HealthPoint Medical Navigator

by

Nick E. Brenner

Submitted to
the Faculty of the Information Technology Program
in Partial Fulfillment of the Requirements for
the Degree of Bachelor of Science
in Information Technology

University of Cincinnati
College of Applied Science

June 2009
HealthPoint Medical Navigator

by

Nick Brenner

Submitted to
the Faculty of the Information Technology Program
in Partial Fulfillment of the Requirements
for
the Degree of Bachelor of Science
in Information Technology

© Copyright 2009 Nick E. Brenner

The information in this document is proprietary and may not be reproduced or distributed in whole or in part without the permission of the owner.

Nick E. Brenner

Robert E. Schlemmer, Faculty Advisor

Hazem Said, Ph.D. Department Head

Date 6/5/09

Date 6/5/09

Date 6/18/09
Acknowledgements

I want to thank everyone who helped me throughout the duration of this project. A special thanks goes out to Caroline Braden, HealthPoint representative that worked tediously with me on developing the requirements for the project and who spent numerous hours in meetings with me and numerous hours testing the product throughout various phases in its design. I would also like to give thanks to my current employer American Financial Group for being very supportive of time constraints that have encompassed this project and their active support of my further knowledge development in regards to this project. Additional thanks must be provided for all those tech bloggers that make projects like this possible for developers such as myself. Links provided in the citations section of this document only scratch the surface of the knowledge made available by the dedication of others to promote technology through the public share of their abilities as developers. A final thanks must be made to my friends, family, and most importantly my fiancé for being understanding of my time and efforts spent throughout the development of this product.
Table of Contents

1. Product Description and Intended Use 1
2. User Profiles 2
3. Design Protocol 3
4. Database 5
5. Testing Plan 8
6. Deliverables 9
7. Proof of Concept 10
8. Budget – Development 21
9. Risk Analysis 22
10. Conclusion 26
   Appendix A 27
   Appendix B 34
   Appendix C 44
   Appendix D 45
   Appendix E 51
   Code Fragments 51
List of Figures

Figure 1 - Medical Navigator Use Case Diagram 2
Figure 2 - Main Page with Links for Entering and Editing Patient Data 3
Figure 3 - Reports Page for Running and Managing Reports 4
Figure 4 - Hosting Company's Website for Administration of Hosting Website Account 4
Figure 6 - Updated timeline demonstrating testing efforts in parallel with development processes 8
Figure 7 - Login page for the HealthPoint Medical Navigator application 10
Figure 8 - View of master page in Visual Studio 2008 11
Figure 9 - User has credential level of 3 and a link to the administrator's page 12
Figure 10 - User has credential level of 2 and no link to the administrator's page 12
Figure 11 - Entering new user 13
Figure 12 - New user result message and result in refreshed table of current users 13
Figure 13 - Entering patient Test Test 14
Figure 14 - Notification of successfully adding patient Test Test 14
Figure 15 - Patient data table before insert statement 15
Figure 16 - SQL used to insert a new patient 15
Figure 17 - Patient data table after insert statement 15
Figure 18 - Visit data table before insert statement 16
Figure 19 - SQL used to insert new visit data 16
Figure 20 - Visit data table after insert statement 16
Figure 21 - Encounter data table before adding data join 17
Figure 22 - SQL used to insert logical join 17
Figure 23 - Encounter data table after adding data join 18
Figure 24 - SQL used to join patient and visit data 18
Figure 25 - SQL with results of joined data 19
List of Tables

Table 1: Hosting .................................................... 20
Table 2: Development ............................................. 21
Table 3: Risk Analysis ............................................ 24
Abstract

The HealthPoint Medical Navigator (HPMN) is an application customized for the nonprofit HealthPoint Family Care organization of Northern Kentucky. The application allows a HealthPoint representative also known as a HealthPoint Medical Navigator the ability to track and report on patient visit data between emergency departments located at five different Northern Kentucky hospital locations. The website code is written in html, asp.net, and C#. The database is SQL Server 2005 and supports the built-in Reporting Services feature. The HPMN user is able to view data at various emergency departments, then record desired data into this application via either the hospital's internet connection while visiting the emergency department, or from any other location where an internet is available when not in the hospital. The HPMN user is able to search patient entries, enter new patients and visit data, update, and delete data. Users with elevated credentials are also able add, edit, and delete users of the system. The HPMN user will also be able to link to a report server which hosts various reports for reporting back to management. The goal of the HealthPoint organization is to better educate individuals on the proper use of the emergency department and direct them to the most beneficial treatment for their condition. The HealthPoint Medical Navigator application will provide a means to track HealthPoint representatives' interaction with the patients and view the progress of their program.
HealthPoint Medical Navigator

1. Product Description and Intended Use

"There are many residents of the Greater Cincinnati area who cannot afford proper health care and who do not understand where to find proper healthcare even when funding does exist." (3) Institutions do exist to help these individuals in times of need, but funding and knowledge of how to use these resources is limited. This product is designed to help organizations track the activities of these individuals and better plan how to spend funding designed to address this problem. The primary organization this product is being developed for is known as HealthPoint Family Care, and will be referred to as HealthPoint throughout the remainder of this document. This product can be adapted to other organizations that perform similar activities; however, for simplicity, this report will discuss directly the actions and use of this product as developed for HealthPoint. HealthPoint has the task of recording data from five area hospitals in the Northern Kentucky region. A representative is to travel from hospital to hospital and manually record data from patients who are in need of greater assistance. This product will allow the representative to log into a remote location (website) and enter the data they have collected into a web form. There will be hyperlinks on the main page of the website which will allow the user to navigate to a patient entry form, patient update form, and an external site to be used for reporting purposes. Reports will be predefined; however, there will be the ability to upload new reports as needed. There will also be reports that allow the user to select various parameters to achieve more dynamic results for their queries. The intent of this product is to provide a digital means of recording data as it relates to patients' visits to the emergency rooms of local hospitals and provide a reporting mechanism to show a pattern of their visits. This information will help to demonstrate how the HealthPoint organization has helped provide a program that helps increase the patients' knowledge of health care resources that are available to them and hopefully cut back on the local hospitals' expenditures of caring for patients who arrive without a means to pay for the care.
2. User Profiles

Figure 1 demonstrates the type of users and the associated access they will have with the system. There will be two types of users accessing the system outside of the development team. A HealthPoint Associate will be seen as an individual who only needs to enter patient data and run reports from the reporting site. A HealthPoint lead will act more as an administrator and will be able to create additional HealthPoint Associate users within the system. (4)
3. Design Protocol

This application utilizes .NET 2.0 as its framework. The code behind the pages on the website is written in the C# development language and coded using Microsoft’s Visual Studio environment. There are two main components that accompany this application. First, there is a website which the user will utilize to enter and edit patient visit data. Second, there is a report management website that allows users to run, export, and edit reports. There is an additional component which may be utilized by management or administration to edit the account information in regards to both services provided by the hosting company and creating additional users for the report site. Below, Figure 2 demonstrates the preliminary layout of the patient data entry and edits screens. Figure 3 demonstrates the report management site. Figure 4 demonstrates the hosting company’s administration page.
Figure 3 - Reports Page for Running and Managing Reports

Figure 4 - Hosting Company's Website for Administration of Hosting Website Account
4. Database

The database backend that will be utilized is Microsoft's SQL Server 2005. This database will consist of two main tables which will store information about the patients in one table and information about the patient's visits in another table. There will also be two tables which administer secure access to the website by storing usernames, passwords, authority levels, and descriptions of those authorities. In addition, there will be several smaller tables which drive the dropdown lists that are shown on the patient data entry form. These smaller tables will contain data for fields that may change periodically throughout the lifetime of the application. Examples of this data would be the names of hospitals available for recording data for, types of insurance carried by the patients, and lists of reasons for referral of the patient to for other medical treatment. Furthermore, with the main driving force of developing this application being the need for reporting, there are several views that will be created to assist with generation of reports from the report site. The report site utilizes Microsoft SQL Server Reporting Services which comes packaged with MS SQL Server 2005. Please reference the database diagram on the following page for details on table and view design.
Figure 5 - Database Diagram of HealthPoint Medical Navigator application. Tables are shown on the left with views shaded on the right.
5. Application Code

The code that is used in creating this application primarily utilizes the ASP.NET architecture with programmatic code being in the language of C#. The website contains source code files which contain a combination of HTML code and ASP control items. Each page also has a code file associated with it containing C# code that handles database connectivity, user authentication and authorization, and error catching. In addition, the site will also utilize a master page document to provide a consistent look and feel to the web application through the various pages. Pages included in the site will be:

- Login page, unauthorized users will be redirected to this page for authentication.
- Index page, referred to as home page within the site. This page will contain a patient search function to look up information on a particular patient via search by lastname. From here, the user will be able to select this patient to view full patient details and make updates to the data. The index page will also contain an authorization check to verify what level of authority the user has and will determine whether or not to provide a link for the site configuration page.
- Site configuration page. This page will allow the administrative users of the application to add new users, update a user, and delete a user.
- Enter new patient page. This page will contain a form which allows the user to enter new information in regards to a patient visit they have recorded at one of the area hospitals. Upon successful submission of the form the user will be prompted stating that the entry was a success.
- Update page. This page will contain a form for updating patient data. This page is linked to the search feature on the index page.
6. Testing Plan

This application has been tested throughout the whole project as changes have been made. The database has been tested for successful storage and retrieval of encrypted data types. The reports will have been tested to ensure they meet the requirements of the user. In order to ensure this, several meetings were held with the end user to determine the effectiveness of the reports. Also discussed in the meetings was the effectiveness of data entry and relevance of fields used. This project relied heavily upon user acceptance testing (UAT). The driving force behind this project was an end user with a need for an IT solution, not an IT solution in need of an owner. Numerous meetings were held with the end user, a co-worker of the end user who would also be using the system, and management members of the end users whom eventually had to give the okay for the project development and implementation. These meetings were crucial in helping with the development and testing of the project as the developer had little if any knowledge of the task they were developing for. Continually meeting with the users enabled the development of the final project to achieve the goal set forth from day one. This goal was the creation of a useful application that would not only assist the HealthPoint representatives in collecting and reporting on data, but to also assist in solving the larger problem of educating citizens on the proper use of the emergency departments at their local hospitals.

<table>
<thead>
<tr>
<th>Task Name</th>
<th>Start</th>
<th>Finish</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 Development</td>
<td>Mon 12/1/08</td>
<td>Wed 5/6/09</td>
</tr>
<tr>
<td>2 Creation of SQL Server DB</td>
<td>Mon 12/1/08</td>
<td>Fri 1/3/09</td>
</tr>
<tr>
<td>3 Creation of Web Pages (No program)</td>
<td>Mon 12/1/08</td>
<td>Fri 1/3/09</td>
</tr>
<tr>
<td>4 Creation of Programmatic Code</td>
<td>Mon 1/5/09</td>
<td>Wed 5/6/09</td>
</tr>
<tr>
<td>5 Creation of Reports on Report Server</td>
<td>Mon 2/6/09</td>
<td>Wed 5/6/09</td>
</tr>
<tr>
<td>6 Testing</td>
<td>Thu 1/1/09</td>
<td>Wed 5/4/09</td>
</tr>
<tr>
<td>7 Test SQL Server DB</td>
<td>Mon 1/2/09</td>
<td>Wed 5/6/09</td>
</tr>
<tr>
<td>8 Test Web Page Design</td>
<td>Thu 1/1/09</td>
<td>Wed 5/6/09</td>
</tr>
<tr>
<td>9 Test Programmatic Code</td>
<td>Mon 1/6/09</td>
<td>Wed 5/6/09</td>
</tr>
<tr>
<td>10 Testing of Reports</td>
<td>Mon 2/6/09</td>
<td>Wed 6/6/09</td>
</tr>
<tr>
<td>11 Request Design Freeze for approval</td>
<td>Mon 4/20/09</td>
<td>Mon 4/20/09</td>
</tr>
<tr>
<td>12 Present at Tech Expo</td>
<td>Thu 2/16/09</td>
<td>Thu 5/7/09</td>
</tr>
</tbody>
</table>

Figure 6 - Updated timeline demonstrating testing efforts in parallel with development processes
7. **Deliverables**

1. A Web-based data entry system for recording patient visit data.
2. Report pages site with ability to manage various reports, run reports, and save the results of report execution.
3. Documentation of how to utilize the data entry portion of the site.
4. Documentation of how to manage the report folders, save report execution results, and setup timed execution of reports with email delivery.
5. A master page to ensure consistent design throughout the site and ease in the assistance of corporate logo changes.
6. A secure method of data entry requiring custom user authentication and authorization via users stored in an external database.
8. Proof of Concept

The login screen. This page greets the user as they navigate to the site. Since the user is not authenticated they are directed to this login screen to enter their credentials. See Figure 7 below for a screenshot of this page.

![Figure 7 - Login page for the HealthPoint Medical Navigator application](image)

At this point the user will enter their credentials and be directed to either the default web page which is set to index.aspx, or if they were trying to navigate to a separate page while not authenticated they will be redirected to the page in which they were attempted to navigate to upon entering proper credentials.
The master page. This page is used as the format for all pages contained within the site. The items displayed on this page are to be found on each page throughout the website. These items are the corporate logo, a display of the current user, a link back to the home page, a logout mechanism for the user, and background formatting. The image displayed in Figure 8 below demonstrates the master page as it appears in the development environment (in this case, Visual Studio 2008). The image is shown in the development environment because pages with the extension of .master are not served by the web server. An attempt to view this page within a browser will inform the user as such.

![Figure 8 - View of master page in Visual Studio 2008](image-url)
The index page. This page is referred to as the home page of the web site. This is the default page a user is brought to upon authentication. Note in Figure 9 below that this user has a credential level of 3 and is provided the link to site configuration to administer users of the site. In Figure 10 below the user demonstrated has a credential level of 2 and is not provided with the link.

![Figure 9 - User has credential level of 3 and a link to the administrator's page](image)

![Figure 10 - User has credential level of 2 and no link to the administrator's page](image)
The administrator's page. This page provides the higher credential user of level 3 access to update user credentials, add users, and delete users. See Figure 11 for an example of enter data for a new user, and then see Figure 12 for demonstration of the user being added.

Figure 11 - Entering new user

Figure 12 - New user result message and result in refreshed table of current users
The patient entry form. This page contains all of the fields that have been requested to be tracked for each patient visit. Not all fields are required for entry, but may be updated later upon searching for a particular patient on the index page and updating the record with additional data. Reference Figure 13 below for an example of entering a patient named Test Test with an EncounterNum of 0123456789 and reference Figure 14 below for result of submitted data.

Figure 13 - Entering patient Test Test

Figure 14 - Notification of successfully adding patient Test Test
The database. Below is a demonstration of structured query language (SQL) that is used to enter patient data into the system and to extract patient data from the system. With over 60 columns of data associated to each patient and their visit data only a few columns will be used for demonstration as not all of the data will be visible in the provided screen shots. See Figure 15 below for a view of patient data table with one patient in it before inserting an additional patient. See Figure 16 for the SQL used to insert a new patient. See Figure 17 for a view of the patient data table after inserting an additional patient. The following pages will demonstrate inserting visit data and joining of the tables.

![SQL code and table views](image)

Figure 15 - Patient data table before insert statement

![SQL code and table views](image)

Figure 16 - SQL used to insert a new patient

![SQL code and table views](image)

Figure 17 - Patient data table after insert statement
The following images will demonstrate inserting visit information for a patient. See Figure 18 for a view of the visit data table before insertion of new visit data. See Figure 19 for the SQL used to insert the visit data. See Figure 20 for a view of the visit data table after inserting visit data.

![Figure 18 - Visit data table before insert statement](image1)

![Figure 19 - SQL used to insert new visit data](image2)

![Figure 20 - Visit data table after insert statement](image3)
This page will provide us with the SQL used to link the patient data with the visit data. To do this within the website, code was used to store the identity for the primary key of each record that is inserted to the patient and visit tables before inserting the keys into a joining table. For the sake of demonstration we will substitute the key values shown in Figures 17 and 20 on the preceding pages for the joining keys in the SQL below in place of C# code variables that are used within the code files of the website for the SQL insert statement. See Figure 21 below for a view of the Encounter data table before a new join has been added. Figure 22 demonstrates the SQL used to insert a new logical join into the joining table. Figure 23 provides a view of the Encounter data table after the join has been added.

```
<table>
<thead>
<tr>
<th>Encounter_ID</th>
<th>Patient_ID</th>
<th>Visit_ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>NULL</td>
<td>NULL</td>
<td>NULL</td>
</tr>
</tbody>
</table>
```

Figure 21 - Encounter data table before adding data join

```
INSERT INTO [SQL2K505_445579_hpmn].[dbo].[tblEncounterData] VALUES (2, 3)
```

Figure 22 - SQL used to insert logical join
This page will provide us with the SQL used to join the patient data and visit data together. Figure 24 below provides us with the SQL used to join the tables. Only a few fields are shown in this example although many more may be used in developing reports. Figure 24 shows the joining SQL along with the results of executing the SQL statement.

```
select p.Patient_Firstname, p.Patient_Lastname, 
v.EncounterNum, v.Hospital, v.Date_Patient_Seen
from tblPatientData p
inner join dbo.tblEncounterData e on p.Patient_ID = e.Patient_ID
inner join dbo.tblVisitData v on v.Visit_ID = e.Visit_ID
```

Figure 24 - SQL used to join patient and visit data
```sql
-- SQL Query

SELECT p.Patient_Firstname, p.Patient_Lastname,
       v.EncounterNum, v.Hospital, v.Date_Patient_Seen
FROM tblPatientData p
INNER JOIN dbo.tblEncounterData e ON p.Patient_ID = e.Patient_ID
INNER JOIN dbo.tblVisitData v ON v.Visit_ID = e.Visit_ID
```

Figure 25 - SQL with results of joined data
9. Budget – Hosting

There are two aspects to consider when evaluating project cost. The first aspect is the fact that an annual cost will be incurred whether the project is completed by the student or by a consultant from a commercial entity. This cost can be observed in Table 1.

Table 1: Hosting

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
<th>Retail Cost</th>
<th>Cost Incurred</th>
</tr>
</thead>
<tbody>
<tr>
<td>DiscountASP.net Web Hosting Plan</td>
<td>Annual cost of web hosting service</td>
<td>$90.00</td>
<td>$90.00</td>
</tr>
<tr>
<td>Domain Name Registration Service</td>
<td>Registration of website domain</td>
<td>$15.00</td>
<td>$15.00</td>
</tr>
<tr>
<td>DiscountASP.net Database Hosting Plan</td>
<td>Annual cost of MS SQL 2005 database hosting service</td>
<td>$119.67</td>
<td>$119.67</td>
</tr>
<tr>
<td>DiscountASP.net Reporting Services</td>
<td>Annual cost of SQL 2005 Reporting Services hosting service</td>
<td>$59.84</td>
<td>$59.84</td>
</tr>
<tr>
<td>Windows Server 2003 running</td>
<td>Provided by hosting company</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Internet Information Services (IIS)</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Retail Total: $284.51 $284.51
10. Budget – Development

The second cost aspect to consider is that of hiring a consultant to work on the project in house.

Table 2 demonstrates the cost of providing software needed by the consultant for the project and the difference in cost between student labor and private consultant labor. The software used by the student is used at no cost to the client or to the student as it is provided as an educational tool for a student working on projects at the University of Cincinnati’s College of Applied Science.

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
<th>Retail Cost</th>
<th>Cost Incurred</th>
</tr>
</thead>
<tbody>
<tr>
<td>Microsoft Windows XP Professional</td>
<td>Provided by MSDN Academic Alliance program</td>
<td>139.99</td>
<td>0</td>
</tr>
<tr>
<td>Microsoft Visual Studio .NET 2005</td>
<td>Provided by MSDN Academic Alliance program</td>
<td>689.99</td>
<td>0</td>
</tr>
<tr>
<td>Microsoft SQL Server 2005</td>
<td>Provided by MSDN Academic Alliance program</td>
<td>1,189.95</td>
<td>0</td>
</tr>
<tr>
<td>Student Labor</td>
<td>Cost of student work for 100 hours @ $15/hr</td>
<td>0</td>
<td>1,500.00</td>
</tr>
<tr>
<td>Commercial Labor</td>
<td>Cost of consultant work 100 hours @ $50/hr</td>
<td>5,000.00</td>
<td>0</td>
</tr>
<tr>
<td><strong>Retail Total:</strong></td>
<td></td>
<td><strong>$7,019.93</strong></td>
<td><strong>0</strong></td>
</tr>
<tr>
<td><strong>Proposed Student Project Cost:</strong></td>
<td></td>
<td>0</td>
<td><strong>$1,500.00</strong></td>
</tr>
<tr>
<td><strong>Total Savings:</strong></td>
<td></td>
<td>0</td>
<td><strong>$5,519.93</strong></td>
</tr>
</tbody>
</table>
11. Risk Analysis

Due to the nature of this project and the sensitivity of the data being reported on, a great deal of consideration was taken towards risks of both functionality and security of the project.

Included in this document is a table which provides a risk analysis of the project which is broken down by the application or resource involved, the risk that is observed, the risk rating of: "High", "Mid", or "Low", and any mitigating controls that will lessen or remove the risk. The level of risk is based on both security concerns and system functionality. For example, if there were an area of risk where an unauthorized individual may be able to easily obtain access to data the level of risk would be "High". For an area of risk that may demonstrate the ability for an unauthorized individual to obtain data specifically through the use of advanced tools and possibly illegal activities, the level of risk would receive a rating of "Mid" or "Low" and be accompanied by a mitigation control to help reduce this risk.

Risks that surround the functionality and availability of the system are rated in a similar fashion. For instances where the system may be temporarily unavailable due to updates or code enhancements, the level of risk would receive a level of "Mid" or "Low" based upon the timeliness of the situation. For instances where the system is unavailable and there is a loss of unrecoverable data, a rating of "High" would be assigned. Please reference Table 3 on the following pages for a complete listing of the risk analysis.
<table>
<thead>
<tr>
<th>Application / Resource</th>
<th>Statement of Risk</th>
<th>Level of Risk</th>
<th>Mitigating Control</th>
</tr>
</thead>
<tbody>
<tr>
<td>ASP.NET Web Service / IIS</td>
<td>Unable to access service due to account credentials.</td>
<td>Low</td>
<td>Setup development user with credentials to access appropriate web services.</td>
</tr>
<tr>
<td></td>
<td>IIS version not compatible with currently installed .NET architecture.</td>
<td>Mid</td>
<td>Ensure most recent IIS version installed on development system and is compatible with IIS version on hosting server.</td>
</tr>
<tr>
<td></td>
<td>IIS version not compatible with code pages for site.</td>
<td>Low</td>
<td>Ensure most recent IIS version installed on development system and is compatible with IIS version on hosting server.</td>
</tr>
<tr>
<td></td>
<td>Development system is configured for a different version of IIS than the hosting server.</td>
<td>Mid</td>
<td>Ensure most recent IIS version installed on development system and is compatible with IIS version on hosting server.</td>
</tr>
<tr>
<td></td>
<td>Uploading code and html files to remote server via ftp is not allowed.</td>
<td>Low</td>
<td>Connect via secure connection only when uploading files to hosting server.</td>
</tr>
<tr>
<td></td>
<td>Uploading code and html files to remote server via ftp is not secure.</td>
<td>Mid</td>
<td>Only connect via secure connection when uploading files to hosting server.</td>
</tr>
<tr>
<td></td>
<td>Website is not configured for secure authentication methods.</td>
<td>High</td>
<td>Properly configure web.config file for authorization and authentication methods.</td>
</tr>
<tr>
<td>SQL Server</td>
<td>Development edition varies from server edition.</td>
<td>Low</td>
<td>Ensure development system has compatible version of SQL Server with hosting server.</td>
</tr>
<tr>
<td>------------</td>
<td>--------------------------------------------------</td>
<td>-----</td>
<td>----------------------------------------------------------------------------------</td>
</tr>
<tr>
<td></td>
<td>Remote connections are not enabled to allow for access to remote database.</td>
<td>Mid</td>
<td>Run surface area configuration to allow for remote connections and make proper adjustments to firewall to enable proper database connectivity.</td>
</tr>
<tr>
<td></td>
<td>Multiple users have access to data in development/production environments.</td>
<td>Mid</td>
<td>Configure database to allow only select users access to database objects.</td>
</tr>
<tr>
<td>SQL Server edition varies from the hosting server’s Reporting Services edition.</td>
<td>Mid</td>
<td>Ensure development system has compatible version of Reporting Services with hosting server.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Back-up of database is not present, resulting in loss of data in the event of possible service failure.</td>
<td>High</td>
<td>Create a back-up procedure to have data archived in timeframe deemed necessary by business requirements.</td>
</tr>
<tr>
<td>SQL Server Reporting Services</td>
<td>Reporting Services is not configured with the same access for users as the website.</td>
<td>Mid</td>
<td>Configure security for users and roles within Reporting Services as identified by business rules for report generation.</td>
</tr>
<tr>
<td></td>
<td>Restrictions are not in place on report definitions, allowing any user to alter reports.</td>
<td>Low</td>
<td>Configure security for users and roles within Reporting Services as identified by business rules for report generation.</td>
</tr>
<tr>
<td></td>
<td>Reporting Services edition varies from the hosting server’s SQL Server edition.</td>
<td>Low</td>
<td>Ensure development system has compatible version of SQL Server with hosting server.</td>
</tr>
<tr>
<td>Development / HTML and Code Pages</td>
<td>Report definitions are not archived in the event of possible service failure.</td>
<td>Mid</td>
<td>Save archive set of report definitions onto back-up media off of server in the event of need for restore.</td>
</tr>
<tr>
<td>----------------------------------</td>
<td>--------------------------------------------------------------------------------</td>
<td>-----</td>
<td>--------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Configuration files are not encrypted, allowing users to view connection strings and other critical settings.</td>
<td>Low</td>
<td>Utilize encryption method to protect web.config file such as those provided on the CodeProject Web Site (5).</td>
<td></td>
</tr>
<tr>
<td>Configuration files have debugging enabled in production code.</td>
<td>Low</td>
<td>Alter web.config file, disabling the debugging method.</td>
<td></td>
</tr>
<tr>
<td>Files are not backed-up on separate media, allowing for quick restore of Web site after possible service failure.</td>
<td>Mid</td>
<td>Save archive set of files onto back-up media off of server in the event of need for restore.</td>
<td></td>
</tr>
</tbody>
</table>
12. Conclusion

The HealthPoint Medical Navigator project will simplify the arduous task of manual data recording and reporting for the HealthPoint organization. In doing so, the project will also allow for a greater education of resources available to residents of the regions surrounding the local hospitals. Finally, the reporting will allow for a better understanding of how to budget funding in the future.

Many challenges were faced throughout the development of this project. The largest hurdle to overcome was the comprehension of the business processes involved with the project. The development had to help drive the business in defining a clear logical process of exactly what steps were being taken throughout the data gathering and reporting process. In addition, there were a lot of fields being created that the developer did not initially understand the meaning behind the data and exactly how the business was going to use this data in a desired report form. It is strongly suggested that when embarking on a project that the developer has little background on in regards to the business processes, that there needs to be a considerable amount of time allotted to requirements gathering and communication with the end users before development even begins. There was a lot of redevelopment throughout the course of the project as insight was built upon to the overall driving purpose behind it. In the end, both the developer and the end user have learned valuable lessons from this experience in the means of project development and management. The end product took longer to complete than anticipated. Even with lessons learned and bumps and surprises encountered along the way, the user and developer are happy with the end result. In fact, the developer has already received requests for additional features to be added which were never discussed before. The users are currently using this product in a live environment, have seen its benefits, and are eager to have more work done.
Appendix A
Navigating the data entry website

The url for the website is http://hpmn.org. When navigating to the provided url, the unauthenticated users are taken to the login page as shown in Figure A1 below. Users will be provided a username and password by their manager.

A 1- Users first must login to access the site

Once a user has been authenticated from the login screen they are taken to the default page of the website which we will refer to as the homepage. The hyperlink in the upper center of the page for "Home" will link the user back to this homepage when it is clicked from anywhere within the site. In Figure A1 above we see that there is not a name to the right of the text "Currently logged in as". The space to the right of this text will be populated with the user's username upon logging into the site.
The homepage shown in Figure A2 on the following page shows the username of Nick, the name of the developer used for documentation purposes only, after both the “Currently logged in as” text and the “Welcome” text. The “Welcome” text is specific to the homepage and will not be shown throughout the website as the “Currently logged in as” section will. The homepage contains 4 hyperlinks below the “Welcome” text. The first hyperlink “Click here to Administer website user accounts” only shows for users with elevated privileges which and add, edit, and delete users of the site. The second hyperlink “Click here to enter data for a new Patient Encounter” is available to all users and will direct the user to a page for entering in new patient and visit data. The third hyperlink “Click here to navigate to Reports site” will direct the user an external report management site for executing and administering reports. The fourth hyperlink “Click here to navigate to Current Stats Page” will direct the user to a temporary page which will be unavailable in an upcoming release. This page is currently used for depicting a distinct count of patients seen and total count of visits that have been recorded in the system. This mechanism was put into place while requirements were still being determined and was deemed useful on a temporary basis by the end users.

Below the hyperlinks is a search feature which allows a user to search for a particular patient they have entered by last name. The contents of the results grid have been hidden within the screen shot in Figure A2 on the follow page to help protect the identity of individuals whom may be displayed. When the results grid is populated, the Patient_ID column to the far left of the grid appears as a hyperlink and will direct the user to a page where they can update data for this particular patient. The amount of fields displayed in the grid on the homepage was limited to just enough to be able to determine the specific individual a user was searching for was that individual. When the user is directed to an update page they are provided with all available fields for updating.
A 2 - Homepage of the hpmn.org website

After a user with elevated privileges clicks on the "Click here to Administer website user accounts" hyperlink they are directed to the User Configuration page as shown in Figure A3 below.

A 3 - User Configuration page for administering website users
Figure A3 on the previous page consists of 3 distinct areas for user administration. The panel on the left-hand side allows a user with elevated permissions to alter a user by selecting the username from a dropdown list then entering a password and authority levels for the user. A chart explaining the various authority levels is provided as reference at the bottom of the panel. The panel on the upper right portion of the screen allows a user with elevated permissions the ability add a new user to the system, whereas the panel below the add new user allows for a quick view of current website users and permits deletion of a user by clicking on the “Delete” hyperlink to the left of the username. Note, a user cannot be added to the system which already exists, please reference the Proof of Concept section contained in this document for evidence of this control being in place and operating as intended. When a user has completed their user administration tasks they may return to the homepage by simply clicking on the “Home” hyperlink in the upper middle portion of the screen.

After clicking on the “Click here to enter data for a new Patient Encounter” hyperlink the user is directed to the EnterNewPatient.aspx page as shown in Figure A4 below. Screen shot is zoomed out to enable viewing of numerous fields available for data entry.
Figure A4 on the previous page provides a view of the numerous fields that data is being collected for. Some of these fields are available as dropdown lists as only specific data should be assigned to these areas. Other fields have validation mechanisms placed on them to ensure proper format is used when entering data into the text boxes. For example, the EncounterNum field will only allow up to 10 digits and may not contain any letters; therefore a regular expression must be used to prevent a malformed entry by the user. See Figure A5 below and Figure A6 on the following page for examples of this validation control in place and working as intended. Additional validation controls have been put into place to manage the entry of data within the date fields to check for date format and valid dates (those from year 1904 and beyond).
A 5 - Too many digits cause a validation error

A 6 - Use of a letter also causes a validation error

Once the user has successfully entered and submitted their data they are provided with a success at the lower portion of the screen as the submit button also appears in the lower portion of the screen. If an error is received, that message will display in place of the success message. At this point if the user is finished using the system they may choose to logout using the logout button found in the upper right-hand portion of the page or they may return to the homepage by clicking the “Home” hyperlink in the upper-middle portion of the page.
From the homepage, after a user clicks the “Click here to navigate to Reports site” hyperlink, the user is prompted to enter a different set of username and password credentials. A different set is used because not all users of the report site will have access to the patient and visit data entry website. For example, in a meeting with HealthPoint’s IT infrastructure provider DPS, one of their representatives Scott Balmos, stressed some familiarity with MS SQL Server Reporting Services and agreed to provide some support and possibly create some reports for HealthPoint upon the completion of this project after initial development was completed. Scott would not be a user for the data entry portion of the site, but would be called upon for support with regards to reporting, therefore a separate username and password are needed. See Figure A7 below for demonstration of user being prompted for additional credentials after clicking on “Click here to navigate to Reports site” hyperlink. Navigation and use of the “Reports site” will be demonstrated in Appendix B Navigation of Reports Site.

A 7 - User is prompted to enter additional credentials to access reporting site
Appendix B
Navigating the Reports Site

The Reports site is a separate entity from the data entry site. Upon entering valid credentials at the prompt from the homepage of the data entry site as described in Appendix A the user is directed to the Reports site as demonstrated in Figure B1 below. This is the root folder location of the Reports site for HealthPoint. From this point the user should click on the folder hyperlink “hpmnorg0000”.

B1 - Root folder and top level page for Reports site

After clicking the “hpmnorg0000” hyperlink as demonstrated in Figure B1 above, the user is taken to a secondary level folder called Reports as shown in Figure B2 on the following page. These top two level folders are controlled by the hosting company and are not to be modified as this will cause headaches in attempting to reconfigure proper report deployment settings. Figure B2 shows a “Contents” and a “Properties” tab above the folder window. Notice in Figure B3 on the following page that the properties tab demonstrates the folder was created by the report server administrator, in this case, the hosting company. Also notice the only available property is “General”, there is not “Security” property available.
B 2 - First level folder within hpmnorg0000 root folder

B 3 - Properties of Reports folder shown in Figure B2

At this point the user can click the contents tab to return to the view shown in Figure B2, then click on the “Reports” folder hyperlink. The user is now taken to where the reports and data sources are stored for the reports. Reference Figure B4 on the following page for a view of the Reports folder contents.
Notice additional items have appeared in the bar between the folder contents and the "Contents" and "Properties" tabs. Users have more control over the contents within this folder. Users are able to create new folders, define data sources, upload report definitions, and more. The amount of freedom a user has can be limited to security placed on the folder by administrative users. The hosting company has provided a default administrative user which has been used to assign credentials for the current users. From this folder additional folders can be created with various security levels placed on each folder. Security levels can also be placed on individual reports and data sources. See Figure B5 and Figure B6 on the following page for a view of the security settings within this folder. Screen shots in Figure B5 and Figure B6 on the following page were taken after clicking on the properties tab of this current folder.
B 5 - Properties tab of Reports folder

Notice the security section that is available in the panel on the left now. This option was not available on the higher level folder security as the top level folder is maintained by the hosting company. See Figure B6 below for a view of the security settings.

B 6 - Security settings for the Reports folder
Editing security of the folders and items is made easy through the graphical user interface. See Figure B7 through Figure B10 for an example of how to edit security settings. Beginning with Figure B6 a list of users and their credentials for that folder are displayed. Clicking on the “Edit” hyperlink to the left of the user to edit security for will direct the user to security page shown in Figure B7 below.

B 7 - Security page for the Reports folder

Notice the list of Roles available for user Caroline as shown in Figure B7 above. To change the security settings for this individual on this particular folder simply click the check box for Roles that are desired.

For demonstration purposes the Browser role will be deselected for this user. After deselecting the checkbox by “Content Manager” and clicking the “Apply” button the user is redirected to security overview of the folder as shown in Figure B8 on the following page. Notice in Figure B8 that user Caroline no longer has the role of “Content Manager”.
B 8 - Content Manager role removed for user Caroline

The following screen shot in Figure B9 demonstrates using a different browser – in this case the Opera browser – to log in to the Reports site as a Caroline to demonstrate security credentials.

B 9 - Opera browser demonstrating logging in as a user Caroline
B 10 - User Caroline is unable to view/edit security as her role as Content Manager was removed.

Running a report is as simple as selecting the report by name from the Reports folder. For demonstration, begin with the content view of the Reports folder as shown in Figure B.11 below, then select the “InsuranceWithParameters” report hyperlink which takes the user to a page view as shown in Figure B.12 on the following page.

B 11 - Contents of the Reports folder
Figure B12 below shows the report page before rendering. This report is configured to accept user parameters and therefore will only be displayed after the user enters parameters and clicks on the “View Report” button on the right-hand side of the page. Figure B13 shows results of report running after selecting data for the first quarter of 2009, i.e. selecting the months January, February, and March, then selecting the year 2009.

B 12 - Report waiting for user parameter selection

B 13 - Report with count totals, patient data removed from image for security
Note: A user may also choose to select to export the results of the report. Figure B14 below demonstrates the available export options.

Additional properties are also available for each report. Note the tabbed section above the report. A user has the ability to “View”, edit the “Properties” (i.e. the security for this individual report), view the “History” to view when the report was last ran, and one of the most important features is that a user can create “Subscriptions” for timed report execution and delivery.
Clicking on the "Subscriptions" tab will bring the user to a page view as shown in Figure B15 below.

![Image](image.png)

**B 15 - Current subscription for this report**

From this screen the user can either create a new subscription or edit a subscription that is already in place. For demonstration, Figure B16 on the following page shows the view after clicking on the "Edit" hyperlink to show a current subscription setup. The user is able to select e-mail or file distribution, choose email addresses or folder locations, choose file format of the report, provide comments, redefine default parameter values for reports with parameters, and assign a timeframe that the report should be automatically distributed.

![Image](image.png)

**B 16 - Subscription configuration form**
Appendix C
Report Deployment Settings

C 1 - Deployment settings for deploying reports from MS Visual Studio

C 2 - Deployment settings for deploying models from MS Visual Studio
Appendix D
DiscountASP.net Hosting Site

Figure D1 below greets the user for login at my.discountasp.net. The control panel (CP) login and password can be obtained from management personnel and are not included in the documentation for security and management purposes.

D 1 - Control Panel login page at my.discountasp.net
Figure D2 below demonstrates the default view the user will see after providing proper credentials at the login screen. The list of account features, management, billing, and tools are found in the left-hand column, a list of news and messages can be found in the center column, and the right-hand column is typically reserved for advertising account add-ons and services that are provided by third party vendors.

Figures D3 through D10 over the next several pages will provide screen shots of some of the various information and management options that are available via clicking the links in the left-hand column of this page.
D 3 - SSL Management view for security certificate management

D 4 - Usage Summary depicting site usage bandwidth
D 5 - Usage Summary depicting disk space usage

D 6 - IIS Tools depicting site running ASP.NET version 2.0 or higher
D 7 - IIS Tools depicting revised Default Documents list

D 8 - IIS Tools depicting FrontPage Extensions installed to assist in development using Microsoft development tools
D 9 - MS SQL 2005 demonstrating usage, server name/location, and connection string info

D 10 - MS SQL SSRS 2005 demonstrating Reporting Services connection info and user management console for adding Reporting Services users and modifying usernames and passwords. User rs_445579 cannot be edited as that is the default account provided by DiscountASP.net
Appendix E

Code Fragments

protected void OnLogin(object sender, EventArgs e)
{
    string LoginComn = "Data Source=sq12X511.discountasp.net;Initial Catalog=SQL2005_445579_hpem;Persist Security Info=True;U
SqlConnection cn = new SqlConnection(LoginComn);
SqlCommand cmd = new SqlCommand();
    cmd.Parameters.AddWithValue("UserName", m_textboxUserName.Text);
    cmd.Parameters.AddWithValue("Password", m_textboxPassword.Text);
    cmd.CommandText = "Select username, password, cred_ID from HP_LOGINS WHERE username = @UserName and password = @Password";
    cmd.Connection = cn;
SqlDataReader Dr;
    cn.Open();
    Dr = cmd.ExecuteReader();
if (Dr.Read())
    {
        Session["Login"] = (string)Dr["username"];
Session["Password"] = (string)Dr["password"];
Session["Credential"] = (string)Dr["cred_ID"];  
FormsAuthentication.RedirectFromLoginPage(m_textboxUserName.Text.Trim(), false);
    }
else
    {
        lblLoginError.Visible = true;
        lblLoginError.Text = "Login failed. Please try again.";
    }
}

E 1 - Login page utilizing custom database authentication methods. Note use of parameters to prevent against
SQL Injection attacks
try
{
    cmd.ExecuteNonQuery();

    strSQLSuccess = "Select Patient.Firstname, Patient.Lastname from Patient_Data where Patient_ID = @Identity";
    SqlCommand cmdSuccess = new SqlCommand(strSQLSuccess, cn);
    SqlDataReader reader = cmdSuccess.ExecuteReader();
    while (reader.Read())
    {
        FirstName = reader[0].ToString();
        LastName = reader[1].ToString();
    }
    if (FirstName == null)
    {
        FirstName = "No record added";
        LastName = "No record added";
        EncounterNumSuccess = "No record added";
    }
    lbSQLMessage.Text = "Successfully added " + FirstName + " " + LastName + ";";
}
catch (SqlException sqlEx)
{
    catchEX = sqlEx.ToString();
    lbSQLMessage.Text = catchEX;
    cn.Close();
}
finally
{
    cn.Close();
}

E 2 - Obtaining identity of last user entered when SQL insert statement is successful
protected void btnAlterUser_Click(Object sender, EventArgs e)
{
    if (dropDownList1.SelectedValue.Value != null)
    {
        string alterUserPwd = "Update HP_Logins set password = '" + textAltUserPwd.Text + ", cred_ID = " + Convert.ToInt16(textAltAuthLevel.Text) + ";"
        SqlConnection alterUserConn = new SqlConnection("Data Source=sq2kk351.discontoyr.net;Initial Catalog=SQL2005_445579_open;Persist Security Info=True;");
        alterUserConn.Open();

        SqlCommand cmdAlterUser = new SqlCommand("alterUserConn = new SqlConnection("alterUserConn.Close();
    }
}

E3 - Updating site user from SiteConfiguration page
```csharp
protected void Page_Load(object sender, EventArgs e)
{
    HyperLink4.Visible = false;

    string strCheckCredSql = "Select username, cred_ID from HP_Logins where username = " + Page.User.Identity.Name + ";";

    SqlConnection checkCredConn = new SqlConnection("Data Source=sql2k511.discountasp.net;Initial Catalog=SQL2005_4");
    checkCredConn.Open();
    SqlCommand cmdCheckCred = new SqlCommand(strCheckCredSql, checkCredConn);
    try
    {
        SqlDataReader rdrCheckCred = cmdCheckCred.ExecuteReader();
        while (rdrCheckCred.Read())
        {
            string strRdrFirstVal = rdrCheckCred[0].ToString();
            int strRdrVal = Convert.ToInt16(rdrCheckCred[1]);
        }
        if (strRdrVal == 3)
        {
            HyperLink4.Visible = true;
        }
    }
    catch (SqlException sqlEx)
    {
        string strCheckCredEx = sqlEx.ToString();
        checkCredConn.Close();
    }
    finally
    {
        checkCredConn.Close();
    }
}
```

**E 4 - Checking credentials at login on Index.aspx**
E 5 - Sample XML section of report definition language (rdl) file. This example demonstrates section including SQL command using parameters
References


(2) Braden, Caroline. HealthPoint representative tracking patient data. Personal interview. 10 Oct. 2007.

(3) Braden, Caroline. HealthPoint representative tracking patient data. Personal interview. 14 Nov. 2007.


http://www.northernkentuckyusa.com/AboutNorthernKentucky.aspx

(16) HealthPoint Family Care – Quality Healthcare for Everyone! HealthPoint Family Care.