Executive Summery

The Business Problem

Last semester, I was called to be the activities coordinator for my church in my area. I quickly realized that there was a "business problem" that made fulfilling my responsibilities more difficult. The congregation where I attend church is unique compared to other congregations of our church in that it is made up entirely of college students. This means that the congregation is very transient with people coming and going all the time. As part of my responsibilities, I am sending out correspondence via email/text all the time. The "business problem" that I faced was how to keep up communications about activities with an ever-changing member base. In fact, the member base changes so much that almost weekly I received a new member directory with new members added and old members removed. In short, my email list was out of date almost every week. I needed a way to import the new list every time it was sent to me. In addition to this, I want to be able to send group texts with out having to enter in every number manually each time. Finally, I thought that I could increase turnout at activities by sending both emails and texts to everyone to make sure that everyone is informed about the activity. The solution that I built solves all of these problems by allowing me to import the new directory into Excel and send batch emails/texts (or both) to either the entire list of members or certain people that I select. In short, my business is a combination of data importing and batch messaging and while this solution solves my business problem, I believe that it could also be easily modified to help anyone who is in need of an efficient way of sending batch emails/texts to lists of subscribers, registrants, students, etc.

System Overview

As stated before, my system allows for a comma-delimited .txt file (this was how my list was formatted, but other options are available with slight modification of the code) containing user data to be imported in to Excel. Once imported, the data is automatically organized in to a table with headers at the top for first name, last name phone number, email and mobile provider. Once data is imported, the system contains three forms that allow the user to work with the data. The first form is the Send Form, which allows the user to select both email addresses and phone numbers to send emails/text too. If the user wants to find a specific member or group of members, you can access the Find Form from the Send Form by click the respective "Add Phone Number" and "Add Email" buttons. This takes you to the Find Form where you can search by all or part of the first or last name of the person you are attempting to find. Once found the form lets you add the personal information to either the email or phone list. Finally, an Add Form has been included, which allows the user to add a new member to the table if you don't yet have the new file containing the data for that person. More in-depth information about the implementation of these forms is included below in the implementation documentation section.

Last but not least, one unique feature of this system has to do with its ability to send text messages. In order to send a text message from Excel, the user needs to know the current mobile provider of the phone number that will receive the text. In creating this system, I assumed that users working with large amounts of data (i.e., long lists of phone numbers) wouldn't be able to know the provider of every phone number in their dataset. My system solves this problem with automation of the internet. The system takes each number in the dataset and runs it through whitepages.com's reverse phone search. This returns the provider for that given phone number and the system adds it to the last column of the table. Once the system is done importing the provider data for all of the phone numbers, the user can then send text messages to any of the phone numbers in the list. While some reverse phone number search engines are very inaccurate, whitepages.com's search returned the correct mobile provider for every phone number I tested, even phone numbers that have been ported from one provider to another. The result is a system that is very accurate for sending text messages. This feature will also be further discussed in the implementation documentation section below.

Implementation Documentation

The following section documents each component of this system and their respective roles.

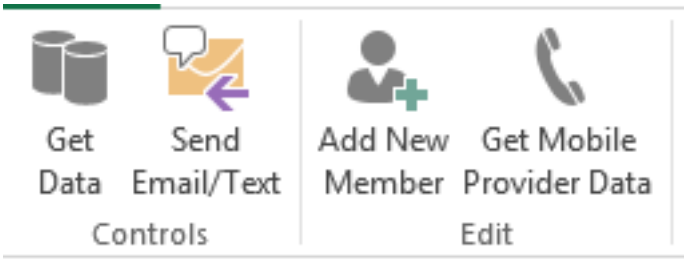


Figure 1 The four buttons make using the system easy and intuitive and the icons let the user know the general function of each button.

Upon opening the Excel file, the user should immediately notice that a new tab, "Send Email/Text", has been added to the Excel ribbon. Clicking this tab will reveal a set of four custom buttons (see Figure 1) that provide for easy, intuitive control of the system. Using these buttons as an outline, a walk through of the entire system follows.

Get Data

In order to send batch emails/texts, we need to import our list of emails/texts into Excel. The "Get Data" button provides this function. The "Get Data" button allows the user to import a comma-delimited .txt file into Excel. For an example of a comma-delimited .txt file, see Figure 2.

```
Miles, 801-369-6441, bencmiles@me.com, Ben
Miles, 801-369-3225, kelsiesmith@gmail.com, Kelsie
```

Figure 2 This is system is designed to read and import text files formatted like the example of above into Excel. Other formats are possible with slight modification of the code

When clicked, the "get data" button will prompt the user with a normal Window's dialogue box that allows the user to select the .txt file that they wish to be important into Excel. Once the user has selected the file to be imported, Excel reads each line and

inputs the first name, last name, phone number and email address of each line into a table (see Figure 3).

First	Last	Phone #	Email	Mobile Carrier
Anthony	Abbott	707-580-8371	anthonyabbott911@gmail.com	T-Mobile
Brad	Ackerson	503-505-0318	brad.ackerson@gmail.com	Verizon Wireless
Tyler	Allan	615-636-4550	tylerallan87@gmail.com	Verizon Wireless
Dexter	Allred	509-989-0449	allreddexter@gmail.com	AT&T Wireless

Figure 3 An example of the table that results from importing a comma-delimited .txt file.

This table is a list object named “eqdata.” Giving this object a name makes it much easier to reference this object later in the code. The user should also see that the active sheet has been named “eqData” as well.

Once Excel is done importing the .txt file, Excel will automatically begin importing the mobile-provider data for each phone number in the table. Excel accomplishes this task by taking each phone number and concatenating it onto the end of the address, "http://www.whitepages.com/phone/", and retrieves the inner html code containing the mobile provider for the respective phone number. The provider is then put into the table in the same row as the respective phone number. This system uses the agent module set up from class to access the internet and retrieve the data.

While this system can import the mobile provider for over 100 different phone numbers at a relatively quick pace, it does take more time than was required to import the .txt file. For this reason, I included a progress bar (see Figure 4) that is automatically initialized when Excel begins to retrieve the provider data from the internet. This form lets the user know that the system is downloading data and the progress bar shows the user how much of the task has been completed. Once Excel has completed the process of getting the provided data, the form is automatically closed.

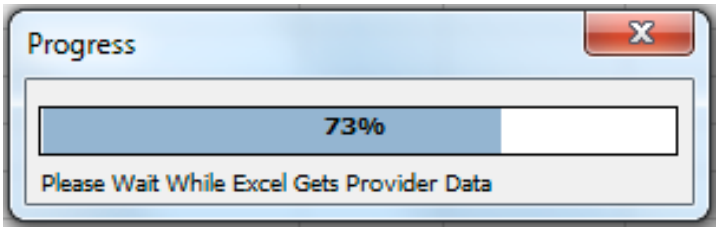


Figure 4 The progress form lets the user know that data is being retrieved and the progress of the task.

Send Email/Text

The next button is the “Send Email/Text” button. When clicked, the button calls the Send Form (see Figure 5) from which the user can add email addresses/phone numbers and send emails/texts. The following table provides a walk through of each aspect of this form.

The button for calling the Send Form.

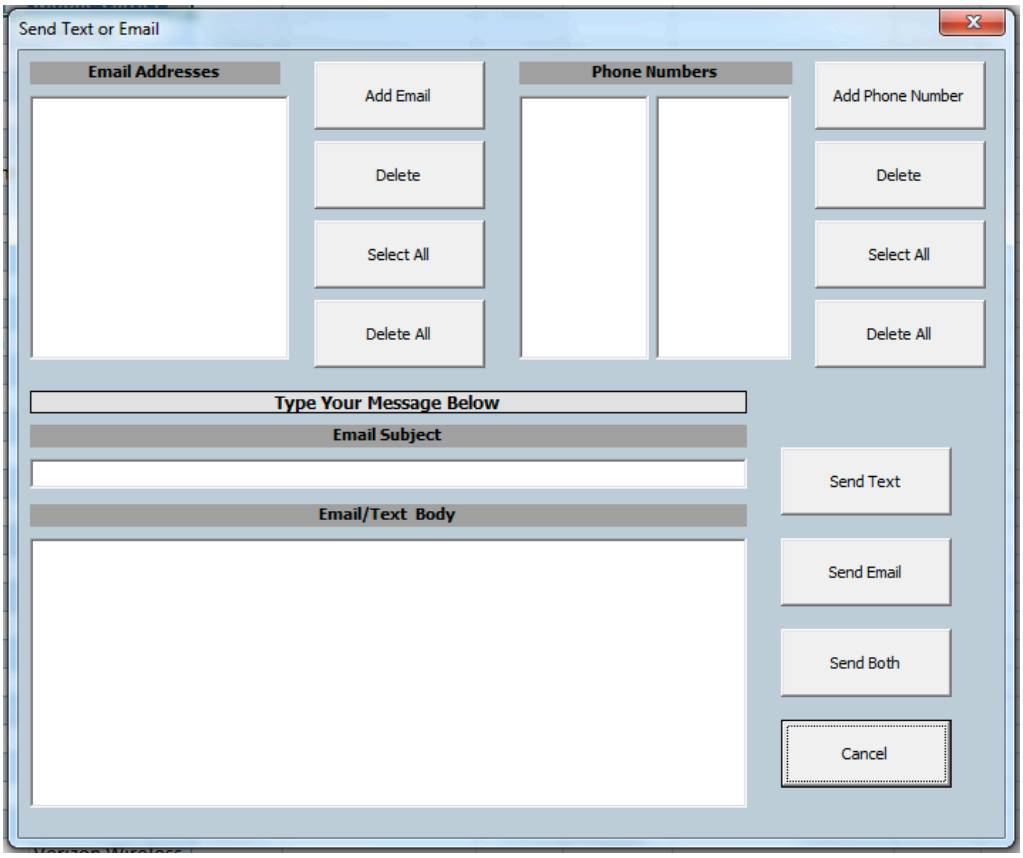
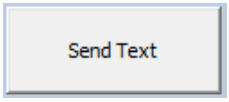
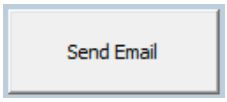
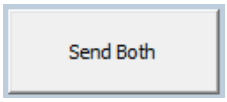
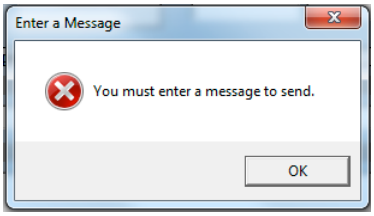
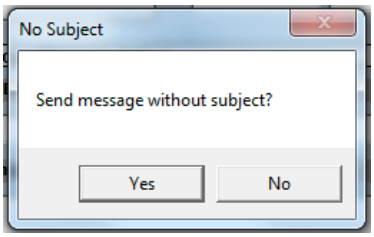
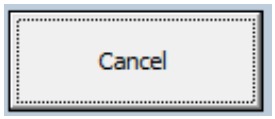
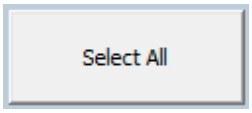
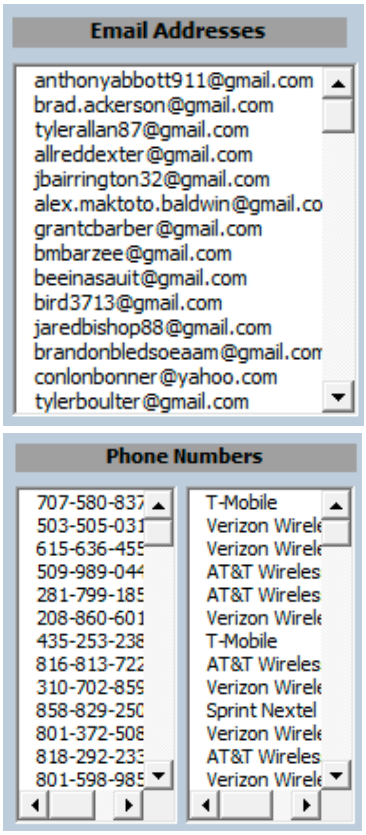
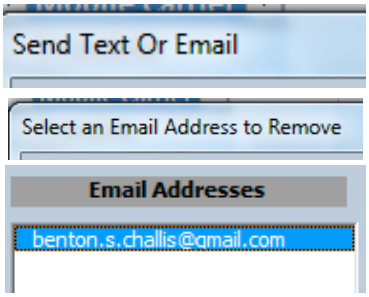
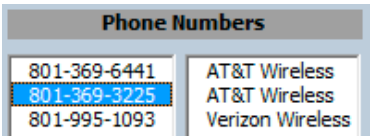
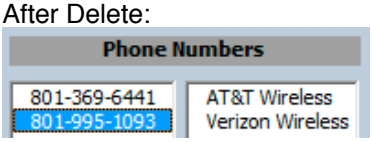


Figure 5 The Send Form provides the user with a simple and intuitive user interface for sending emails/texts.

Form Component	Description	Additional Screen Captures
  	These three buttons do exactly what they say they will do. The user can send a text, email or both at the same time. They will send what is written in the subject and body text boxes (see Figure 5). If the body text-box is blank the user will be asked to enter a message. If the subject text-box is blank the user will be give the option to send a message without a subject. This was included because you may not want a subject when sending a text. Once clicked they will close the form. If the user clicks one of these buttons without having added and email address or phone number, the form will prompt the user to first add an email or phone number.	 

Form Component	Description	Additional Screen Captures
	This button simply closes or unloads the form.	
	This button was included in case you want to send a batch message to either all of the emails or all of the phone numbers. Clicking the respective "Select All" button will add all of the email addresses to email list box in the table or all of the phone numbers and their respective providers to the phone number list box. The list boxes allow the form to use the information contained in the boxes when sending messages.	

Form Component	Description	Additional Screen Captures
Delete All Delete	The “delete all” button will delete all of the values in the respective listbox if the user decides he wants to quickly delete a long list of phone numbers/email addresses. The “delete” button allows the user to delete specific email addresses/phone numbers. The user can do this by selecting the entry that the user wishes to delete and clicking the “delete” button. For the phone number list box, this will also delete the mobile provider associated with the phone number that the user selected. If the user clicks the button without first selecting a entry, the form beeps and the form captions changes from “Send Email or Text” to “Select an Email Address (or Phone Number) to Remove.”	 Selected items will be highlighted in blue. Before Delete:  After Delete: 
Add Phone Number Add Email	These two buttons both have the same function. They allow the user to add a specific phone number or email address by calling the Find Form and letting user search the table for the member whose email address or phone number the user wishes to add. The Find Form is discussed in further detail below.	

Gmail Account

In order to properly use this system the user is required to have a Gmail account. This is necessary for sending both emails and texts as the system uses the user’s gmail account to send the messages. When sending a text you are actually sending an email to that person’s mobile provider who then forwards that email on as a text message to the mobile user. For example, if my phone number were 801-369-6441 and my provider was AT&T (a list of providers currently supported by this system is provided in the table below), this system would send the text message as an email to 8013696441@txt.att.net. AT&T would then send the email as a text to that number. For this reason, when the user clicks either the “send text”, “send email” or “send both”

button, the Send Form will close and the user will be presented with a form (see Figure 6) that asks the user to input his Gmail account credentials (i.e., username and password). This is a necessary role because in order for the system to work properly, the system must actually log into the user's email account. The Gmail Credentials From provides this functionality.

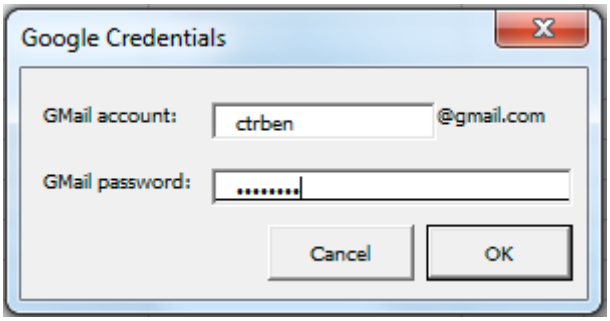


Figure 6 The Google Credentials Form allows the system to receive the user's Google username and password so it can send messages from the user's Gmail account.

Once the user's credentials have been entered and the user selects OK, the form closes and the system sends out the messages from the Send Form. Since this system was designed to send batch message, Excel will also generate message-confirmation tables for both emails and text sent (see Figure 7). These Tables are list objects as well with names so they can be easily referenced in the code for deleting or modification. For each batch, Excel will create a table that displays the email address or phone number in the first column and either the word "success" or "failure" in the the next column depending on whether or not the system was able to send the message. This is a crucial feature in batch messaging as it lets the user know who didn't receive the message.

Email	Status
bencmiles@me.com	Success
kelsiesmith@gmail.com	Success
Phone #	**Status**
8013696441@txt.att.net	Success
8013693225@txt.att.net	Success

Figure 7 Examples of the message-confirmation tables that are created after the user sends the messages.

Providers Currently Supported

at&t verizon Sprint U.S. Cellular alltel wireless T-Mobile boost mobile Virgin mobile

The Find Form

As stated earlier, when the user clicks either the “Add Email” or “Add Phone Number” button on the Send Form, the Find Form (see Figure 8) appears. In fact, the Find Form is actually part of the Send Form; it is just hidden until the user clicks one of the aforementioned buttons. Once clicked the Send Form automatically resizes to fit the Find Form and the Find Form’s components appear. The form’s caption also changes to “Find Member.” The form’s components are discussed in the table below .

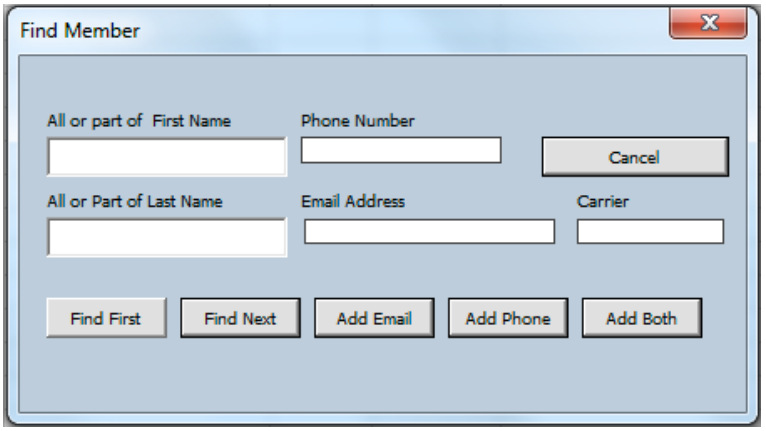
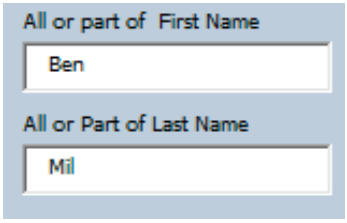
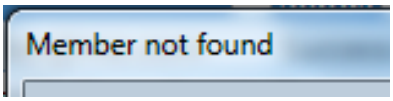
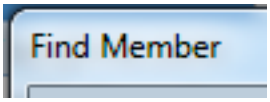
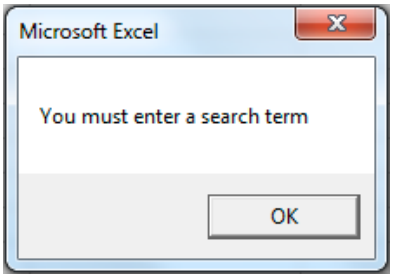
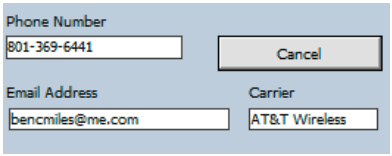
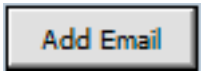
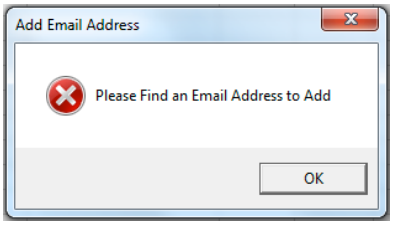
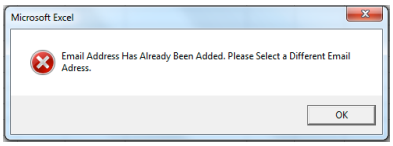
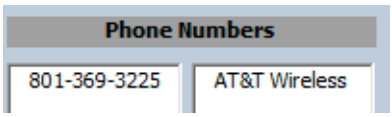
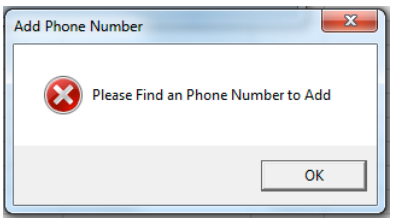
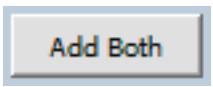
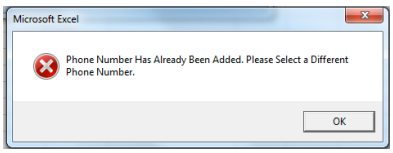


Figure 8 The Find Form allows the user to find, by all or part of a last or first name, a member from the table and add that member’s email or phone number to the Send Form.

Form Component	Description	Additional Screen Captures
	These two text boxes allow the user to search for all or part of a first or last name. If both texts boxes are filled only records containing both entries will be returned. Only one needs to be filled in order to run a search.	

Form Component	Description	Additional Screen Captures
Find First Find Next	The “Find First” Button will return the first record that matches all of the given criteria. If a record cannot be found, the form’s caption changes to “Member not found” and the form beeps. If the user changes the criteria and searches again the caption returns to “Find Member.” If the user selects the button without specifying any criteria, the user is presented with a message box informing the user that he must first enter search criteria. Subsequent records can be found by selecting the “Find Next” button. The search continues below the current active cell. Once a record is found, the phone number, email address and mobile provider are shown in the labels to the right of the text boxes. If the user cycles through the entire list, the caption will change to “Member not found” and the form will beep. The labels will also clear. If the user clicks the find next button again, the search begins from the top of the table and the caption returns to “Find Member.”	   
Cancel	Clicking this button simply returns the user to the Send Form. The form is resized for the Send Form and the Find Form is hidden. Any data in the labels and text boxes in cleared out.	

Form Component	Description	Additional Screen Captures
	These three buttons all have basically the same function with slight differences. Clicking the "Add Email" button will add the email address currently in the label to the email address list box on the send form. Clicking the button also returns you to the Send Form. If the user clicks the button without having completed a valid search, the user is presented with a message informing the user that he must run a search first. If the user tries to add an email that is already on the list the user is again presented with a message informing the user that that email is already in the list box.	 Email added from Find Form.  
	The "Add Phone" Button is essentially the same as the "Add Email" Button except that it adds the phone number currently in the label to the phone number list box on the send form. It also adds the mobile provider for the respective phone number to the carrier list box. The same input error message will show for this button as it did for the last button.	 
	The "Add Both" button adds both the email address and the phone number and carrier currently in the labels to their respective list boxes.	

Add New Member

When the “Add New Member” button is clicked, the Add Member Form is called (see Figure 9). This form was included in the system because it provides the user with the ability to add a new individual member with out having to reimport the entire data set. This is also a useful feature if you want to update your table but don’t have an updated .txt file to import. When the “Add” button is clicked, the system adds the information to the bottom of the table. Additionally, the code is formatted to accept a variety of phone number formats and will convert the phone number into the following format “XXX-XXX-XXXX” when the “Add” button is clicked. Clicking the “Cancel” button will close the form.

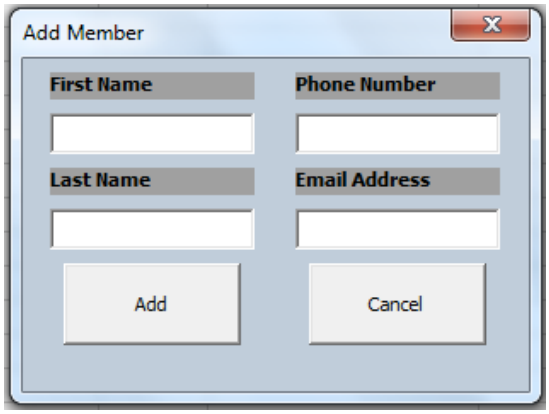


Figure 9 The Add Form allows the user to add a new member rather than having to reimport the entire database.

The button for calling the Add New Member Form.

Get Mobile Provider Data

The “Get Mobile Provider Data” button simply calls the sub procedure that retrieves the mobile provider for all of the phone numbers that don’t yet have a mobile provider associated with them. While this same sub procedure already runs when the user clicks the “Get Data” button, this button is important to the system because it allows the user to retrieve the provider for phone numbers that have been manually added using the Add Member Form. Without this button, the user would have to reimport the .txt file each time he wanted to run the procedure that retrieves mobile providers for the phone numbers in the table. The code is written in such a way that it will only retrieve mobile provider data for the phone numbers that don’t yet have providers associated with them.

The button for calling the sub procedure that retrieves the mobile providers for phone numbers in the table.

Difficulties Encountered

In the process of creating this system, I encountered three specific learning/ conceptual difficulties.

The first difficulty that I encountered dealt with having one user form that contained two different forms, which needed to be hidden and shown during different

parts of the code. This was difficult because I had to change both the height and the width of the from along with the location of certain buttons. Additionally, I needed to know what buttons would switch the form back to its original size. While this was difficult, it was a great learning experience. I learned that I could use the properties window in the VBA Editor to figure out the the location numerically where I wanted the objects to appear. (i.e., how far left and to the top I wanted the buttons to be located). I then wrote that number down and told the code that when called to put the button in that location. Examples from class and the internet helped to facilitate this learning process.

The second difficulty I encountered dealt with getting two list box entries to be deleted at once. You will notice on the Send Form (see Figure 5) that there are two list boxes under the phone number label. The first list box contains phone numbers and the second list box contains the respective mobile provider for each phone number. When the user selects a phone number and then clicks delete, I wanted the code to automatically remove the associated mobile provider as well. This was difficult because I couldn't tell the code to remove the selected carrier (i.e., using the .listindex property) because the user selected a phone number and not a carrier. I overcame this problem by setting a variable equal to the selected phone number. This gave me the location numerically of the phone number in the list. I then told Excel to delete the carrier located in the same numbered position in the carrier list. I have included a screen capture of this code below for further illustration (see Figure 10).

```
Private Sub CmddeletePhone_Click()
Dim listnumber As Integer
 If LstPhoneNumbers.ListIndex > -1 Then
    listnumber = LstPhoneNumbers.ListIndex
    lstCarriers.RemoveItem (listnumber)
    LstPhoneNumbers.RemoveItem LstPhoneNumbers.ListIndex
    FrmSend.Caption = "Send Text Or Email"
  Else
    FrmSend.Caption = "Select a Phone Number to Remove"
    Beep
  End If
End Sub
```

Figure 10 The fifth line of code here shows how the system is designed to delete the corresponding carrier of the phone number that has been selected and deleted.

The final and most difficult problem I encountered dealt with getting the mobile provider data from the internet for each of the phone numbers in the dataset. I originally planned to send texts by sending emails to each of the providers listed above for each phone number in the table. This way I could ensure that one of the carriers would eventually accept the message and forward on the text. However, I soon realized that

this was not only inefficient but also very abusive. I then came up with the idea that I could get the data from the internet by automating a reverse phone search from VBA.

This was a difficult aspect of the system to code because I kept having to rewrite the code to compensate for new problems that would manifest themselves each time I would change the code. My first iteration of the code for this section of the system used a very antiquated IE set up for using the internet to retrieve the provider data. This worked well but was very slow and often times would freeze from script errors because I was calling internet explorer for each phone number. This also ate up a ton of computer memory. At this point, I also realized the the reverse phone number search I was using, "www.fonefinder.net", was very inaccurate. I found the website "http://www.whitepages.com/phone/", which was much more accurate and rewrote the code to utilize this website. However, for some reason the IE set up that I had written didn't work with this website. I solved this problem by rewriting my code to work with the agent set up that we learned in class and this successfully worked with the White Pages' website. However, I still faced the problem that the system was very slow retrieving the data and would time out before finishing. I attempted to solve this problem by modifying the agent code to allow the code more time to load in order to avoid script errors. This didn't work. After speaking with my professor, I learned that unless I needed to log into a website, it isn't necessary to launch internet explorer for each number. I rewrote my code to concatenate the phone number onto the url of the website. Doing so, allows the code to access the inner HTML (where the mobile provider is listed) without having to open internet explorer for each phone number in the table. The result was a drastically faster system. While this part of the project was difficult, I learned a lot about the various functions of the agent module that will be useful for future projects. Examples include, decoding HTML code (using the .HTMLDecode function), moving to a specific section in the HTML code (using the .moveTo function and specifying the text to go to) and retrieving the provider name (using the .getText function).

Assistance

As stated above, I did receive assistance from Professor Gove Allen when I was working with the code I had written to download the provider data from the internet. I do want to state here, however, that I always sought out the professor's help after I had produce my own working code. I sought his help in order to learn how I could optimize (i.e., speed up, etc.) the working code that I had already written. Additionally, I utilized the "sendGmail" function and the "Web Agent" module to support various components of my system, which were both written by Professor Gove Allen. Finally, I also used the Progress Form (I slightly modified it to work with my code and system) to support my "GetData" module and the Gmail Credentials Form to support my Send Form. Both of those forms were created by Professor Gove Allen.