Joel Ribble
MBA 614
My Final Project
Oh Yeah

Project Write-Up

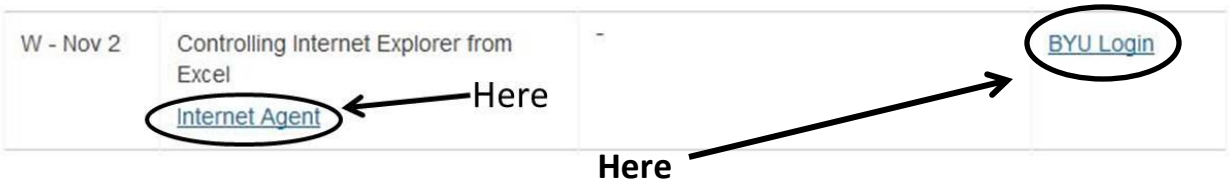
Executive Summary

Original Project Scope:
I am not very good at managing personal finances. I want to change that. I plan to create a program that pulls data from all of the banks I transact with in order to populate an ongoing personal budget and automate it such that I can either push a button to find out my monthly cash flow and/or have the excel sheet periodically update my budget on its own. I hope to see real time cash flow and time to pay off each long term debt that I have so that I can project the date and time at any given moment when I will break even and at which points in time I am worth what.

In retrospect:
I started this project convinced that I wouldn't finish it. As you know, I was quite behind in the course up until recently. And yet, I believe I have learned more about VBA, and programming at all, in the past week than I ever have before. This project was a great way to get comfortable with it. Here's how it happened…

Implementation Documentation

First I had to figure out how to gain access to the websites of the banks I transact with. I started that by importing the web agent that had been used in class:



Then, I took code from the BYU Login File and started to play around. Task one was to access my checking account and get the balance. Changing some of the values around in "BYU Login" got the job done pretty quick. After looking at a saved page however, I saw some opportunity to lessen the wait time by getting to a direct link. "wellsfargo.com" changed to "https://online.wellsfargo.com/signon" and I had to change the frmlogin,Label1.Caption name to "userid". I also made some of the code inactive including having my login credentials put right into the page instead of using the frmLogin.Show method. Here's the first part of that code:

```
Sub wellsLogIn()

  Dim un As String
  Dim pw As String
  Dim checking As String
```

```
Dim a As New agent


a.visible = False

a.openpage "https://online.wellsfargo.com/signon"

a.position = 1
'If Not a.moveTo("Log In Successful") Then
  ' user not logged in
 ' Load frmLogin
  frmLogin.Label1.Caption = "userid"
  'frmLogin.Show

  un = "**************" 'frmLogin.txtUsername.Text
  pw = "**************" 'frmLogin.txtPassword.Text


  a.explorer.document.all("password").Value = pw
  a.explorer.document.all("userid").Value = un
  a.explorer.document.all("continue").Click
  a.waitForLoad ' wait until the page is done loading
  a.savePage
```

Once logged in I looked at the newly saved page in order to get the value for my checking account balance. Here are the values from the saved page that I used:

```
s="cashTotal cashAvailableBalance">$1,234.69 </TD>
cashRelatedActivities> </TD></TR></TFOOT></TABLE><!
```

And here's how I integrated it into the code and had the value put onto a spreadsheet:

```
a.position = 1
  a.moveTo "cashTotal cashAvailableBalance"
  a.position = a.position + 2
  checking = a.getText("</TD>")
  Sheets("sheet1").range("c3").Value = checking
  range("c3").Activate
```

Then I added the following to format that value to the number style "accounting" by recording myself doing so and putting the code into the sub procedure:

```
Selection.NumberFormat = "_($* #,##0.00_);_($* (#,##0.00);_($* ""-""??_);_(@_)"
```

Next I needed to get my Credit Card balances. The first one, AAA Bank of America, was pretty tricky. Some of the code was in Java Script, specifically the action of clicking the "Sign On" Buttons. Furthermore, the Username and Password inputs were on 2 different pages each executed by a different Java Script function. Again, I made any code I didn't need or didn't want to use from the BYU Login File inactive.

Here is a snap shot of the username input on the website:

And here is what the source code behind it looks like:

```
<TD noWrap><!-- Begin Sign In Button --><NOSCRIPT></NOSCRIPT>
<SCRIPT language=Javascript1.1 type=text/javascript>
<!--

function signonForm_0_submit()
{
        document.signonForm.nextAction.value = 'signon';

        if(checkFormInputNSubmit()==false)
        {
        }
                return document.signonForm.submit();
}
getButton('Sign In', 'javascript:signonForm_0_submit();', '', '',
-->
</SCRIPT>
```

And the preceding snapshot will explain the following code I input into VBA to get to the password input:

```
Sub aaaLogIn()

  Dim un As String
  Dim pw As String
  Dim aaa As String
```

```
Dim a As New agent

a.visible = False

a.openpage "https://sitekey.bankofamerica.com/sas/signonScreen.do?reason=AAA"
a.savePage

a.position = 1
'If Not a.moveTo("Log In Successful") Then
  ' user not logged in
 ' Load frmLogin
  frmLogin.Label1.Caption = "onlineID"
  'frmLogin.Show

  un = "***********" 'frmLogin.txtUsername.Text
  pw = "***********" 'frmLogin.txtPassword.Text

  a.explorer.document.all("onlineID").Value = un
  'a.explorer.document.all("___signon").Click    'Use this one if javascript is disabled
  a.execScript "signonForm_0_submit()"
  a.waitForLoad ' wait until the page is done loading
  a.savePage
```

At this point a reviewed the most recent saved page to move forward again. Here is the password challenge on the website:

And here is the code behind it:

```
<DIV id=buttonJS><!-- Begin Sign In Button --><NOSCRIPT></NOSCRIPT>
<SCRIPT language=Javascript1.1 type=text/javascript>
<!--

function verifyImageForm_0_submit()
{
        document.verifyImageForm.nextAction.value = 'signon';

                return document.verifyImageForm.submit();
}
getImageButton('Sign In', 'javascript:verifyImageForm_0_submit();', '',' onclick="submit_form();
'signon', 'sign_in','sas-docs/en_US/images/sign_in.gif','sas-docs/en_US/images/sign_in_over.gif'
-->
</SCRIPT>
```

Reviewing the script yielded the following VBA code to move forward to my account info:

```
    a.explorer.document.all("passcode").Value = pw
    'a.explorer.document.all("___signon").Click    'Use this one if javascript is disabled
    a.execScript "verifyImageForm_0_submit()"
    a.waitForLoad ' wait until the page is done loading
    a.savePage
```

The area on the newly saved source page where my account info was located looked like this:

```
<DIV class=right-column-content>$4,473.77 </DIV>
```

Like before, the values surrounding the information I wanted was unique so I was able to use the following VBA code to get my debt balance on put in on my spreadsheet (in addition to formatting with "accounting" number style):

```
a.position = 1
    a.moveTo "<DIV class=right-column-content"
    a.position = a.position + 1
    aaa = a.getText("</DIV>")
    Sheets("sheet1").range("c5").Value = aaa
    range("c5").Activate
Selection.NumberFormat = "_($* #,##0.00_);_($* (#,##0.00);_($* ""-""??_);_(@_)"
```

The last info I needed came from my second Credit Card website. Again, it was a little unique from the previous two. What was different here was that after logging in there were many elements on that account summary page that took some time to individually low. This made it so that when I populated my spreadsheet with the value I'd gotten for my account balance, the value input was not always consistent. Sometimes it returned some href or other random line of code. With a little experimentation, I discovered that having the internet agent wait for the page to load in several iterations I was able to have my account balance consistently populate on my spreadsheet. Here's what that code looks like:

```
Sub wescomLogIn()

  Dim un As String
  Dim pw As String
  Dim wescom As String

  Dim a As New agent

  a.visible = False

  a.openpage "https://www.wescom.org/"

  a.position = 1
  'If Not a.moveTo("Log In Successful") Then
    ' user not logged in
```

```
    ' Load frmLogin
     frmLogin.Label1.Caption = "userid"
     'frmLogin.Show

     un = "*********" 'frmLogin.txtUsername.Text
     pw = "**********" 'frmLogin.txtPassword.Text

     a.explorer.document.all("txtPassword").Value = pw
     a.explorer.document.all("ctlUserName").Value = un
     a.explorer.document.all("Submit").Click
     a.waitForLoad ' wait until the page is done loading
     a.waitForLoad
     a.waitForLoad
     a.waitForLoad
     a.waitForLoad
     a.waitForLoad
     a.waitForLoad
     a.savePage

     a.position = 1
     a.moveTo "Visa Platinum"
     a.moveTo "TD align=right"
     a.position = a.position + 1
     wescom = a.getText("</TD>")
     Sheets("sheet1").range("c6").Value = wescom
     range("c6").Activate
Selection.NumberFormat = "_($* #,##0.00_);_($* (#,##0.00);_($* ""-""??_);_(@_)"
```

After I had gotten to the point where I could fetch all the bank data I needed, I began to set up my spreadsheet. I organized the data by “Equity “and “Debt”, had a cell total my credit card debt, and then added buttons to execute each individual sub procedure for obtaining balances, and then one more button to have all three procedures executed in succession.

Here is a snapshot of the spreadsheet up to that point:

	A	B	C	D
1	Get All			
2		Equity		
3	Get Checking	Wells Fargo Checking Account Balance	$ 1,234.69	
4		Debt		
5	Get AAA	AAA CC Balance	$ 4,473.77	
6	Get Wescom	Wescom CC Balance	$ 725.76	
7		Total	$ 5,199.53	
8				

And here is the sub procedure linked to the “Get All” button:

```
Sub GetBankData()

wellsLogIn
aaaLogIn
wescomLogIn

End Sub
```

Next I began to expand the spreadsheet interface by adding an area to put past results every time I pushed the “Get All” button and to indicate the time in which the data was pulled:

	A	B	C	D	E	F	G	H
1	Get All		As of:		Cash Balance History	Debt History	Net of Cash and Debt	As of:
2		Equity						
3	Get Checking	Wells Fargo Checking Account Balance	$ 1,234.69					
4		Debt						
5	Get AAA	AAA CC Balance	$ 4,473.77					
6	Get Wescom	Wescom CC Balance	$ 725.76					
7		Total	$ 5,199.53					
8								

Now after the hitting the “Get All” button a couple times, the new fields look like this:

C	D	E	F	G	H
As of:		Cash Balance History	Debt History	Net of Cash and Debt	As of:
12/8/2011 21:39		$ 1,234.69	$ 5,200.00	$ (3,965.31)	12/8/2011 16:41
$ 1,234.69		$ 1,234.69	$ 5,199.53	$ (3,964.84)	12/8/2011 19:59
$ 4,473.77					
$ 725.76					
$ 5,199.53					

In order to do this, I wrote sub procedures to get the time, and to copy and re-paste the new data under the headings by recording myself doing it as well as posting my “net worth” snap shot. Here the code for each procedure:

```
Sub getTime()
   range("C2").Select
   ActiveCell.FormulaR1C1 = "=NOW()"
End Sub

Sub copyCash()

   range("c3").Copy
   range("e1").Select
   Selection.End(xlDown).Select
   Selection.End(xlDown).Select
   Selection.End(xlUp).Select
   ActiveCell.Offset(1, 0).Select
   ActiveSheet.Paste
```

```
    Application.CutCopyMode = False
    Selection.NumberFormat = "_($* #,##0.00_);_($* (#,##0.00);_($* ""-""??_);_(@_)"
End Sub

Sub copyDebt()

    range("c7").Copy
    range("f1").Select
    Selection.End(xlDown).Select
    Selection.End(xlDown).Select
    Selection.End(xlUp).Select
    ActiveCell.Offset(1, 0).Select
    Selection.PasteSpecial Paste:=xlPasteValues
    Application.CutCopyMode = False
    Selection.NumberFormat = "_($* #,##0.00_);_($* (#,##0.00);_($* ""-""??_);_(@_)"
End Sub

Sub copyNow()

    range("c2").Copy
    range("h1").Select
    Selection.End(xlDown).Select
    Selection.End(xlDown).Select
    Selection.End(xlUp).Select
    ActiveCell.Offset(1, 0).Select
    Selection.PasteSpecial Paste:=xlPasteValues
    Application.CutCopyMode = False
End Sub

Sub net()

    range("g1").Select
    Selection.End(xlDown).Select
    Selection.End(xlDown).Select
    Selection.End(xlUp).Select
    ActiveCell.Offset(1, 0).Select
    ActiveCell.FormulaR1C1 = "=RC[-2]-RC[-1]"
    Selection.NumberFormat = "_($* #,##0.00_);_($* (#,##0.00);_($* ""-""??_);_(@_)"

End Sub
```

To figure the time to pay off each credit card and project the month at which I'd be debt free was no easy task for me. I added the following fields and to work with:

9	Date of Pay Off*:		
10	Months to Pay Off*:		
11	*Assuming Minimum Payments and		
12	Debt Pay Off Schedule:	AAA CC	Wescom CC

I wanted cascading values to populate downward for each monthly credit card balance and then to measure the number of cells under the headings to get the date and count of months until pay off. First I recorded my doing the .xldown method and tried to manipulate it:

```
'range("b14").Select

'Selection.AutoFill destination:=range(Selection, Selection.End(xlDown)), Type:=xlFillDefault
'
'End

'range("c14").Select

'Selection.AutoFill destination:=range(Selection, Selection.End(xlDown)), Type:=xlFillDefault
'
'End
```

This was a complete failure. After a desperate phone call to Professor Gove, I got help in the form of a procedure I could start from to get each month to be calculated off the month before it until the value was less than 0. I wrote =If formulas in the cells just below the headings in order to calculate balances after interest and minimum payments and then had other procedures capture the “date of pay off” and “Months to pay off”. Here’s that code:

```
Sub fillErUp()

 range("c14").Formula = "=IF(((C13*0.39)/12)>25,(C13+((C13*0.26)/12))-
((C13*0.39)/12),(C13+((C13*0.26)/12))-25)"

  fillDownToZero range("c14")

  range("b14").Formula = "=IF(((B13*0.39)/12)>25,(B13+((B13*0.26)/12))-
((B13*0.39)/12),(B13+((B13*0.26)/12))-25)"

  fillDownToZero range("b14")

  ActiveCell.Offset(1, -1).Select
  range("b9").Value = Selection

End Sub

*the code immediately below is compliments of Gove Allen*

Sub fillDownToZero(StartCell As range)
   Dim theCell As range
   Set theCell = StartCell

   Do Until theCell.Value < 0
     theCell.Select
     Debug.Print theCell.Address, theCell.Value, theCell.FormulaR1C1
```

```
    theCell.Offset(1, 0).FormulaR1C1 = theCell.FormulaR1C1
    Set theCell = theCell.Offset(1, 0)
  Loop

End Sub
```

This procedure counted cells under the higher balance credit card to find months to pay off:

```
Sub payOffMonthCount()

   'range("b13").Select
   'Selection.End(xlDown).Select
   'range("b9").Value = Selection


   range("B13").Select
   range(Selection, Selection.End(xlDown)).Select
   range("b10").Value = Selection.Cells.Count
   range("B10").Select

End Sub
```

The only other things I put in were a procedure that aligns the C column to the right:

```
Sub alignCollumnC()
'
' alignCollumnC Macro
'

'
   Columns("C:C").Select
   With Selection
      .HorizontalAlignment = xlRight
      .VerticalAlignment = xlBottom
      .Orientation = 0
      .AddIndent = False
      .IndentLevel = 0
      .ShrinkToFit = False
      .ReadingOrder = xlContext
      .MergeCells = False
   End With
   'Selection.NumberFormat = "_($* #,##0.00_);_($* (#,##0.00);_($* ""-""??_);_(@_)"
   'range("B1").Select
End Sub
```

And a method to autofit the collumns that the balance data go in:

```
Columns("B:H").EntireColumn.AutoFit
```

The master sub procedure then looks like this:

```
Sub GetBankData()
'
' GetBankData Macro
'

wellsLogIn
aaaLogIn
wescomLogIn
getTime
alignCollumnC
copyCash
copyDebt
copyNow
net
Columns("B:H").EntireColumn.AutoFit
fillErUp
payOffMonthCount
MsgBox "All Done"

'
End Sub
```

The final spreadsheet looks like this (after executing the master “Get All & Update” button a couple times):

	A	B	C	D	E	F	G	H
1	Get All & Update		As of:		Cash Balance History	Debt History	Net of Cash and Debt	As of:
2		Equity	12/8/2011 22:10		$ 1,234.69	$ 5,200.00	$ (3,965.31)	12/8/2011 16:41
3	Get Checking	Wells Fargo Checking Account Balance	$ 1,234.69		$ 1,234.69	$ 5,199.53	$ (3,964.84)	12/8/2011 19:59
4		Debt						
5	Get AAA	AAA CC Balance	$ 4,473.77					
6	Get Wescom	Wescom CC Balance	$ 725.76					
7		Total	$ 5,199.53					
8								
9	Date of Pay Off*:	8/8/2029						
10	Months to Pay Off*:	213						
11	*Assuming Minimum Payments							
12	Debt Pay Off Schedule:	AAA CC	Wescom CC					
13	Dec-11	$ 4,425.30	$ 716.48					
14	Jan-12	$ 4,377.36	$ 707.01					
15	Feb-12	$ 4,329.94	$ 697.33					
16	Mar-12	$ 4,283.03	$ 687.44					
17	Apr-12	$ 4,236.63	$ 677.33					
18	May-12	$ 4,190.74	$ 667.01					
19	Jun-12	$ 4,145.34	$ 656.46					
20	Jul-12	$ 4,100.43	$ 645.68					
21	Aug-12	$ 4,056.01	$ 634.67					

Discussion of Learning

This may not be the most complicated set of procedures but I am very proud of the fact that I was able to learn to do it and was able to get each element of the original project scope to work. I had my doubts about being able to figure it out but again, I learned more about VBA doing this than I thought I could.

I learned what to do when getting stuck (message boards online, trying what I know in various ways until it works, recording myself doing things and then updating the code). I learned that you can do almost anything in VBA (at least it seems that way) with patience. I learned that I have way more to learn after this class but at least I know how to learn it.