

Email/Text Message Sender System

Executive Summary

In the past I have worked as a referee for an indoor soccer facility. Every week I would get a text message from them asking me for my available schedule. Once they compiled the schedule, they would then send out each person's schedule via text or email. There are a number of problems with this system that I have tried to solve, mainly:

- The soccer facility doesn't have the ability to send out text messages because (like most businesses) their only phone is a traditional land line. The employees use their own personal cell phones.
- A different worker sends out the texts and emails each week. I never know who is texting me. The workers have to look up each referee's file and send out either an email or a text depending on the ref's preference.
- If a ref is slow to respond to the text, the employee who sent it might be off work and they would have to forward the text to whoever is working. This is a HUGE pain.
- There is no central database of referee's contact information.
- Some refs like to be notified via text, others prefer email. There is no system to do them both.

In addition to notifying referees, this system can also be used to notify captains of teams about their game time. Each week the managers are using their personal cell phones to send out texts and the owners don't reimburse these expenses. The managers are loving the fact they don't have to use their cellphones anymore.

Overview

The system I built gives the user access to a database in Excel. They can search for contacts, edit contacts, add new contacts, or delete contacts. Each contact has its own information that is stored (once it has been entered and saved). I believe this system solves all of the above problems.

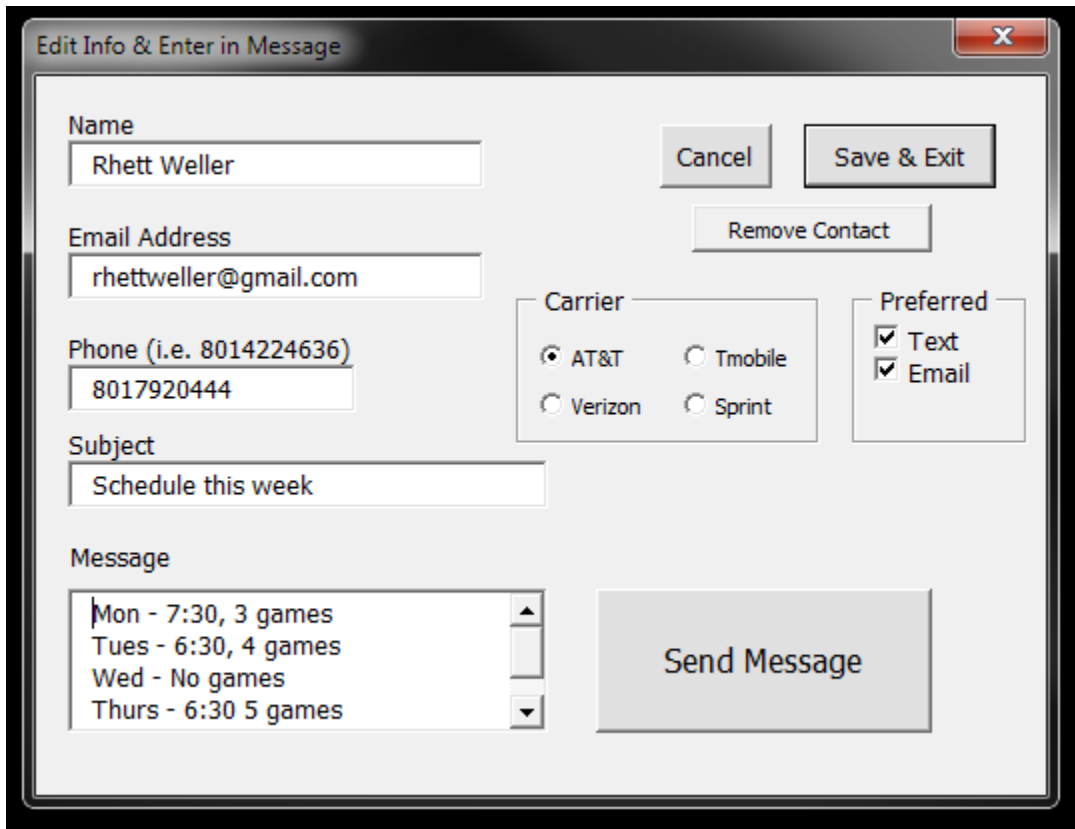
Implementation

The key I found for implementation is to map out exactly what problems I wanted to solve. I found myself getting distracted at times with extraneous features that really didn't solve the problem.

The data sheet is hidden so a user can't accidentally make changes. After clicking the "send messages" button that is shown when the user first opens the document, the user sees a find form:

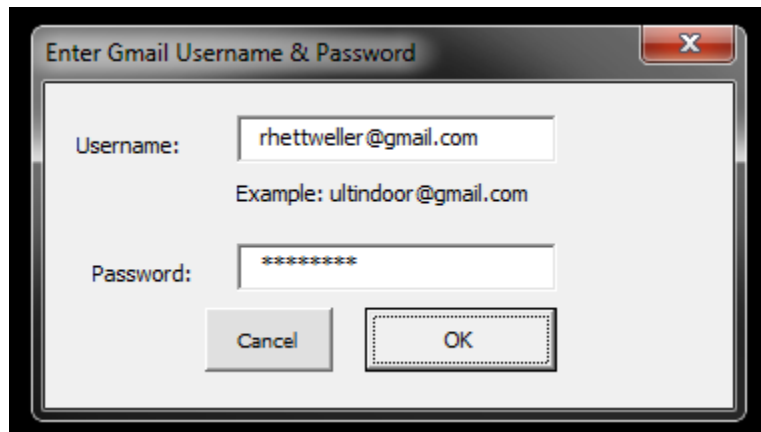
A dialog box titled "Find Recipient" with a close button (X) in the top right corner. It contains two text input fields: "Name" with the value "rhett" and another field with the value "Rhett Weller". Below these fields are three buttons: "Find First", "Find Next", and "Add New Recipient". At the bottom are two more buttons: "Send a Message" and "Close".

The user can enter a partial name of a person and the form will find the name. If no name is found, the user can click "Add New Recipient" to add a new person. This button and the "Send Message" button open another form (the "Add New Recipient" is blank).

A dialog box titled "Edit Info & Enter in Message" with a close button (X) in the top right corner. It contains several input fields and buttons. The "Name" field has "Rhett Weller". The "Email Address" field has "rhettweller@gmail.com". The "Phone (i.e. 8014224636)" field has "8017920444". The "Subject" field has "Schedule this week". The "Message" field contains a list of game times: "Mon - 7:30, 3 games", "Tues - 6:30, 4 games", "Wed - No games", and "Thurs - 6:30 5 games". To the right of the input fields are buttons for "Cancel", "Save & Exit", and "Remove Contact". Below the "Email Address" field is a "Carrier" section with four radio buttons: "AT&T" (selected), "Tmobile", "Verizon", and "Sprint". To the right of the "Carrier" section is a "Preferred" section with two checked checkboxes: "Text" and "Email". At the bottom right is a large "Send Message" button.

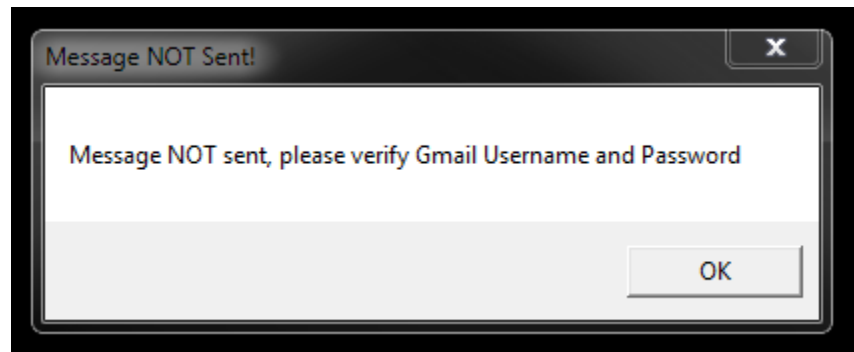
The soccer facility is EXTREMELY excited to have the option to send both a text and an email. Also the fact that the form loads with all the prior information entered (including subject and message) was valuable because the schedule only varies slightly each week.

After clicking send, a form asking for a Gmail username and password pops up:



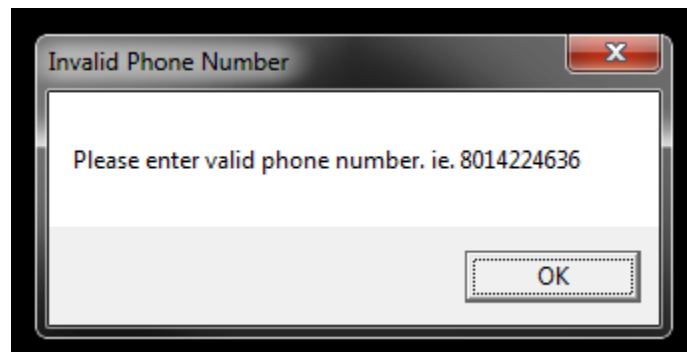
A Windows-style dialog box titled "Enter Gmail Username & Password" with a close button (X) in the top right corner. It contains two input fields: "Username:" with the text "rhettweller@gmail.com" and an example "Example: ultindoor@gmail.com" below it; and "Password:" with a masked password "*****". At the bottom are "Cancel" and "OK" buttons.

If the wrong username and password were entered into the form, this message box pops up:

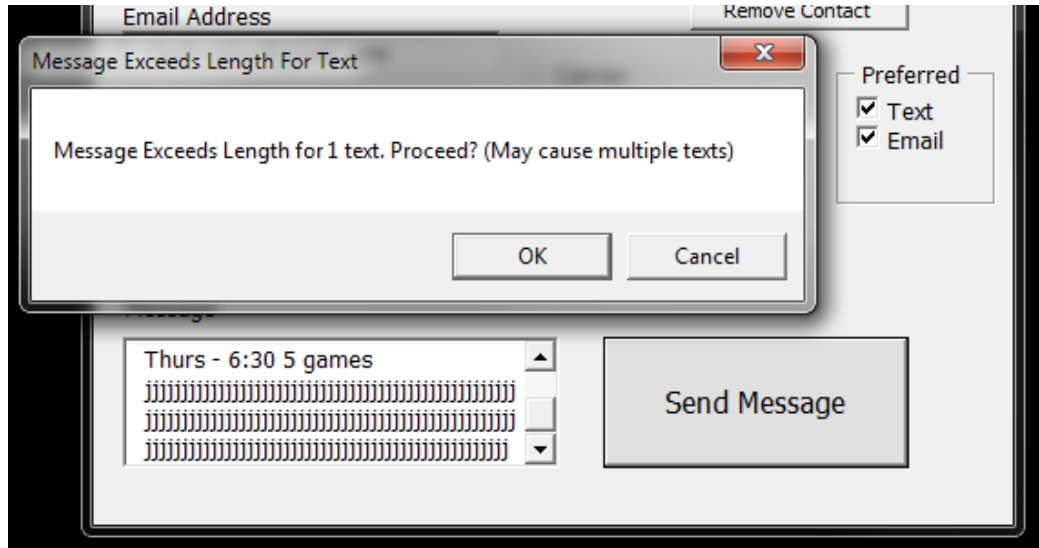


The code is the sendGmail Boolean that returns a true/false value and the msgbox pops up if the code returns FALSE. Once the correct information is entered, the messages are sent and the user can search for a new recipient.

Another verification is for the phone number. By using an IF and IsNumeric function, a FALSE return alerts the user with another popup box:



Standard text messages are 160 characters. I implemented a character count check if the message is too long for 1 text message. The code also checks if the “preferred text” option is clicked. This prevents the warning prompt from popping up if the user is only sending an email:



Instead of using 160 as my character count, I used 132 as 28 characters are necessary to show who the text is from. Also, if the person responds to the text, it goes to the Gmail account for the facility. This allows any of the staff to check the email and respond accordingly.



Things I Learned

Not having a coding background, at first I was overwhelmed by the project. However, I found as I looked at other examples of code from class (and the internet), I found I could read through the code, figure out what it was doing, and then construct my own code to do what I wanted it to do.

Something basic that I learned (that experienced coders probably would scoff at me for) was defining variables to make the code cleaner and easier to read and construct. Here is an example of variables I defined:

```
Dim message As String
Dim subject As String
Dim email As String
Dim row As Integer

Dim ATT As String
Dim verizon As String
Dim sprint As String
Dim tmobile As String
Dim carrier As String

ATT = "@txt.att.net"
verizon = "@vtext.com"
sprint = "@messaging.sprintpcs.com"
tmobile = "@tmomail.net"
```

By defining each carrier as its own variable (with each having its own value) and then defining “carrier” as a variable, I was able have the code execute the different carrier’s depending on which carrier was selected.

Adding a new user was a great example of learning by looking at some other code we had done in class. I defined a variable, then assigned a value to that variable (in this case the last row with data), and then move one row down to add in the new contact info:

```
Private Sub cmdNew_Click()
Dim LastRow As Integer

LastRow = [A65536].End(xlUp).row

Sheets("Contacts").Cells(LastRow + 1, "a").Select

Load frmEdit
frmEdit.Show

End Sub
```

The frmEdit code was the part of the project where I spent most of my time. I learned how to control message boxes (i.e vbOKCancel), new functions (i.e. IsNumeric), and how to call a private sub procedure inside of another private sub procedure (i.e. the Call function). This is where I spent a lot of time analyzing other code and finding the cleanest and simplest way to get my code to work.

I experimented with a few other functionalities like having a sound play when a message was sent. But I took it out because it got annoying hearing that sound over and over. I think simplicity and cutting out functionality is just as important as having good functionality. “Know when to say no” helps keep the user interface clean and cut down on confusing when I had friends test my system.

Finally I learned the importance of a clean interface. I had many friends and family beta test my program and I found that ease of use and a clean simple interface were the most important items. One thing to out of the beta test was that people had a hard time reading the captions on the forms. Just by adjusting the font size it made my whole program a better user experience. I also discovered the “tab order” function and the default option in the VBA editor. By adjusting the buttons, I made the program that much easier to use.

Final Thoughts...

The indoor facility loves the program! As I did this pro-bono they also loved the price. However we are now discussing additional functionality they would like to have included, but this time they are willing to pay me. At times during the semester, VBA was difficult. But when I finally got my program to work for me (the middle of the night “YES!” moment), I was hooked. I’m looking forward to coding in the future.