BYU INDEPENDENT STUDY

IVENTORY AUTOMATION THROUGH VBA

April 13, 2011

Adam Wingate, ISYS 540

EXECUTIVE SUMMARY

BYU Independent Study employs a number of processes to accomplish its day-to-day operations. One of the processes allows employees (field agents, sales, marketers) to place orders and requests for SWAG and booth supplies to be delivered to destination locations before arriveing. These locations may be schools, hotels, or residential.

Before this program, BYUIS managed this process manually. The users are the field agents, sales specialists, and marketers. Users place orders with the shipping office at BYUIS. Orders were maintained in a notebook with sticker labels to match the shipping labels. Inventory was adjusted manually despite being kept in an excel spreadsheet.

Objective:

- Give users a simple way to place orders.
- Track order from placement to receipt in one place.
- Inform users by email/text of delivery information.
- Update inventory after shipment.
- Preserve a platform that can handle employee turnover.
- Provide a Macros to:
    - Show a form (dashboard) that will roll up all inventory information into one screen.
    - Update Inventory from shipped requests.

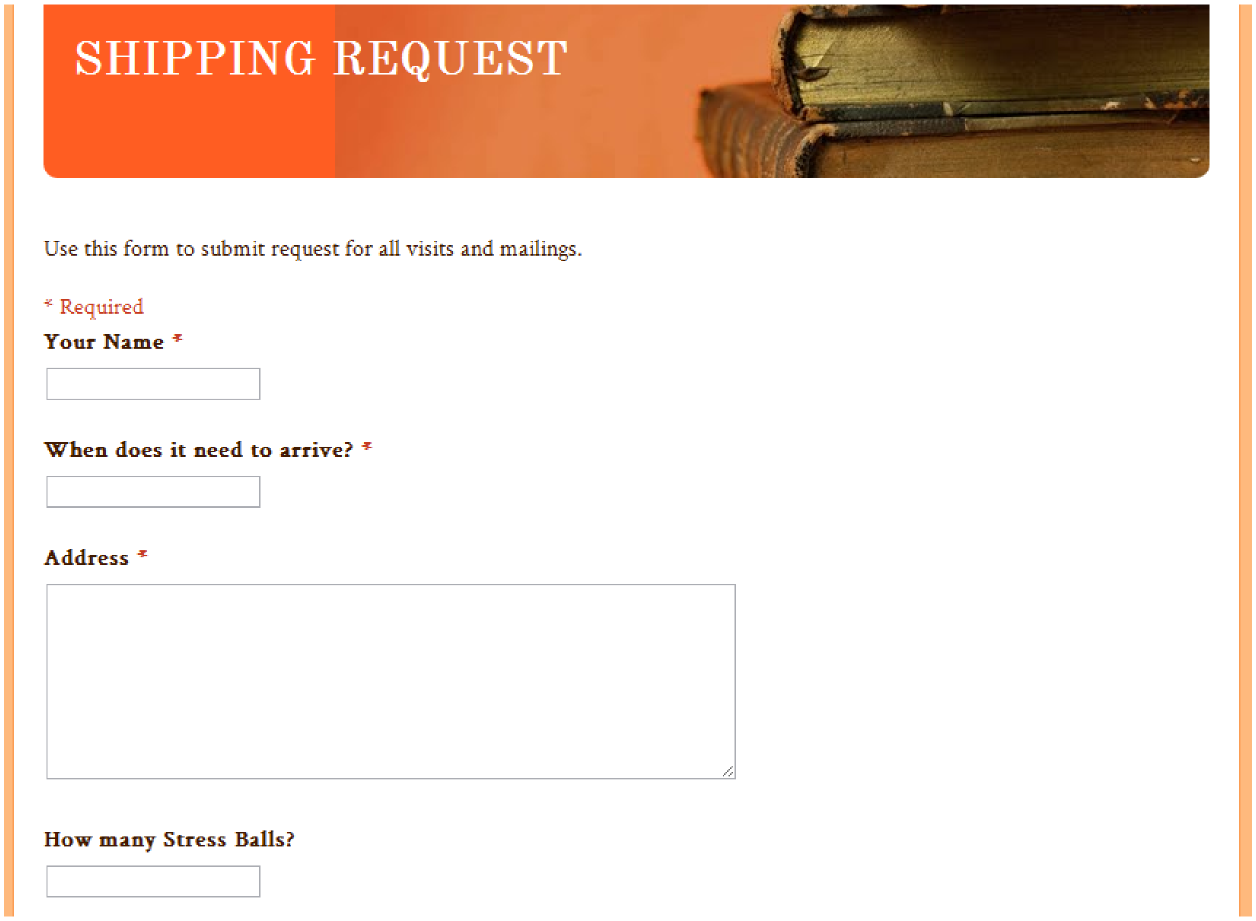
These are the issues expressed by the BYUIS in this process. Excel is used to store all inventory information. Google docs and Google forms gather requests from users. Only the shipping office deals with the excel spreadsheet. He will receive orders and process then through third party software provided by FedEx. After recording all tracking numbers, the Google doc spreadsheet is scraped into Excel with VBA. Information from the orders/shipments update the inventory. A form has been created. It is not interactive. It is a dashboard. The dashboard of inventory data will simply display the inventory data (contained on 8 sheets) in one form view.

The programming cost estimate is approximately 30 hours. The larger portion of the project will be the support of the programs and processes through technical documentation. Because this is a “real world” project, documentation is essential. This project estimates an additional 20 hours of documentation including:

- Technical overview of Macros and program.
- “How to” customization examples
- Change Management Scenarios
    - New User Employee
    - New Inventory Item
    - Change Web Resources

IMPLEMENTATION

The elements of the program from the users perspective include only a few new interfaces. The frist is the Google from:



SHIPPING REQUEST

Use this form to submit request for all visits and mailings.

* Required

Your Name *

When does it need to arrive? *

Address *

How many Stress Balls?

This form populates a Google spreadsheet that is published with a secure link. The sheet is only accessible to those who have the link. The spreadsheet does not contain confidential information but it is requested that the link not be published in this doc. You will see the link address and any passwords crossed out in the images.

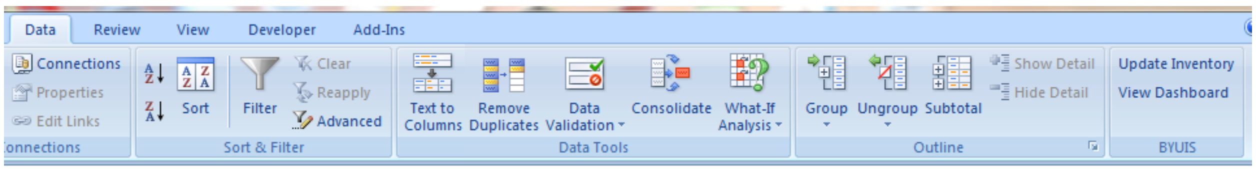
The above form requests information on:

- Requestor's Name
- When it needs to arrive (for shipping)
- Address (for shipping)
- Amount of Inventory Items
    - Stress Balls
    - Pens
    - CEGs
    - Catalogs
    - HS/MS Credit Brochures
    - Notepads

  - “What’s New” Postcards
  - “Free Courses” Postcards
  - Mice with Batteries
  - Umbrellas
  - Golf Balls
  - CDs
- Inventory Type
  - Visits
  - Mailings
  - Field Representatives

This information is all important to track in the inventory sheet. BYUIS shipping already uses an annual template to record and store inventory information. The inventory type is especially important in determining the type of inventory (which inventory sheet) needs to be updated. Any miscellaneous items are one time exceptions and will be managed manually. Those items might include iPods, tickets, posters, etc.

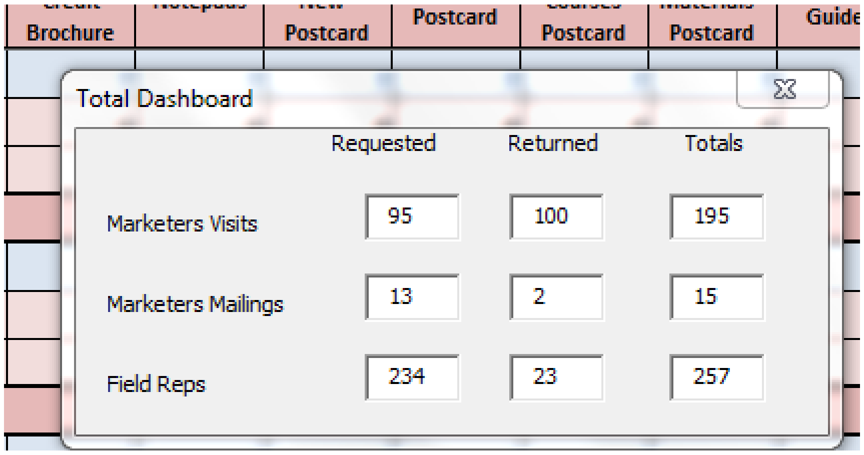
The shipper will select the new button (Update Inventory) on the ribbon to record inventory and the rest of the process will be automated.



When Update Inventory is selected the application will:

- Scrape the Google Doc spreadsheet and add it as a new sheet in the current workbook.
- Update Inventory on the sheet specified by the Inventory Type listed in the form.
- Send a generic email with listed Tracking Numbers.
- Send a text message with the listed Tracking Numbers.

This may seem redundant but it was a project request. From the Ribbon we can actually select the option to View inventory in a dashboard.



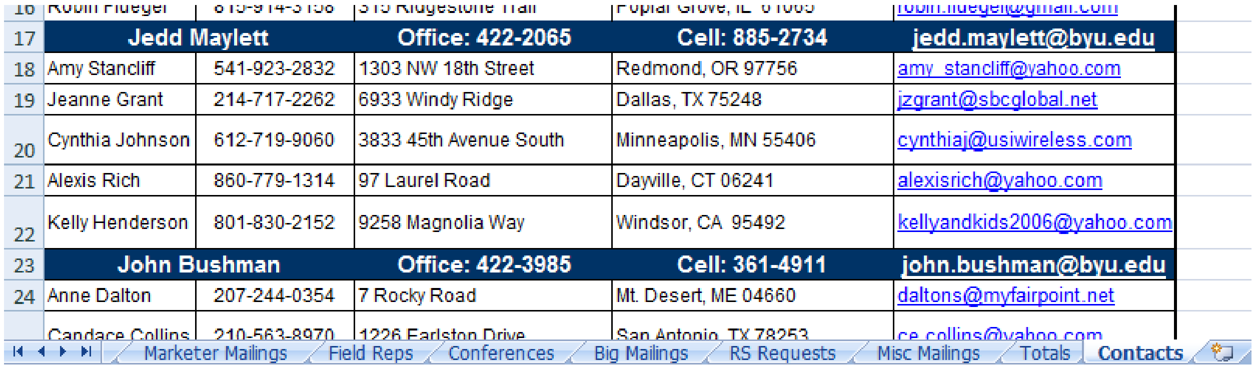
Total Dashboard

	Requested	Returned	Totals
Marketers Visits	95	100	195
Marketers Mailings	13	2	15
Field Reps	234	23	257

CHANGE MANAGEMENT

New Employee:

When a new employee is added to the process the only requirement is that they be added with appropriate information to the “Contacts” sheet.

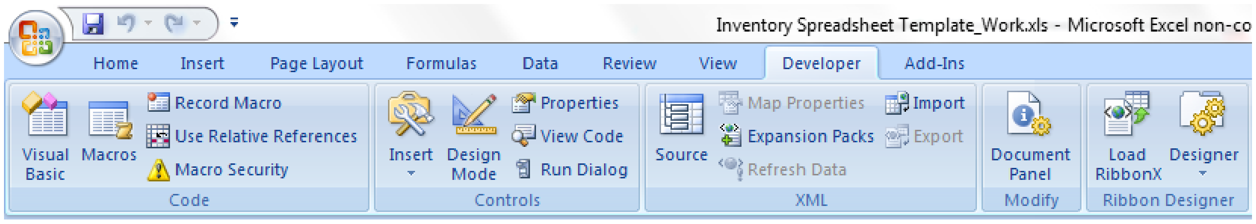


16	Robin Flueger	815-914-3158	315 Ridgestone Trail	Poplar Grove, IL 61065	robin.flueger@gmail.com
17	Jedd Maylett	Office: 422-2065		Cell: 885-2734	jedd.maylett@byu.edu
18	Amy Stancliff	541-923-2832	1303 NW 18th Street	Redmond, OR 97756	amy_stancliff@yahoo.com
19	Jeanne Grant	214-717-2262	6933 Windy Ridge	Dallas, TX 75248	jzgrant@sbcglobal.net
20	Cynthia Johnson	612-719-9060	3833 45th Avenue South	Minneapolis, MN 55406	cynthiaj@usiwireless.com
21	Alexis Rich	860-779-1314	97 Laurel Road	Dayville, CT 06241	alexisrich@yahoo.com
22	Kelly Henderson	801-830-2152	9258 Magnolia Way	Windsor, CA 95492	kellyandkids2006@yahoo.com
23	John Bushman	Office: 422-3985		Cell: 361-4911	john.bushman@byu.edu
24	Anne Dalton	207-244-0354	7 Rocky Road	Mt. Desert, ME 04660	daltons@myfairpoint.net
	Candace Collins	210-563-8970	1226 Earlston Drive	San Antonio, TX 78253	ce.collins@yahoo.com

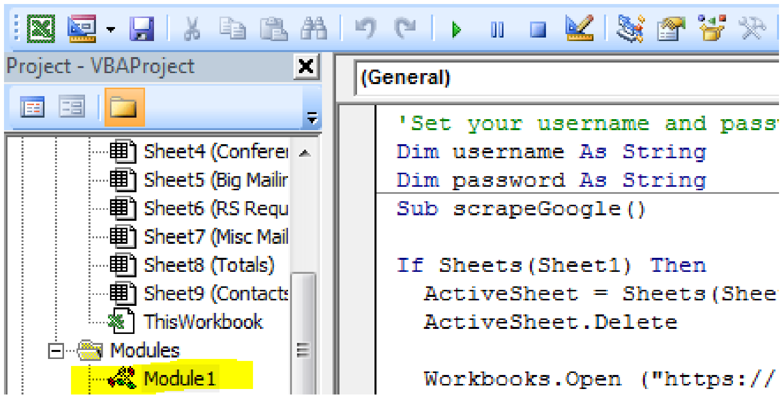
New Email Address:

This completed application is set up to use a Gmail account. The supported email account change will only be another Gmail account. If it becomes necessary to change to an exchange server the project will need to be revisited.

Changing to a different account and/or password in the application will require selecting the Developer tab and clicking Visual Basic.

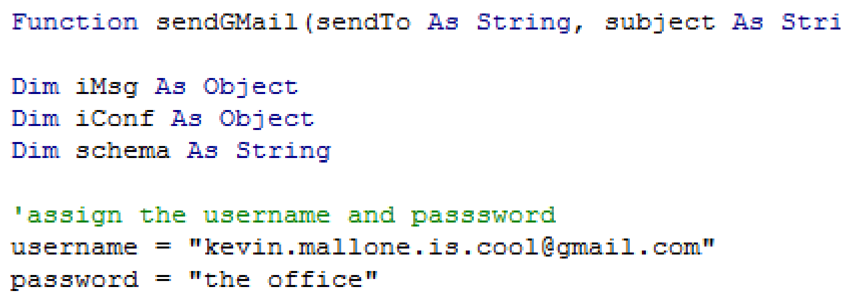


When open, double click on Module 1 to make sure you’re in the right place.



```
'Set your username and pass
Dim username As String
Dim password As String
Sub scrapeGoogle()

If Sheets(Sheet1) Then
  ActiveSheet = Sheets(Shee
  ActiveSheet.Delete

  Workbooks.Open ("https://
```

You'll quickly find the password and account by using Ctrl + f and typing in the current account. If you do not know the current account and password then use Ctrl + f and search "sendGmail". The second instance of sendGmail is the sub procedure that you want. You should see something that looks like this:



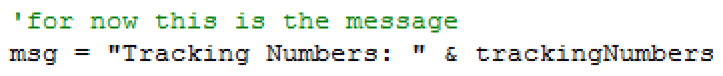
```
Function sendGMail(sendTo As String, subject As Stri

Dim iMsg As Object
Dim iConf As Object
Dim schema As String

'assign the username and passsword
username = "kevin.mallone.is.cool@gmail.com"
password = "the office"
```

You are interested in changing the password, or the email account, or both.

Changing the Email Message

In order to change the message you will again need to go into the code. A change can be made quickly to send a text document as the body of the email. In this case the demonstration will be to add additional text to the message as it is sent today.

The body of the email is contained in the String variable msg. A Ctrl + f for the variable msg will reveal where the msg text is being set. It will look something like this:



```
'for now this is the message
msg = "Tracking Numbers: " & trackingNumbers
```

The result of this code is:

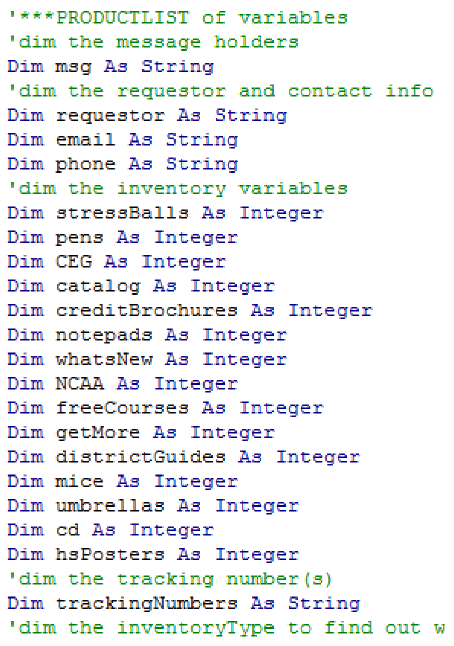
Tracking Numbers: 90800089, 90830934

If 90800089 and 90830934 are the tracking numbers associated with that package. This is the entire body of the email. If you add text before "Tracking Numbers: " be sure that it appears inside the quotation marks like: "Hi,   The tracking numbers are " be sure it is always followed by & trakcingNumbers. This will place the actual tracking numbers into the email.

Additional Products

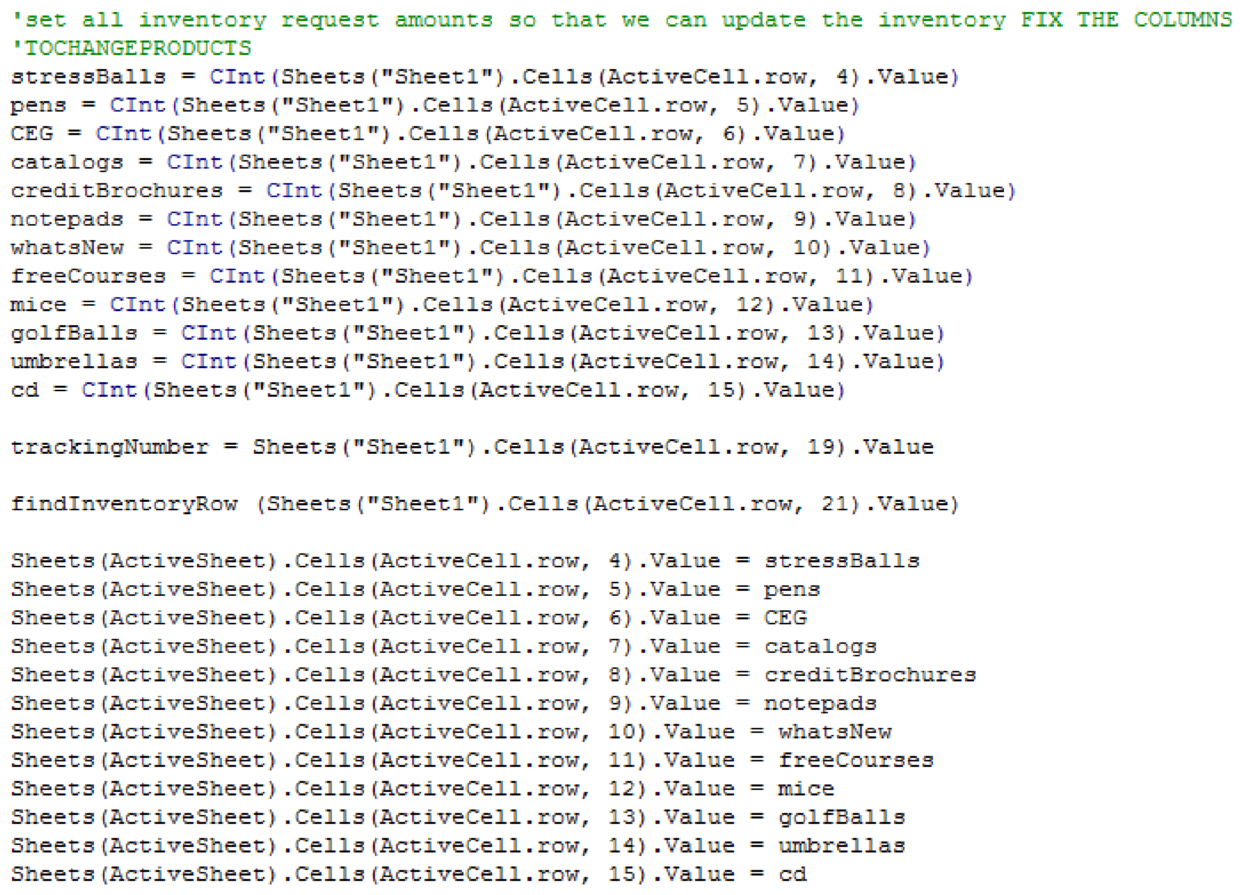
If in future spreadsheets you add a new product to be tracked in inventory there are 3 places that this will need to be changed.

First, you will need to create a new variable for the product. You can simply add this to the list. Search (Ctrl + f) for "PRODUCTLIST". You should see this:



```
'***PRODUCTLIST of variables
'dim the message holders
Dim msg As String
'dim the requestor and contact info
Dim requestor As String
Dim email As String
Dim phone As String
'dim the inventory variables
Dim stressBalls As Integer
Dim pens As Integer
Dim CEG As Integer
Dim catalog As Integer
Dim creditBrochures As Integer
Dim notepads As Integer
Dim whatsNew As Integer
Dim NCAA As Integer
Dim freeCourses As Integer
Dim getMore As Integer
Dim districtGuides As Integer
Dim mice As Integer
Dim umbrellas As Integer
Dim cd As Integer
Dim hsPosters As Integer
'dim the tracking number(s)
Dim trackingNumbers As String
'dim the inventoryType to find out w
```

You'll notice that there are some products that even today aren't used. Simply add a new product by typing "Dim <new product variable name> as Integer" on a new line. Then you'll need to grab the amount of the form and move it to the right inventory sheet. This is done in these next two sections. You'll find them by searching (Ctrl + f) "TOCHANGEPRODUCTS".



```
'set all inventory request amounts so that we can update the inventory FIX THE COLUMNS
'TOCHANGEPRODUCTS
stressBalls = CInt(Sheets("Sheet1").Cells(ActiveCell.row, 4).Value)
pens = CInt(Sheets("Sheet1").Cells(ActiveCell.row, 5).Value)
CEG = CInt(Sheets("Sheet1").Cells(ActiveCell.row, 6).Value)
catalogs = CInt(Sheets("Sheet1").Cells(ActiveCell.row, 7).Value)
creditBrochures = CInt(Sheets("Sheet1").Cells(ActiveCell.row, 8).Value)
notepads = CInt(Sheets("Sheet1").Cells(ActiveCell.row, 9).Value)
whatsNew = CInt(Sheets("Sheet1").Cells(ActiveCell.row, 10).Value)
freeCourses = CInt(Sheets("Sheet1").Cells(ActiveCell.row, 11).Value)
mice = CInt(Sheets("Sheet1").Cells(ActiveCell.row, 12).Value)
golfBalls = CInt(Sheets("Sheet1").Cells(ActiveCell.row, 13).Value)
umbrellas = CInt(Sheets("Sheet1").Cells(ActiveCell.row, 14).Value)
cd = CInt(Sheets("Sheet1").Cells(ActiveCell.row, 15).Value)

trackingNumber = Sheets("Sheet1").Cells(ActiveCell.row, 19).Value

findInventoryRow (Sheets("Sheet1").Cells(ActiveCell.row, 21).Value)

Sheets(ActiveSheet).Cells(ActiveCell.row, 4).Value = stressBalls
Sheets(ActiveSheet).Cells(ActiveCell.row, 5).Value = pens
Sheets(ActiveSheet).Cells(ActiveCell.row, 6).Value = CEG
Sheets(ActiveSheet).Cells(ActiveCell.row, 7).Value = catalogs
Sheets(ActiveSheet).Cells(ActiveCell.row, 8).Value = creditBrochures
Sheets(ActiveSheet).Cells(ActiveCell.row, 9).Value = notepads
Sheets(ActiveSheet).Cells(ActiveCell.row, 10).Value = whatsNew
Sheets(ActiveSheet).Cells(ActiveCell.row, 11).Value = freeCourses
Sheets(ActiveSheet).Cells(ActiveCell.row, 12).Value = mice
Sheets(ActiveSheet).Cells(ActiveCell.row, 13).Value = golfBalls
Sheets(ActiveSheet).Cells(ActiveCell.row, 14).Value = umbrellas
Sheets(ActiveSheet).Cells(ActiveCell.row, 15).Value = cd
```

The top half contain the product variable name followed by an equals sign (=). The bottom half end with an equals sign (=) and then the product variable name. The product variable name should be the one you added to the PRODUCTLIST above. Copy the exact notation.

The only important thing you will need to change is the number. The number corresponds with the column in the form where that variable can be found. You'll notice they aren't the same top to bottom. When a new variable is added you'll also want to make sure you haven't tampered with the order of products in the inventory.

Example:

cd = CInt(Sheets("Sheet1").Cells(ActiveCell.row, 15).Value) will set cd equal to whatever is in column 15 or column "O".

Sheets(ActiveSeet).Cells(ActiveCell.row, 4).Value = stressBalls will set column 4 equal to the value that was found for stressBalls from the Google doc.

Contact Adam Wingate (awingate86@gmail.com) or a member of ISYS 540 for additional help adding products.

PROJECT CHALLENGES

This project posed a number of difficult challenges. The first challenge was scraping the Google spreadsheet and pulling Google Form information into the Excel spreadsheet. I was unable to get this to work using the built in explorer while recording the macro. I also came up short attempting to accomplish this with the agent tool. I finally was able to access this as a simple spreadsheet by publishing the workbook in CSV format and referencing it as a workbook. This still poses a few challenges. If the connection is not broken, edits cannot be made to the Google doc. This means that I was forced to remove the sheet after running the script. This is not a large programmatic issue but it does leave some residual risk.

The residual risk is that if a user happens to submit a form when the macro is running it will not record the request.

Gathering Requirements

The biggest difficulty came when transforming a near manual process into an automated process. There is never sense in recreating an incorrect process through an application if it is in fact incorrect. Working with BYUIS I was able to advise in enhancing the current process. 8 hours were spent working with BYUIS redesigning this process. In the end we were able to reach many compromises that allow users to continue to work as they always have and eliminating the overhead required historically by shipping.

Avoiding Scope Creep

This project is to be implemented in the summer, as a result I me frequently with BYUIS representatives to gather requirements and verify functionality. A common result of meetings like this is the introduction of additional features. The only feature to make it into the project is the VBA Form dashboard. Other requirements were labeled future and will be addressed post production.

Cell Phone Providers

It should be known that people are the most common source of problems in a given project. This project was no exception. In order to send text messages the application needs to know which providers each marketer has. Today the code is in place to add the provider column to the contact information page.

```vb
Sub sendText(number As String, tracking As String, provider As String)

'This section requires some time becuase the Marketers need to post their cell phone provider. Until we know their provider
'there is no way to send them a text.
If provider = "AT&T" Then
  provider = "txt.att.net"
ElseIf provider = "T-Mobile" Then
  provider = "tmomail.net"
ElseIf provider = "Sprint" Then
  provider = "messaging.sprintpcs.com"
ElseIf provider = "Verizon" Then
  provider = "vtext.com"
ElseIf provider = "Alltel" Then
  provider = "message.alltel.com"
Else
  'catch all to do  nothing
End If
'Sending the Text
sendGMail number & "@" & provider, "Tracking Numbers", "Tracking Numbers: " & trackingNumbers

End Sub
```

Each provider has a different address for sending the messages:

- AT&T: @txt.att.net
- T-Mobile: @tmomail.net
- Sprint: @messaging.sprintpcs.com
- Verizon: @vtext.com
- Alltel: @message.alltel.com

In this case the problem is not the logic to send to the correct provider. The project is hindered by field representative's willingness to contribute information and the next major challenge. Today, as it stands I'm unable to send emails.

Temporary Show Stopper

The current show stopper for this project is the inability to send a Gmail as we did in class. I currently have little information on the problem except that the application is unable to create the connection that would send the message.

All of the documentation appears to have the listed Gmail server as smtp.gmail.com with port 465 as the SSL port. The connection currently takes no time to fail. Without documentation for resolutions I'm left to hope I can come up with a solution soon.

This problem also effects the requirement to send text messages. (listed above)

Potential Alternative

I may be able to solve the problem by using Exchange. Documentation is readily available on sending Outlook messages in VBA. Outlook sets a limit to the number of these messages at 500. This will extend the duration of the project beyond the ISYS 540 class.