Michael Vaughn
IS 520 Final Project Report
4/14/2015

Summary

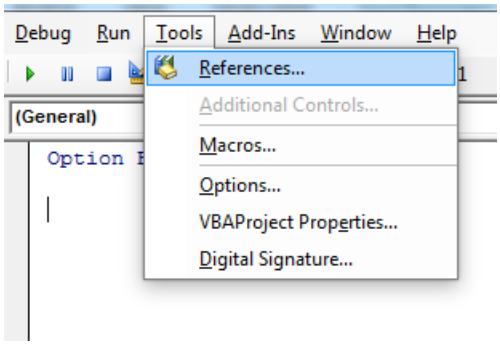
In office settings, efficiency is always a high priority. Within the Academic Advisement Center for the College of Family, Home, and Social Sciences at Brigham Young University, efficiency is in short supply. When I began working in the Advisement Center nearly a year ago, I immediately recognized that there were several processes that were extremely tedious and repetitive. Our office is utilized mostly by students who need assistance in scheduling classes for future semesters and who are looking for guidance in their career decisions. We have seven different academic advisors for students to meet with, depending upon their declared major or minor.

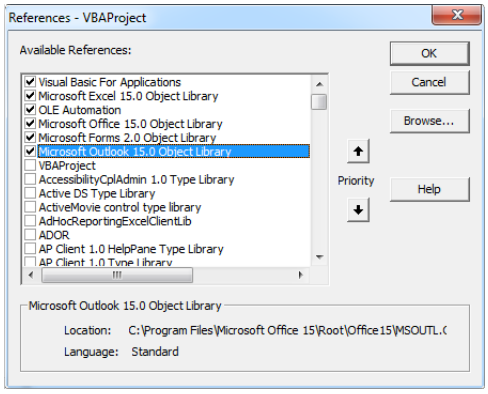
One of the most inefficient processes was that of sending emails to remind the students of their upcoming appointment in our office. The process to send just one email was long and tedious. A student staff member in the office would click on an appointment in Microsoft Outlook and copy the student's BYU ID number. With that ID number, that staff member would open a Web browser and sign in to their account on BYU's website. After navigating through a few pages, the copied ID number could be inserted into a text box on the page and, after submitting that page, the student's email address could be obtained. And finally, a new email from MS Outlook could be created and sent to the student reminding them of the appointment that they have made in our office. The email that was sent was identical for each student, just saved as an email signature to help the process move just a little quicker.

With this project, I have been able to automate that entire process, using MS Excel to do all of the grunt work. With the click of a button now, the calendar appointments for a specified day will be imported to Excel and the student's BYU ID number will be extracted automatically. An Internet browser will pop up and prompt the user to sign in to BYU's website and then will retrieve the email address of each student who has an appointment on the specified day. After collecting the email addresses, MS Excel will call on MS Outlook to send a personalized reminder email to each student, addressing the student by name and including the appointment date and time. What used to take a staff member nearly two hours per day to complete can now be done in less than two minutes.

Implementation

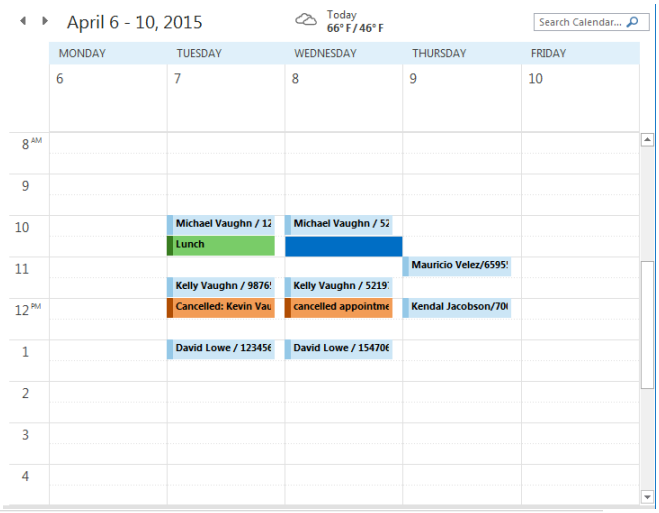
To begin trying to solve this problem, I needed to learn how to make the connection between Microsoft Outlook and Excel. I found that the first step in making this connection was to add a reference in VBA Editor. To do this, I selected "Resources" from the "Tools" tab and scrolled to find the Object Library for MS Outlook, as shown below.

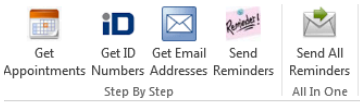




Once this reference is added, we can use objects and methods that are unique to the MS Outlook application. This allowed me to create my first sub procedure with the intent to access the calendar data from MS Outlook. Because I am unfamiliar with the objects and methods of MS Outlook, I did a lot of researching, reading and looking for examples online from others who have done similar things. I discovered that most people who use MS Excel to access MS Outlook must do the opposite of what I am trying to do. Nearly all the examples and information I could find were directed at people who were trying to take information from MS Excel and use it to create appointments on a calendar in MS Excel. The only examples that I found that seemed to be applicable to my desires were poorly coded and caused several problems each time I attempted to implement similar code. Finally, it dawned on me that I could "reverse engineer" several pieces of other's code and have a decent outline of my desired procedure. I was able to revamp the code and make it more efficient several times throughout the process as I added other sub procedures and refined my desired outcome.

This procedure, "GetCalData," now retrieves all the appointments scheduled on a user specified day from MS Outlook; if for one reason or another MS Outlook cannot open or if there are no appointments on this day, a message box will alert the user and exit the procedure. Otherwise the appointments are arranged in chronological order while separating





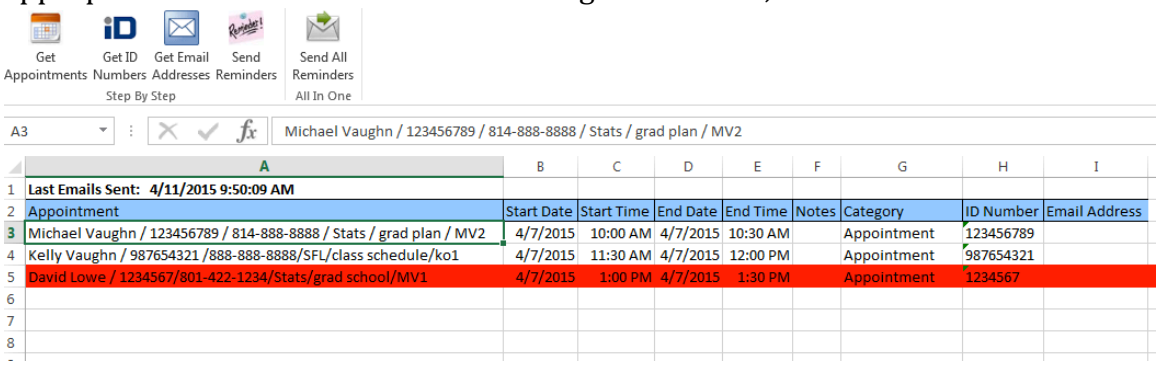
A8

	A	B	C	D	E	F	G	H	I
1	**Last Emails Sent: 4/11/2015 9:50:09 AM**								
2	Appointment	Start Date	Start Time	End Date	End Time	Notes	Category	ID Number	Email Address
3	Michael Vaughn / 123456789 / 814-888-8888 / Stats / grad plan / MV2	4/7/2015	10:00 AM	4/7/2015	10:30 AM		Appointment		
4	Lunch	4/7/2015	10:30 AM	4/7/2015	11:00 AM		Green Category		
5	Kelly Vaughn / 987654321 /888-888-8888/SFL/class schedule/ko1	4/7/2015	11:30 AM	4/7/2015	12:00 PM		Appointment		
6	Cancelled: Kevin Vaughn/111111111//OBHR/graduation/sc2	4/7/2015	12:00 PM	4/7/2015	12:30 PM		Orange Category		
7	David Lowe / 1234567/801-422-1234/Stats/grad school/MV1	4/7/2015	1:00 PM	4/7/2015	1:30 PM		Appointment		
8									

appointment details into different columns. The appointment information is arranged in the first column and includes details about the appointment, including the student's name, BYU ID number, phone number, and the purpose of the appointment. The next columns separate the date of the appointment and the starting and ending times. If there is an extensive amount of information about the appointment, the column labeled "Notes" will carry that information. A column labeled "Category" is created to help filter the appointments for each advisor. Once each column is created, the data from the MS Outlook calendar can be inserted below. An example of what the MS Excel worksheet might look like is above.

Once we have all of the appointment information imported into MS Excel, we want to filter out any calendar items that are not student appointments, whether those are things like "Lunch" or a student's appointment that has been rescheduled or cancelled. Up to this point, our list of imported appointments contains everything on an advisor's agenda, but many of these will be unimportant to us. Because each of these types of appointments is categorized differently, this program can delete any calendar item that is not an active appointment. The program accomplishes this by deleting all entries that are categorized as anything other than "Appointment."

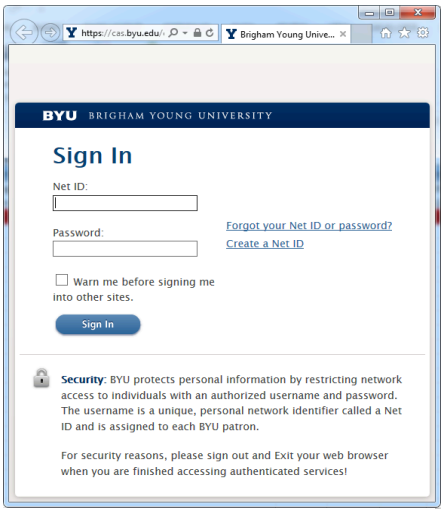
The next step of the whole process is to obtain the student's BYU ID number. This number is always directly following the student's name in the appointment information, separated from the other appointment details by forward slashes (/). Once we have isolated what we anticipate will be the student's ID number, we strip it of any non-numeric characters (hyphens commonly separate several of the digits of an ID number) and confirm that the length of the clean ID number is equal to nine. If there is no valid ID number, whether what we have obtained is too long, too short, or non-existent, we highlight the entire row so that the user can address those particular cases individually. The ID number for each student is then inserted in the appropriate row under the column heading "ID Number," as shown below.

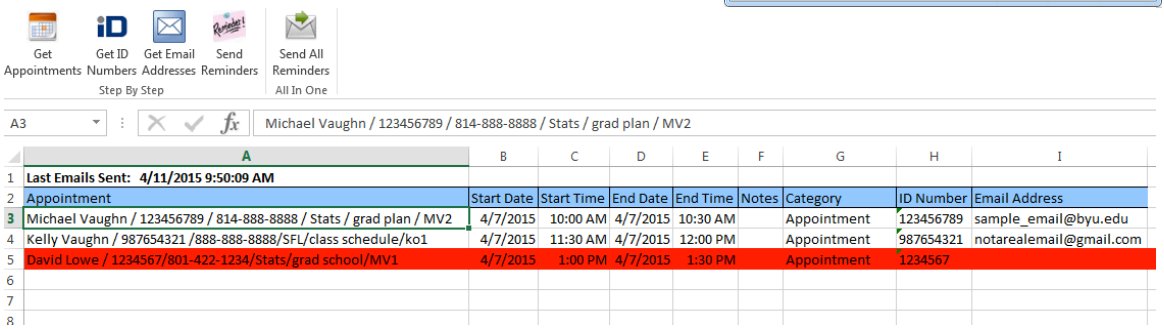


	A	B	C	D	E	F	G	H	I
1	**Last Emails Sent: 4/11/2015 9:50:09 AM**								
2	Appointment	Start Date	Start Time	End Date	End Time	Notes	Category	ID Number	Email Address
3	Michael Vaughn / 123456789 / 814-888-8888 / Stats / grad plan / MV2	4/7/2015	10:00 AM	4/7/2015	10:30 AM		Appointment	123456789	
4	Kelly Vaughn / 987654321 /888-888-8888/SFL/class schedule/ko1	4/7/2015	11:30 AM	4/7/2015	12:00 PM		Appointment	987654321	
5	David Lowe / 1234567/801-422-1234/Stats/grad school/MV1	4/7/2015	1:00 PM	4/7/2015	1:30 PM		Appointment	1234567	
6									
7									
8									

Moving on, we are ready to find the student's email address. The email address is available to us on BYU's website now that we have each ID number for the students. Running the next part of the code will bring us to a Web browser, where the user will be prompted to enter their BYU NetID and password, shown at below. This keeps everyone's information safe, both the user and the students who we will look up. Once the user has entered their correct credentials, the Web browser will become invisible, but it will still be operating. The program will send an ID number to a text box in the Web browser that allows us to search the University's archives

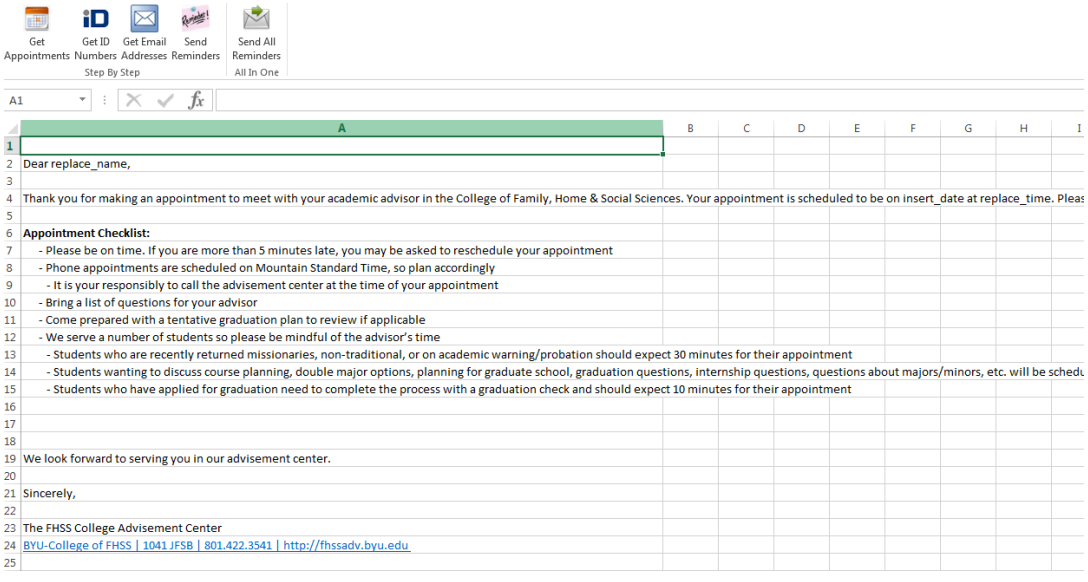
for any student's information. Once this has happened, the program will use the HTML source code to locate the student's email address. This email address will be inserted in the proper row under the column heading "Email Address." At this point, the program will continue to use the remaining ID numbers to obtain the rest of the email addresses. If, as in the example below, there is an ID number that does not fit the criteria of a typical student's ID number, the program will skip this row and allow the user to inspect the entry individually.





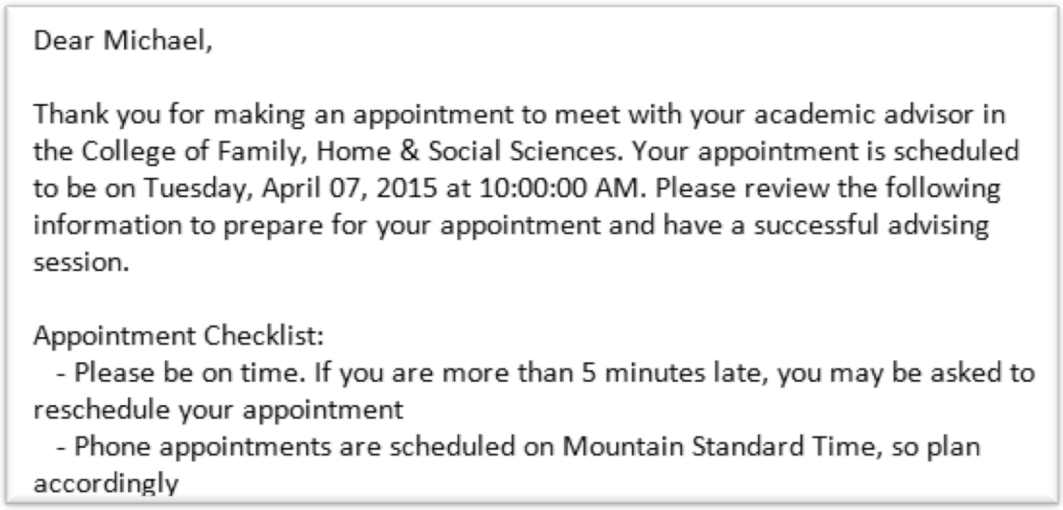
	A	B	C	D	E	F	G	H	I
1	**Last Emails Sent: 4/11/2015 9:50:09 AM**								
2	Appointment	Start Date	Start Time	End Date	End Time	Notes	Category	ID Number	Email Address
3	Michael Vaughn / 123456789 / 814-888-8888 / Stats / grad plan / MV2	4/7/2015	10:00 AM	4/7/2015	10:30 AM		Appointment	123456789	sample_email@byu.edu
4	Kelly Vaughn / 987654321 /888-888-8888/SFL/class schedule/ko1	4/7/2015	11:30 AM	4/7/2015	12:00 PM		Appointment	987654321	notarealemail@gmail.com
5	David Lowe / 1234567/801-422-1234/Stats/grad school/MV1	4/7/2015	1:00 PM	4/7/2015	1:30 PM		Appointment	1234567	
6									
7									
8									

Now that we have all of the email addresses, we can finally send an email. To send an email with MS Outlook from MS Excel, you must pass a few parameters to MS Outlook. First, the email needs to be given a destination, which is provided in the last column, "Email Address." Next, the subject line of the email must be created. Because this will be the same in every email we send, I hard coded the subject line into the program so that it will be constant every time an email is sent. This is a very rigid and inflexible way of inserting the subject line, but it will protect it from accidental user error. And lastly, the body of the email message must be created. To do this, I typed the message on a sheet titled "Email." The program creates this message one line at a time so that it is formatted properly in an email message.



	A
1	
2	Dear replace_name,
3	
4	Thank you for making an appointment to meet with your academic advisor in the College of Family, Home & Social Sciences. Your appointment is scheduled to be on insert_date at replace_time. Pleas
5	
6	**Appointment Checklist:**
7	- Please be on time. If you are more than 5 minutes late, you may be asked to reschedule your appointment
8	- Phone appointments are scheduled on Mountain Standard Time, so plan accordingly
9	- It is your responsibly to call the advisement center at the time of your appointment
10	- Bring a list of questions for your advisor
11	- Come prepared with a tentative graduation plan to review if applicable
12	- We serve a number of students so please be mindful of the advisor's time
13	- Students who are recently returned missionaries, non-traditional, or on academic warning/probation should expect 30 minutes for their appointment
14	- Students wanting to discuss course planning, double major options, planning for graduate school, graduation questions, internship questions, questions about majors/minors, etc. will be schedu
15	- Students who have applied for graduation need to complete the process with a graduation check and should expect 10 minutes for their appointment
16	
17	
18	
19	We look forward to serving you in our advisement center.
20	
21	Sincerely,
22	
23	The FHSS College Advisement Center
24	BYU-College of FHSS | 1041 JFSB | 801.422.3541 | http://fhssadv.byu.edu
25	

As shown in the image above, there are a couple places in the message that need to be edited before it can be sent to the student. We do not want to send the email addressed, “Dear replace_name.” The Advisement Center used to send generic email messages that were identical for everyone, addressing “Dear Student.” I felt that this was too impersonal for the student to recognize that we are working to help and support them the best that we can. So instead, I decided to use the appointment information from the calendar item to find the student’s name and send a personalized message saying, for example, “Dear Michael.” I also thought it would be a much more effective reminder email if it actually reminded the student when the appointment was scheduled. Rather than merely telling the student how to prepare for an advisement session with an advisor, this new program will insert into the email message the day, date, and time of the appointment. This information is drawn from the imported calendar information. If I had scheduled an appointment with an advisor in our office, the reminder email sent to me would look something like the following:



Dear Michael,

Thank you for making an appointment to meet with your academic advisor in the College of Family, Home & Social Sciences. Your appointment is scheduled to be on Tuesday, April 07, 2015 at 10:00:00 AM. Please review the following information to prepare for your appointment and have a successful advising session.

Appointment Checklist:
- Please be on time. If you are more than 5 minutes late, you may be asked to reschedule your appointment
- Phone appointments are scheduled on Mountain Standard Time, so plan accordingly

Having written all of the code to perform these tasks explained above, I modified the Ribbon in MS Excel. I created five buttons on a tab titled “Send Reminders.” When the first button, labeled “Get Appointments,” is clicked, an input box appears asking the user to specify the date of the appointments for which they would like to send reminder emails to. The program then retrieves the appointment information for the specified day. The second button, “Get ID Numbers,” determines where in the appointment information the BYU ID number is and extracts it, entering it in to the column labeled “ID Numbers.” “Get Email Addresses” is the third button and it uses the ID numbers that we recently extracted to find the email addresses of these students on BYU’s secure website. After the Web browser opens and the user signs into their account on BYU’s website, the browser becomes invisible and the program loops through each of the ID numbers until each email address has been obtained and entered into the last column labeled “Email Address.” Next, the user can use the fourth button, “Send Reminders,” to send all of the students a personalized reminder email from the information gathered in the previous three steps. The last button, “Send All Reminders,” is the all-in-one option that the user has, if they would like to

import the information, get ID numbers and email addresses, and send the reminder emails all at once. The process is exactly the same as the previous four buttons, but has combined everything to work as quickly and effectively as possible. The only downside to using this last button is that the user cannot interrupt the process to fix a mistyped ID number, for example. If the user executes each step individually, the user can manage any irregularity as it happens instead of after all is said and done.

Learning & Conceptual Difficulties

When I proposed this project to serve as my final project for IS 520, I understood that it would be difficult, but I certainly overestimated my abilities. At the start of the project, I would spend an hour or so at a time working and come away with what seemed to be nothing. I knew that it was possible to access MS Outlook with MS Excel VBA, but I was having a really hard time learning how exactly to do it. I tried reading and studying using online sources, as well as looking for examples of people who may have done similar things before. I was frustrated and losing hope when my good friend, David Lowe (who was also in this same class), suggested that I use bits and pieces from each of the readings and examples that I had looked for to help me build the procedure that I needed. This was a much-needed outside look that enabled me to start making progress.

Using ideas and tips learned in class, I was able to code most of the project rather quickly. I used two sub procedures to extract the student's BYU ID number from the appointment information and to remove any non-numeric characters (hyphens are commonly used to separate pieces of the ID number, but leaving these would have caused other problems in my solution. These procedures were not identical, but closely related to a process we made in class earlier in the semester.

One concept that was particularly difficult for me to fine tune was that of error catching. Having a program that depends on the user acting in a specific way requires a lot of checking and rechecking. One part of my code, when the Web browser prompts the user to enter their BYU NetID and password, was particularly difficult. If the user enters incorrect credentials or none at all, I was able to have the Web browser refresh and ask again for the correct information. In the case that the user changes their mind and decides not to enter their login information, I was able to successfully have the program terminate and alert the user that no email addresses were imported.

Another part of the code that is troublesome to me is dealing with the student's BYU ID number. If by chance the number is typed incorrectly, we could get the wrong person's email address, or it could produce an error saying that the ID number does not match any student. If the ID number is linked to a student's account that does not have an email address, this also causes problems in the solution. However, I was able to successfully avoid some of these concerns. Now, if the ID number is typed incorrectly and is not linked to a student at BYU, the program will highlight that student's appointment information and move on to the next appointment. If the student does not have an email address linked to the account on BYU's website, "No Email Address" will be inserted into the list of emails. This caused a bit of a hiccup later as I tried to send the emails, and there were some

values in the column of email addresses that were not valid email addresses. To overcome this challenge, I have the program search each item in the "Email Address" column and if there is not a "@" symbol in the string, it will not attempt to send the email.

Assistance

During the course of this project, I received various amounts and forms of help from a couple different people. As I mentioned earlier, my good friend, David Lowe, helped me by suggesting that I look at the problem from a different perspective and that allowed me to move forward a lot. I also used David as a "sound board" because it was easier for me to solve my own problems when I could vocalize them and hear myself think out loud. I also visited Dr. Allen on two separate occasions. On the first occasion, I came seeking a clarification and a deeper understanding of a class module that was introduced to us in class. He was able to explain several of its functions and it helped me to utilize it more efficiently in my project. The second visit came after I struggled for nearly an hour trying to get my ribbon modifications to work. It turns out, spelling is important when programming, especially when trying to reference a specific procedure.