Michael West

ISYS 540

**Final Project Write-Up**

**Executive Summary**

*Background*

This project comes from an old work project I was working on at my previous place of employment. I used to work for MyComfort, a Utah-based mattress company. I worked in the warehouse organizing paperwork part time and I did IT support part time. The VP of operations decided that he wanted to keep a tracking system of all the mattresses that were manufactured with a serial number system. Since we didn’t have money for an advanced serial number system, I created a simple Excel spreadsheet and a small VBA module in which the warehouse worker could select the type of mattress, the size, and it would generate a unique SN. I wanted to make a more sophisticated system but my lack of VBA knowledge and limited time prevented me from doing so before I left the company for another job. Now, with greater knowledge of VBA, I will complete what I intended.

*Business Problem*

The reason why this serial number system was recommended by the VP of Operations was for tracking inventory and warranty issues with the beds. First of all, there were numerous problems with MyComfort being over and under billed for inventory manufactured. The paper work and POs for orders were a mess and not organized. Many times an order would not be 100% finished and/or delivered and MyComfort would still be billed. Using serial numbers and a tracking system as to when things got shipped out would make sure this problem wouldn’t happen.

Second, there had been a lot of poor quality mattresses manufactured and this tracking system would be able to identify beds that were manufactured around the same time that had similar problems to a product that was reported to have issues and be inspected by a customer service representative.

Also, to help with these problems, the company also decided lease out an office in the manufacturer’s warehouse to keep track of production and do QA onsite. This way a MyComfort worker could attach serial numbers to the beds even before they left the manufacturer’s site.

*Program Solution*

My solution to this problem includes an Excel workbook with a two forms and two sheets. The procedures and program would facilitate the following:

1. As POs are created, an employee would enter in (using the 1st VBA form) each product with the quantity. Also entered includes
2. The worker would submit the PO into the system and a Word document would be generated with the serial numbers for easy printing (perhaps Avery label sticker templates would be used for an easy ‘stick-on’ serial number). The serial number would be unique and is based on the mattress type, size, and month in which it was ordered. Two stickers would be created for every serial number. This is because one needs to go on the mattress itself and the corresponding PO.
3. As each serial number is produced, a worksheet is populated. Each item produced is listed along side of its serial number, the PO number, and the date of order.
4. As each bed is produced, the worker would have to stick the serial number on each mattress (on tag). Also, the worker would place the stickers on the POs (actual document) in which they were found
5. When a shipment is about to go out, the worker would return to the program and enter in exactly what product is going out by selecting the serial numbers/product. This would be done using a multi-select list box.
6. The worksheet would be populated with the date the product was actually shipped. These worksheets would be organized by month and would be able to be sorted by each column for reporting purposes.
7. Other features will be added to enhance and facilitate the process as identified.

**Implementation Documentation**

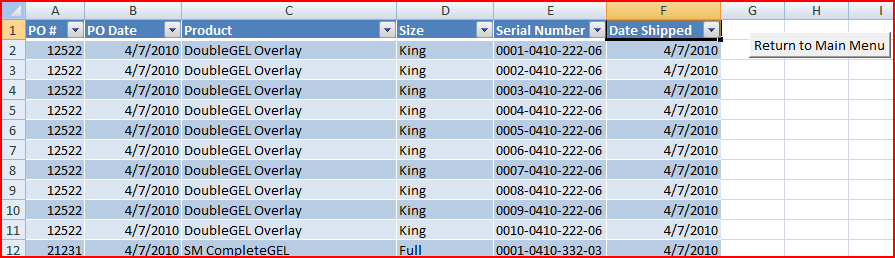
*Worksheets*

To start it all off, I created a user-friendly main page for user access to the features of the project. This includes 3 buttons to click on – one to enter a new PO, one to ship out products, and one to route to the second sheet – which displays all the information.



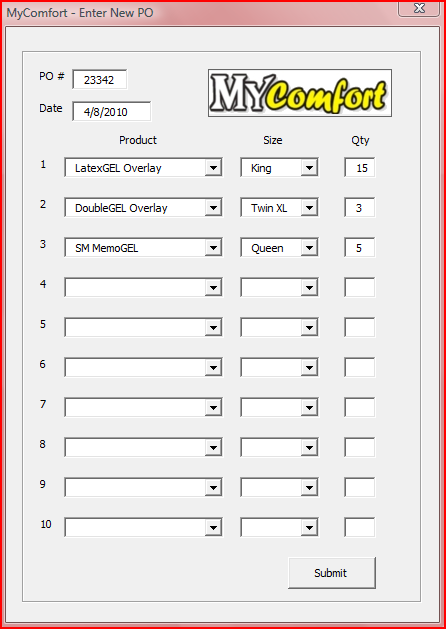
Main Page (1st worksheet)

This second sheet is a product page that includes a table with 6 columns. They are PO #, PO date, Product, Size, Serial Number, and Date Shipped. It also includes a button to return to the main menu. All the buttons are linked to macros that initialize the specific forms.

Product Page with Sample Data (2nd worksheet)

*Enter New P.O.*

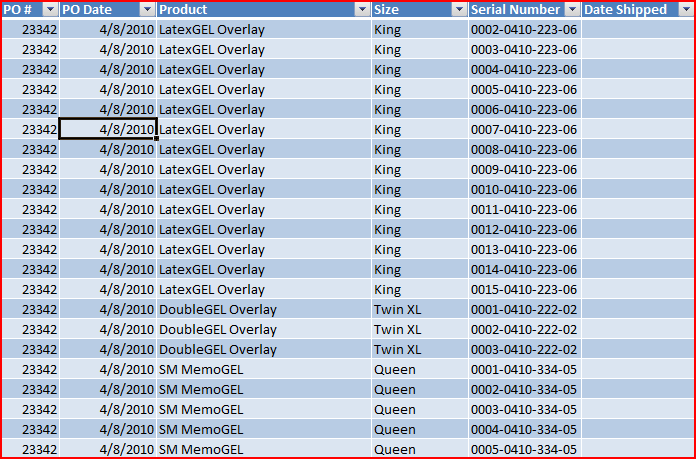
Once the user clicks on ‘Enter New P.O’, this form appears.



Enter New P.O. User Form

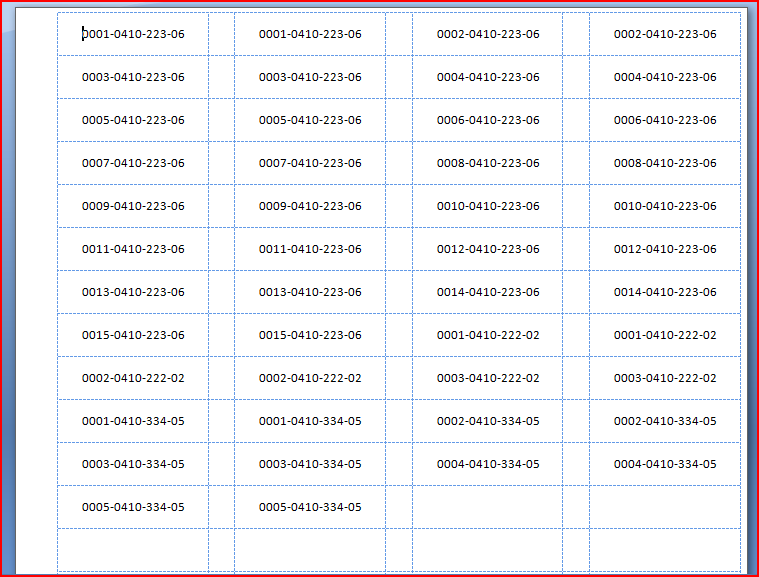
This is a simple form that includes the PO number field, the date field, and ten PO lines (normal POs from the company are very small as the company runs on a JIT system.) Each PO line has 2 dropdown boxes for product and size and a quantity field. The products and sizes are populated from a hidden sheet with that info upon the form initializing (userform\_initialize). The date default is set to the current date. There are size restrictions on the PO number (5) and the quantity (no more than 3 digits). Also, tab order is set in the downward-right direction.

Once the ‘Submit’ button is pushed a few things happen. First, the product page is populated with the corresponding input and a unique generated serial number. See these entries for example (this is populated from the input as seen in the previous screenshot:



Enter New P.O. User Form

Second, the serial number document is created from an Avery label. The program opens up the template, takes the serial numbers just generated from the new PO, and then pastes them into the document and saves it. Two stickers are generated for each serial number - one for the mattress and one for the hard copy of the PO. Here is a screenshot of the word document created:

****

Word Document with Serial Numbers

I’ll now step through how these two events occur starting with the worksheet population.

*Worksheet Population*

Once the submit button is pressed, I have the input directly read from the input boxes to program variables for manipulation. Once everything is in variables, it’s time to sort through them and place them in the spreadsheet in order. I did this by activating cells and populating row by row.

Here is the code (w/ comments) that I use to determine where to start inserting the new PO info by determining where the next blank row is:

'Set Up for Inserting new PO into sheet - finds the right cell to start populating

OrderSheet.Activate

OrderSheet.Range("A1").Activate

If Range("A2").Value = Empty Then 'if nothing is in the sheet it starts from the top

Range("A2").Activate

Else

ActiveCell.End(xlDown).Activate

ActiveCell.Offset(1, 0).Activate

End If

After it is determined where to insert the info, each PO line is inserted using a function I created called populatesheet. This function requires the quantity, PO number, PO date, product, and size for each PO line. As there are 10 lines, those required variables were denoted as the following in the function where ‘x’ is the line number:

populatesheet(qtyx, podate, productx, sizex, ponum)

This function then goes line by line, looping by quantity, and fills out the sheet. Here is the code:

Function populatesheet(qty As String, podate As Date, product As String, size As String, ponum As Double)

If bedbox <> "" Then 'if the field is populated with something...then populate the sheet accordingly

For i = 1 To qty

serial = getSerialNum(podate, product, size)

ActiveCell = ponum

ActiveCell.Offset(0, 1) = podate

ActiveCell.Offset(0, 2) = product

ActiveCell.Offset(0, 3) = size

ActiveCell.Offset(0, 4) = serial

ActiveCell.Offset(1, 0).Activate

Next i

End If

*Serial Number Creation*

As seen above, this function calls another function called getSerialNum(). This returns a unique thirteen digit serial number (ex. 0024-0410-221-05). The first four digits (0024) are the unique identifier. The next four digits are the month and year the mattress was produced (0410 – April 2010). The next 3 are predetermined and based on what type of mattress (222 = SingleGEL Overlay). The last two are also predetermined from the bed size 05 = Queen). Accordingly, the method takes three parameters: the PO date, the product name, and the size). The unique number is determined by how many beds there are of the same type, same manufactured date and month, and same size and it increases by one accordingly.

Here is the code that shows how the serial numbers are generated. This code is summarized where noted w/ brackets. Also, please note the comments as they explain in detail which each part does.

Public Function getSerialNum(podate As Date, bed As String, size As String)

[variables are declared]

'sets date code

year = DatePart("yyyy", podate)

year = Right(year, 2)

month = DatePart("m", podate)

If Len(month) = 1 Then

month = 0 & month

End If

datecode = month & year

'set bed code

Select Case bed

Case Is = "Chiro"

bedcode = "111"

Case Is = "SingleGEL Overlay"

bedcode = "221"

Case Is = "DoubleGEL Overlay"

bedcode = "222"

[cases are set and codes are given for all beds]

End Select

'set size code

Select Case size

Case Is = "Twin"

sizecode = "01"

Case Is = "Twin XL"

sizecode = "02"

[cases are set and codes are given for all sizes]

End Select

'creates a searchstring in order to search for how many of the same bed were made during the same month and same size

searchstring = "-" & datecode & "-" & bedcode & "-" & sizecode

'get id code - this count shows many of the same beds were made during the same month and same size

count = WorksheetFunction.CountIf(OrderSheet.Columns(5), "\*" & searchstring)

'If there aren't any beds, the serial number starts with 1. Otherwise, it adds one for the next unique serial

If count = "0" Then

count = "1"

Else

count = count + 1

End If

'this formats the count with zeroes in front of it - depends on the number of digits

If Len(count) = 1 Then

count = "000" & count

ElseIf Len(count) = 2 Then

count = "00" & count

ElseIf Len(count) = 3 Then

count = "0" & count

End If

'returns the completed serial number

getSerialNum = count & searchstring

End Function

*Print Serials*

The last thing the “Submit” button does on the new PO form is print out the serial numbers onto a word document. This is done using the printserials() function. This function is called after the worksheet is populated with the PO information. The cells.find() function is then called to find first instance of the PO number on the sheet. Using that cell address, a count is made as to how many beds were produced under that PO number. Then, using that number in a for loop, each serial number of all beds found with the PO number is put into an array of serial numbers. See the following code:

Cells.Find(ponum).Activate 'finds the first instance of the PO#

bedcount = Range(ActiveCell.Address, ActiveCell.End(xlDown)).count 'finds how many products are on the specific PO

'puts all the serial numbers from the PO in an array

ReDim serials(1 To bedcount)

For i = 1 To bedcount

serials(i) = ActiveCell.Offset(0, 4).Text

ActiveCell.Offset(1, 0).Activate

Next i

The next part takes the array of serial numbers and opens a Word document Avery labels template. The Word document is then populated with the serials. As mentioned above, there are 2 stickers made for each serial number. After the document is populated, it saved with a new name based on the PO number. Then the document is closed and the word application is closed. See the following code as to how this is accomplished. Also, you may reference the Word document screen shot at the beginning of the document to see how the Avery table is formatted.

'opens word document from template, populates the document, and then saves the word document

Dim wrdApp As Word.Application

Dim wrdDoc As Word.Document

Set wrdApp = CreateObject("Word.Application")

wrdApp.Visible = False

Set wrdDoc = wrdApp.Documents.Open(ThisWorkbook.Path & "\serialnumbertemplate.docx") 'open template

'Takes the serial numbers in the array and prints them (2 stickers for each) on the page in the different tables

With wrdApp

With .Selection

For x = 1 To UBound(serials) - LBound(serials) + 1 ‘length of the array

.TypeText serials(x)

.moveright ‘moves to the next cell in the table

.moveright

.TypeText serials(x)

.moveright

.moveright

Next x

End With

End With

'Save and quit word

wrdDoc.SaveAs (ThisWorkbook.Path & "\PO" & ponum & "serials.docx")

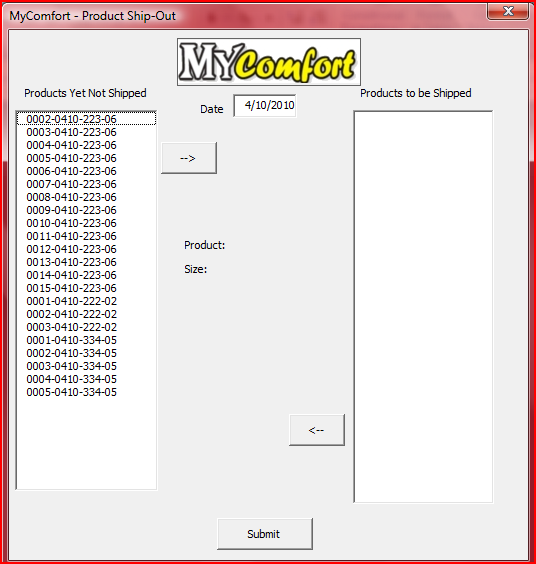
wrdApp.Quit

Set wordapp = Nothing

As shown in the code, the Avery template document needs to be in the same directory as the spreadsheet and the output will be put there as well. The worker would consequentially open the ofler and word file and print.

*Ship Out Product Functionality*

This functionality is pretty straightforward. This second user form basically allows the user to select what beds are shipping out using two list boxes. Once selected, the ‘Date Shipped’ column is populated with the date specified (current date is the default). Here is the form that appears when the button is pressed:

****

Ship-out User Form

As seen, there are a few steps taken in the UserFormInitialize() sub procedure. First of all, it determines how many lines are populated on the second worksheet. Then a loop is set up to go from top to bottom to check if that certain bed has been shipped by checking if that column in populated. If not, if adds it to the list box list. Here’s the for loop:

For i = 1 To sncount

If ActiveCell.Offset(0, 1).Text = Empty Then

SerialNumBox.AddItem ActiveCell.Text

End If

ActiveCell.Offset(1, 0).Activate

Next i

As far as the selecting the serial numbers go, there’s the ‘move right’ button and the ‘move left’ button. I created simple sub procedures for their event calls so the serial numbers move from one side to the other. A for loop was created to check if each was selected, and then moved them accordingly. Here’s the code:

Private Sub moveright\_Click()

For i = SerialNumBox.ListCount - 1 To 0 Step -1

If SerialNumBox.Selected(i) = True Then

SelectedSerials.AddItem SerialNumBox.List(i)

SerialNumBox.RemoveItem i

End If

Next i

End Sub

----------

Private Sub moveleft\_Click()

If SelectedSerials.ListIndex <> -1 Then

For i = SelectedSerials.ListCount - 1 To 0 Step -1

If SelectedSerials.Selected(i) = True Then

SerialNumBox.AddItem SelectedSerials.List(i)

SelectedSerials.RemoveItem i

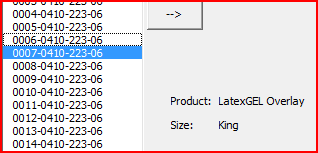
End If

Next i

End If

End Sub

For an easy check, I added an additional functionality. To verify that the user has the right bed, he/she can double click on any serial number, and the bed and size will appear like so:

****

Bed Details

The code for this is as follows:

Private Sub SerialNumBox\_DblClick(ByVal Cancel As MSForms.ReturnBoolean)

Dim serialnum As String

For i = SerialNumBox.ListCount - 1 To 0 Step -1

If SerialNumBox.Selected(i) = True Then

serialnum = SerialNumBox.List(i)

productlabel.Caption = OrderSheet.Cells.Find(serialnum).Offset(0, -2)

sizelabel.Caption = OrderSheet.Cells.Find(serialnum).Offset(0, -1)

End If

Next i

End Sub

Once the ‘Submit’ button is pressed, a for loop goes through each serial number in the left hand box , finds each serial number, then populates the ‘Date Shipped’ column with the date provided in the form. Here’s the for loop:

For i = 1 To size

OrderSheet.Activate

OrderSheet.Range("e1").Activate

serialnum = SelectedSerials.List(i - 1)

Cells.Find(serialnum).Offset(0, 1) = DateBox.Text

Next i

Here’s an example of the finished spreadsheet after the form has been filled:

Completed and Shipped Orders

With all this functionality, this table is easy to filter, sort, and run pivot tables on. This program provides greater organization and order to a once messy process.

**Learning Discussion/Difficult Concepts Encountered**

This was a very worthwhile project to work on. The only regret I have is that I don’t work at MyComfort anymore and I wish I had this when I did work for them. Besides that, I enjoyed problem solving for the project and learning more about VBA.

I think the most valuable things I learned were how to use list boxes and how to manipulate word documents. Those were the most challenging things for me to get working. The list boxes took me a while as I finally figured out the best way was to use for loops to go through the whole list and see what items were selected. In other languages, functionality provides you info on what specific list item is selected and you can manipulate it from there. It was tough trying to find a workaround for that lack of functionality in VBA.

The other thing that took some patience in figuring out was how to manipulate the Word document. For the life of me, I couldn’t figure out how to add to the document elements that were already there without wiping everything out. I did some online research and the only way I could get it to work the way I wanted was using the ‘With Selection’ statement (as seen in the code). It seems that might be the only way to do it. Fortunately enough, the Avery label templates were just tables and I could easily navigate within those to populate the template.

Overall, this was a very enjoyable process and I feel proud of the work I accomplish.