Stephen Jensen
IS 520
Final Project
12/11/14
Personal Budget

Executive Summary

For my personal project I decided to work on my personal budget. I designed this project to help alleviate two pain points I have, first categorizing my spending and second analyzing home and car loans.

Keeping a budget is something that I take very seriously. I really try to keep track of where I am spending money and for areas where I can save. To track my spending, I use a VBA code to categorize each transaction I make. In the past, I would spend three to four hours a month manually entering each transaction into each designated category. Needless to say this process is tedious and frustrating.

Along with tracking my spending, I am also concerned about getting a car loan and home loan. I currently have a car loan and will be looking to get a home loan in the next two to three years. Being able to quickly crunch the numbers for different loan amounts, down payments, interest rates, and terms is something that I would find very useful now and in the future. I also am very interested to see how much interest I will save over the life of the loan by making extra payments each month on my loan.

The first part of my system is designed to ease the annoyance of manually categorizing each of my transactions. Now all I need to do is download my credit card transactions from my bank and put them in a sheet in my workbook. Then simply click the button “Input Transactions” and the code will automatically place each transaction in the desired month and category. This code will save me three to four hours each month and allow me to track my spending month to month.

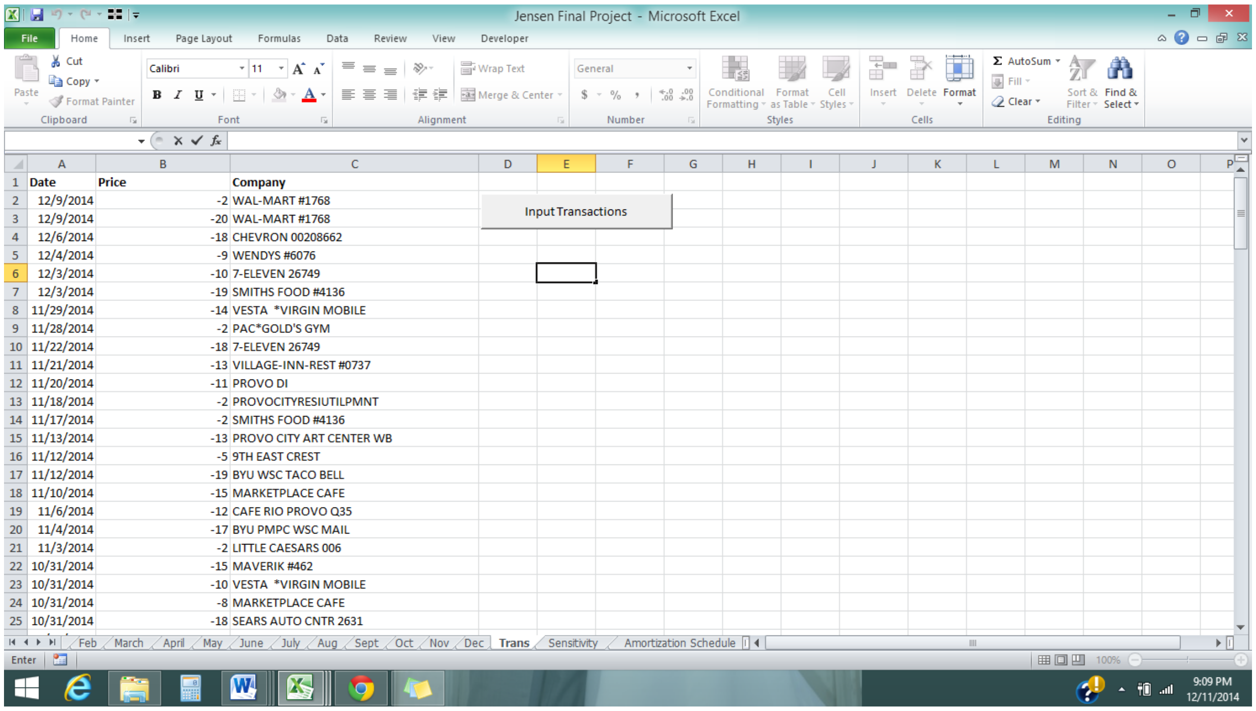
The second part of my system allows me to quickly analyze loans based upon the loan amount, down payment, interest rate, and terms. This system allows me to enter these four inputs and then runs a sensitivity analysis on each input. That is it shows me how the monthly payment and total interest paid varies if each input were to increase or decrease. I also included an amortization schedule based off of those four inputs. The amortization schedule also allows me to enter an extra payment amount, how many months I will make that extra payment and the number of months remaining on the loan then calculates how much interest I will save. This will allow me to analyze loans quickly and intelligently.

This system will be very helpful for me to keep track of my spending and analyze home and car loans.

Implementation

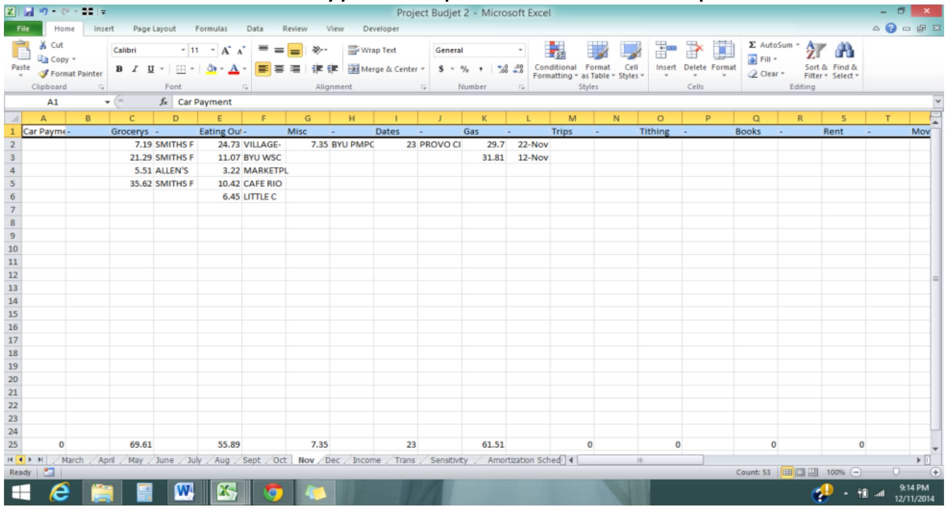
Organizing my Spending

The first part of my system is categorizing my spending from month to month. In order to do this, I first download my credit card statements and put them into a sheet in my workbook labeled “Trans.” The following screen shot shows the “Trans” sheet with each expense item.



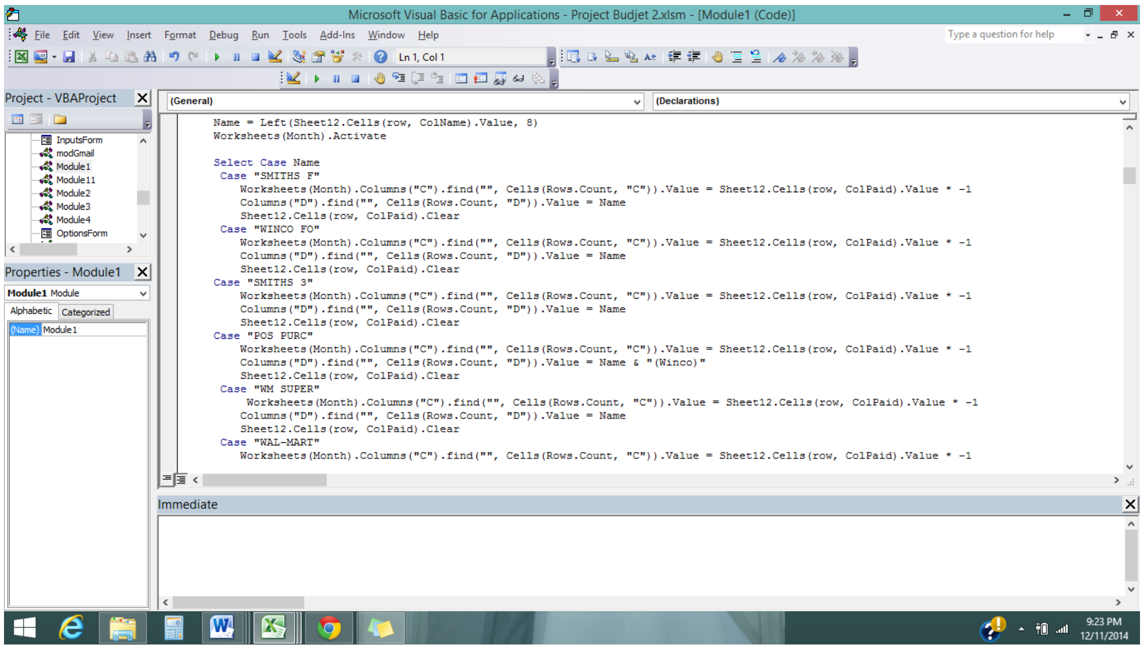
	Date	Price	Company
1	Date	Price	Company
2	12/9/2014	-2	WAL-MART #1768
3	12/9/2014	-20	WAL-MART #1768
4	12/6/2014	-18	CHEVRON 00208662
5	12/4/2014	-9	WENDYS #6076
6	12/3/2014	-10	7-ELEVEN 26749
7	12/3/2014	-19	SMITHS FOOD #4136
8	11/29/2014	-14	VESTA *VIRGIN MOBILE
9	11/28/2014	-2	PAC*GOLD'S GYM
10	11/22/2014	-18	7-ELEVEN 26749
11	11/21/2014	-13	VILLAGE-INN-REST #0737
12	11/20/2014	-11	PROVO DI
13	11/18/2014	-2	PROVOCITYRESIUTILPMNT
14	11/17/2014	-2	SMITHS FOOD #4136
15	11/13/2014	-13	PROVO CITY ART CENTER WB
16	11/12/2014	-5	9TH EAST CREST
17	11/12/2014	-19	BYU WSC TACO BELL
18	11/10/2014	-15	MARKETPLACE CAFE
19	11/6/2014	-12	CAFE RIO PROVO Q35
20	11/4/2014	-17	BYU PMPC WSC MAIL
21	11/3/2014	-2	LITTLE CAESARS 006
22	10/31/2014	-15	MAVERIK #462
23	10/31/2014	-10	VESTA *VIRGIN MOBILE
24	10/31/2014	-8	MARKETPLACE CAFE
25	10/31/2014	-18	SEARS AUTO CNTR 2631

Then all I have to do is click the “Input Transactions” button and the system will put each expense item in the designated month and category. As seen on the above screen shot, I have a tab labeled for each month. Each of these tabs looks like the following screenshot. These sheets have a column for each type of expense I want to keep track of.



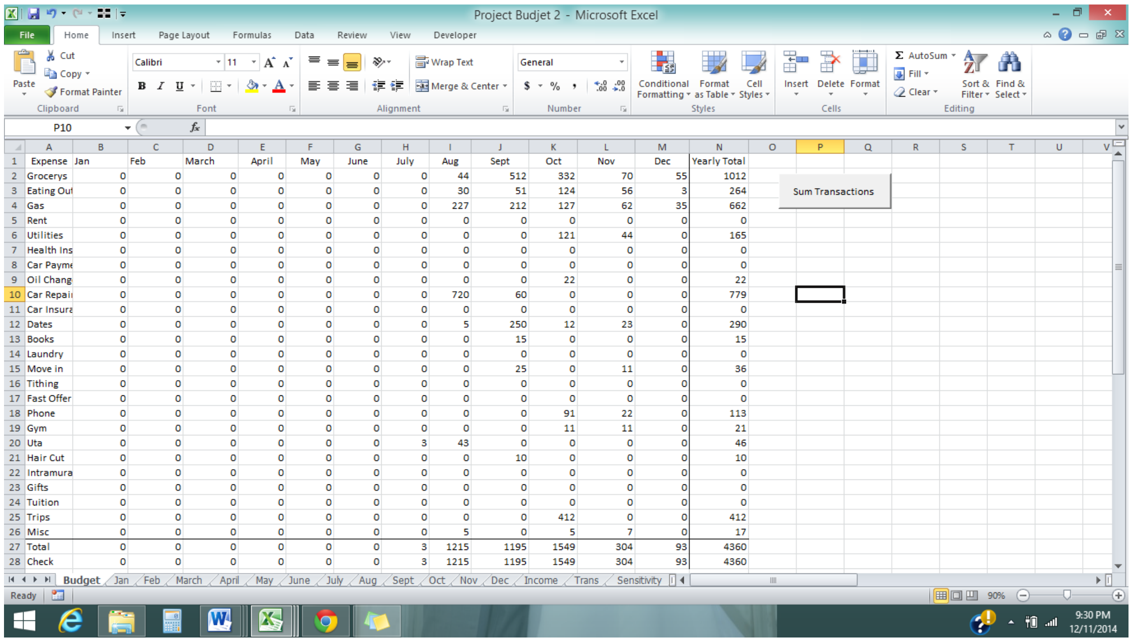
	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
1	Car Payme	-	Grocerys	-	Eating Out	-	Misc	-	Dates	-	Gas	-	Trips	-	Tithing	-	Books	-	Rent	-
2			7.19	SMITHS F	24.73	VILLAGE-	7.35	BYU PMPC	23	PROVO CI	29.7	22-Nov								
3			21.29	SMITHS F	11.07	BYU WSC					31.81	12-Nov								
4			5.51	ALLEN'S	3.22	MARKETPL														
5			35.62	SMITHS F	10.42	CAFE RIO														
6					6.45	LITTLE C														
25	0		69.61		55.89		7.35		23		61.51		0		0		0		0	

In order to make this code work, I had to loop through each transaction on the page. I used the left function as well as select case statements. The left functions captured the month and name of each store. Based on the month and store the case statements then directed where each transaction should be put. The following screenshot shows an example of the left function and case statement for the name of the store.



```
        Name = Left(Sheet12.Cells(row, ColName).Value, 8)
        Worksheets(Month).Activate

        Select Case Name
         Case "SMITHS F"
            Worksheets(Month).Columns("C").find("", Cells(Rows.Count, "C")).Value = Sheet12.Cells(row, ColPaid).Value * -1
            Columns("D").find("", Cells(Rows.Count, "D")).Value = Name
            Sheet12.Cells(row, ColPaid).Clear
         Case "WINCO FO"
            Worksheets(Month).Columns("C").find("", Cells(Rows.Count, "C")).Value = Sheet12.Cells(row, ColPaid).Value * -1
            Columns("D").find("", Cells(Rows.Count, "D")).Value = Name
            Sheet12.Cells(row, ColPaid).Clear
        Case "SMITHS 3"
            Worksheets(Month).Columns("C").find("", Cells(Rows.Count, "C")).Value = Sheet12.Cells(row, ColPaid).Value * -1
            Columns("D").find("", Cells(Rows.Count, "D")).Value = Name
            Sheet12.Cells(row, ColPaid).Clear
        Case "POS PURC"
            Worksheets(Month).Columns("C").find("", Cells(Rows.Count, "C")).Value = Sheet12.Cells(row, ColPaid).Value * -1
            Columns("D").find("", Cells(Rows.Count, "D")).Value = Name & "(Winco)"
            Sheet12.Cells(row, ColPaid).Clear
         Case "WM SUPER"
             Worksheets(Month).Columns("C").find("", Cells(Rows.Count, "C")).Value = Sheet12.Cells(row, ColPaid).Value * -1
            Columns("D").find("", Cells(Rows.Count, "D")).Value = Name
            Sheet12.Cells(row, ColPaid).Clear
         Case "WAL-MART"
            Worksheets(Month).Columns("C").find("", Cells(Rows.Count, "C")).Value = Sheet12.Cells(row, ColPaid).Value * -1
```

The final step is consolidating each month onto one sheet so I can easily view my spending by month. The next screen shot shows this consolidated page.



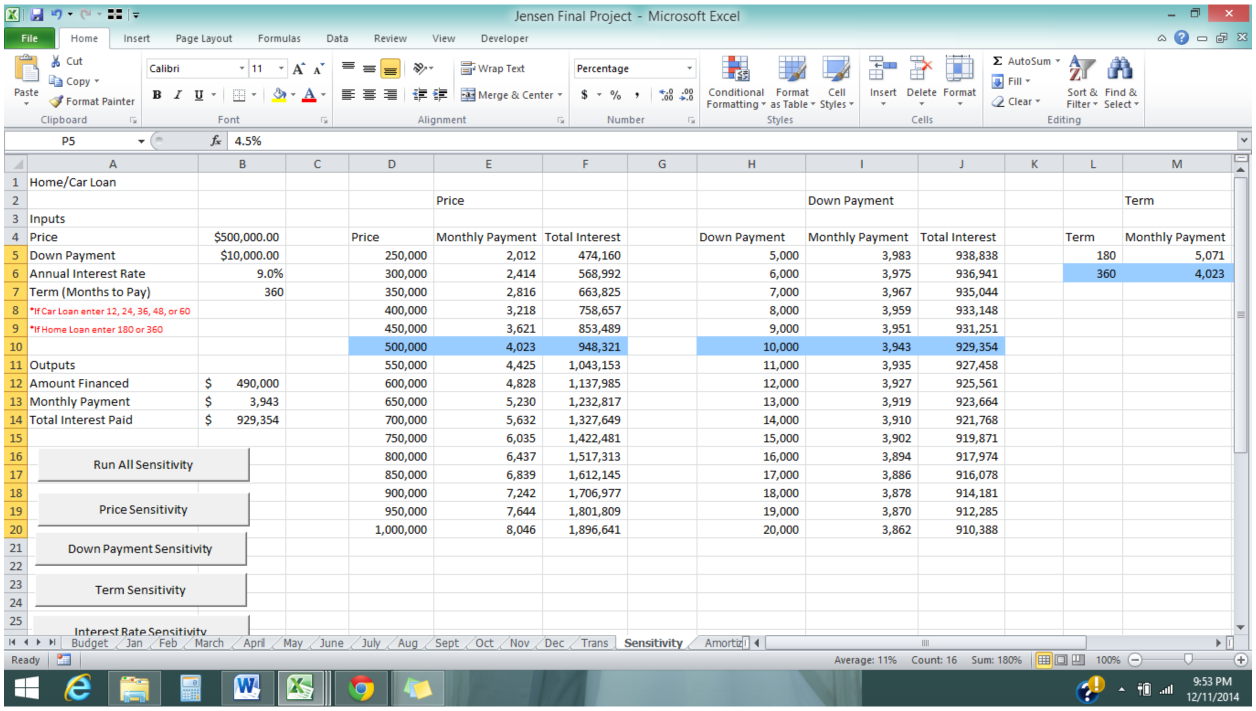
Expense	Jan	Feb	March	April	May	June	July	Aug	Sept	Oct	Nov	Dec	Yearly Total
Grocerys	0	0	0	0	0	0	0	44	512	332	70	55	1012
Eating Out	0	0	0	0	0	0	0	30	51	124	56	3	264
Gas	0	0	0	0	0	0	0	227	212	127	62	35	662
Rent	0	0	0	0	0	0	0	0	0	0	0	0	0
Utilities	0	0	0	0	0	0	0	0	0	121	44	0	165
Health Ins	0	0	0	0	0	0	0	0	0	0	0	0	0
Car Payme	0	0	0	0	0	0	0	0	0	0	0	0	0
Oil Chang	0	0	0	0	0	0	0	0	0	22	0	0	22
Car Repair	0	0	0	0	0	0	0	720	60	0	0	0	779
Car Insura	0	0	0	0	0	0	0	0	0	0	0	0	0
Dates	0	0	0	0	0	0	0	5	250	12	23	0	290
Books	0	0	0	0	0	0	0	0	15	0	0	0	15
Laundry	0	0	0	0	0	0	0	0	0	0	0	0	0
Move in	0	0	0	0	0	0	0	0	25	0	11	0	36
Tithing	0	0	0	0	0	0	0	0	0	0	0	0	0
Fast Offer	0	0	0	0	0	0	0	0	0	0	0	0	0
Phone	0	0	0	0	0	0	0	0	0	91	22	0	113
Gym	0	0	0	0	0	0	0	0	0	11	11	0	21
Uta	0	0	0	0	0	0	3	43	0	0	0	0	46
Hair Cut	0	0	0	0	0	0	0	0	10	0	0	0	10
Intramura	0	0	0	0	0	0	0	0	0	0	0	0	0
Gifts	0	0	0	0	0	0	0	0	0	0	0	0	0
Tuition	0	0	0	0	0	0	0	0	0	0	0	0	0
Trips	0	0	0	0	0	0	0	0	0	412	0	0	412
Misc	0	0	0	0	0	0	0	5	0	5	7	0	17
Total	0	0	0	0	0	0	3	1215	1195	1549	304	93	4360
Check	0	0	0	0	0	0	3	1215	1195	1549	304	93	4360

This page is a table that allows me to easily look at each expense and see how they fluctuate from month to month. The sum transactions button in the top right corner is attached to the code that pulls the transactions from each spreadsheet. For this part of the code I used three for loops that scanned through each expense sheet, then looped through each column of expenses, then looped through each expense category on the consolidated "Budget" sheet to place the sum total for each month in the proper category. This is a really helpful part of code because otherwise I would have to manually link each individual box on the consolidated "budget" sheet to each expense total on each month tab. That is a very long process that now only takes a few seconds with a VBA code.

Loans

The next part of my system deals with analyzing loans. One sheet is for analyzing how changing the four main inputs of loan amount, down payment, interest rate, and length of the term affect how much my monthly payment is and what the total interest paid will be. The next sheet is an amortization schedule that calculates how much interest I can save over the life of the loan, depending on the remaining life of the loan, how much extra I pay and the number of months I make the extra payment.

The "sensitivity" sheet examines the difference in total interest paid and monthly payments based on the four main inputs of loan amount, down payment, interest rate, and length of the term. The following screenshot shows how the page is setup.

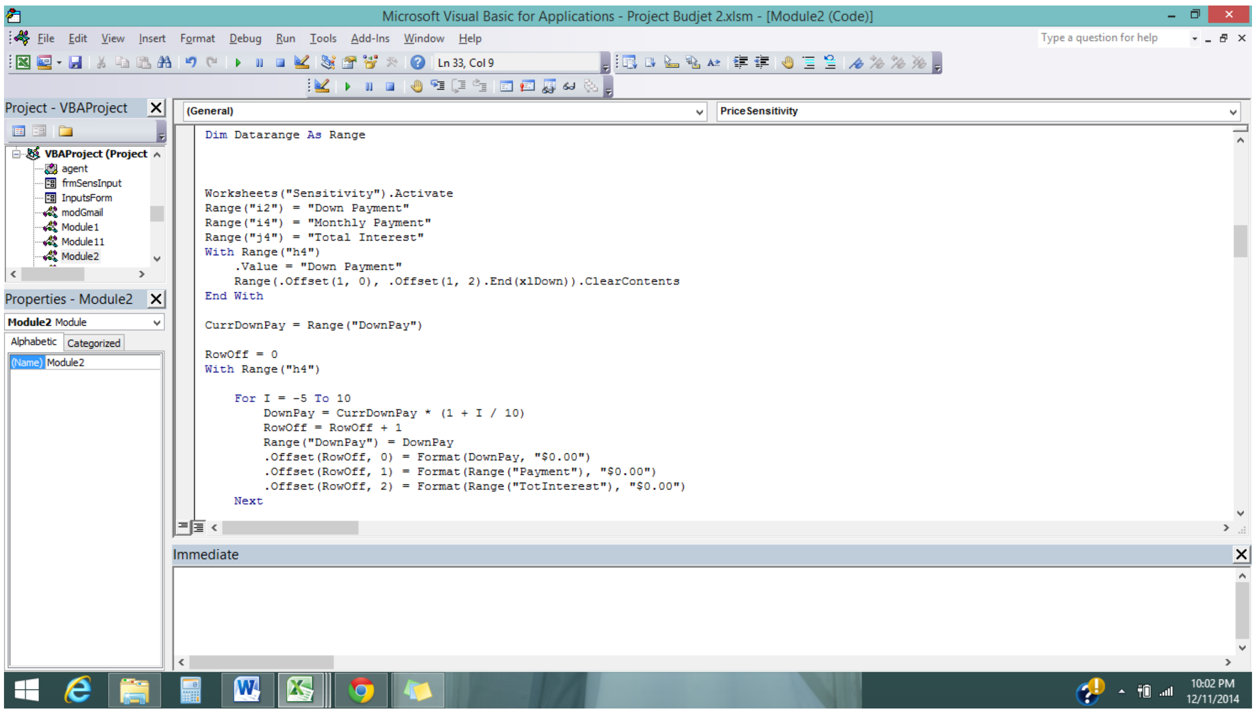


P5 4.5%

	A	B	C	D	E	F	G	H	I	J	K	L	M
1	Home/Car Loan												
2					Price				Down Payment				Term
3	Inputs												
4	Price	$500,000.00		Price	Monthly Payment	Total Interest		Down Payment	Monthly Payment	Total Interest		Term	Monthly Payment
5	Down Payment	$10,000.00		250,000	2,012	474,160		5,000	3,983	938,838		180	5,071
6	Annual Interest Rate	9.0%		300,000	2,414	568,992		6,000	3,975	936,941		360	4,023
7	Term (Months to Pay)	360		350,000	2,816	663,825		7,000	3,967	935,044			
8	*If Car Loan enter 12, 24, 36, 48, or 60			400,000	3,218	758,657		8,000	3,959	933,148			
9	*If Home Loan enter 180 or 360			450,000	3,621	853,489		9,000	3,951	931,251			
10				500,000	4,023	948,321		10,000	3,943	929,354			
11	Outputs			550,000	4,425	1,043,153		11,000	3,935	927,458			
12	Amount Financed	$ 490,000		600,000	4,828	1,137,985		12,000	3,927	925,561			
13	Monthly Payment	$ 3,943		650,000	5,230	1,232,817		13,000	3,919	923,664			
14	Total Interest Paid	$ 929,354		700,000	5,632	1,327,649		14,000	3,910	921,768			
15				750,000	6,035	1,422,481		15,000	3,902	919,871			
16				800,000	6,437	1,517,313		16,000	3,894	917,974			
17				850,000	6,839	1,612,145		17,000	3,886	916,078			
18				900,000	7,242	1,706,977		18,000	3,878	914,181			
19				950,000	7,644	1,801,809		19,000	3,870	912,285			
20				1,000,000	8,046	1,896,641		20,000	3,862	910,388			

The inputs on the top left are what I can change. The outputs just below show the total monthly payment and total interest paid. The tables then show how the monthly payment and interest change if I were to increase or decrease any one of the inputs. The price, down payment, and interest rate (table on the right out of screenshot) each vary from half to double the current input in 10% increments. The highlighted row is what the current input is. The term varies from 12 to 60 months in 12 month increments if it is a car loan, or from 180 to 360 months if it is a home loan. The buttons on the bottom left allow me to run just one of the sensitivity tables leaving the rest unchanged or run them all at the same time with the "Run All Sensitivity."

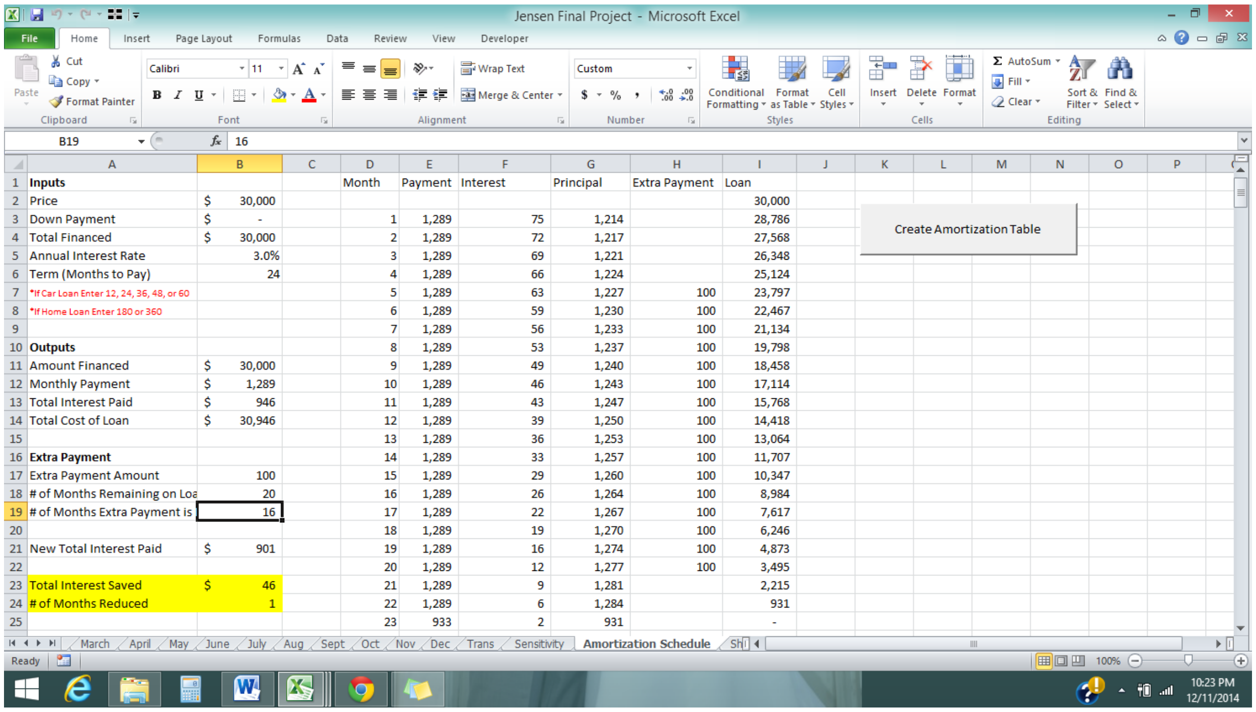
To create the sensitivity tables, I wrote four different subs that are all very similar. First it is important to mention that the formula in the monthly payment cell in the above screenshot is instrumental in writing this code. The formula is =PMT(IntRate/12,Term,-Loan). Then the formula for the total interest paid is =(Payment*Term)-Loan. When one input changes so does the monthly payment and total interest paid. So each sensitivity sub loops through and changes each input and then places each different monthly payment and total interest paid in the correlating table. Then the last part of the sub resets the input to the original input and highlights the row on the table that matches the original input. The following screenshot shows the for loop that changes the input.



```
Dim Datarange As Range



Worksheets("Sensitivity").Activate
Range("i2") = "Down Payment"
Range("i4") = "Monthly Payment"
Range("j4") = "Total Interest"
With Range("h4")
    .Value = "Down Payment"
    Range(.Offset(1, 0), .Offset(1, 2).End(xlDown)).ClearContents
End With

CurrDownPay = Range("DownPay")

RowOff = 0
With Range("h4")

    For I = -5 To 10
        DownPay = CurrDownPay * (1 + I / 10)
        RowOff = RowOff + 1
        Range("DownPay") = DownPay
        .Offset(RowOff, 0) = Format(DownPay, "$0.00")
        .Offset(RowOff, 1) = Format(Range("Payment"), "$0.00")
        .Offset(RowOff, 2) = Format(Range("TotInterest"), "$0.00")
    Next
```

The "Run All Sensitivity" button simply calls each of the individual sensitivity subs.

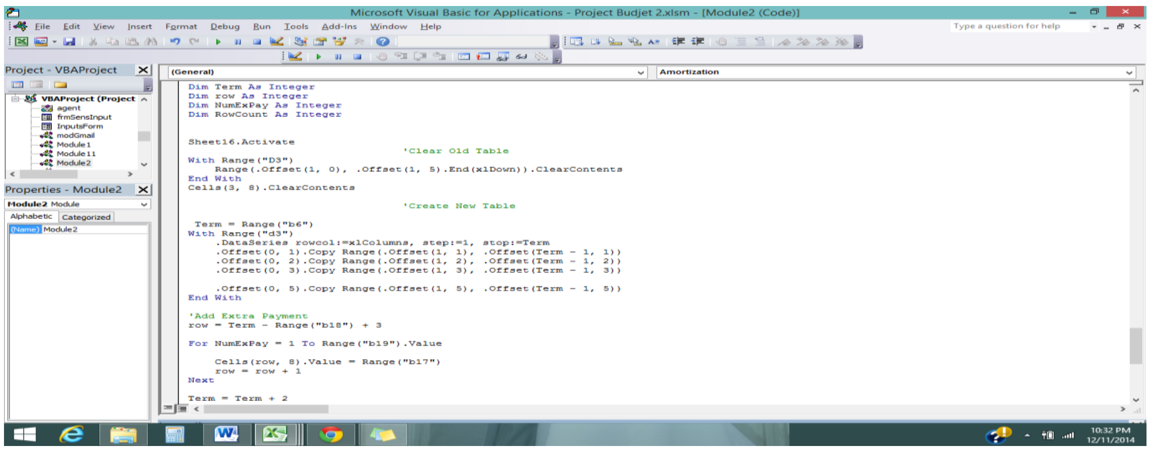
The next sheet in my workbook is the amortization schedule. This schedule allows me to build an amortization schedule and analyze how adding extra payments can affect how much interest I save. The following screenshot shows an example of the table that is created.



	A	B	C	D	E	F	G	H	I
1	**Inputs**			Month	Payment	Interest	Principal	Extra Payment	Loan
2	Price	$ 30,000							30,000
3	Down Payment	$ -		1	1,289	75	1,214		28,786
4	Total Financed	$ 30,000		2	1,289	72	1,217		27,568
5	Annual Interest Rate	3.0%		3	1,289	69	1,221		26,348
6	Term (Months to Pay)	24		4	1,289	66	1,224		25,124
7	*If Car Loan Enter 12, 24, 36, 48, or 60			5	1,289	63	1,227	100	23,797
8	*If Home Loan Enter 180 or 360			6	1,289	59	1,230	100	22,467
9				7	1,289	56	1,233	100	21,134
10	**Outputs**			8	1,289	53	1,237	100	19,798
11	Amount Financed	$ 30,000		9	1,289	49	1,240	100	18,458
12	Monthly Payment	$ 1,289		10	1,289	46	1,243	100	17,114
13	Total Interest Paid	$ 946		11	1,289	43	1,247	100	15,768
14	Total Cost of Loan	$ 30,946		12	1,289	39	1,250	100	14,418
15				13	1,289	36	1,253	100	13,064
16	**Extra Payment**			14	1,289	33	1,257	100	11,707
17	Extra Payment Amount	100		15	1,289	29	1,260	100	10,347
18	# of Months Remaining on Loa	20		16	1,289	26	1,264	100	8,984
19	# of Months Extra Payment is	16		17	1,289	22	1,267	100	7,617
20				18	1,289	19	1,270	100	6,246
21	New Total Interest Paid	$ 901		19	1,289	16	1,274	100	4,873
22				20	1,289	12	1,277	100	3,495
23	Total Interest Saved	$ 46		21	1,289	9	1,281		2,215
24	# of Months Reduced	1		22	1,289	6	1,284		931
25				23	933	2	931		-

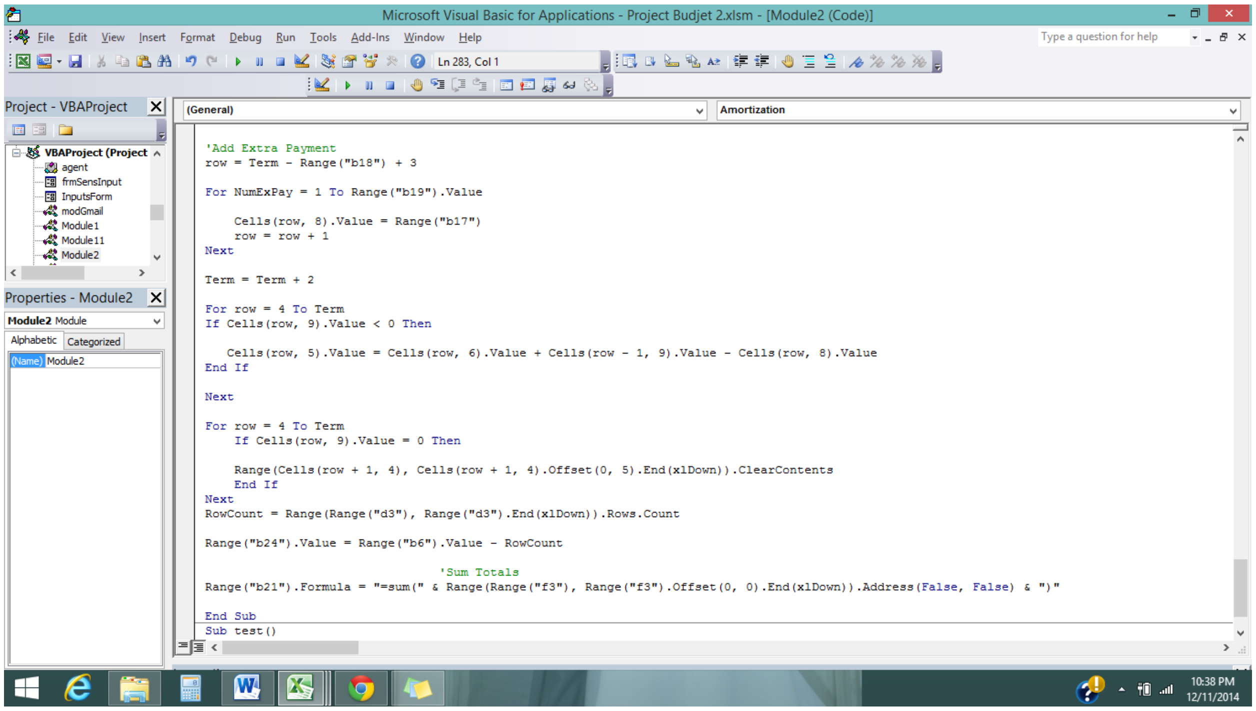
This shows that this code will first create an amortization schedule based upon the user inputs in the upper left corner of the screen. Then if the user decides to add extra payments the code will recalculate the table adding in the extra payments and show how much total interest is saved and how many months are reduced. This example shows that if I added $100 to the next 16 monthly payments with 20 months remaining on the loan, then I would save $46 of interest and the term would be reduced by one month. These numbers become much more significant as the loan amount, interest rate, and term increases.

The first part of this code clears the old table by using end(xldown). The next part creates the new table with the inputs by specifying the number of rows using the length of the term. Row 1 of the table has hard coded formulas that do not get touched by the code. The code then copies and drags these formulas down the length of the table. The screenshot below displays this part of code.



```
Dim Term As Integer
Dim row As Integer
Dim NumExPay As Integer
Dim RowCount As Integer


Sheet16.Activate
                                          'Clear Old Table
With Range("D3")
    Range(.Offset(1, 0), .Offset(1, 5).End(xlDown)).ClearContents
End With
Cells(3, 8).ClearContents

                                          'Create New Table

 Term = Range("b6")
With Range("d3")
    .DataSeries rowcol:=xlColumns, step:=1, stop:=Term
    .Offset(0, 1).Copy Range(.Offset(1, 1), .Offset(Term - 1, 1))
    .Offset(0, 2).Copy Range(.Offset(1, 2), .Offset(Term - 1, 2))
    .Offset(0, 3).Copy Range(.Offset(1, 3), .Offset(Term - 1, 3))

    .Offset(0, 5).Copy Range(.Offset(1, 5), .Offset(Term - 1, 5))
End With

'Add Extra Payment
row = Term - Range("b18") + 3

For NumExPay = 1 To Range("b19").Value

    Cells(row, 8).Value = Range("b17")
    row = row + 1
Next

Term = Term + 2
```

The next part of the code uses the inputs of extra payment, remaining life of the loan, and number of months the extra payment is used to create a new amortization schedule. The first for loop places the number of extra payments in the extra payment column. The next for loop finds where the first loan amount on the right hand column equals zero. It then makes the last payment (column E) equal to the remaining loan amount plus the interest due to make balance out the total loan amount to zero. The last for loop clears out the remaining contents that have a negative balance. Then the number of months the loan is reduced is found by subtracting the new term from the original term. Also the code uses the sum function to sum the total interest paid with the extra payments and places that function in cell b21. Then lastly the interest paid with the extra payments (cell(b21)) is compared to the interest paid without extra payments (cell(b13)). This is all shown in the code below.



```
'Add Extra Payment
row = Term - Range("b18") + 3

For NumExPay = 1 To Range("b19").Value

    Cells(row, 8).Value = Range("b17")
    row = row + 1
Next

Term = Term + 2

For row = 4 To Term
If Cells(row, 9).Value < 0 Then

   Cells(row, 5).Value = Cells(row, 6).Value + Cells(row - 1, 9).Value - Cells(row, 8).Value
End If

Next

For row = 4 To Term
    If Cells(row, 9).Value = 0 Then

    Range(Cells(row + 1, 4), Cells(row + 1, 4).Offset(0, 5).End(xlDown)).ClearContents
    End If
Next
RowCount = Range(Range("d3"), Range("d3").End(xlDown)).Rows.Count

Range("b24").Value = Range("b6").Value - RowCount

                                'Sum Totals
Range("b21").Formula = "=sum(" & Range(Range("f3"), Range("f3").Offset(0, 0).End(xlDown)).Address(False, False) & ")"

End Sub
Sub test()
```

Learning

This project gave me the opportunity to learn many new things. One of the most valuable lessons I learned is that I can teach myself VBA code by proactively researching things on my own and by trial and error. I also learned that VBA code can be used to make long, tedious tasks quick and easy.

For this project, I got most of my help from looking up bits of code from both the internet and the book. I did not get much help from other resources like friends or classmates. The book was very helpful in my process. I was able to look up how to write an example sensitivity code and then modify it to suit my needs. Looking up the answers on my own was frustrating at times but taught me the important lesson that I can teach myself. This skill will come in handy later in my life as I start my career.

I also learned that VBA takes lots of effort and time through trial and error. The more I worked at the code the more naturally the logic came to me and the more efficient I became. Tasks that were very difficult for me at the beginning of the project like counting the number of rows in a range, or setting a range to match the amount of data on a worksheet became easier at the end of the project. There is no substitute for trial and error when trying to learn VBA code.

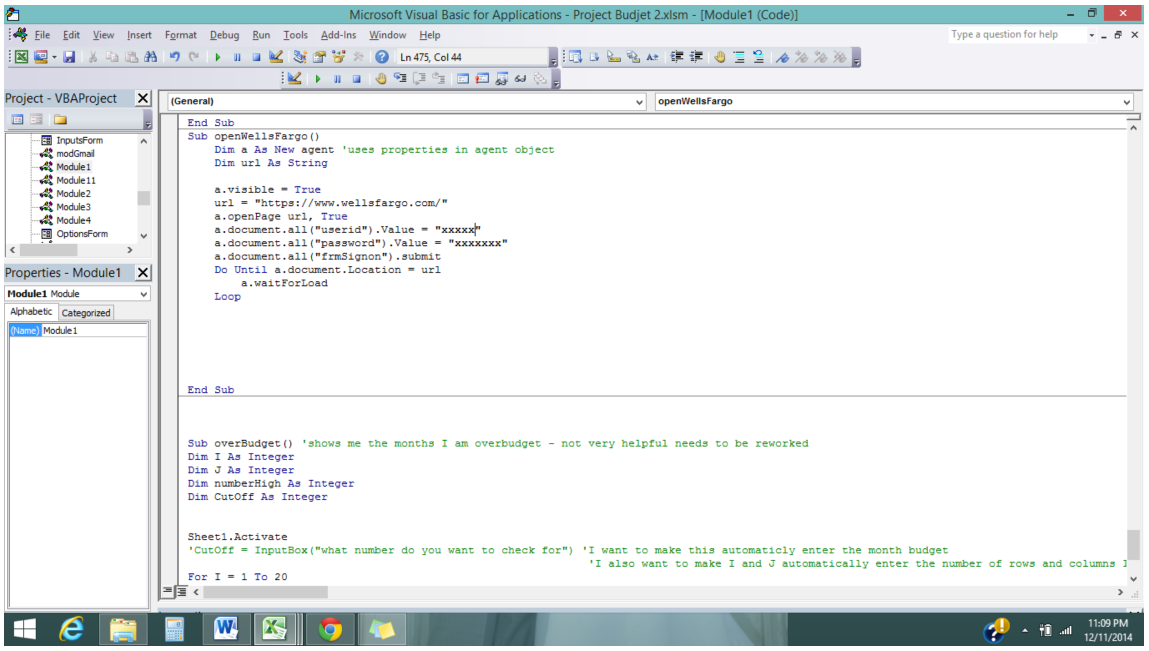
The last thing I learned is that VBA can be used in real life situations to make long and frustrating tasks quick and easy. For example, while I was working on the project I wanted to zoom each sheet in my workbook to 30%. Rather than go sheet by sheet and manually do it, I quickly wrote a VBA code that looped through each sheet and did the task for me.

As I mentioned previously, the task of categorizing my expenses each month took two or three hours and now I can do it in the click of a button. This budget is something that I will use the rest of my life because of how easy it is. I will also be motivated to continue to learn VBA because I saw just how useful it can be in a real life situation.

Difficulties

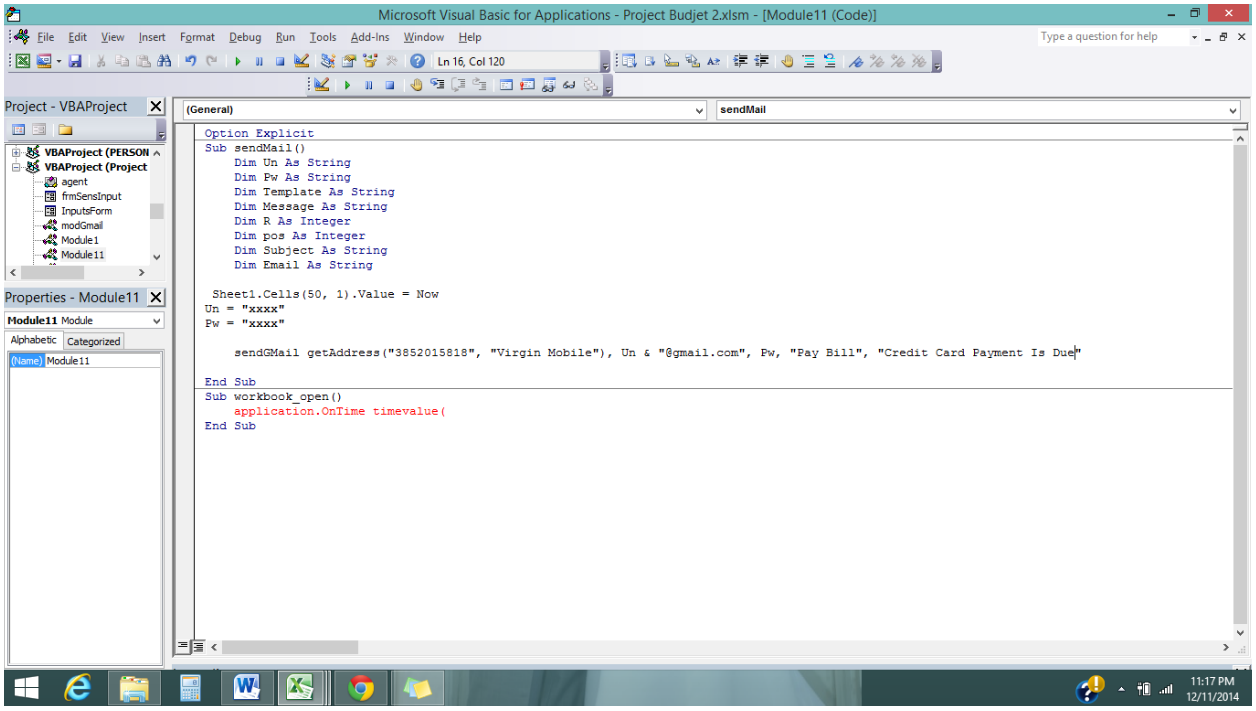
I faced several difficulties throughout this project. I wanted to accomplish a few things with this budget that I wasn't able to because I lacked the ability to write the code. Another problem I ran into was working on something just to realize that I wanted to go a different way with the project.

I initially set out to download my credit card transactions from the website using VBA code. I was able to log on to the website with my username and password but could not navigate the site well enough to get to the area where I could download the transaction report. The following is a screenshot of the code I used to log on to my bank account.



```
End Sub
Sub openWellsFargo()
    Dim a As New agent 'uses properties in agent object
    Dim url As String

    a.visible = True
    url = "https://www.wellsfargo.com/"
    a.openPage url, True
    a.document.all("userid").Value = "xxxxx"
    a.document.all("password").Value = "xxxxxxx"
    a.document.all("frmSignon").submit
    Do Until a.document.Location = url
        a.waitForLoad
    Loop




End Sub



Sub overBudget() 'shows me the months I am overbudget - not very helpful needs to be reworked
Dim I As Integer
Dim J As Integer
Dim numberHigh As Integer
Dim CutOff As Integer


Sheet1.Activate
'CutOff = InputBox("what number do you want to check for") 'I want to make this automaticly enter the month budget
                                                          'I also want to make I and J automatically enter the number of rows and columns I
For I = 1 To 20
```

I also wanted to send a text message to myself, reminding me when I need to pay credit cards and other bills. I could write the code that sends me the message; however, could not create the code to that would send me message on a specified day and time without me opening the workbook. Here is a screenshot of the code I wrote to send me a text message.



```
Option Explicit
Sub sendMail()
    Dim Un As String
    Dim Pw As String
    Dim Template As String
    Dim Message As String
    Dim R As Integer
    Dim pos As Integer
    Dim Subject As String
    Dim Email As String

 Sheet1.Cells(50, 1).Value = Now
Un = "xxxx"
Pw = "xxxx"

    sendGMail getAddress("3852015818", "Virgin Mobile"), Un & "@gmail.com", Pw, "Pay Bill", "Credit Card Payment Is Due"

End Sub
Sub workbook_open()
    application.OnTime timevalue(
End Sub
```

Along with getting stuck on certain aspects of code, I also spent a lot of time working on things that didn't ultimately make it in my final deliverable. For example, I spent a lot of time working on creating user input forms then deciding that I really didn't need the input forms. This happened a few times throughout the project, causing me to spend a lot of extra time working on it.

Overall, the project was a great learning experience for me and something that I will use for the rest of my life.