Fall 2014

Administrative System for a Speech Pathology Office

Devin Raymond Peterman
Governors State University

Follow this and additional works at: http://opus.govst.edu/capstones

Part of the Databases and Information Systems Commons

Recommended Citation
http://opus.govst.edu/capstones/83

For more information about the academic degree, extended learning, and certificate programs of Governors State University, go to
http://www.govst.edu/Academics/Degree_Programs_and_Certifications/

Visit the Governors State Computer Science Department
This Project Summary is brought to you for free and open access by the Student Capstone Projects at OPUS Open Portal to University Scholarship. It has been accepted for inclusion in All Capstone Projects by an authorized administrator of OPUS Open Portal to University Scholarship. For more information, please contact opus@govst.edu.
Administrative System for a Speech Pathology Office

By

Devin Raymond Peterman

B.S., Illinois State University, 2011

Thesis
Graduate Project

Submitted in Partial fulfillment of Requirements

For the Degree of Master of Science,
With a Major in Computer Science

Governors State University
University Park, IL 60484

Fall 2014
Abstract

The office that this application was designed for is a Speech Pathology office. Therapists are sent to the houses of patients to treat them. After each visit, the speech pathologists record the data collected from the treatment. This data is then collected and type manually into an excel sheet by the office administrator.

The way the office is currently run is inefficient and outdated. Because all the information from the visits has to be typed manually into the excel sheets there is a back log of information that has to be entered. This leads to disorganization and possible loss of important data. The office administrator is also a therapist themselves and because of all of this extra data entry work they are often unable to focus solely on their therapist tasks and visits. Another problem with the current system is it’s initially entirely non-electronic. All of the documents besides with the exception of a few are filled out on paper before they are entered into the computer. This approach is very wasteful. This is expensive as well because the company has to purchase the paper and print the documents used to record the visit data.

The solution to the inefficiency and ineffectiveness of the old system was to create a web application that could do all the same functions of the current system in an entirely electronic approach. The application would cut out the unnecessary step of collecting the data from the therapists manually and entering the data by hand. The application would allow the therapists to enter their own information through an online web portal thus eliminating the middleman, saving resources and making the company more efficient.
Table of Contents

Abstract ................................................................................................................................................................... i
List of Figures ....................................................................................................................................................... 2
1.0 Introduction .................................................................................................................................................... 4
2.0 Discovery and Planning Process .................................................................................................................. 5
  2.1 Interview with the Office Administrator ................................................................................................. 5
3.0 Database ....................................................................................................................................................... 7
  3.1 Database Design and Schema ..................................................................................................................... 7
  3.2 Database Design Outline ........................................................................................................................... 7
  3.3 Database Structure .................................................................................................................................... 13
4.0 Application Architecture/ Website Structure ............................................................................................. 15
  4.1 Application Architecture ........................................................................................................................... 15
  4.2 Web Site Structure .................................................................................................................................... 16
5.0 The Web Application: General Access Pages ............................................................................................ 17
  5.1 Login Page ............................................................................................................................................... 17
  5.2 Master Page ............................................................................................................................................. 18
  5.3 Homepage .............................................................................................................................................. 18
6.0 The Web Application: Administrator Pages ............................................................................................... 19
  6.1 View Visits ............................................................................................................................................. 19
  6.2 Edit Visits ............................................................................................................................................... 24
  6.3 Create New Account ................................................................................................................................. 25
  6.4 Patients ................................................................................................................................................... 27
  6.5 Add New Patient .................................................................................................................................... 30
  6.6 View Time Sheets .................................................................................................................................. 32
7.0 The Web Application: Contractor Pages ..................................................................................................... 37
  7.1 My Visits Page ....................................................................................................................................... 37
  7.2 Add New Visit Form ................................................................................................................................. 39
  7.3 My Time Sheet ....................................................................................................................................... 42
  7.4 My Patients ............................................................................................................................................ 43
8.0 Continuing Work and Future of the Website ............................................................................................. 44
9.0 Conclusion ................................................................................................................................................... 44
References .......................................................................................................................................................... 45
List of Figures

Figure 2.1 Monthly Progress Report
Figure 2.2 Time Sheet
Figure 3.1 Database Diagram
Figure 4.1 Application Architecture Diagram
Figure 4.2 Website Structure
Figure 5.1 Login Interface
Figure 5.2 Master Page
Figure 6.1 View Visits Page
Figure 6.2 Stored Procedure spDisplayVisitsByContractorName
Figure 6.3 spDeleteVisit
Figure 6.4 Session Variables C#
Figure 6.5 getContractorId C#
Figure 6.6 spGetEmployeeIdFromVisit
Figure 6.7 Retrieving Session Variables
Figure 6.8 Step 1 of Create New User Wizard
Figure 6.9 Step 2 of Create New User Wizard
Figure 6.10 addNewUserInfo() Function
Figure 6.11 spUpdateNewUserInfo Stored Procedure
Figure 6.12 Patients Page Interface
Figure 6.13 spDisplayPatientsByContractor
Figure 6.14 Assign Patient to Contractor Interface
Figure 6.15 spAssignClient
Figure 6.16 spUnassignClient
Figure 6.17 assignPatient()
Figure 6.18 Add New Patient Form
Figure 6.19 addNewPatient() Function
Figure 6.20 View Time Sheets Interface
Figure 6.21 spGetTimeSheets
Figure 6.21 spTimeSheetTotalHours
Figure 6.22 ExportGridToExcel()
Figure 6.23 Excel Export Example
Figure 7.1 My Visits Page Interface
Figure 7.2 Add New Visit Form
Figure 7.3 addNewVisit() function
1. Introduction

This report will detail the process of constructing a web application for a speech pathology office. Purpose of the application is to streamline the company’s business operations with a more efficient paper free system. The following sections will explain the inner workings of the different components that make up the application. These components include the database that holds the site data, the user interface, and the code behind the various modules in the application. The next section will describe the interview conducted with the office administrator to get a better illustration to why this application is needed.
2. Discovery and Planning Process

2.1 Interview with the Office Administrator

The first step in starting the task of building the web application was to learn as much as possible about the company and its operations. This was done via an interview with the office administrator. The administrator was asked questions about the company that may be relevant to the construction of the web application. Some of these questions included how many people worked for the company, how many patients were treated and details of how the office is managed.

The way their operation works is the company contracts out patients to speech and developmental therapists who then in turn visit and treat the patients in their homes. Because the company finds the patients and the therapists go on the visits it is mutually beneficially for both parties involved. Since the therapists are contracted to go on these visits, they are not actual employees of the company.

The therapists collect data from each visit on a document called a monthly progress report seen in figure 2.1. Some of the data includes patient name, session details, and recommendations for the patient. Each month the contracted therapists submit the document to the office administrator who then enters all the data into an Excel spreadsheet. The following figure shows the monthly progress report. The therapists are required to fill in this form for each visit. As can be seen from figure, the therapist writes in the date of the visit, the time the visit started (In), the time the visit ended (Out), their signature, tx (signifying it was a therapy visit), 6 month (tells if the visit is a six month evaluation or assessment), annual (specifies whether the visit is an annual assessment), and the blank space in the middle which is used for additional information such as recommendations for the patients and details about the session.

![Figure 2.1 Monthly Progress Report](image)

Once the monthly progress report is submitted and entered into the spreadsheet, the information is used to construct a timesheet that is used to add up visitation hours and pay the contractors. An example of the timesheet is shown in figure 2.2. The timesheet contains the several fields derived from the monthly progress reports. Therapy date comes from the “visit date” field, “child’s name” is the name of the patient being treated, “IFSP minutes” is a certain type of therapy that would be specified in the in the blank space in the monthly progress report, “total time” is calculated using the time in and time out fields of the progress report,
and finally “cancellation dates” is filled in if the visit was cancelled. There is also another field not shown in the figure which is “total hours.” This is the total calculated from the total time column. This field is used to pay the therapists.

<table>
<thead>
<tr>
<th>Therapy Dates</th>
<th>Childs Name</th>
<th>IFSP Minutes</th>
<th>Total Time</th>
<th>Cancellation Dates</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Figure 2.2 Time Sheet**

Another document used by the office is the client list. This is a list of all the current and former patients in excel sheet format. The document contains information about each client including name, address, phone number, contact information, date of birth and so on.

Once the background information about the company was established. The requirements for the web application were given by the office administrator. The application had to have all the elements of the monthly progress report, client list and timesheet. Therapists had to have the ability to add and delete visitation data, view their clients, and their timesheet. The administrator had to have the ability to create accounts, give roles to the therapists to allow them to only access certain parts of the site, add and assign clients, view visit data and create timesheets.

Because the site deals with private health records, patient confidentiality was a factor. It was required that therapists could only view their own clients and not the entire list. For this same reason, the site itself had to also be secure. Site visitors would need to use a login screen to access the content and an SSL (secure socket layer) certificate would need to be purchased to encrypt the site data. Before the interview ended, it was agreed upon to meet once a month to discuss the progress of the application.
3. Database

3.1 Database Design and Schema

The next step in constructing the site was to create a database that could hold all the information needed for the company. The database was designed using SQL Server 2012. The first step to designing the database schema was to list all the data elements from the monthly progress report, timesheet, client list, as well as general information gathered in the discovery interview. Many of these forms had repeating information such as patient name and visit date. Once all the data fields were laid out, the next step was to place them in corresponding tables of related data. The next section describes in detail each table and how they relate to each other and the schema as a whole.

3.2 Database Design Outline

The outline describes the tables and data fields included in the tables and data types for each field. Tables with the prefix “aspnet” were automatically generated using the membership framework tool. The generated tables and fields that were not important to this application were omitted from the outline.

I.        Clients Table

The clients table holds all the data about the clients or in other words the patients. This table was derived primarily from the client list excel sheet gathered in the discovery interview with the exception of a few modifications. The ClientID field was added to give the table an automatically generated unique primary key in order to tell patients apart and relate them to other tables. Also some fields were broken up into smaller subsets. Name was divided into first name (FName) and last name (LName), and address was made into Street, City, State, and Zip.

A.        ClientID
   i.        SQL Data Type: Integer
   ii.       Primary Key to uniquely define rows of clients

B.        FName
   i.        SQL Data Type: varchar
   ii.       First names of clients

C.        LName
   i.        SQL Data Type: varchar
   ii.       Last names of clients

D.        EI#
   i.        SQL Data Type: Integer
   ii.       The ID the company uses to define clients

E.        DateOfBirth
   i.        SQL Data Type: date
ii. The date of birth of the client

F. **StartDate:**
   i. SQL Data Type: date
   ii. Date the client started treatment

G. **EndDate:**
   i. SQL Data Type: date
   ii. Date the client ended treatment

H. **Street:**
   i. SQL Data Type: varchar
   ii. The street address of the client’s residence

I. **City:**
   i. SQL Data Type: varchar
   ii. The city of the client’s residence

J. **Zip:**
   i. SQL Data Type: varchar
   ii. The zip code of the client’s residence

K. **State:**
   i. SQL Data Type: varchar
   ii. The state of the client’s residence

L. **Phone:**
   i. SQL Data Type: varchar
   ii. The phone number of the client

M. **ContactLName:**
   i. SQL Data Type: varchar
   ii. The last name of the client’s contact

N. **ContactFirstName:**
   i. SQL Data Type: varchar
   ii. The first name of the client’s contact

O. **ContactRelationship:**
   i. SQL Data Type: varchar
   ii. The relationship of the contact to the client

P. **CDC:**
   i. SQL Data Type: varchar
   ii. Number code the district the patient is from

Q. **VisitFrequency:**
   i. SQL Data Type: varchar
   ii. Represents the amount of times per week the client is visited
II. Employees Table

The Employees table holds all contractor information data. It is actually misnamed it should be called contractor table because the therapists aren’t actually employees of the company.

A. EmployeeID:
   i. SQL Data Type: integer
   ii. Primary key to uniquely identify rows of the contractors

B. UserId:
   i. SQL Data Type: uniqueidentifier
   ii. Foreign key to associate employee with username

C. LName:
   i. SQL Data Type: varchar
   ii. The last name of the contractor

D. FName:
   i. SQL Data Type: varchar
   ii. The first name of the contractor

III. Visits Table

The visits table holds all the visit data recorded by the contracted therapists. The columns of this table come mainly from the monthly progress report form as well as additional fields requested by the office administrator. These additional fields are based on what information is usually included in the monthly progress report’s blank section that can be seen in figure 2.1. Also added was the VisitID to uniquely identify each visit.

A. VisitID:
   i. SQL Data Type: integer
   ii. Primary Key to uniquely identify each visit

B. ClientID:
   i. SQL Data Type: integer
   ii. Foreign key to link table to employee table

C. Therapy_BillingID:
   i. SQL Data Type: integer;
   ii. Foreign key to Therapy_Billing table

D. EmployeeID:
   i. SQL Data Type: integer
   ii. Foreign key to relate table to Employees table

E. VisitDate:
   i. SQL Data Type: date
   ii. Date of visit

F. StartTime:
i. SQL Data Type: time
ii. Time of day visit started

G. **EndTime:**
   i. SQL Data Type: time
   ii. Time of day visit ended

H. **SessionDetails:**
   i. SQL Data Type: varchar
   ii. Details contractor records about session

I. **CancellationReason:**
   i. SQL Data Type: varchar
   ii. Reason for cancellation if there is one

J. **PhoneCalls:**
   i. SQL Data Type: varchar
   ii. Phonecalls made during visit

K. **Recommendations:**
   i. SQL Data Type: varchar
   ii. Recommendations for the patient/ client

L. **SupervisionDetails:**
   i. SQL Data Type: varchar
   ii. If visit was supervised, details about supervision

M. **IFPSMinutes:**
   i. SQL Data Type: int
   ii. Number of minutes of IFPS in visit

N. **IFSPDetails:**
   i. SQL Data Type: varchar
   ii. The details about the IFSP meeting

O. **ServiceCoordinator:**
   i. SQL Data Type: varchar
   ii. The service coordinator assigned to the patient for that meeting

IV. **Aspnet_Roles**
   This is an automatically generated table to hold the roles of users of the website.

A. **RoleId:**
   i. SQL Data Type: int
   ii. Primary Key to uniquely identify each role

B. **RoleName:**
   i. SQL Data Type: varchar
   ii. The name of the role of employee on visit
V. Aspnet_Users

This table was also generated using the SQL Membership tool. It holds the usernames of the website users.

A. UserID:
   i. SQL Data Type: uniqueidentifier
   ii. The primary key of the users table to uniquely identify each user.

B. UserName:
   i. SQL Data Type: varchar
   ii. The usernames of the website members

VI. Aspnet_UsersInRoles

This is the final auto-generated table used in this project. This table creates a relationship between the aspnet_Users and aspnet_Roles tables.

A. UserID:
   i. SQL Data Type: uniqueidentifier
   ii. Foreign key to associate UserId with RoleId

B. RoleID:
   i. SQL Data Type: uniqueidentifier
   ii. Foreign key to associate RoleID with UserID.

VII. TherapyType

This table holds the different types of therapy services that company provides.

A. TherapyID:
   i. SQL Data Type: int
   ii. Primary Key to uniquely identify each row in TherapyType table

B. TherapyDescription:
   i. SQL Data Type: varchar
   ii. The different types of therapy services

VIII. TherapyTypeModifier

This table holds information about type of therapy visit.

A. ModifierID:
   i. SQL Data Type: int
   ii. Primary Key to uniquely identify each row in table

B. ModifierDesc:
   i. SQL Data Type: varchar;
   ii. Modifier that further describes the type of therapy visit
IX. **BillingAlphaCode**

This table stores the alpha codes that further describe type of billing.

A. **AlphaCodeID**:
   i. SQL Data Type: int
   ii. Primary key to uniquely identify alpha codes

B. **AlphaCode**:
   i. SQL Data Type: varchar
   ii. The different alpha codes used for billing

X. **Therapy_Billing**

This table links all the different tables with information about the therapy and billing. It contains the foreign keys of each table associated with the type of therapy visit and the billing.

A. **TherapyBillingID**:
   i. SQL Data Type: int
   ii. Primary Key, uniquely identifies each row in Therapy_Billing table

B. **TherapyID**:
   i. SQL Data Type: int
   ii. Foreign key from TherapyType table

C. **ModifierID**:
   i. SQL Data Type: int
   ii. Foreign key from TherapyModifier table

D. **BillingNumber**:
   i. SQL Data Type: varchar
   ii. Number to identify type of billing for visit

E. **AlphaCodeID**:
   i. SQL Data Type: int
   ii. Foreign key from BillingAlphaCodes table
3.3 Database Structure

This section will briefly describe the layout of the database and how the tables relate to each other and why it is structured this way.

Figure 3.1 Database Diagram

The primary table of the database is the visits table. As mentioned in the database outline, the visits table hold all the data that has to do with each visit. The fields for this table were taken from the monthly progress report and the requests of the office administrator. The visit table has four keys. VisitID is the primary key of the table to identify each visit. The foreign keys are ClientID from the Client table, EmployeeID from the Employees table, and Therapy_BillingID from the Therapy_Billing table. Each Key is to link the each visit with another table. The ClientID identifies which patient is being visited. EmployeeID identifies which therapist is visiting, and Therapy_BillingID identifies the type of visit and the billing associated with it. Which leads to the next table.

The next table to be discussed is the Therapy_Billing table. This table is a composite of three different tables linked with foreign keys. There are a finite amount entries in the tables associated with the Therapy_Billing table. So unlike a table like visits where new entries are
constantly being added, these tables will remain relatively static. The first of the tables related to the Therapy_Billing table is TherapyType. Therapy type is self-explanatory. It’s the type of therapy being administered. Some of the entries in this table include speech, developmental, translation, etc.

The next table is the TherapyTypeModifier table. This table defines the type of visit and adds more detail about the visit rather than just the type of therapy. The entries in this table include 6 month assessment, initial evaluation, or just plain therapy, among others.

The next table is BillingAlphaCodes. The alpha codes are suffixes that go on the end of the billing number for certain types of visits when the patient is billed. They need to be included for the sake of the third party billing company. The Therapy_Billing table also has a column called BillingNumber. This is also used by the billing company to bill patients. The two pieces of information needed to get the proper billing number and alpha code is the therapy type and therapy type modifier (or visit type). So for example, the primary key for “speech” in the therapy table is 1 and the primary key for the visit type “therapy” is 2. Both of these are foreign keys in the Therapy_Billing table to and give the proper billing number and foreign key for the proper alpha code. So all together they make up the Therapy_BillingID which is entered in the visits table as a foreign key to associate a certain with the therapy type, visit type, and proper billing information.

Another main relation of the database is the Employees (contractors) and Clients tables. Certain contractors have certain clients that are assigned to them and they can only view these assigned clients in the front end of the application. This relation is accomplished by utilizing the Employees_Clients table. The Employees_Clients table is consisted of two foreign keys the EmployeeID and the ClientID. If the administrator wants to associate an employee with a certain client they will use the application to add a row with the EmployeeID and ClientID of the desired contractor and patient.

The final part of the database was auto-generated with the asp membership framework tool. The tables include aspnet_Roles which hold the different roles in the site, aspnet_Users which holds the user data, and aspnet_UsersInRoles which associates which user is in which role. The Roles the site utilizes are “Administrator” and “Contractor.” These roles are used to allow and restrict certain users to certain areas of the site. These tables are linked to the other areas of the database by including the UserID column as a foreign key in the employees table. This in turn gives the username to each employee and what role they are in.
4. Application Architecture/ Website Structure

4.1 Application Architecture

The following diagram illustrates the application architecture or the web application. It shows the layers of the site from the user to the database. The website was constructed using HTML and CSS for the UI, C# the code behind files, and SQL Server 2012 Management Studio for the database. The development environment used was Visual Studio 2012. The site was uploaded to the internet using the web hosting site GoDaddy.com.

![Figure 4.1 Application Architecture Diagram]
4.2 Web Site Structure:

Below describes the basic set up of the website and how it will be navigated. Since the Administrator has access to the contractors’ pages as well as Administrator pages the chart is divided by the roles that are authorized to access each page.

![Website Structure Diagram]

Figure 4.2 Website Structure
5.0 The Web Application: General Access Pages

The following sections will detail the web application itself describing in detail the web server controls used on each page, any notable user interface elements and code behind when applicable. This first section will describe the general access pages that can be accessed by either administrator or contractor.

5.1 Login Page

Before any user can access the website they must first go through the login screen. Usernames and passwords are assigned to them from the administrator. Since this is an application for a private company, outside users will not be able to register. The Administrator will be able to add users and give them either an administrator role or an employee role in the Create New User page. Depending on the user’s role they will only be able to access certain areas of the website. There are two roles in the website, Administrator and Contractor. The Contractors are only allowed on the home page and the menu items under the “Contractors” drop down navigation bar. Administrators are allowed anywhere on the site. The login interface was constructed using the Login Control from the Asp.net web controls library. From here the user will be directed to the main page of the website where they can navigate using the navigation bar on the master page.

![Login Interface](image_url)

Figure 5.1 Login Interface
5.2 Master Page

The Site’s master page has some features of note, primarily the navigation bar. The navigation bar has three items. “Home,” “Administrator Access,” and “Contractors.” While home brings you directly to the homepage, if the user hovers over Administrator Access or Contractors a drop down navigation will appear. This is how to navigate to the rest of the site. If the user wants to log out there is a logout link in the upper right hand side of the site. This was constructed using the LoginStatus server control. The website also recognizes the user logged in with the asp LoginName web server control. The logo can also be pressed to return to the homepage.

![Figure 5.2 Master Page](image.png)

5.3 Homepage

The home page of the site is the start page the user is sent to upon log in. Currently displays basic information about how to use the site, however there are future plans to utilize this page differently. Though not added yet, there will eventually be alert messages that show, based on the contractor, whether they have a certain type of visit due. For instance, every six months therapists are required to do a six month assessment. This will be calculated from the last six month assessment and display an alert for the user on the homepage.
6.0 The Web Application: Administrator Only Pages

This next sections will detail the page that can only be accessed by users in the administrator role.

6.1 View Visits

The first of the Administrator only pages is the view visits page. The purpose of this page is to let the administrator view all of the visits being recorded. The administrator will also have the power to edit and delete visits. These functions are made possible through a series of SQL stored procedures that are called in the C# code behind file that the user accesses through a series of web server controls.

As can be seen in figure 6.1, this page contains two web controls that allow the admin to view all the visit data in the database. At the top of the screen is a drop down list used to filter the results in the grid view below. The results are filtered based on the contractor’s name in the drop down list. This is accomplished by calling the stored procedure spDisplayVisitsByContractorName shown in figure 6.2 below.

![Figure 6.1 View Visits Page](image-url)
ALTER proc [dbo].[spDisplayVisitsByContractorName]
@EmployeeId int
as
If @EmployeeId='0'
Begin
select VisitId, clients.lname + ', ' + clients.fname as 'Patient Name',
CONVERT(VARCHAR(10),visitdate,111) as 'Visit Date', StartTime as 'Start Time',
endtime as 'End Time', TherapyDescription as 'Discipline', ModifierDesc as 'Visit Type',
SessionDetails,
CancellationReason as 'Cancellation Reason', PhoneCalls as 'Phone Calls',
Recommendations, SupervisionDetails as 'Supervision Details', IFSPMinutes as 'IFSP Minutes', IFSPDetails as 'IFSP Details'
from clients
join visits
on clients.ClientID = visits.ClientID
join employees
on employees.EmployeeID = visits.Employeeid
join Therapy_Billing
on therapy_billing.therapy_billingid = visits.therapy_Billingid
join therapyType
on therapytype.therapyid = therapy_Billing.therapyId
join therapytypemodifier
on therapytypemodifier.modifierid = therapy_billing.modifierID
order by VisitDate desc;
End
select VisitId, clients.lname + ', ' + clients.fname as 'Patient Name',
CONVERT(VARCHAR(10),visitdate,111) as 'Visit Date', StartTime as 'Start Time',
endtime as 'End Time', TherapyDescription as 'Discipline', ModifierDesc as 'Visit Type',
SessionDetails,
CancellationReason as 'Cancellation Reason', PhoneCalls as 'Phone Calls',
Recommendations, SupervisionDetails as 'Supervision Details', IFSPMinutes as 'IFSP Minutes', IFSPDetails as 'IFSP Details'
from clients
join visits
on clients.ClientID = visits.ClientID
join employees
on employees.EmployeeID = visits.Employeeid
join Therapy_Billing
on therapy_billing.therapy_billingid = visits.therapy_Billingid
join therapyType
on therapytype.therapyid = therapy_Billing.therapyId
join therapytypemodifier
on therapytypemodifier.modifierid = therapy_billing.modifierID
where employees.employeeid = @EmployeeId
order by VisitDate desc;

Figure 6.2 Stored Procedure spDisplayVisitsByContractorName

The user selects a contractor name from the contractor filter drop down list. The list contains all the names of the contractors and an item called “All Contractors.” The drop down list shows the names of the contractors but uses their unique employeeid number as the value. The stored procedure spDisplayVisitsByContractorName has one parameter, EmployeeID. The user passes this parameter into the stored procedure by selecting the contractor they desire from the drop down list. If the user selects all contractors the stored procedure is passed a zero
and the IF condition is fulfilled and the grid shows all the visits for all the contractors. If the user selects any of the contractors’ names, the stored procedure is passed the ID of that contractor and the grid view shows all the visits belonging to that therapist.

Another function of the View Visits page is deleting visits which is done with the stored procedure spDeleteVisit. This is activated by the user clicking delete in the grid view. This procedure is shown in figure 6.3.

```sql
ALTER proc [dbo].[spDeleteVisit]
@VisitId int
as
Delete From Visits
where VisitId = @VisitId
```

**Figure 6.3 spDeleteVisit**
The final function of this page is the Edit button which allows the administrator to edit visit data for a particular entry. When the user clicks edit, the information from the grid view is taken and stored into session variables so they can be transferred from one page to another. This can be seen in C# code in Figure 6.3. The contractor id is also set by calling the function getContractorId. This function connects to the database and calls the stored procedure spGetEmployeeIDFromVisit and uses visitid as an argument. This stored procedure is shown in figure 6.4. The function getContractorId uses a SqlDataReader object to read the results of the stored procedure.

```csharp
protected void GridView1_SelectedIndexChanged(object sender, EventArgs e)
{
    string visitId = GridView1.SelectedRow.Cells[1].Text;
    string contractorId = getContractorId(visitId);
    string patientName = GridView1.SelectedRow.Cells[2].Text;
    string visitDate = GridView1.SelectedRow.Cells[3].Text;
    string startTime = GridView1.SelectedRow.Cells[4].Text;
    string endTime = GridView1.SelectedRow.Cells[5].Text;
    string discipline = GridView1.SelectedRow.Cells[6].Text;
    string visitType = GridView1.SelectedRow.Cells[7].Text;
    string sessionDetails = GridView1.SelectedRow.Cells[8].Text;
    string cancellationReason = GridView1.SelectedRow.Cells[9].Text;
    string phoneCalls = GridView1.SelectedRow.Cells[10].Text;
    string supervisionDetails = GridView1.SelectedRow.Cells[12].Text;
    string ifspMinutes = GridView1.SelectedRow.Cells[13].Text;
    string ifspDetails = GridView1.SelectedRow.Cells[14].Text;

    Session["VisitId"] = visitId;
    Session["ContractorId"] = contractorId;
    Session["PatientName"] = patientName;
    Session["VisitDate"] = visitDate;
    Session["StartTime"] = startTime;
    Session["EndTime"] = endTime;
    Session["Discipline"] = discipline;
    Session["VisitType"] = visitType;
    Session["SessionDetails"] = sessionDetails;
    Session["CancellationReason"] = cancellationReason;
    Session["PhoneCalls"] = phoneCalls;
    Session["Recommendations"] = recommendations;
    Session["SupervisionDetails"] = supervisionDetails;
    Session["IFSPMinutes"] = ifspMinutes;
    Session["IFSPDetails"] = ifspDetails;

    Response.Redirect("~/AdministratorPages/EditVisitAdmin.aspx");
}
```

**Figure 6.4 Session Variables C#**
```csharp
protected string getContractorId(string visitid)
{
    string connstr =
        ConfigurationManager.ConnectionStrings["SLFTherapyConnectionString"].ConnectionString;
    SqlConnection connection = new SqlConnection(connstr);
    connection.Open();

    SqlCommand command = new SqlCommand("spGetEmployeeIdFromVisit", connection);
    command.CommandType = CommandType.StoredProcedure;
    command.Parameters.AddWithValue("@visitid", visitid);

    SqlDataReader rdr = null;
    rdr = command.ExecuteReader();
    int conid = 0;
    while (rdr.Read())
    {
        // get the results of each column
        conid = (int)rdr["EmployeeId"];
        conid.ToString();
    }
    //Close Reader
    if (rdr != null)
    {
        rdr.Close();
    }

    return conid.ToString();
}
```

**Figure 6.5 getContractorId C#**

```sql
ALTER proc [dbo].[spGetEmployeeIdFromVisit]
@visitid int
as
select Employeeid from visits
where visitid = @visitid
```

**Figure 6.6 spGetEmployeeIdFromVisit**
6.2 Edit Visits

Once the Edit button is clicked in the grid of the View Visits page the user is redirected to the Edit Visits page. This page is identical to the add new visit that will be discussed later except for the fact that it contains code to retrieve the values from the GridView in the view visits page and display them on the controls of the Edit Visit page to allow for easy editing. Figure 6.7 shows a code example of how the session variable is retrieved and the corresponding control is set.

```csharp
string sessionDetails = Session["SessionDetails"].ToString();
txtSessionDetails.Text = sessionDetails;

string cancellationReason = Session["CancellationReason"].ToString();
txtCancellationReason.Text = cancellationReason;

string phoneCalls = Session["PhoneCalls"].ToString();
txtPhoneCalls.Text = phoneCalls;

string recommendations = Session["Recommendations"].ToString();
txtRecommendation.Text = recommendations;

string supervisionDetails = Session["SupervisionDetails"].ToString();
txtSupervision.Text = supervisionDetails;
```

**Figure 6.7 Retrieving Session Variables**

Once the user is satisfied with their changes they click the submit button which calls the function to update the data in the visit table for the selected record. This function is very similar to the insert function so to not be redundant it will be discussed later in the Add New Visit section.
6.3 Create New Account

The next page that the Admin only has access to is the Create New Account page. Here the Admin can create an account and assign it a role. The account creation is done via a create user wizard server control. The user is added with the membership framework so that the username and user id can interact with the rest of the database. Figures 6.8 and 6.9 show the various steps to creating a new user. The first step (Figure 6.8) is simply to create a user name and password for the user. This is the standard step for creating a new user and it’s the default for the Create User Wizard.

![Sign Up for Your New Account](image)

**Figure 6.8 Step 1 of Create New User Wizard**

![Specify Name and Role](image)

**Figure 6.9 Step 2 of Create New User Wizard**
private void addNewUserInfo()
{
    MembershipUser newUser = Membership.GetUser(NewUserWizard.UserName);
    Guid newUserGuidId = (Guid)newUser.ProviderUserKey;

    string connStr = ConfigurationManager.ConnectionStrings["SQLTherapyConnectionString"].ConnectionString;

    SqlConnection connection = new SqlConnection(connStr);
    connection.Open();
    SqlCommand command = new SqlCommand("spUpdateNewUserInfo", connection);
    command.CommandType = CommandType.StoredProcedure;
    command.Parameters.AddWithValue("@UserId", newUserGuidId);
    command.Parameters.AddWithValue("@LName", SqlDbType.VarChar).Value = txtLastName.Text.Trim();
    command.Parameters.AddWithValue("@FName", SqlDbType.VarChar).Value = txtFirstName.Text.Trim();
    command.ExecuteNonQuery();
    connection.Close();
    Roles.AddUserToRole(newUser.ToString(), ddUserRoleSelect.SelectedValue);
}

Figure 6.10 addNewUserInfo() Function

ALTER proc [dbo].[spUpdateNewUserInfo]
@UserId uniqueidentifier, @LName varchar(50), @FName varchar(50)
as
INSERT INTO Employees(UserId, Lname, Fname)
VALUES(@UserId, @LName, @FName)

Figure 6.11 spUpdateNewUserInfo Stored Procedure

The second step (Figure 6.9) is the custom step that was added for the sake of the web application. In this step the administrator can give a first and last name to the user as well as assign a role to the user. This additional information about the user is inserted with the stored procedure spUpdateUserInfo which is called through the addNewUserInfo function. SpUpdateUserInfo and addNewUserInfo are shown in figures 6.10 and 6.11 respectively. SpUpdateUserInfo takes three parameters: @UserId, @Fname, and @LName. These pass in the values for userid first name and last name to be inserted into the employees table in the database. The addNewUserInfo() function begins by getting the user id of the user that was just made in the previous step. Next the function makes a connection to the Sql database, calls the stored procedure and passes in the three parameters. This inserts the data. Next the user is assigned a role with the addUserToRole function called at the end of the addNewUserInfo function. This concludes how the application add new users and assigns them roles.
6.4 Patients

Another Page the admin has exclusive rights to is the Patients page. Here the admin can add new patients, and assign them to employees. The patients are added through the Add New Patient form which will be discussed in the next section. Figure 6.12 shows the Patients page’s interface.

![Patients Page Interface](image)

The patients are displayed in a grid view based on the contractor drop down list. The grid view displays the patients by calling the spDisplayPatientsByContractor stored procedure shown in figure 6.13. This stored procedure is very similar to the spDisplayVisitsByContractorName shown earlier. The procedure has one parameter, @ContractorId, and it displays patients’ data based on which contractor is selected. If “All Contractors” is selected from the drop down box, all patients are displayed.

```sql
ALTER proc [dbo].[spDisplayPatientsByContractor]
```
@ContractorId nvarchar(256)
as
IF(@ContractorId = '0')
BEGIN
Select Distinct clients.Lname+', '+Clients.fname as 'Patient Name', EI#,
CONVERT(VARCHAR(10),DateofBirth,111) as 'DOB', CONVERT(VARCHAR(10),StartDate,111) as
'Auth Start Date', CONVERT(VARCHAR(10),EndDate,111) as 'Auth End Date',VisitFrequency,
Street+' '+City+', '+State+' '+Zip as 'Address', Phone, ContactLName+', '+ContactFname as
'Contact Name', ContactRelationship as 'Contact Relationship', ServiceCoordinator as
'Service Coordinator', CDC
from Clients
Order By 'Patient Name'
END
Else
BEGIN
Select Distinct clients.Lname+', '+Clients.fname as 'Patient Name', EI#,
CONVERT(VARCHAR(10),DateofBirth,111) as 'DOB', CONVERT(VARCHAR(10),StartDate,111) as
'Auth Start Date', CONVERT(VARCHAR(10),EndDate,111) as 'Auth End Date',VisitFrequency,
Street+' '+City+', '+State+' '+Zip as 'Address', Phone, ContactLName+', '+ContactFname as
'Contact Name', ContactRelationship as 'Contact Relationship', ServiceCoordinator as
'Service Coordinator', CDC
from Clients join Employees_Clients
on Clients.clientid = Employees_Clients.ClientID
join Employees
on employees.EmployeeID = Employees_Clients.Employeeid
where Employees.EmployeeId = @ContractorId
order by 'Patient Name';
END

Figure 6.13 spDisplayPatientsByContractor

One of the most important functions of the site is to assign and un-assign different
patients to different contractors. This is because of confidentiality reasons contractors aren’t
allowed to see all the patients’ names. So before they can add a visit they need to be assigned a
patient so they can choose them in the Add New Visit form. Patients are assigned and
unassigned by adding rows into the Employees_Clients table and making a relation between
them or deleting the relation.

This is done with a series of list boxes shown below in figure 6.14. Each list box is
populated based on whether a patient is assigned to a contractor or not. Each time the arrow
buttons are pressed, one of two stored procedures are called depending on the button,
spAssignClients and spUnassignPatient. Validation controls are in place to make sure a
contractor is selected form the contractor filter drop down and a patient is chosen from one of
the list boxes. To assign a patient the user must select an unassigned patient from the list box
and press the “>>” button. This initiates the spAssignClients stored procedure shown in figure
6.15. This stored procedure just stores the employeeid and the clientid in the
Employees_Clients table. The opposite is true for the procedure spUnassignClients. The row is
simply deleted from the table.
These stored procedure are called through the assignPatient() and unassignPatient() functions. Since these functions are nearly identical except for the procedures they call, an explanation of unassignPatient() will be omitted. The function assignPatient(), shown in figure 6.17, works just like when user information was entered into the employees table. The function makes a connection to the database and passes the proper parameters to the stored procedure. In this case, the selected contractor’s id and the selected patient’s id from the server controls.

---

**Figure 6.14 Assign Patient to Contractor Interface**

ALTER proc [dbo].[spAssignClients]
@Employeeid int, @clientId int
as
IF NOT EXISTS(SELECT 1 FROM Employees_clients WHERE employeeid = @employeeid and clientid = @ClientId)
insert into Employees_Clients
(Employeeid,clientId)
values
(@Employeeid,@clientId)

---

**Figure 6.15 spAssignClient**

ALTER proc[dbo].[spUnassignClients]
@employeeid int, @clientId int
as
Delete From employees_clients
where employeeid = @employeeId and clientId = @ClientId

---

**Figure 6.16 spUnassignClient**
private void assignPatient()
{
    if (ddlContractorFilter.SelectedValue != "0")
    {
        lblContractorSelect.Visible = false;
        string connStr = ConfigurationManager.ConnectionStrings["SLFTherapyConnectionString"].ConnectionString;
        SqlConnection connection = new SqlConnection(connStr);
        connection.Open();
        SqlCommand command = new SqlCommand("spAssignClients", connection);
        command.CommandType = CommandType.StoredProcedure;
        command.ExecuteNonQuery();
        lbAssignedPatients.DataBind();
        lbUnassignedPatients.DataBind();
        GridView1.DataBind();
    }
    else lblContractorSelect.Visible = true;
}

Figure 6.17 assignPatient()

6.5 Add New Patient

To add a new patient the clients table in the database the admin clicks the Add New Patient button on the Patients Screen and they are sent to the Add New Patient form. This form is complete with validation for the important fields and contains the following items pictured below in figure 6.18. Once all the desired information is filled out the user clicks the submit button to call the function to insert the information into the clients table. This function is shown in figure 6.19.

The addNewPatient() function works the same way as the other insert functions discussed so far. The all of fields that are filled out correspond to a parameter passed into the insert stored procedure. When the function is complete the user is redirected to the Patients page.
Figure 6.18 Add New Patient Form
The final section of the site that is exclusively for Administrators is the View Time Sheets page. Once again the contractors have a similar function but they are only allowed to read their own time sheets the administrator is allowed to get them for any one on the site. The Time Sheet page uses a series of controls to filter the data and get the total hours worked for a given period of time. The Admin just needs to select the pay period with a series of calendar controls and the employee name and the time sheet generates. The Admin then has the option of exporting the timesheet to an excel file. The interface for the View Time Sheets page is shown in figure 6.20.

The time sheet will display the following fields:

- Therapy date: Displays the date of the therapy visit.
- Client’s name: Displays the client who was being visited.
- Discipline: Displays the Type of therapy performed on the visit.
- IFSP Minutes: Displays the amount of IFSP minutes for each visit, if applicable.
- Total Time: displays a calculated field of the amount of hours the visit took based on Time In / Time Out from the Record Visit Data form.
- Total Hours: Will appear below the Total Time column and will give a total amount of hours for all the visits combined in the selected pay period.
Figure 6.20 View Time Sheets Interface

The View Time Sheets display is generated using the spGetTimeSheets stored procedure shown in figure 6.21. spGetTimeSheets takes three parameters: @EmployeeId, @StartDate, and @EndDate. Once again this is very similar to how the patients or the visits are displayed on the other pages. The interesting part about the spGetTimeSheets stored procedure is the “Elapsed Visit Time” column. It is calculated by taking the time differential of the start time of the visit to the end time of the visit and gets the total time for each visit from that time period.

The total time is displayed in another grid view that calls a different stored procedure, spTimeSheetTotalHours shown in figure 6.22. This procedure adds up the column of the elapsed visit time.

ALTER proc [dbo].[spGetTimeSheet]
    @EmployeeId int, @StartDate Date, @EndDate date
    as
BEGIN
    IF (@EmployeeID = '0' or @StartDate = @EndDate)
        BEGIN
            Select employees.lname+', '+employees.fname as 'Contractor', cast(VisitDate AS date) as 'Visit Date', clients.LName+ ', '+clients.FName as 'Client''s Name', TherapyDescription as 'Discipline', ModifierDesc as 'Visit Type' , IFSPMinutes as 'IFSP Minutes',
            CONVERT(varchar(6), datediff(second,starttime,endtime) /3600) + ':' + RIGHT('0' + CONVERT(varchar(2), (datediff(second,starttime,endtime) % 3600) / 60), 2) as 'Elapsed Visit Time'
            from Employees
            join Visits
            on Employees.EmployeeID = Visits.EmployeeID
            join clients
            on Clients.ClientID = Visits.ClientId
            join Therapy_Billing
        END
    END

    Select employees.lname+', '+employees.fname as 'Contractor', cast(VisitDate AS date) as 'Visit Date', clients.LName+ ', '+clients.FName as 'Client''s Name', TherapyDescription as 'Discipline', ModifierDesc as 'Visit Type' , IFSPMinutes as 'IFSP Minutes',
    CONVERT(varchar(6), datediff(second,starttime,endtime) /3600) + ':' + RIGHT('0' + CONVERT(varchar(2), (datediff(second,starttime,endtime) % 3600) / 60), 2) as 'Elapsed Visit Time'
    from Employees
    join Visits
    on Employees.EmployeeID = Visits.EmployeeID
    join clients
    on Clients.ClientID = Visits.ClientId
    join Therapy_Billing
on Therapy_Billing.Therapy_Billingid = Visits.Therapy_Billingid
join TherapyType
on Therapytype.TherapyId = Therapy_Billing.TherapyId
join TherapyTypeModifier
on TherapytypeModifier.ModifierId = Therapy_Billing.modifierId
order by VisitDate
END
ELSE
Begin
Select employees.lname+', '+employees.fname as 'Contractor', cast(VisitDate AS date) as 'Visit Date', clients.LName+ ', '+clients.FName as 'Client''s Name', TherapyDescription as 'Discipline', ModifierDesc as 'Visit Type', IFSPMinutes as 'IFSP Minutes',
CONVERT(varchar(6), datediff(second,starttime,endtime) /3600) + ':' + RIGHT('0' + CONVERT(varchar(2), (datediff(second,starttime,endtime) % 3600) / 60), 2) as 'Elapsed Visit Time'
from Employees
join Visits
on Employees.EmployeeID = Visits.EmployeeID
join clients
on Clients.ClientID = Visits.ClientId
join Therapy_Billing
on Therapy_Billing.Therapy_Billingid = Visits.Therapy_Billingid
join TherapyType
on Therapytype.TherapyId = Therapy_Billing.TherapyId
join TherapyTypeModifier
on TherapytypeModifier.ModifierId = Therapy_Billing.modifierId
Where Visits.Employeeid = @EmployeeId and VisitDate >= @StartDate and VisitDate <= @EndDate
order by VisitDate
END

Figure 6.21 spGetTimeSheets
ALTER proc [dbo].[spTimeSheetTotalHours]
@EmployeeId int, @StartDate Date, @EndDate date
as
IF(@EmployeeID = '0' or @StartDate = @EndDate)
BEGIN
    SELECT CONVERT(varchar(6), sum(datediff(second,starttime,endtime)/3600))
    + ':' + RIGHT('0' + CONVERT(varchar(2), sum((datediff(second,starttime,endtime) % 3600) / 60)), 2) as 'Total Hours This Pay Period'
    FROM Employees
    JOIN Visits
    ON Employees.EmployeeID = Visits.EmployeeID
END
ELSE
BEGIN
    SELECT CONVERT(varchar(6), sum(datediff(second,starttime,endtime)/3600))
    + ':' + RIGHT('0' + CONVERT(varchar(2), sum((datediff(second,starttime,endtime) % 3600) / 60)), 2) as 'Total Hours This Pay Period'
    FROM Employees
    JOIN Visits
    ON Employees.EmployeeID = Visits.EmployeeID
    WHERE Visits.Employeeid = @EmployeeId AND VisitDate >= @StartDate AND VisitDate <= @EndDate
END

Figure 6.21 spTimeSheetTotalHours

The final element of the Time Sheets page is the export to excel feature. Once the user presses the “Export to Excel” button, the function ExportGridToExcel() is initiated. The function is shown in figure 6.22. The function works by creating an HtmlTextWriter object and using it to read the grid views on the page. Both grid views are read and displayed in an excel sheet like the one featured in figure 6.23.

    private void ExportGridToExcel()
    {
        Response.Clear();
        Response.Buffer = true;
        Response.ClearContent();
        Response.ClearHeaders();
        Response.Charset = "";
        string strName = ddlContractorFilter.SelectedItem.Text;
        string FileName = strName.ToString() + " TimeSheet " + calStart.SelectedDate.ToShortDateString();
        StringWriter strwriter = new StringWriter();
        HtmlTextWriter htmlTextWriter = new HtmlTextWriter(strwriter);
        Response.Cache.SetCacheability(HttpCacheability.NoCache);
        Response.ContentType = "application/vnd.ms-excel";
        Response.AddHeader("Content-Disposition", "attachment;filename=" + FileName);
        GridView1TimeSheet.GridViewLines = GridLines.Both;
        GridView1TimeSheet.HeaderTextStyle.Font.Bold = true;
        GridView1TimeSheet.RenderControl(htmlTextWriter);
        GridViewTotalHours.GridViewLines = GridLines.Both;
        GridViewTotalHours.RenderControl(htmlTextWriter);
        Response.Write(strwriter.ToString());
        Response.End();
    }

Figure 6.22 ExportGridToExcel()
Figure 6.23 Excel Export Example

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
<th>G</th>
</tr>
</thead>
<tbody>
<tr>
<td>Contractor</td>
<td>Visit Date</td>
<td>Client's Name</td>
<td>Discipline</td>
<td>Visit Type</td>
<td>Time</td>
<td>Planned</td>
</tr>
<tr>
<td>chistoff, chad</td>
<td>1/1/2001 12:00:00 AM</td>
<td>Arken, Allen</td>
<td>Speech</td>
<td>Therapy</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>Smith, Ashley</td>
<td>9/22/2014 0:00:00</td>
<td>White, James</td>
<td>Speech</td>
<td>Therapy</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Smith, Ashley</td>
<td>11/29/2014 0:00:00</td>
<td>White, James</td>
<td>Speech</td>
<td>Therapy</td>
<td>0</td>
<td>4</td>
</tr>
<tr>
<td>User, New</td>
<td>12/2/2014 0:00:00</td>
<td>White, James</td>
<td>Speech</td>
<td>Therapy</td>
<td>60</td>
<td>16</td>
</tr>
<tr>
<td>Donaldson, Donny</td>
<td>12/2/2014 0:00:00</td>
<td>Newhart, Bob</td>
<td>Speech</td>
<td>Therapy</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>Smith, Ashley</td>
<td>12/2/2014 0:00:00</td>
<td>White, James</td>
<td>Developmental</td>
<td>Initial Evaluation</td>
<td>30</td>
<td>10</td>
</tr>
<tr>
<td>brennen, tom</td>
<td>12/6/2014 0:00:00</td>
<td>White, James</td>
<td>Developmental</td>
<td>Initial Evaluation</td>
<td>15</td>
<td>3</td>
</tr>
<tr>
<td>brennen, tom</td>
<td>12/15/2014 0:00:00</td>
<td>White, James</td>
<td>Speech</td>
<td>Therapy</td>
<td>0</td>
<td>4</td>
</tr>
<tr>
<td>Donaldson, Donny</td>
<td>12/16/2014 0:00:00</td>
<td>Crammer, Craig</td>
<td>Speech</td>
<td>Therapy</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>Smith, Ashley</td>
<td>12/16/2014 0:00:00</td>
<td>Johnson, Fred</td>
<td>Feeding</td>
<td>Therapy</td>
<td>30</td>
<td>3</td>
</tr>
<tr>
<td>brennen, tom</td>
<td>12/23/2014 0:00:00</td>
<td>White, James</td>
<td>Speech</td>
<td>Therapy</td>
<td>15</td>
<td>3</td>
</tr>
<tr>
<td>chistoff, chad</td>
<td>12/24/2014 0:00:00</td>
<td>123, test</td>
<td>Speech</td>
<td>Therapy</td>
<td>0</td>
<td>6</td>
</tr>
</tbody>
</table>

Total Hours This Pay Period: 55:90
7. The Web Application: Contractor Pages

The Contractor pages are the pages that can be accessed by the user in the role contractor as well as administrator but not anonymous users. These pages are where the contracted therapists can view and enter data. These pages are basically the same as the administrator versions with a few exceptions. Rather than explain them in as much detail as the administrator pages, it is more appropriate to point out the subtle differences.

7.1 My Visits Page

The My Visits Page is where users are allowed to add, edit and delete their visit information. This page is very similar to the View Visits page in the administrator section. The key difference being that there is no drop down menu to select which contractor’s visits the user wants to view. The Visit data is automatically displayed based on the user logged in. Recall the stored procedure that displayed visits in the view visits page, `spDisplayVisitsByContractorName`. This stored procedure took one parameter, contractor name and displayed the data for that contractor. The stored procedure for this page uses the same idea but instead passed that parameter a default value of the person currently logged in. The page then displays the most recent visit data entered in a grid view control. Next to each entry there is an edit button that the user can click to edit previous visits in case they forgot any information. The edit button works identical to the one in the View Visits page. The different fields that can be edited will have all the same validations as the Add New Visit form. Below, in figure 7.1, is an example of what the My Visits page interface looks like.

![My Visits Page Interface](image)

**Figure 7.1 My Visits Page Interface**
Below the visit table there is a button that says add new visit. If clicked, this button redirects the user to a form where they fill out visitation data that upon completion, is added to the visits table in the database.
7.2 Add New Visit Form

This is the page where the user fills in server controls with information to be added into the visits table. Figure 7.2 shows the Add New Visit form. The visit data form consists of the following fields:

- **Patient Name**: Used to record the patient’s name who is being visited. It is selected via dropdown menu from a group of patient names assigned from the admin to the employee from the clients table.
- **Date**: Used to record the date of the visit. This is selected from a calendar control and stored in the VisitDate column of the Visits table.
- **Time In**: This is used to record the start time of the visit. This is selected from a series of drop down lists ranging from 1 -12 for hours and 00 to 60 for minutes as well as a selector for am or pm. This is stored in the StartTime column of the Visits table.
- **Time Out**: This is used to record the end time of the visit. This is selected from a series of drop down lists ranging from 1 -12 for hours and 00 to 60 for minutes as well as a selector for am or pm. This is stored in the EndTime column of the Visits table.
- **Therapy Type**: This is used to record the type of therapy performed during the visit. It is selected via two dropdown menus from the database. The drop down menus display the therapy types from the TherapyType table and the therapy type modifier from the TherapyTypeModifier table. The IDs of these fields are then stored in the Visit_Therapy table to link the visit with the type of therapy.
- **Session details**: This is used to record the details of what happened during the therapy section. It is a text bow with a limit of 600 characters and is stored in the SessionDetails column of the visits table.
- **Cancellation Checkbox**: If there is a cancellation the user will check this box. Once the box is checked the Cancellation Reason Textbox will appear.
- **Cancellation Reason**: A text box with a 200 character limit that is only visible once the Cancellation Checkbox is checked. Records the cancellation reason in the CancellationReason field of the Visits table.
- **Supervision**: A textbox used to record whether the visit was supervised and who it was supervised by. Recorded into the Supervision field of the Visits table.
- **Recommendations**: A textbox with a limit of 200 characters to record any recommendations given by the therapist to the patient. Recorded in the recommendations field of the visits table.
- **IFSP minutes**: A certain type of therapy that has to be recorded if performed. The user records this with a drop down list control and picks the amount of minutes. Records the amount of minutes into the IFSPMinutes field of the Visits table.
- **IFSP Details**: This item stores any details about the IFSP meeting into the IFSPDetails attribute in the Visits table. This textbox has a limit of 200 characters and is only visible if a value other than zero is selected in the IFSP minutes drop down list above.
Validation controls are linked to all the fields that are required for completion so that all necessary information is filled out before submission. Once all the necessary information is complete the user clicks the submit button below the form which submits the data into the database and redirects the user back to the My Visits page. If the user wants to cancel a submission they can click the cancel button to redirect back to the My Visits page without submitting.

![Add New Visit Form](image)

**Figure 7.2 Add New Visit Form**
Once again, the way to add new visit data is nearly identical to the way to add new patients. Once the user is satisfied with the information they inserted into the controls they can hit the submit button to enter the data into the visits table. This is done by calling the addNewVisit() function shown in figure 7.3. This function starts by opening a connection to the database and calling the first stored procedure, spGetTherapy_BillingID. This takes the users choices for the TherapyType and TherapyTypeModifier drop down boxes and passes them in as parameters to get the proper TherapyBillingId to be inserted into the visits table as a foreign key. The function then uses a Data reader to read the result of the function. Next the function does a similar sequence to get the Employeeld from the Userid of the current user. Once this process is complete, the function then takes the rest of the parameters from the web server controls and passes them in as parameters into the stored procedure spAddNewVisit which works just like spAddNewPatient. One this function is complete he user is redirected back to the My Visits page.

```csharp
private void addNewVisit()
{
    string connStr =
        ConfigurationManager.ConnectionStrings["SLFTherapyConnectionString"].ConnectionString;
    SqlConnection connection = new SqlConnection(connStr);
    connection.Open();

    SqlCommand command = new SqlCommand("spGetTherapy_BillingId", connection);
    command.CommandType = CommandType.StoredProcedure;
    command.Parameters.Add("@TherapyId", SqlDbType.VarChar).Value = ddlTherapyType.SelectedValue;
    command.Parameters.Add("@ModifierId", SqlDbType.VarChar).Value = ddlTherapyModifier.SelectedValue;

    SqlDataReader rdr = null;
    rdr = command.ExecuteReader();
    int billingId = 0;
    while (rdr.Read())
    {
        // get the results of each column
        billingId = (int)rdr["Therapy_BillingID"];
        billingId.ToString();
    }
    //Close Reader
    if (rdr != null)
    {
        rdr.Close();
    }

    MembershipUser u = Membership.GetUser(User.Identity.Name);
    Guid userId = (Guid)u.ProviderUserKey;

    command = new SqlCommand("spGetContractorIdFromUserId", connection);
    command.CommandType = CommandType.StoredProcedure;
    command.Parameters.Add("UserId", userId);
```
Figure 7.3 addNewVisit() function

7.3 My Time Sheet

This page is nearly identical to the Time Sheets administrator page except that there is no drop down menu to select a contractor. The Time sheet is generated from the time period and the current user logged on so the user can only see their own time sheet. The user can still select the time period they desire using the calendar controls.
7.4 My Patients

The final page of the website is just a way for the contractors to view information about the patients assigned to them. The web page uses their username to find what patients belong to them through the Employees_Clients table in the database and bind it to a grid view.
8. Continuing Work and Future of Web Application

Though the web applications primary tasks are all complete there are a few more functionalities that need to be added:

1. User Management
   The website administrators need to be able to modify, delete and possibly lock out users if they please. Users also need to be able to change/ retrieve their password if its lost, compromised, or just want to change it for security purposes.

2. Edit and delete Patients
   Much like the users of the site, the administrator can only currently add patients they have no way of editing or deleting them.

3. Install SSL certificate
   As mentioned earlier in the report this application deals with sensitive medical information, The SSL certificate for encrypting has already been purchased it just needs to be installed.

4. Automatic Billing Functionality
   The billing numbers and alpha codes are included in the database because that data will eventually make out reports and bill patients through the third party billing company.

9. Conclusion

To conclude, this application should help the speech pathology office in a number of ways including cutting down on waste of time, effort, and paper. The application will help modernize the office and help the administrators and contractors use their time more efficiently in helping patients rather than struggling with an outdated administrative system.
Appendix A: References


