

# Final Project

MBA 614

Eric Davies



10

## Contents

Executive Summary.....	3
Business Description.....	3
Purpose of System.....	3
Overview of System.....	3
Formatting and User Experience.....	4
Development of user interface.....	4
Hyperlinks.....	4
Every Sheet a Form.....	4
Toggle Buttons.....	5
Summarize/View Detail Buttons.....	5
Update Data w/Interface Response.....	5
Easy to maintain and controls on data.....	6
Change Vital Data without Changing Formulas and Macros.....	6
Conditional Formatting and Data Validation.....	6
Error Handling for Incomplete Data.....	6
Clear Data.....	6
Flexible Reporting Features.....	7
Generate Report.....	7
Multiple Series.....	7
Clear and Print Report.....	7
Description of Code Used and Issues Addressed.....	8
Relative References.....	8
Update Data.....	8
Populate Arrays.....	9
Update Results in Database.....	9
Hide/Summarize Data on Worksheets.....	10
Remove Checkmark if Data is Changed.....	10
Adding Charts.....	11
Removing Additional Series & Printing.....	11
Learning Experience.....	12

## **Executive Summary**

### *Business Description*

The Pendulum Court in the Eyring Science Center is a restaurant managed and run by dietetics students. The restaurant, while successful at training its students and providing high-quality food for affordable prices, did not have a viable system to record daily production and sales and generate reports that were useful to the managers. The spreadsheet presented here will be used each semester to record production and sales and generate reports to show the activity for the semester.

### *Purpose of System*

I set out to create this system with the following goals in mind.

- Attractive and intuitive user interface
- Able to be maintained and updated by those with little exposure to Excel and data validation and controls around the data-entry
- Able to generate reports at any time and provide a snapshot of weekly activity

### *Overview of System*

The system is a series of worksheets built on a similar template with hyperlink navigation that allows the user to jump from sheet to sheet without realizing that they are simply changing worksheets. Within each week, each day can be opened separately or hidden and the detail can be summarized or hidden by the click of a button.

The data entry has many data validation controls built in that allow even an inexperienced user to use the spreadsheet and the system still maintains the integrity of the data. After the user enters the data, they click a button that submits the data to a database from which reports can be generated. When they submit, the interface produces a checkmark next to the day to show that the data was updated successfully. If anything is changed, the checkmark is erased as well as the corresponding data on the database so that the user will be prompted to resubmit complete data. Reports can then be generated from the numerous metrics measured each day and week.

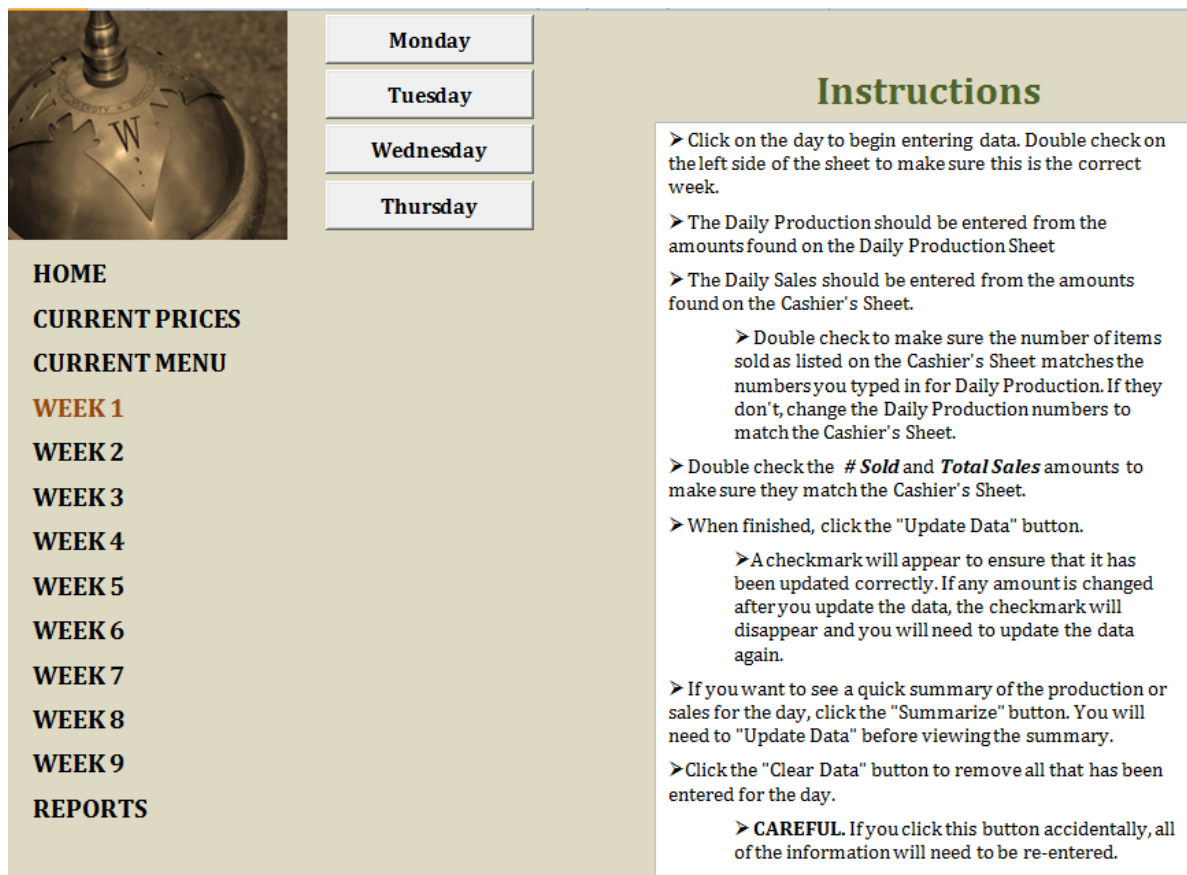
## Formatting and User Experience

### *Development of user interface*

I was hired by the Pendulum Court to create a system that would be used by their students to enter the total production and sales data each day. *Each student will only use the spreadsheet once or twice the entire semester, as they rotate through their duties.* The user will have very limited experience with Excel, and so I set out to create a system where the interface is intuitive and easy to use and understand.

### Hyperlinks

Below is a picture of the user interface while on Week 1. On the side of the page are the hyperlinks to jump to the separate pages where information can be entered and updated. Having this navigation menu helps the user feel like each separate sheet is part of one fluid program.



The screenshot displays a web-based user interface for data entry. On the left, a vertical navigation menu lists various sections: HOME, CURRENT PRICES, CURRENT MENU, WEEK 1 (highlighted in orange), WEEK 2, WEEK 3, WEEK 4, WEEK 5, WEEK 6, WEEK 7, WEEK 8, WEEK 9, and REPORTS. Above this menu is a small image of a metal object with a 'W' on it. To the right of the menu is a vertical stack of four buttons labeled Monday, Tuesday, Wednesday, and Thursday. The main content area on the right is titled 'Instructions' in green. It contains a list of instructions for using the system, including clicking on the day to enter data, double-checking the left side of the sheet, entering data from specific sheets (Daily Production and Cashier's Sheet), double-checking item counts and sales amounts, clicking the 'Update Data' button, and a 'CAREFUL' warning about the 'Clear Data' button.

**Monday**  
**Tuesday**  
**Wednesday**  
**Thursday**

### Instructions


- Click on the day to begin entering data. Double check on the left side of the sheet to make sure this is the correct week.
- The Daily Production should be entered from the amounts found on the Daily Production Sheet
- The Daily Sales should be entered from the amounts found on the Cashier's Sheet.
  - Double check to make sure the number of items sold as listed on the Cashier's Sheet matches the numbers you typed in for Daily Production. If they don't, change the Daily Production numbers to match the Cashier's Sheet.
- Double check the **# Sold** and **Total Sales** amounts to make sure they match the Cashier's Sheet.
- When finished, click the "Update Data" button.
  - A checkmark will appear to ensure that it has been updated correctly. If any amount is changed after you update the data, the checkmark will disappear and you will need to update the data again.
- If you want to see a quick summary of the production or sales for the day, click the "Summarize" button. You will need to "Update Data" before viewing the summary.
- Click the "Clear Data" button to remove all that has been entered for the day.
  - **CAREFUL.** If you click this button accidentally, all of the information will need to be re-entered.

### Every Sheet a Form

I originally wanted to create separate userforms for the data entry, but after discussion with the managers at the Pendulum Court they preferred to have the data entered into the spreadsheet. Part of their class is helping their students become familiar with business processes and the use of a spreadsheet. In order to achieve this goal, I created the "form" for daily production and sales within each spreadsheet.

## Toggle Buttons

As shown below, at the top are the toggle buttons that allow the user to view or hide any particular day when the toggle button is depressed. A user can view multiple days at a time or hide them all. When a day is clicked, the instructions move to the side of the day and the data is ready to be entered. The toggle button is depressed and the caption changes to “Hide Monday” so that the user can quickly hide the day when finished entering data. This allows the user to avoid information overload as they only work on entering data one day at a time.



Hide Monday

Tuesday

Wednesday

Thursday

Update Data

Clear Data

**Monday**

Summarize

**Daily Production**

Type	Name of Menu Item	# Prod.	# Sold
Feature	Chicken Puff Pastry		
Soup	Taco Soup		
Cold Plate	Vermont Turkey Club w/Spinach		
Taco Salad	Beef Taco Salad		
Dessert	Decadent Fudge Cake (+ 12)		
Muffin	Cherry Chocolate Scone		
Cookie	Oatmeal Golden Raisin Cookie		

Summarize

**Cashier Sheet**


Type	# Sold	Price	Total Sales
Feature		\$5.35	
Cold Plate		\$5.00	
Taco Salad		\$4.50	
Soup		\$2.50	
Dessert		\$2.50	
Muffin		\$1.25	
Cookie		\$0.75	
<b>A La Carte:</b>			
Entrée		\$3.00	\$0.00
Vegetable		\$1.00	\$0.00
Starch		\$1.35	\$0.00
<b>Misc. Items:</b>			
Fruit		\$0.60	\$0.00

## Summarize/View Detail Buttons

The user can click the “Summarize” button to hide the detail of that section and the program will provide a quick summary of the information on it below the heading.

## Update Data w/Interface Response

When the user is finished entering the data, they can click “Update data” which populates various arrays and then submits the data to a database. A checkmark appears next to the day updated to show that the entry for that day has been completed. If any data-entry cell for that day is changed, the checkmark will go away and the user will need to update the data again.



Hide Monday ✓

Tuesday

Wednesday

Thursday

Update Data

Clear Data

**Monday**

View Detail

Summarize

**Daily Production**

There were 154 basic items produced and 122 basic items sold

Type	# Sold	Price	Total Sales
Feature	25	\$5.35	\$133.75
Cold Plate	12	\$5.00	\$60.00
Taco Salad	14	\$4.50	\$63.00
Soup	18	\$2.50	\$45.00
Dessert	13	\$2.50	\$32.50
Muffin	18	\$1.25	\$22.50
Cookie	22	\$0.75	\$16.50
<b>A La Carte:</b>			
Entrée	4	\$3.00	\$12.00
Vegetable	6	\$1.00	\$6.00
Starch	7	\$1.35	\$9.45
<b>Misc. Items:</b>			
Fruit	24	\$0.60	\$14.40
Roll	35	\$0.60	\$21.00
Soda	6	\$0.60	\$3.60
Chocolate Milk	3	\$1.00	\$3.00
2% Milk	5	\$1.00	\$5.00
Skim Milk	2	\$1.00	\$2.00
V-8 Splash	1	\$1.50	\$1.50
Misc.	0		\$0.00

Subtotal of Net Income: \$451.20

Minus Discount Meals: \$42.50

Net Income: \$408.70

Number of Guests: 55

### Easy to maintain and controls on data

I anticipate that the users or managers of this system will not have a lot of exposure to Excel. As the restaurant changes (menu, prices, etc.) then the managers will need to be able to update the data. The managers also want to make sure that individual users don't erase formulas or other vital information while entering data.

### Change Vital Data without Changing Formulas and Macros

A user can navigate to the current menu or current price list and edit these master lists, which will then edit the data on the rest of the worksheet.

### Conditional Formatting and Data Validation

The cells available for data entry are highlighted and the rest of the sheet is locked so that it can't be edited. As the user enters data, the cells change color. Individual cells have input boxes to prompt the user, and if the user enters inconsistent data (sales don't add up correctly) then they will be prompted that there is a discrepancy.

A La Carte:			
Entrée	3	\$3.00	\$9.00
Vegetable	4	\$1.00	\$4.00
Starch	5	\$1.35	\$6.75

Misc. Items:			
Fruit	18	\$0.60	\$10.80
Roll	17	\$0.60	\$10.20
Soda	12	\$0.60	\$7.20
Chocolate Milk	5	\$1.00	\$5.00
2% Milk	5	\$1.00	\$5.00
Skim Milk	2	\$1.00	\$2.00
V-8 Splash	0	\$1.50	\$0.00
Misc.	0		
Misc. Income:			\$0.00

Subtotal of Net Income:	\$567.00	DISCREPANCY
Minus Discount Meals:	\$56.00	
Net Income:	\$511.00	
Number of Guests:	4	

Instructions  
"Total Other Disc"  
on the Sales Report

Hide Report Data

### Error Handling for Incomplete Data

As explained, when the "Update Data" button is clicked, an array is populated and the data is populated in a database. If the user clicks this button before having entered data, the program has no values to populate this array and a runtime error occurs. I have created an error handler and the result is shown below when this error occurs. Then the checkmark is removed from the corresponding day and the user never has to see the runtime error.

Monday

Tuesday

Wednesday

Hide Thursday

Update Data

Clear Data

Thursday

Summarize

Sum

There were 215 items sold with total sales of \$451.2

Hide Report Data

DATA HIDDEN - USED FOR REPORTS

Total Gross Sales: \$0.00

Total Sales Count: 0

Equivalent Meals Served: 0

Average Check: \$2.10

We "normalize" the data by minutes so that

Complete Form

Please Fill Out The Form Completely Before Updating

OK

### Clear Data

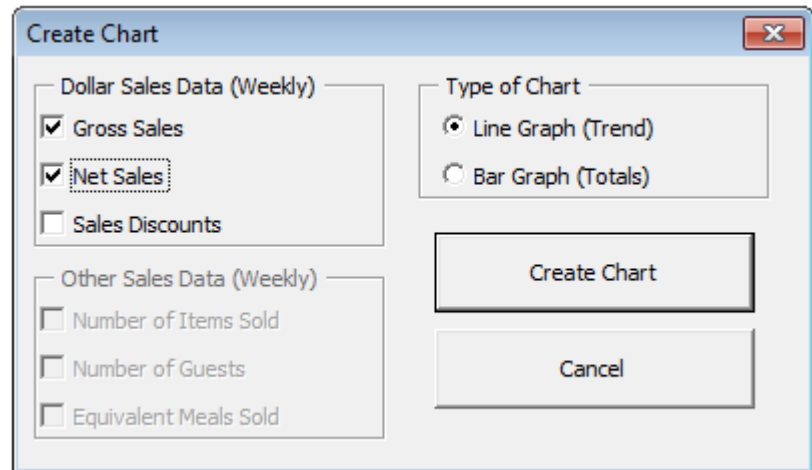
If a user needs to re-enter data or is clearing out the system for use in a different semester, they can click the "Clear Data" button to quickly clear all data out for the day and it also clears the corresponding data from the database. This provides some assurance that the database doesn't have old or inaccurate data.

### *Flexible Reporting Features*

The purpose of entering the daily production and sales of the Pendulum Court is not simply to provide a learning experience for its students. I wanted the managers who use this system to be able to generate reports to quickly view summarized data and trends.

#### Generate Report

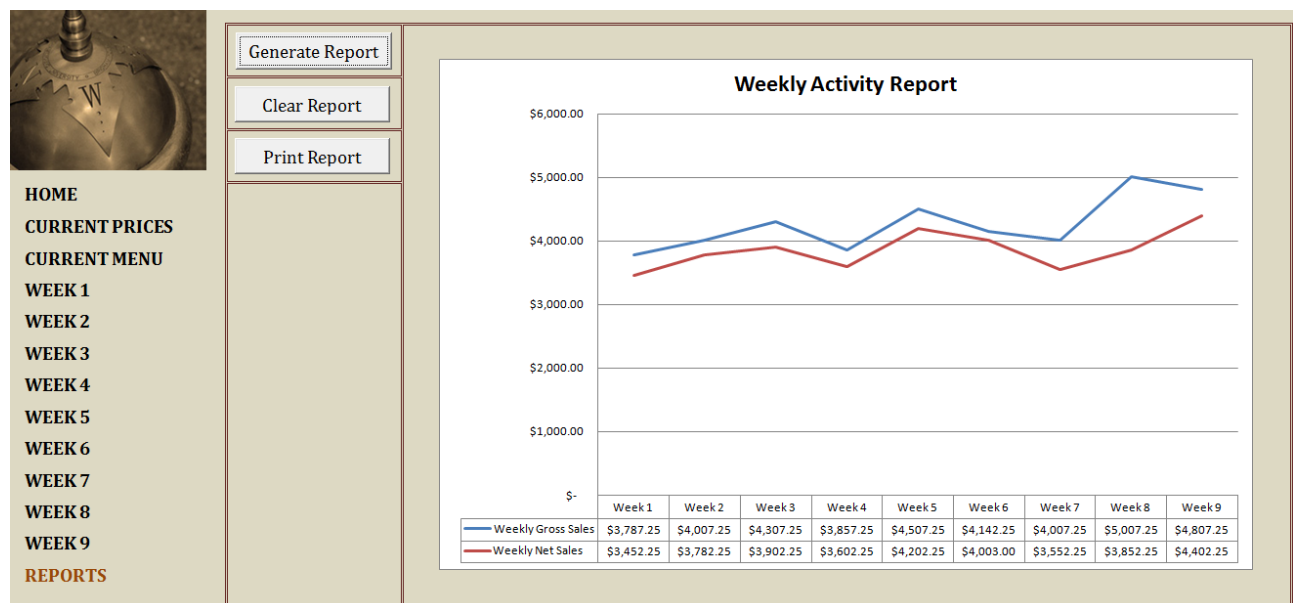
After navigating to the “Reports” menu, the user can click a button to generate a report. After doing so, the following userform appears. They can choose to create either a line or bar graph with various sources of data.



The 'Create Chart' dialog box allows users to select data series and chart types. It features two main sections: 'Dollar Sales Data (Weekly)' and 'Other Sales Data (Weekly)'. The 'Dollar Sales Data' section includes checkboxes for 'Gross Sales', 'Net Sales', and 'Sales Discounts'. The 'Other Sales Data' section includes checkboxes for 'Number of Items Sold', 'Number of Guests', and 'Equivalent Meals Sold'. On the right, the 'Type of Chart' section offers radio buttons for 'Line Graph (Trend)' and 'Bar Graph (Totals)'. At the bottom right, there are 'Create Chart' and 'Cancel' buttons.

#### Multiple Series

The great thing about this reporting feature is it can compare multiple series of data at the same time, and can be changed by a simple click of a button. After selecting a series, the incomparable data (i.e. gross sales and number of items sold) will not be able to be selected.



#### Clear and Print Report

If the user clicks the “Clear Report” button then the series are erased and an empty chart stays on the sheet. If the user finds the generated report useful and wants to print the summary I have included a button to print the report. By clicking that button, the user will be sent to the print preview screen with the chart optimally formatted and ready to print.

## Description of Code Used and Issues Addressed

### Relative References

Every sheet has dozens of objects (text boxes, various buttons) and many only affect certain cells on a sheet. For example, the “Update Data” button for Monday, Week 1 should only populate arrays and submit data for that particular day. In order to achieve this, every subroutine that is connected to the buttons on each sheet is passed a parameter called startingposition. In the code displayed to the right are the individual buttons on each sheet. Each button calls another sub (ClearData or UpdateData) and then passes a parameter, which is the range that displays the day this button refers to.

```
Private Sub btnClearMonday_Click()  
  
    'This is for Monday  
    Call ClearData(Range("H4"))  
  
End Sub  
  
Private Sub btnUpdateMonday_Click()  
  
    'This is for Monday  
    Call UpdateData(Range("H4"))  
  
End Sub
```

### Update Data

This subroutine updates the data relative to the startingposition parameter by populating several arrays and then printing this data on a specific row in a database table. This specific line of code shows when update data creates the checkmark in the user interface to show that the data has been updated in the database.

```
Sub UpdateData(startPosition As Range)  
    'Run this code when clicking "Update Data"  
    'The starting position allows this sub to run relative for each day  
  
    Dim x As Byte  
  
    'This shows the green checkmark for each respective day  
    If startPosition.Value = "Monday" Then  
        With Range("F1").Font  
            .ThemeColor = xlThemeColorAccent3  
            .TintAndShade = -0.249977111117893  
        End With  
    End If  
  
    If startPosition.Value = "Tuesday" Then  
        With Range("F3").Font  
            .ThemeColor = xlThemeColorAccent3  
            .TintAndShade = -0.249977111117893  
        End With  
    End If  
  
    If startPosition.Value = "Wednesday" Then  
        With Range("F4").Font  
            .ThemeColor = xlThemeColorAccent3  
            .TintAndShade = -0.249977111117893  
        End With  
    End If  
  
    If startPosition.Value = "Thursday" Then  
        With Range("F6").Font  
            .ThemeColor = xlThemeColorAccent3  
            .TintAndShade = -0.249977111117893  
        End With  
    End If
```



## Populate Arrays

The UpdateData subroutine also calls other subs to populate various arrays. Below is the code for one particular array that uses a For loop to populate sales from the data entered on the sheet.

```
Sub PopulateDailyDollarSales(startPosition As Range)

Dim x As Byte
Dim r As Byte

'Populate the dailyDollarSales array

r = 1

'Populate array with basic item sales
For x = 0 To 6
dailyDollarSales(x) = Range(startPosition.Offset(15, 2), startPosition.Offset(21, 2)).Cells(r).Value
r = r + 1
Next

'Populate array with A La Carte item sales
aLaCarteSales = 0
For x = 1 To 3
aLaCarteSales = aLaCarteSales + _
Range(startPosition.Offset(24, 2), startPosition.Offset(26, 2)).Cells(x).Value
Next

dailyDollarSales(7) = aLaCarteSales

'Populate array with Misc item sales
miscItemSales = 0
For x = 1 To 8
miscItemSales = miscItemSales + _
Range(startPosition.Offset(29, 2), startPosition.Offset(36, 2)).Cells(x).Value
Next

dailyDollarSales(8) = miscItemSales

End Sub
```

## Update Results in Database

This was one of the most difficult problems I solved while working on this project. I wanted the results of each day to be updated in a specific row in the database. I finally succeeded in doing this by creating the following algorithm.

This creates a value for the day and week. Based on those values, I calculate "r" with this algorithm (which represents the row number) and then proceed to populate various columns on this specific row of the database. This would allow a more advanced user to use PivotTables with this database if needed.

```
Dim r As Integer
Dim c As Integer

'populate week value
currentWeek = ActiveSheet.Range("e13").Value

Dim weekName As String

weekName = startPosition.Value

'populate day value|
If weekName = "Monday" Then
currentDay = 1
ElseIf weekName = "Tuesday" Then
currentDay = 2
ElseIf weekName = "Wednesday" Then
currentDay = 3
ElseIf weekName = "Thursday" Then
currentDay = 4
End If

'Choose which row to insert data on data sheet
r = 4 * (currentWeek - 1) + currentDay + 1

'Week Number
ActiveWorkbook.Sheets("data").Cells(r, 1).Value = currentWeek

'Day
ActiveWorkbook.Sheets("data").Cells(r, 2).Value = startPosition.Value
```

### *Hide/Summarize Data on Worksheets*

Each worksheet represents a data-entry interface that the user can manipulate to hide and show specific parts of the data. This is accomplished with the creative use of toggle buttons and hiding rows. Here is the code for my toggle button that hides or shows the “Monday” data form. The code for the other toggle buttons is similar.

This uses several loops to hide and show rows depending on whether the toggle button is depressed or not.

```
Private Sub ToggleButton9_Click()  
Dim x As Byte  
Application.ScreenUpdating = False  
  
If ToggleButton9.Value = True Then  
    'This is what happens when you click "Monday"  
    Range("A1").Select  
  
    For x = 7 To 10  
        Columns(x).EntireColumn.Hidden = False  
    Next  
  
    ToggleButton9.Caption = "Hide Monday"  
  
Else  
    'This is what happens when you click "Hide Monday"  
    Range("A1").Select  
  
    For x = 7 To 10  
        Columns(x).EntireColumn.Hidden = True  
    Next  
  
    ToggleButton9.Caption = "Monday"  
  
End If  
Application.ScreenUpdating = True  
End Sub
```

### *Remove Checkmark if Data is Changed*

One particular feature I wanted to implement was for the checkmark that appears when a user “Updates Data” to disappear when something in that particular range is changed. This would alert the user to resubmit the data so that the database would stay up-to-date. This was much more complicated than I anticipated, and required a much greater understanding of parameters and events within this worksheet. I found the worksheet\_change event that allowed me to target specific ranges that could be changed to trigger this event. The code required a significant use of “And” and “Or” statement to identify the exact portion of the sheet I wanted to trigger the event. The first part of the code looked like this.

```
Private Sub Worksheet_Change(ByVal Target As Range)  
'These statements will trigger if any of the selected cells are changed  
  
    'Monday  
    If Target.Row >= 8 And Target.Row <= 14 And Target.Column >= 9 And Target.Column <= 10 Or _  
        Target.Row >= 28 And Target.Row <= 30 And Target.Column = 8 Or _  
        Target.Row >= 33 And Target.Row <= 40 And Target.Column = 8 Or _  
        Target.Row = 40 And Target.Column = 10 Then  
        'Clears the green checkmark  
        With Range("f1").Font  
            .ThemeColor = xlThemeColorDark2  
            .TintAndShade = -9.99786370433668E-02  
        End With  
    End If
```

## Adding Charts

Developing my code to use my chart userform and create flexible reporting depending on multiple inputs was an incredible amount of work. I needed to develop a greater understanding of objects and their properties as well as how to manipulate them in this workbook. The first part of my addchart code that I will go over is simply creating variables based on the inputs from my userform. After I create these variables then I use a variety of If statements to populate each series.

I also created code to let me edit the type of chart. I ran this in my addchart code.

## Removing Additional Series & Printing

One of the more difficult pieces of code I needed to write included removing the extra series in the chart. I found help from an online resource which saved me quite a bit of time and I eventually used to better understand the properties of chart objects.

I also used this online resource to learn about the various methods used to print specific objects

That code is listed here below.

```
Sub RemoveAllSeries()  
    'This clears the graph  
    Dim c As Shape  
    Set c = Sheets("reports").Shapes(15)  
  
    With c.Chart  
        Do Until .SeriesCollection.Count = 0  
            .SeriesCollection(1).Delete  
        Loop  
    End With  
End Sub  
  
Sub PrintChart()  
    Dim c As Shape  
    Set c = Sheets("reports").Shapes(15)  
  
    c.Chart.PrintPreview  
End Sub
```

## Option Explicit

```
Sub AddChart()  
    'This sets the chart shape as a variable  
    Dim c As Shape  
    Set c = Sheets("reports").Shapes(15)  
  
    'These variables hold properties of the form  
    Dim grossSales As Boolean  
    Dim salesDiscounts As Boolean  
    Dim netSales As Boolean  
    Dim itemsSold As Boolean  
    Dim equivalentMeals As Boolean  
    Dim numberOfGuests As Boolean  
  
    grossSales = frmCreateChart.chkGrossSales.Value  
    salesDiscounts = frmCreateChart.chkSalesDiscounts.Value  
    netSales = frmCreateChart.chkNetSales.Value  
    itemsSold = frmCreateChart.chkItemsSold.Value  
    equivalentMeals = frmCreateChart.chkEquivalentMeals.Value  
    numberOfGuests = frmCreateChart.chkNumberOfGuests.Value  
  
    'These are the variables for each new series  
    Dim seriesName As String  
    Dim valuesRange As Range  
    Dim xvaluesRange As Range  
  
    Call RemoveAllSeries  
    Call EditChartType  
  
    c.Chart.ChartTitle.Text = "Weekly Activity Report"  
  
    If grossSales = True Then  
        seriesName = "Weekly Gross Sales"  
        Set valuesRange = Sheets("reportdata").Range("G2:G10")  
        Set xvaluesRange = Sheets("reportdata").Range("A2:A10")  
  
        With c.Chart.SeriesCollection.NewSeries  
            .Name = seriesName  
            .Values = valuesRange  
            .XValues = xvaluesRange  
        End With  
    End If
```

## **Learning Experience**

This was a wonderful learning experience where I solved a real business problem and added value to a specific organization. Everything I implemented was a challenge and stretched me to learn and study more about VBA. I learned about the activeX controls in Excel, as I made use of those to provide a rich user interface experience for the users of this program.