Itimecard timesheet management system

Arpita Prashant Parikh

Follow this and additional works at: https://scholarworks.lib.csusb.edu/etd-project

Part of the Databases and Information Systems Commons

Recommended Citation
https://scholarworks.lib.csusb.edu/etd-project/3421

This Project is brought to you for free and open access by the John M. Pfau Library at CSUSB ScholarWorks. It has been accepted for inclusion in Theses Digitization Project by an authorized administrator of CSUSB ScholarWorks. For more information, please contact scholarworks@csusb.edu.
ITIMECARD TIMESHEET MANAGEMENT SYSTEM

A Project
Presented to the
Faculty of
California State University,
San Bernardino

In Partial Fulfillment
of the Requirements for the Degree
Master of Science
in
Computer Science

by
Arpita Prashant Parikh
March 2008
ITIMECARD TIMESHEET MANAGEMENT SYSTEM

A Project
Presented to the
Faculty of
California State University,
San Bernardino

by
Arpita Prashant Parikh
March 2008

Approved by:

Dr. David Turner, Advisor

Dr. Richard J. Botting

Dr. Ernesto Gomez

3/10/2008
ABSTRACT

In the era of opportunistic expansion, small companies are heading towards expansion and development. In such a period, efficient data processing and management have always aided the companies to progress towards their goal. This project is one of the very few efficient ways of processing time sheets.

Computerized submissions of time sheets make it easy for the company to generate a paycheck without having to fill out numerous forms. It also enables the company to know right away if there are any missing time sheets or errors in the time sheets and therefore an employee could be notified immediately. The employer or administrator of the system only has to follow a few steps to set up an employee and project, with the help of the user-friendly interface of this system. Once set up is complete, the administrator can easily sign the submitted time sheet of an employee, which expedites the whole process.

This system is very user-friendly, provides high system availability, easy access to timesheet, and high level of security to the employee. An employee also follows
only a few steps to fill out the time sheet using a web browser, which is most convenient.
ACKNOWLEDGEMENTS

I am grateful for the assistance of the staff members of the Computer Science Department of CSUSB who provided guidance on the study, in particular Dr. David Turner, Dr. Richard Botting, Dr. Ernesto Gomez and Dr. Jospine Mendoza. I would like to specially thank my advisor, Dr. David Turner for giving me the inspiration and direction to write this paper.

My sincere thanks to the following employees of iSPACE: Vinod Kottapalli, who helped me with Microsoft InfoPath and Microsoft SharePoint, and Rushi Patel, who taught me ASP.NET and Microsoft SQL Server. Also, I would thank iSPACE for their generosity in letting me use their computing resources for conducting my project results, and providing me with a rewarding professional experience. Finally, I would like to thank my family for supporting me and giving me courage.
# TABLE OF CONTENTS

ABSTRACT......................................................................................... iii

ACKNOWLEDGEMENTS........................................................................ v

LIST OF TABLES.................................................................................... viii

LIST OF FIGURES.................................................................................. x

CHAPTER ONE: INTRODUCTION

1.1 Rationale and Importance of Project.............. 1
1.2 Background............................................................... 2
1.3 Current Business Process......................... 3

CHAPTER TWO: TECHNICAL REQUIREMENT SPECIFICATION

2.1 Use Case Analysis.............................. 6
2.2 Conceptual Model.............................. 28
2.3 Error Handling................................. 29
2.4 System Security................................. 30

CHAPTER THREE: TOOLS AND DEPENDENCIES

3.1 Microsoft InfoPath......................... 32
3.2 Microsoft ASP.NET......................... 33
3.3 Microsoft SQL Server......................... 34
3.4 Microsoft SharePoint......................... 35

CHAPTER FOUR: THE ITIMECARD SOFTWARE SYSTEM

4.1 Overview......................................................... 37
4.2 Architecture......................................................... 38
4.3 Database Design......................................................... 39
CHAPTER FIVE: TESTING

5.1 Unit/System Testing .............................. 44
5.2 Problems ........................................ 71

CHAPTER SIX: SYSTEM ADMINISTRATION MANUAL

6.1 Server Setup ..................................... 74
6.2 Client Setup ..................................... 80

CHAPTER SEVEN: END USER MANUAL

7.1 Accessing the iTimeCard System ................. 81
7.2 Submitting Timesheet ............................. 84

CHAPTER EIGHT: ADMIN USER MANUAL

8.1 Managing Submitted Timesheet .................. 87
8.2 Managing Employee Data ......................... 90
8.3 Managing Project Data ........................... 98

CHAPTER NINE: CONCLUSION

9.1 Summary ........................................... 104
9.2 Future Directions ................................. 105

APPENDIX: APPLICATION CODE ....................... 106

REFERENCES .......................................... 123
LIST OF TABLES

Table 1. Steps to Add a New Employee ......................... 8
Table 2. Steps to Add a New Project ......................... 11
Table 3. Steps to View Employee ......................... 13
Table 4. Steps to View Project ......................... 14
Table 5. Steps to Delete a Project ......................... 16
Table 6. Steps to Delete an Employee ......................... 17
Table 7. Steps to Edit Employee Data ......................... 18
Table 8. Steps to Edit Project Data ......................... 20
Table 9. Steps to Manage Submitted Timesheet ......................... 22
Table 10. Steps to Create a Timecard ......................... 23
Table 11. Steps to Submit a Timesheet ......................... 26
Table 12. Employee Table Description ......................... 41
Table 13. Project Table Description ......................... 42
Table 14. ProjectEmployee Table Description ......................... 43
Table 15. Test Case of Add Employee ......................... 45
Table 16. Test Case of Add Project ......................... 49
Table 17. Test Case of View Employee ......................... 52
Table 18. Test Case of View Project ......................... 54
Table 19. Test Case of Delete Employee ......................... 56
Table 20. Test Case of Delete Project ......................... 57
Table 21. Test Case of Edit Employee Data ......................... 58
Table 22. Test Case of Edit Project Data ......................... 62
Table 23. Test Case of Manage Submitted Timesheet...... 65
Table 24. Test Case of Create Timecard.................. 66
Table 25. Test Case of Add Employee.................... 69
LIST OF FIGURES

Figure 1. Current Timesheet.......................... 3
Figure 2. Timesheet Submission Process.................. 5
Figure 3. Use Case Diagram................................. 7
Figure 4. Entity-Relationship Diagram....................... 28
Figure 5. Try Catch Block.................................. 29
Figure 6. System Flow Diagram............................... 38
Figure 7. Database Diagram.................................. 40
Figure 8. Microsoft SharePoint Setup Error Step 1........... 71
Figure 9. Microsoft SharePoint Setup Error Step 2........... 72
Figure 10. Microsoft SharePoint Setup Step 1............... 75
Figure 11. Microsoft SharePoint Setup Step 2................. 76
Figure 12. Microsoft SharePoint Setup Step 3................. 77
Figure 13. Microsoft SharePoint Setup Step 4................. 78
Figure 14. Microsoft SharePoint Setup Step 5................. 79
Figure 15. Accessing the iTimesCard Home Page............. 81
Figure 16. Select a Project to Access the iTimesCard...... 83
Figure 17. Submitting the iTimesCard Screen Step 1........ 85
Figure 18. Submitting the iTimesCard Screen Step 2........ 86
Figure 19. Managing Submitted iTimesCard Screen.......... 88
Figure 20. Submitted iTimesCard List....................... 89
Figure 21. Managing Employee Detail Home Page............. 90
Figure 22. View Employee Screen............................ 92
Figure 23. Add Employee Screen........................... 94
Figure 24. Record Save Message............................... 95
Figure 25. Edit Employee Screen............................... 97
Figure 26. Managing Project Detail Home Page............. 98
Figure 27. View Project Screen................................. 99
Figure 28. Add a New Project Screen....................... 101
Figure 29. Edit Project Screen................................. 103
CHAPTER ONE

INTRODUCTION

1.1 Rationale and Importance of Project

The project was proposed by iSpace, an IT and Business Process Outsourcing (BPO) solutions company. There are four major purposes in why I accepted the project. The first purpose was to gain experience with Microsoft SharePoint. Using SharePoint, I performed research on how to develop a website and how to store documents on the website. The second purpose was to gain experience with Microsoft Visual Studio to develop websites in .NET using C# language. The third purpose was to gain experience with Microsoft SQL Server. The fourth purpose was to gain experience with Microsoft InfoPath. By working on this project I not only got a chance to learn new and demanding technologies but also gained experience in various facets of the software life cycle such as requirement gathering, designing, development and testing.

This project will help iSpace automate their timesheet submission process by storing timesheets electronically rather than storing them in the paper form. This project
also allows employees to resubmit or modify their timesheets in case of any change in their respective timesheets.

1.2 Background

Headquartered at Los Angeles, iSpace is an IT and Business Process Outsourcing (BPO) solutions company serving fortune 1000 companies in healthcare, technology, government, retail, entertainment, automotive and transportation industries. iSpace has more than 400 employees working for clients worldwide.

iSpace employees need to submit their timesheets every week to the accounting department. The timesheets need to pass through an approval process as well.

Currently, paper timesheets are manually approved and submitted to the accounting department either via fax or email. The accounting department needs to manually collect and organize all the timesheets from all the employees. This is a time consuming task that needs lots of follow up emails, reminders and filing paper timesheets.

In order to eliminate drawbacks associated with paper based transactions and to gain benefits of electronic
transactions, iSpace has decided to develop the iTimeCard System.

1.3 Current Business Process

iSpace employees submit their timesheets once in every week. Their current timesheet format is shown in Figure 1.

![Current Timesheet](image)

Figure 1. Current Timesheet

An Employee submits their timesheet for projects that belong to a client. In case an employee works on multiple
projects for the same client, only one timesheet needs to be submitted. In case an employee works on multiple projects for different clients, different timesheets per client need to be submitted.

On the timesheet an employee identifies the project name they are working on and the number of hours they have worked for each project.

Once the timesheet is filled out, an employee has to get a signature from his/her supervisor. An employee who works at a client site has to get a signature from their respective client manager.

Finally, an employee has to fax or email (the scanned copy) of the filled timesheet to the account department. Once the account department receives the timesheet, timesheet information is reviewed and recorded in the database. The account department also files an employee’s timesheet for future references. The paycheck is then issued to the employee biweekly.

Figure 2 describes current timesheet submission process.
Figure 2. Timesheet Submission Process
CHAPTER TWO

TECHNICAL REQUIREMENT SPECIFICATION

2.1 Use Case Analysis

The user community will consist of many people working on various locations and will be divided into two types of roles:

- Administrator (admin)
- Users

Admin user is responsible for managing Employees, Projects and submitted timesheets.

Users are employees of iSpace who will create, fill out and submit timesheets for payroll and billing purposes.
Figure 3. Use Case Diagram
Add a New Employee:

This use case deals with adding a new employee to the system who needs to submit a timesheet for payroll and billing purpose.

Table 1. Steps to Add a New Employee

<table>
<thead>
<tr>
<th>Step</th>
<th>User Action</th>
<th>Business Rules</th>
<th>System Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>User clicks on Add button on Employee List page</td>
<td>None</td>
<td>- Add new Employee page is displayed</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- All projects must be listed in Available Project list box</td>
</tr>
<tr>
<td>2</td>
<td>User selects projects to assign to</td>
<td></td>
<td>- The selected project must be added to</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>---</td>
<td>---</td>
<td>---</td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>employee (&gt; button)</strong></td>
<td>Assigned Project list box and removed from Available Project list box</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>User selects projects to unassigned to employee (&lt; button)</td>
<td>- The selected project must be added to Available Project list box and removed from Assigned Project list box</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>User clicks on save &amp; go back button</td>
<td>- Employee Name must not be blank - Email must not be blank - Message must be displayed in case business rules are validated - In case all the business rules must not be blank</td>
<td></td>
</tr>
<tr>
<td>Step</td>
<td>Description</td>
<td>Expected Outcome</td>
<td></td>
</tr>
<tr>
<td>------</td>
<td>-------------</td>
<td>------------------</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>User click on Go Back button</td>
<td>None</td>
<td></td>
</tr>
</tbody>
</table>

- View Employee list screen must be displayed
- View Employee page is displayed
- Will not save any changes made in any field

**Note:**
- Login ID must not be blank
- Message must be displayed to the user, stating that data is saved
- Data must be saved in database
Add a New Project:

This use case deals with adding a new project that an employee needs to work on.

Table 2. Steps to Add a New Project

<table>
<thead>
<tr>
<th>Step</th>
<th>User Action</th>
<th>Business Rules</th>
<th>System Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>User clicks on Add button on Project List page</td>
<td>None</td>
<td>- Add new Project page is displayed</td>
</tr>
</tbody>
</table>
| 2    | User clicks on save & go back button             | - Employee Name must not be blank
- Email must not be blank and should | - Message must be displayed in case business rules are validated
- In case all the business rules are met, data must be saved |
<table>
<thead>
<tr>
<th></th>
<th></th>
<th>be incorrect format</th>
<th>in database</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Login ID must not be blank</td>
<td>Message must be displayed to the user, stating that data is saved</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>View Project list screen must be displayed</td>
</tr>
<tr>
<td>3</td>
<td>User click on Clear button</td>
<td>None</td>
<td>Project Name, Client Name, Client Manager, Is Active and Is billable field should be empty</td>
</tr>
<tr>
<td>4</td>
<td>User click on Go Back</td>
<td>None</td>
<td>View Project page is displayed</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Will not save any changes</td>
</tr>
</tbody>
</table>
View employee:

This use case deals with viewing a list of employees using iTimeCard system.

Table 3. Steps to View Employee

<table>
<thead>
<tr>
<th>Step</th>
<th>User Action</th>
<th>Business Rules</th>
<th>System Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>User clicks on Employee link</td>
<td>None</td>
<td>- View Employee page is displayed</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- Employee name, Login ID, Email and IsRSM should be display in data grid</td>
</tr>
<tr>
<td>Step</td>
<td>User Action</td>
<td>Business Rules</td>
<td>System Response</td>
</tr>
<tr>
<td>------</td>
<td>------------------------------</td>
<td>----------------</td>
<td>-----------------------</td>
</tr>
<tr>
<td>2</td>
<td>User clicks on Delete link</td>
<td>None</td>
<td>- Refer to Delete Employee Use Case</td>
</tr>
<tr>
<td>3</td>
<td>User click on Add Employee button</td>
<td>None</td>
<td>- Refer to Add Employee Use Case</td>
</tr>
<tr>
<td>4</td>
<td>User click on Go Back</td>
<td>None</td>
<td>- Master page is displayed</td>
</tr>
</tbody>
</table>

View project:

This use case deals with viewing a list of projects available for Employees to be assigned for iT imeCard system.

Table 4. Steps to View Project
<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>User clicks on Delete link</td>
<td>None</td>
<td>- Refer to Delete Project Use Case</td>
</tr>
<tr>
<td>3</td>
<td>User click on Add Project button</td>
<td>None</td>
<td>- Refer to Add Project Use Case</td>
</tr>
<tr>
<td>4</td>
<td>User click on Go Back</td>
<td>None</td>
<td>- Master page is displayed</td>
</tr>
</tbody>
</table>

displayed
- Project Name,
Client Name,
Client Manager,
Is Active and
Is Billable
should be
display in data grid
Delete a project:

This use case deals with deleting a project in case project is wrongly entered in the system.

Table 5. Steps to Delete a Project

<table>
<thead>
<tr>
<th>Step</th>
<th>User Action</th>
<th>Business Rules</th>
<th>System Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>User click on Delete Link on the data grid of the Project list</td>
<td>None</td>
<td>- Should delete the selected project information - If project is assigned to employee then user should not able to delete the record</td>
</tr>
</tbody>
</table>
Delete an Employee:

This use case deals with deleting an employee in case employee is wrongly entered in the system or is no more with iSpace.

Table 6. Steps to Delete an Employee

<table>
<thead>
<tr>
<th>Step</th>
<th>User Action</th>
<th>Business Rules</th>
<th>System Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>User click on Delete Link on the data grid of the Employee list</td>
<td>None</td>
<td>- Should delete the selected employee information</td>
</tr>
</tbody>
</table>
Edit Employee information:

This use case deals with editing employee related information such as name, email or login id.

Table 7. Steps to Edit Employee Data

<table>
<thead>
<tr>
<th>Step</th>
<th>User Action</th>
<th>Business Rules</th>
<th>System Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>User click on Employee name link</td>
<td>None</td>
<td>- Will open the Update Employee page</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- All the current employee information should be fill up in the update employee page.</td>
</tr>
<tr>
<td>2</td>
<td>User clicks on save &amp; go back button</td>
<td>- Employee Name must not be blank</td>
<td>- Message must be displayed in case business rules are</td>
</tr>
<tr>
<td>- Email must not be blank and should be incorrect format</td>
<td>validated</td>
<td></td>
<td></td>
</tr>
<tr>
<td>- In case all the business rules are met, data must be saved in database</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>- Message must be displayed to the user, stating that data is saved</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>- View Employee list screen must be displayed</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Edit Project information:

This project use cases deals with editing project information such as project name, client name etc.

Table 8. Steps to Edit Project Data

<table>
<thead>
<tr>
<th>Step</th>
<th>User Action</th>
<th>Business Rules</th>
<th>System Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>User click on Project name link</td>
<td>None</td>
<td>- Will open the Update Project page</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- All the current project information should be fill up in the update employee page.</td>
</tr>
<tr>
<td>2</td>
<td>User clicks on save &amp; go back button</td>
<td>- Project Name must not</td>
<td>- Message must be displayed in case business</td>
</tr>
<tr>
<td>be blank</td>
<td>rules are validated</td>
<td></td>
<td></td>
</tr>
<tr>
<td>---</td>
<td>---</td>
<td></td>
<td></td>
</tr>
<tr>
<td>- Client Name</td>
<td>- In case all the business rules are met, data must be saved in database</td>
<td></td>
<td></td>
</tr>
<tr>
<td>must not be blank</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>be blank</td>
<td>- Message must be displayed to the user, stating that data is saved</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>- View Project list screen must be displayed</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Administration manages submitted timesheet:

This use case deals with managing submitted timesheets. The Admin can view timesheets submitted by employee, review them or delete them and ask for resubmission if required.

Table 9. Steps to Manage Submitted Timesheet

<table>
<thead>
<tr>
<th>Step</th>
<th>User Action</th>
<th>Business Rules</th>
<th>System Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Type the URL to open the page to access the submitted timesheet.</td>
<td>None</td>
<td>List of all submitted timesheet will be available to view</td>
</tr>
<tr>
<td>2</td>
<td>Admin click on Timesheet link</td>
<td>None</td>
<td>Submitted timesheet must be open in view mode</td>
</tr>
<tr>
<td>3</td>
<td>Delete the timesheet if any</td>
<td>None</td>
<td>TimeCard must be deleted</td>
</tr>
</tbody>
</table>
information is incorrect

Create a Timecard:

Each employee is required to create a timecard using an InfoPath form by selecting a project and date.

Table 10. Steps to Create a Timecard

<table>
<thead>
<tr>
<th>Step</th>
<th>User Action</th>
<th>Business Rules</th>
<th>System Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Type the URL to open the page to submit the Timesheet.</td>
<td>None</td>
<td>- Should open up the page</td>
</tr>
<tr>
<td>2</td>
<td>User click on the Timesheet link to open the Timesheet</td>
<td>None</td>
<td>- A timesheet should open using Window Authentication.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- Project should</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

3. Select the Project from drop down list.

4. Select the ending day for that week

5. Click on "Create the Timecard"

be populated into dropdown list box for each user.

Ending day cannot be greater than today's date

- Should open up timesheet page
- In case all the business rules are met.
- The employee name, Client Name and Project Name should be
- Carried forward to next page
- Initially all hours should be zero.
- Date should be filled out according to previous screen information.
- Submit button should be inactive.
Submitting the timesheet:

This use case deals with an employee submitting the timesheet once it is filled and complete. The submitted timesheet is emailed to accounts department and published on the SharePoint server.

Table 11. Steps to Submit a Timesheet

<table>
<thead>
<tr>
<th>Step</th>
<th>User Action</th>
<th>Business Rules</th>
<th>System Response</th>
</tr>
</thead>
</table>
| 1    | Users fills out hours they for selected project | - Total= hour work + Holiday + Paid Time Out
- Hours cannot be greater than 24 hrs | - Should show total according to business rule |
<p>| 2    | Check the checkbox regarding | - acknowledge that filled information | - Submit button will be activated |</p>
<table>
<thead>
<tr>
<th>acknowledgement</th>
<th>is correct</th>
</tr>
</thead>
</table>
| Click on submit button | - Send email to account department  
- Save the timesheet into SharePoint | - Meet the business rule mention. |
2.2 Conceptual Model

For the conceptual model I have used an E-R (Entity - Relationship) diagram.

![Diagram of Employee, Project, and TimeSheet entities with attributes]

Figure 4. Entity-Relationship Diagram

Employee is an entity type that contains attributes such as EmpID, FullName, LoginID, IsRSM and EmailID.
Project is an entity type that contains attributes such as ProjID, ProjectName, ClientName, ClientManager, IsActive and IsBillable.

TimeSheet is an entity type that contains attributes such as TimeSheetID, EndDate, RegHrs, PaidTimeOff and HolidayHrs.

An employee can work on more than one project. And a project can be assigned to more than one employee. This illustrates that an employee and a project has a many-to-many relationship.

2.3 Error Handling

The InfoPath iTimeCard form uses Visual Studio Tool Application (C#) to handle custom code. Exception handling is done using a try catch block.

```csharp
try
{
    loginID = e.InputParameters["login"];  
}
catch (Exception ex)
{
    loginID = "ruship";
}
```

Figure 5. Try Catch Block
All code is written in a try block and if exception is raised the same code is handled in catch block.

The Administrator Module written in C# handles exception in the same way. It uses .Net Validation Controls to handle various types of validations.

- The Regular Expression Validation is used for checking the email field which it accepted in that format.
- The Required Field Validation is used to not accept blank fields. It is applied on null field in the tables.

Validation summary is used to display error messages on the screen.

2.4 System Security

The iTimeCard form is published on the SharePoint server system that uses Windows authentication. An employee needs to be on a network and should be a domain user in order to access the iTimeCard System.

Once an Employee accesses iTimeCard using a web browser, internally an employee’s Windows credentials are sent to SharePoint server for authentication. An Employee profile is stored in a database in the Employee table that contains Login ID of an Employee. When an Employee accesses
iTimeCard, the login id is retrieved by code written in VSTA.

Using the login id and by calling the SQL stored procedure, employee details are retrieved for that employee from the database and displayed on the form wherever applicable.
CHAPTER THREE
TOOLS AND DEPENDENCIES

3.1 Microsoft InfoPath

Microsoft InfoPath 2007 is part of the Microsoft Office System that is used to design and fill out electronic forms. InfoPath is based on XML (Extensible Markup Language).

Organizations use various forms such as expense reports, insurance forms, time cards etc. These form templates can be designed using InfoPath with no or little knowledge about programming. InfoPath forms can be published easily on shared folders on a network, SharePoint services or a web server that allows users to easily access and fill out these forms when needed.

InfoPath forms have document like features that allow them to check spelling, format text, insert graphics etc. Some forms that are browser enabled can be accessed and filled out easily using the web browser directly without installing InfoPath on the local system.

InfoPath forms can get data from many data sources and can call web services as well. It can be integrated with
Visual Studio IDE in order to write custom code in .Net languages.

In the iT imeCard system, the timecard is designed using InfoPath 2007 form templates. Some data required to populate the form is fetched by calling the web services. Custom code is written using integrated Visual Studio Tools Application (VSTA).

3.2 Microsoft ASP.NET

Microsoft .Net Framework is an environment that is used to develop and deploy web based windows applications and web services as well. There are various technologies that are part of the .Net Framework – ASP.NET, Windows Forms, Web Services, Remoting etc.

ASP.NET is a server side scripting technology that enables scripts to be executed by an Internet server. ASP.Net applications can be developed in one of the .Net languages (usually C# or VB.NET).

Unlike traditional ASP pages, ASP.NET pages are pre-compiled and have less response time. It has world class tool support, is simple to use, is highly scalable and comes with a good security framework. CLR (Common Language Runtime)
Runtime) is the core part of the Microsoft Framework and also of ASP.NET.

ASP.NET applications are easy to manage since applications or changes to them can be easily be deployed by copying files to the server. Moreover, no server restart is required.

The Admin Tool to manage employees and projects used in iTimercard System is developed in ASP.NET 2.0. The application is developed using Visual Studio 2005 IDE. The application/web site is deployed in ISS 6.0 as a virtual directory to default web sites.

3.3 Microsoft SQL Server

Microsoft SQL Server 2005 is a data management and analysis software. Databases are created using SQL Server tools. The SQL server is highly scalable and ensures businesses with the highest level of system availability. It provides better security features and it is user friendly.

SQL Server is a Relational Database Management System (RDBMS) that uses T-SQL as a programming language. Databases contain a number of objects such as tables,
stored procedures, views, triggers and user defined functions.

Enterprise Manager is the SQL Server design time tool that helps to create database and database objects. SQL Query Analyzer is used for developing database objects using T-SQL scripts.

SQL Server database can be integrated with any application developed in any platform using SQL Data Provider or OLEDB Data Provider.

iTimeCard System uses Microsoft SQL Server 2005 as its database server. Database objects such as tables and stored procedures are developed to support the application. The tables are normalized and procedures are optimized.

ASP.NET pages access SQL Server database using ADO.NET technology.

3.4 Microsoft SharePoint

Office SharePoint Server 2007 is a server program that is part of the 2007 Microsoft Office system. SharePoint is used to facilitate collaboration and implement business processes. It also contains content management features.
that help to store and supply information essential to an organization.

SharePoint sites can be created easily and quickly. The sites can support content (including forms) publishing, content management, records management, or business intelligence needs. It has a powerful search engine that helps to search for people, documents, and data.

iTimeCard InfoPath Form is published on SharePoint Server 2007 and can be accessed by end users (Employees) using the web browser. The end users need not have InfoPath installed on their local machines to access and fill the report. Also, the powerful search engine enables the admin to search any data (time records) in an organized manner. The submitted time sheet is recorded in an ordered way which is easy for an admin to access.

The Time Card, once filled, will be submitted by the Employee. The submitted iTimeCard will be stored on SharePoint server for future reference.
CHAPTER FOUR

THE ITIMECARD SOFTWARE SYSTEM

4.1 Overview

The iTimeCard Software System is an Employee Time Card processing system in which an employee submits his/her time card for payroll and billing purposes. The system is developed using the latest technologies such as InfoPath 2007, SharePoint 2007, ASP.Net 2.0 and SQL Server 2005.
4.2 Architecture

As shown in the Figure 6, the iTimeCard System contains different components.

The user interface component contains the iTimeCard form developed in InfoPath 2007. The iTimeCard form is stored on the SharePoint Server.

The system login is managed using the SharePoint Windows authentication security feature. For an employee to
fill out and submit this form, he/she needs to login to the iSpace network using a domain user name since SharePoint uses Windows authentication for security purposes.

InfoPath will call a web service to get project information for that employee. The Web services will call the database stored procedures, which will get project information from tables. InfoPath displays project information on its Create TimeCard form to allow an employee to select a project for which he/she is filling out a timesheet.

Once the timesheet is filled out in InfoPath, it is submitted to the SharePoint Document Library and an email is also sent to the Administrator.

The Administrator is also responsible for updating the employee and the project details. For these, there is a separate Administrator module developed in ASP.NET that manages an employee and project details that are used by the iT imeCard System.

4.3 Database Design

In database, there are three tables. The Employee table contains employee information. The Project table
contains project information. Whereas, the ProjectEmployee table connects the Project and the Employee tables.

Figure 7. Database Diagram
Table 12. Employee Table Description

<table>
<thead>
<tr>
<th>Column Name</th>
<th>Data type</th>
<th>Allow Null</th>
<th>Constraints/Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>EmployeeID</td>
<td>Int</td>
<td>No</td>
<td>- Unique key into the table</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- Auto Generate by the system.</td>
</tr>
<tr>
<td>LogonID</td>
<td>Varchar(50)</td>
<td>No</td>
<td>- Will store Domain\Username</td>
</tr>
<tr>
<td>FullName</td>
<td>Varchar(50)</td>
<td>Yes</td>
<td>- Use to Store Employee Name into the table</td>
</tr>
<tr>
<td>IsRSM</td>
<td>Bit</td>
<td>Yes</td>
<td>- If employee regular or not</td>
</tr>
<tr>
<td>Email</td>
<td>varchar(50)</td>
<td>Yes</td>
<td>- Contains email address of an employee</td>
</tr>
<tr>
<td>Column Name</td>
<td>Data type</td>
<td>Allow Null</td>
<td>Constraints/Description</td>
</tr>
<tr>
<td>-------------</td>
<td>------------</td>
<td>------------</td>
<td>---------------------------------------------------------------------------------------</td>
</tr>
</tbody>
</table>
| ProjectID   | Int        | No         | - Unique key into the table  
- Auto Generate by the system.                                                          |
| ProjectName | varchar(50)| Yes        | - Contains name of project                                                             |
| ClientName  | varchar(50)| Yes        | - Contains name of client                                                              |
| ClientManager | varchar(50) | Yes    | - Contains name of manager                                                             |
| IsActive    | Bit        | Yes        | - Is project active project or not                                                    |
| IsBillable  | Bit        | Yes        | - Is project billable or not                                                           |
Table 14. ProjectEmployee Table Description

<table>
<thead>
<tr>
<th>Column Name</th>
<th>Data type</th>
<th>Allow Null</th>
<th>Constraints/Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ProjectID</td>
<td>Int</td>
<td>No</td>
<td>- Foreign key of the Project table</td>
</tr>
<tr>
<td>EmployeeID</td>
<td>Int</td>
<td>No</td>
<td>- Foreign key of the Employee table</td>
</tr>
</tbody>
</table>
CHAPTER FIVE

TESTING

5.1 Unit/System Testing

Testing was performed for each unit individually and the system in general in the iTimeCard System and the Administrator Module. Although separate test cases were not created for unit and system testing, we have to ensure that the individual units work fine with the testing system in general via thorough code review.

Exhaustive test cases are written to test units and the system with expected results. Use case analysis is the source for generating these test cases.

Add Employee screen:

This test case deals with adding a new employee to the system who needs to submit a timesheet for payroll and billing purposes.
### Table 15. Test Case of Add Employee

<table>
<thead>
<tr>
<th>Test ID</th>
<th>User Action</th>
<th>Expected Output</th>
<th>Result (P/F)</th>
<th>Actual Output in case of Failure</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>User clicks on Add button on Employee List page</td>
<td>Add new Employee page is displayed</td>
<td>P</td>
<td>-</td>
</tr>
<tr>
<td>2</td>
<td>User clicks on Add button on Employee List page</td>
<td>All projects must be listed in Available Project list box</td>
<td>P</td>
<td>-</td>
</tr>
<tr>
<td>3</td>
<td>User selects projects to assign</td>
<td>The selected project must be added to Assigned Project list box</td>
<td>P</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>to employee (&gt; button)</td>
<td>and removed from Available Project list box</td>
<td></td>
<td></td>
</tr>
<tr>
<td>---</td>
<td>------------------------</td>
<td>-------------------------------------------</td>
<td>---</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>User selects projects to unassign to employee (&lt; button)</td>
<td>The selected project must be added to Available Project list box and removed from Assigned Project list box</td>
<td>P</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>User clicks on save &amp; go back button and Employee Name is blank</td>
<td>Message must be displayed for the same</td>
<td>P</td>
<td></td>
</tr>
</tbody>
</table>

46
<table>
<thead>
<tr>
<th></th>
<th>User clicks on save &amp; go back button and Email is blank or incorrect format</th>
<th>Message must be displayed for the same P-</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>User clicks on save &amp; go back button and LoginID is blank</td>
<td>Message must be displayed for the same P-</td>
</tr>
<tr>
<td>8</td>
<td>User clicks on save &amp; go back</td>
<td>Data must be saved in database</td>
</tr>
<tr>
<td></td>
<td></td>
<td>And Message must be displayed to P-</td>
</tr>
<tr>
<td></td>
<td>button and all business rules are met</td>
<td>the user, stating that data is saved</td>
</tr>
<tr>
<td>---</td>
<td>----------------------------------------</td>
<td>----------------------------------------</td>
</tr>
<tr>
<td>9</td>
<td>User clicks on save &amp; go back button and all business rules are met and data is saved to database</td>
<td>View Employee list screen must be displayed with the new records added to the list</td>
</tr>
<tr>
<td>10</td>
<td>User click on Go Back button</td>
<td>View Employee page is displayed without save any changes made in any field</td>
</tr>
</tbody>
</table>
Add a New Project:

This test case deals with adding a new project that an employee needs to work on.

Table 16. Test Case of Add Project

<table>
<thead>
<tr>
<th>Test ID</th>
<th>User Action</th>
<th>Expected Output</th>
<th>Result (P/F)</th>
<th>Actual Output in case of Failure</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>User clicks on Add button on Project List page</td>
<td>Add a new Project page is displayed</td>
<td>P</td>
<td>-</td>
</tr>
<tr>
<td>2</td>
<td>User clicks on save &amp; go back button</td>
<td>Message must be displayed for the same</td>
<td>P</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>3</td>
<td>User clicks on save &amp; go back button and Client Name is blank</td>
<td>Message must be displayed for the same</td>
<td></td>
<td>P</td>
</tr>
<tr>
<td>4</td>
<td>User clicks on save &amp; go back button and all business rules are met</td>
<td>Data must be saved in database And Message must be displayed to the user, stating that data is saved</td>
<td></td>
<td>P</td>
</tr>
<tr>
<td></td>
<td>User clicks on save &amp; go back button and all business rules are met and data is saved to database</td>
<td>View Project list screen must be displayed with the new records added to the list</td>
<td></td>
<td></td>
</tr>
<tr>
<td>---</td>
<td>------------------------------------------------------------------------------------------------------------------------</td>
<td>---------------------------------------------------------------------------------</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td>P</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>User click on Go Back button</td>
<td>View Project page is displayed without saving any changes made on add screen</td>
<td>P</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td>-</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
View employee:

This test case deals with viewing a list of employees using iTimeCard system.

Table 17. Test Case of View Employee

<table>
<thead>
<tr>
<th>Test ID</th>
<th>User Action</th>
<th>Expected Output</th>
<th>Result (P/F)</th>
<th>Actual Output in case of Failure</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>User clicks on Employee link on Master page</td>
<td>View Employee page is displayed</td>
<td>P</td>
<td>-</td>
</tr>
<tr>
<td>2</td>
<td>User clicks on Employee link on Master page</td>
<td>All Employee data that is stored in the database must be listed in data grid</td>
<td>P</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>page</td>
<td>Description</td>
<td></td>
<td></td>
</tr>
<tr>
<td>---</td>
<td>------</td>
<td>-------------</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>User click on employee name link to edit employee information</td>
<td>Add New Employee page is displayed. And all information must be populated into that page.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>User clicks on delete link</td>
<td>Record must be deleted from the database</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>User clicks on Add a new Employee</td>
<td>Add New Employee page must be displayed and all field must be blank</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>User click on Go Back button</td>
<td>Master page is displayed</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
View project:

This test case deals with viewing a list of projects available for Employees to be assigned for iTImeCard system.

Table 18. Test Case of View Project

<table>
<thead>
<tr>
<th>Test ID</th>
<th>User Action</th>
<th>Expected Output</th>
<th>Result (P/F)</th>
<th>Actual Output in case of Failure</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>User clicks on project link on Master page</td>
<td>View Project page is displayed</td>
<td>P</td>
<td>-</td>
</tr>
<tr>
<td>2</td>
<td>User clicks on</td>
<td>All Project data that is stored in</td>
<td>P</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>action</td>
<td>event</td>
<td></td>
<td></td>
</tr>
<tr>
<td>---</td>
<td>------------------------------</td>
<td>----------------------------------------------</td>
<td>---</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>User clicks on project name</td>
<td>Add New Project page is displayed.</td>
<td>P</td>
<td></td>
</tr>
<tr>
<td></td>
<td>link</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>User clicks on delete link</td>
<td>Record must be deleted from the database</td>
<td>P</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>User clicks on Add a new</td>
<td>Add New Project page must be displayed and all</td>
<td>P</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Project</td>
<td>field must be blank</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>User click on Go Back button</td>
<td>Master page is displayed</td>
<td>P</td>
<td></td>
</tr>
</tbody>
</table>

project link on Master page

the database must be listed in data grid
Delete an Employee:

This test case deals with deleting an employee in case the employee is wrongly entered in the system or is no more with iSpace.

Table 19. Test Case of Delete Employee

<table>
<thead>
<tr>
<th>Test ID</th>
<th>User Action</th>
<th>Expected Output</th>
<th>Result (P/F)</th>
<th>Actual Output in case of Failure</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>User clicks on delete link on the View Employee page</td>
<td>Record must be deleted from the database</td>
<td>P</td>
<td>-</td>
</tr>
</tbody>
</table>
Delete a project:

This test case deals with deleting a project in case the project is wrongly entered in the system.

Table 20. Test Case of Delete Project

<table>
<thead>
<tr>
<th>Test ID</th>
<th>User Action</th>
<th>Expected Output</th>
<th>Result (P/F)</th>
<th>Actual Output in case of Failure</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>User clicks on delete link the View Project page</td>
<td>Record must be deleted from the database if project is not assigned to employee</td>
<td>P</td>
<td>-</td>
</tr>
<tr>
<td>2</td>
<td>User clicks on delete link</td>
<td>Record must be not deleted from the database if project is</td>
<td>F</td>
<td>Record is deleted from the</td>
</tr>
</tbody>
</table>
### Edit Employee information:

This test case deals with editing employee related information such as name, email or login id.

#### Table 21. Test Case of Edit Employee Data

<table>
<thead>
<tr>
<th>Test ID</th>
<th>User Action</th>
<th>Expected Output</th>
<th>Result (P/F)</th>
<th>Actual Output in case of Failure</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>User click on</td>
<td>Add New Employee page is displayed.</td>
<td>P</td>
<td>-</td>
</tr>
<tr>
<td>Employee Name Link to Edit Employee Information on View Employee Page</td>
<td>And all information must be populated into that page.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>------------------------------------------------------------------------</td>
<td>--------------------------------------------------------</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2 User clicks on save &amp; go back button and Employee Name is blank</td>
<td>Message must be displayed for the same</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3 User clicks on save &amp; go back</td>
<td>Message must be displayed for the same</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

59
<table>
<thead>
<tr>
<th>Step</th>
<th>Event Description</th>
<th>Message stating data saved in database and message displayed to user stating data saved</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>User clicks on save &amp; go back button and LoginID is blank. Message must be displayed for the same.</td>
<td>P</td>
</tr>
<tr>
<td>5</td>
<td>User clicks on save &amp; go back button and all business rules are. Data must be saved in database and message must be displayed to the user, stating that data is saved.</td>
<td>P</td>
</tr>
<tr>
<td></td>
<td>met</td>
<td></td>
</tr>
<tr>
<td>---</td>
<td>-----</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>User clicks on save &amp; go back button and all business rules are met and data is saved to database. View Employee List screen must be displayed with the updated records to the list.</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>User click on Go Back button. View Employee page is displayed without save any changes made in any field.</td>
<td></td>
</tr>
</tbody>
</table>
Edit Project information:

This project test cases deals with editing project information such as project name, client name etc.

Table 22. Test Case of Edit Project Data

<table>
<thead>
<tr>
<th>Test ID</th>
<th>User Action</th>
<th>Expected Output</th>
<th>Result (P/F)</th>
<th>Actual Output in case of Failure</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>User click on project name link to edit employee information on the View Project page</td>
<td>Add New Project page is displayed. All information of selected project must be populated into that page.</td>
<td>P</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>User clicks on save &amp; go back button and Project Name is blank</td>
<td>Message must be displayed for the same</td>
<td></td>
<td></td>
</tr>
<tr>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td>P</td>
<td></td>
</tr>
<tr>
<td></td>
<td>User clicks on save &amp; go back button and Client Name is blank</td>
<td>Message must be displayed for the same</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td>P</td>
<td></td>
</tr>
<tr>
<td></td>
<td>User clicks on save &amp; go back</td>
<td>Data must be saved in database And Message must be displayed to</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td>P</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>---</td>
<td>---</td>
<td>---</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>button and all business rules are met</strong></td>
<td>the user, stating that data is saved</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>User clicks on save &amp; go back button and all business rules are met and data is saved to database</td>
<td>View Project List screen must be displayed with the updated record to the list</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>User click on Go Back button</td>
<td>View Project page is displayed without save any changes made in any field</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Administration manages submitted timesheet:

This test case deals with managing submitted timesheets. The Admin can view timesheets submitted by employee, review them or delete them and ask for resubmission if required.

Table 23. Test Case of Manage Submitted Timesheet

<table>
<thead>
<tr>
<th>Test ID</th>
<th>User Action</th>
<th>Expected Output</th>
<th>Result (P/F)</th>
<th>Actual Output in case of Failure</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Admin type the URL to access the submitted timesheet</td>
<td>View submitted timecard page is displayed.</td>
<td>P</td>
<td>-</td>
</tr>
<tr>
<td>2</td>
<td>Admin click on</td>
<td>All submitted timesheet is</td>
<td>P</td>
<td>-</td>
</tr>
</tbody>
</table>
Create a Timecard:

Each employee is required to create a timecard using an InfoPath form by selecting a project and date.

Table 24. Test Case of Create Timecard

<table>
<thead>
<tr>
<th>Test ID</th>
<th>User Action</th>
<th>Expected Output</th>
<th>Result (P/F)</th>
<th>Actual Output in case of</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>timesheet link</td>
<td>listed</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Employee type the URL to create a timesheet</td>
<td>A page to create a timesheet is displayed.</td>
<td>Failure</td>
<td></td>
</tr>
<tr>
<td>---</td>
<td>------------------------------------------</td>
<td>------------------------------------------</td>
<td>---------</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>User click on the Timesheet link to create the Timesheet</td>
<td>Project assigned to that employee is populated into dropdown list box</td>
<td>P</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>User click on &quot;Create the Timecard&quot; and Ending date is</td>
<td>Message is Displayed for the same</td>
<td>P</td>
<td></td>
</tr>
</tbody>
</table>

67
<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>User click on &quot;Create the Timecard&quot; and all business rules are met</td>
<td>Timesheet page is displayed with employee name, client name and project name populated</td>
</tr>
<tr>
<td></td>
<td></td>
<td>P</td>
</tr>
<tr>
<td>5</td>
<td>On load of timesheet page</td>
<td>Initially all hours is zero.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>P</td>
</tr>
<tr>
<td>6</td>
<td>On load of timesheet page</td>
<td>Date is filled out according to previous screen information</td>
</tr>
<tr>
<td></td>
<td></td>
<td>P</td>
</tr>
</tbody>
</table>
Submitting the timesheet:

This test case deals with an employee submitting the timesheet once it is filled and complete. The submitted timesheet is emailed to accounts department and published on the SharePoint server.

Table 25. Test Case of Add Employee

<table>
<thead>
<tr>
<th>Test ID</th>
<th>User Action</th>
<th>Expected Output</th>
<th>Result (P/F)</th>
<th>Actual Output in case of Failure</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Users fills out</td>
<td>Total is updated accordingly</td>
<td>P</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>---</td>
<td>---</td>
<td>---</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Check the checkbox regarding acknowledgment</td>
<td>Submit button is activated</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Click on submit button</td>
<td>Email sent to account department</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Click on submit button</td>
<td>A timesheet is save into SharePoint</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
5.2 Problems

While installing the Microsoft SharePoint Server, I got two errors. One was that the product required .Net Framework 3.0. Another was that the ASP.Net 2.0 needs repairing.

Figure 8. Microsoft SharePoint Setup Error Step 1
To solve these errors, first I installed Framework 3.0 then again I got the ASP.NET error that stated ASP.NET needs repairing.

Figure 9. Microsoft SharePoint Setup Error Step 2
After repairing .Net Framework 2.0, Microsoft SharePoint Server was running successfully.
6.1 Server Setup

The server setup describes the steps needed to install SharePoint Server. It guides through each steps required for installing the same.

To install SharePoint Server:

Step 1: Insert the Microsoft SharePoint installation CD to install Microsoft SharePoint 2007. On connect to a server farm screen, it gives an option to connect to server farm or not. In our case we don’t want to create a server farm. Click next.
A server farm is a collection of two or more computers that share configuration data. Do you want to connect to an existing server farm?

- Yes, I want to connect to an existing server farm
- No, I want to create a new server farm

Figure 10. Microsoft SharePoint Setup Step 1
Step 2: In database settings screen, enter Database server name and Database name (as shown below). Fill out the Windows Authentication information as needed. Click Next.

![Specify Configuration Database Settings](image)

Figure 11. Microsoft SharePoint Setup Step 2
Step 3: On SharePoint Central Administration configuration screen, select an option to specify port number that we can use for web application hosted on web server. In our case we opted to select a random number. For security settings select NTLM or Kerberos. We selected MTML that will work with any application pool account. Click Next.

Figure 12. Microsoft SharePoint Setup Step 3
Step 4: Verify the information. To make changes, Click Back. If information seems to be proper to install SharePoint.

![Completing the SharePoint Products and Technologies Configuration Wizard](image)

Figure 13. Microsoft SharePoint Setup Step 4
Once installed, Sharepoint server screen as shown below will appear. The Central Administration screen is used by Administrator of SharePoint server for various operations and application management.

![Central Administration Screen](image)

**Figure 14. Microsoft SharePoint Setup Step 5**
6.2 Client Setup

The client setup describes the steps needed to install InfoPath 2007. It guides through each steps required for installing the same.

Steps to install Microsoft Office 2007:

1) Insert the Microsoft Office 2007 CD

2) Click Accept to agree with terms and conditions of the Microsoft Office 2007.

3) After accepting the terms and condition, click Continue

4) Click Ok on the next screen.

5) Microsoft Office will start installing.

6) After installation is done, click Ok.
7.1 Accessing the iT imeCard System

This section includes steps required by an employee to access and create his/her timesheet.

Step 1: To access a iT imeCard, login to machine on iSpace network as a domain user. After logging in, access the system by entering following url

http://sharepoint03/iSpaceForms on a Web Browser

Figure 15. Accessing the iT imeCard Home Page
Step 2: All the forms that can be used by an employee of iSpace will be listed on the page. Click on iTimeCard to create time sheet shown in Figure 15.

Step 3: Create iTimeCard page will be displayed as shown in Figure 15. iTimeCard is created for a week ending on Friday and for each project on which employee has worked during that week period. Project dropdown will list all the projects assigned to that employee. Select the appropriate project from the drop down list. Also select the week ending date by clicking the calendar icon. Click on Create Time Card button to create time card in Figure 16.
Figure 16. Select a Project to Access the iTimeCard
7.2 Submitting Timesheet

This section includes steps required by an employee to submit his/her timesheet by entering necessary information about the hours he/she has worked on a particular project.

Step 1: Timesheet is displayed as shown in Figure 17 for a particular project and week ending on Friday. Verify your name, client name and project name.

Hours worked is the actual hours that employee have worked on that project during that week.

Holiday hours need to be entered if there are any holidays during that week.

PTO (Paid Time-off) needs to be entered if employee has taken PTO during that week.

Total will be calculated based on the entered values in these boxes.
Employee: Arpita Parikh  
Client: Demo  
Project: Demo Project

<table>
<thead>
<tr>
<th></th>
<th>Sat 9/8/07</th>
<th>Sun 9/9/07</th>
<th>Mon 9/10/07</th>
<th>Tue 9/11/07</th>
<th>Wed 9/12/07</th>
<th>Thu 9/13/07</th>
<th>Fri 9/14/07</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hours worked:</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Holiday:</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>PTO (Paid Time-off)</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Total:</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

☐ I hereby acknowledge that above information is correct

Figure 17. Submitting the iTimeCard Screen Step 1
Step 2: Submit button will be enabled once you acknowledge that information submitted is correct by clicking the check box as shown in Figure 18. Click on submit button to submit timesheet. Conformation Message will be displayed upon successful submission of time sheet.

Figure 18. Submitting the iTimeCard Screen Step 2
8.1 Managing Submitted Timesheet

This section includes steps for an Admin to manage submitted timesheet. It helps Admin to view submitted timesheet, delete timesheet etc.
Step 1: Type http://sharepoint03/sites/SI on a web browser to access submitted timesheet. The following Figure 19 screen is displayed that list all the documents that is accessible on left frame of the page.

Figure 19. Managing Submitted iTimeCard Screen
Step 2: Click on the iTimeCard link under Documents section in left frame of the page. The following page as shown in Figure 20 is displayed with list of timesheets submitted by different employees. The information such as client name, period ending, employee name, project name is displayed on this page. You can order by this list with any of these listed columns.

Figure 20. Submitted iTimeCard List
Step 3: In order to view the latest submitted time sheet submitted by an employee, click on link under column name as indicated above in Figure 20. The timesheet as submitted by an employee will be displayed.

8.2 Managing Employee Data

This section helps Admin to manage employee profiles. It describes steps the admin needs to take in order to add, edit or delete an employee.

Step 1: Click on Employee link as shown in Figure 21 to manage Employee information.

Figure 21. Managing Employee Detail Home Page
Step 2: List of employees will be displayed as shown in Figure 22 with information such as employee name, login id, email and whether he is RSM or not.

To add an employee, click on Add Employee Button to go to add employee page as shown in Figure 23. Go to Step 3.

To edit employee information, click on employee full name link to go to edit page as shown in Figure 25. Go to Step 4

To delete an employee, click on delete link against employee name to be deleted.
Figure 22. View Employee Screen

Step 3: Fill out the employee’s information as shown below.

Assign projects to employee from list of available project by clicking on > button after selecting that project in available project list box. To unassign a project click on < button by selecting project in assinged project list box.

Click on Save & Go Back button after entering informaton to add an employee and go back to list page.
The following business rules apply to while adding an employee:

- Employee Name must not be blank
- Email must not be blank and should be incorrect format
- Login ID must not be blank

Message will be displayed in red if these business rules are not met.
Confirmation message will be displayed as shown in Figure 24 if business rules are met.
Figure 23. Add Employee Screen
Figure 24. Record Save Message

Step 4: The Figure 25 screen is displayed with information about an employee populate in the controls when user clicks an employee in link page.

Make necessary changes to employee information. Change Assigned projects to employee from list of available project by clicking on > button after selecting that project in available project list box. To unassign a project click on < button by selecting project in assigned project list box.

Click on Save & Go Back button after changing information to edit an employee and go back to list page.

The following business rules apply to while editing an employee:

- Employee Name must not be blank
• Email must not be blank and should be incorrect format

• Login ID must not be blank

Message will be displayed in red if these business rules are not met.

Confirmation message will be displayed as shown in Figure 24 if business rules are met.
Figure 25. Edit Employee Screen
8.3 Managing Project Data

This section helps Admin to manage project details. It describes steps the admin needs to take in order to add, edit or delete a project.

Step 1: Click on the Project link as shown in Figure 26.

Figure 26. Managing Project Detail Home Page

Step 2: List of projects will be displayed as shown in Figure 27 with information such as project name, client name, Client Manager1, whether project is active or not and whether that project is billable project or fixed price.
To add a project, click on Add Project Button to go to add project page as shown in Figure 28. Go to Step 3.

To edit a project, click on project name link to go to edit page as shown in Figure 29. Go to Step 4.

To delete a project, click on delete link against project name to be deleted.

![Table of Projects]

<table>
<thead>
<tr>
<th>Delete</th>
<th>Project Name</th>
<th>ClientName</th>
<th>ClientManager</th>
<th>IsActive</th>
<th>IsBillable</th>
</tr>
</thead>
<tbody>
<tr>
<td>Delete</td>
<td>EBilling</td>
<td>PMSI</td>
<td>Aram</td>
<td>☑️</td>
<td>☑️</td>
</tr>
<tr>
<td>Delete</td>
<td>TEMP</td>
<td>TMP</td>
<td>PMK</td>
<td>☐</td>
<td>☐</td>
</tr>
<tr>
<td>Delete</td>
<td>TIMECARD</td>
<td>ARPITA</td>
<td>VINOD</td>
<td>☑️</td>
<td>☑️</td>
</tr>
<tr>
<td>Delete</td>
<td>SHAREPOINT DEVELOPMENT</td>
<td>AMERICAN HONDA</td>
<td>LOKH SELLAPAN</td>
<td>☑️</td>
<td>☑️</td>
</tr>
</tbody>
</table>

![Add Project and Go Back Buttons]

Figure 27. View Project Screen
Step 3: Fill out the project information as shown below in Figure 29. Click on Save & Go Back button after entering information to add a project and go back to list page.

The following business rules apply to while adding an employee:

- Project Name must not be blank
- Client Name must not be blank
- Client Manager must not be blank

Message will be displayed in red if these business rules are not met.

Confirmation message will be displayed as shown in Figure 24 if business rules are met and record is saved.
Figure 28. Add a New Project Screen

Step 4: The Figure 29 screen is displayed with information about a project populated in the controls when user clicks a project in link page.

Make necessary changes to project information.

Click on Save & Go Back button after changing information to edit a project and go back to list page.
The following business rules apply to while editing an employee:

- Project Name must not be blank
- Client Name must not be blank
- Client Manager must not be blank

Message will be displayed in red if these business rules are not met.

Confirmation message will be displayed as shown in Figure 24 if business rules are met and record is saved.
<table>
<thead>
<tr>
<th><strong>Project Name</strong></th>
<th>DUMMY</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Client Name</strong></td>
<td>WILL SMITH</td>
</tr>
<tr>
<td><strong>Client Manager</strong></td>
<td>JEFF SMITH</td>
</tr>
<tr>
<td><strong>Is Active</strong></td>
<td>✔</td>
</tr>
<tr>
<td><strong>Is Billable</strong></td>
<td>✔</td>
</tr>
</tbody>
</table>

**Figure 29. Edit Project Screen**
CHAPTER NINE

CONCLUSION

9.1 Summary

The iTimeCard timesheet management system has an end user interface that allows employees to submit their timesheets online. The biggest advantage of such a system is that not only it is extremely user friendly but also any employee, at their convenience, using his/her web browser without installing any drivers, could access it. The administrator user interface allows the employer or administrator to create and edit a project assigned to the employee; access the information and hours worked, and use the supplied information for issuing a paycheck.

The timesheet forms based on Microsoft InfoPath 2007 ensure easy access to the server. The pre-complied ASP.NET scripts take the least response time and SQL (Sequential Query Language) server ensures high level of system availability along with good security features. The stored timesheet on Microsoft SharePoint Server 2007 allows the company to track the progress of the company via analyzing the amount of time certain groups of employees spent on a specific project. It can also manage an employee’s progress
with the same feature. The powerful search feature of SharePoint also allows tracking any past time-record information quickly.

9.2 Future Directions

Currently the iTimeCard System is designed only to eliminate paper timecards and form filling issues. The next phase will be enhancing this system by adding some functionality to make things better for iSpace.

The following will be the enhancements to the iTimeCard system in the near future:

1. The data on the iTimeCard will be stored in a database that can be used directly to generate reports or can be used by the accounting department directly for payroll purposes.

2. An electronic approval process will be added to the system that will allow managers to approve timesheets submitted by the employee just by the click of a button or using digital signatures.

3. A Login page will be added to the administrator module.
APPENDIX

APPLICATION CODE
Add a new Employee

public partial class AddEmployee : System.Web.UI.Page
{
    public static String str;
    //public static bool flag = false;
    protected void Page_Load(object sender, EventArgs e)
    {
        //populating the list box
        if (!IsPostBack)
        {
            SqlConnection conn = new SqlConnection();

            conn.ConnectionString = @"Data Source =
sharepoint03; Initial Catalog= TimeCard; Integrated
Security=True;Pooling =False";

            conn.Open();

            SqlCommand cmd = new SqlCommand();
            SqlCommand cmd1 = new SqlCommand();
            SqlCommand cmdProjEmp = new SqlCommand();

            cmd.Connection = conn;
            cmd1.Connection = conn;
            cmdProjEmp.Connection = conn;

            cmd.CommandType = CommandType.Text;
            cmd1.CommandType = CommandType.Text;
            cmdProjEmp.CommandType = CommandType.Text;

            cmd.CommandText = "select * from Project";
            cmd1.CommandText = "select * from Employee";
            //cmdProjEmp.CommandText = "select * from
ProjectEmployee";
            SqlDataReader reader;

            //if edit employee info
            if (Request.QueryString["id"] != null)
            {

        107
str = Request.QueryString["id"].Trim();
// populating the data for selected update
reader = cmd1.ExecuteReader();
while (reader.Read())
{
    // Response.Write(str);
    if (str.Equals(reader["EmployeeID"].ToString()))
    {
        txtEmail.Text = reader["Email"].ToString();
        txtLogonID.Text =
        reader["LogonID"].ToString();
        txtName.Text =
        reader["FullName"].ToString();
        ckIsRSM.Checked = bool.Parse(reader["IsRSM"].ToString());
        //flag = true;
    }
}
reader.Close();

//populating the data for Assigned Project
ListBox

cmdProjEmp.CommandText = "select
ProjectEmployee.ProjectID, Project.ProjectName from
ProjectEmployee " +
"INNER JOIN Project ON
ProjectEmployee.ProjectID = Project.ProjectID where
ProjectEmployee.EmployeeID = " + str;
reader = cmdProjEmp.ExecuteReader();
while (reader.Read())
{
    lbAssignProj.Items.Add(reader["ProjectName"].ToString());
    reader["ProjectID"].ToString();
}

reader.Close();

//populating the data for Available Project
ListBox
cmdProjEmp.CommandText = "select Project.ProjectID, Project.ProjectName from Project where projectid not in(select projectid from ProjectEmployee where Employeeid = "+ str + ")";
reader = cmdProjEmp.ExecuteReader();
while (reader.Read())
{
    lbAvailProject.Items.Add(reader["ProjectName"].ToString());
}
reader.Close();

// if newly added employee then populating the data for Available Project List box
else
{
    reader = cmd.ExecuteReader();
    while (reader.Read())
    {
        lbAvailProject.Items.Add(reader["ProjectName"].ToString());
    }
    reader.Close();
}
conn.Close();
ListboxSize();

protected void btClear_Click(object sender, EventArgs e)
{
    txtEmail.Text = "";
    txtLogonID.Text = "";
    txtName.Text = "";
    ckIsRSM.Checked = false;
}
protected void btSave_Click(object sender, EventArgs e)
SqlConnection conn = new SqlConnection();

conn.ConnectionString = @"Data Source = sharepoint03; Initial Catalog= TimeCard; Integrated Security=True;Pooling =False";

conn.Open();

//update or edit an existing employee
if (Request.QueryString["id"] != null)
{
    SqlCommand cmd1 = new SqlCommand();
    cmd1.Connection = conn;
    cmd1.CommandType = CommandType.Text;
    cmd1.CommandText = "UPDATE EMPLOYEE SET LogonID=@LogonID," +
                        " Email=@Email," +
                        " FullName=@FullName," +
                        " IsRSM=@IsRSM" +
                        " WHERE (EmployeeID = @EmployeeID)";
    cmd1.Parameters.Add("@LogonID", SqlDbType.VarChar);
    cmd1.Parameters.Add("@Email", SqlDbType.VarChar);
    cmd1.Parameters.Add("@FullName", SqlDbType.VarChar);
    cmd1.Parameters.Add("@IsRSM", SqlDbType.Bit);
    cmd1.Parameters.Add("@EmployeeID", SqlDbType.Int);

    cmd1.Parameters["@LogonID"].Value = txtLogonID.Text.Trim().ToUpper();
    cmd1.Parameters["@Email"].Value = txtEmail.Text.Trim().ToUpper();
    cmd1.Parameters["@FullName"].Value = txtName.Text.Trim().ToUpper();
    cmd1.Parameters["@IsRSM"].Value = bool.Parse(ckIsRSM.Checked.ToString());
    cmd1.Parameters["@EmployeeID"].Value = int.Parse(str.Trim());

    cmd1.ExecuteNonQuery();
}
/ Update the ProjectEmployee table
SqlCommand cmdprojemp = new SqlCommand();
cmdprojemp.Connection = conn;
cmdprojemp.CommandType = CommandType.Text;

// Deleting all records from ProjectEmployee for the selected employee
SqlCommand cmdprojemp = new SqlCommand();
cmdprojemp.Connection = conn;
cmdprojemp.CommandType = CommandType.Text;

// Inserting new records into ProjectEmployee table
SqlCommand cmdprojempl = new SqlCommand();
cmdprojempl.Connection = conn;
cmdprojempl.CommandType = CommandType.Text;

if (lbAssignProj.Items.Count != 0)
{
    // Adding records for ProjectEmployee table
    foreach (ListItem listItem in lbAssignProj.Items)
    {
        SqlCommand cmdprojempl = new SqlCommand();
cmdprojempl.Connection = conn;
cmdprojempl.CommandType = CommandType.Text;

        cmdprojempl.CommandText = "INSERT INTO
PROJECTEMPLOYEE(ProjectID,EmployeeID)"
    "values(@ProjectID,@EmployeeID)";

        cmdprojempl.Parameters.Add("@ProjectID", SqlDbType.Int);

        cmdprojempl.Parameters.Add("@EmployeeID", SqlDbType.Int);

        cmdprojempl.Parameters["@ProjectID"].Value =
listItem.Value;

        cmdprojempl.Parameters["@EmployeeID"].Value =
int.Parse(str.Trim());
}
cmdprojempl.ExecuteNonQuery();

// 'Add a new employee
else
{
SqlCommand cmd = new
SqlCommand("AddEmployee", conn);

cmd.CommandType = CommandType.StoredProcedure;

cmd.Parameters.Add("@LogonID", SqlDbType.VarChar);

// SqlParameter sqlParam =
// cmd.Parameters.Add("@EmpID", SqlDbType.Int);
// int EmpID =
// (int)cmd.Parameters["@EmpID"].Value;

int EmpID =
(int)cmd.Parameters["@OEmpID"].Value;

// SqlParameter sqlParam =
// cmd.Parameters.Add("@EmpID", SqlDbType.Int);
// int EmpID =
// (int)cmd.Parameters["@EmpID"].Value;
// SqlParameter sqlParam =
// cmd.Parameters.Add("@EmpID", SqlDbType.Int);
Console.Write(EmpID);

// if project assigned to Employee
if (lbAssignProj.Items.Count != 0)
{
    foreach (List< ListItem listItem in lbAssignProj.Items)
    {
        SqlCommand cmdEmpProj = new SqlCommand();
        cmdEmpProj.Connection = conn;
        cmdEmpProj.CommandType = CommandType.Text;
        cmdEmpProj.CommandText = "INSERT INTO PROJECTEMPLOYEE(ProjectID,EmployeeID)" + 
            "values(@ProjectID,@EmployeeID)";
        cmdEmpProj.Parameters.Add("@ProjectID", SqlDbType.Int);
        cmdEmpProj.Parameters.Add("@EmployeeID", SqlDbType.Int);
        cmdEmpProj.Parameters["@ProjectID"].Value = listItem.Value;
        cmdEmpProj.Parameters["@EmployeeID"].Value = EmpID;
        cmdEmpProj.ExecuteNonQuery();
    }
}
//Response.Write("<script language='javascript'>alert('Record saved!');</script>");

connect.Close();
Response.Write("<script language='javascript'>alert('Record Saved!'); window.location.href = 'EditEmployee.aspx';</script>");
protected void btAddProject_Click(object sender, EventArgs e)
{
    try
    {
        if (lbAvailProject.SelectedItems != null)
        {
            lbAssignProj.Items.Add(lbAvailProject.SelectedItems.ToString());
            lbAvailProject.Items.Remove(lbAvailProject.SelectedItems);
        }
    }
    catch (NullReferenceException ex)
    {
        throw (ex);
    }
}
protected void btDelProj_Click(object sender, EventArgs e)
{
    try
    {
        if (lbAssignProj.SelectedItems != null)
        {
            lbAvailProject.Items.Add(lbAssignProj.SelectedItems.ToString());
            lbAssignProj.Items.Remove(lbAssignProj.SelectedItems);
            //lbAssignProj.SelectedItems.Enabled = false;
        }
    }
    catch (NullReferenceException ex)
    {
        throw (ex);
    }
}
protected void btBack_Click(object sender, EventArgs e)
{
    Response.Redirect("EditEmployee.aspx");
}
protected void ListboxSize()
{
    lbAssignProj.Height = 150;
    lbAssignProj.Width = 200;
    lbAvailProject.Height = 150;
    lbAvailProject.Width = 200;
}

View Employee

public partial class _Default : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
    }
    protected void btAdd_Click(object sender, EventArgs e)
    {
        Response.Redirect("AddEmployee.aspx");
    }
    protected void btBack_Click(object sender, EventArgs e)
    {
        Response.Redirect("HomePage.aspx");
    }
}

Add a new Project
public partial class AddEmployee : System.Web.UI.Page
{
    public static String str;
    protected void Page_Load(object sender, EventArgs e)
    {
        if (!IsPostBack)
        {
            SqlConnection conn = new SqlConnection();

            conn.ConnectionString = @"Data Source = sharepoint03; Initial Catalog= TimeCard; Integrated Security=True;Pooling =False";

            conn.Open();

            SqlCommand cmd = new SqlCommand();
            cmd.Connection = conn;
            cmd.CommandType = CommandType.Text;
            cmd.CommandText = " select * from Project";

            SqlDataReader reader = cmd.ExecuteReader();
            if (Request.QueryString["id"] != null)
            {
                str = Request.QueryString["id"].Trim();
                while (reader.Read())
                {
                    // Response.Write(str);

                    if (str.Equals(reader["ProjectID"].ToString()))
                    {
                        txtClient.Text = reader["ClientName"].ToString();
                        txtManager.Text = reader["ClientManager"].ToString();
                        txtProject.Text = reader["ProjectName"].ToString();
                        ckIsActive.Checked = bool.Parse(reader["IsActive"].ToString());
                    }
                }
            }
        }
    }
}
CkIsBillable.Checked = 
bool.Parse(reader["IsBillable"]).ToString();
}
}
reader.Close();
conn.Close();
}
}

protected void btClear_Click(object sender, EventArgs e)
{
    txtClient.Text = "";
    txtManager.Text = "";
    txtProject.Text = "";
    ckIsActive.Checked = false;
    CkIsBillable.Checked = false;
}

protected void btSave_Click(object sender, EventArgs e)
{
    SqlConnection conn = new SqlConnection();
    conn.ConnectionString = @"Data Source = sharepoint03; Initial Catalog = TimeCard; Integrated Security=True;Pooling =False";
    conn.Open();

    if (Request.QueryString["id"] != null)
    {
        SqlCommand cmd1 = new SqlCommand();
        cmd1.Connection = conn;
        cmd1.CommandType = CommandType.Text;
        cmd1.CommandText = "UPDATE PROJECT SET ProjectName=@ProjectName," +
            " ClientName=@ClientName," +
            " ClientManager=@ClientManager," +
            " IsActive=@IsActive," +
            " IsBillable=@IsBillable " +
            " WHERE (ProjectID =
"
cmd1.Parameters.AddWithValue("©ProjectName", SqlDbType.VarChar);
        cmd1.Parameters.AddWithValue("©ClientName", SqlDbType.VarChar);
        cmd1.Parameters.AddWithValue("©ClientManager", SqlDbType.VarChar);
        cmd1.Parameters.AddWithValue("©IsActive", SqlDbType.Bit);
        cmd1.Parameters.AddWithValue("©IsBillable", SqlDbType.Bit);
        cmd1.Parameters.AddWithValue("@ProjectID", SqlDbType.Int);

        cmd1.Parameters["@ProjectName"].Value = txtProject.Text.Trim().ToUpper();
        cmd1.Parameters["@ClientName"].Value = txtClient.Text.Trim().ToUpper();
        cmd1.Parameters["@ClientManager"].Value = txtManager.Text.Trim().ToUpper();
        cmd1.Parameters["@IsActive"].Value = bool.Parse(ckIsActive.Checked.ToString());
        cmd1.Parameters["@ProjectID"].Value = int.Parse(str.Trim());

        cmd1.ExecuteNonQuery();
        Response.Write("Record Updated Successfully");
    }
    else
    {
        SqlCommand cmd = new SqlCommand();
        cmd.Connection = conn;
        cmd.CommandType = CommandType.Text;
        cmd.CommandText = "INSERT INTO PROJECT(ProjectName,ClientName,ClientManager,IsActive,IsBillable)" +
        "values(@ProjectName,@ClientName,@ClientManager,@IsActive,@IsBillable)";
cmd.Parameters.Add("@ProjectName", SqlDbType.VarChar);
cmd.Parameters.Add("@ClientName", SqlDbType.VarChar);
cmd.Parameters.Add("@ClientManager", SqlDbType.VarChar);
cmd.Parameters.Add("@IsActive", SqlDbType.Bit);
cmd.Parameters.Add("@IsBillable", SqlDbType.Bit);

    cmd.Parameters["@ProjectName"].Value = txtProject.Text.Trim().ToUpper();
    cmd.Parameters["@ClientName"].Value = txtClient.Text.Trim().ToUpper();
    cmd.Parameters["@ClientManager"].Value = txtManager.Text.Trim().ToUpper();
    cmd.Parameters["@IsActive"].Value = bool.Parse(ckIsActive.Checked.ToString());

    cmd.ExecuteNonQuery();
    Response.Write("Record Save Successfully");
}  
conn.Close();
Response.Redirect("EditProject.aspx");

protected void btBack_Click(object sender, EventArgs e) {
    Response.Redirect("EditProject.aspx");
}

public partial class ViewProject : System.Web.UI.Page
{
protected void BtAddProj_Click(object sender, EventArgs e)
{
    Response.Redirect("AddProject.aspx");
}
protected void btBack_Click(object sender, EventArgs e)
{
    Response.Redirect("HomePage.aspx");
}

ALTER PROCEDURE dbo.AddEmployee
(
    @LogonID varchar(50),
    @FullName varchar(50),
    @Email varchar(50),
    @IsRSM bit,
    @oEmpID int OUTPUT
)
/
/*
(  @parameter1 int = 5,
  @parameter2 datatype OUTPUT
)
*/
AS
BEGIN

SET NOCOUNT ON

INSERT INTO EMPLOYEE(LogonID,FullName,Email,IsRSM)
values(@LogonID,@FullName,@Email,@IsRSM)

Select @oEmpID = @@identity

END

120
ALTER PROCEDURE [dbo].[GetActiveProjectsForEmployee]
-- Add the parameters for the stored procedure here
@logonID VARCHAR(50)
AS
BEGIN
-- SET NOCOUNT ON added to prevent extra result sets from
-- interfering with SELECT statements.
SET NOCOUNT ON;

SELECT Project.ProjectID,
    Project.ProjectName,
    Project.ClientName,
    Project.ClientManager,
    Project.IsBillable,
    Employee.FullName as EmployeeName
FROM Project
INNER JOIN ProjectEmployee
    ON ProjectEmployee.ProjectID = Project.ProjectID
INNER JOIN Employee
    ON Employee.EmployeeID = ProjectEmployee.EmployeeID
WHERE Employee.LogonID = @logonID
END

ALTER PROCEDURE dbo.GetEmployeeIDFromInsertedEmployee
(   @iLogonID varchar(50),
    @iFullName varchar(50),
    @iEmail varchar(50),
    @iIsRSM bit,
    @oEmpID int OUTPUT
)  /*
(   @parameter1 int = 5,
    @parameter2 datatype OUTPUT
) */
AS
BEGIN
SET NOCOUNT ON

INSERT INTO EMPLOYEE(LogonID, FullName, Email, IsRSM)
values(iLogonID, iFullName, iEmail, iIsRSM)
Select @@identity
END
RETURN @@identity

SET NOCOUNT OFF
REFERENCES


