

sub SearchByPatient_click(a as object, e as eventargs)
    strQuery = "select i.last_name+ ','+i.first_name as patient_name," & _
            "p.last_name + ',' + p.first_name as provider_name," & _
            "d.diagnosis_date, dc.diagnosiscode_name " & _
            "from insured i, provider p,diagnosiscode dc, diagnosis d " & _
            "where i.pk_insured = d.fk_insured and p.pk_provider = d.fk_provider
" &             "and dc.pk_diagnosiscode = d.fk_diagnosiscode and i.last_name =
'"
            & txtsearch.text & "'"
    Dim blnResult as boolean = clsdb.FillDataSet(strQuery, ds, "Diagnosis")

    if blnResult then
        With dg_searchByPatient
            .Datasource = ds
            .DataBind()
        end with
        if panel_searchByProvider.visible= true then
```

```
            panel_searchByProvider.visible= false
        end if
        panel_searchByPatient.visible = true
    else
        'dg_diagnosis.visible=false
    end if
end sub

sub SearchByprovider_Click(a as object, e as eventargs)
    Databind()
end sub

sub Databind()
    strQuery = "select i.last_name+ ','+i.first_name as patient_name, " & _
               "p.last_name + ',' + p.first_name as provider_name," & _
               "d.diagnosis_date, dc.diagnosiscode_name from insured i, " & _
               "provider p,diagnosiscode dc, diagnosis d " & _
               "where i.pk_insured = d.fk_insured and p.pk_provider =
d.fk_provider
               " & "and dc.pk_diagnosiscode = d.fk_diagnosiscode " & _
               "and d.fk_provider = (select pk_provider from provider p,[user] u "
&
               "where p.fk_usertable = u.pk_user and u.username = '" &
               strsessionuser & "')"
    txtsearch.text = ""
    Dim blnValue as boolean = clsdb.FillDataset(strQuery, ds, "Diagnosis")
    if blnValue then
        with dg_searchByProvider
            .Datasource = ds
            .DataBind()
        end with
        if panel_searchByPatient.visible = true then
            panel_searchByPatient.visible = false
        end if
        panel_searchByprovider.visible = true
    end if
end sub

sub Insert_Click(a as object, e as eventargs)
    panel_inserttable.visible= true
end sub
```

```
sub Save_Click(a as object, e as eventargs)
    'TEST THE INPUT NAME IS IN THE INSURED TABLE
    strQuery = "select pk_insured from insured where last_name='" &
txtlastname.text
        & "' and first_name='" & txtfirstname.text & "'"
    Dim intPKInsured as integer= clsdb.execScalar(strQuery, strErrorMsg)
    if intPKInsured <> 0 then
        'lblfirst.text &= "<BR>pkinsured = ." & intpkInsured
    else
        lblreminder.text &= "<BR><font color=red>Error: The patient name
            is incorrect.</font>"
        exit sub
    end if

    'GET THE PROVIDER'S NAME FOR THE INSERTT INTO DATABASE
    strQuery = "select p.pk_provider from provider p, [user] u " & _
            "where p.fk_usertable = u.pk_user and u.username='" & strsessionuser
            & "'"
    Dim intPKProvider as integer = clsdb.execScalar(strQuery, strErrorMsg)
    if intPKProvider <> 0 then
        'lblfirst.text &= "<BR>pkprovider= " & intpkprovider
    else
        lblreminder.text &= "<br><font color=red>Error: The provider name
            is incorrect.</font>"
    end if

    'CHECK TXTDIAGNOSISDATE IS DATA FORMATE OR NOT
    if txtdiagnosisdate.text = "" and IsDate(txtdiagnosisdate.text) = false then
        lblreminder.text &= "<BR><font color=red>Error: The diagnosis is blank
or
            isnot proper format.</font>"
        exit sub
    else
        Dim dt as datetime = Convert.Todatetime(txtdiagnosisdate.text)
        if datetime.op_greaterthan(dt, datetime.now) then
            lblreminder.text &= "<br>diagnosis date should be less than or
                equal today."
            exit sub
        end if
    end if

    'INSERT INFORMATION TO DIAGNOSIS TABLE
```

```
    strQuery ="insert into diagnosis(fk_insured, fk_provider, fk_diagnosiscode,
            diagnosis_date, date_created, date_modified)" & _
            " values(" & intPKInsured & "," & intPKProvider & "," &
            ddldiagnosisname.selectedvalue & ",'" &
            Convert.Todatetime(txtdiagnosisdate.text) & "','" & datetime.Now &
            "','" & datetime.now & "' )"
    Dim intInsertValue as integer = clsdb.execSQL(strQuery, false, strErrorMsg)
        if strErrorMsg = "" and intInsertValue <> 0 then
        lblreminder.text &= "<br><font color=red size=3>A new record
            is inserted.</font>"
        panel_inserttable.visible= false
        clearTextBox()
    else
        lblreminder.text &= "Error:failed to insert into DIAGNOSIS TABLE.
            <BR>" & strErrorMsg
    end if

    databind()
end sub

sub Close_Click(a as object, e as eventargs)
    panel_inserttable.visible= false
end sub

sub Reset_click(a as object, e as eventargs)
    clearTextBox()
end sub

sub ClearTextBox()
    txtlastname.text = ""
    txtfirstname.text = ""
    txtdiagnosisdate.text = ""
end sub
</script>
```

drgCode.aspx

```
<script language="VB" runat="server" >
public strSessionuser as string
public strQuery as string
public strErrormsg as string
Dim blnReturn as boolean
```

```
public ds as new dataset
public clsdb as new clsMedasolution
sub Page_load(a as object, e as eventargs)
    strSessionuser = httpcontext.current.session("username").Tostring()
    lbltop.text = "Welcome, " & strSessionuser & " !"
    if Not Page.IsPostBack then
        'databind()
    end if
end sub

sub DataBind()
    strQuery = "Select pk_drgcodes, drg_code, drg_name, drg_wgt, drg_year
            from drgcode"
    blnReturn = clsdb.Filldataset(strQuery, ds, "drgcode")
    if blnReturn then
        dg_drgcode.datasource = ds
        dg_drgcode.databind()
    else
        lblfirst.text &="<br>Bad. <BR> " & strErrorMsg
        exit sub
    end if
end sub

sub viewall_click(a as object, e as eventargs)
    txtsearch.text = ""
    databind()
end sub

sub search_click(a as object, e as eventargs)
    strQuery = "Select pk_drgcodes, drg_code, drg_name, drg_wgt, drg_year
        from drgcode where drg_name like '%" & txtsearch.text & "%'"
    blnReturn = clsdb.FillDataSet(strQuery, ds, "drgcode")
    if not blnreturn then
        lblreminder.text &="<font color=red size=3pt>No drg is found, please
            enter another key word.</font>"
        panel1.visible = false
        exit sub
    else
        dg_drgcode.currentPageIndex = 0
        dg_drgcode.datasource = ds
        dg_drgcode.databind
    end if
```

```
end sub

sub PageIndex_change(a as object, e as datagridPageChangedEventArgs)
    dg_drgcode.currentpageIndex = e.newpageindex
    DataBind()
end sub
</script>
```

APPENDIX C

ACRONYMS AND ABBREVIATIONS

Terminology	Definition
CSUSB	California State University, San Bernardino
ASP.Net	ASP.NET is a technology for creating dynamic Web Applications developed by Microsoft. It is part of the .NET Framework, and it can be used to author ASP.NET applications in any .NET compatible language, including Visual Basic .NET, C#, and J.
ADO.Net	The latest data access technology developed by Microsoft, and as an integral part of the .Net Framework.
Flash MX	A tool of web design and animation tool developed by Macromedia Company.
Dreamweaver MX	A tool of web design developed by Macromedia Company
IIS	Information Internet Services, ASP.Net server engine developed by Microsoft.
HTTP	Hypertext Transfer Protocol is the protocol used for delivering web pages on the Internet.
World Wide Web	World Wide Web—The subset of the Internet that utilizes the Hypertext Transfer Protocol(HTTP)
HTML	Hyper Text Markup Language
IEEE	Institute of Electrical and Electronics Engineering
HHS	The U.S. Department of Health and Human Services
HIPPA	Health Insurance Portability and Accountability Act of 1996
NIH	National Insurance Health
HMO	Health Maintenance Organization
PPO	Preferred Provider Organization
Drg code	Diagnosis-related Group. A classification system that groups patients according to diagnosis, type of treatment, age and other critera.

REFERENCES

[1] Beginning ASP.Net Databases Using VB.Net John Kauffman, Fabio Claudio, et al., Wrox 2003

[2] Using Microsoft Visual Basic.Net Brian Siler and Jeff Spotts, QUE 2002

[3] Visual Basic.Net Bible Bill Evjan, Jason Beres, et al., Hungry Minds 2002

[4] UML distilled: a brief guide to the standard object modeling language, second edition Martin Fowler, Addison Welsey Longman Inc. 1999

[5] http://healthnet.com/portal/member/home.do Healthnet insurance company home page

[6] http://www.cms.hhs.gov/default.asp Center for medicare and medicaid services

[7] http://bannerhealth.com/channels/ Banner Health company home page