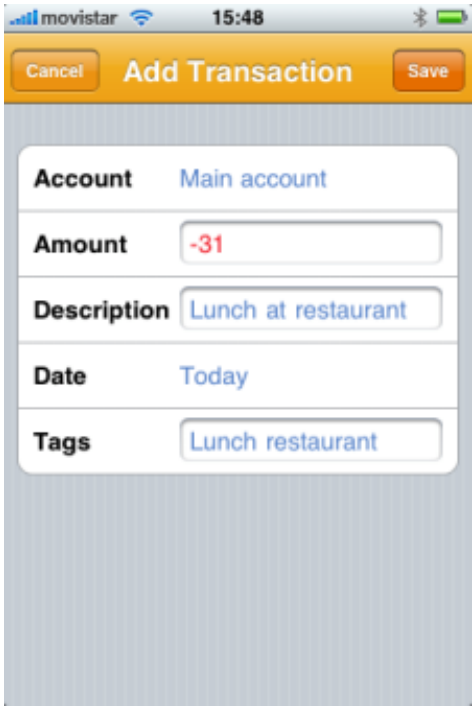
Jordan Beasley
ISYS 520/Dr. Gove Allen
Final Project Write-up
December 7, 2010

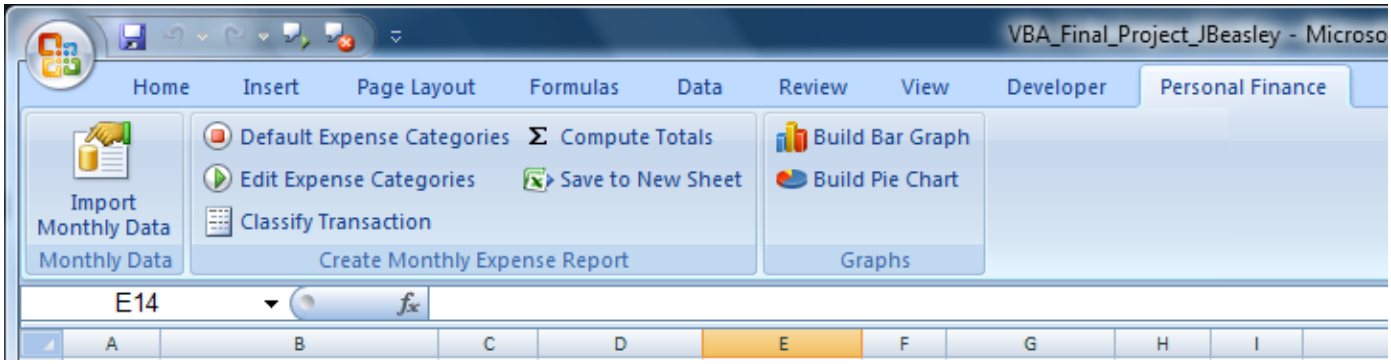
Executive Summary

Moneytrackin is a self-described "free online tool to regain control of your financial life." Through the organization's website at www.moneytrackin.com, users can access a simple but elegant piece of software that allows for the tracking of individual inflows and outflows of cash. Moneytrackin also offers a free iPhone app, which I have been using to track all my income and expenses since September of 2008. Unlike popular sites like www.mint.com, Moneytrackin is entirely user-controlled and does not link to bank accounts. I enjoy this feature for two reasons: (1) it gives me the ability to specifically track cash-based transactions, and (2) it gives me the power to judgmentally classify each transaction.

The downside of this style of budgeting tool is the significant investment in time and labor. Each month for the past two years, I have had to spend approximately 30 minutes a month to manually transfer my data from Moneytrackin into a spreadsheet, classify each transaction into an expense category, and format the data. I have written a program that will request a date from the user and automatically download that month's list of transactions to Excel. The program then formats the transaction data and allows the user to classify each transaction into a set of specified expense categories (a set of default expense categories is also provided). My program also allows the user to build a bar graph or a pie chart from the data, in order to see how money was allocated among different expense categories.



Implementation Documentation

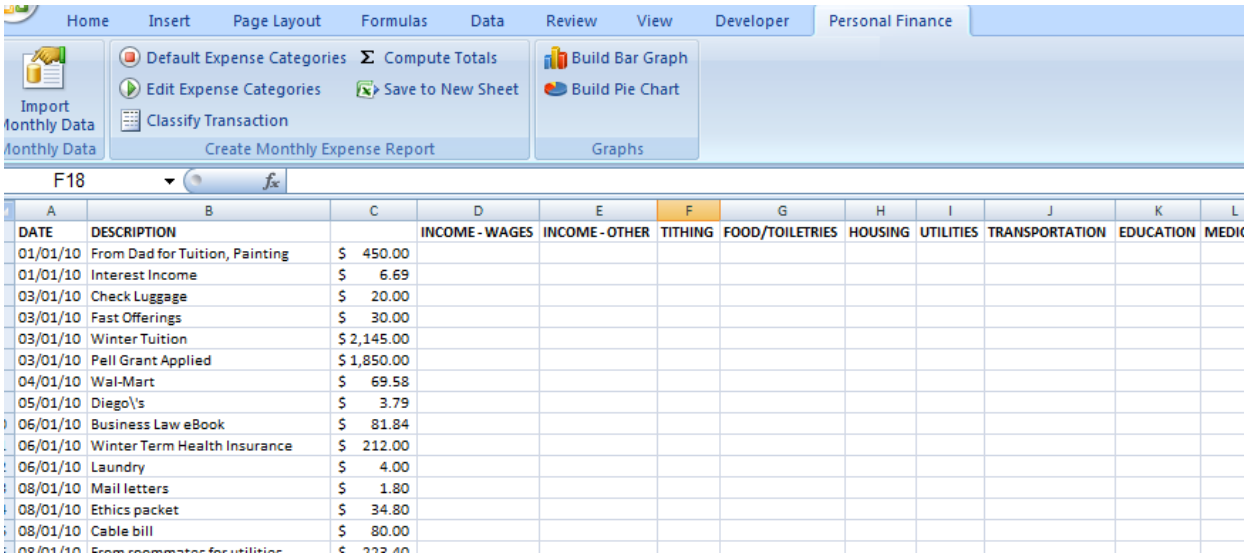


Ribbon Modification

To guide the user through my program, I created a new tab in the Excel ribbon (as shown above). As you can see, this new "Personal Finance" tab is composed of eight buttons split among three different groups. For purposes of organization, I will divide my implementation documentation into eights sections, based on each of these eight buttons.

Import Monthly Data

When users click this button, they are greeted with an input box asking them to enter a date. If a valid date is entered, the program will log into www.moneytrackin.com (using a username and password specified in the code) and download all the transaction data from the month of the inputted date. The format of the downloaded data is messy, so the program opens a new sheet and formats the transactions in a pleasing manner. Date, description, and amount are included, along with a list of default expense categories.



DATE	DESCRIPTION		INCOME - WAGES	INCOME - OTHER	TITHING	FOOD/TOILETRIES	HOUSING	UTILITIES	TRANSPORTATION	EDUCATION	MEDI
01/01/10	From Dad for Tuition, Painting	$ 450.00									
01/01/10	Interest Income	$ 6.69									
03/01/10	Check Luggage	$ 20.00									
03/01/10	Fast Offerings	$ 30.00									
03/01/10	Winter Tuition	$ 2,145.00									
03/01/10	Pell Grant Applied	$ 1,850.00									
04/01/10	Wal-Mart	$ 69.58									
05/01/10	Diego\'s	$ 3.79									
06/01/10	Business Law eBook	$ 81.84									
06/01/10	Winter Term Health Insurance	$ 212.00									
06/01/10	Laundry	$ 4.00									
08/01/10	Mail letters	$ 1.80									
08/01/10	Ethics packet	$ 34.80									
08/01/10	Cable bill	$ 80.00									

```
inputData = InputBox("Please enter target date for expense report.  Use the following format: MM/DD/YYYY.", "Enter Date")


If inputData = "" Then Exit Sub
If Not IsDate(inputData) Then
  MsgBox "Please enter a valid date.", vbInformation, "No Valid Date"
  Exit Sub
End If

targetDate = inputData

key = DateDiff("m", targetDate, Date)
login2

End Sub
```

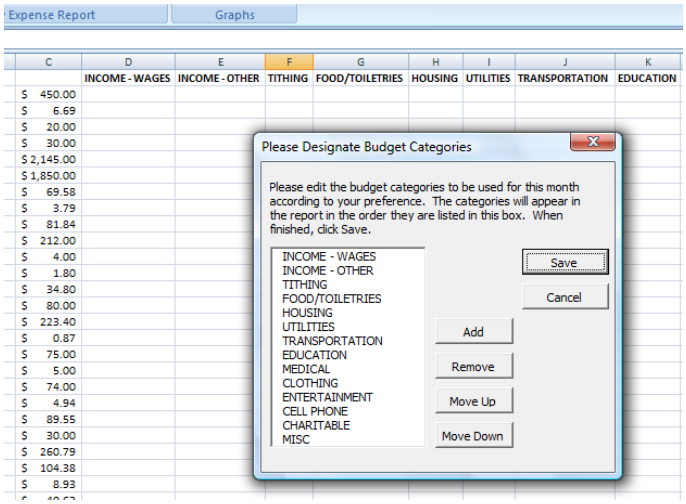
```
Public Sub login2()

agent1.visible = True
agent1.openpage "http://www.moneytrackin.com"
agent1.savePage
agent1.explorer.document.all("user").Value = "jordanbeas"
agent1.explorer.document.all("pass").Value = ""
'agent1.explorer.document.all("fm1").submit
agent1.explorer.document.all("signin").Click
agent1.waitForLoad
If key = 0 Then
agent1.openpage "https://www.moneytrackin.com/accounting/transactions/"
ElseIf key > 0 Then agent1.openpage "https://www.moneytrackin.com/accounting/transactions/?offset=-" & key & "&project=NULL"
End If
deleteSheet "Moneytrackin Data"
agent1.importPage "Moneytrackin Data"

formatMoneytrackinData

End Sub
```

Default Expense Categories

A set of suggested expense categories has been programmed into the code. If at any point users wish to return the monthly report to the default categories, they can click this button. Any existing set of categories will be deleted and replaced with the default categories.

Edit Expense Categories

Users will click this button if they wish to add, remove, or change the expense categories from the defaults. When the button is clicked, the following user form will appear:

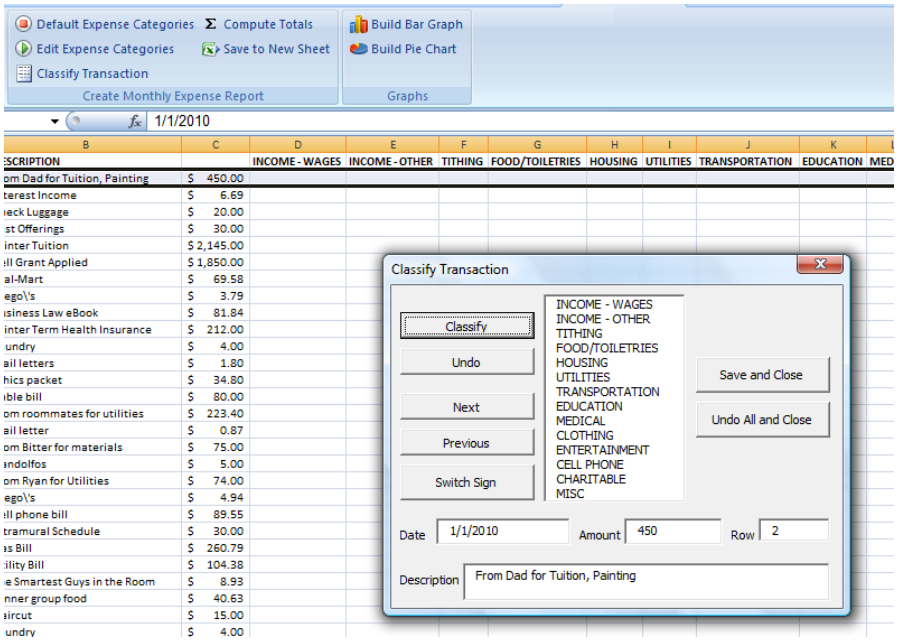


Not only can users add or remove expense categories with this form, but they can change the order in which the categories will appear in the spreadsheet. An excerpt of the code is found below.

```
Private Sub cmdMoveUp_Click()

Dim x As Long
Dim tempHolding3 As String
Dim tempHolding4 As String

x = lstCategories.ListCount - 1

Do
    If frmCategories.lstCategories.Selected(x) = True And x > 0 Then

    tempHolding3 = lstCategories.List(x)
    tempHolding4 = lstCategories.List(x - 1)
    lstCategories.List(x) = lstCategories.List(x - 1)
    lstCategories.List(x - 1) = tempHolding3
    lstCategories.Selected(x - 1) = True

    Exit Sub

    ElseIf x > 0 Then

        If frmCategories.lstCategories.Selected(x) = False Then

        x = x - 1

        ElseIf frmCategories.lstCategories.Selected(x) = True Then

        MsgBox "The category is already at the bottom of the list.", vbInformation, "Error"

        End If

    End If

Loop Until x = 0

MsgBox "Either no category is selected or the selection is already at the top of the list.", vbInformation, "Error"
```

Classify Transaction

Users will click this button when they are ready to begin classifying their transactions for the month. The program takes the transaction data from the active row and populates a user form (shown below). Users can then use this form to designate an expense category for the transaction. The form also offers functionality to undo a classification, move between rows in the spreadsheet, and change the sign on the transaction amount. The “change of sign” functionality was written in order to handle cash inflows that are not really income but rather offset previously incurred transactions (e.g. roommates chipping in to help pay a utility bill). An excerpt of the code written to classify transactions is also included below.



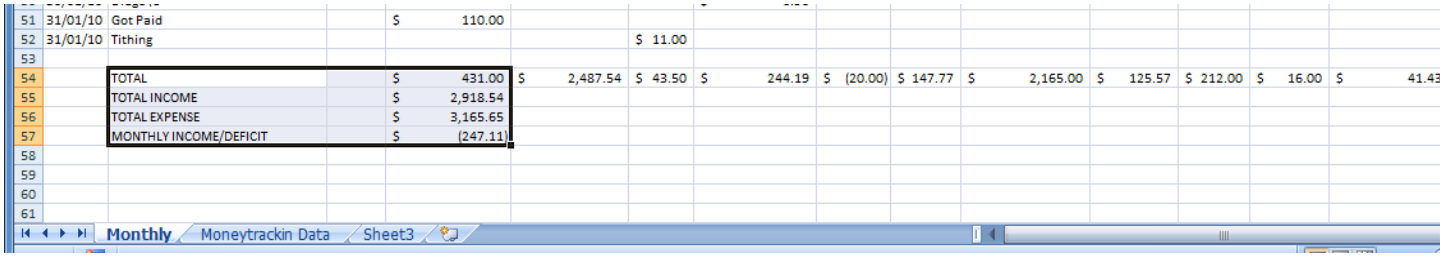

```
Public Sub classifyFirstTime()

Dim x As Long
Dim r As Long

r = frmClassify.txtRow.Text
x = 0

Do

    If frmClassify.lstCategories2.Selected(x) = True Then

        findAmount
        Cells(r, 3 + x + 1).Select
        ActiveSheet.Paste

        Exit Sub

    ElseIf frmClassify.lstCategories2.Selected(x) = False Then

        x = x + 1

    End If

Loop Until x = frmClassify.lstCategories2.ListCount

MsgBox "Please select a budget category to which this transaction will be assigned.", vbInformation, "No category selected"


End Sub
```

```
Public Sub findAmount()

Dim x As Long
Dim r As Long
```

Compute Totals

When users have finished classifying each of the month's transactions into one category, they will click this button to calculate the month's totals (if one or more transactions have not been classified, an error message will be displayed). Clicking this button will also calculate the month's total income, total

expenses, and overall income/deficit. In order to accomplish this, an input box will ask the user to specify how many income categories are present. This calculation will only work properly if all the income categories are arranged to the left of any expense categories. I realize that this is not a very elegant solution, and given more time I'm sure I could have arranged for a more sophisticated system in which each category is tagged with a designation as either "expense" or "income." As presently written, the code will work just fine as long as the user remembers to order any and all income categories before any expense categories.



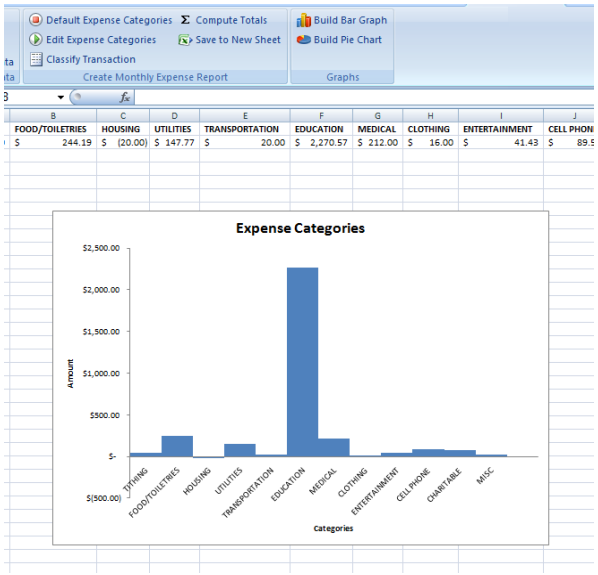

```
Selection.Offset(3, 0).Value = "MONTHLY INCOME/DEFICIT"

c = 2
x = Selection.Row - 1
Selection.Offset(0, c).Formula = "=SUM(D2:D" & x & ")"
Selection.Offset(0, c).Select

Do

    Selection.copy
    Selection.Offset(0, 1).Select
    ActiveSheet.Paste

    c = c + 1

Loop Until Selection.Offset(-x, 0) = ""

Columns("A:Z").EntireColumn.AutoFit
Selection.Delete

z = InputBox("How many income categories are included at the left of your report?", "A Quick Question")

l = Chr((4 Mod 26) + 64)
m = Chr(((4 + (z - 1)) Mod 26) + 64)
n = Chr(((4 + z) Mod 26) + 64)

Cells(Selection.Row + 1, 4).Formula = "=SUM(" & l & x + 1 & " : " & m & x + 1 & ")"
Cells(Selection.Row + 2, 4).Formula = "=SUM(" & n & x + 1 & ": Z" & x + 1 & ")"
Cells(Selection.Row + 3, 4).Formula = "=D" & Selection.Row + 1 & " - D" & Selection.Row + 2 & ""
Columns("A:Z").EntireColumn.AutoFit

Cells(Selection.Row + 3, 4).Select

MsgBox "Congratulations! Your monthly totals have been computed.", vbInformation, "Totals Computed"


End Sub
```

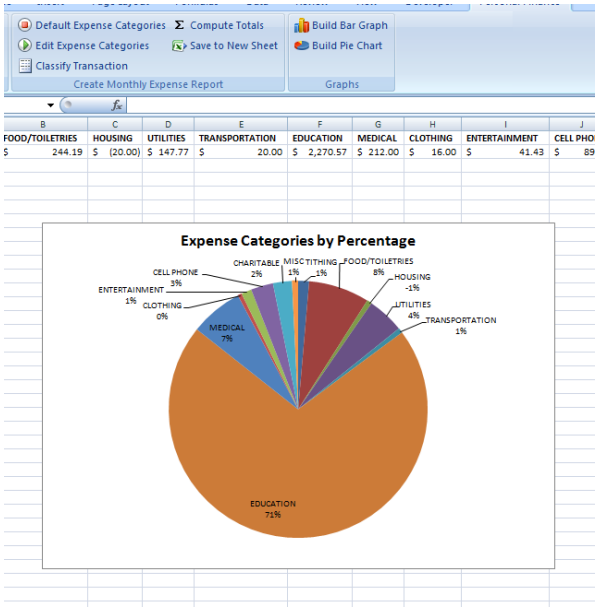
Save to New Sheet

Clicking this button will prompt users to assign a name to the new sheet. The program will then make a copy of the "Monthly" sheet and copy it to a new worksheet of the designated name. This is an important step because if users perform additional queries then the "Monthly" sheet will be deleted and replaced with a new sheet.

Build Bar Graph/Build Pie Chart

Clicking this button will open a new worksheet and provide users with either a bar graph or a pie chart, showing the amount they spent for the month on each expense category.





```
Cells.find(What:="Description", After:=ActiveCell, LookIn:=xlFormulas, _
        LookAt:=xlPart, SearchOrder:=xlByRows, SearchDirection:=xlNext, _
        MatchCase:=False, SearchFormat:=False).Activate

 ActiveCell.Offset(0, 1 + z).Activate

 c = 1

    Do
        found = False

        If Len(ActiveCell.Offset(0, c).Value) > 0 Then

            ActiveCell.Offset(0, c).Select

            found = True

            Exit Do

        ElseIf ActiveCell.Offset(0, c).Value = "" Then

            c = c + 1

        End If

    Loop Until c = 50

    If found = False Then MsgBox "Expense Categories Needed to Build Graph"


    Range(Selection.Offset(0, -1), Selection.End(xlToRight)).Select
    Selection.copy
    Sheets("Monthly").Select
    Sheets.Add
    ActiveSheet.Name = "Pie Chart"
    ActiveSheet.Paste
```

Learning and Conceptual Difficulties Encountered

I learned many things and encountered many challenges during the many hours I spent on this project, linked by the main themes included in the following list:

- All good programs are user-friendly. Although it took much longer to modify my ribbon and design good user forms, I found that these types of measures were critical to making my experience with using the program not only stress-free, but fast and efficient.
- Importing the correct web page from within a password-protected web site. After getting help from Dr. Allen on importing from behind a password-protected site, I ran into an obstacle when

I noticed that the simple query I had written would only import the current month's data. Eventually I noticed that a certain part of the URL would "count up" by one for each month earlier than the current month. I solved this problem by requesting a date from the user with an input box and then using the DateDiff function in VBA to calculate the number of months between the inputted date and the current date. This number could then be plugged into the URL to import the appropriate page.

- Planning for varying sizes of each month's report. I encountered many challenges as I was writing my code knowing that not only would each month's data set have a unique number of rows, but that each data set would have an unknown number of columns (based on the number of expense categories specified by the user). I solved this problem by using loops and methods like xlDown, xlUP, xlRight, and Offset.
- Returning the letter of a column based on its number. I became familiar with the "CHR" function when trying to insert a SUM function into cells based on the number of the cell's column.
- Changing the order of items in a user form's list box. To solve this problem, I ended up creating a "holding variable" for the values of the list box. When the "Move Up" or "Move Down" button is clicked, the value of the destination spot gets assigned to the temporary variable and then replaced by the value to be moved. The value is then reassigned out of the temporary variable to the moving value's old spot in the list box.

Conclusion

Overall, this challenging project was an enjoyable way to cement some basics of the VBA language into my mind and create a tool that I will personally use to help track my expenses. I incorporated many of the techniques taught throughout the semester, including loops, IF statements, web queries, user forms, and ribbon modification. I feel that I succeeded in writing all the functionality into the program that I personally would require.