WEB BASED MEDICAL INTERFACE AND MANAGEMENT TOOL

Sahithi Jammula
B.Tech, Nagarjuna University, 2007

PROJECT

Submitted in partial satisfaction of
the requirements for the degree of

MASTER OF SCIENCE

in

COMPUTER SCIENCE

at

CALIFORNIA STATE UNIVERSITY, SACRAMENTO

FALL
2011
WEB BASED MEDICAL INTERFACE AND MANAGEMENT TOOL

A Project

by

Sahithi Jammula

Approved by:

______________________________, Committee Chair
Jinsong Ouyang, Ph.D.

______________________________, Second Reader
Ahmed Salem, Ph.D.

______________________________
Date
Student: Sahithi Jammula

I certify that this student has met the requirements for format contained in the University format manual, and that this project is suitable for shelving in the Library and credit is to be awarded for the Project.

__________________________, Graduate Coordinator       ________________
Nick Faroughi, Ph.D.             Date

Department of Computer Science
Abstract

of

WEB BASED MEDICAL INTERFACE AND MANAGEMENT TOOL

by

Sahithi Jammula

Statement of Problem:

Currently there are very few websites, which provide all the information related to cosmetic treatments. At present times, we see a lot of new cosmetic treatment centers being opened and operated by a large number. Even with these facilities increasing, it still gets difficult for someone to get more information or specifics about the various options that are available to them. This project is specific to addressing this problem, where a user can get all the information related to treatment options in one single website. This project aims at developing a web-based system for the hospitals who offer natural therapy treatments. This system gives the users the ability to get more information about their treatment options and compare the different pricing options available.

Conclusions Reached:

This project is mainly designed keeping the end user in mind, providing easy navigation options and lots of information made readily available. This has also been developed keeping various hospital networks in mind. With this comes proper handling of different databases, making them secure and providing the user with their facility information.
Also it has the capability for doctors to access the patient information and being able to
directly communicate to them. The client “Enorbis” wanted to make sure this is a one-
stop information website and treatment center network that can make users really get all
the information needed in order to pick a right plan that’s suitable for them, This project
is developed keeping the customer interface as simple as possible. The user or the
administrator or even the doctors have the flexibility to navigate through different options
with the website, as easy it can be possible.

__________________________, Committee Chair
Jinsong Ouyang, Ph.D.

__________________________
Date
ACKNOWLEDGMENTS

I would like to thank all the people around me, who helped directly or indirectly to successfully complete this project.

Particularly, I want to acknowledge my supervisor, Dr. Jinsong Ouyang for his support and encouragement in making this project a success. His excellent suggestions and timely encouragement helped me complete the project and report.

Also, thanks to Dr. Ahmed Salem for his willingness to serve on the committee. I benefited very much from the professor’s course offerings and advice.

Lastly, I would like to thank the entire faculty and staff of the Computer Science Department, California State University, Sacramento.
# TABLE OF CONTENTS

| Acknowledgements                                                                 | vi  |
| List of Tables                                                                   | ix  |
| List of Figures                                                                  | x   |
| Software and Hardware Requirements                                               | xiii|

## Chapter

1. INTRODUCTION ........................................................................................................... 1
   1.1 Project Background .......................................................................................... 1
   1.2 Project Objective ......................................................................................... 2

2. REQUIREMENT ANALYSIS .......................................................................................... 4
   2.1 Project Perspective ........................................................................................ 4
   2.2 Functional Requirements and Use Cases ...................................................... 8
   2.3 Non Functional Requirements ........................................................................ 31

3. SYSTEM DESIGN ...................................................................................................... 34
   3.1 Building a Web Application .......................................................................... 34
   3.2 Web Application Architecture ...................................................................... 35
   3.3 High Level Design ......................................................................................... 37

4. DATABASE DESIGN ................................................................................................. 47
   4.1 About SQL SERVER 2008 ................................................................................ 47
4.2 Database Tables and Database Diagram .................................................................48

5. IMPLEMENTATION DETAILS ..................................................................................53
  5.1 Development of Web Interface .............................................................................53
  5.2 Storing and Retrieving Data using ADO.NET .......................................................62
  5.3 Implementation of Business Logic .......................................................................65

6. CONCLUSION ..............................................................................................................66

Appendix .User Guide .....................................................................................................67

Bibliography ....................................................................................................................90
<table>
<thead>
<tr>
<th>Table</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Table 2.1 Administrator Features</td>
<td>5</td>
</tr>
<tr>
<td>2. Table 2.2 Customer Features</td>
<td>6</td>
</tr>
<tr>
<td>3. Table 2.3 Partner Features</td>
<td>7</td>
</tr>
</tbody>
</table>
# LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>Figure 2.1 Use Case Notations</td>
<td>9</td>
</tr>
<tr>
<td>2.</td>
<td>Figure 2.2 User Authentication</td>
<td>10</td>
</tr>
<tr>
<td>3.</td>
<td>Figure 2.3 Administrator Module</td>
<td>11</td>
</tr>
<tr>
<td>4.</td>
<td>Figure 2.4 Organization Master</td>
<td>12</td>
</tr>
<tr>
<td>5.</td>
<td>Figure 2.5 Manage Partners</td>
<td>13</td>
</tr>
<tr>
<td>6.</td>
<td>Figure 2.6 Manage Users</td>
<td>15</td>
</tr>
<tr>
<td>7.</td>
<td>Figure 2.7 Manage Enquires</td>
<td>16</td>
</tr>
<tr>
<td>8.</td>
<td>Figure 2.8 Manage Quotations</td>
<td>17</td>
</tr>
<tr>
<td>9.</td>
<td>Figure 2.9 Customer Module</td>
<td>19</td>
</tr>
<tr>
<td>10.</td>
<td>Figure 2.10 Customer Registration</td>
<td>19</td>
</tr>
<tr>
<td>11.</td>
<td>Figure 2.11 User Profile Update</td>
<td>21</td>
</tr>
<tr>
<td>12.</td>
<td>Figure 2.12 Customer Enquiry</td>
<td>22</td>
</tr>
<tr>
<td>13.</td>
<td>Figure 2.13 Customer User Quotation</td>
<td>23</td>
</tr>
<tr>
<td>14.</td>
<td>Figure 2.14 Partner Module</td>
<td>25</td>
</tr>
<tr>
<td>15.</td>
<td>Figure 2.15 Partner Module Use Cases</td>
<td>25</td>
</tr>
<tr>
<td>16.</td>
<td>Figure 2.16 Partner Organization Profile</td>
<td>26</td>
</tr>
<tr>
<td>17.</td>
<td>Figure 2.17 Partner Account Update</td>
<td>27</td>
</tr>
<tr>
<td>18.</td>
<td>Figure 2.18 Partner Services</td>
<td>28</td>
</tr>
<tr>
<td>19.</td>
<td>Figure 2.19 Partner Enquiry Box</td>
<td>29</td>
</tr>
<tr>
<td>20.</td>
<td>Figure 2.20 Partner Quotations</td>
<td>30</td>
</tr>
</tbody>
</table>
21. Figure 3.1 Client-Server Architecture .............................................................34
22. Figure 3.2 Three-tier Architecture...................................................................35
23. Figure 3.3 High Level Design for the Medical Interface Management Tool........37
24. Figure 3.4 Basic Class Diagram ......................................................................39
25. Figure 3.5 Administrator Module Class Diagram ............................................40
26. Figure 3.6 Administrator Module Entity Class Diagram ..................................41
27. Figure 3.7 Customer Module Class Diagram ...................................................42
28. Figure 3.8 Customer Module Entity Class Diagram .........................................43
29. Figure 3.9 Partner Module Class Diagram .......................................................45
30. Figure 3.10 Partner Module Entity Diagram ....................................................46
31. Figure 4.1 Database Diagram ........................................................................50
32. Figure 7.1 Administrator Login Page ...............................................................67
33. Figure 7.2 Customer Login Page ....................................................................68
34. Figure 7.3 Partner Login Page ..........................................................................69
35. Figure 7.4 New User Registration Form ............................................................70
36. Figure 7.5 Update Customer Profile .................................................................71
37. Figure 7.6 Customer Enquires .........................................................................73
38. Figure 7.7 Attach Case Sheet ...........................................................................74
39. Figure 7.8 Choose File to Upload ....................................................................74
40. Figure 7.9 Customer Quotations .....................................................................76
41. Figure 7.10 Organization Master .....................................................................78
42. Figure 7.12 Manage Partners .........................................................................79
43. Figure 7.13 Add a New User to the Partner ..................................................... 80
44. Figure 7.14 Manage Users .......................................................................... 81
45. Figure 7.15 Partner Organization Profile ..................................................... 82
46. Figure 7.16 Update Partner Profile ............................................................... 83
47. Figure 7.17 Partner Hospital Services ......................................................... 84
48. Figure 7.18 Update and Register Hotel Services ......................................... 85
49. Figure 7.19 Update and Register Travel Services ........................................ 86
50. Figure 7.20 Partner Enquiry ........................................................................ 87
51. Figure 7.21 View the Customer Profile ...................................................... 88
52. Figure 7.22 Partner Quotations .................................................................. 89
53. Figure 7.23 View of the Enquiry Information from the Quotation ............ 89
HARDWARE AND SOFTWARE SPECIFICATIONS

The software and hardware requirements for this project are defined in this section.

Software Requirements

- Operating System WINDOWS NT 4.0 | 2000 | 9.X | VISTA | XP or higher
- Microsoft .Net Framework 3.5 (Minimal Requirement for deploying application on Client Machine)
- Internet Information Services (IIS)
- Microsoft SQL Server 2008
- Programming Languages Used – C#.NET, ASP.NET, Java script

Hardware Requirements

- Pentium-IV Processor with 500MHZ or above
- 1GB of Free Hard disk space
- 512MB of RAM
- LAN Network (For Remote Sources)
- Network interface card or Modem (For Remote Access)
Chapter 1

INTRODUCTION

1.1 Project Background

This project aims at developing a web based management system for the hospitals who offer cosmetic and natural therapy treatments around the world. We see a lot new cosmetic treatment centers being opened and operated by a large number. Even with these facilities increasing, it still gets difficult for someone to get more information or specifics about the various options that are available to them. It also helps the user understand the different treatment options available to them. This project also aims at bringing awareness among the users, about the pricing options available to them and plan their treatment cost efficiently. Since these facilities are distributed across the globe, the user can choose to have extended stay, make the necessary accommodation, and travel arrangements.

The goal is to make the interface more customers friendly keeping the end user in mind, providing easy navigation options and lots of information made readily available. There are websites currently that provide some level of features; but this particular interface is user friendly and has every piece of information that the client can possibly think off in one single place. This project has also been developed keeping various hospital networks in mind and is customized for their services. In addition to the interface, the different databases are securely handled providing the user with their facility information.
1.2 Project Objective

This project is for client named “Enorbis”, and is being designed based on the client’s business model to achieve maintainability, reliability and usability.

Create easily maintainable system: “Maintainability is defined as the probability of performing a successful repair action within a given time. In other words, maintainability measures the ease and speed with which a system can be restored to operational status after a failure occurs [10].” The organization wishes to create a system that can handle natural and cosmetic treatments from their hospital allies and make them available to the end users. They also want make sure the system is as maintainable as possible.

Improve client functionality: It requires the system to give end users the capability to better communicate with service providers and request all the information that will help make their decision. This system will also provide the ability to make accommodation arrangements according to their scheduled travel.

Usability: The system should be easy to use and flow with the information. And also the interface should be something easily adaptable with minimum training efforts.

Some of the major tasks are to develop an interface for three different types of end users.

- Options to choose different types of treatments that the medical network provides.
- Ability to choose different doctor's based of patient preference.
- Get an initial quote of the medical charges so they can plan financially.
• Being able to input their prior medical history, which helps the doctor to review the case in advance.

• Reserve appointment with the doctor's once the doctor has completed the medical record checks.

• Search for accommodation (Hotel) information for the people who come along with the patients.

• Reserve rental cars using the same portal or even transportation facilities.

• Check for any tourist destinations that people want to visit during their stay.

• Create a secure login and password for every customer.

• Incase if the patient has to travel from different country, this website provides all the necessary information that the user can use to make the travel arrangements.
Chapter 2

REQUIREMENT ANALYSIS

2.1 Project Perspective

The Project aims at completing the requirements from the client “Enorbis” and to create a real time application to support their business model. The model of this project is to develop a system that support three different end users, and provide the project with a full functionality of a management tool. This tool ensures communication between the modules.

The modules are defined based on end user roles, each module has its independent features and all the transactions made are recorded with the help of a database. A user session will be maintained for each user logged into the system. Each user role is defined as follows:

Administrator: The Administrator is one who oversees the operations of this website. For the administrator module, this project aims in providing a secure login. It also provides the administrator, ability to manage and control its functions among its partners and customers. All the features for the Administrator are depicted in Table [2.1].

Customers: The end users using this system for planning their natural treatments will be considered as the customers. The customer is the main focus of the application and will provide the customer the ability to wise plan his treatment options. All the features supported by the customer module are depicted in the Table[ 2.2].
Partners: The end users who want to provide their services through this system are considered as the partners. The partners are also referred to as the service providers. The project is designed to support different types of end users like hospitals, hotels, and travel services. The features this project provides for the partner modules are defined in Table [2.3].

Table 2.1: Administrator Features

<table>
<thead>
<tr>
<th>Feature Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Manage Users</td>
<td>The administrator needs to be provided with the access to manage the user accounts (for making them active or inactive)</td>
</tr>
<tr>
<td>Manage Partners</td>
<td>The administrator needs to be provided with the access to manage the partner accounts</td>
</tr>
<tr>
<td>Manage Enquires</td>
<td>The administrator must be provided access to manage and view enquires posted by the users</td>
</tr>
<tr>
<td>Manage Quotations</td>
<td>The administrator must be provided with access to manage and view quotations submitted by the partners</td>
</tr>
<tr>
<td>Feature Item</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>User Registration</td>
<td>New user registration form needs to be filled out for registering with system.</td>
</tr>
<tr>
<td>Registration Approval</td>
<td>Upon Successful registration, the registered user needs to be approved by administrator.</td>
</tr>
<tr>
<td>User Login</td>
<td>The user login account must be provided to login into the portal.</td>
</tr>
<tr>
<td>Update Profile</td>
<td>The user needs to setup his profile.</td>
</tr>
<tr>
<td>Post Enquiry</td>
<td>The user will be provided with a portal to post enquiries related to hospital, hotel and travel services</td>
</tr>
<tr>
<td>Update Enquiry</td>
<td>The user must be provided with the provision to update the enquiry posted</td>
</tr>
<tr>
<td>View Quotations</td>
<td>The user must be provided to view the quotations submitted by the service providers for the enquiries (posted by user)</td>
</tr>
<tr>
<td>Confirmation of the Quotation</td>
<td>The user must be provided to select a quotation and confirm the quotation</td>
</tr>
</tbody>
</table>
Table 2.3: Partner Features

<table>
<thead>
<tr>
<th>Feature Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Update Profile</td>
<td>The Partner must be provided with an ability to update his profile and contact information.</td>
</tr>
<tr>
<td>Register/ Update and Delete</td>
<td>The partner organization must be provided with options to update his hospital services.</td>
</tr>
<tr>
<td>Hospital Services</td>
<td></td>
</tr>
<tr>
<td>Register/ Update and Delete</td>
<td>The partner organization must be provided with options to update his hotel services.</td>
</tr>
<tr>
<td>Hotel Services</td>
<td></td>
</tr>
<tr>
<td>Register/ Update and Delete</td>
<td>The partner organization must be provided with options to organize his Travel services</td>
</tr>
<tr>
<td>Travel Services</td>
<td></td>
</tr>
<tr>
<td>Register/ Update and Delete</td>
<td>The partner organization must be provided with options to organize his travel services</td>
</tr>
<tr>
<td>Travel Services</td>
<td></td>
</tr>
<tr>
<td>View Enquires</td>
<td>The partner organization need to be with support to view enquires posted</td>
</tr>
<tr>
<td>Submit Quotation</td>
<td>The partner need to be able to submit quotation for the enquiry posted to him. The quotation will have complete information regarding the cost and available dates in the hospitals.</td>
</tr>
</tbody>
</table>
2.2 Functional Requirements and Use Cases

The behavior of the system under different circumstances is defined as functional requirements. The functional requirements lay a foundation for the system design and implementation steps of a project. Use Case modeling is used to illustrate the behavioral scenarios of the functional requirements.

Use Case Modeling:

Use case modeling is an abstraction of the system used to describe what the new software being developed should do. It is used in the analysis phase of software development to express projections of the high-level requirements of the product. Analysis of the project using use cases helps the developer to, captures the actual requirements of the customer. The use case-modeling diagram is a use case and actor relationship, it is used to define the functionality of the system.

Actor: An actor is an element used in the use case. An actor is someone external to the system. It is someone or something, which interacts with the system. There are different kinds of relationships that exist between use cases and actors [5].

The following notations and relations are used in the use case modeling; the notations in Figure 2.1 will be used through the rest of the document.
The following are the Use Cases for each of the features and tasks specified in the above Tables [2.1 to 2.3]

2.2.1 User Authentication

This describes how can log into the medical system. The three different users will be provided with three different links to the login page as depicted in the following figure.
Preconditions

The following pre-conditions must be true to initiate this use case.

1. User has logged into the Internet.

2. The web based system has to validate user credentials entered in the login Page.

Flow Description

The following steps describe the flow for the login page:

1. User selects the appropriate link from the website home page.

2. System determines user's role from login.

3. System displays appropriate websites page.

4. The system also displays the username in the top left corner of the webpage.

5. The side with the appropriate links is provided to user interacting with system.

Post Conditions

The following post conditions must be true after the completion of the use case.
1. User has logged into the application.
2. User is able to view/select various applications depending on role.
3. The user will be able to log out of the system.

### 2.2.2 Administrator

When the administrator successfully logs into the system he is provided with the options defined in the following use cases, also depicted in the following figure.

1. Organization Master
2. Manage Partners
3. Manage Users
4. Manage Enquiries
5. Manage Quotations

![Figure 2.3: Administrator Module](image-url)
2.2.3 Organization Master

This Use Case describes how the administrator can manage all service providers.

Preconditions

The following pre-condition must be true to initiate this use case.

1. User has logged into the website with administrative credentials.

Flow Description:

The following steps describe the flow in the login page:

1. The Organization Master page gets displayed with the grid view of all service providers.

2. The Administrator will have option to select the service provider from the grid and will be able to view the details.
3. The selection of the administrator from the grid will auto populate respective fields.

4. The Administrator will have the option to add, edit and delete a service provider.

5. All the Add, edit and delete operations will be validated on the client side. If the validations fail the webpage will pop up a message box showing the failures.

6. Also has the link to manage users, enquires and also manage partner enquires.

**Post Conditions**

The following post conditions must be true after the completion of the use case.

1. The Administrator will be able to logout of the application.

2. User is able to view/select other links available.

### 2.2.4 Manage Partners

The manage partners use case describes how login accounts are created for different partners.

![Diagram of Manage Partners](attachment:image.png)

**Figure 2.5: Manage Partners**
The Administrator will be able to use this functionality by selecting the Manage Partners from the side menu.

**Preconditions**

The following pre-conditions must be true to initiate this use case.

1. User should be logged in the system, with validated credentials.
2. The user should select the tab for manage partners.

**Flow Description**

The following steps describe the follow for the login page:

1. The administrator will have the grid view of all the partners in the system.
2. The administrator will have the ability to add new user accounts to the partners.
3. The administrator has access to change the status of the partner. He can set the active bit on the application.

**Post Conditions**

The following post conditions must be true after the completion of the use case.

1. The admin has the option to logout of the application
2. The admin has the option to check out other tabs.

**2.2.5 Manage Users**

This user case helps the administrator, to view and manage the users registered with the system.
**Preconditions**

The following preconditions must be true to initiate this use case.

1. User should be logged in the system, with validated credentials.
2. The user should select the tab for manage partners.

**Flow Description:**

The following steps describe the flow for the use case:

1. The Administrator will have the grid view of all the users registered with the system.
2. The administrator will be able to select a user and will be able to see the user registration profile.
3. The admin will also have the access to update or delete the user profile. The edits to user profile will be validated before writing to database.

**Post Conditions**

The following post conditions must be true after the completion of the use case.
1. The admin has the option to logout of the application.

2. The admin has the option to check out other tabs.

**2.2.6 Manage Enquiries**

This use case enable the administrator to manage the user enquires.

![Diagram of Manage Enquiries]

**Figure 2.7 Manage Enquires**

**Preconditions**

The following pre-conditions must be true to initiate this use case.

1. User should be logged in the system, with validated credentials.

2. The user should select the tab for manage enquires.

**Flow Description:**

The following steps describe the flow for the use:

1. The Administrator will have the grid view of all the enquires posted to the system.

2. The admin will be able select and view the details of the enquires posted.

3. The admin has access to case sheets posted.
4. The details of the selected (hospital, hotel and travel) enquires will displayed in separate tabs.

5. The user will also be able to mark them as spam.

**Post Conditions**

The following post conditions must be true after the completion of the use case.

1. The admin has the option to logout of the application.
2. The admin has the option to check out other tabs.

**2.2.7 Manage Quotations**

This use case enables the administrator to manage quotations posted by the partners.

![Diagram of Manage Quotations](image)

**Preconditions**

The following pre-conditions that must be true to initiate this use case.

1. User should be logged in the system, with validated credentials.
2. The user should select the tab for manage quotations.
Flow Description:

The following steps describe the flow for the use case:

1. The Administrator will have the grid view of all the quotations posted to the system.
2. The admin will be able select and view the details of the quotations posted.
3. The admin has access to case sheets posted.
4. The details of the selected (hospital, hotel and travel) quotation will displayed in separate tabs.
5. The user will also be able to mark them as spam.
6. The quotation will include details of expected budget amount for the treatment.

Post Conditions

The following post conditions must be true after the completion of the use case.

1. The admin has the option to logout of the application.
2. The admin has the option to check out other tabs.

2.2.8 Customer

When the customer uses the system he can successfully log in to the system only if he is a existing user. But if the user is a new user, then he can create a new profile. The use cases will defined as follows and depicted in the figure.

1. New Customer Account and registration
2. View /Update user profile.
3. Post Enquires to the Partners.
4. View/Confirm Quotations.

2.2.9 Customer Registration

This is for the customers who want to use the services of the system to make medical appointments, schedule hotel and travel reservation.

Preconditions

The following pre-conditions that must be true to initiate this use case.
1. Should be a new user to the system.

Flow Description:

The following steps describe the flow for the user page:

1. The user will fill in the online registration form to register with the system.
2. The form contains all the details about the user credentials.
3. The user details also include the username and password for the system.
4. Once the user registers and hits “save”, popup window shows the validation failures.
5. If there are no validation failures, the user information gets saved and the page gets redirected to the login page.

Post Conditions

The following post conditions must be true after the completion of the use case.

1. The user can log in to the application with the user credentials.

2.2.10 User Profile Update

Based on the customer’s preference, the customer can update his profile when he logs into the application.
Preconditions

The following preconditions that must be true to initiate this use case.

1. Should be logged into system successfully

Flow Description:

The following steps describe the flow for the login page:

1. After logging into the system the user will be able to see all his profile information.

2. This page contains all the details about his credentials

3. The user details also include the username and password for the system.

4. The user can update his profile and also save the changes.

5. Once the user clicks on save all the updated properties go through validations.

Post Conditions

The following post conditions must be true after the completion of the use case.

1. The user can log out of the application

2. The user will be able to proceed with posting enquires and requesting quotations.
2.2.11 Enquiry

The customers who wanted to post enquires regarding the hospitals, hotels, vehicle booking can use this page.

![Diagram of Enquiry Process]

Figure 2.12: Customer Enquiry

**Preconditions**

The following pre-conditions that must be true to initiate this use case.

1. Should be an existing user of the system.
2. Should be logged into system successfully.
3. The user should select the link to access enquires from the side panel.

**Flow Description:**

The following steps describe the flow for the login page:

1. After logging into the system the user will have the grid view of all the hospitals, hotels, travel service posted in three different tabs.
2. The will be three different tabs to post enquires based on the type of services.
3. The user will have the option to select the service provider and post enquire regarding services.

4. The user can upload the case sheet for the reference by the hospital.

5. The user can also update the previously posted enquires.

6. For this use case each event click is validated, and any error will show up in the message box.

**Post Conditions**

The following post conditions must be true after the completion of the use case.

1. The user can log out of the application.

2. The user will be able to proceed with posting enquires and requesting quotations.

**2.2.12 User Quotations**

The customers who wanted to view quotations regarding the hospitals, hotels and vehicle booking can use this page. The flow is depicted by the figure.

![Figure 2.13: Customer User Quotation](image-url)
Preconditions

The following pre-conditions that must be true to initiate this use case.

1. Should be an existing user of the system.
2. Should be logged into system successfully.
3. The user should select the link to access quotations.

Flow Description:

The following steps describe the flow for the login page:

1. After logging into the system the user will have the grid view of all quotations posted in different tabs based on the type (hospital, hotel and travel service).
2. The will be three different tabs to which the quotations get posted based on the type of services.
3. The user can select and view the quotations posted to him.

Post Conditions

The following post conditions must be true after the completion of the use case.

1. The user can approve the quotation.
2. The user can log out of the application.

2.2.13 Partner

A partner can successfully log in to the system if he is a existing user. But if he is a new user he will need account access from the administrator. The profile detail and pages vary according to the type of the partner. The following features will be provided to the module.
The use cases for each module is depicted as follows.

1. Organization Profile
2. View /Update user accounts.
3. Upload Partner services.
4. View enquires from the customer.
5. Post Quotations to the customer.
2.2.14 Partners Organization Profile

This use case helps the partner to access the organization profile.

![Diagram of Partner Organization Profile]

Figure 2.16: Partner Organization Profile

**Preconditions**

The following pre-conditions must be true to initiate this use case.

1. The partners should be logged in with a valid username and password.

**Flow Description**

The following steps describe features for the partner organization profile:

1. All the fields in the profile are auto populated.
2. The logged in user can make updates to the organization.
3. On click of the save all the fields are validated and written to the database.

**Post Conditions**

The following post conditions must be true after the completion of the use case.

1. The partner can log out of the application.
2. The partner can choose other options available.
2.2.15 Update Partner User Accounts

There can be multiple users from a particular partner. The logged user has the ability to update the passwords from his user account.

Preconditions
The following pre-conditions must be true to initiate this use case.

1. The partners should be logged in with a valid username and password.

Flow Description

1. After logging into the system the user will be able to see all his profile information.
2. This page contains all the details about his credentials
3. The user can update account information and also save the changes.
4. Once the user clicks on save all the updated passwords go through validations.

Post Conditions
The following post conditions must be true after the completion of the use case.
1. The partner can log out of the application

2. The partner can select from other available options.

### 2.2.16 Upload Partner Service

![Diagram of Partner Services]

**Preconditions**

The following pre-conditions must be true to initiate this use case.

1. The Partner must be logged into the system with valid credentials.

2. The Partner must make a selection to view the services.

**Flow Description**

The flow is described as follows for this use case:

1. The user has the grid view of the existing services in the system.

2. On Click of any selection from the grid, the fields in the forms are auto populated.

3. Additional new services can be added to the system or update existing service.
4. Each service is validated on the client side before writing to database.

**Post Conditions**

The following post conditions must be true after the completion of the use case.

1. The partner can logout of the application.
2. The partner can make other selection from the application.

### 2.2.17 Partners Enquiry Box

The enquires from the customers gets posted to corresponding partners.

![Diagram of Partner Enquiry Box](image-url)

**Figure 2.19: Partner Enquiry Box**

**Preconditions**

The Following pre-conditions must be true to initiate this use case.

**Flow Description**

The following steps describe the flow for the partner enquiry box:

1. All the enquires to the partner are posted to his enquiry box in the grid view.
2. The partner can click on the enquiry and view its details.
3. The partner does not have permission to edit the enquires posted to him.

4. The partner can respond to the enquires with the quotations with button click on the bottom of the enquiry page.

Post Conditions

The following post conditions must be true after the completion of the use case.

1. The partner can log out of the application.

2. The partner can choose other options available.

2.2.18 Partners Quotations

The quotations posted to the customers gets posted to corresponding partners.

![Figure 2.19: Partner Quotations](image)

Preconditions

The Following pre-conditions must be true to initiate this use case.
**Flow Description**

The following steps describe the flow for the partner enquiry box:

1. All the quotation responses by the partner gets posted to his grid view.
2. The partner can click on the quotation and view its details.
3. The partner can update an existing quotation.
4. The partner has a link to view the enquiry for the corresponding quotations.

**Post Conditions**

The following post conditions must be true after the completion of the use case.

1. The partner can log out of the application.
2. The partner can choose other options available.

**2.3 Non Functional Requirements**

The nonfunctional requirements define the operation of the system rather than behavior of the system. They define the governing factors of implementation platform like operating and development environment. The system is implemented using Microsoft ASP.NET technology, which is based on the CLR class libraries and tools that are integrated into the .NET Framework. IIS (Internet information service) needs to be enabled for the development of the application.

Common Language Runtime: The Common Language Runtime (CLR) is the virtual machine component of Microsoft’s .NET framework and is responsible for managing the execution of .NET programs. In a process known as just-in-time (JIT) compilation,
CLR compiles the intermediate language code known as CIL into the machine instructions that in turn are executed by the computer's CPU. The CLR provides additional services including memory management, type safety and exception handling. All programs written for the .NET framework, regardless of programming language, are executed by the CLR [1].

Class Libraries: A class library is a collection of prewritten classes or coded templates, any of which can be specified and used by a programmer when developing an application program. Each class as an object that can be called when the program is executed. Access to and use of a class library greatly simplifies the workload of development.

ASP.Net Server Controls and browser capabilities: ASP.NET determines browser capabilities by reading the user-agent information that is passed from the browser to the server during a request. Once ASP.NET finds a match between the user-agent strings it loads the corresponding browser capabilities into the HttpBrowserCapabilities object. Based on these capabilities, the controls on the page render Web controls using appropriate markup[1].

SQL Server 2008: The system will also require the SQL database installed in the host space, as well as any additional software configuration and drivers required sending email to users of the system. SQL is closely integrated into Visual Studio and it supports all kinds of business data with native support for relational data, xml and file stream.
server provides support for stored procedures, triggers and view that are used to retrieve the data from the database.

Security Requirements: Passwords shall be displayed as “*” in the web pages wherever required. Proper authentication is required for users to access any of the web pages including the home page. Every user of the system is assigned a unique login and password to access the application over the internet.

Assumptions and Dependencies: It is assumed that the system will be developed using the ASP.NET technology. The is assumed to interact with an email server in order send conformation emails.
Chapter 3
SYSTEM DESIGN

This project is designed to be used in a client-server model. In the client-server model the server is service provider it can be a (ftp server, web server or an email server). This application needs hosted by a web server. A web server is a program located on a server to process requests made by a client. This project runs on MS-IIS (Microsoft Internet Information Services) Web Server. IIS is a web server application, which is a set of feature extension modules created by Microsoft for use with Microsoft Windows.

3.1 Building a Web Application

Web application is a collection of all files, pages, handlers, modules, and executable code that can be invoked from a given directory on a web server. A web application is usually hosted in a browser-controlled environment. The web applications are scalable and maintainable without distributing and installing software on each of the client machine.

Figure 3.1: Client-Server Architecture
3.2 Web Application Architecture

In three-tier design of a .NET application, the architecture breaks down into three layers based on functionality:

1. The data layer manages the storage and retrieval of data from the data store.
2. The business layer maintains business rules and logic.
3. The presentation layer has the user interface and any user process components.

Figure 3.2: Three-tier Architecture [6]
Presentation Layer: It contains the components that are required to enable user interaction with the application. The user process components are defined for complex user interaction to store the entities and objects. The code behind mechanism helps the developers to create and update their applications easily.

Business Layer: A business object is a component that encapsulates the data and business processing logic for a particular business entity. Business tier contains logic for retrieving persistent data from the data-tier and placing it into business objects[4].

Data Access layer: The logic to access the data from the database is encapsulated in the data access layer. This layer consists of components that access the data from tables that are defined in a database, stored procedures and views that allow manipulating data as it goes into and out of those tables.

Since the three tiers are independent of each other, it makes the system independent and easily maintainable. I will also be a scalable and flexible system based on the deployment of the layers independently. The following is a block diagram shows all the modules in the application and the relationships between them.
3.3 High Level Design

The High Level Design covers the overall design to the solution. The high-level design is depicted using the following.

- The class model and relationships
- The sequence diagrams which outline key use case scenarios
- The data/object model with relational table design
- User interface style and design
This creates analysis between the requirements and architecture interface. This process includes the decomposition of system requirements and detailed analysis of each requirement.

The purpose of the high-level design is to define key classes, as well as the interactions among instances of these classes (objects). Inputs to the HLD consist of the functional requirements defined in the above sections.

Following are the key goals and constraints governing the design decisions:

- High-level design should decompose complex business logic and identify dependencies among components.
- High-level design should address core functionality by identifying and defining the key classes, their responsibilities, behavior and attributes.
- High-level design should form the basis for planning and scoping subsequent phases of the project.
- High-level design should adopt object-oriented design principles such as encapsulation, information hiding, loose coupling, and high cohesion.

The following figure depicts the basic class diagram, which gives an overview of the different types of end users using the system.
The loading page of the web application, which is the login page, is designed separately for each user role. The login page drives the flow of the application based on the user role. Individual features will be defined for each role of the module and their properties vary accordingly.

**Administrator Module**

The following class diagram Figure 3.5 shows all the classes and methods defined for this module. The entity objects used to read and write the data from the database are depicted in Figure 3.6. It also depicts the properties of each object defined. When the user logins as administrator, it will be able to choose from features and will be able to view all the organizations registered with system. The administrator access is powerful since it has access to all the user information. The restrictions are applied on the administrator module; the administrator has no permission to edit enquiries or quotations. Based on the requirement the administrator will be able to access change the update organization information. The admin only has the ability to mark any suspicious enquiries as spam.
Figure 3.5: Administrator Module Class Diagram
Figure 3.6: Administrator Module Entity Class Diagram
Customer Module

The following class diagram Figure 3.8, 3.9 shows all the methods and class relationships between the webpages. When the customer logs in to system, the customer will be able to post different enquiry to the different service providers. A dropdown list of all the services from service providers will be auto populated. The properties of each object are passed from the presentation layer to the database.

Figure 3.7: Customer Module Class Diagram
Figure 3.8: Customer Module Entity Class Diagram
Partner Module

The following class diagram Figure 3.10 show all the methods and class relationships between the webpages. The entity objects used to read and write the data from the database are shown in Figure 3.11. The partners are categorized into three different types (Hospital, Hotel, Travel) based on the organization types. When the Partner logs in to system, will auto detect the type organization the partner belong to and will auto populate the organization details. Based on the user roles the properties of each end-user will vary and are depicted in the class diagram. The following class diagram show all the entity objects and relationships between the objects. It also depicts the properties and of each object.
Figure 3.9: Partner Module class diagram
Figure 3.10: Partner Module Entity diagram.
Chapter 4

DATABASE DESIGN

Database design is an important component of the application design. A good database design will lay a solid foundation to the application development process. Database that is designed according to the relation model will have efficient performance and are easy to modify. The database design plays crucial impact on performance of an application.

The basics behind a relational model are that, a database consisting of a series of unordered tables or relations can be manipulated using non-procedural operations on the tables. The non-procedural operations will help following operations [9]:

- Data entry, updates and deletions will be efficient.
- Data retrieval, summarization and reporting will also be efficient.
- Since the database follows a well-formulated model, it behaves predictably.
- Since much of the information is stored in the database rather than in the application, the database is more self-documenting.
- Changes to the database schematic are easy to make.

4.1 About SQL Server 2008

Microsoft SQL Server is a relational model database server produced by Microsoft. Instead storing the raw data into the database, the tables are normalized with the primary and foreign key relationships. Transact-SQL is the primary procedural programming
language that uses the SQL Server. Any application that communicates with a SQL Server instance sends requests on form of transact sql statement.

Transact-SQL also offers efficient solutions to transaction processing and error handling that is if an error occurs while processing a transaction, a rollback of the transaction can be handled in the stored procedure call. The ability to write stored procedures and use compiled procedures dramatically improves the performance of SQL statements and batches. The stored procedures can be defined as precompiled sql statements. For relational data, T-SQL has been augmented with error handling features and support for recursive queries and expressions [3].

4.2 Database Tables and Database Diagram
Considering the project requirements, the tables for the database are designed to minimize redundancy. There are 17 tables in the database. This kind of a structure facilitates CASCADE DELETE and avoids any duplicate entries. The tables have also been designed to adhere to the Fourth Normal Form by using a systematic way ensuring that the database structure is suitable for general purpose querying.

Table Organization Master and its dependencies:
All the service providers registering with the system will be saved to the [tbl_OrganizationMaster]. The type of organization will be defined by the organization_type_code in the [tbl_organization_type] table; this table is used to populate the dropdown list dynamically.
Table ServiceMaster and its dependencies:
Each organization will have different services to offer. Every service that is created by the service provider will have a foreign key relation with the org_code from the table [tbl_OganizationMaster]. Since the data fields are different for each type organization, the customized service information is stored in [tbl_HospitalServices]; [tbl_HotelServices] and [tbl_TravelServices].

Table Enquiry Master and its dependencies:
Each enquiry is associated to a service_id in the [tbl_service_master] table with a foreign key relation. The [tbl_Enquirymaster] is the main table having the generic information with [enquiry_code] as the primary key and [organization_type_code] is the foreign key relating it to [tbl_organization_types]. Since the data fields are different for each type organization, the customized enquiry information is stored in [tbl_HospitalEnquiry]; [tbl_HotelEnquiry] and [tbl_TravelEnquiry]. The custom tables have the primary key and foreign key relation with the enquiry code. A response to an enquiry is considered as a quotation; in the quotation tables, the all entries have a foreign key association with the enquiry_code.

Additional functionality can be easily added to the system. These functionalities are discussed in the Future Scope section of the document. The following figure shows all the tables in the database and the relationships between them.
Figure 4.1: Database Diagram
The following are some of the table definition scripts

OrganizationMaster

CREATE TABLE [dbo].[tbl_OrganisationMaster](
    [Org_Code] [varchar](50) NOT NULL as primary key,
    [Org_Name] [char](200) NOT NULL,
    [Description] [varchar](200) NULL,
    [Org_Type_Code] [varchar](50) NOT NULL,
    [Address] [varchar](500) NOT NULL,
    [Phone_No1] [char](20) NOT NULL,
    [Phone_No2] [char](20) NULL,
    [Mail_Id] [varchar](200) NOT NULL,
    [Contact_Person] [varchar](200) NOT NULL,
    [Web_site] [varchar](200) NULL,
    [Active] [bit] NOT NULL,
    [Created_By] [varchar](50) NOT NULL,
    [Updated_By] [varchar](100) NULL,
    [Updated_Date] [datetime] NULL,
    [Is_Deleted] [bit] NOT NULL,
    PRIMARY KEY CLUSTERED
    (    [Org_Code] ASC
    )WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
    ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]

ALTER TABLE [dbo].[tbl_OrganisationMaster]  WITH CHECK ADD FOREIGN
KEY([Org_Type_Code])
REFERENCES [dbo].[tbl_OrgTypeMaster] ([Org_Type_Code])
GO

Service Master

CREATE TABLE [dbo].[tbl_ServicesMaster](
    [Service_Id] [varchar](50) NOT NULL,
    [Service_Name] [varchar](100) NOT NULL,
    [Org_Code] [varchar](50) NOT NULL,
    [Service_Description] [varchar](300) NOT NULL,
    [Created_By] [varchar](50) NOT NULL,
    [Created_Date] [datetime] NOT NULL,
    [Updated_By] [varchar](100) NULL,
    [Updated_Date] [datetime] NULL,
    [Is_Deleted] [bit] NOT NULL,
    PRIMARY KEY CLUSTERED
    (    [Service_Id] ASC
    )WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
    ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]

ALTER TABLE [dbo].[tbl_ServicesMaster]  WITH CHECK ADD FOREIGN
KEY([Org_Code])
REFERENCES [dbo].[tbl_OrganisationMaster] ([Org_Code])
GO
Enquiry Master:

CREATE TABLE [dbo].[tbl_EnquiryMaster](
    [Enquiry_Code] [varchar](50) NOT NULL,
    [User_Name] [varchar](50) NOT NULL,
    [Org_Type_Code] [varchar](50) NOT NULL,
    [Enquiry_Date] [datetime] NOT NULL,
    [Description] [varchar](500) NOT NULL,
    [Enquiry_Status] [varchar](20) NOT NULL,
    [Created_Date] [datetime] NOT NULL,
    [Updated_By] [varchar](100) NULL,
    [Updated_Date] [datetime] NULL,
    [Is_Deleted] [bit] NOT NULL,
    PRIMARY KEY CLUSTERED
    (        [Enquiry_Code] ASC
    )WITH (PAD_INDEX  = OFF, STATISTICS_NORECOMPUTE  = OFF, IGNORE_DUP_KEY = OFF,
    ALLOW_ROW_LOCKS  = ON, ALLOW_PAGE_LOCKS  = ON) ON [PRIMARY]
) ON [PRIMARY]

ALTER TABLE [dbo].[tbl_EnquiryMaster]  WITH CHECK ADD FOREIGN
KEY([Org_Type_Code])
REFERENCES [dbo].[tbl_OrgTypeMaster] ([Org_Type_Code])
GO
Chapter 5

IMPLEMENTATION DETAILS

This chapter describes in detail the .NET technologies used in this project and shows some sample code to explain in detail, the main features of the application. There are different features provided by the Medical interface management tool, the features are categorized under three different end-users.

The major implementation details are list as follows

- Loading the information to the Grid from the database
- Auto populating the drop down list.
- JavaScript Validations on each page
- New/ update and delete events.
- Adding the case sheet of the customer.
- Access the enquiry info in a new window.
- Adding additional row to the grid for Hotel service information.
- Use of CSS style sheets for maintaining uniform interface.
- Referencing the external dll for the side panel.

5.1 Development of Web Interface

ASP.NET is web application framework that allows programmers to build dynamic web sites, web applications and web services. ASP.NET provides a wide range of controls that can used to build Web applications. Web applications are popular due to the ubiquity of
web browsers, and the convenience of using a web browser as a client, sometimes called a thin client. The ability to update and maintain web applications without distributing and installing software on potentially thousands of client computers is a key reason for their popularity, as is the inherent support for cross-platform compatibility.

Advantages using ASP.NET

1. ASP.NET is easy to use because of the auto generation of the source code in HTML. In addition, there is easy integration of with JavaScript frameworks.

2. ASP.NET drastically reduces the amount of code required to build large applications.

3. The source code is compiled the first time the page is requested. Execution is fast as the Web Server compiles the page the first time it is requested. The server saves the compiled version of the page for use next time the page is requested.

4. ASP.NET makes for easy deployment. There is no need to register components because the configuration information is built-in.

5. The Web server continuously monitors the pages, components and applications running on it. If it notices any memory leaks, infinite loops, other illegal activities, it immediately destroys those activities and restarts itself.

6. Asp.Net easily works with ADO.NET using data-binding and page formatting features. All the database connections of the application will not affect the performance. The following code depicts how data grid is defined in webpage [7].
5.1.1 Implementation of creation and loading the data grid

For the implementation of the data grid is dragged and dropped from the toolbox to the design page. The binding is specified in the aspx page using the Data Binder expression. The data binder expression using eval helps bind the read only elements. In addition, the css style sheet helps enhance the display of by alternating the colors on grid. The following is the code depicts how the data binding expressions are defined.

```html
<div class="divstyles" style="WIDTH: 600px">
<asp:datagrid id="dgViewData" runat="server" AutoGenerateColumns="False" Width="100%">
  <SelectedItemStyle ForeColor="White"></SelectedItemStyle>
  <AlternatingItemStyle CssClass="alternateitems"></AlternatingItemStyle>
  <ItemStyle CssClass="itemstyle"></ItemStyle>
  <HeaderStyle CssClass="gridheader"></HeaderStyle>
  <Columns>
    <asp:BoundColumn Visible="False" DataField="Org_Code" HeaderText="Org_Code"></asp:BoundColumn>
    <asp:TemplateColumn HeaderText="Organisation Name">
      <ItemTemplate>
        <asp:LinkButton id="lbtnSelect" runat="server" CssClass="linktext" CommandName="Select">
          <%# DataBinder.Eval(Container.DataItem, "Org_Name") %>
        </asp:LinkButton>
      </ItemTemplate>
    </asp:TemplateColumn>
    <asp:BoundColumn DataField="Org_Type_Name" HeaderText="Type"></asp:BoundColumn>
    <asp:BoundColumn DataField="Contact_Person" HeaderText="Contact Person"></asp:BoundColumn>
  </Columns>
</asp:datagrid>
</div>
```

The actual c# code the helps create a binding between the database and the data grid using the data bind function. In addition, five default rows are added to the data grid using AddDummyRows functions. The following code shows the c# binding and the dummy rows function.
5.1.2 Auto populating the drop down list

The drop down list is dynamically populated with all the services from service providers. Based on the user selection of the service provider, the drop down list gets auto populated. The following code snippet shows the aspx code for the dropdown list. The data bound property is defined by the dropdown list id.

```csharp
OrgMasterBLL objOrgMast = new OrgMasterBLL();
//Creating a Object for DataSet and Calling the GetAllOrgMaster method from BLL
DataSet dsOrgDet = objOrgMast.GetAllOrgMaster(objOrgMastEntity);
DataTable dtOrgdet = dsOrgDet.Tables[0];
dtOrgdet = AddDummyRows(dtOrgdet, 5, 0);
dgViewData.DataSource = dsOrgDet;
//Binding the DataGrid
dgViewData.DataBind();
```

The backend C# Code support the ASP dropdown list is fetched from the database. The logic that actually depicts the data binding with a list from the database.

```csharp
ddOrgType.Items.Clear(); //Clear the items from Drop Down
OrgMasterBLL objOrgMast = new OrgMasterBLL
DataSet dsOrgType = objOrgMast.GetOrgPType(); //Creating an Object for
//DataSet and Calling the GetOrgPType method
DataTable dtOrgtype = dsOrgType.Tables[0];
ddOrgType.Items.Clear(); //Clear the items from Drop Down
 DataTable dtDrop= BindDropDown(dtOrgtype, "Org_Type_Name", "Org_Type_Code");
 ddOrgType.DataSource = dtDrop;
 ddOrgType.DataTextField = "Org_Type_Name";
 ddOrgType.DataValueField = "Org_Type_Code";
 ddOrgType.DataBind(); //Binding the DropDown List
```
5.1.3 Java Script Validations

The following code depicts the use of JavaScript for the client side validation. The java script is referenced in each fields that are entered by the user need to be validated on the client side. Some of the examples of the pattern matching expressions being used for validations are as follows

- Expression (/^\w+(\[\.-\]?\w+)*@\w+(\[\.-\]?\w+)*\s\w\{2,3\}\+$/) used for email and website Validations
- Expression (/^[0-9]*$/) used for the phone number validation
- Expression (/^\b(1[0-2]|0\[1-9\])\[-\//\](0\[1-9\]|\[12\]\[0-9\]|3\[01\])\[-\//\]((19|20)\d\{2\})-to check valid date.
- Expression (/^[0-9]*$/) used for the any number format validation.

```javascript
//Validation for Empty Fields
function ValidateOrgMaster()
{
    if(document.getElementById('txtOrgname').value=="")
    {
        alert("please enter Organisation Name");
        document.getElementById('txtOrgname').focus();
        return false;
    }
    if(IsValidText('txtOrgname','Organisation Name',2,100,1)==false)
    {
        return false;
    }
    if(document.getElementById('ddlOrgType').value=="-1")
    {
        alert("please select Organisation Type");
        alert(document.getElementById('ddlOrgType').value);
        document.getElementById('ddlOrgType').focus();
        return false;
    }
}
```
5.1.4 Save Update and Delete event

Most part of this application writes to the database. These major events get triggered based on the user selection. The asp web page just defines the button and formatting of the text. On click of each button the data from the web form is transferred into a entity object with the predefined getters and setter.

```html
<asp:button id="btnNew" runat="server" CssClass="buttonstyle" Width="60px"
Text="New" onclick="btnNew_Click"></asp:button></TD>
<td align="center" colSpan="1" rowSpan="1">

All the data populated in the web form is stored in an object ojOrgMastEntity.Org_Name.
In addition, if the existing record is selected from the grid for updating, there some hidden labels in the asp pages the pass the value of the primary key.

```csharp
if(blnUpdate)
{
    //Assign the Org_Name to the hidden field
    //Assign the Org_Name to the TextBox
    objOrgMastEntity.Org_Name =
    txtOrgname.Text.ToString().Trim();
    //Assign the Org_Type_Code to the DropDown List
    objOrgMastEntity.Org_Type_Code =
    dd1OrgType.SelectedValue;
    //Assign the Phone_No1 to the TextBox
    objOrgMastEntity.Phone_No1 = txtPhoneno1.Text.ToString();
    //Assign the Phone_No2 to the TextBox
    objOrgMastEntity.Phone_No2 = txtPhoneno2.Text.ToString();
    //Assign the Web_site to the TextBox
    objOrgMastEntity.Web_site = txtWebsite.Text.ToString();
    //Assign the Mail_Id to the TextBox
    objOrgMastEntity.Mail_Id = txtEmail.Text.ToString();
    //Assign the Description to the TextBox
    objOrgMastEntity.Description =
    txtDescription.Text.ToString();
    //Assign the Contact_Person to the TextBox
    objOrgMastEntity.Contact_Person =
    txtCntPerson.Text.ToString();
    //Assign the Address to the TextBox
    objOrgMastEntity.Address = txtAddress.Text.ToString();
    objOrgMastEntity.Active = chkActive.Checked;
    objOrgMastEntity.User_Name =
    Session["Admin_User_Name"].ToString();
```
5.1.5 Attaching the case sheet to the customer

The case sheet can be attached by the customer and viewed by the hospital. The link of the case sheet is provided for the hospital service provider. On click of the link display, the module uses the windows downloader to download and view the file. When the case sheet is attached the case sheets are stored in folder on the server. Therefore, when retrieving the case sheet, the name is retrieved from the database and url is formatted to point to a location on the server. There is a link to attached case sheet that is made visible if the user tries to update an enquiry.

```csharp
strFile = dsHosEnqDetails.Tables[0].Rows[0]["Case_Sheet"].ToString();
if(strFile !="")
{
    string strUrl =Request.Url.AbsoluteUri;
    //strFile = strpath + strFile;
    string[] str = strUrl.Split('/');
    strUrl= string.Empty;
    int iCount =0;
    for(int iCnt=0; iCnt<str.Length ;iCnt++)
    {
        if(str[iCnt]=="Administrator" || str[iCnt]=="Customer" ||
            str[iCnt]=="Partner")
        {
            iCount =iCnt;
            break;
        }
    }
    for(int iRow=0;iRow<iCount;iRow++)
    {
        strUrl +=str[iRow] +"/";
    }
    strpath =strUrl +"CASESHEETS/";
    strUrl =strpath + strFile;
    hdnFilePath.Value = strFile;
    string[] strArr=strUrl.Split('\');
    string strRealLoc=string.Empty;
    for(int iCnt=0;iCnt<strArr.Length;iCnt++)
    {
        if(iCnt!=strArr.Length-1)
            strRealLoc+=strArr[iCnt]+"/";
        else
            strRealLoc+=strArr[iCnt];
    }
    lnkCaseSheet.Visible =true;
    lnkCaseSheet.Attributes.Add("onclick","javascript:return
OpenCaseSheet("+strRealLoc +"\")");
```
The following code snippet shows, the attaching of case sheets and saving to the location on the server. The http file collection class provides access to all files uploaded by the client. On click of the browse button, the C# code logic is defined to save the case sheet in a location. The file will not be uploaded if the file is blank. The file will be saved to a location on the server. On a successful save, the case sheet is saved to location on the server. The name of the case sheet will be saved as object property of objECustomer. This object is passed to the stored procedure and saved as a new enquiry in the database.

```csharp
string TempFileName = string.Empty;
//Creating an object for the class BCustomerEnq
BCustomerEnq objeBCusEnq = new BCustomerEnq();
str = SaveFile();
setHospitalEnquiry(str);
if(strFileLoc != "") //checking the File name is blank or not
{
    //HttpFileCollection class provides access to all the files uploaded
    HttpFileCollection MyFileCollection = Request.Files;
    for (int Loop1 = 0; Loop1 < MyFileCollection.Count; Loop1++)
    {
        strpath = strpath + "CASESHEETS/"; // Create a new file name.
        //strpath = Request.Url.AbsolutePath;
        TempFileName = strpath + strFileLoc;
        string[] strArr = TempFileName.Split('\\');
        string strRealLoc = string.Empty;
        for (int iCnt = 0; iCnt < strArr.Length; iCnt++)
        {
            if (iCnt != strArr.Length - 1)
                strRealLoc += strArr[iCnt] + "/";
            else
                strRealLoc += strArr[iCnt];
        }
        //saves all the files uploaded by the client to a folder named AllFiles
        MyFileCollection[Loop1].SaveAs(strRealLoc);
    }
    string strMess =
    objeBCusEnq.CreateBHospitalEnquiry(objECustomer);
    Page.RegisterStartupScript("onclick","<Script
type=\"text/javascript\">alert(" + str + "+ TempFileName + 
\")\</Script>\);
    Page.RegisterStartupScript("onclick","<Script
type=\"text/javascript\">alert(" + TempFileName + 
\")\</Script>\);
```
5.1.6 Adding a window view of the profile information

The javascript function redirects the data and calls the customer profile page and populates with user information. The function is written on aspx page. Whenever there is a on click event for the user profile, It is opened up in a new window.

```javascript
function Profile(user)
{
    window.open('CustomerProfile.aspx?User='+user,'','height=350,width=350');
    return false;
}
function OpenCaseSheet(filename)
{
    if(filename!='')
        window.open(filename);
    return false;
}
```

5.1.7 Use CSS Style sheets

These CSS style sheets are used to maintain the style of the web pages and maintain consistency in the flow from one page to the other. The predefined styles have the consistency and make the it easy to develop the GUI interface. The CSS style sheets increases the usability and maintainability of the code.

```
.divstyles
{
    overflow-y:auto;overflow-x:hidden;OVERFLOW:auto;height:100;BACKGROUND-COLOR:#e7e7e7;/*BACKGROUND-COLOR:#6f939b;*/
    SCROLLBAR-FACE-COLOR: #749482;SCROLLBAR-HIGHLIGHT-COLOR: gray;SCROLLBAR-SHADOW-COLOR:gray;SCROLLBAR-ARROW-COLOR:black;
    SCROLLBAR-TRACK-COLOR:darkgray;SCROLLBAR-DARKSHADOW-COLOR:#6f939b;
}
 textbox
{
    font-family: Verdana, Arial, Helvetica, sans-serif, Tahoma;font-size: 10px;font-style: normal;line-height: normal;font-weight: normal;
    font-variant: normal;text-transform: none;height: 18px;width: 150px;padding-bottom:2px;padding-top:2px;border-style:double;border-top-color: #6f939b;
    border-right-color: #6f939b;border-bottom-color: #6f939b;border-left-color: #6f939b;color: #000000;border-width:1px;
}
```
5.1.8 Referencing the external dll for the side panel [9].

The use of the external dlls helps me integrate the date whenever the customer or the partner wanted to pick the date of the treatment. This is a custom control and is integrated into the project by using the following code, and custom controls can increase the usability of the project. Any change can be made separately and after that, the dll can be placed in any project. Invoking this dll helped enhance the appearance of the web pages.

The date picker is invoked using the following code.

```csharp
<%@ Register TagPrefix="cgspnl" Namespace="Web.UI.Controls.WebCollapsiblePanel" Assembly="Web.UI.Controls.WebCollapsiblePanel" %>
<%@ Register TagPrefix="cgsdp" Namespace="Web.UI.Controls.WebDatePicker" Assembly="Web.UI.Controls.WebDatePicker" %>
```

5.2 Storing and Retrieving Data using ADO.NET

ADO.NET is a set of classes that expose data access services to the .NET programmer. ADO.NET provides functionality to developers, for writing managed code. Data access logic components provide simple access to database functionality queries and data operations, returning both simple and complex data structures. When an application requires large numbers of data access components to access the same data source, there is need to access the database using generic methods. These generic methods are usually called the Helper class.

All ADO.NET functionality in the .NET Runtime class library comes in three general namespaces and number of provider-specific namespaces. The three general namespaces
are System.Data, System.Data.Common and System.Data.SqlTypes. Some of the provider specific namespaces are System.Data.OleDb, Microsoft.Data.Odbc and System.Data.SqlClient. The following code snippet shows retrieving all events like insert or update events using the stored procedure call from the database [5]. In the following code the input parameter are fetched from the stored procedure and associated with the entity object created.

```csharp
SqlParameter[] sqlParam = null;
DataTable dtTransDet = null;
dtTransDet = SQLCommonDAL.PrepareTransTable();
DataRow droTrans = dtTransDet.NewRow();
droTrans[0] = 1;
droTrans[1] = "usp_CreateOrgDetails";
sqlParam = SQLCommonDAL.GetParams("usp_CreateOrgDetails", objOrgMastEntity.GetEntity());
droTrans[2] = sqlParam;
dtTransDet.Rows.Add(droTrans);
//Call the ExecuteNonQuery method from Common DAL and Return its value
SQLCommonDAL.ExecuteNonQuery(dtTransDet);
return (string) sqlParam[11].Value;

public static SqlParameter[] GetParams(string strSpName, Hashtable htOrgMastEntity)
{
    strConn = ConfigurationSettings.AppSettings["DBconnection"].ToString();
    SqlParameter[] sqlParams = new SqlParameter[1];
    sqlParams[0] = new SqlParameter("@procedure_name", strSpName);
    return PrepareParams(SqlHelper.ExecuteDataset(strConn, CommandType.StoredProcedure, "sp_sproc_columns", sqlParams), htOrgMastEntity);
}
```
The following method is used to execute a non-query; this code depicts how the connection is created using the Ado.Net SqlConnection class defined in the System.Data.SqlClient namespace. All the connections to the database are maintained as sql transactions. The maintaining a database call as a transaction helps us to rollback if an error occurs on the server side.

```csharp
public static int ExecuteNonQuery(DataTable dtTransDet)
{
    SqlConnection sqlCon = null;
    SqlTransaction sqlTrans = null;
    SqlParameter[] sqlParams = null;
    string strSpName = string.Empty;
    int iResult = 0;
    DataView dvTransDet = dtTransDet.DefaultView;
    dvTransDet.Sort = "Priority ASC";
    using (sqlCon = SQLCommonDAL.GetConnection())
    {
        try
        {
            sqlCon.Open();
            sqlTrans = sqlCon.BeginTransaction();//IsolationLevel.RepeatableRead
            for (int iRowCount = 0; iRowCount < dvTransDet.Table.Rows.Count; iRowCount++)
            {
                sqlParams = (SqlParameter[])dvTransDet.Table.Rows[iRowCount]["Sql_Params"];
                strSpName = dvTransDet.Table.Rows[iRowCount]["Sp_Name"].ToString();
                if (sqlParams != null && sqlParams.Length > 0)
                {
                    iResult = SqlHelper.ExecuteNonQuery(sqlTrans, CommandType.StoredProcedure, strSpName, sqlParams);
                }
            }
            sqlTrans.Commit();
            return iResult;
        }
        catch (SqlException sqlEx)
        {
            sqlTrans.Rollback();
            throw sqlEx;
        }
        catch (Exception Ex)
        {
            sqlTrans.Rollback();
            throw Ex;
        }
    }
}
```
5.3 Implementation of Business Logic

Business Logic Layer (BL) that serves as an intermediary for data exchange between the presentation layer and the DAL. In a real world application, the BLL should be implemented as a separate class library project. The methods in each class vary based the functionality and returning result set. The business logic layer has methods defined to call the methods form the DAL and save the results in the form of an object. The resulting object will be passed back to the presentation layer. The following examples depict a few methods of the business logic layer. The method calls the DAL and passes the object as the input or the output parameter.

```csharp
class OrgMasterDAL
{
    //Creating instance for OrgMasterDAL
    OrgMasterDAL objOrgMasterDal = new OrgMasterDAL();
    //Call the UpdateOrgMaster method with entity object
    objOrgMasterDal.UpdateOrgMaster(objOrgMastEntity);
}
class OrgMastEntity
{
    //Creating instance for OrgMasterDAL
    OrgMasterDAL objOrgMastDal = new OrgMasterDAL();
    //Call the GetOrgTypeDetails method with entity object
    return objDOrgMaster.GetOrgTypeDetails();
}
```
Chapter 6

CONCLUSION

This project is being developed as real time application for a client. It aims at providing an efficient management solution for the client. It allows the users to interact with the system and schedule natural therapy appointments. This system would ensure better communication and quick access to the related information for their end users. This project is efficient in realizing the goals of the different end users and customizing according to the role of the end user. This project is designed in a modularized way for easily maintainability.

The following are some of the few enhancements that can be made to the project.

- This project can be enhanced using third party API to make flight reservation.
- Show customer a map of the different service providers and give the customer an estimate of time per travel.
- An enhancement to the partner module will be giving them the ability to add videos of their facility and attract customers.
- As current system only provides email authentication, sending out reminder’s to the customer as per the customer’s request would be an enhancement.
APPENDIX

User Guide

The functionality and usability of the system is described in this section.

User Login: The user will select the appropriate link from the homepage and it will redirect him to his specific role.

Admin login: If the user logs in as an Administrator, he will be redirected to the following page. The username and password entered in the login page, are validated against the entries in database. The database table being used is the [tbl.Administrators]. The stored procedure is the sp.AuthenticateAdmin. It also avoids processing, if the user enters an incorrect username and password.

Figure 7.1 Administrator Login Page
Customer login: The username and password entered in the login page are validated against the entries in the database. The database table being used is the [tbl.Customers]. The stored procedure is the sp.AuthenticateCustomer. This page avoids processing, if the user does not enter a username and password. There is also a new user registration available on this page.

Figure 7.2 Customer Login page Screen Shot

Partner Login: The username and password entered in the login page are validated against the entries in the database. The database table used is the [tbl.PartnerLogin]. The stored procedure is the [sp.AuthenticatePartner].
7.1 New Customer Registration

If the user selects to login as customer and if he is a new user, he will be redirected to the new user registration page. All the credentials selected will be validated on the client side with the following validations

- Password mismatch.
- Empty is not allowed in the first name and last name.
- Numbers are not allowed in the first name and last name.
- Alphabets are not allowed in age.
- Phone number should be 10 digits.
- Generic email check for @ and special characters.

Information entered by the user is written to the database. The database table to which the data is written is [tbl.Customers]. The stored procedure is the [usp.CreateCustomer]
7.2 Update Customer Profile

1. Once the customer logs into the system; the first page (which is My Profile) gives him all the details of his profile.

2. It also provides him the ability to update his profile and all the updates will be saved to the database.

3. It also provides the customer with the side menu options, to select other web pages linked to the customer profile like enquires and quotations.

4. Dynamic display of the username logged into the system.

5. It has the option for the user to sign out.
6. The client side validations would be able catch the following when the user is updating his profile

   - Empty is not allowed in the first name and last name.
   - Numbers are not allowed in the first name and last name.
   - Alphabets are not allowed in age.
   - Phone number should be 10 digits.
   - Generic email check for @ and special characters.

7. Personal information of the users is fetched from the database. The database table [tbl.Customers] will be queried based on the user name and all the appropriate fields will be returned to the web page. The stored procedure being used to fetch the data [usp_GetCustomerDetails].
7.3 Manage Enquires

1. The three options for enquiries will be represented with three different tabs as Hospital Enquires, Hotel enquires, Travel Enquiries.

2. The visibility of enquires, will be defined based on user selection (defaulted to the Hospital enquires).

3. Each of these pages is loaded with a grid which shows up any previous enquires of the user fetch from the database. And all the previous enquires will be fetched from database by using the following store procedures based on the type.
   - usp_GetHospitalEnquiryDetails --for Hospital Enquiry
   - usp_GetHotelEnquiryDetails-- for Hotel Enquiry
   - usp_GetTravelEnquiryDetails--for Travel Enquiry

4. The web forms load up with appropriate question for the user, based on the enquiry type.

5. List of all hospitals will display in the drop down.

6. Based on the hospital selected the list of the services will be displayed in another dropdown.

7. Similarly the Hotel list drop down and the travel list drop down are populated dynamically from the server side.

The client side validations would catch the following when the user requests a enquiry:

- Empty field is not allowed in the name.
- Numbers are not allowed in the name field.
- Alphabets are not allowed in age.
- Phone number should be 10 digits.
- Generic email check for @ and special characters.
- Nationality and country cannot be empty or numeric.
- Currently all the fields are defined as required in the client end.

Figure 7.6 Customer Enquiry

The customer will be able to attach a case sheet to when he is posting the enquires to the hospital. The hospitals will be able post back their responses reviewing the patient’s previous medical history. When customer clicks on he will be able to attach a case sheet and upload the case sheet into the server as shown in figure 7.7 and figure 7.8.
Figure 7.7 Attach Case sheet

Figure 7.8 Choose File to Upload
7.4 Manage Quotations

1. The three options for the enquires will be represented in three different tabs as Hospital Quotations, Hotel Quotations, Travel Quotations.

2. The visibility of the quotations will be defined based on user selection (defaulted to the Hospital Quotations).

3. Each of the pages is loaded with a grid which shows up any previous enquires of the user fetch from the database. In addition, all the quotations will be fetched from database by using the following store procedures based on the type.

4. The page load not only fetches the quotation details but also fetches the enquiry details.
   - usp_GetHospitalEnquiry - For Hospital Enquiry
   - usp_GetHospitalQuotions - For Hospital Quotations
   - usp_GetHotelEnquiry - For Hotel Enquiry
   - usp_GetHotelQuotions - For Hotel Quotations
   - usp_GetTravelEnquiryDetails - For Travel Enquiry
   - usp_GetTravelQuotions - For Travel Quotations

5. The web forms load up with appropriate enquiry for the user, based on the type.

6. All the enquiry details will be fetched based on the quotation type and quotation selected from the grid.
7. When the user confirms the quotation, the corresponding partner database will be updated with the username and details. If the user does not confirm within the given time the quotation will expire.

8. The client’s side validations are not defined for this page, because there is no data entry in this page.

Figure 7.9 Customer Quotations
7.5 Organization Master

1. Once the administrator logs into the system the login page redirects to the Organization Master.

2. This page loads up with the grid view of all the organizations registered with the system.

3. As the administrator, makes his selection from the grid, the page loads up with all the details about that particular organization.

4. The administrator has the ability to add a new organization, delete an existing organization.

5. When add a new user, there is a drop down added for the organization type selection.

6. Dynamic display of the username logged into the system.

7. Option for the user to sign out.

8. The client side validations would catch the following when the admin is adding a new a partner profile

   - Empty is not allowed in the Organization Name and Organization Type.
   - Alphabets are not allowed in age
   - Phone number should be 10 digits.
   - Generic email check for @ and special characters.
   - Generic website check for string.
7.6 Manage Partners

1. Once the administrator logs in, he will be able to select and manage partners.

2. This page loads up with the grid view of all the organizations registered with the system.

3. This page will give the administrator the ability to view and change usernames and passwords for an organization.

4. The administrator has the ability to assign passwords and usernames for the organization.

5. This gives the administrator the ability to make an organization active or inactive.
6. The ability to edit the usernames and passwords will appear only when the administrator makes a selection.

7. The administrator has the ability to add new usernames to a single partner profile.

8. Dynamic display of the username logged into the system.

9. Option for the user to sign out.

10. All personal information of the user will be fetched from the database. The database table will be queried based on the user name and all the appropriate fields will be returned to the web page.

11. The client side validations would catch any missing fields when the administrator clicks to add a new partner profile. The username and passwords fields should be filled in order to save to the data in to the database.
The Administrator will have ability to add multiple user accounts for an organization by clicking new button.

![Add New User to Partner](image)

Figure 7.13 Add a New User to the Partner

7.7 Manage Users

1. Once the Admin logs in the can manage users.

2. This page loads up with the grid view of all the users registered with the system.

3. This page will give administrator the ability to view and change user details other than the username and passwords.

4. This will give the admin the ability to make a user active or inactive.

5. Once the admin selects a user from the grid, the details of the user will load the page. This load will provide the admin with user details so that he can make his edits.

6. Dynamic display of the username logged into the system.
7. Option for the user to sign out.

8. All personal information of the user will be fetched from the database. The database table will be queried based on the user name and all the appropriate fields will be returned to the web page.

![Figure 7.14 Manage Users](image)

**Figure 7.14 Manage Users**

7.8 Partner Organization Profile

1. Once the Partner representative logs in, he will be able to review the partner profile entered by the admin.

2. The appropriate pages are picked up automatically based on the organization type.

3. Representative will be able to make updates to their Organization Profile and save the changes.

4. The organization type will get auto populated based the type user logged in.
5. For the hospital: the following is the Organization Profile page it is retrieved using the stored procedure [usp_GetPartnerOrgDetails].

6. The organization type is auto populated for the hotel and works similarly for travels.

7. The partner representative does not have the permission to change the organization type.

8. The organization type is auto populated for the Hotel and works similarly for travels.

9. The partner representative does not have the permission to change the organization type.

Figure 7.15 Partner Organization Profile
7.9 Update Partner Account

By selecting my profile, the partner representative will be able to update his username and password and will be able to save it to database.

7.10 Update and Register Hospital Services

1. The Hospital provider will be able to enter their new services.

2. Once the service is entered, it’s validated on client side.

3. If they select an existing service from grid the button options will change to update (update, save and delete options).

4. The services are saved using this stored procedure usp_CreateHospitalService.

5. All the services are returned to grid, using the following stored procedure [usp_GetHospitalServices].
Figure 7.17 Partner Hospital Services

7.11 Register and Update Hotel Services

1. The Hotel provider will be able to enter their new services.

2. Once the service is entered, it is validated on client side looking for phone number and addresses.

3. If they select an existing service from grid the button options will change to update (update, save and delete options).
4. Also there special grid for the hotel provider is that they can add details about different type of suites that are available. In addition, there is validation on the hotel details tab.

5. In addition to the above features multiple hotel details can inserted using the add button.

6. The services are saved using this stored procedure usp_CreateHotelService.

7. All the services are returned to grid using the stored procedure usp_GetHotelServices.

Figure 7.18 Update and Register Hotel Services
7.12 Register and Update Travel Services

1. The travel provider will be able to enter their new services.

2. Once the service is entered its validated on client side looking for numeric in the seating capacity and availability fields etc.

3. If they select an existing service from grid the button options will change to update (update, save and delete options).

4. The services are saved using this stored procedure usp_CreateTravelService.

5. All the services are returned to grid using the stored procedure usp_GetTravelServices.

Figure 7.19 Update and Register Travel Services
7.13 Partner Enquiries

1. All the enquiries posted by the customer are posted to the Grid on the enquiries page. The partner can review them by category, by selecting on tab.

2. The fields entered by the customer are viewed here.

3. The Partner does not have permission to update or delete these enquiries.

4. The Partner can view the customer profile in a window by selecting a customer profile link.

5. The only option available to the partner is the to prepare a quotation described in the next use case.

Figure 7.20 Partner Enquiry
7.14 Partner Quotations

1. Based on the type of the partner, the grid loads up with all the quotations posted by the partner.

2. The representative can select the quotations from grid and will be able to update the quotation posted.

3. New quotation is prepared with the `prepare quotation` button in the Enquiry tab

4. All fields are validated on the client side with appropriate validation like date format, numbers day of treatment and others.

5. Also added the date picker for the date selection.
6. This page functionality is similar to the other partner modules like Hotel and travel services.

7. When the representative selects a quotation, he will be able to view the posted enquiry information by clicking on the left hand side top corner of the page.

8. The Enquiry information comes up in a popup window as shown below.
BIBLIOGRAPHY

1. Jesse Liberty and Dan Hurwitz (2006); Programming ASP.NET – Building Web Applications and Services, 3rd Edition; O’Reilly Media

2. Jeffery P. McManus and Chris Kinsman (2002); C# Developer’s Guide to ASP.NET, XML and ADO.NET; Pearson Education

3. Itzik Ben-gan (2008); Microsoft SQL Server 2008 T-SQL Fundamentals, Microsoft Press

4. Ray Toal and John David Dionisio; The JavaScript Programming Language,

5. Use Case Modeling

6. Application Architecture

7. Advantages of ASP.NET
   http://ezinearticles.com/?What-is-ASP.NET?--Top-12-Advantages-of-ASP.NET&id=1358877

8. Introducing .NET Framework
   http://www.developerfusion.com/samplechapter/1678/introducing-net/2/

9. Date Picker Module
   http://www.codeproject.com/KB/customcontrols/DatePicker_Custom_Control.aspx

10. Systems Maintainability