DYNAMIC WEB-BASED ON-CALL SCHEDULE APPLICATION

Chaitanya Deepti Polavarapu
B.E., Jawaharlal Nehru Technological University, 2004

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
DYNAMIC WEB-BASED ON-CALL SCHEDULE APPLICATION

A Project

by

Chaitanya Deepti Polavarapu

Approved by:

____________________________, Committee Chair
Ahmed Salem, Ph.D.

____________________________, Second Reader
Chung E. Wang, Ph.D.

____________________________
Date
Student: Chaitanya Deepti Polavarapu

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.

Nikrouz Faroughi, Ph.D.

Date

Department of Computer Science
Abstract

of

DYNAMIC WEB-BASED ON-CALL SCHEDULE APPLICATION

by

Chaitanya Deepti Polavarapu

Statement of Problem: With today’s advancement of technology and software, manually creating an on-call schedule for a geographically distributed team is unsophisticated and error prone. A dynamic web-based calendar for creating an on-call schedule is easy and maintainable without any dependencies on man-made errors.

Conclusions Reached: This project is aimed at creating an on-call schedule by workweeks for geographically distributed team members. The team and users will have a better way of viewing the on-call schedule for all the future workweeks. In addition, it provides the capability of easily modifying the existing schedule and/or adding a new schedule for the upcoming workweeks, while restricting the team members’ access.

__________________________, Committee Chair
Salem Ahmed, Ph.D.

__________________________
Date
ACKNOWLEDGMENTS

I would like to thank all the people who helped me directly or indirectly in completing this project successfully.

I would like to extend my sincere gratitude to my supervisor, Dr. Ahmed Salem for his support and encouragement in making this project a success. His excellent suggestions and timely encouragement have helped me complete the project.

Also, thanks to Dr. Chung E. Wang 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

<table>
<thead>
<tr>
<th>Chapter</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Acknowledgements</td>
<td>v</td>
</tr>
<tr>
<td>List of Tables</td>
<td>viii</td>
</tr>
<tr>
<td>List of Figures</td>
<td>ix</td>
</tr>
<tr>
<td>1. INTRODUCTION</td>
<td>1</td>
</tr>
<tr>
<td>1.1 Background</td>
<td>1</td>
</tr>
<tr>
<td>1.2 Problem Statement</td>
<td>2</td>
</tr>
<tr>
<td>1.3 Project Objective</td>
<td>3</td>
</tr>
<tr>
<td>2. REQUIREMENT ANALYSIS</td>
<td>4</td>
</tr>
<tr>
<td>2.1 Product Perspective</td>
<td>4</td>
</tr>
<tr>
<td>2.2 Product Features</td>
<td>4</td>
</tr>
<tr>
<td>2.3 Hardware and Software Requirements</td>
<td>5</td>
</tr>
<tr>
<td>2.3.1 Hardware Requirements</td>
<td>5</td>
</tr>
<tr>
<td>2.3.2 Software Requirements</td>
<td>5</td>
</tr>
<tr>
<td>2.4 Operating Environment</td>
<td>6</td>
</tr>
<tr>
<td>2.5 Assumptions and Dependencies</td>
<td>6</td>
</tr>
<tr>
<td>2.6 Functional Requirements</td>
<td>7</td>
</tr>
<tr>
<td>2.7 External Interface Requirements</td>
<td>8</td>
</tr>
<tr>
<td>2.7.1 User Interfaces</td>
<td>8</td>
</tr>
</tbody>
</table>
LIST OF TABLES

<table>
<thead>
<tr>
<th></th>
<th>Table 3.3.1 – On Call Rotation</th>
<th>20</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>Table 3.3.2 – Roles</td>
<td>21</td>
</tr>
<tr>
<td>3</td>
<td>Table 3.3.3 – Users</td>
<td>22</td>
</tr>
<tr>
<td>4</td>
<td>Table 3.3.4 – UserRoles</td>
<td>23</td>
</tr>
<tr>
<td>5</td>
<td>Table 3.3.5 – RoleGroups</td>
<td>24</td>
</tr>
<tr>
<td>6</td>
<td>Table 3.3.6 – User Authentication</td>
<td>24</td>
</tr>
<tr>
<td>Figure</td>
<td>Description</td>
<td>Page</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
<td>------</td>
</tr>
<tr>
<td>1.</td>
<td>Figure 3.1 – 3-Tier Architecture</td>
<td>12</td>
</tr>
<tr>
<td>2.</td>
<td>Figure 3.2 – DotNetNuke Architecture</td>
<td>13</td>
</tr>
<tr>
<td>3.</td>
<td>Figure 3.2.1 – Data flow diagram for on-call rotation</td>
<td>15</td>
</tr>
<tr>
<td>4.</td>
<td>Figure 3.2.2 – Data flow diagram for Sending page</td>
<td>16</td>
</tr>
<tr>
<td>5.</td>
<td>Figure 3.2.3 – Data flow diagram for event calendar</td>
<td>17</td>
</tr>
<tr>
<td>6.</td>
<td>Figure 3.3 – Database design</td>
<td>19</td>
</tr>
<tr>
<td>7.</td>
<td>Figure 4.1.1 – ASP.NET Architecture</td>
<td>26</td>
</tr>
<tr>
<td>8.</td>
<td>Figure 4.1.2 – DotNetNuke Stack</td>
<td>29</td>
</tr>
<tr>
<td>9.</td>
<td>Figure 4.2 – User View Screen</td>
<td>30</td>
</tr>
<tr>
<td>10.</td>
<td>Figure 4.3 – Manage Screen</td>
<td>32</td>
</tr>
<tr>
<td>11.</td>
<td>Figure 4.4 – Manage - Edit Screen</td>
<td>33</td>
</tr>
<tr>
<td>12.</td>
<td>Figure 4.5 – Manage - Delete Screen A</td>
<td>34</td>
</tr>
<tr>
<td>13.</td>
<td>Figure 4.6 – Manage - Delete Screen B</td>
<td>35</td>
</tr>
<tr>
<td>14.</td>
<td>Figure 4.7 – Manage - Add Items to Empty Grid Screen</td>
<td>35</td>
</tr>
<tr>
<td>15.</td>
<td>Figure 4.8 – Manage - Add More Items Screen</td>
<td>36</td>
</tr>
<tr>
<td>16.</td>
<td>Figure 4.9 – Security Roles Screen</td>
<td>37</td>
</tr>
<tr>
<td>17.</td>
<td>Figure 4.10 – Filter By Role Group Screen</td>
<td>38</td>
</tr>
<tr>
<td>18.</td>
<td>Figure 4.11 – Add New Role Group screen</td>
<td>39</td>
</tr>
<tr>
<td>19.</td>
<td>Figure 4.12 – Add New Role Screen</td>
<td>40</td>
</tr>
<tr>
<td>20.</td>
<td>Figure 4.13 – User Settings Screen</td>
<td>41</td>
</tr>
</tbody>
</table>
21. Figure 4.14 – Add Users Screen
22. Figure 4.15 – Add New User Form
23. Figure 4.16 – Edit User Account Screen
24. Figure 4.17 – Edit User Account - Manage User Credentials Screen
25. Figure 4.18 – Edit User Account - Manage Roles Screen
26. Figure 4.19 – Edit User Account - Manage Password Screen
27. Figure 4.20 – Edit User Account - Manage Profile Screen
28. Figure 4.21 – Send Page Screen
29. Figure 4.22 – Pager Information Screen
30. Figure 4.23 – Event Calendar Screen
31. Figure 4.24 – Event Description Screen
32. Figure 4.25 – View Event Detail Screen
33. Figure 4.26 – Add/Edit Event Screen
34. Figure 4.27 – Delete Event Screen
Chapter 1

INTRODUCTION

This chapter introduces the project, outlines the background to the problem, identifies the problem owner, and gives an outline of the project objectives.

1.1 Background

The On-Call Schedule application is built on a Three – tier architecture [3]. The three – tier architecture is a client – server architecture [4] where the user interface, business logic, and data access layer are maintained as independent modules. This web based application uses DotNetNuke technology [2]. DotNetNuke is an open source web content management platform for Microsoft ASP.NET [1]. DotNetNuke is chosen because it empowers custom app development with an open API for developers and from the end users perspective it is an easy to use platform. Granular user permissions provide administrators with flexible control.

The On-Call Schedule application displays the on-call person for the current workweek followed by the list of on-call persons for the coming 11 workweeks so customers having issues will contact the respective on-call person to resolve the issue. The On-Call Schedule application is used by both the customers and team members. The team members will have access to add, modify, and manage the schedule in a timely manner. Customers will have read-only access. The event calendar displays the events, if any are scheduled. These events can be edited or deleted by the team members. The team members can also add new events to the calendar.
1.2 Problem Statement

The traditional On-Call Schedule application is a static and hard coded HTML page. The team member in the respective region across the world enters the schedule manually in the HTML source for the specified number of workweeks. The team member is required to change the schedule promptly, such that the page reflects the current schedule. The customer can only see the schedule for 13 workweeks, which includes one workweek prior to the current workweek, the current workweek, and 11 upcoming workweeks.

1.3 Project Objective

This project is being developed to meet the following primary goals:

- Make the application more user-friendly to both the team members and customers
- The customer will have a better view of the schedule in which he/she can see the schedule for any number of past workweeks or the schedule for any number of future weeks depending on the number of records in the database.
- The team members can create the schedule for the coming workweeks through this application.
- The application provides a more flexible way to add, modify, and manage the on-call schedule.
- The application also has an event calendar on which the team members can add events like vacations, downtimes for projects, go-live events of the projects, etc.
• The events on the calendar can be edited and/or deleted.
Chapter 2

REQUIREMENT ANALYSIS

2.1 Product Perspective

The goal of this project is to create an on-call schedule to be used by several customers for critical issues on a daily basis. One needs this application to know who to contact, when to contact them, and how to contact them to resolve the issues. This application should be user-friendly, easy to learn, and reliable. This project is intended to be a stand-alone product and should not depend on the availability of other software. It should run on both UNIX and Windows-based platforms.

2.2 Product Features

The dynamic web-based application for creating an on-call schedule allows distributed team members to associate the schedule with workweeks, manage and store the associations, and produce a report showing on-call persons by workweek for various geographical areas for that workweek/year. The report has pager information and contact details of the on-call person. The customer can also send an email/message through a mailbox interface/default mail client to the on-call person or to the entire group by clicking on the on-call person’s name or the on-call team name. A team member can grant access privileges to another member of his/her team and query and search for on-call persons for a given workweek as well as in what time slots those respective persons are on call. Also, the team member will have the privilege of configuring it such that the team member can alter the team member’s own workweeks and on-call schedule and can
assign other team members to cover for his/her workweek by viewing the other team members’ on-call schedules. Also, the user can view the list of users associated in all the geographical regions. The team member can associate the year and workweek with any other team member and then save these associations to the database. This application also has a calendar control known as the event calendar on which the team members can add events like vacations or go-live for events for specific projects and add downtimes for projects, etc. The team members can also edit event details and even delete the event. This application allows the customer to send an email/page to the on-call person with the customer’s message through a mailbox interface. The reports also have additional detailed information explaining how to reach the on-call person and how to get support if they cannot reach the on-call person.

2.3 Hardware and Software Requirements

The hardware and software requirements for the project are as below.

2.3.1 Hardware Requirements

Pentium-IV Processor with 500MHZ or above

40 GB of Free hard disk space

1 GB RAM

LAN Network (for remote sources)

Network interface card or modem (for remote access)

2.3.2 Software Requirements

WINDOWS XP | Windows 2008
2.4 Operating Environment

Since the system will be implemented in Microsoft ASP.NET technology, the software will need to be hosted on an ASP.NET-compatible site. The system will also require one SQL database to be installed on the host space, as well as any additional software required for sending email to users of the system. The system must be completely compatible with any browser that fully supports Microsoft ASP.NET technology. The users of the software will be expected to have an Internet connection that, at a minimum, shall be a 56kbps modem.

2.5 Assumptions and Dependencies

1. It is assumed the system will be developed using the ASP.NET technology.

2. It is assumed the system will be able to interface with an email server in order to send the page or email to the on-call person in the system.

3. It is assumed the system will interface with a SQL Server 2008 database.
2.6 Functional Requirements

The functional requirements define the intended behavior of the system in a complete and unambiguous form. Following are the tasks that can be achieved through this application:

- Team members can create an on-call rotation schedule for any number of weeks or years for any geographical area.
- Team members can edit, delete, and update any existing instance of the schedule.
- Team members can add the schedule for the coming weeks to the already existing schedule.
- Users and team members can view the past, current, and future records of the on-call schedule for any geographical area.
- Administrators of this application can provide administrator/super user access to any team member.
- Administrators of this application can add new team members to the on-call rotation schedule.
- Administrators of this application can delete any existing members from the on-call rotation schedule.
- Administrators can generate or reset the password of any team member.
- Administrators can grant authorization to any of the team members.
➢ Calendar control of this application allows the user to view any events that may exist.

➢ Team members can add events to the calendar control, such as posting downtimes for the projects, team member vacations, and a go-live schedule for the projects.

➢ Users/customers will have the ability to send a page to the on-call person for any critical production issue.

➢ Team members can edit, delete, and update the existing event of the calendar control.

2.7 External Interface Requirements

2.7.1 User Interfaces

The user interface for the system will be a web page on the Internet with the DotNetNuke platform. The user interface will be limited to the types of controls that can be generated using HTML and ASPX web user controls. When a function is performed like adding, editing, or deleting, the current window will be performing the action.

2.7.2 Software Interfaces

1. Operating System: The software is being designed to run on Windows 7/Vista/XP/2000. The operating systems include the latest version of Internet Information Services.

2. Web Server: The software is being designed to run on Internet Information Server version 7.0.
3. Database: The software will access the SQL Server 2008 Enterprise Edition database for adding, updating, and deleting entries in the database.

4. Libraries: The software will be created using the Microsoft.NET version 3.5 and DotNetNuke frameworks.

2.8 Non-functional Requirements

2.8.1 Security Requirements

1. Passwords need to be displayed as “*” in all the forms wherever required.

2. Every user of the system needs to be assigned a unique login and password to access the application over the Internet.

3. Proper authentication is required for users to access any of the web pages including the home page.

4. Microsoft .NET framework ensures security of data, such as passwords that are being transmitted over the Internet, and there is proper authentication while accessing the system.

2.8.2 Software Quality Attributes

Reliability: Data, as entered, must be correctly stored in the database. In addition, data must be retrieved as entered and saved on the web page.

Maintainability: The code and design need to be documented in detail and designed in a way that a new project member with the same amount of academic and co-op experience can easily ramp up the project.
Availability: The application needs to be available 24/7 as the customer needs to know whom to contact if there is a critical issue.
Chapter 3
SYSTEM DESIGN AND ARCHITECTURE

This project is implemented using the DotNetNuke architecture. The application is hosted on a web server and responds to clients per the request. Web applications deliver content from the server to the client. Clients view the content from the Internet using a web browser. In a multitier client/server system [4], the client application logic is run in two locations:

- The thin client is run on the user's local computer and is focused on displaying results to the user.
- The business logic is located in server applications running on a server. Thin clients request functions from the server application, which is itself a multithreaded application capable of working with many concurrent users. The server application is the one that opens connections to the database server and can be running on the same server as the database, or it can connect across the network to a separate server operating as a database server.

For example, a server application can run on a Microsoft Internet Information Services (IIS) and service thousands of thin clients running on the Internet or an intranet. The server application uses a pool of connections to communicate with a copy of SQL Server. SQL Server can be installed on the same computer as IIS, or it can be installed on a separate server in the network.
3.1 High Level Design

The High Level Design (HLD) describes all the design components of the system and their interaction with one another. The HLD represents a decomposition of all the sub-systems down to the class level. This provides the basis for the detailed design efforts. A high-level design document will usually include a high-level architecture diagram depicting the components, interfaces, and networks that need to be further specified or developed. High-level design should address core functionality by identifying and defining key classes, their responsibilities, behavior, and attributes. High-level design should incorporate object-oriented design principles such as encapsulation, information hiding, and high cohesion. The architecture diagram is shown in Figure 3.2.
3.2 Data Flow Diagram

A data flow diagram represents the flow of data through an information system. A data flow diagram defines the inputs and outputs to the system and where the data is...
stored. A DFD is a visual representation of the processes, data stores, data flow, and external entities. This section has all the data flow diagrams for this application.

3.2.1 On-Call Rotation

This module discusses the data flow for on-call rotation. The user can create a new schedule, add additional entries to the schedule, or can perform edit and delete operations. The user then updates his/her changes and the changes will be saved to the database. Data will be displayed from the database to the user. Figure 3.2.1 shows the data flow diagram for the on-call rotation schedule.
Figure 3.2.1 – Data flow diagram for on-call rotation
3.2.2 Send Page

This module discusses the data flow of the send page interface. Here, the user selects a team to which he/she needs to send a page based on the time zone. The person types the message in the message box and then sends the page. The page is sent to the on-call person’s mailbox. Figure 3.2.2 shows the data flow diagram for the Sending Page module.

Figure 3.2.2 – Data flow diagram for Sending Page
3.2.3 Event Calendar

This module discusses the event calendar. The event calendar is used by the user to create a new event. The event can be of any type such as a team member’s vacation, downtime dates, go-live dates of a project, etc. The user can also edit and delete the event, then update the changes. All the changes are saved and stored in the database. Figure 3.2.3 shows the data flow diagram for the event calendar module.

Figure 3.2.3 – Data flow diagram for event calendar
3.3 Database Design

Database design plays an important role in the software development process and in improving the performance of the system. Database design facilitates data retrieval, storage, and modification. Advantages of using the databases include:

- **Integrity**
  The key feature of the databases is that the data is stored as a single logical unit and distributed across different physical locations without the need of duplicating the data, thus, maintaining integrity. The data can be manipulated according to the needs of the developer by querying the data.

- **Sharing and Security**
  Database design facilitates more control over how the data is managed across the system. Thus, the data can be shared and secured among users of the application as the data is maintained as a single repository.

Figure 3.3 depicts the database design for this project.
Figure 3.3 – Database design
A detailed description of the tables used in this project follows.

3.3.1 On_Call_Rotation

This is the main table from which the application gets the data to be displayed to the user. This table consists of the following columns: on_call_rot_ID, yr, ww, UserID, RoleID, on_call_rot_inact_dtm

Table 3.3.1 – On Call Rotation

<table>
<thead>
<tr>
<th>Column Name</th>
<th>Description</th>
<th>Data Type</th>
<th>Constraints</th>
</tr>
</thead>
<tbody>
<tr>
<td>on_call_rot_ID</td>
<td>Gives the unique ID of the on-call person</td>
<td>Int</td>
<td>Primary Key</td>
</tr>
<tr>
<td>yr</td>
<td>The year when the person is scheduled to be on call</td>
<td>Int</td>
<td></td>
</tr>
<tr>
<td>ww</td>
<td>The workweek when the person is scheduled to be on call</td>
<td>Int</td>
<td></td>
</tr>
<tr>
<td>UserID</td>
<td>The ID assigned to the on-call person</td>
<td>Int</td>
<td>Foreign Key</td>
</tr>
<tr>
<td>RoleID</td>
<td>The ID assigned to the role of the on-call person based on the geographical area of the on-call person</td>
<td>Int</td>
<td>Foreign Key</td>
</tr>
<tr>
<td>on_call_rot_inact_dtm</td>
<td>Date of when the person is scheduled to be on call</td>
<td>DateTime</td>
<td></td>
</tr>
</tbody>
</table>

3.3.2 Roles

This table has the information about the description of the roles, role names, and associated role group. This table consists of the following columns: RoleID, PortalID, RoleName, Description, RoleGroupID, CreatedByUserID, CreatedOnDate, LastModifiedByUserID, LastModifiedOnDate.
### Table 3.3.2 – Roles

<table>
<thead>
<tr>
<th>Column Name</th>
<th>Description</th>
<th>Data Type</th>
<th>Constraints</th>
</tr>
</thead>
<tbody>
<tr>
<td>RoleID</td>
<td>The unique ID for each role</td>
<td>Int</td>
<td>Primary Key</td>
</tr>
<tr>
<td>PortalID</td>
<td>The unique ID for each portal</td>
<td>Int</td>
<td></td>
</tr>
<tr>
<td>RoleName</td>
<td>The name of the specific role</td>
<td>nvarchar</td>
<td></td>
</tr>
<tr>
<td>Description</td>
<td>Description of the role</td>
<td>nvarchar</td>
<td></td>
</tr>
<tr>
<td>RoleGroupID</td>
<td>The ID for each role group</td>
<td>Int</td>
<td></td>
</tr>
<tr>
<td>CreatedByUserID</td>
<td>User ID of the user who created the role</td>
<td>Int</td>
<td></td>
</tr>
<tr>
<td>CreatedOnDate</td>
<td>Date when the role was created</td>
<td>DateTime</td>
<td></td>
</tr>
<tr>
<td>LastModifiedUserID</td>
<td>User ID of the user who last modified the role</td>
<td>Int</td>
<td></td>
</tr>
<tr>
<td>LastModifiedOnDate</td>
<td>Date when the role was last modified</td>
<td>DateTime</td>
<td></td>
</tr>
</tbody>
</table>

#### 3.3.3 Users

This table has the information about the on-call personnel, such as first name, last name, and display name. This table has the following columns: UserID, UserName, FirstName, LastName, IsSuperUser, Email, DisplayName, UpdatePassword, CreatedByUserID, CreatedOnDate, LastModifiedUserID, LastModifiedOnDate.
Table 3.3.3 – Users

<table>
<thead>
<tr>
<th>Column Name</th>
<th>Description</th>
<th>Data Type</th>
<th>Constraints</th>
</tr>
</thead>
<tbody>
<tr>
<td>UserID</td>
<td>Unique ID of the on-call person</td>
<td>Int</td>
<td>Primary Key</td>
</tr>
<tr>
<td>Username</td>
<td>Name of the on-call person</td>
<td>nvarchar</td>
<td></td>
</tr>
<tr>
<td>FirstName</td>
<td>First name of the on-call person</td>
<td>nvarchar</td>
<td></td>
</tr>
<tr>
<td>LastName</td>
<td>Last name of the on-call person</td>
<td>nvarchar</td>
<td></td>
</tr>
<tr>
<td>IsSuperUser</td>
<td>Is the user a super user</td>
<td>bit</td>
<td></td>
</tr>
<tr>
<td>Email</td>
<td>Email of the on-call person</td>
<td>nvarchar</td>
<td></td>
</tr>
<tr>
<td>DisplayName</td>
<td>Display name of the on-call person</td>
<td>nvarchar</td>
<td></td>
</tr>
<tr>
<td>UpdatePassword</td>
<td>If password is updated</td>
<td>bit</td>
<td></td>
</tr>
<tr>
<td>CreatedByUserID</td>
<td>User ID of the user who created the user</td>
<td>Int</td>
<td></td>
</tr>
<tr>
<td>CreatedOnDate</td>
<td>Date when the user was created</td>
<td>DateTime</td>
<td></td>
</tr>
<tr>
<td>LastModifiedUserID</td>
<td>User ID of the user who last modified the user</td>
<td>Int</td>
<td></td>
</tr>
<tr>
<td>LastModifiedOnDate</td>
<td>Date when the user was last modified</td>
<td>DateTime</td>
<td></td>
</tr>
</tbody>
</table>

3.3.4 UserRoles

This table associates Users and Roles. It has the information about UserID and RoleIDs. This table has the following columns: UserRoleID, UserID, RoleID, ExpiryDate, EffectiveDate, CreatedByUserID, CreatedOnDate, LastModifiedByUserID, LastModifiedOnDate.
Table 3.3.4 – UserRoles

<table>
<thead>
<tr>
<th>Column Name</th>
<th>Description</th>
<th>Data Type</th>
<th>Constraints</th>
</tr>
</thead>
<tbody>
<tr>
<td>UserRoleID</td>
<td>Unique ID of the user for the specific role</td>
<td>Int</td>
<td>Primary Key</td>
</tr>
<tr>
<td>UserID</td>
<td>Unique ID of the on-call person</td>
<td>Int</td>
<td>Foreign Key</td>
</tr>
<tr>
<td>RoleID</td>
<td>Unique ID of the role</td>
<td>Int</td>
<td>Foreign Key</td>
</tr>
<tr>
<td>ExpiryDate</td>
<td>Date when the user account will be expired for that role</td>
<td>DateTime</td>
<td></td>
</tr>
<tr>
<td>EffectiveDate</td>
<td>Date when user account will be effective for that role</td>
<td>DateTime</td>
<td></td>
</tr>
<tr>
<td>CreatedUserID</td>
<td>User ID of the user who assigned the on-call person for the specific role</td>
<td>Int</td>
<td></td>
</tr>
<tr>
<td>CreatedOnDate</td>
<td>Date when the on-call person was assigned for the specific role</td>
<td>DateTime</td>
<td></td>
</tr>
<tr>
<td>LastModifiedUserID</td>
<td>User ID of the user who last modified the on-call person for the specific role</td>
<td>Int</td>
<td></td>
</tr>
<tr>
<td>LastModifiedOnDate</td>
<td>Date when the on-call person was last modified for the specific role</td>
<td>DateTime</td>
<td></td>
</tr>
</tbody>
</table>

3.3.5 RoleGroups

This table has the information about the role groups. All the roles are categorized into specific role groups. This table has the following column names: RoleGroupID, PortalID, RoleGroupName, Description, CreatedUserID, CreatedOnDate, LastModifiedUserID, LastModifiedOnDate.
Table 3.3.5 – RoleGroups

<table>
<thead>
<tr>
<th>Column Name</th>
<th>Description</th>
<th>Data Type</th>
<th>Constraints</th>
</tr>
</thead>
<tbody>
<tr>
<td>RoleGroupID</td>
<td>Unique ID of the Role Group</td>
<td>Int</td>
<td>Primary Key</td>
</tr>
<tr>
<td>PortalID</td>
<td>Unique ID of the portal</td>
<td>Int</td>
<td></td>
</tr>
<tr>
<td>RoleGroupName</td>
<td>Name of the role group</td>
<td>nvarchar</td>
<td></td>
</tr>
<tr>
<td>Description</td>
<td>Description of the role group</td>
<td>nvarchar</td>
<td></td>
</tr>
<tr>
<td>CreatedByUserID</td>
<td>User ID of the user who created the role group</td>
<td>Int</td>
<td></td>
</tr>
<tr>
<td>CreatedOnDate</td>
<td>Date when the role group was created</td>
<td>DateTime</td>
<td></td>
</tr>
<tr>
<td>LastModifiedByUserID</td>
<td>User ID of the user who last modified the role group</td>
<td>Int</td>
<td></td>
</tr>
<tr>
<td>LastModifiedOnDate</td>
<td>Date when the role group was last modified</td>
<td>DateTime</td>
<td></td>
</tr>
</tbody>
</table>

3.3.6 UserAuthentication

This table has the information about the authentication details for every user. This table has the following columns: UserAuthenticationID, UserID, AuthenticationType, AuthenticationToken, CreatedByUserID, CreatedOnDate, LastModifiedByUserID, LastModifiedOnDate.

Table 3.3.6 – User Authentication

<table>
<thead>
<tr>
<th>Column Name</th>
<th>Description</th>
<th>Data Type</th>
<th>Constraints</th>
</tr>
</thead>
<tbody>
<tr>
<td>UserAuthenticationID</td>
<td>Unique ID of the User Authentication</td>
<td>Int</td>
<td>Primary Key</td>
</tr>
<tr>
<td>UserID</td>
<td>Unique ID of the user</td>
<td>Int</td>
<td>Foreign Key</td>
</tr>
<tr>
<td>AuthenticationType</td>
<td>Type of Authentication</td>
<td>nvarchar</td>
<td></td>
</tr>
<tr>
<td>AuthenticationToken</td>
<td>The authentication token generated for the username and password</td>
<td>nvarchar</td>
<td></td>
</tr>
<tr>
<td>CreatedByUserID</td>
<td>User ID of the user who created the authentication</td>
<td>Int</td>
<td></td>
</tr>
<tr>
<td>CreatedOnDate</td>
<td>Date when the authentication was created</td>
<td>DateTime</td>
<td></td>
</tr>
<tr>
<td>LastModifiedByUserID</td>
<td>User ID of the user who last modified the authentication</td>
<td>Int</td>
<td></td>
</tr>
<tr>
<td>LastModifiedOnDate</td>
<td>Date when the authentication was last modified</td>
<td>DateTime</td>
<td></td>
</tr>
</tbody>
</table>
Chapter 4
IMPLEMENTATION

The implementation phase is the key phase and final phase of the project cycle. The requirements of the project are achieved with the proposed design through the implementation phase. This section describes the technologies and platforms used to implement this project and user interface with the screenshots.

4.1 Overview of Technologies

ASP.NET Framework:

The .NET Framework [1] is an integral Windows component that supports building and running the next generation of applications and XML Web services. The .NET Framework is designed to fulfill the following objectives:

- To provide a consistent object-oriented programming environment whether object code is stored and executed locally, executed locally but Internet-distributed, or executed remotely.
- To provide a code-execution environment that minimizes software deployment and versioning conflicts.
- To provide a code-execution environment that promotes safe execution of code, including code created by an unknown or semi-trusted third party.
- To provide a code-execution environment that eliminates the performance problems of scripted or interpreted environments.
• To make the developer experience consistency across widely varying types of applications, such as Windows-based applications and web-based applications.

• To build all communication on industry standards to ensure that code based on the .NET Framework can integrate with any other code.

Figure 4.1.1 – ASP.NET Architecture

The .NET Framework consists of two main components: Common Language Runtime and .NET Framework class library.
Common Language Runtime: The core runtime engine in the Microsoft .Net framework for executing various applications. The CLR supplies the managed code with services like cross-language integration, code access security, object lifetime management, resource management, type safety, Meta data, debugging memory management, thread management, and remoting, while also enforcing strict type safety and other forms of code accuracy that promote security and robustness. Asp.net and Internet Explorer are the examples of hosting CLRs. The CLR provides a solid foundation for developers to build various applications like the asp.net application, windows form application, web services, the Mobile code application, and the distributed application. It provides a seamless integration of code written in various languages. It is a multi-executing environment. The CLR is responsible for compiling .NET applications to native machine code.

.NET Framework Class Library: The .NET Framework class library is a collection of reusable types that tightly integrate with the common language runtime. The class library is object-oriented, providing types from which your own managed code can derive functionality. This not only makes the .NET Framework types easy to use, but also reduces the time associated with learning new features of the .NET Framework. In addition, third-party components can integrate seamlessly with classes in the .NET Framework. For example, the .NET Framework collection classes implement a set of interfaces we can use to develop our own collection classes. These collection classes will blend seamlessly with the classes in the .NET Framework.
DotNetNuke:

DotNet nuke [2] is one of the Microsoft’s Web Content Management Systems for ASP.NET. DotNetNuke platform functions as both a Web Content Management System and an Application Development Framework. It allows a user to create interactive and highly dynamic websites for any industry or size of company. DotNetNuke runs on windows server 2008, Microsoft .Net, Microsoft ASP.NET and uses the Microsoft SQL Server. One instance of DotNetNuke supports an unlimited number of portals and each portal can have an unlimited number of users and pages. DotNetNuke has a strong three-tiered architecture, comprised of the Presentation layer, the Business layer, and the Database layer with a core framework supporting extensible modular architecture. There is a clear separation between the layers.
Microsoft SQL Server is an application used to create computer databases for the Microsoft Windows family of server operating systems. Microsoft SQL Server provides an environment used to generate databases that can be accessed from workstations, the Internet, or other media such as a personal digital assistant (PDA). It can also be viewed as a computer application used to create desktop, enterprise, and web-based database applications.

Its primary query languages are T-SQL and ANSI SQL. It includes native support for managing XML data, in addition to relational data. For this purpose, it defined an xml data type that could be used either as a data type in database columns or as literals in queries. SQL Server 2008 allows embedding XQuery queries in T-SQL.
2008 also allows a database server to be exposed over web services via SOAP requests. When the data is accessed over web services, results are returned as XML.

For relational data, T-SQL has been augmented with error handling features (try/catch) and support for recursive queries with CTEs (Common Table Expressions). SQL CLR was introduced with SQL Server 2008 to let it integrate with the .NET Framework.
4.2 User Interface

This section describes the user interface of this project through the screenshots:

![User View Screen](image)

**Figure 4.2 – User View Screen**
4.2.1 User View Screen

This is the view-only screen for users and team members. This screen displays the on-call rotation schedule for both US and India regions for 13 workweeks, which includes a workweek prior to the current workweek, the current workweek, and 11 weeks after the current workweek. Team members who have administrative rights can only see the “Manage On-call Schedule” button so they can manage. If there are no records to be shown, the screen will be displayed with the message saying, “There are no records to display” for that geographical area as shown above. If there no records other than the ones displayed, the next button will be grayed out as shown in Figure 4.2. The same is the case with the previous button.

![Image of User View Screen]

Figure 4.3 – Manage Screen
4.2.2 Manage Screen

The screenshot in Figure 4.3 is the manage screen where the on-call rotation person can manage the schedule which includes edit, delete, and update operations. He/she can also add more entries to the schedule using the “Add More Items” button shown in the figure.

Figure 4.4 – Manage – Edit Screen
4.2.3 Manage –Edit Screen

When the on-call person clicks edit on the manage screen, the line item changes to the edit mode and the Name field appears as a drop down menu from which the on-call person can choose to assign any other person that corresponding workweek. The edit text will be changed to update so the on-call person can update the changes. Once he/she updates the schedule, all changes will be saved to the database.

![Figure 4.5 – Manage - Delete Screen A](image)

4.2.4 Manage –Delete Screen

The right-hand part of the screenshot in Figure 4.5 depicts that the on-call person is about to hit the delete button. When the on-call person hovers the mouse over the delete button, the button is underlined. When he/she clicks the button, the entry will be deleted, and the empty grid will allow the addition of more items being displayed, as shown to the right in Figure 4.6.
Figure 4.6 – Manage - Delete Screen B

Figure 4.7 – Manage - Add Items to Empty Grid Screen
4.2.5 Manage – Add Items to Empty Grid Screen

The screenshot in Figure 4.7 depicts inserting items onto a blank grid. The empty grid template contains all the fields as drop down menus allowing the on-call person to choose the workweeks and the on-call persons for those corresponding workweeks. After selecting the items, he/she can click the insert button to add items to the empty grid. Once the insert button is clicked, all changes will be saved to the database.

Figure 4.8 – Manage - Add More Items Screen
4.2.6 Manage – Add More Items Screen

The screenshot in Figure 4.8 depicts the ability to add more items to the existing schedule. When the on-call person clicks the “Add More Items” button on the screen, a new row will be displayed with the drop down menus for workweeks and the name fields at the end of the grid as shown in the figure. The on-call person can choose from the options in the drop down list and hit the “insert” button to insert the entry. Once he/she inserts the changes, they will be saved to the database. If he/she wishes to not add more items after clicking the “Add More Items” button, he/she can click the “Cancel” button and the grid will be displayed with the original schedule.

Figure 4.9 – Security Roles Screen
4.2.6 Security Roles Screen

The screen in Figure 4.9 shows the security roles, which include adding new role groups, adding new roles to existing role groups, and user settings.

![Security Roles Screen](image)

Figure 4.10 – Filter By Role Group Screen

4.2.8 Filter By Role Group Screen

The screen in Figure 4.10 shows all the roles defined for the role group named “KTBR Group.” “India EIE KTBR On Call” and “US EIE KTBR On Call” are the roles added to the “KTBR Group” Role Group. The administrator can add new roles to the KTBR Group or add a new role group from this screen.
4.2.9 Add New Role Group Screen

The screen in Figure 4.11 is displayed when the user clicks “Add New Role Group” on the Security Roles page. The administrator can name the role group and add a brief description of the role. A new role group will be created once he/she clicks the update button. The new role group will be saved to the database.
4.2.10 Add New Roles Screen

The screen in Figure 4.12 is displayed when the user clicks “Add New Role” on the security roles page. The administrator can name a new role, add a brief description, and assign that role to an existing role group on this screen. A new role will be created after he/she clicks the update button. The new role will be saved to the database.
User Settings

Membership Provider Settings

In this section you can review the membership provider's settings. Some providers may allow you to edit the settings. The default provider based on the ASP.NET MemberRole requires you to edit web.config, so the settings can not be updated here.

- Requires Unique Email: [ ]
- Password Format: Encrypted
- Password Retrieval Enabled: ✓
- Password Reset Enabled: ✓
- Minimum Password Length: 7
- Minimum No of Non AlphaNumeric Characters:
- Requires Question and Answer: [ ]
- Password Regular Expression:
- Maximum No of Invalid Attempts: 5
- Invalid Attempt Window (in mins): 10

Password Aging Settings

You can set up Password aging settings in this section.

- Password Expiry (in days): 0
- Password Expiry Reminder (in days): 7

User Accounts Settings

- Show Address Column: ☑ True ☐ False
- Show Authorized Column: ☑ True ☐ False

Figure 4.13 – User Settings Screen
4.2.11 User Settings Screen

The above screen depicts all the settings an administrator can set for the users with a specific role. These settings will be applied to all the users for that role.

![User Settings Screen](image)

Figure 4.14 – Add Users Screen
4.2.12 Add Users Screen

The screenshot in Figure 4.14 depicts adding new users with the dotnetnuke interface. Only administrators have the access to add new users and grant them necessary roles and access. An administrator can add a new user by clicking the “Add New User” button shown in the above screen. Once he/she clicks the “Add New User” button, the screen in Figure 4.15 will be displayed.
4.2.13 Add New User Form Screen

The administrator will be navigated to the screen in Figure 4.15 upon clicking the “Add New User” button. He/she can fill in the fields and can choose to authorize a user if needed. He/she can also choose a password for the user when setting up the account. Otherwise, a system-generated password will be provided.
4.2.14 Edit User Account Screen

The edit functionality is achieved by clicking the little pencil symbol next to the user. Hovering the mouse on the pencil symbol shows the cursor as “Edit” as shown in Figure 4.16. Clicking the pencil symbol will enable the edit functionality, and the administrator can perform various edit operations. These functionalities are discussed further in the screenshots that follow.
4.2.15 Edit User Account – Manage User Credentials Screen

The screenshot in Figure 4.17 depicts editing the user credentials functionality. Only administrators will have access to editing the user accounts. He/she can edit any fields shown in the above figure and click the “update” button to update the user account. The changes will be saved to the database. He/she can also manage roles for the user by clicking the “Manage Roles for this User” button, can manage the user’s password by clicking the “Manage Password” button, or can manage the user’s profile by clicking the “Manage Profile” button. These screens are shown in Figures 4.18-4.20.
4.2.16 Edit User Account – Manage Roles Screen

The screenshot in Figure 4.18 depicts managing roles for the user functionality. The administrator can assign any of the listed roles in the drop down shown in Figure 4.18. He/She can choose any of the listed roles in the drop down menu in the above screen and enter the effective date indicating the date from when the role will be effective for the user. The expiry date indicates the date when the role will expire for the user. Clicking “Add Role to User” will add the role to the user and the changes will be saved to the database.
4.2.17 Edit User Account – Manage Password Screen

The screenshot in Figure 4.19 depicts the managing user’s password functionality. The administrator can change the old password and assign a new password by entering the new password in the appropriate field and clicking “Change Password” as shown in the above figure. Alternatively, he/she can choose to reset the password and the system-generated password will be provided. All the changes will be saved to the database.
4.2.18 Edit User Account – Manage Profile Screen

The screenshot in Figure 4.20 depicts the Managing User’s Profile functionality. The administrator can add or edit the fields, add a photo, add a biography and click the “Update” button to save the changes. All the changes will be saved to the database.
Figure 4.21 – Send Page Screen

Send Page Screen

The screenshot in Figure 4.21 depicts the functionality of sending a page to the respective team through the web interface. The user can select the team from the choices listed in the drop down menu, type a message in the message box, and then click “Send Page” to send a page to the on-call person.
Figure 4.22 – Pager Information Screen

4.2.20 Pager Information Screen

The screenshot in Figure 4.22 shows how the user sees the pager information. The user will know who to call depending upon the geographical area from this screen.
4.2.21 Event Calendar Screen

The screenshot in Figure 4.23 shows the event calendar. Team members can add any event, such as their vacation, to the calendar, which can be viewed by anyone within the team. He/she can also edit the event. Adding events and editing the events is shown in the screenshots in Figures 4.24-4.26.
4.2.22 Event Description Screen

The screen in Figure 4.24 shows the brief event description including dates of the event when the user hovers the mouse on the event on the calendar.
4.2.23 View Event Detail

The screenshot in Figure 4.25 depicts the view event functionality. When the team member clicks the event on the calendar, the above screen will be displayed with all the details, which include start date, end date, recurrence, importance, category, and description of the event. The team member can also edit, delete, or go to the home screen from this page.
4.2.24 Add/ Edit Event Screen

The screenshot in Figure 4.26 depicts the add event functionality. When the team member clicks the little plus symbol on top of the calendar on the home screen, or when he/she clicks the “Edit” button on the view event page, he/she will be navigated to the above screen. The team member can fill in the details in the above screen and then click the “Update” button to add an event to the calendar or update the changes for that event.
4.2.25 Delete Event

The screenshot in Figure 4.27 depicts the delete functionality of an event. When the team member clicks the “Delete” button on the view event page (see Figure 4.26), a dialog box appears to confirm deleting the event. Upon clicking “OK,” the event will be deleted from the calendar.
REFERENCES


   http://edn.embarcadero.com/article/10343