**Meetings Application Documentation**

Group D:

Brent Silvernagel; Deborah Snyder; Joshua Thomas; Clayton Townley

University of Maryland University College

Revision: 3.3

Date: 04/29/2018

Class #: CMSC 495 7981

**Revision History**

|  |  |  |  |
| --- | --- | --- | --- |
| **Revision** | **Date** | **Author** | **Description** |
| 1.0 | 03/30/2018 | Brent Silvernagel | Document Creation |
| 2.0 | 04/06/2018 | Brent Silvernagel | Merged Project Plan, Test Plan, and User Guide |
| 3.0 | 4/8/2018 | Clayton, Deb Snyder | Merged Project Design |
| 3.1 | 4/8/2018 | Brent Silvernagel | Updated UML diagrams added deployment section |
| 3.2 | 4/29/2018 | Brent Silvernagel, Deb Synder | Integrated phase reports into main documentation body. Created phase 3 report. |
| 3.3 | 4/29/2018 | Brent Silvernagel | Updated UML diagrams and documentation to reflect shift in application from Web based to console based. |

Table of Contents

[1. Project Plan 5](#_Toc512773567)

[Purpose 5](#_Toc512773568)

[Project Description 5](#_Toc512773569)

[System Specification 5](#_Toc512773570)

[System Requirements 5](#_Toc512773571)

[Software Management 6](#_Toc512773572)

[Work Breakdown Structure 6](#_Toc512773573)

[Project Schedule 7](#_Toc512773574)

[2. Test Plan 8](#_Toc512773575)

[Purpose 8](#_Toc512773576)

[Requirement Matrix 8](#_Toc512773577)

[Test Steps 9](#_Toc512773578)

[3. User Guide for the Meeting Time Calculator Web Application 13](#_Toc512773579)

[Introduction 13](#_Toc512773580)

[System Specification 13](#_Toc512773581)

[System Requirements 13](#_Toc512773582)

[Accessing the Web Application 14](#_Toc512773583)

[Inputting Data: 14](#_Toc512773584)

[Running the Web Application: 15](#_Toc512773585)

[Results: 16](#_Toc512773586)

[4. Test Plan 16](#_Toc512773587)

[Purpose 16](#_Toc512773588)

[Requirement Matrix 16](#_Toc512773589)

[Test Steps 17](#_Toc512773590)

[5. Project Design 21](#_Toc512773591)

[Overall Approach: 21](#_Toc512773592)

[UML Class Diagram: 22](#_Toc512773593)

[Class Breakdown: 22](#_Toc512773594)

[Data Layer Breakdown: 24](#_Toc512773595)

[Deployment 25](#_Toc512773596)

[6. Phase Reports 25](#_Toc512773597)

[Phase 1 25](#_Toc512773598)

[Phase 2 26](#_Toc512773599)

[Phase 3 27](#_Toc512773600)

# Project Plan

## Purpose

The purpose of this project is to provide visibility to the time and cost of meetings. It should be noted that the value meetings bring to a company varies from company to company and meeting to meeting. The purpose is not to discern the value of meetings, but to provide visibility of their opportunity cost. Additionally, it can give insight into the projected costs of recurring meetings throughout the life of a program to provide more accurate program costs estimates.

## Project Description

The application will take a given input of meeting duration, attendees, and frequency and output an approximate cost in time/money. The input will be a user provided input, but the project will also explore the capability of a calendar API if time permits. The tool will be capable of calculating the costs of meetings over any desired time scale whether the user wishes it to view costs by each meeting, over a week, a month, a year, or the life of the program. Additionally, the cost metrics can be exported in a comma-separated value (.csv) file to be read by Excel for further analysis.

The application shall consist of three distinct functional areas: a front end for user input, a business logic layer to crunch numbers, and a SQL database to store and retrieve meeting records. The GUI shall be provided for the end user and can be run on any Windows based machine by running the installer included with the application. The front end of the application will take all necessary user input and use that information to create a SQL database element. The business layer will use all input in that given database element to calculate cost metrics over the user selected time scale. The costs metrics will be displayed on the front end, with the option of exporting the data.

## System Specification

Software: Modern Windows OS.

Web server running SQL.

## System Requirements

**Req – 1** The application shall allow the user to enter the following meeting information:

1. Meeting Duration
2. Frequency
3. Attendees
4. Attendee Pay Grade
5. Time Scale (meeting, day, week, month, year, program life)

**Req – 2** The application shall create a unique database element for each meeting calculation.

**Req – 3** The application shall calculate the meeting cost based on user input data.

**Req – 4** The application shall provide a quick reference lookup table for contractor rates for cost estimation.

**Req – 5** The application shall provide a quick reference lookup table for government rates for cost estimation.

**Req – 6** The application shall allow the user to delete meeting records.

**Req – 7** The application shall allow the user to load saved meeting records.

**Req – 8** The application shall run on modern web browsers.

**Req – 9** The application shall allow the user to browse and load all saved meeting records.

**Req – 10** The application shall provide a report generation utility to export meeting records as comma-separated value (.csv) files.

**Req – 11** The application shall tag each record with a timestamp.

**Req – 12** The application shall notify user if any required inputs are not supplied for cost calculation.

**Req – 13** The application shall allow user to create a unique name for saving meeting records.

**Req – 14** The application shall prevent the user from using invalid characters when saving meeting records.

**Req – 15** The application shall prevent user from entering invalid values for all cost estimation input fields, such as negative costs.

**Req – 16** The application shall allow each user to create a unique user account for accessing saved meeting records.

## Software Management

Git will be used for software management. Git is an open source distributed control system designed to handle software version control. The project’s repository is hosted on GitHub: <https://github.com/BSilvernagel/Meeting-Tracker-Home>.

## Work Breakdown Structure

|  |  |
| --- | --- |
| **Name** | **Area of Responsibility** |
| Brent Silvernagel | Test, Integration, and User Manual Development |
| Joshua Thomas | Database Management Development |
| Clayton Townley | Front End Development |
| Deborah Snyder | Business Logic Development |

## Project Schedule

|  |  |  |  |
| --- | --- | --- | --- |
| **Task** | **Start Date** | **Due Date** | **Assignee(s)** |
| **Project Plan:** | | | |
| Draft Project Plan | 3/12/2018 | 3/18/2018 | Brent |
| Derive Requirements | 3/12/2018 | 3/18/2018 | Brent |
| Finalized Project Plan | 3/18/2018 | 3/25/2018 | Clayton |
| **Test Plan:** | | | |
| Test Plan Draft | 3/18/2018 | 3/25/2018 | Brent |
| Develop User Guide | 3/25/2018 | 4/1/2018 | Brent |
| Finalized Test Plan | 3/25/2018 | 4/1/2018 | Brent |
| **Project Design:** | | | |
| Preliminary Project Design | 3/12/2018 | 3/25/2018 | Clayton |
| Peer Reviews | 3/25/2018 | 4/1/2018 | All |
| Finalized Project Design | 4/1/2018 | 4/8/2018 | All |
| **Sprint 1 Development: Business Layer** | | | |
| Code Business logic component | 4/8/2018 | 4/13/2018 | Brent, Deborah |
| Test software component | 4/13/2018 | 4/15/2018 | Brent |
| **Sprint 2 Development: Database Layer** | | | |
| Code SQL component | 4/8/2018 | 4/20/2018 | Joshua |
| Test SQL database | 4/20/2018 | 4/22/2018 | Brent |
| Integrate Business Layer | 4/20/2018 | 4/22/2018 | Joshua, Brent |
| **Sprint 3 Development: Web App Front End** | | | |
| Code Web App | 4/8/2018 | 4/24/2018 | Clayton |
| Test Web App | 4/24/2018 | 4/26/2018 | Brent |
| Integration | 4/26/2018 | 4/29/2018 | Clayton, Brent |
| **Final Delivery:** |  |  |  |
| Integration Testing | 4/29/2018 | 5/5/2018 | Brent |
| Deliver Source Code | 5/5/2018 | 5/6/2018 | Clayton |

Table 1: Project Schedule

# Test Plan

## Purpose

The purpose of this document is to validate the functionality of the Meetings Cost application and verify the application meets all requirements outlined in the project plan.

## Requirement Matrix

|  |  |
| --- | --- |
| **ID** # | **Description** |
| Req – 1 | The application shall allow the user to enter the following meeting information:   1. Meeting Duration 2. Frequency (Weekly) 3. Number of Attendees 4. Attendee Pay Grade 5. Time Scale (per meeting, day, week, month, year) |
| Req – 2 | The application shall create a unique database element for each meeting calculation. |
| Req – 3 | The application shall calculate the meeting cost based on user input data. |
| Req – 4 | The application shall provide a quick reference lookup table for contractor rates for cost estimation. |
| Req – 5 | The application shall provide a quick reference lookup table for government rates for cost estimation. |
| Req – 6 | The application shall allow the user to delete meeting records. |
| Req – 7 | The application shall allow the user to load saved meeting records. |
| Req – 8 | The application shall run on modern web browsers. |
| Req – 9 | The application shall allow the user to browse and load all saved meeting records. |
| Req – 10 | The application shall provide a report generation utility to export meeting records as comma-separated value (.csv) files. |
| Req – 11 | The application shall tag each record with a timestamp. |
| Req – 12 | The application shall notify user if any required inputs are not supplied for cost calculation. |
| Req – 13 | The application shall allow user to create a unique name for saving meeting records. |
| Req – 14 | The application shall prevent the user from using invalid characters when saving meeting records. |
| Req – 15 | The application shall prevent user from entering invalid values for all cost estimation input fields, such as negative costs. |
| Req – 16 | The application shall allow each user to create a unique user account for accessing saved meeting records. |

## Test Steps

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Step #** | **Req. #** | **Test Description** | **Expected Result** | **Actual Result** | **Pass/Fail** |
| **User Registration** | | | | | |
|  | Req - 8 | Open Modern Web Browser and type in URL; Presses Enter | URL takes User to Welcome Page |  |  |
|  | Req - 16 | User chooses Login | Login Page appears |  |  |
|  | Req - 16 | User chooses Registration | Register Page appears |  |  |
|  | Req - 16 | Populate new user fields and select create account. | User account created, the user should be able to login with the user name and password provided. |  |  |
| **Registration Error Handling** | | | | | |
|  |  | Open user registration. | Registration page displays. |  |  |
|  | Req - 16 | Leave some of registration fields blank;  Select Register | Error Page Notification – “Required field is blank” |  |  |
|  | Req - 16 | User types invalid email address with all fields filled; Clicks Register | Error Page - invalid email address box |  |  |
| **Meeting Information** | | | | | |
|  | Req - 1 | Verify the Meeting Time Calculator has the following fields:   * Meeting Duration * Meetings per Week * Number of Attendees * Salary Range * Time Scale | All fields are available for data entry. |  |  |
|  | Req - 3 | Enter the following data:  Duration: 60 (min)  Times per week: 2  Attendees: 3  Salary Range: 50,000  Scale: 1 week  Select Calculate button. | Verify calculated cost is 144.18$ |  |  |
|  | Req – 4 | Select the contractor rate link. | Verify the link directs user to contractor pay scale link. |  |  |
|  | Req – 5 | Select the government rate link. | Verify the link directs user to government pay scale link. |  |  |
| **Meeting Information Error Handling** | | | | | |
|  | Req - 12 | Enter the following data:  Duration: blank  Times per week: 2  Attendees: 3  Salary Range: 50,000  Scale: 1 week  Select Calculate button. | Error dialog displays with notification about missing data fields. |  |  |
|  | Req - 12 | Enter the following data:  Duration: 30 min  Times per week: blank  Attendees: 3  Salary Range: 50,000  Scale: 1 week  Select Calculate button. | Error dialog displays with notification about missing data fields. |  |  |
|  | Req – 12 | Enter the following data:  Duration: 30 min  Times per week: 3  Attendees: blank  Salary Range: 50,000  Scale: 1 week  Select Calculate button. | Error dialog displays with notification about missing data fields. |  |  |
|  | Req – 12 | Enter the following data:  Duration: 30 min  Times per week: 3  Attendees: 3  Salary Range: blank  Scale: 1 week  Select Calculate button. | Error dialog displays with notification about missing data fields. |  |  |
|  | Req – 12 | Enter the following data:  Duration: 30 min  Times per week: 3  Attendees: 3  Salary Range: 50,000  Scale: blank  Select Calculate button. | Error dialog displays with notification about missing data fields. |  |  |
|  | Req - 15 | Enter the following data:  Duration: 0 min  Times per week: 3  Attendees: 3  Salary Range: 50,000  Scale: 1 week  Select Calculate button. | Error dialog displays with notification stating duration must be greater than 0. |  |  |
|  | Req - 15 | Enter the following data:  Duration: 30 min  Times per week:  Attendees: -5  Salary Range: 50,000  Scale: 1 week  Select Calculate button. | Error dialog displays with notification stating negative values cannot be accepted. |  |  |
| **Database Operations** | | | | | |
|  | Req - 13 | Select the Save Meeting Calculation option.  Create filename.  Select Save. | Save As GUI is displayed to user.  Filename is accepted.  Record Saved  Save As GUI closes. |  |  |
|  | Req – 13 | Select the Save Meeting Calculation option.  Attempt to save as the same name from step 19. | Save As GUI is displayed to user.  Warning is displayed about overwriting existing file. |  |  |
|  | Req - 14 | Select the Save Meeting Calculation option.  Enter the following into name field:   * Test.!   Select Save. | Save As GUI is displayed to user.  Error dialog displayed stating “Invalid Character”. |  |  |
|  | Req - 9 | Select the Load Meeting button. | File explorer opens with history of meeting records. |  |  |
|  | Req – 7 | Select a record.  Select Load. | Verify all meeting fields are populated with data from selected record. |  |  |
|  | Req - 6 | Select the Load Meeting button.  Select a meeting Record.  Select Delete. | File explorer opens with history of meeting records.  Record is deleted. |  |  |
|  | Req – 11 | Select the Load Meeting button. | Verify all saved meetings have a timestamp appended on the filename. |  |  |
| **Export Controls** | | | | | |
|  | Req – 10 | Select the Export option. | Export GUI displays |  |  |
|  | Req – 10 | In the Export GUI:   * Create filename * Select Local file location   Select Export | A .csv file with the selected name is saved in the designated directory. |  |  |
|  | Req – 10 | Navigate to the directory the .csv file is located in.  Open .csv in MS Excel. | All data is captured from Meeting application in Excel spreadsheet. |  |  |

# User Guide for the Meeting Time Calculator Web Application

## Introduction

The purpose of the Meeting Time Calculator application is to provide visibility to the time and cost of meetings. It should be noted that the value meetings bring to a company varies from company to company and meeting to meeting. The purpose is not to discern the value of meetings, but to provide visibility of their opportunity cost. Additionally, it can give insight into the projected costs of recurring meetings throughout the life of a program to provide more accurate program costs estimates.

This application allows the user to input details about reoccurring meetings at their company or establishment and get the estimated cost of the meetings over a period of time (one meeting, one week, one month or one year). Details that the user must input are the estimated duration of the meeting, how many times per week are the meetings occurring, how many people attend the meeting and lastly what the typical salary is for the attendees of the meetings.

## System Specification

Software: Windows OS

Web server running SQL.

## System Requirements

**1:** User should have the following meeting information to enter into the application:

1. Meeting Duration
2. Frequency
3. Attendees
4. Attendee Pay Grade
5. Time Scale (meeting, day, week, month, year, program life)

## Installing the Application

The delivered product includes a installer MeetingAppInstall.exe. To start the installer double click the MeetingAppInstall.exe file. The installer will install all necessary libraries to run the application. The default location will be C:\Users\<username>\AppData\Local\. Navigate to this directory verify the application has installed. The application should now be available from the Start menu to launch.

## Accessing the Application

1. Navigate to the start menu and double click the MeetingApp application.

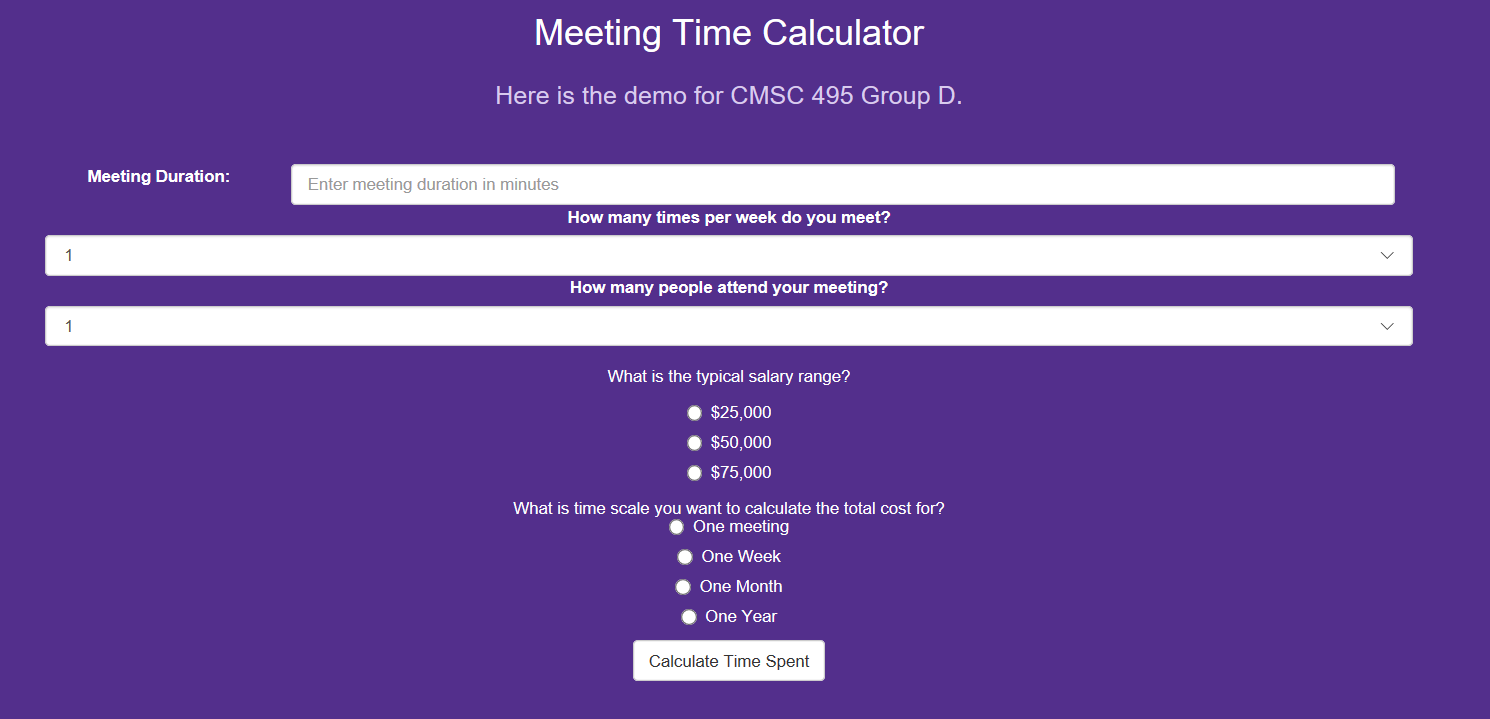


Figure 1: Snapshot of the application when launched.

## Inputting Data:

There are five questions that must be answered before calculating the estimate.

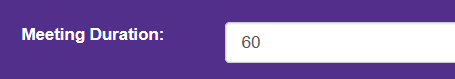
1. Start with the first text box labeled “Meeting Duration”. As stated in the text box, enter an integer of the duration of the meeting in minutes. Proper examples of this would be “30” or “60” not thirty or sixty. Do not enter the word minutes or “min” after the integer. 

Figure 2: Example of Meeting duration input.

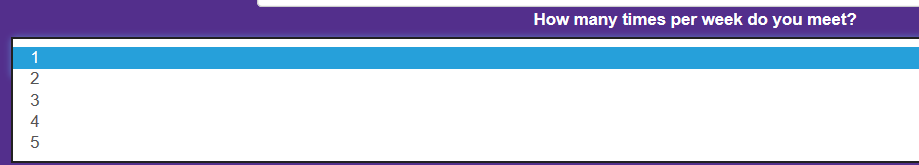
1. Next, refer to the text box with the label “How many times per week do you meet?”. Click on this box and a dropdown menu will appear. Select the amount of times that the meeting is held from the drop down. 

Figure 3: Example of times per week dropdown menu.

1. Move on to the text box with the label “How many people attend your meeting”. Click on the text box and a dropdown menu will appear. Select the amount of individuals who attend the meeting from the dropdown. 

Figure 4: Example of amount of attendees dropdown menu.

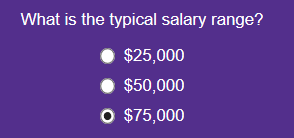
1. After selecting from the dropdown text boxes, refer to the fourth question “What is the typical salary range?”. Click the radio button that corresponds to the salary range of attendees. It should look like the following: 

Figure 5: Example of typical salary selection.

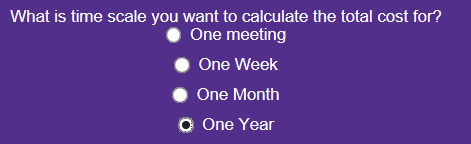
1. Refer to the fifth and final question “What is the time scale that you want to calculate the total cost for?”. Click the radio button that corresponds to the time scale that best suits your calculations whether it be one day, one week, one month or one year. This will break down the cost of meetings over that span of time. 

Figure 6: Example of time scale selection.

## Running the Application:

1. To run the application, user must select the button at the bottom of the screen that is labeled “Calculate Time Spent”.

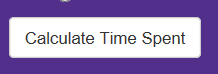


Figure 7: Snapshot of button that runs the web application calculations.

## Results:

1. Estimated dollar amount will appear below the Calculation Button.
2. If unsatisfied with the cost, user can adjust the amounts and click the Calculation button again.

# Test Plan

## Purpose

The purpose of this document is to validate the functionality of the Meetings Cost application and verify the application meets all requirements outlined in the project plan.

## Requirement Matrix

|  |  |
| --- | --- |
| **ID** # | **Description** |
| Req – 1 | The application shall allow the user to enter the following meeting information:   1. Meeting Duration 2. Frequency (Weekly) 3. Number of Attendees 4. Attendee Pay Grade 5. Time Scale (per meeting, day, week, month, year) |
| Req – 2 | The application shall create a unique database element for each meeting calculation. |
| Req – 3 | The application shall calculate the meeting cost based on user input data. |
| Req – 4 | The application shall provide a quick reference lookup table for contractor rates for cost estimation. |
| Req – 5 | The application shall provide a quick reference lookup table for government rates for cost estimation. |
| Req – 6 | The application shall allow the user to delete meeting records. |
| Req – 7 | The application shall allow the user to load saved meeting records. |
| Req – 8 | The application shall run on modern web browsers. |
| Req – 9 | The application shall allow the user to browse and load all saved meeting records. |
| Req – 10 | The application shall provide a report generation utility to export meeting records as comma-separated value (.csv) files. |
| Req – 11 | The application shall tag each record with a timestamp. |
| Req – 12 | The application shall notify user if any required inputs are not supplied for cost calculation. |
| Req – 13 | The application shall allow user to create a unique name for saving meeting records. |
| Req – 14 | The application shall prevent the user from using invalid characters when saving meeting records. |
| Req – 15 | The application shall prevent user from entering invalid values for all cost estimation input fields, such as negative costs. |
| Req – 16 | The application shall allow each user to create a unique user account for accessing saved meeting records. |

## Test Steps

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Step #** | **Req. #** | **Test Description** | **Expected Result** | **Actual Result** | **Pass/Fail** |
| **User Registration** | | | | | |
|  | Req - 8 | Open Modern Web Browser and type in URL; Presses Enter | URL takes User to Welcome Page |  |  |
|  | Req - 16 | User chooses Login | Login Page appears |  |  |
|  | Req - 16 | User chooses Registration | Register Page appears |  |  |
|  | Req - 16 | Populate new user fields and select create account. | User account created, the user should be able to login with the user name and password provided. |  |  |
| **Registration Error Handling** | | | | | |
|  |  | Open user registration. | Registration page displays. |  |  |
|  | Req - 16 | Leave some of registration fields blank;  Select Register | Error Page Notification – “Required field is blank” |  |  |
|  | Req - 16 | User types invalid email address with all fields filled; Clicks Register | Error Page - invalid email address box |  |  |
| **Meeting Information** | | | | | |
|  | Req - 1 | Verify the Meeting Time Calculator has the following fields:   * Meeting Duration * Meetings per Week * Number of Attendees * Salary Range * Time Scale | All fields are available for data entry. |  |  |
|  | Req - 3 | Enter the following data:  Duration: 60 (min)  Times per week: 2  Attendees: 3  Salary Range: 50,000  Scale: 1 week  Select Calculate button. | Verify calculated cost is 144.18$ |  |  |
|  | Req – 4 | Select the contractor rate link. | Verify the link directs user to contractor pay scale link. |  |  |
|  | Req – 5 | Select the government rate link. | Verify the link directs user to government pay scale link. |  |  |
| **Meeting Information Error Handling** | | | | | |
|  | Req - 12 | Enter the following data:  Duration: blank  Times per week: 2  Attendees: 3  Salary Range: 50,000  Scale: 1 week  Select Calculate button. | Error dialog displays with notification about missing data fields. |  |  |
|  | Req - 12 | Enter the following data:  Duration: 30 min  Times per week: blank  Attendees: 3  Salary Range: 50,000  Scale: 1 week  Select Calculate button. | Error dialog displays with notification about missing data fields. |  |  |
|  | Req – 12 | Enter the following data:  Duration: 30 min  Times per week: 3  Attendees: blank  Salary Range: 50,000  Scale: 1 week  Select Calculate button. | Error dialog displays with notification about missing data fields. |  |  |
|  | Req – 12 | Enter the following data:  Duration: 30 min  Times per week: 3  Attendees: 3  Salary Range: blank  Scale: 1 week  Select Calculate button. | Error dialog displays with notification about missing data fields. |  |  |
|  | Req – 12 | Enter the following data:  Duration: 30 min  Times per week: 3  Attendees: 3  Salary Range: 50,000  Scale: blank  Select Calculate button. | Error dialog displays with notification about missing data fields. |  |  |
|  | Req - 15 | Enter the following data:  Duration: 0 min  Times per week: 3  Attendees: 3  Salary Range: 50,000  Scale: 1 week  Select Calculate button. | Error dialog displays with notification stating duration must be greater than 0. |  |  |
|  | Req - 15 | Enter the following data:  Duration: 30 min  Times per week:  Attendees: -5  Salary Range: 50,000  Scale: 1 week  Select Calculate button. | Error dialog displays with notification stating negative values cannot be accepted. |  |  |
| **Database Operations** | | | | | |
|  | Req - 13 | Select the Save Meeting Calculation option.  Create filename.  Select Save. | Save As GUI is displayed to user.  Filename is accepted.  Record Saved  Save As GUI closes. |  |  |
|  | Req – 13 | Select the Save Meeting Calculation option.  Attempt to save as the same name from step 19. | Save As GUI is displayed to user.  Warning is displayed about overwriting existing file. |  |  |
|  | Req - 14 | Select the Save Meeting Calculation option.  Enter the following into name field:   * Test.!   Select Save. | Save As GUI is displayed to user.  Error dialog displayed stating “Invalid Character”. |  |  |
|  | Req - 9 | Select the Load Meeting button. | File explorer opens with history of meeting records. |  |  |
|  | Req – 7 | Select a record.  Select Load. | Verify all meeting fields are populated with data from selected record. |  |  |
|  | Req - 6 | Select the Load Meeting button.  Select a meeting Record.  Select Delete. | File explorer opens with history of meeting records.  Record is deleted. |  |  |
|  | Req – 11 | Select the Load Meeting button. | Verify all saved meetings have a timestamp appended on the filename. |  |  |
| **Export Controls** | | | | | |
|  | Req – 10 | Select the Export option. | Export GUI displays |  |  |
|  | Req – 10 | In the Export GUI:   * Create filename * Select Local file location   Select Export | A .csv file with the selected name is saved in the designated directory. |  |  |
|  | Req – 10 | Navigate to the directory the .csv file is located in.  Open .csv in MS Excel. | All data is captured from Meeting application in Excel spreadsheet. |  |  |

# Project Design

## Overall Approach:

The Meeting Cost application is meant to be a very user-friendly application, so we tried to apply a very simple and straight forward approach design wise. Below is a visual representation of the data, business and presentational layers of the design. This design is a living document that will be added to as the source code of the application I written.

## UML Class Diagram:



Figure 1: Presentation Layer

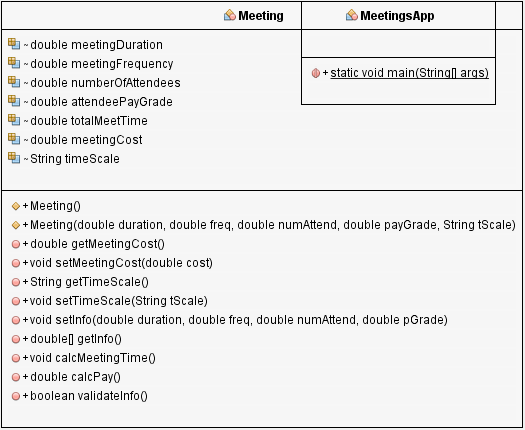
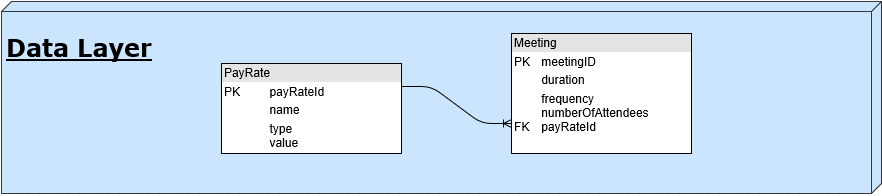


Figure 2: Business Layer



## Class Breakdown:

The following list describes the fields and methods utilized in the design of the Meeting Cost web application.

* Class : Meeting

This class is the main class where the calculations actually take place.

* + Fields:
    - meetingDuration: double
    - meetingFrequency: double
    - numberOfAttendees: double
    - attendeePayGrade: double
    - timescale: String
    - meetingCost: double
  + Methods:
    - getMeetingCost(): Get method to retrieve the set meetings cost value.
    - setMeetingCost(cost): Setter method to store meeting cost in the Meeting class.
    - calculatePay(timeScale): This method will calculate the pay based on the time scale. It will use the getInfo() method to retrieve all the other data necessary for the calculation.
    - validateInfo(): will verify all the data fields are populated with valid values prior to performing the meeting calculation.
    - exportMeetingData(): this method will collect all the meeting data and format the data so the information can be saved as a .csv.
    - setInfo(duration, freq, numAttend, pay): sets all the respective class variables for all the user inputted meeting data.
    - getInfo(): Returns a array list of the meeting information for duration, frequency, numberOfAttendees, and paygrade.
* Class: DatabaseController

This is the class that controls the database that keeps track of the meetings the user has calculated in case they have calculated multiple meetings with different parameters. It is used to add data to the database and retrieve data from the database.

* + Fields:
    - Meeting: int – This integer is to keep track of different meetings
    - tableName: String – This string names the table so the user can tell the tables apart from within the different calculated meetings.
    - user: String – This string keeps track of the user that is inputting the parameters.
  + Methods:
    - addMeeting(meeting, user): this method is used to pass the meeting class data to the database.
    - removeMeeting(meeting, user): this method is used to remove the meeting record based on a meeting and user name lookup.
* Class: UserInfo

This class is used to collect and store the account information for creating a user account. This class is intended to be a container class to leverage off built in libraries for account creation.

* + Fields:
    - username: String to store the username created by the end user.
    - email: String to store the users email address for account verification.
    - password: String to store the password associated with the account.
  + Methods:
    - createAccount(username, email): This method collects the account information to be passed to the account create library.
    - validateInfo(username, email): This method verifies the user entered a valid user name and password for account creation.
* Class: GUI

This is the class that creates the GUI interface that the user will utilize when estimating the cost of a meeting (Presentation/Application layer).

* + Fields:
    - meetingDuration: textfield integer – This will be editable and allow user to input the duration of the meeting.
    - meetingFrequency: ComboBox - This is a comboBox drop menu that has different options of meeting frequencies the user can select from. Example: once a day, twice a week, 3 times or 4.
    - numberOfAttendees: ComboBox - This is a comboBox drop menu that has the number of attendees the user can select from.
    - attendeePayGrade: ComboBox – This is a comboBox drop menu that has different pay grades for the attendees the user can select from.
    - timescale: radioButton – This will be a group of selectable radioButtons of different scales of time for example 1 year, 1 week, 1 day.
    - meetingCost: textfield double – This will display the output and will not be editable.
    - submitMeetingCost: button – This is the button to submit form.
    - exportData: button – This button will call the export method to output all the users information in a .csv file.
  + Methods:
    - GUI(): This method is where the JFrame will be created and the parts of the GUI will be added.
    - getMeetingCost(): This method calls on the getMeetingCost method from the Meeting class that calculated the output. Method outputs the meetingCost (double).
    - actionPerformed(): Action method for submit button

## Data Layer Breakdown:

* Tables:
  + PayRate
    - payRateId (primary key, autonumbered int)// ID for this table
    - name(varchar) //Name of the pay rate
    - type(varchar) //This is the type of payrate; for example, government, military, \*\*See Req 5
    - value(smallint) //The hourly rate for the given payrate/type
    - Sample row: {9,Engineer,Government,40}
  + Meeting
    - meetingId(primary key, autonumbered int) //ID for this table
    - duration(tinyInt) //how many minutes for each meeting
    - frequency(tinyInt) //number of meetings/week
    - payRateId(Foreign key to PayRate table) //lookup payrate information in the payrate table; prevents having to store the payrate information multiple times.
    - Sample row: {10, 30, 5, 9} //This example meeting with ID 10, 30 minutes long, 5 days a week, and has payRate 9 attendees (Government Engineers who make $40/hour).

## Deployment

The meetings web application will be deployed using Heroku. Heroku is a cloud platform as a service that allows developers to leverage Heroku’s existing web infrastructure. Due to the short time frame of the development schedule Heroku’s hosting service is necessary to allow the team to focus on developing the application. Additionally, Heroku easily integrates with our GitHub repository. We can easily deploy using Git to instantly view our changes live.

Update: the application will no longer be web based therefore a installer will be built to deploy the application on any Windows machine that is 64 bit.

# Phase Reports

## Phase 1

**Project Roles:**

|  |  |  |
| --- | --- | --- |
| **Name** | **Area of Responsibility** | **This Week Accomplishments** |
| Brent Silvernagel | Test, Integration, and User Manual Development | Developed Business Logic Code and Hard Coded Test |
| Joshua Thomas | Database Management Development | Did not Contribute |
| Clayton Townley | Web Application Front End Development | Added Heroku files to GitHub for Web Application |
| Deborah Snyder | Business Logic Development | Created Phase 1 Report and began developing code for Error Reporting |

**Planned Milestones:**

Develop code for Business Logic components and a hard-coded test to ensure business logic is working at expected. This business logic includes the calculations behind the web application that takes in user inputted information and outputs the cost of the meeting described.

**Schedule:**

Team D is currently on schedule and will now be focusing on incorporating the Heroku Front End work and the SQL Database development. The Business Logic portion will be focusing on developing better error reporting.

**Problems and Lessons Learned**

**Problem – 1** Learning how to best integrate SQL, Heroku and Java Business Logic into one seamless web application:

1. Some are not used to using SQL or Heroku, which makes it difficult to picture how everything goes together or how to best create code that will ensure easy integration.

**Problem – 2** Fluid team communication and collaboration.

1. As an online class and due to time differences and schedules, it is a bit difficult getting the team together at the same time to discuss or brainstorm.
2. Utilizing GitHub and Slack has helped improve this, but we still need to get together more often.

**Reevaluation**

Since the beginning, Team D has reevaluated the features provided to the user. The new feature is to create a SQL database that allows the user to save the parameters inputted. This allows the user to easily compare the cost between 2 different inputs for one meeting or multiple meetings.

**Changes to Previous Documentation:**

The biggest change in previous documentations was to integrate everything into one document like the project plan, user guide and test plan. Also, we have created a more in-depth schedule for the project and planned out project roles. Future changes will be to the User Guide as the Front End is developed to better reflect what the user will be seeing.

## Phase 2

**Project Roles:**

|  |  |  |
| --- | --- | --- |
| **Name** | **Area of Responsibility** | **This Week Accomplishments** |
| Brent Silvernagel | Test, Integration, and User Manual Development | Kept team on track and distributed tasks |
| Joshua Thomas | Database Management Development | Did not Contribute |
| Clayton Townley | Web Application Front End Development | Familiarized self with server side programming and updated Heroku files in GitHub for Web Application |
| Deborah Snyder | Report Development and Business Logic | Created Phase 2 Report |

**Planned Milestones:**

Develop source code for Heroku Web Application that syncs with last weeks Business logic code. By the end of the week, Team D wanted a working demo of the web application.

**Schedule:**

Team D is currently slightly behind schedule due to programming the Heroku Web Application being a bit more difficult than expected. Source code for the Web application is nearly complete but does not integrate the Business Logic as we would have liked.

**Problems and Lessons Learned**

**Problem – 1** Web application building was not as easy as we at first thought it would be. Our team member in charge of front end development has had to familiarize himself with server-side programming to build the source code. Lesson Learned: When there is a time limit, sometimes things need to be reevaluated.

**Reevaluation**

After this week, the team has been reevaluating how we would like to present the application to the user (whether Heroku Web Application or plain Java GUI). Due to time constraints and hitting unexpected bumps, we are reevaluating which interface to use and considering dropping the SQL Database idea in order to make time to sync the Business Logic and GUI and then use the final week for refinement.

**Changes to Previous Documentation:**

The biggest changes have been to the original Heroku files that were being developed the past 3 weeks. Now that there is working business logic, the team has focused on getting the front end developed and connecting the two layers. The HTML files are complete and are ready to be integrated with the business logic. The request mapping for the result page is hard coded with a result. The parameters are being passed successfully in the URL. Next week will be integrating those results into the business logic.

## Phase 3

**Project Roles:**

|  |  |  |
| --- | --- | --- |
| **Name** | **Area of Responsibility** | **This Week Accomplishments** |
| Brent Silvernagel | Test, Integration, and User Manual Development | Integration of Project Plan and phase 3 documentation. |
| Joshua Thomas | Database Management Development | Working database integration. |
| Clayton Townley | Web Application Front End Development | Made attempts to integrate Business Logic and Heroku Web Application GUI code. |
| Deborah Snyder | Report Development and Business Logic | Created Phase 2 Report and initial GUI code |

**Planned Milestones:**

Integrate Web application with Business Logic and prepare for final touches.

**Schedule:**

Team D is behind schedule due to being unable to integrate business logic into the Heroku Web application code. Due to this, the team has decided to move forward with the creating a Java console GUI in order to complete the project in allotted time.

**Problems and Lessons Learned**

**Problem – 1** Inability to integrate business logic to Heroku Web Application.

Lesson Learned: When there is a time limit, make changes that need to be done that still meet requirements.

**Problem – 2** Delegation of duties. The team did not do a good job delegating work and statusing current workloads. This failure has forced the team to fallback on a console based application to complete the application within the allotted time.

Lesson Learned: More emphasis on project management to keep the team on track to ensure the team is meeting project goals and deadlines.

**Problem – 3** Poor communication. The team has not communicated as well as need to properly integrate efforts. Mostly, the team has relied too heavily on a single person to complete the goals for the given sprint. This is partially due to schedule conflicts with work, personal life, and geographic location.

Lesson Learned: Again, having a more dedicated person to project management to keep the team’s efforts coordinated would have been helpful.

**Reevaluation**

After many attempts, the team has reevaluated the user interface of choice and have decided to utilize Java console GUI instead. Initial GUI code was quickly created, integrated business logic and tested against test plan. This kind of change means the team will have to redo multiple parts of documentation especially the user guide.

**Changes to Previous Documentation:**

The biggest changes this week were to the project plan and source code. The team has added initial GUI class to source code in order to display application. Project plan must be updated with the change of web application/user interface as well as to add the Phase 3 report. In addition, there will be some changes to our UML diagrams that need to incorporated for the final delivery.