MBA 614 – VBA Final Project

Kaden Feller

Executive summary

Jed the owner of Polly Auto is a good friend of mine. Jed started the business a few years ago and has worked hard to make it successful. Polly auto is a buy-here pay-hear used car lot that specializes in cars that are usually less than $10,000. Jed finances the cars that he sells and has a portfolio of loans outstanding. Sometimes customers stop paying and after repeated attempts to contact them if they are not willing to pay Jed has to repossess the vehicle. After repossession he will either work out new terms with the customer or he will resale the car. The margins are quite high because of the high default rate and all of the costs associated with financing people who usually cannot get financed through traditional methods. Polly Auto gets the majority of its inventory from the Auto Auctions; however, Jed is always on the lookout for good deals on used cars. Jed often looks on the local classifieds to try and find vehicles that he can buy and then sell on his lot. This process can be very time consuming and we have talked about how nice it would be to have a computer program that automatically searched the web and notified you of potentially good deals. As I started taking VBA Jed's problem came to mind and I realized that I could potentially help solve his problem using VBA. I have built a program in excel that does what Jed needs. The program I developed takes parameters about a specific type of vehicle and goes online to the local classifieds and pulls data from cars fitting those parameters. It then pulls market price data about each vehicle using the VIN and compares that data do the asking price. If the car is considered a good deal an email is sent to the user notifying him of the potential good deal and includes the link to the listing.

Implementation documentation

I used the following concepts and others to complete my project:

- Using VBA to pull data from webpages
- Using excel to send email messages
- Working with arrays for multiple purposes including to store and to access data efficiently
- Implementing the "redim" statement to make arrays more dynamic
- Using loops to simplify repetitive tasks including for and do loops
- Passing variables between sub procedures
- Using functions to return a value
- Using user forms to collect data and use it in the procedure
- Using Len, Right, Mid, etc functions in VBA to work with and to pars text
- Writing data to and pulling data from worksheet cells

- Using public, and module level variables to access data a different levels of the procedure
- Loading combo boxes to create options for the user
- Declaring multiple types of variables for different tasks including Long, Integer, string, etc.
- Using VBA to format worksheets and cell ranges
- Using "offset", "xlDown", "fillDown", etc in order to accomplish tasks more efficiently

This is a list of many of the concepts that I used to complete my project. In the write up I go into detail about how this tools were used.

Learning and conceptual difficulties

I ran into a number of problems as I completed this project; however, I was able to resolve all of the issues I ran into and I was able to accomplish what I proposed earlier in the semester. The one problem that this project still has is that the procedure takes a long time to run because it has to access so many different web pages and wait for them to load. With additional time there are also a few areas where I could streamline some of the code and some of the procedures. For example right now every time the code runs it goes to the webpage to access the data to load the combo boxes in the user form. This takes up time waiting for the webpage to open and I could just hard code this data into arrays to speed up the process. I have learned a lot from this project and I anticipate using VBA throughout the rest of my career. One of the most important lessons that I learned during this process is that pretty much anything imaginable can be done using VBA it just takes the time and effort to figure it out.

Assistance

In the beginning of this project I received personal help from Professor Allen. Professor Allen helped me better understand how to use the agent that he provided us with. During a TA session I got help to understand how to use the Custom UI to modify the ribbon and add additional buttons. In the TA session I also got help with understanding how to download an entire webpage to a worksheet before parsing through the text.

Write up detail

I started off by trying to decide which classifieds webpage I would use to search for good deals. I considered KSL.com classifieds, Craig's List, and EBay. Each of these webpages has pros and cons but I eventually went with KSL.com classifieds. KSL.com classifieds ads all follow a very similar format which makes it easier to pull data using VBA. All of the KSL.com ads also seem to be quite thorough and most of the ads have VIN numbers included which is an important aspect of this project. Besides being well formatted and relatively easy to access I also know that Jed is already using KSL.com classifieds and he will be familiar with the listings. In addition to these

reasons after investigating scraping data from eBay I get the idea that it is illegal to access listings on eBay with a robot and I did not want to deal with any legal issues!!

After deciding which classifieds to used I had to figure out how to search for and pull the data from each listing. I went to ksl.com/auto/ to start the search and there are a number of parameters that can be used. I did not want to accommodate all of the options available because it would have made this project much too time consuming and Jed does not need all of these options for his needs. From around 30 different options I selected the ones that would be needed by Jed and either ignored or hard coded the other parameters. For example the user of my program can select the Make, Model, the range of years, and the area for the search. The program searches vehicles with odometers from 0-1,000,000 miles, all price ranges, and listings that are for sale by owner. The program ignores parameters such as how many cylinders the engine has, and the number of doors. I did this because Jed is just looking for cars that are selling for a good price. It also does not make sense to look at adds from other dealers and he doesn't care about a lot of the parameters as long as the car is selling for a good deal that he can take advantage of.

After deciding what parameters to use I wrote code using Professor Allen's agent code that accesses KSL.com classifieds and searches the parameters. I did this by using the URL from the search criteria and making it dynamic by replacing the actual values with placeholders. The user can specify the criteria he or she wants and the KSL.com search will automatically be updated (I will go into how the user updates these parameters later). Here is the URL before and after I replaced the values:

Before:

http://www.ksl.com/auto/search/index?keyword=&make%5B%5D=Ford&model%5B%5D=F-150&yearFrom=2008&yearTo=2016&mileageFrom=0&mileageTo=1000000&priceFrom=1&priceTo=1000000&zip=84606&miles=25&newUsed%5B%5D=All&page=0&sellerType=For+Sale+By+Owner&postedTime=&titleType=&body=&transmission=&cylinders=&liters=&fuel=&drive=&numberDoors=&exteriorCondition=&interiorCondition=&cx_navSource=hp_search&search.x=39&search.y=12&search=Search+raquo%3B

After (I have highlighted the parameters that have been replaced):

http://www.ksl.com/auto/search/index?o_facetClicked=true&o_facetValue=2008%2C2016&o_facetKey=yearFrom%2C+yearTo&resetPage=true&keyword=&make%5B%5D=" & make & "&model%5B%5D=" & model & "&yearFrom=" & yearFrom & "&yearTo=" & yearTo & "&priceFrom=" & priceFrom & "&priceTo=" & priceTo & "&mileageFrom=" & mileageFrom & "&mileageTo=" & mileageTo & "&zip=" & zip & "&miles=" & searchRadius & "&sellerType%5B%5D=For+Sale+By+Owner

I declared a module level variable called “webpage” and wrote a sub that collects the parameters writes that into the webpage and sets it equal to “webpage”. This allows me to use “webpage” anywhere in the module to go back to the search results.

Almost immediately I realized a problem. When the search is run the classifieds are all presented in a list form but it is not possible to pull all of the data that I need without going to each of the individual listings. In order to pull all of the data that I needed I had to find a way to search each listing link and pull the data. Once I had followed that link I would need to return to the original search results and go to the next link and on and on until I had pulled the data from all of the listings. This represented a number of issues. First of all I had to find a way to identify each individual link for each listing and only follow it once, second when I returned to the original search results I had to tell the agent at what position I was in so that it could continue the search. The first way I tried to solve this problem is by recording the agent position before following a link and then telling the agent to go back to that same position after returning to the search results. This worked to a degree but the result was often unpredictable and it often followed the wrong link. I then decided to use an array to solve the problem. I wrote a sub called “loadItemNumbers” that goes to the search results webpage and finds out how many listings there are and then finds and writes all of the listing numbers to an array called “itemNumbers”. “itemnumbers” is a module level array so that once it contains the item numbers I can use it anywhere in the module.

```
Sub loadItemNumbers()
Dim a As New agent

Dim i As Integer

  a.visible = True
  a.openpage webpage, True
  a.position = 1

    a.moveTo ("total_listings"">")
    numberOfListings = a.getText("&")
    ReDim itemNumbers(numberOfListings - 1)

    For i = 0 To (numberOfListings - 1)
        a.moveTo ("#srp_listing_description_")
        itemNumbers(i) = a.getText("'))")
    Next

End Sub
```

Once I have the array of item numbers I can use it to more easily find the links and follow them and I don’t have to keep track of what position the agent is in. I used the agent “followLinkByHref” sub to accomplish this. Below is the code that demonstrates how I combined the root URL with the listing number and then followed the link. The code also shows how I used the number of listings to make my for-loop dynamic based on different search results.

```
For x = 0 To numberOfListings - 1

link = "http://www.ksl.com/auto/listing/" & itemNumbers(x)

  a.followLinkByHref link
```

Once I followed each link I wanted to pull certain data and write it to an excel sheet. The sub searches certain criteria such as the title, price, and year and writes it to the Outputs sheet of the open workbook. It also includes a for-loop that uses a preloaded array (called "specs") to pull data from a table in each link and write it to the sheet.

```
counter = 0
For Each element In specs
  item = specs(counter)
  data = getSpec(specs(counter), a)
  writeData data, item
  counter = counter + 1
Next element
```

The function "getSpec" is the procedure that actually pulls the data from the listing and returns it as a string variable. This function tests the data so that if the value is left blank or not specified it returns "N/A" instead of returning gibberish:

```
Function getSpec(item As String, a As agent) As String
Dim data As String

    a.moveTo (item)
    a.moveTo ("<td>")

    data = a.getText("</td>")

    If InStr(1, data, "not specified", vbTextCompare) > 0 Then
      data = "N/A"
      ElseIf Left(data, 2) = "<a" Then
      data = "pollywogs"
    End If

    getSpec = data

End Function
```

I wrote a sub called "writeData" that writes each item of data to the sheet. I pass two variables to the sub "writeData" which are "data" and "item". "item" is the name of the variable such as Title, Price, Mileage, VIN, etc, and "data" is the information collected about that specific listing such as "2013 Ford F-150", "$35,000", "26,500", "1FTFW1EF3DKG29838", etc. The sub "writeData" finds the item in the sheet and writes the data in the next available line below it. To make this sub dynamic I had it go to the end of the column and then "xlup" to ensure that I would write data to the next empty cell.

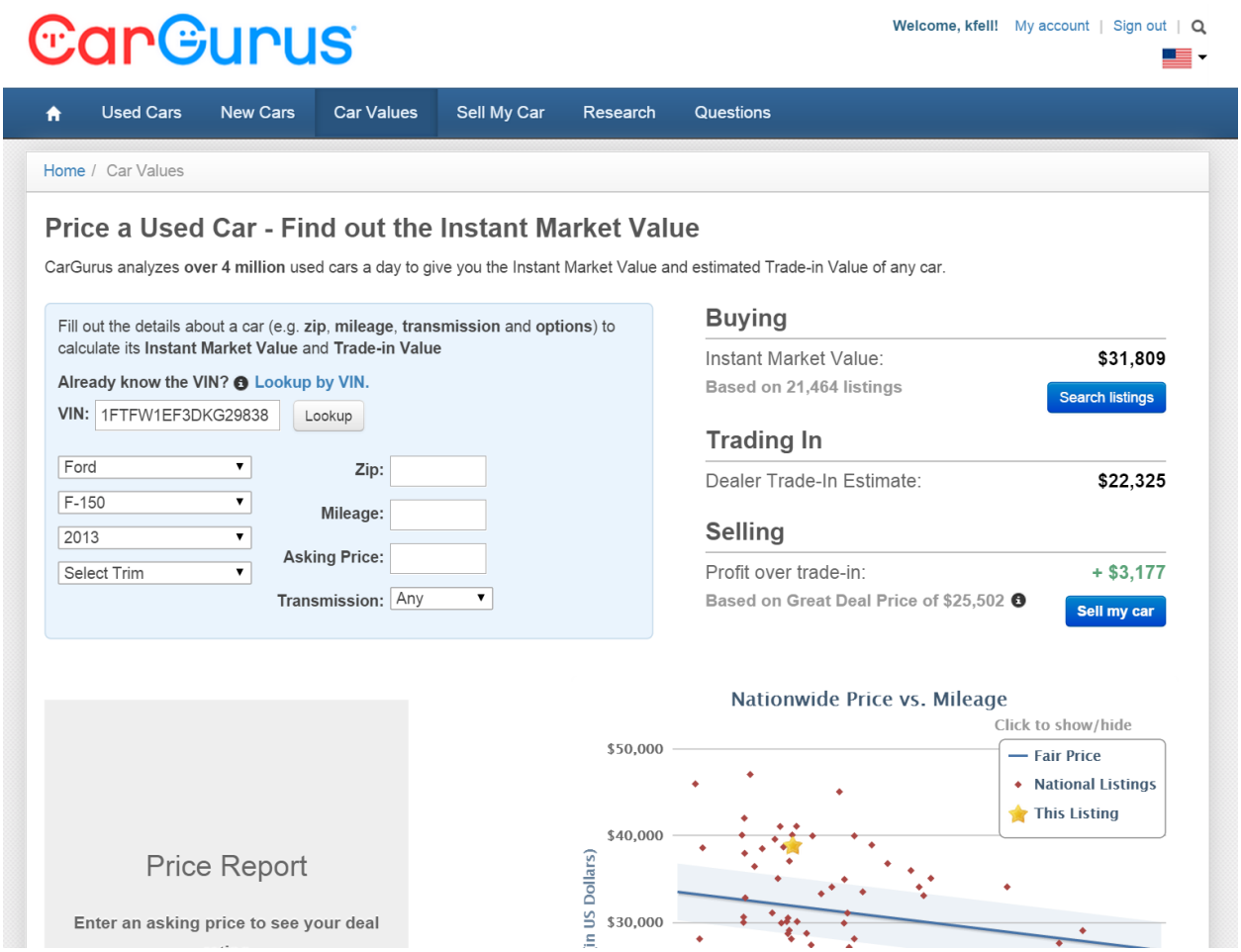
```
Sub writeData(data As String, item As String)

Worksheets("Outputs").Activate
    Cells.find(What:=item).Offset(1048560, 0).Activate
    ActiveCell.End(xlUp).Offset(1, 0).Activate
    ActiveCell.value = data

End Sub
```

After the program loops through all of the listings, collects the data, and writes it to the worksheet I wanted to check the listed prices to find out if they are a good deal. To check the values I used a webpage called “cargurus.com”. On this webpage car values can be looked up based on VIN numbers. I wrote a sub called “checkValue”. “checkValue” starts by loading all of the VIN numbers into an array called “autos”. It does this by writing the VIN numbers that are already in the worksheet to the array:

```
Worksheets("Outputs").Activate
Range("B2").Select
    Range(Selection, Selection.End(xlDown)).Select
   y = Range(Selection, Selection).Count - 1

a.visible = True
ReDim autos(y - 1)

For i = 0 To (y - 1)
  Worksheets("Outputs").Activate
  Cells.find(What:="Vin").Offset(i + 1, 0).Activate
  autos(i) = ActiveCell.value
Next
```

The sub “checkValue” uses the “autos” array to insert the VIN onto the end of the root webpage which searches Car Gurus for the prices. At this point I ran into another problem. The webpage pulls up the data just fine but the source code does not actually include the prices so I could not use the previous method of “a.getText()” to pull the data. I ended up importing the entire webpage to a worksheet in excel called “data” and then parsing through the text to pull the data that I needed. I used the right, mid, and len, funtions to parse through the final text. Below is a screen shot of Car Gurus webpage. I pulled the “Market Value”, “Trade-In Estimate”, and the “Great Deal Price” and set them equal to the module level variables: “market”, “trade”, and “deal”.

In order to allow for errors in the case that the VIN is wrong or not available, I used two if-statements. One if-statement ensures that there is a value, and the other ensures that the data is in the right format. After the data has been error checked the outputs worksheet is activated and the data is written into the appropriate cells.



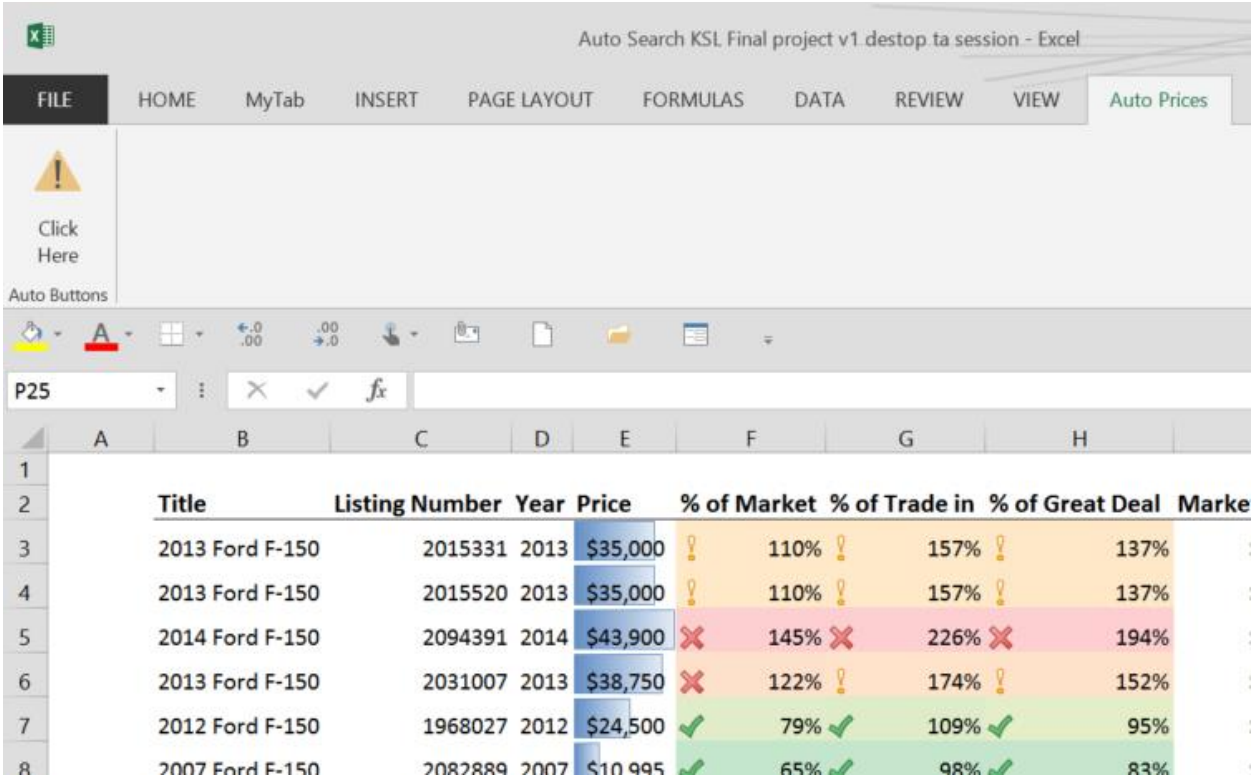
After all of the data has been imported I wanted it to look presentable and understandable. I wrote a sub call "formatCells" that takes care of that. The "formatCells" sub's primary function is to make the data more understandable. The first thing the sub does is format all of the headings as bold and underlines the entire row. The sub then highlights the data and auto fits columns so that all of the data fits in the cells. Once that is done the price and mileage data set are highlighted and data bars are added.

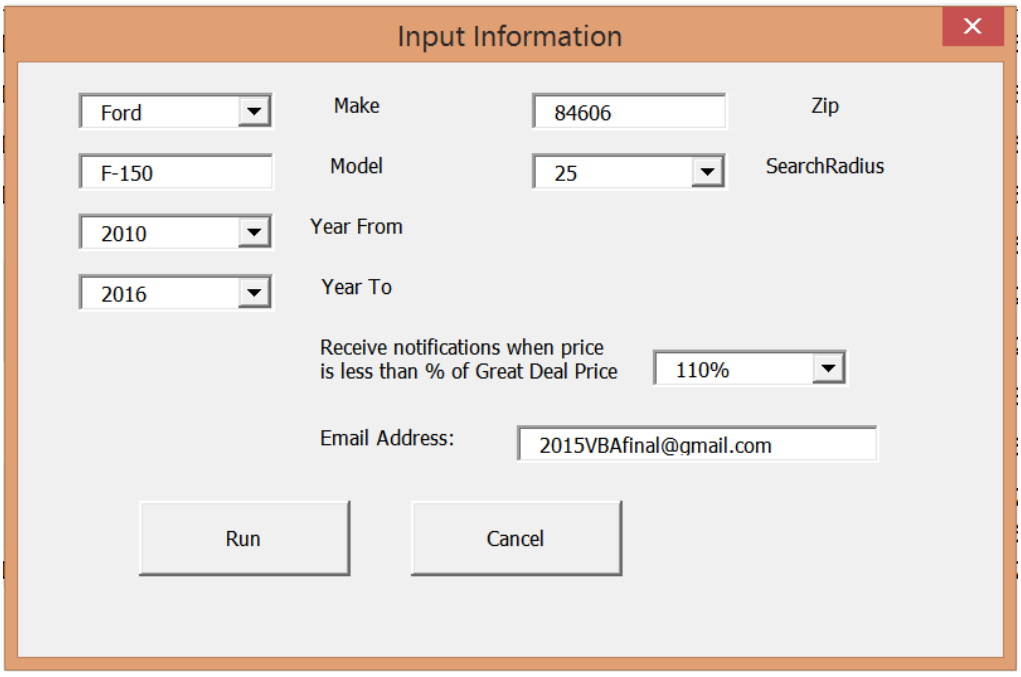
After the existing data has been formatted the "formatCells" sub adds formulas to compare the classified ad list price to the market data. It does this by adding ratios in three columns titled: % of Market, % of Trade in, and % of Great Deal. In each of these columns the list price is divided by each of the respective market measures to come up with a percentage. After the ratios have been added the cells containing the ratios are formatted to contain color scales and an icon set to indicate which of them represent the best deal. Below is a snapshot of the spreadsheet.

Title	Listing Number	Year	Price	% of Market	% of Trade in	% of Great Deal	Market Value	Trade in Value	Great Deal Price	Make	Model	Trim	Body	Mileage	VIN
2013 Ford F-150	2015331	2013	$35,000	110%	157%	137%	$31,809	$22,325	$25,502	Ford	F-150	Lariat	Truck	26,500	1FTFW1EF3
2013 Ford F-150	2015520	2013	$35,000	110%	157%	137%	$31,809	$22,325	$25,502	Ford	F-150	Lariat	Truck	26,000	1FTFW1EF1
2014 Ford F-150	2094391	2014	$43,900	145%	226%	194%	$30,277	$19,458	$22,635	Ford	F-150	Lariat	Truck	8,900	1FTFW1ET3
2013 Ford F-150	2031007	2013	$38,750	122%	174%	152%	$31,809	$22,325	$25,502	Ford	F-150	Limited	Truck	12,592	1FTFW1ET3
2012 Ford F-150	1968027	2012	$24,500	79%	109%	95%	$30,896	$22,503	$25,680	Ford	F-150	XLT	Truck	66,500	1FTFW1EF4
2007 Ford F-150	2082889	2007	$10,995	65%	98%	83%	$16,845	$11,265	$13,285	Ford	F-150	XL	Truck	47,300	1FTRF14W8
2008 Ford F-150	1966043	2008	$12,500	66%	97%	82%	$18,859	$12,910	$15,226	Ford	F-150	N/A	Truck	164,000	1FTPW14V9
2010 Ford F-150	1887660	2010	$33,345	90%	110%	100%	$37,057	$30,217	$33,394	Ford	F-150	SVT Raptor	Truck	73,689	1FTEX1EV8A
2007 Ford F-150	2066985	2007	$19,700	98%	132%	112%	$20,201	$14,890	$17,561	Ford	F-150	Lariat	Truck	98,500	1FTPW14V5
2010 Ford F-150	2033126	2010	$18,950	78%	113%	96%	$24,272	$16,759	$19,766	Ford	F-150	N/A	Truck	108,715	1FTFW1EV2
2011 Ford F-150	2055594	2011	$22,500	79%	110%	95%	$28,356	$20,511	$23,688	Ford	F-150	XLT	Truck	123,000	1FTFW1ET4
2010 Ford F-150	2041983	2010	$25,675	106%	153%	130%	$24,272	$16,759	$19,766	Ford	F-150	XLT	Truck	50,180	1FTFW1EV2

Click Here

As can be seen in the above picture I added a button to the spreadsheet which runs the procedure. I also added a button to the ribbon that does the same thing.



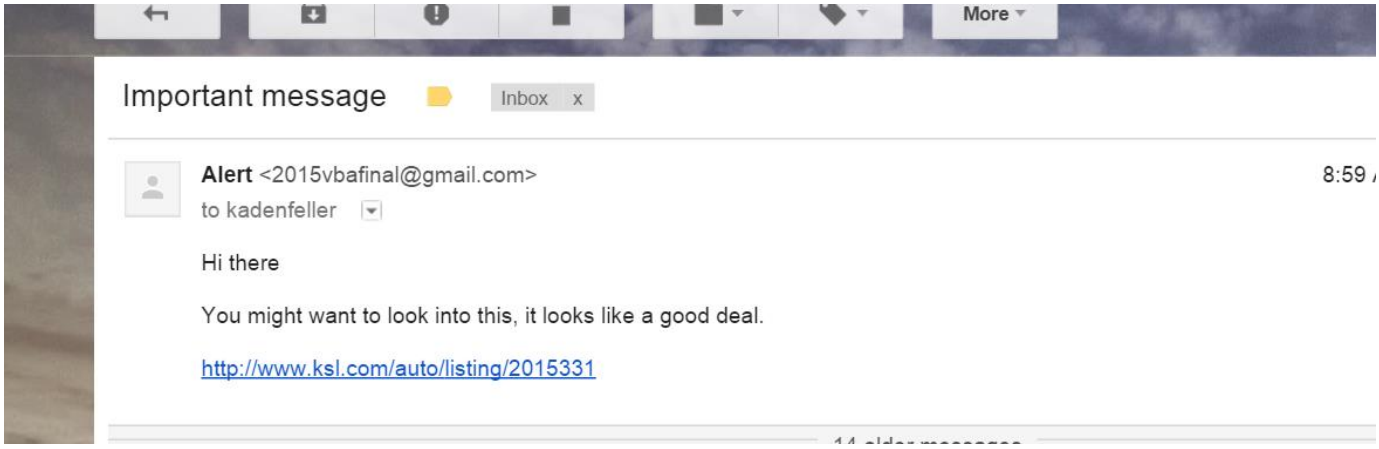


When the user presses either of the buttons a user form appears allowing them to select the options that they want. For most of the fields I used combo boxes. I did this so that there would be less likelihood of the user inputting a value that would not work with the code or the URL. The Make, Model, Year From, Year To, Zip, and Search Radius fields are collected and spliced into the URL to generate search results. Because of this I wanted the options to match the actual options that KSL offers. To do this I wrote a sub that goes to the KSL Auto Classifieds page and pulls the data, loads the data in to arrays, and then loads them into the combo boxes. I have attached screen shots that show the "fillComboBoxes" sub which loads the data into the combo boxes, and the "loadComboBoxArrays" sub which gets the data from the webpage and loads it into the arrays. The user form allows the user to input a threshold for receiving notifications and the user's email where they want to receive the notifications.

```
Sub fillComboBoxes()
Dim element As Variant
Dim i As Integer
loadComboBoxArrays

    For i = 0 To 94
        cboMake.AddItem makes(i)
    Next
    For i = 0 To 116
        cboYearFrom.AddItem yearFroms(i)
    Next
    For i = 0 To 116
        cboYearTo.AddItem yearFroms(i)
    Next
    For i = 0 To 7
        cboSearchRadius.AddItem searchR(i)
    Next

cboHurdle.AddItem "115%"
cboHurdle.AddItem "110%"
cboHurdle.AddItem "105%"
cboHurdle.AddItem "100%"
cboHurdle.AddItem "95%"
cboHurdle.AddItem "90%"
cboHurdle.AddItem "85%"
cboHurdle.AddItem "80%"

End Sub
```

```
Sub loadComboBoxArrays()
Dim a As New agent

Dim value As String
Dim i As Integer
Dim nuOfMakes As Integer

a.openpage "www.ksl.com/auto/", True
a.position = 1
a.moveTo ("searchFormMake")
a.moveTo ("value=""")
i = 0
nuOfMakes = 0
'this is to count how many makes there are
    Do Until value = "Volvo"
        value = a.getText("""")
        a.moveTo ("value=""")
        nuOfMakes = nuOfMakes + 1
    Loop
value = ""
ReDim makes(nuOfMakes - 1)

a.position = 1
a.moveTo ("searchFormMake")
a.moveTo ("value=""")

    Do Until value = "Volvo"
        value = a.getText("""")
        a.moveTo ("value=""")
        makes(i) = value
        i = i + 1
    Loop
        a.moveTo ("searchFormYearFrom")
        a.moveTo ("value=")

            For i = 0 To 116
            a.moveTo ("value=""")
            yearFroms(i) = a.getText("""")
            Next
        a.moveTo ("searchFormMilesFrom")
            For i = 0 To 7
            a.moveTo ("value=""")
            searchR(i) = a.getText("""")
            Next
End Sub
```

After all of the code has run and the data is in the worksheet and formatted correctly the procedure runs a sub that checks each of the ratios. If the ratio is below the threshold the user specified an email is sent to the user including a short message and a link to the listing. The user can then follow the link and investigate further.



Conclusion

This project has taught me a lot and I am excited to see how this program will function in real life. I am sure that there will be additional work that needs to be done to make this more robust and better streamlined; however, I am happy with the results up to this point. I have enjoyed applying the concepts learned in class to this project and I look forward to continuing to apply these lessons to problems that I will encounter throughout my career.