A WEB-BASED CALENDAR AND CASE MANAGEMENT TOOL

Kalyana Valli Veeramachaneni
B.E, Jawaharlal Nehru Technological University, 2005

PROJECT

Submitted in partial satisfaction of the requirements for the degree of

MASTER OF SCIENCE

in

COMPUTER SCIENCE

at

CALIFORNIA STATE UNIVERSITY, SACRAMENTO

SPRING
2010
A WEB-BASED CALENDAR AND CASE MANAGEMENT TOOL

A Project

by

Kalyana Valli Veeramachaneni

Approved by:

__________________________________, Committee Chair
Jinsong Ouyang, Ph.D.

__________________________________, Second Reader
William Mitchell, Ph.D.

Date
Student: Kalyana Valli Veeramachaneni

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

__________________________, Graduate Coordinator       ________________
Cui Zhang, Ph.D                     Date

Department of Computer Science
Abstract

of

A WEB-BASED CALENDAR AND CASE MANAGEMENT TOOL

by

Kalyana Valli Veeramachaneni

Statement of Problem: In today’s environment, people operate in a hectic environment than ever before. For Lawyers and certain practices, the ability to do online scheduling of appointments may work well. Hence, a web-based scheduling calendar is developed for all the employees in a law firm. In addition, a complete case management module for the attorneys to record and retrieve their case information is developed.

Conclusions Reached: This project contributes to improve attorney & staff productivity by becoming more efficient in client and case management and handles a greater caseload without increasing staff. It provides the capability to better communicate with clients, provide immediate information on case status to clients, and to have a quicker turn around on work product.

_______________________, Committee Chair
Jinsong Ouyang, Ph.D

_______________________
Date

iv
ACKNOWLEDGMENTS

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

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

 Also, thanks to Dr. William Mitchell 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

Page

Acknowledgments........................................................................................................... v
List of Figures ................................................................................................................ viii
Hardware and Software Specifications.......................................................................... x

Chapter
1. INTRODUCTION .................................................................................................. 1
   Background ........................................................................................................... 1
   Problem Statement .............................................................................................. 1
   Project Objectives .............................................................................................. 2

2. REQUIREMENTS ANALYSIS ............................................................................. 4
   Product Perspective ............................................................................................ 4
   Product Features ............................................................................................... 4
   Operating Environment ...................................................................................... 6
   Assumptions and Dependencies ...................................................................... 6
   Use Cases ......................................................................................................... 7
   External Interface Requirements ..................................................................... 45
   Non-Functional Requirements ........................................................................ 46
   Software Quality Attributes ............................................................................ 47
   Developer Environment .................................................................................... 47

3. SYSTEM DESIGN ............................................................................................... 50
   Building a Web Application ........................................................................... 50
   .NET Application Architecture ....................................................................... 51
   High Level Design .......................................................................................... 52

4. DATABASE DESIGN.......................................................................................... 57
   About SQL Server 2005.................................................................................. 57
   Database Tables and Database Diagram ......................................................... 57
   Sample of Database Scripts ............................................................................ 58
5. IMPLEMENTATION DETAILS ................................................................. 60
   Implementation Details of Scheduling Calendar Feature ......................... 60
   Development of Web Interface using ASP.NET ........................................ 60
   Storing and Retrieving Data using ADO.NET .......................................... 63
   Implementation of Business Logic using C#.NET ..................................... 67
   Implementation Details of Case Management Feature ............................ 70
   Implementation Details of Task Management Feature ............................ 74
   Implementation Details of Contacts Management Feature ...................... 75
6. CONCLUSION .......................................................................................... 77
   Appendix Sample Screens ................................................................. 79
   References ......................................................................................... 99
<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>Figure 1 User Authentication</td>
<td>9</td>
</tr>
<tr>
<td>2.</td>
<td>Figure 2 Date Calculator</td>
<td>11</td>
</tr>
<tr>
<td>3.</td>
<td>Figure 3 Motion Date Calculator</td>
<td>12</td>
</tr>
<tr>
<td>4.</td>
<td>Figure 4 Judgment Interest Calculator</td>
<td>14</td>
</tr>
<tr>
<td>5.</td>
<td>Figure 5 Scheduling Calendar</td>
<td>15</td>
</tr>
<tr>
<td>6.</td>
<td>Figure 6 Add New Event</td>
<td>17</td>
</tr>
<tr>
<td>7.</td>
<td>Figure 7 Edit/Delete an Event</td>
<td>18</td>
</tr>
<tr>
<td>8.</td>
<td>Figure 8 Re-Schedule Event</td>
<td>20</td>
</tr>
<tr>
<td>9.</td>
<td>Figure 9 Add New Rule</td>
<td>22</td>
</tr>
<tr>
<td>10.</td>
<td>Figure 10 Email Calendar</td>
<td>23</td>
</tr>
<tr>
<td>11.</td>
<td>Figure 11 Print Calendar</td>
<td>25</td>
</tr>
<tr>
<td>12.</td>
<td>Figure 12 Add New Contact</td>
<td>26</td>
</tr>
<tr>
<td>13.</td>
<td>Figure 13 Add/Edit./Delete What Entries</td>
<td>28</td>
</tr>
<tr>
<td>14.</td>
<td>Figure 14 Add/Edit/Delete Where Entries</td>
<td>30</td>
</tr>
<tr>
<td>15.</td>
<td>Figure 15 Add New Docket Item</td>
<td>32</td>
</tr>
<tr>
<td>16.</td>
<td>Figure 16 Add/Edit/Delete Tasks to Assign</td>
<td>34</td>
</tr>
<tr>
<td>17.</td>
<td>Figure 17 View/Edit Tasks Assigned to User</td>
<td>35</td>
</tr>
<tr>
<td>18.</td>
<td>Figure 18 Add New Case</td>
<td>37</td>
</tr>
<tr>
<td>19.</td>
<td>Figure 19 Edit/Delete a Case</td>
<td>38</td>
</tr>
<tr>
<td>20.</td>
<td>Figure 20 View Case Details</td>
<td>40</td>
</tr>
<tr>
<td>21.</td>
<td>Figure 21 Add Case Notes</td>
<td>42</td>
</tr>
<tr>
<td>22.</td>
<td>Figure 22 View Clipboard Statutes Data</td>
<td>43</td>
</tr>
<tr>
<td>23.</td>
<td>Figure 23 View Clipboard Discovery Log</td>
<td>45</td>
</tr>
<tr>
<td>24.</td>
<td>Figure 24 .NET Framework Components</td>
<td>47</td>
</tr>
<tr>
<td>25.</td>
<td>Figure 25 Client – Server Architecture</td>
<td>50</td>
</tr>
<tr>
<td>26.</td>
<td>Figure 26 Block Diagram</td>
<td>52</td>
</tr>
</tbody>
</table>
27. Figure 27 Class Diagram: Overview ................................................................. 54
28. Figure 28 Class Diagram for Scheduling Calendar Module ........................... 55
29. Figure 29 Class Diagram for Case Management Module ............................... 56
30. Figure 30 Entity-Relationship Diagram .......................................................... 58
31. Figure 31 ADO. NET Data Architecture .......................................................... 63
32. Figure 32 Date Calculator Screen Shot ............................................................ 79
33. Figure 33 Judgment Interest Calculator Screen Shot ...................................... 80
34. Figure 34 Motion Date Calculator Screen Shot .............................................. 81
35. Figure 35 Scheduling Calendar Screen Shot .................................................. 82
36. Figure 36 Add New Event Screen Shot ......................................................... 83
37. Figure 37 Edit/Delete Event Screen Shot ....................................................... 84
38. Figure 38 Re-Schedule Event Screen Shot .................................................... 85
39. Figure 39 Add New Rule Screen Shot ............................................................. 86
40. Figure 40 Delete Rule Screen Shot ............................................................... 87
41. Figure 41 Email Calendar Screen Shot .......................................................... 88
42. Figure 42 Print Calendar Screen Shot ............................................................ 89
43. Figure 43 Add/Edit/Delete What Entries Screen Shot ..................................... 90
44. Figure 44 Add/Edit/Delete Where Entries Screen Shot .................................. 91
45. Figure 45 Add Docket Item Screen Shot ....................................................... 92
46. Figure 46 Add/Edit/Delete Tasks to Assign Screen Shot ............................... 93
47. Figure 47 View/Edit Tasks Assigned to User Screen Shot .............................. 94
48. Figure 48 Add New Case Screen Shot ............................................................ 95
49. Figure 49 Edit/Delete Case Screen Shot ....................................................... 96
50. Figure 50 View Case Details Screen Shot ..................................................... 97
51. Figure 51 Add Case Notes Screen Shot .......................................................... 98
HARDWARE AND SOFTWARE SPECIFICATIONS

The Hardware and software requirements for the project are mentioned below.

**Hardware Requirements**

- Pentium-III Processor with 500MHZ or above
- 100MB of Free Hard disk space
- 256MB of RAM
- LAN Network (For Remote Sources)
- Network interface card or Modem (For Remote Access)

**Software Requirements**

- WINDOWS NT 4.0 | 2000 | 9.X | VISTA | XP
- Microsoft .Net Framework 3.5 (Minimal Requirement for deploying application on Client Machine)
- Internet Information Services (IIS)
- Microsoft SQL Server 2005
- Programming Languages Used – C#.NET, ASP.NET, Java script
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.

Background

Peter Stone Law Firm is a small to medium sized law firm dedicated to providing law, legal and claims services to personal injuries. The firm has an enviable record in commercial collections, transportation collections, premium collections, leases and credit cards. The law firm undertakes work for both claimant and defendant parties and offers services for each of these different types of client.

Problem Statement

One of the fundamental tasks for any law firm is to provide efficient Scheduling Calendars for staff who are working on the different aspects of every case assigned to the firm. This is important for two main reasons:

First, the events recorded by the firm’s employees on a particular case, whether they are solicitors, secretarial or other auxiliary staff, can be charged at an hourly rate to clients.

Second, the calendars should be efficient enough to maintain docket items and case events both at a single place, making it easy for tracking the accountability of a case.
Some of the potential problems identified with the existing software used by the firm that formed the basis for coming up with a new solution are as follows:

**No single software:** There is a need for a single web-based solution to perform all the daily tasks by all the employees at the organization. The major concerns include not having single software for all the required activities, one software has one feature that serves better for all the employees, while another software has other things better.

**Database not maintained at the Firm:** Any third party soft wares used in the firm are charging heavily for maintaining database servers on their own. In addition, the existing soft wares are missing important features like importing and exporting data to other soft wares used by the firm.

**Costs Increased due to Third Party tools:** Maintaining and hosting server by a third party has increased the costs incurred and has significant impact on providing services to the clients.

**Project Objectives**

This project is being developed for the “Law stone Firm” based in Sacramento, CA with the primary goals as follows:

**Improve overall productivity:** The firm wishes to improve attorney & staff productivity by becoming more efficient in client and case management and handle a greater caseload without increasing staff.

**Improve client communications and work product delivery:** The firm requires the capability to better communicate with clients, provide immediate information on case
status to clients, and to have a quicker turn around on work product. Hence, this project aims at providing an efficient web-based solution that ensures better communication and quick access to case related information.

**Ease of use:** The system should be easy to use and flow with the information and procedures of the firm.

**Flexible System:** The firm should be able to either customize the system in house, or request the developer to customize the system, based on the firm’s requirements.

**Training:** Training is essential to any new implementation. Computerized case management is no exception and the firm should able to adapt to the new software with limited training.
Chapter 2

REQUIREMENTS ANALYSIS

Product Perspective

This project is aimed toward a law firm that has considerable number of employees who intend to use this product for their daily activities, and so need software assistance for efficient client and case management. This product should be user-friendly, quick to learn and reliable for the above purpose.

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 platform.

Product Features

This project provides various functionalities including Attorney calculators for different calculations such as date calculator for calculating court days, judgment interest calculator for partial payments and a motion date calculator. The Scheduling calendar developed for all the employees in the law firm has options to create events for the given dates, create docket items for future dates, and reschedule events for specific dates. In addition, the Clipboards provide two types of data, first one with “statutes” data displaying the cases with last date due today or in past or in future two weeks, and the second one with “discovery” data.

The functionalities in scope are as follows:

1. Application Administration forms for set-up of the following:
- Create users

- Create Roles (clients, employees, attorneys)

- Users and Role Assignments

2. Attorney calculator allows different calculations such as date calculator for calculating court days, judgment interest calculator for partial payments and a motion date calculator to find the motion date.

3. Scheduling calendar for all the employees in the law firm with options to create events for the given dates, create docket items for future dates, reschedule events for specific dates and delete completed events.

4. Daily and weekly views of the calendar to all the employees after being logging in and also options to send calendar for the selected week by e-mail & also print calendar for the specified week.

5. Create clipboards with two types of data, first one with “statutes” data displaying the cases with last date due today or in past or in future two weeks, and the second one with “discovery” data.

6. Create and Assign Tasks to all employees in the law firm. Ability to view, edit comments on the assigned task. The comments edited by a person are viewed immediately by the assignee.
7. Create, view and Edit case information by all the employees in the firm. A particular case can be retrieved by entering either Client Reference Number or Last Name, First Name or Attorney Internal Case #.

8. Tabbed views of case related information such as Linked Events, Linked documents, Linked Notes and Contacts. Employees can edit any of the case related information by selecting the appropriate tabs.

9. Each of the Clients is assigned with a unique Login name and password, using which they can login and view the case status details & the detailed notes that are logged by the law firm employees periodically.

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 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.

Assumptions and Dependencies

1. It is assumed that the system will be developed using the ASP.NET technology.
2. It is assumed that the system will be able to interface with an email server in order to send the calendar by email to the contacts in the system.

3. It is assumed that the system will interface with a SQL Server 2005 database.

**Functional Requirements**

The functional requirements of this project are described using use cases.

**Use Cases**

Use case modeling is a technique used to describe what the new software being developed should do or what the already available software does. It is used in the Analysis phase of software development to express clearly the high-level requirements of the software. Use case modeling helps to ensure that it captures the actual requirements of the customer for whom the system is being developed. Often, software projects fail when the developers do not understand customer’s needs.

**Actor:** - An actor is user of the system. An actor is someone or something, which interacts with the system. An actor need not be a human being; it can be any application. An actor interacts with the system by sending or receiving messages.

**Use Case:** - The functionality provided by the system is represented as a use case. It is typically described in verb form.

**Relation Ships**

Different kinds of relationships exist between use cases and actors.
This use case describes how to log into the Attorney website from the Internet. Each employee at the law firm is assigned with login credentials.

**Actors**

Admin, Attorney Peter, Employee, Client, System
Preconditions

The following preconditions must be true before the initiation of the use case.

1. User has logged into the Internet.
2. System has validated user credentials entered in the Login Page.

Basic Flow

The following steps represent the basic flow of events occurring from the home page also shown in Figure 1 below.

1. User selects the appropriate link from the Attorney Personal website home page.
2. System determines User’s role from login.
3. System displays the Case/Client Management website home page.
4. System captures Today’s Date for Home Page/Date.
5. System captures User’s Login Name for Home Page/Welcome Message.

Figure 1: User Authentication
Post Conditions

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

1. User has logged into the application.
2. User is able to view/select various applications depending on assigned role.

UC: Date Calculator

This use case describes how to use date calculator after logging in successfully.

Preconditions

The following preconditions must be true before the initiation of the use case.

1. User has logged into the Internet.
2. System has validated user credentials entered in the Login Page.

Basic Flow

The following steps represent the basic flow of events in Date Calculator screen also shown in Figure 2 below.

1. User navigates to the home page and selects Date calculator.
2. User selects the required date by clicking on the calendar icon or can directly enter a date in mm/dd/yyyy format.
3. User enters the number of days (integer value) in the text box.
4. User can choose to type data and then increment or decrement the value by using appropriate buttons appearing on right side of the text box.
5. User then chooses to display the days, court days, weeks, months, quarters or years for the selected date interval.
6. User selects “Calculate” button.

7. System calculates and displays the interval on the screen.

8. Similarly, user can use the screen to calculate difference between two given dates.

9. System calculates and displays the total days and workdays for the given interval.

---

**Figure 2: Date Calculator**

**Post Conditions**

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

1. User is able to enter values multiple times and display results on the same screen.

2. User can choose to go back to the home page by clicking on “Go back” link.

**UC: Motion Date Calculator**

This use case describes how to use motion date calculator after logging in successfully.

**Preconditions**

The following preconditions must be true before the initiation of the use case.

1. User has logged into the Internet.
2. System has validated user credentials entered in the Login Page.

Basic Flow

The following steps represent the basic flow of events in the Motion Date Calculator screen also shown in Figure 3 below.

1. User navigates to the home page and selects Motion Date calculator.
2. System displays the Motion Date Calculator screen.
3. User selects the required date by clicking on the calendar icon or can directly enter a date in mm/dd/yyyy format.
4. User selects “Calculate” button.
5. System calculates and displays the motion date (adds 16 court days and 5 calendar days to the given date) on the screen.

![Diagram](image)

Figure 3: Motion Date Calculator

Post Conditions

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

1. User is able to enter values multiple times and display results on the same screen.
2. User can choose to go back to the home page by clicking on “Go back” link.
UC: Judgment Interest Calculator

This use case describes how to use Judgment Interest calculator after logging in successfully.

 Preconditions

The following preconditions must be true before the initiation of the use case.

1. User has logged into the Internet.
2. System has validated user credentials entered in the Login Page.

 Basic Flow

The following steps represent basic flow of events in Judgment Interest Calculator screen also shown in Figure 4 below.

1. User navigates to the home page and selects “Judgment Interest calculator”.
2. System displays the Judgment Interest Calculator screen.
3. User enters the “Total Judgment Amount” in the text box.
4. User enters the “Interest per Year” (integer value) in the text box.
5. User clicks on “Add Partial Payment”.
6. User then enters “Number of days” and “Payment Amount” in the pop-up window.
7. User selects “Calculate” button.
8. System calculates and displays the “New accrued daily interest” on the screen.
9. User can add as many partial payments as needed and system recalculates the daily interest based on the remaining balance.
Post Conditions

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

1. User is able to enter values multiple times and display results on the same screen.
2. User can choose to view the home page by clicking on “Go to Home Page” link.

UC: Scheduling Calendar

This use case describes how to use scheduling calendar after logging in successfully.

Preconditions

The following preconditions must be true before the initiation of the use case.

1. User has logged into the Internet.
2. System has validated user credentials entered in the Login Page.

Basic Flow

The following steps represent the basic flow of events in Scheduling Calendar screen also shown in Figure 5 below.
1. User navigates to the home page and selects “Scheduling Calendar”.

2. System displays the Scheduling Calendar screen.

3. By default, system displays the calendar with events scheduled for Today’s date.

4. User can view the calendar for desired date by entering in the text box.

5. System refreshes the screen and displays calendar for the selected date.

![Diagram]

Figure 5: Scheduling Calendar

**Post Conditions**

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

1. User is able to enter values multiple times and display results on the same screen.

2. User can select various options such as Reschedule, Edit or Delete an event by clicking on the options.

3. User can choose to go back to the home page by clicking on “Go Home” link.

**UC: Add a New Event**

This use case describes how to add a new event in the scheduling calendar.
Preconditions

The following preconditions must be true before the initiation of the use case.

1. User has logged into the Internet.
2. System has validated user credentials entered in the Login Page.

Basic Flow

The following steps represent basic flow of events while adding a new event in the Calendar also shown in Figure 6 below.

1. User navigates to the home page and selects “Scheduling Calendar”.
2. System displays the Scheduling Calendar screen.
3. User can view the calendar for desired date by entering in the text box.
4. System refreshes the screen and displays calendar for the selected date.
5. User then clicks on “Add a new event” link on the screen.
7. User enters the event information such as – Who, When, What, Where, Description, Name, Case #, Priority, Type and Status.
8. User clicks on “Save” button.
9. System creates a new event and displays it on the scheduling calendar for the specified date.
**Post Conditions**

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

1. User is able to create multiple events and display them on calendar.
2. User can select various options such as Edit or Delete the newly created event.
3. User can choose to go back to the home page by clicking on “Go Home” link.

**UC: Edit/Delete an Event**

This use case describes how to edit or delete an event already created in the scheduling calendar.

**Preconditions**

The following preconditions must be true before the initiation of the use case.

1. User has logged into the Internet.
2. System has validated user credentials entered in the Login Page.
Basic Flow

The following steps represent the basic flow of events while editing/deleting an event also shown in Figure 7 below.

1. User navigates to the home page and selects “Scheduling Calendar”.
2. User can view the calendar for desired date by entering in the text box.
3. System refreshes the screen and displays calendar for the selected date.
4. User then selects “Edit/Delete an Event” option.
5. User can edit any of the event information such as – Who, When, What, Where, Description, Name, Case #, Priority, Type, Status.
6. User clicks on “Save” button.
7. System saves the modified event and user can view the updated scheduling calendar by clicking on “View Calendar” link.

Figure 7: Edit/Delete an Event
**Post Conditions**

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

1. User is able to edit multiple events and display them on calendar.
2. User can select various options such as Edit or Delete a saved event.
3. User can choose to go back to the home page by clicking on “Go Home” link.

**UC: Re-schedule an Event**

This use case describes how to re-schedule an event already created in the scheduling calendar.

**Preconditions**

The following preconditions must be true before the initiation of the use case.

1. User has logged into the Internet.
2. System has validated user credentials entered in the Login Page.

**Basic Flow**

The following steps represent the basic flow of events while re-scheduling an event in the Scheduling Calendar also shown in Figure 8 below.

1. User navigates to the home page and selects “Scheduling Calendar”.
2. System displays the Scheduling Calendar screen.
3. User can view the calendar for desired date by entering in the text box.
4. System refreshes the screen and displays calendar for the selected date.
5. User selects “Re-schedule” option.
6. System enters the following information
   - Event Time & Date to Reschedule
- New Date and Time for the Event

7. User clicks on “Save” button.

8. System saves the re-scheduled event and the user can view the updated scheduling calendar by clicking on “View Calendar”.

Figure 8: Re-Schedule Event

**Post Conditions**

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

1. User is able to reschedule multiple events and refresh calendar on same screen.

2. User can choose to go back to the home page by clicking on “Go Home” link.

**UC: Add New Rule**

This use case describes how to add a new rule for docketing items in the scheduling calendar.

**Preconditions**

The following preconditions must be true before the initiation of the use case.
1. User has logged into the Internet.

2. System has validated user credentials entered in the Login Page.

Basic Flow

The following steps represent the basic flow of events while adding a new rule also shown in Figure 9 below.

1. User navigates to the home page and selects “Scheduling Calendar”.

2. System displays the Scheduling Calendar screen.

3. User clicks on “Add a new rule” link on the screen.

4. System opens a new page for entering the details of new rule.

5. User enters the Rule information such as - Rule Name, County, Number of Events

6. User clicks on “Create Events” to create a set of events that belong to this rule.

7. User enters the required information for each of the events – What, Interval, Duration, Time, Type, and Description.

8. User can click on “Remove Event” to delete any unwanted events before saving the entire rule with the set of events.

9. User clicks on “Save” button.

10. System saves the new rule and its associated events and refreshes the entire page for the user to create new rules.
Post Conditions

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

1. User is able to create multiple rules at once using the same screen.
2. User can select various options such as Edit or Delete the newly created rule.
3. User can choose to view the home page by clicking on “Go to Home Page” link.

UC: Email Calendar

This use case describes how to send a calendar within specific date range from the scheduling calendar by e-mail.

Preconditions

The following preconditions must be true before the initiation of the use case.

1. User has logged into the Internet.
2. System has validated user credentials entered in the Login Page.
Basic Flow

The following steps represent the basic flow of events while sending calendar by e-mail also shown in Figure 10 below.

1. User navigates to the home page and selects “Scheduling Calendar”.
2. System displays the Scheduling Calendar screen.
3. By default, system displays the calendar with events scheduled for Today’s date.
4. User can view the calendar for desired date by entering in the text box.
5. System refreshes the screen and displays calendar for the selected date.
6. User then clicks on “Email Calendar” link on the screen.
7. System opens a new window for entering the email address.
8. User enters the Email Address.
9. User clicks on “Send” and closes the current window.
10. System sends the scheduling calendar to the specified email address.

Figure 10: Email Calendar
**Post Conditions**

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

1. User can email calendars for specified dates multiple times using the same screen.
2. User can choose to view the home page by clicking on “Go to Home page” link.

**UC: Print Calendar**

This use case describes how to print a calendar within specific date range from the scheduling calendar.

**Preconditions**

The following preconditions must be true before the initiation of the use case.

1. User has logged into the Internet.
2. System has validated user credentials entered in the Login Page.

**Basic Flow**

The following steps represent the basic flow of events while printing the calendar also shown in Figure 11 below.

1. User navigates to the home page and selects “Scheduling Calendar”.
2. System displays the Scheduling Calendar screen.
3. By default, system displays the calendar with events scheduled for Today’s date.
4. User can view the calendar for desired date by entering in the text box.
5. System refreshes the screen and displays calendar for the selected date.
6. User then clicks on “Print Calendar” link on the screen.
7. System opens a new window with a confirmation message.
8. User clicks on “Yes” button.
9. System opens the “default printer” options page for setting options on how to print the scheduling calendar.

Post Conditions

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

1. User can view the print area i.e. the calendar to be printed in a separate window.

UC: Add/Edit/Delete Contacts

This use case describes how to add, edit or delete contacts in “Contacts Manager” screen.

Preconditions

The following preconditions must be true before the initiation of the use case.

1. User has logged into the Internet.

2. System has validated user credentials entered in the Login Page.

Basic Flow

The following steps represent the basic flow of events while adding a new contact also shown in Figure 12 below.
1. User navigates to the home page and selects “Contacts Manager”.

2. System displays the Contacts Management screen.

3. User clicks on “Add New Contact” link on the screen.

4. System opens a new page for the user to enter the details of the new contact.

5. User can edit the contact information - Last Name, First Name, Label, Day Phone, Evening Phone, City, State, Zip code, Fax, Pager, Email, and Attorney.

6. User clicks on “Save” button.

7. User clicks on “OK” to close the current window.

**Figure 12: Add New Contact**

**Post Conditions**

The following post conditions must be true after the completion of the use case.
1. User is able to add, edit or delete contacts using the same screen.

2. User can choose to view the home page by clicking on “Go to Home page” link.

**UC: Add/Edit/Delete What**

This use case describes how to add, edit or delete “What” entries while adding new events in the Scheduling Calendar screen.

**Preconditions**

The following preconditions must be true before the initiation of the use case.

1. User has logged into the Internet.

2. System has validated user credentials entered in the Login Page.

**Basic Flow**

The following steps represent the basic flow of events for adding, editing or deleting “What” column entries while adding new event in the Scheduling Calendar also shown in Figure 13 below.

1. User navigates to the home page and selects “Scheduling Calendar”.

2. System displays the Scheduling Calendar screen.

3. User clicks on “Add New Event” link on the screen.

4. System opens a new page for the user to enter the details of the new event.

5. User clicks on “…” button against the “What” dropdown.

6. System opens a new window where the user can either add new “What” entries or edit/delete any existing entries. User can edit the following information:
   
   - What
   - Description
7. User clicks on “Save” button.

8. User clicks on “OK” to close the current window.

Post Conditions

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

1. Users are able to add, edit or delete entries in the same screen and refresh the list in “What” dropdown after closing the pop-up window.

2. User can choose to view the home page by clicking on “Go to Home page” link.

UC: Add/Edit/Delete Where

This use case describes how to add, edit or delete “Where” entries while adding new events in the Scheduling Calendar screen.

Preconditions

The following preconditions must be true before the initiation of the use case.
1. User has logged into the Internet.
2. System has validated user credentials entered in the Login Page.

**Basic Flow**

The following steps represent the basic flow of events for adding, editing or deleting “Where” column entries while adding new event in the Scheduling Calendar also shown in Figure 14 below.

1. User navigates to the home page and selects “Scheduling Calendar”.
2. System displays the Scheduling Calendar screen.
3. User clicks on “Add New Event” link on the screen.
4. System opens a new page for the user to enter the details of the new event.
5. User clicks on “…” button against the “Where” dropdown.
6. System opens a new window where the user can either add new “Where” entries or edit/delete any existing entries. User can edit the following information:
   - Where
   - Description
7. User clicks on “Save”
8. User clicks on “OK” to close the current window.
Figure 14: Add/Edit/Delete Where Entries

Post Conditions

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

1. Users are able to add, edit or delete entries in the same screen and refresh the list in “Where” dropdown after closing the pop-up window.

2. User can choose to view the home page by clicking on “Go to Home page” link.

UC: Add a New Docket Item

This use case describes how to add a new docket items in the scheduling calendar.

Preconditions

The following preconditions must be true before the initiation of the use case.

1. User has logged into the Internet.

2. System has validated user credentials entered in the Login Page.
Basic Flow

The following steps represent the basic flow of events while adding a new docket item also shown in Figure 15 below.

1. User navigates to the home page and selects “Scheduling Calendar”.
2. System displays the Scheduling Calendar screen.
3. User can view the calendar for desired date by entering in the text box.
4. System refreshes the screen and displays calendar for the selected date.
5. User then clicks on “Add new Docket Item” link on the screen.
6. System opens a new window for entering the details of new item.
7. User selects the “Rule” to which the Docket Item has to be applied.
8. User enters the Event information – Who, When, Where, Name, Case #, Priority and Status.
9. User clicks on “Save” button.
10. System calculates the date for each particular event in the rule based on the interval formula entered while creating the rule.
11. System creates multiple events associated to the rule at once and displays it on the scheduling calendar for the specified date.
Post Conditions

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

1. User is able to create multiple events and display them on calendar.
2. User can select various options such as Edit or Delete the newly created event.
3. User can choose to view the home page by clicking on “Go to Home Page” link.

UC: Add/Edit/Delete Tasks to Assign

This use case describes how to add new tasks and assign them to other users in the “Tasks” page.

Preconditions

The following preconditions must be true before the initiation of the use case.

1. User has logged into the Internet.
2. System has validated user credentials entered in the Login Page.
Basic Flow

The following steps represent the basic flow of events for adding, editing or deleting tasks to be assigned to user in the “Tasks Manager” screen also shown in Figure 16 below.

1. User navigates to the home page and selects “Go to Tasks”.
2. System displays the “Tasks Management” screen.
3. User navigates to the “Tasks Assigned by You” data grid.
4. User then clicks on “Add” link to add a new task.
5. System makes all the fields editable for entering the details of new task.
6. User enters the following information.
   - Task
   - Description
   - Assigned To
7. User clicks on “Save”
8. System creates the new task and updates the two fields – “Created By” and “Last Edited By” fields with current logged in user; and “Last Edited Date” with today’s date.
9. System displays the new task immediately in the same grid for the user to view.
Post Conditions

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

1. User is able to create multiple tasks and display them on the grid.

2. User can select various options such as Edit or Delete the newly created task.

UC: View/Edit Tasks Assigned to user

This use case describes how to view the tasks assigned to you by other users; and edit the comments against each task in the “Tasks” page.

Preconditions

The following preconditions must be true before the initiation of the use case.

1. User has logged into the Internet.

2. System has validated user credentials entered in the Login Page.
Basic Flow

The following steps represent the basic flow of events for viewing or editing tasks already assigned to user in the “Tasks Manager” screen also shown in Figure 17 below.

1. User navigates to the home page and selects “Go to Tasks”.

2. User navigates to the “Tasks Assigned to You” in the “Tasks” screen.

3. User then clicks on “Edit” link to edit a task.

4. System makes the fields editable for entering the details of the task.

5. User enters the information - Mark as Read, Comments.

6. User clicks on “Save” button.

7. System saves the task and updates the two fields “Last Edited By” with current logged in user; and “Last Edited Date” with today’s date.

8. System displays the modified task immediately in the same data grid.

Figure 17: View/Edit Tasks Assigned to User
**Post Conditions**

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

1. User is able to edit comments on multiple tasks and display them on the grid.
2. User can choose to view the home page by clicking on “Go to Home Page” link.

**UC: Add new case**

This use case describes how to add new case details in the “Cases” page.

**Preconditions**

The following preconditions must be true before the initiation of the use case.

1. User has logged into the Internet.
2. System has validated user credentials entered in the Login Page.

**Basic Flow**

The following steps represent the basic flow of events for adding new case details also shown in Figure 18 below.

1. User navigates to the home page and selects “Go to Case Management”.
2. System displays the “Case Management” screen.
3. User clicks on “Add new case” link
4. User enters the Case information - Case Name, Client Ref Num, Attorney, Court, Opened Date, Closed Date, Court Num, Insurance Claim, DOL, Case Type, Case Due Date, Case File Num and Contact.
5. User clicks on “Save” button.
6. System creates a new case with automatically generated case number and displays it on the screen for the user to view.
Post Conditions

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

1. User is able to create multiple cases and display them on the screen.

2. User can choose to view the home page by clicking on “Go to Home Page” link.

UC: Edit/Delete a case

This use case describes how to edit or delete case details in the “Cases” page.

Preconditions

The following preconditions must be true before the initiation of the use case.

1. User has logged into the Internet.

2. System has validated user credentials entered in the Login Page.

Basic Flow

The following steps represent the basic flow of events for editing or deleting case details also shown in Figure 19 below.
1. User navigates to the home page and selects “Go to Case Management”.

2. System displays the “Case Management” screen.

3. User clicks on “Edit/Delete a case” link

4. User enters the case number to perform edit or delete operation.

5. User clicks on “Retrieve Case”

6. User can modify the Case information - Case Name, Client Ref Num, Attorney, Court, Opened Date, Closed Date, Court Num, Insurance Claim, DOL, Case Type, Case Due Date, Case File Num and Contact.

7. User clicks on “Save” button.

8. System saves the case details and displays on the screen for the user to view.

Figure 19: Edit/Delete a Case

**Post Conditions**

The following post conditions must be true after the completion of the use case.
1. User is able to edit multiple cases and display them on the screen.

2. User can choose to view the home page by clicking on “Go to Home Page” link.

**UC: View Case Details**

This use case describes how to view case details using different tabs in the “Cases” page.

**Preconditions**

The following preconditions must be true before the initiation of the use case.

1. User has logged into the Internet.

2. System has validated user credentials entered in the Login Page.

**Basic Flow**

The following steps represent the basic flow of events for viewing case details also shown in Figure 20 below.

1. User navigates to the home page and selects “Go to Case Management”.

2. System displays the “Case Management” screen.

3. User clicks on “View case details” link.

4. User enters the case number to retrieve the details.

5. User clicks on “Retrieve Case” button.

6. System displays the following tabs with details of the case.
   
   - Case Details: This tab provides all the general information about the case including the case opened date, status etc.
   
   - Linked Events: This tab displays all the events linked to this particular case number for which user created events in scheduling calendar.
- Linked Documents: This tab displays all the documents linked to this particular case number. User can upload new documents to be stored on the server for this case. User can also download or view the latest copy of a document available on the server.

- Linked Notes: This tab displays all the notes entered at various stages of the case proceedings. This information will be available for the clients to view as they retrieve particular case information.

![Diagram](image)

Figure 20: View Case Details

**Post Conditions**

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

1. User is able to view different tabs with case details on the same screen.

2. User can choose to view the home page by clicking on “Go to Home Page” link.
UC: Add case notes

This use case describes how to add notes about a case that can be later viewed by the client in the “Cases” page.

Preconditions

The following preconditions must be true before the initiation of the use case.

1. User has logged into the Internet.
2. System has validated user credentials entered in the Login Page.

Basic Flow

The following steps represent the basic flow of events for adding case notes as shown in Figure 21 below.

1. User navigates to the home page and selects “Go to Case Management”.
2. User enters the case number to retrieve the details.
3. User clicks on “Retrieve Case” button.
4. System displays the following tabs with case information
   - Case Details
   - Linked Events
   - Linked Documents
   - Linked Notes
5. User clicks on “Linked Notes” tab to view notes for that case.
6. User clicks on “Add Notes” button.
7. User enters the text information relevant to the case.
8. User clicks on “Save” button.
9. System displays the newly added notes and the corresponding “Added Date” with the current system date.

Figure 21: Add Case Notes

Post Conditions

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

1. User is able to add multiple case notes and display them on the screen.

2. User can choose to view the home page by clicking on “Go to Home Page” link.

UC: View Clipboard Statutes

This use case describes how to view statutes data and edit comments on them in the “Clipboards” screen.

Preconditions

The following preconditions must be true before the initiation of the use case.

1. User has logged into the Internet.
2. System has validated user credentials entered in the Login Page.

**Basic Flow**

The following steps represent basic flow of events for viewing Clipboards - Statutes Data also shown in Figure 22 below.

1. User navigates to the home page and selects “Go to Clipboards”.
2. System displays the “clipboards” with separate grids for statutes and discovery.
3. User can view the statutes data with the information - Date due, File Num, Case Num, Contact, Phone Num, Case Type and Comments.
4. User can edit the comments on a particular case statute.
5. User clicks on “Save” button.
6. System saves the comments and updates the “Edited By” column with the current user and “Edited Date” column with today’s date.

![Diagram of Basic Flow](Image)

**Figure 22: View Clipboard Statutes Data**
**Post Conditions**

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

1. User is able to edit multiple comments and display them on the screen.
2. User can choose to view the home page by clicking on “Go to Home Page” link.

**UC: View Clipboard Discovery log**

This use case describes how to view discovery log data and edit comments on them in the “Clipboards” screen.

**Preconditions**

The following preconditions must be true before the initiation of the use case.

1. User has logged into the Internet.
2. System has validated user credentials entered in the Login Page.

**Basic Flow**

The following steps represent the basic flow of events for viewing clipboards – Discovery Log also shown in Figure 23 below.

1. User navigates to the home page and selects “Go to Clipboards”.
2. System displays the “clipboards” with separate grids for statutes and discovery.
3. User can view the discovery data with the information such as - Date due, File Num, Case Num, Contact, Phone Num, Case Type and Comments.
4. User can edit the comments on a particular case discovery log.
5. User clicks on “Save” button.
6. System saves the comments and updates the “Edited By” column with the current user and “Edited Date” column with today’s date.
Post Conditions

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

1. User is able to edit multiple comments and display them on the screen.

2. User can choose to go back to home page by clicking on “Go to Home Page” link.

External Interface Requirements

User Interfaces

The user interface for the system will be a web page on the Internet. The user interface will be limited to the types of controls that can be generated using HTML, Java script, AJAX and Cascading Style Sheets.
When a function is performed like adding, editing or deleting, the active window is the one performing the action. At this time, the main window is inactive and cannot be accessed unless the current active window is closed.

**Software Interfaces**

1. Operating System: The software is being designed to run on Windows Vista/XP/2000. These include the latest version of Internet Information Services.
2. Web Server: The software is being designed to run on Internet Information Server version 6.0.
4. Libraries: The software will be created using the Microsoft .NET version 3.5 framework.

**Non-functional Requirements**

**Security Requirements**

1. Passwords shall be displayed as “*” in the web pages wherever required.
2. Proper authentication is required for users to access any of the web pages including the home page.
3. Every user of the system is assigned a unique login and password to access the application over the internet.
4. Microsoft .NET framework ensures security of data, for example passwords that are being transmitted over the internet.
Software Quality Attributes

Reliability: Data, as entered, must be correctly stored in the database. In addition, the database should commit transactions so that partial entries are not stored in the database.

Usability: Individuals of varying skill level and technical competence will use the system.

Maintainability: The code and design need to be documented well enough and designed such that a new project member with the same amount of academic and co-op experience can easily ramp up the project.

Developer Environment

This project is developed using Microsoft Visual Studio .NET Framework along with Microsoft SQL Server 2005 relational database system. The Microsoft .NET Framework, is a prefabricated infrastructure to develop desktop and Internet applications. The infrastructure has four main parts, as shown in Figure 24 below.

![Figure 24: .NET Framework Components](image-url)
1. **Common Language Runtime (CLR):** The CLR acts like a virtual machine that runs managed codes and offers advanced features such as automatic memory management (also known as garbage collection), standardized versioning, code access security, and seamless interoperability with Common Object Model (COM) components and Dynamic Link Libraries (DLLs). The managed code is written in a high-level CLR-compliant language (such as Visual Basic .NET or C#) and is then compiled into an Intermediate Language (IL). The IL code itself cannot run directly on any computer, i.e., a Just-In-Time compiler or JITter interprets it. This intermediate step slows down the performance but provides the .NET Framework with a certain amount of platform independence as each platform can have its own JITter [3].

2. **Class Library:** The class library contains the foundation classes that are used to build applications. The library has a tree structure where each class inherits the functionalities of its parent. The developer can extend the .NET Framework by creating custom classes that inherit from those of the prebuilt tree.

3. **Windows Forms:** Windows forms are a package that provides ready-to-use user interface elements to build powerful front ends for desktop applications. Such elements, which are also called Windows controls, include windows, buttons, dialog boxes, tree views, data grids, and so on.

4. **ASP.NET:** ASP.NET provides support to build and run Web applications. Its main functions include the following:
   - Prefabricated controls that do for Hypertext Markup Language (HTML) pages what Windows controls do for desktop applications.
− A Web Server runtime environment that dynamically generates HTML pages in response to input received from the client.

− Advanced services such as data caching to speed up documents that are often downloaded, session state to personalize clients, and security to block malicious clients. [4]
Chapter 3
SYSTEM DESIGN

This project is implemented using the client-server architecture. It is hosted on a web server and it responds to clients as shown in figure 25 below. Web Applications provide content from the server to the client. Clients obtain those contents from internet and view them using a web browser.

Building a Web Application

“Web applications are popular due to the ubiquity of the browser as a client, sometimes called a thin client. The ability to update and maintain web applications without distributing and installing software on potentially thousands of client computers is a key reason for their popularity.” [4]
This project runs on MS IIS (Microsoft Internet Information Services) Web Server. IIS is a web server application and set of feature extension modules created by Microsoft for use with Microsoft Windows. It is one of the world’s most popular web servers in terms of overall websites behind the industry leader Apache HTTP Server.

.NET Application Architecture

In the classic three-tier design of a .NET application, the architecture breaks down into three major areas of functionality:

1. The data layer manages the physical storage and retrieval of data
2. The business layer maintains business rules and logic
3. The presentation layer houses the user interface and related presentation code.

Presentation Tier: In the presentation layer, the code-behind mechanism for ASP.NET pages and user controls is a prominent example of a layered design. Designers do not have to worry about messing up code to make user interface changes, and developers do not have to worry about sifting through the user-interface to update code.

Data Tier: The data tier has the tables that define the physical storage of data in a database, stored procedures and views that allow manipulating data as it goes into and out of those tables.

Business Tier: A business object is a component that encapsulates the data and business processing logic for a particular business entity. Business tier contains logic for retrieving persistent data from the data-tier and placing it into business objects. [3]
The following is a block diagram (figure 26) of all the modules in the application and the relationships between them. The following view includes arrows to indicate flow of data and/or control.

Figure 26: Block Diagram

**High Level Design**

The High-Level Design (HLD) provides an overview of the design components of the web application. The design is represented through elaborations on class diagrams for each of the core sub-systems. The purpose of the high-level design is to define key
classes, as well as the interactions among instances of these classes (objects). Inputs to the HLD consist of the functional requirements defined in the above sections.

The HLD represents a decomposition of all the sub-systems down to the class level. It is envisioned that the HLD will form the basis for the detailed design efforts.

Following are the key goals and constraints governing the design decisions in the HLD.

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

The figure 27 below depicts the class diagram, which gives an overview of the entire system by showing its classes and relationships among them.
The figure 28 below depicts the class diagram for the scheduling calendar developed for the law firm. The central class is Calendar. Associated with it are NewDocketItem, AddNewEvent, EditDeleteRule, RescheduleEvent and EventUserControl.
The figure 29 below depicts the class diagram for the case management module developed for the law firm. The central class is Cases. Associated with it are AddNewCase, sqlbuilder and Master. The sqlbuilder encompasses all the functions needed to retrieve case details, create, edit and update case details into the database.
Figure 29: Class Diagram for Case Management Module
Chapter 4
DATABASE DESIGN

Database design is very crucial for the performance of the system. A good database design will contribute to software development and improve performance. It facilitates data retrieval, storage and modification.

About SQL Server 2005
Microsoft SQL Server is a relational model database server produced by Microsoft. Its primary query languages are T-SQL and ANSI SQL. SQL Server 2005 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. For relational data, T-SQL has been augmented with error handling features (try/catch) and support for recursive queries (Common Table Expressions). SQL CLR was introduced with the SQL Server 2005 to let it integrate with the .NET Framework. [5]

Database Tables and Database Diagram
Considering the nature of the project, the tables for the database are designed to minimize redundancy. There are 14 tables in the database. This kind of a structure facilitates CASCADE DELETE and avoids any duplicate entries. The tables have also been designed to adhere to the Fourth Normal Form (4NF of BCNF) by using a systematic way ensuring that the database structure is suitable for general-purpose querying. Another
advantage of this project is that the system can easily be extended to incorporate additional functionalities. These functionalities are discussed in the Future Scope section of the document. The following figure 30 shows all the tables in the database and the relationships between them.

Figure 30: Entity-Relationship Diagram

Sample of Database Scripts

Following are some sample of the database scripts used in the project.

```sql
CREATE TABLE dbo.appointments (
    ID INT NOT NULL AUTO_INCREMENT,
    Date VARCHAR(50) NOT NULL,
    ...)
```
CREATE TABLE dbo.Cases (
  [Case Num] INT NOT NULL AUTO_INCREMENT,
  [Case Name] VARCHAR(50) NOT NULL,
  [Client Ref Num] VARCHAR(50),
  Attorney VARCHAR(50),
  Court VARCHAR(50),
  [Opened Date] VARCHAR(50),
  [Closed Date] VARCHAR(50),
  [Court Num] VARCHAR(50),
  [Insurance Claim] VARCHAR(200),
  SOL VARCHAR(50),
  [Case Type] VARCHAR(200),
  [Case Due Date] VARCHAR(50),
  [Case File Num] VARCHAR(50),
  Contact VARCHAR(50),
  PRIMARY KEY (Case Num)
);
This chapter describes in detail the .NET technologies used in this project and shows some sample code to explain in detail, the main features of the application.

**Implementation Details of Scheduling Calendar Feature**

The code snippets in the following sections depict the functionality of the Scheduling Calendar, which is an integral module in the entire application. The Scheduling Calendar provides the following major features:

- Load Calendar data grid with the events between specific dates
- Display Calendar with weekly view starting from specific date
- Create a new event for the given date and time
- Update details of an existing event
- Delete a completed event

**Development of Web Interface using ASP.NET**

ASP.NET is a web application framework developed to allow programmers to build dynamic web sites, web applications and web services.

Advantages using ASP.NET:

1. ASP.NET drastically reduces the amount of code required to build large applications.
2. ASP.NET makes development simpler and easier to maintain with an event-driven, server-side programming model.
3. ASP.NET pages are easy to write and maintain because the source code and HTML are together. The source code is executed on the server.

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

5. The HTML produced by the ASP.NET page is sent back to the browser.

6. The Web server continuously monitors the pages, components and applications running on it. If it notices memory leaks, infinite loops, other illegal software or activities, it seamlessly kills those activities and restarts itself.

7. ASP.NET validates information (validation controls) entered by the user without writing a single line of code.

8. ASP.NET easily works with ADO.NET using data binding and page formatting features. [2]

Firstly, we can create the design of the web page by dragging and dropping appropriate controls in the Design view of Microsoft Visual Studio .net. Here, the Scheduling Calendar can be displayed as a data grid with Rows as time intervals and Columns as dates for which the calendar is displayed. Following is the code generated when the Scheduling Calendar is created as a data grid of rows and columns.

```xml
<asp:GridView ID="GridView1" runat="server"
    onrowcreated="GridView1_RowCreated" Width="100%" CellPadding="4"
    onselectedindexchanged="GridView1_SelectedIndexChanged" BackColor="White"
    BorderColor="#3366CC" BorderStyle="None" BorderWidth="1px"
    AlternatingRowStyle-BackColor="#CCCCFF" Height="167px">
    <FooterStyle BackColor="#003399" Font-Bold="True" ForeColor="#CCCCFF"/>
    <HeaderStyle BackColor="#003399" Font-Bold="True" ForeColor="#CCCCFF"/>
</asp:GridView>
```
For displaying the Scheduling Calendar for specific dates, a text box can be created which specifies the starting date for the Weekly Calendar. A calendar extender AJAX control can be added to this text box so that when user clicks on the text box, a calendar appears for the user to select a particular date. Following is the code generated for creating a text box control and attaching an AJAX calendar extender to it.

```
<asp:TextBox ID="txtcalendardate" runat="server" Enabled="False" Height="22px"></asp:TextBox>
<asp:Calendar ID="caldisplay" runat="server" BackColor="White" BorderColor="#3366CC" BorderWidth="1px" CellPadding="1" DayNameFormat="Shortest" Font-Names="Verdana" Font-Size="8pt" ForeColor="#003399" Height="100px" NextMonthText="" onselectionchanged="caldisplay_SelectionChanged" PrevMonthText="" SelectMonthText="" SelectWeekText="" ShowGridLines="True" ShowNextPrevMonth="False" Width="150px">
    <SelectedDayStyle BackColor="#009999" Font-Bold="True" ForeColor="#CCFF99"/>
    <SelectorStyle BackColor="#99CCCC" ForeColor="#336666"/>
    <WeekendDayStyle BackColor="#CCCCCC"/>
    <TodayDayStyle BackColor="#99CCCC" ForeColor="White"/>
    <OtherMonthDayStyle ForeColor="#999999"/>
    <NextPrevStyle Font-Size="8pt" ForeColor="#CCCCCC"/>
    <DayHeaderStyle BackColor="#99CCCC" ForeColor="#336666" Height="1px"/>
    <TitleStyle BackColor="#003399" BorderColor="#3366CC" BorderWidth="1px" Font-Bold="True" Font-Size="8pt" ForeColor="#CCCCCC" Height="25px"/>
</asp:Calendar>
```
Storing and Retrieving Data using ADO.NET

ADO.NET is a set of classes that expose data access services to the .NET programmer. ADO.NET provides functionality to developers writing managed code. ADO.NET provides consistent access to data sources such as Microsoft SQL Server, as well as data sources exposed through OLE DB and XML. Data-sharing consumer applications can use ADO.NET to connect to these data sources and retrieve, manipulate, and update data.


Figure 31: ADO .NET Data Architecture
After creating the design view, the next step is to display appropriate data in the Scheduling Calendar by retrieving appointments for specific dates from the database. This can be done by creating a new SQL connection using the Connection Manager. Then, an appropriate SQL command is called using this SQL connection.

The following code snippet shows retrieving all events between particular dates in the Scheduling Calendar:

```csharp
Loadgridview(string username, string date)
{
    String connect = ConfigurationManager.ConnectionStrings["msdbConnection String"].ToString();
    SqlConnection con = new SqlConnection(connect);
    SqlCommand cmd = new SqlCommand("sp_weekschedule", con);
    SqlCommand cmd1 = new SqlCommand("sp_gettooltip", con);
    SqlCommand cmd2 = new SqlCommand("select UserName from aspnet_Users a,
        aspnet_Roles b, aspnet_UsersInRoles c where a.UserId = c.UserId and b.RoleId =
        c.RoleId;", con);
    cmd.CommandType = CommandType.StoredProcedure;
    cmd.Parameters.Add(new SqlParameter("@dte", date));
    cmd.Parameters.Add(new SqlParameter("@user", username));
    cmd1.Parameters.Add(new SqlParameter("@dte", date));
    cmd1.Parameters.Add(new SqlParameter("@user", username));
    cmd1.CommandType = CommandType.StoredProcedure;
    SqlDataAdapter da = new SqlDataAdapter(cmd);
    SqlDataAdapter da1 = new SqlDataAdapter(cmd1);
    SqlDataAdapter da2 = new SqlDataAdapter(cmd2);
    con.Open();
    DataSet ds = new DataSet();
    da.Fill(ds);
    DataSet ds1 = new DataSet();
    DataTable dt1 = new DataTable();
    dt1 = ds1.Tables.Add();
    da1.Fill(ds1, "dt1");
    DataSet ds2 = new DataSet();
}
In the above code, the results of the Sql query are stored into the Sql adapter. This Sql adapter is used to fill data into the dataset after opening the Sql connection. Then, a data table is created to save the table of data from the dataset.

Now, the data table can be used to fill the data grid i.e. the Scheduling Calendar with associated events. Following is the code to fill the data grid with table data.

```csharp
GridView1.DataSource = ds;
GridView1.DataBind();
```

After retrieving and displaying appointments in the Scheduling calendar, the next step is to create new events to be added to the appointments scheduler.

The following code snippet shows getting a connection and inserting new events into the “appointments” table:

```csharp
string connect = ConfigurationManager.ConnectionStrings["msdbConnectionString"].ToString();
SqlConnection conn = new SqlConnection(connect);
SqlCommand cmd = new SqlCommand("Insert into appointments values(@Date,@Time,@Who,@What,@Where,@Event,@Priority,@CaseNum,@Type, @Status,@Uname)", conn);
try {
    cmd.Parameters.Add(new SqlParameter("@Date", SqlDbType.VarChar, 50));
    cmd.Parameters["@Date"].Value = Server.HtmlEncode(txtDate.Text);
    cmd.Parameters.Add(new SqlParameter("@Time", SqlDbType.VarChar, 50));
    cmd.Parameters["@Time"].Value = Server.HtmlEncode(ddTime.SelectedItem.Text);
    cmd.Parameters.Add(new SqlParameter("@Who", SqlDbType.VarChar, 50));
    cmd.Parameters.Add(new SqlParameter("@What", SqlDbType.VarChar, 50));
    cmd.Parameters.Add(new SqlParameter("@Where", SqlDbType.VarChar, 50));
    cmd.Parameters.Add(new SqlParameter("@Event", SqlDbType.VarChar, 50));
    cmd.Parameters.Add(new SqlParameter("@Priority", SqlDbType.VarChar, 50));
    cmd.Parameters.Add(new SqlParameter("@CaseNum", SqlDbType.VarChar, 50));
    cmd.Parameters.Add(new SqlParameter("@Type", SqlDbType.VarChar, 50));
    cmd.Parameters.Add(new SqlParameter("@Status", SqlDbType.VarChar, 50));
    cmd.Parameters.Add(new SqlParameter("@Uname", SqlDbType.VarChar, 50));
    cmd.ExecuteReader();
}
```
cmd.Parameters.Add(new SqlParameter("@Who", SqlDbType.VarChar, 50));
cmd.Parameters["@Who"].Value = Server.HtmlEncode(ddWho.SelectedItem.Text);

cmd.Parameters.Add(new SqlParameter("@What", SqlDbType.VarChar, 50));
cmd.Parameters["@What"].Value = Server.HtmlEncode(ddWhat.SelectedItem.Text);

cmd.Parameters.Add(new SqlParameter("@Where", SqlDbType.VarChar, 50));
cmd.Parameters["@Where"].Value = Server.HtmlEncode(ddWhere.SelectedItem.Text);

cmd.Parameters.Add(new SqlParameter("@Event", SqlDbType.VarChar, 500));
cmd.Parameters["@Event"].Value = Server.HtmlEncode(txtDescription.Text);

cmd.Parameters.Add(new SqlParameter("@Priority", SqlDbType.VarChar, 50));
cmd.Parameters["@Priority"].Value = Server.HtmlEncode(ddPriority.SelectedItem.Text);

cmd.Parameters.Add(new SqlParameter("@CaseNum", SqlDbType.VarChar, 50));
cmd.Parameters["@CaseNum"].Value = Server.HtmlEncode(ddCasenum.SelectedItem.Text);

cmd.Parameters.Add(new SqlParameter("@Type", SqlDbType.VarChar, 50));
cmd.Parameters["@Type"].Value = Server.HtmlEncode(ddType.SelectedItem.Text);

cmd.Parameters.Add(new SqlParameter("@Status", SqlDbType.VarChar, 50));
cmd.Parameters["@Status"].Value = Server.HtmlEncode(ddStatus.SelectedItem.Text);

conn.Open();
cmd.ExecuteNonQuery();
}
catch (Exception ex)
{
}
finally
{
conn.Close();
}
Similarly, any other functionality can be implemented using the same logic for retrieving and storing data appropriately in the database.

The following code snippet shows deleting a particular event from the Scheduling Calendar in the database:

```csharp
SqlCommand cmd = new SqlCommand("Delete from appointments WHERE Date = "+
date + ", AND Time = " + time + ", con);
con.Open();
cmd.ExecuteNonQuery();
con.Close();
```

**Implementation of Business Logic using C#.NET**

C# is a multi-paradigm programming language encompassing imperative, functional, generic, object oriented and component oriented programming disciplines. C# is intended to be a simple, modern, general-purpose, object-oriented programming language.

Microsoft Visual C# is Microsoft's implementation of the C# language, which targets the .NET Framework, along with the language services that lets the Visual Studio IDE support C# projects. While the language services are a part of Visual Studio, the compiler is available separately as a part of the .NET Framework. The Visual C# 2008 compiler supports version 3.0 of the C# language specifications. Visual C# supports the Visual Studio Class designer, Forms designer, and Data designer among others. [5]

The business logic in the project is implemented using C#.NET. This includes several server-side validations for checking validity of data entered by user before saving data into the database.
After retrieving and displaying events in the Scheduling Calendar, the next step is to add appropriate business logic to the Scheduling calendar functionality.

The following code can be used to populate the grid view with Calendar events starting from the date selected in the “txtcalendar date” textbox and display the calendar for the one complete week.

```csharp
Protected void GridView1_RowCreated(object sender, GridViewRowEventArgs e)
{
    if (e.Row.RowIndex < 0)
    {
        DateTime dt1 = Convert.ToDateTime(txtcalendar date.Text);
        List<DateTime> dati = new List<DateTime>();

        //The list below stores the dates of the week for which the calendar to be retrieved
        for (int j = 0; j < 7; j++)
        {
            dati.Add(dt1.AddDays(j));
        }

        //Dynamically add the header labels to the grid view with the selected dates of the week
        List<Label> headerlbl = new List<Label>();
        headerlbl.Add(new Label());
        headerlbl[0].Text = “Time”; e.Row.Cells[0].Controls.Add(headerlbl[0]);

        for (int k = 0; k < 7; k++)
        {
            headerlbl.Add(new Label());
            headerlbl[k+1].Text = Convert.ToString(dati[k].DayOfWeek) +”<br>”+ dati[k].ToShortDateString();
            e.Row.Cells[k+1].Controls.Add(headerlbl[k+1]);
        }
    }
}
```
In the above code, a temporary list is created to store the dates of the week for which the calendar to be retrieved. In addition, the header labels to the grid view are dynamically added with the selected dates of the week. This ensures that the Scheduling Calendar is displayed with appropriate dates when the user tries to refresh the grid view with newly selected date in the text box. Therefore, the column headers are populated automatically based on the user selection.

Similarly, the data grid has to be refreshed automatically whenever the user changes the date for which the calendar is to be displayed. Following code can be used to load the entire grid view by calling the loadgridview() function.

```
txtcalendardate.Text = caldisplay SelectedDate.ToShortDateString();
loadgridview(ddUsernames.SelectedItem.Value.ToString(), txtcalendardate.Text);
```

In addition, the following code can be added in the java script section of the web page, in order to print the calendar at any time. The function below is invoked upon the On_Click() event of the Print Calendar link button. [6]

```
<script type="text/javascript" language="javascript">
function PrintCalendar()
{
  var prtContent = document.getElementById("<%=gridpanel.ClientID%>");
  var WinPrint = window.open("","left=100,top=20,width=600,height=500,toolbar=0,scrollbars=yes,status=yes,resizable=yes");
  WinPrint.document.write(prtContent.innerHTML);
  WinPrint.document.close();
  WinPrint.focus();
  WinPrint.print();
  WinPrint.close();
}
```

Using the code above, a separate window can be opened displaying the print area.
Implementation Details of Case Management Feature

The code snippets in this section depict the functionality of the Case Management, which is another major module in the entire application. The Case Management module provides the following major features:

- Load Case details based on the selected case number
- Create a new case
- Update details of an existing case
- Delete an existing case

Case details can be displayed in the form of tabbed views having different kind of information pertaining to each case. Following is the code snippet of the design view of the “Cases” web page where case details are displayed.

```html
<cc1:TabContainer ID="TabContainer1" runat="server" ActiveTabIndex="0" Height="431px">
<cc1:TabPanel HeaderText="Case Details" runat="server" ID="TabPanel1">
<HeaderTemplate>
Case Details</HeaderTemplate>
<ContentTemplate>
<asp:Button ID="btnEdit" runat="server" Text="Edit" onclick="btnEdit_Click" Visible="False" />
<asp:Button ID="btnSave" runat="server" Text="Save" Visible="False" onclick="btnSave_Click" />
<asp:Button ID="btnCancel" runat="server" Text="Cancel" Visible="False" onclick="btnCancel_Click" />
</ContentTemplate></asp:UpdatePanel>
</ContentTemplate></cc1:TabPanel>
```

After creating the design view, the next step is to display appropriate data in the Case Management screen, when the user filters to view data of a particular case number. Following is the code snippet to retrieve case details from the database.
public DataTable getcasedetails(string cn)
{
    SqlCommand cmd = new SqlCommand("sp_getcasedetails", con);
    cmd.CommandType = CommandType.StoredProcedure;
    cmd.Parameters.Add(new SqlParameter("@casenum", cn));
    SqlDataAdapter da = new SqlDataAdapter(cmd);
    con.Open();
    DataSet ds = new DataSet();
    // DataTable dt1 = new DataTable();
    // dt1 = ds.Tables.Add();
    da.Fill(ds);
    con.Close();
    return ds.Tables[0];
}

In the above code, the results of the Sql query are stored into the Sql adapter. This Sql adapter is used to fill data into the dataset after opening the Sql connection.

Now, the data table returned can be used to fill the Case Details in the Case Management web page with associated tab details. Following is the code to fill the current tab with table data.

sqlbuilder sb = new sqlbuilder();
DataTable dt1 = sb.getcasedetails(txtCasenum.Text);
if (dt1.Rows.Count == 0)
{
    errlbl.Text = "Matching case not found";
}
else
{
    txtCaseNumber.Text = dt1.Rows[0][0].ToString();
    txtCaseName.Text = dt1.Rows[0][1].ToString();
    txtClientRefNum.Text = dt1.Rows[0][2].ToString();
    txtAttorney.Text = dt1.Rows[0][3].ToString();
    txtCourt.Text = dt1.Rows[0][4].ToString();
    OpenedDate.Text = dt1.Rows[0][5].ToString();
    ClosedDate.Text = dt1.Rows[0][6].ToString();
    txtCourtNum.Text = dt1.Rows[0][7].ToString();
    txtInsuranceClaim.Text = dt1.Rows[0][8].ToString();
    txtSOL.Text = dt1.Rows[0][9].ToString();
}
In the above code, separate stored procedures are called for filling the case related details into appropriate tabs on the screen. For example, the retrieveCaseEvents() procedure is called extracting case related events (that were created in Scheduling Calendar and associated with this case number).

// Gridview for getting event details tab
DataTable dt = sb.retrieveCaseEvents(txtCasenum.Text);
GridView1.DataSource = dt;
GridView1.DataBind();

The retrieveCaseNotes() procedure is used retrieve all case notes associated with the given case number.

// Gridview for getting case notes tab
// DataTable dt2 = sb.retrieveCaseNotes(txtCasenum.Text);
// GridView3.DataSource = dt2;
// GridView3.DataBind();

Similarly, a grid view is created for loading all the documents pertaining to the given case under the “Linked Documents” tab. The following code snippet can be used to extract the full path for each file related to the case and also create a directory with the name of case number if not already exists. The following code snippet is used to load the grid view with appropriate documents and load the document names as hyperlinks for the user to click and open there itself, thus providing immediate access to files.
GridView2.DataSource = alFullName;
GridView2.DataBind();
for (int i = 0; i < GridView2.Rows.Count; i++)
{
    HyperLink hl = (HyperLink)GridView2.Rows[i].FindControl("filenameslink");
    Label fl = (Label)GridView2.Rows[i].FindControl("filenameslabel");
    hl.Text = alFileName[i].ToString();
    hl.NavigateUrl = alFullName[i].ToString();
    fl.Text = alFileName[i].ToString();
}

The following code snippet shows how to add case notes for a particular case and display all the related notes immediately in the “Case Notes” tab. First, a database connection is established to retrieve all the case notes already added and then add new notes to the database that were created by the user.

SqlConnection conn = new SqlConnection(ConfigurationManager.ConnectionStrings["msdbConnectionString"].ConnectionString);
if (e.CommandName.Equals("New"))
{
    LinkButton btnNew = e.CommandSource as LinkButton;
    GridViewRow row = btnNew.NamingContainer as GridViewRow;
    if (row == null)
    {
        return;
    }
    TextBox txtNotes = row.FindControl("txtNotesInsert") as TextBox;
    if (txtNotes.Text==string.Empty || txtNotes.Text==" " || txtNotes.Text == ")
    {
        throw new Exception("Notes cannot be blank");
    }
    SqlCommand cmd = new SqlCommand("INSERT INTO Casenotes (Casenum, Notes, CreatedDate) VALUES (@Casenum, @Notes)", conn);
    cmd.Parameters.AddWithValue("Casenum", txtCasenum.Text);
    cmd.Parameters.AddWithValue("Notes", txtNotes.Text);
    cmd.Parameters.AddWithValue("CreatedDate", DateTime.Today.ToShortDateString());
    conn.Open();
    if (cmd.ExecuteNonQuery() == 1) GridView3.DataBind();
Implementation Details of Task Management Feature

The code snippets in this section depict the functionality of the Task Management feature that provides user, the ability to perform the following tasks.

− Load Tasks Assigned To data grid with the tasks assigned to the user
− Load Tasks Assigned By data grid with the tasks assigned by the user
− Create a new task for the given user
− Update details of an existing task
− Delete a completed task

The following code snippet shows how to determine whether the new task is a valid task before storing into the database.

```csharp
TextBox txtTask = row.FindControl("txtTaskInsert") as TextBox;
TextBox txtComments = row.FindControl("txtCommentsInsert") as TextBox;
TextBox txtAssignedTo = row.FindControl("txtAssignedInsert") as TextBox;
{
    throw new Exception("Task cannot be blank");
}
```

The following code snippet shows how to add new task created by the current logged in user into the database.

```csharp
SqlCommand cmd = new SqlCommand("INSERT INTO Tasks (Task, Comments, [Created By], [Assigned To], [Last Edited By], [Mark Read]) VALUES (@Task, @Comments, @Created_By, @Assigned_To, @Last_Edited_By, @Mark_Read)", conn);
cmd.Parameters.AddWithValue("Task", txtTask.Text);
cmd.Parameters.AddWithValue("Comments", txtComments.Text);
cmd.Parameters.AddWithValue("Assigned_To", txtAssignedTo.Text);
cmd.Parameters.AddWithValue("Mark_Read", "N");
```
conn.Open();
if (cmd.ExecuteNonQuery() == 1)
{
    GridView2.DataBind();
}

In the above code, the GridView2 is the data grid that displays the tasks that are currently assigned by the logged in user to other users in the system.

**Implementation Details of Contacts Management Feature**

The code snippets in this section depict the functionality of the Contact Management feature that provides user, the ability to perform the following tasks.

- Load Contact details for the selected Contact name
- Create a new contact visible to other features
- Update details of an existing contact
- Delete a completed contact

The following code snippet shows how to retrieve details of an existing contact from the database.

```csharp
public DataTable getcontactdetails(string cn)
{
    SqlCommand cmd = new SqlCommand("sp_getcontactdetails", con);
    cmd.CommandType = CommandType.StoredProcedure;
    cmd.Parameters.Add(new SqlParameter("@contact", cn));
    SqlDataAdapter da = new SqlDataAdapter(cmd);
    con.Open();
    DataSet ds = new DataSet();
    da.Fill(ds);
    con.Close();
    return ds.Tables[0];
}
```
The following code snippet shows how to determine whether a contact exists in the system when the user tries to filter using contact name.

```csharp
protected void btnSearchcontact_Click(object sender, EventArgs e)
{
    errlbl.Text = "";
    sqlbuilder sb = new sqlbuilder();
    DataTable dt = sb.getcontactdetails(txtSearchcontact.Text);
    if (dt.Rows.Count == 0)
        errlbl.Text = "Matching contact not found";
}
```

The following code depicts how to add new contact details into the system that are being entered by the logged in user.

```csharp
public void savewnewcontactdetails(string ln, string fn, string dp, string mp, string ep,
string addr, string addr2, Int32 zip, string cty, string st, string email, string cn, string
 caddr, string caddr2, string wp, string cp, string atty)
{
    SqlCommand cmd = new SqlCommand("INSERT INTO Contacts VALUES ('" + ln + ",
    " + fn + "," + dp + "," + mp + "," + ep + "," + addr + "," + addr2 + "," + zip + "," + cty + "," + st + "," + email + "," + cn + "," + caddr + "," +
caddr2 + "," + wp + "," + cp + "," + atty + ")", con);
    con.Open();
    cmd.ExecuteNonQuery();
    con.Close();
}
```

Thus, the above code segments explain in detail all the major functionalities in the system and their implementation details to provide a clear understanding to the readers. The above code segments can be used to understand how actually a particular feature is designed and implemented in the web application.
Chapter 6

CONCLUSION

This project is developed as a real-time project for a real client. It aims at providing an efficient web-based solution that ensures better communication and quick access to case related information. It contributes to improve attorney & staff productivity by becoming more efficient in client and case management and handles a greater caseload without increasing staff. It provides the capability to better communicate with clients, provide immediate information on case status to clients, and to have a quicker turn around on work product.

This project is efficient in realizing the goals of the employees of the firm by efficiently storing the events recorded by the firm’s employees, be it solicitors, secretarial or other auxiliary staff. In addition, it provides the flexibility of customization according to the requirements of the employees.

The scope of the project has been defined in such a way that it can be submitted as a Master’s project. We can also extend some functionality in the project, which will be described in detail in the following section.

**Functionality Extensions**

The following functionalities can be added onto the existing system:

1. Import data from the existing system into the new database with any dependencies identified and prevent loss of data.
2. Maintain ledger accounts payable and receivable with journal entries created for every transaction with the client.

3. Maintain and store credit card information of the clients and setup automatic payment reminders for fee payments and deadlines.

4. Auto filling of forms with frequently used data by the law firm. This should generate appropriate PDF or MS Word format documents with data filled as necessary.
APPENDIX

Sample Screens

The following screenshots capture the main functionalities of the system.

Date Calculator

User navigates to the home page and selects Date calculator.

![Figure 32: Date Calculator Screen Shot](image)

User selects the required date by clicking on the calendar icon or can directly enter a date.

User can choose to display the days, court days, weeks, months, quarters or years for the
selected date interval. System calculates and displays the total days and workdays for the
given interval.

**Judgment Interest Calculator**

User navigates to the home page and selects “Judgment Interest calculator”.

User enters the Judgment Amount, Interest Rate per Year and Date of Judgment. System
then calculates and displays Daily Interest, Number of Days, Interest Accrued and
Current Value based on the given input values.

![Judgment Interest Calculator Screen Shot](image)

Figure 33: Judgment Interest Calculator Screen Shot
Motion Date Calculator

User navigates to the home page and selects Motion Date calculator.

![Motion Date Calculator Screen Shot](image)

Figure 34: Motion Date Calculator Screen Shot

User selects the required date by clicking on the calendar icon or can directly enter a date in mm/dd/yyyy format. System calculates and displays the motion date (adds 16 court days and 5 calendar days to the given date) on the screen.

Scheduling Calendar

User navigates to the home page and selects “Scheduling Calendar”.
By default, system displays the calendar with events scheduled for Today’s date. User can view the calendar for desired date by entering in the text box. System refreshes the screen and displays calendar for the selected date.

Add a New Event
User navigates to the home page and selects “Scheduling Calendar”. User then clicks on “Add a new event” link on the screen.
System opens a new window for entering the details of new event. User clicks on “Save” button. System creates a new event and displays it on the scheduling calendar for the specified date.

**Edit/Delete an Event**

User navigates to the home page and selects “Scheduling Calendar”. User then selects “Edit/Delete an Event” option in the menu.
User selects the required event to edit and clicks on “Search Event”. User can then edit the details of the event and click on “Save”. System saves the modified event and user can view the updated scheduling calendar by clicking on “View Calendar” link.

**Re-schedule an Event**

User navigates to the home page and selects “Scheduling Calendar”. User selects “Re-schedule” option in the menu.
Figure 38: Re-Schedule Event Screen Shot

User enters the following information

− Event Time & Date to Reschedule
− New Date and Time for the Event

System saves the re-scheduled event and the user can view the updated scheduling calendar by clicking on “View Calendar”.

Add a New Rule

User navigates to the home page and selects “Scheduling Calendar”.


User clicks on “Add a new rule” link in the menu.

Figure 39: Add New Rule Screen Shot

System opens a new page for entering the details of new rule. User enters the information and clicks on “Create Events” to create a set of events that belong to this rule. System saves the new rule and its associated events and refreshes the entire page for the user to create new rules.
Delete a Rule

User navigates to the home page and selects “Scheduling Calendar”. User clicks on “Delete a rule” link on the menu.

![Delete Rule Screen Shot](image)

Figure 40: Delete Rule Screen Shot

User selects a rule and clicks on “Delete Rule” button to delete any unwanted rules.

Email Calendar

User navigates to the home page and selects “Scheduling Calendar”. User clicks on Email Calendar link.
System opens a new window for entering the email address. User enters Email Address and clicks on “Send” button. System sends the scheduling calendar for the specified date to the specified email address.

**Print Calendar**

User navigates to the home page and selects “Scheduling Calendar”. User then clicks on “Print Calendar” link on the screen.
System opens a new window with a confirmation message. User clicks on “Print” button. System opens the “default printer” options page for setting options on how to print the scheduling calendar.

**Add/Edit/Delete What**

User navigates to the home page and selects “Scheduling Calendar”. User clicks on “Add New Event” link on the menu. User clicks on “… ” button against the “What” dropdown.
System opens a window where the user can add new values or edit/delete any existing “What” values. User is able to add, edit or delete “What” values in the same screen and refresh the list in “What” dropdown after closing the pop-up window.

**Add/Edit/Delete Where**

User navigates to the home page and selects “Scheduling Calendar”. User clicks on “Add New Event” link on the menu. User clicks on “…” button against the “Where” dropdown.
System opens a window where the user can add new values or edit/delete any existing “Where” values. User is able to add, edit or delete “Where” values in the same screen and refresh the list in “Where” dropdown after closing the pop-up window.

Add Docket Item

User navigates to the home page and selects “Scheduling Calendar”. User then clicks on “Add Docket Item” link on the menu.
Figure 45: Add Docket Item Screen Shot

User selects the “Rule” to which the Docket Item has to be applied. User enters the required information and clicks on “Save” button. System calculates the date for each particular event in the rule based on the interval formula entered while creating the rule. System creates multiple events associated to the rule at once and displays it on the scheduling calendar for the specified date.

**Add/Edit/Delete Tasks to Assign**

User navigates to the home page and selects “Tasks Manager”.
User can create new tasks by clicking on “Add New” button. System creates the new task and updates the two fields – “Created By” and “Last Edited By” fields with current logged in user; and “Last Edited Date” with today’s date. System displays the new task immediately in the same grid for the user to view.

View/Edit Tasks Assigned to user
User navigates to the home page and selects “Tasks Manager”.

Figure 46: Add/Edit/Delete Tasks to Assign Screen Shot
User clicks on “Edit” link to edit a task. System saves the task and updates the two fields “Last Edited By” with current logged in user, and “Last Edited Date” with today’s date. System displays the modified task immediately in the same grid for the user to view.

**Add new Case**

User navigates to the home page and selects “Case Management”. User then clicks on “Add new Case” link on the screen.
User enters the required information and clicks on “Save” button. System creates a new case with automatically generated case number and displays it on the screen for the user to view.

**Edit/Delete a Case**

User navigates to the home page and selects “Case Management”. User enters the case number to perform edit or delete operation. User clicks on “Retrieve Case”.

Figure 48: Add New Case Screen Shot
User can edit any of the case information displayed on the screen. User clicks on “Save” button. System saves the case details and displays on the screen for the user to view.

**View Case Details**

User navigates to the home page and selects “Case Management”. User enters the case num to retrieve the details and clicks on “Retrieve Case”.

Figure 49: Edit/Delete Case Screen Shot
System displays the four tabs with case information – Linked events, Linked contacts, Linked documents and Case Notes.

**Add Case Notes**

User navigates to the home page and selects “Go to Case Management”. User enters the case number to retrieve the details. User clicks on “Retrieve Case” button. User can add case notes in the “Case Notes” tab.
Figure 51: Add Case Notes Screen Shot

User enters the text information relevant to the case. User clicks on “Save” button. System displays the newly added notes and the corresponding “Added Date” with the current system date.
REFERENCES


3. Introducing .NET Framework; Retrieved on 19th April 2010, from Developer Fusion Ltd; Website: http://www.developerfusion.com/samplechapter/1678/introducing-net/2/

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

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