

Danny Patterson
April 7, 2010

Excel Project Write-up

Executive Summary:

MemoSend has been created to fit the needs of multiple organizations and businesses, as well as small groups or families wanting to use it for personal reasons.

MemoSend is a program coded in Visual Basic for Applications (VBA) designed to enable users to send out mass text and email messages. The idea was thought of after a realization that church groups continually perform the same regular and inefficient routine tasks of cutting out hand-out flyers and physically walking around to every apartment building to deliver them on EVERY door. This is time consuming, not to mention that not every roommate even sees the announcement before the event expires.

This program enables users to create & save lists of contact groups and then pull up saved lists from the past to send messages to. The saved lists are stored as text files within the folder or path that the workbook is stored under. Phone number format verification testing is performed and adjusted appropriately so it may be used to identify the initial phone carrier (AT&T, Verizon, Sprint, ect...) the device was purchased with. Logical statements are thrown throughout to identify the specific addresses needed in order to send a text message via a simple email message.

Implementation Documentation:

When I began tackling this assignment, I started out with a basic understanding of the fundamental concepts from one of our class lectures on how to send a single text and email message. I initially designed the user interfaces which consisted of three simple user interface forms:

1. User Login Page
2. Select Recipients Page
3. Compose Message Page

1. User Login:



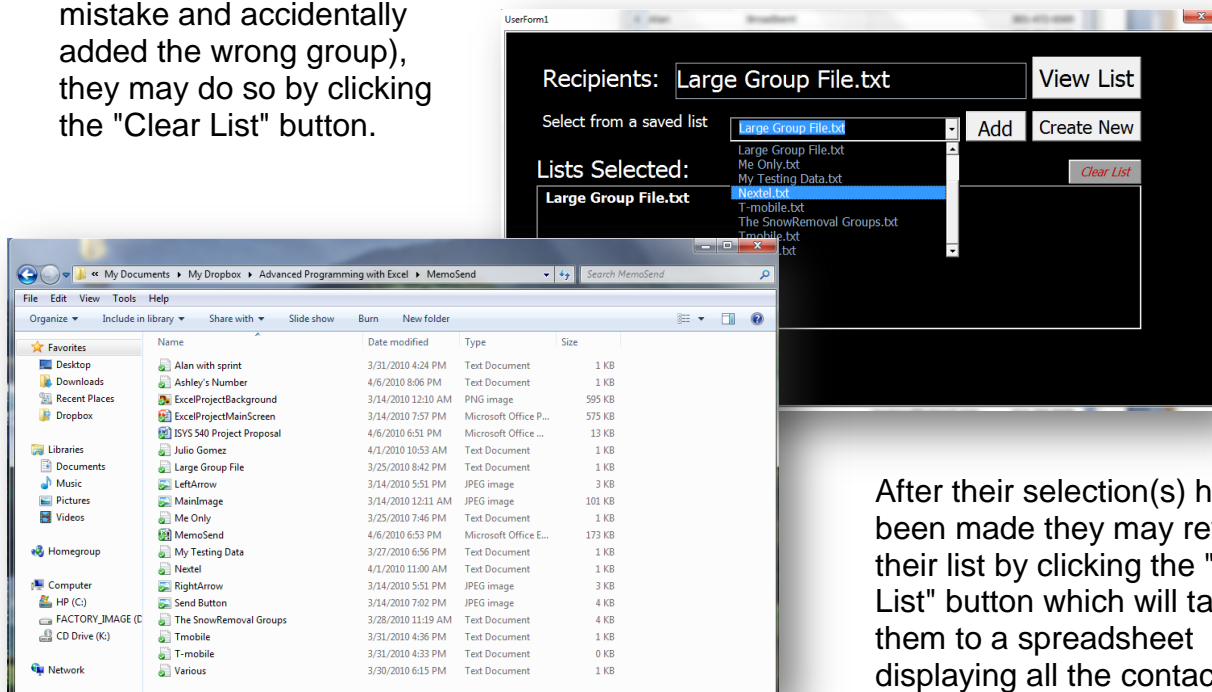
The first form consists of a user login page. I went out to the internet and pulled some images to put together a background to suit the theme of MemoSend. Birds flying to deliver messages to phones and computers receiving email. MemoSend is currently configured to accept email from Gmail, Yahoo, and AOL accounts. This will be improved in the near future to accept hotmail, live, and other common email accounts.

After the user inserts their email and email password values, they will proceed onto the next form by clicking either the blue arrow image or the red "Continue >>" label.

2. Select Recipient Page:

The recipient page allows users to select a single group or multiple groups which have previously been saved using MemoSend. When the form is initiated, a recursive method is run to identify all text files (.txt) within the directory path of the saved workbook.

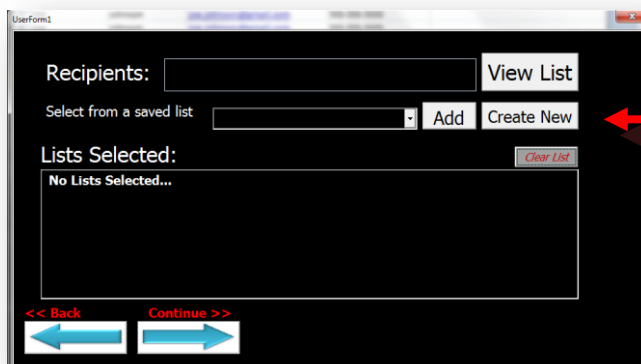
If it's the users first time using the program and they don't have saved data yet. The user would click the "Create New" button which would take them to a sheet where they would be able to enter in their group contact information and be able to save it, making it accessible on this page for future use. If the user wants to clear their selection (if they made a mistake and accidentally added the wrong group), they may do so by clicking the "Clear List" button.



After their selection(s) have been made they may review their list by clicking the "View List" button which will take them to a spreadsheet displaying all the contact information for everyone in the group(s), as well as their

phone carrier providers. (see section Saving Data to learn how the carrier information is obtained). There are controls on every spreadsheet to take the user back to the

Recipient Page. Once the group(s) have been selected and added, the user then proceeds to the third and final page.



3. Compose Message Page:

The compose message phase allows the user to enter in their memo. They may choose to write up a single memo and use the text for both their email and text messages, or they may choose to write separate messages so they could elaborate more in the email message box about their particular memo. The users email is pulled in from the *login page* and the selected recipients field is pulled in from the *select recipients page*. The last step the user has to do is select how they want to send their message(s), whether it would be by text only, email only, or texts & email by simply selecting one of the three radio button options on the page. See the "Sending Messages" section to learn what makes the magic work on how the program sends out its messages.

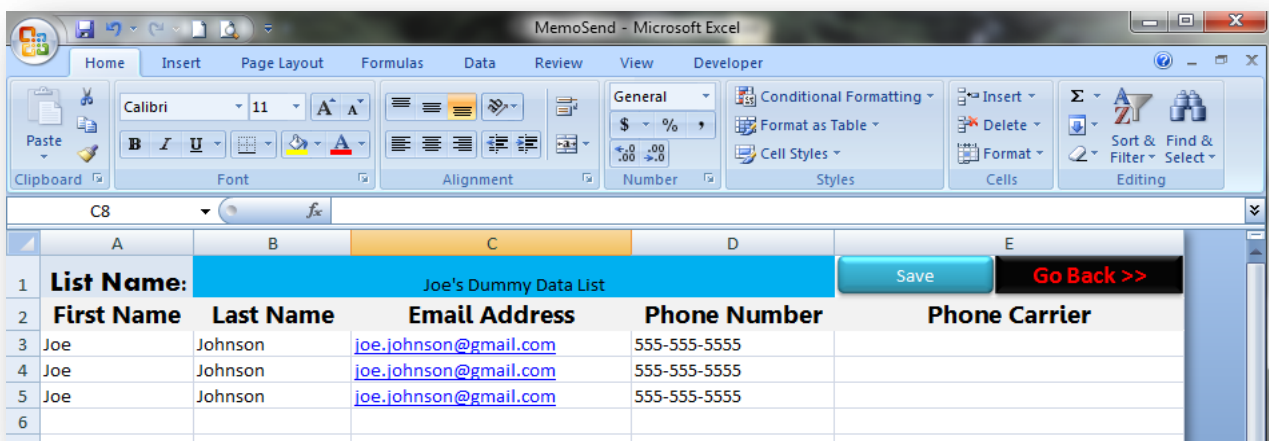
The screenshot shows a window titled "Generate Messages" with a dark background. At the top, there are two input fields: "User Email:" containing "dan42386@gmail.com" and "Selected Recipients:" containing "Large Group File.txt". Below these are two columns for composing messages. The left column is titled "Text Message:" and has a "Subject:" field with "Meeting Time" and a text area containing "We're meeting tonight at the Tanner at 6:00pm to go over our group project. See you there!". The right column is titled "Email Message:" and has a "Subject:" field with "Meeting Time" and a text area containing "We're meeting tonight at the Tanner at 6:00pm to go over our group project. See you there!". Between these two columns is a checkbox labeled "Use Same Text" which is currently checked. At the bottom, there are three radio buttons: "Send Texts Only", "Send Texts & Email", and "Send Email Only". The "Send Texts & Email" option is selected. At the very bottom, there are two large blue arrows: a left-pointing arrow labeled "<< To Recipients" and a right-pointing arrow labeled "Send >>".

Saving Data:

Once the user has entered the information for their newly created group, and has clicked the "Save" button, the following things will occur:

1. The program checks to see and alert the user if they have forgotten to enter in a name for their new group. The name of the group will be the name of the file saved with all the information on it. The code will break if the name is blank so the user may then enter one in.

I have entered some dummy data in the image below to illustrate what an example would look like. The only required fields are the first name, email address, and phone number.¹



2. The program begins to examine the cell number in column D. Since there are many valid ways to enter in a phone number (i.e. (555)555-5555, 555-555-5555, 1-555-555-5555, and 5555555555) in the US, the program will manipulate the number so it's in the format of 555-555-5555. The image on the following page shows the code for manipulating the entered-in number.

¹ If you already know ahead of time that you're only going to be using MemoSend for sending text messages, then you may omit the email address information.

```
'this Sub procedure will put a variety of number formats into the ###-###-####
'Concepts acquired from: http://www.devx.com/getHelpOn/10MinuteSolution/20414
Private Function txtPhone_Validate(txtPhone As String) As String
    Dim strTemp As String
    Dim strPhone As String
    Dim intResult As Integer
    Const DefaultAreaCode = "703"

    strTemp = Replace(txtPhone, "(", "") ' this gets rid of the ( at the beginning
    strTemp = Replace(strTemp, ")", "") ' gets rid of the )
    strTemp = Replace(strTemp, "-", "") ' gets rid of any dashes (-)
    strTemp = Replace(strTemp, " ", "") ' gets rid of spaces

    strPhone = strTemp

    If Left(strPhone, 1) = "1" Then 'this means there's a one at the beginning of
        strPhone = Mid(strPhone, 2)
    End If

    If Len(strPhone) <> 7 And Len(strPhone) <> 10 Then
        MsgBox "Please enter a valid telephone number.", vbExclamation

        Stop ' this exits all code being executed
        Exit Function
    End If

    ' If the Entered number does not have an area code, add the default one...
    '
    If Len(strPhone) = 7 Then
        strPhone = DefaultAreaCode & strPhone
    End If

    txtPhone = Left(strPhone, 3) & "-" & Mid(strPhone, 4, 3) & "-" & Right(strPhone, 4)
    txtPhone_Validate = txtPhone

End Function
```

3. Once the number has been formatted MemoSend opens up internet explorer and sends the number to the site "www.reversephonedirectory.com" where then the page that is returned is pasted into excel, searched for the phone carrier information, and copied over to the original sheet.
4. The sheet with all the information extracted from the site is then deleted and the program checks to see if there is more data in the next row down. If there is, it continues with steps 1-4 until all rows are accounted for.
5. A simple find and replace function is called to clean up some of the text on the page so the carrier info column doesn't contain the word "Carrier: " in it.
6. A text file is then created with the group list name as the file name. Each row is then iterated through and for every row of information, a line of that information is created with a colon put in between every cell's information. This is used to identify where new data begins and ends when reading back in the files.
7. The sheet with the new data just saved is then copied over to the "*All Contacts*"² page and the *New Contacts* sheet is cleared.
8. A message box appears to inform the user that the list was saved successfully.

The user then returns to the *Select Recipients* page and may select the list that they just created.

² The *All Contacts* sheet is populated by parsing in all the text files within the directory of the MemoSend program as soon as the file opens.

Sending Messages:

Once the user has entered their login credentials, selected a list or multiple lists, and drafted their message(s), they will click on the send button which will perform the following functions:

1. The code identifies which option was selected by the user (Texts only, Email only, or Texts & Email) and is directed to its corresponding method.
2. Information is gathered from the *Login Page*, *Recipient Page*, & the *Compose Message* page consisting of:
 - The email address
 - The email password
 - The subject line of the message
 - The body of the message
3. The length of the accumulative list is determined and a loop is set up for its corresponding size
4. The email and/or phone carrier information is gathered from the *Sel. Distribution List* page³
5. Logical statements are run against the phone numbers & carrier information to:
 - Remove all previously formatted dashes
 - Identify the proper address needed to send a text message based on the listed carrier information. (i.e. Verizon Carriers require texts to be sent using the format: #####@vtext.com)⁴
6. The sendMail method is called and is passed all of the information gathered from steps two to five.
7. The Email being used gets identified via a string search and appropriate settings are set in regards to smtp ports to use and whether security needs to be enabled to send a message.
8. The method sends the memos and a message will appear if the message was delivered.
9. The next contact within the loop (listed in step 3) is then selected and steps 4-8 are repeated until every contact has been sent a message.

That's how MemoSend works and the magic behind the scenes.

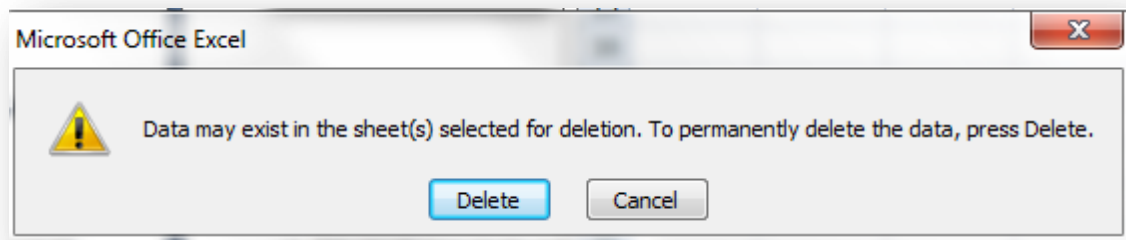
³ The Sel. Distribution Page is populated as soon as the user proceeds from the *Select Recipients* page on to the *Compose Message* page. The files selected are located and parsed into the sheet using the ":" as a delimiter.

⁴ Testing against New Cingular, Cricket, Cellular Inc, & nextel has not been fully performed. MemoSend currently works with Verizon, At&t, T-mobile, & Sprint.

Difficulties Encountered:

There were a few things that really stumped me which took a few hours to figure out. The main difficulties encountered are listed below:

- Phone numbers are entered in a variety of different ways. If a number was entered incorrectly or in a format that the website www.reversephonedirectory.com could not read, it would cause an error and crash the system. I needed to find a way for Excel to take a number and format it the way I needed it to be. I got something to work that involved a lot of code & various recorded macros, but when I did some more digging on the internet I realized that I was only accounting for one specific way a user could enter in a phone number and not looking at all the possible ways someone could potentially enter in a number. I eventually found a site with code that addressed my dilemma pretty well and tweaked it to a degree.
- When extracting information from the internet, I would create a new sheet and need to be able to reference it so I gave it a name and then attempted to simply delete over and over again so I wouldn't have sheet naming conflicts. The problem I found was that a pop-up message would always appear asking if I was sure I wanted to delete the sheet.



I didn't want the user to manually click "yes" over and over again for every record that the computer should know how to do automatically so I looked all over the place and eventually was able to find a solution which consisted of a couple simple lines of code to tack on near the call to delete the sheet.

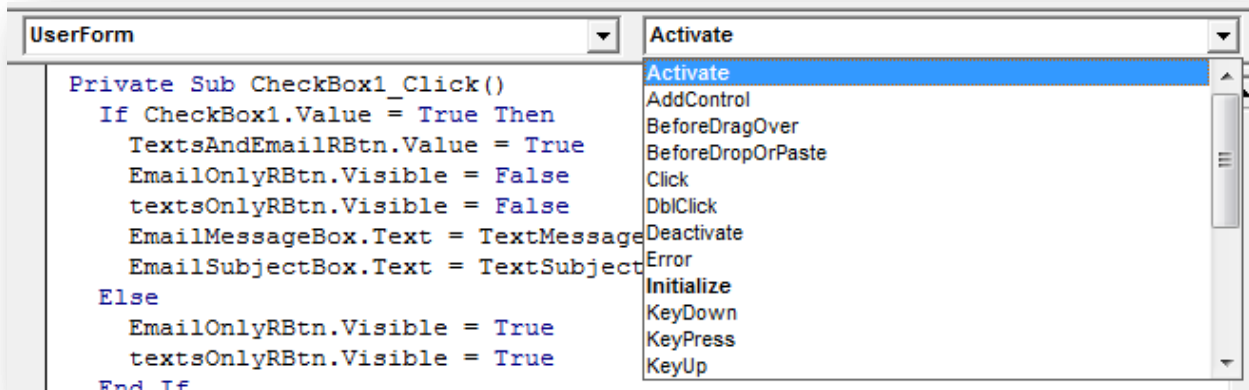
```
Sheets("toDelete").Select
Application.CutCopyMode = False
Application.DisplayAlerts = False
ActiveWindow.SelectedSheets.delete
Sheets("New Contacts").Select
ActiveCell.offset(1, -1).Select

End Sub
```

- Configuring the email settings for the various email clients would typically always vary. I did a lot of searching on the internet for what the settings were but a lot of the posts I found still did not work. What ended up working the best for me was attempting to create an account on outlook and then go into the account settings to see what settings outlook chose to send email.

Concepts Learned:

I was able to learn a ton regarding what & how the various methods within both user-forms and the workbook would do. Once I identified the Activate method within a user-form and understood how it worked, I was able to quickly address a few problems I had sitting around and I could update each sheet regardless of what changes I had just made or created.



I learned a lot about cell phone providers in general as I was searching for all the various extension I would need to send a text message to someone based on their provider. I thought and figured there are only mainly three large providers, Verizon, At&t, and Sprint, and that everything else in the market belonged under one of their umbrellas to some extent or another.

Form controls took a while to figure out how to determine if something was selected. I learned the specific VBA syntax needed to identify if & which radio buttons (belonging to a specified group) was selected. I found it was quite a bit different than VB and Java and when I was able to get the functionality to work for VBA, it was a good day.

```
Sub identifyRBtn()
    If MessageForm.textsOnlyRBtn.Value Then
        SendTextMessages
    End If
    If MessageForm.EmailOnlyRBtn.Value Then
        sendEmailMessages
    End If
    If MessageForm.TextsAndEmailRBtn.Value Then
        sendTextAndEmailMessages
    End If
End Sub
```

Disclaimer:

As the program was being developed and tested, one thing caught my attention really quickly. The website www.reversephonedirectory.com will return valid information on the phone number ONLY if that number hasn't changed providers since they purchased the number. My family has switched from At&t to Verizon. The website still reads that my carrier is still At&t when in fact it's not. When attempting to send a text message to my phone with the At&t attachment (#####@txt.att.net), MemoSend will send the message which will then circulate the internet for approximately three hours and the user will be notified in their email account that the message could not be delivered to that address.

The user should always verify their contact's cell carrier information to ensure accurate delivery of their messages.

To fix this problem, the user needs to manually open the text file(s) containing the misrepresented info and change them to reflect the proper carrier for the number. Use only the following key names as they're the ones compatible with the program:

- Verizon
- T-mobile
- New Cingular
- Cricket
- Cellular Inc
- Sprint
- At&t