BYU BAP Reporting Tool

Spencer Mecham

Executive Summary:

I built an updated reporting tool for the Beta Alpha Psi organization. The reporter is in charge of reporting the minutes for the chapter, both professional and service. Because many members do not remember to report their minutes, and it is difficult to remember which of the recurring weekly meetings they attended, we provide a card swipe service. We swipe their BYU ID cards and match the ID number to their name.

We had an excel workbook that took care of this function, but it was based on access to the BYU database and other websites that have since been made private (and we were not granted access). Thus the only functionality that remained was to create a new event (tab) and to pull the student info from the first tab in order to update the card swipes for the active tab. Because the BYU database was now off limits the reporter had to manually copy student ID numbers and names to the tool. Then the new users had to be added to Canvas before the reporter could (manually) give minutes for any event they attended.

I wanted to revamp the tool to do two more things; import ID numbers (with auto update of all data) and upload the data in the tool directly to Canvas. To import ID numbers the tool needed to access the results from a Google form; then compare those results with what was in the tool to determine if there were new people. Then, if there are new people, add them (both in tool and on Canvas) and update all current events. The Upload minutes button would write to a CSV file then upload the file to Canvas.

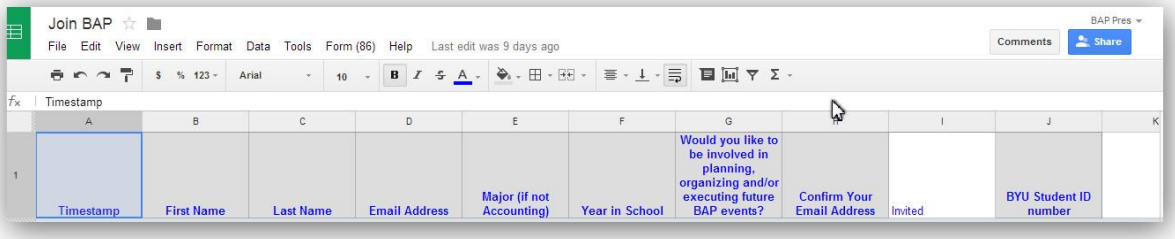
Implementation:

When I first presented this project Professor Allen suggested that I use the API for Canvas. I did try to implement what we learned about dealing with other databases to this project, unfortunately there is no VBA object for speaking directly to Canvas or the Red Ruby which Canvas was built on. However, while working on a project of smaller scope for my IS 515 class I discovered that I could manipulate Canvas using the Agent class module we were given in class. I proceeded to work through the various parts of the project using this and I believe I have created a satisfactory product.

Any URLs that won't change (log in page for Canvas) are hardcoded into the macro. For changing URLs, such as going to the gradebook for the current year, I set a reference to a named cell. In the instructions sheet (first sheet in workbook) the cells have instructions next to them explaining where to get the URLs from. This will allow it to function each year, no matter what name is given to the course.

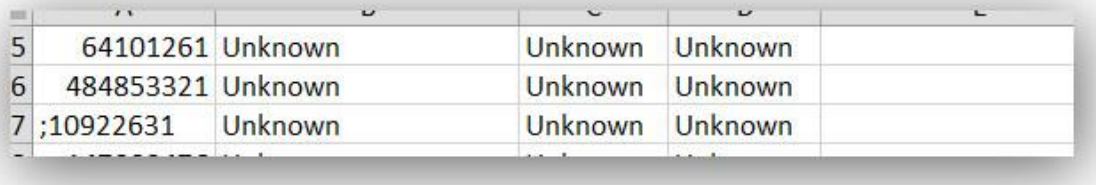
The first step was to pull the working subs out of the old workbook. I'll mention that portion more fully in the assistance section.

Next was creating the subs that would perform the "Import ID" process. The Google form asks the students for the data we need to do card swipes and add them as members.



By selecting file and choosing to publish the results (and making the results public), a hyperlink is created that will send the current data. While this Google form has not changed in the three years I have been reporter, I decided to put it in a cell on the "JoinBAPdata" tab in case the Google form is ever updated. The data is imported into excel using the import from web wizard, which I used a record to initially create. Though the interpreter didn't accept something it wrote later, so I deleted that piece. Also I changed the refresh style to "xlOverwriteCells" so that each new pull of data will write over the old data. Since the year starts with a blank sheet there is never more data in the sheet than is pulled in it.

Then the newName sub runs, checking all the names to see if any of them are new. This is done by taking the ID number of the student (fixing it to be the standard 8 or 9 digit number) and doing a find on the student info page. If the ID is not found the add variable becomes 1, meaning that the person is new and must be added to the student info page. If the ID is found the name is checked, if the name is "unknown" then the person is new but add is 2. This means that the ID number is in the workbook so the name and other pertinent data will be written over the "unknown" that is currently recorded in the cell. (The "unknown was written by the update current sheet, which was primarily created by the prior reporters)

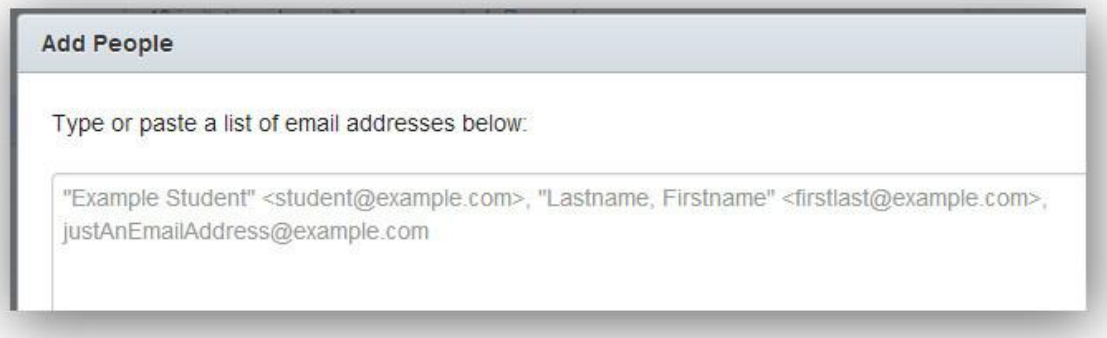


The "add" variable is set to a value of 0 before checking each ID number. If it becomes either 1 or 2 the addNewUser sub is called. This will change the global variable uploadUsers to True and then write the student info of the new student into the pertinent section of the "student

info" tab. The new student's info will then be written to a string variable "newUsers" in the format needed to add them to the Canvas course.

After all ID numbers in the import tab have been checked a message appears and declares whether there are new users (data). If there were no new users the sub ends there. If new users need to be added the sub will take longer than normal since it must access the Canvas site to upload the new users and then download the 'grades.' The reporter will also need to be awake at the keyboard in order to click save as will be explained later.

The addUser sub logs into Canvas (using hardcoded user ID and password of the reporter, which won't change year from year). Then it goes to the people page and clicks the add user button. The next window has a text box for entering the name and email address of each student in the format given by Canvas.



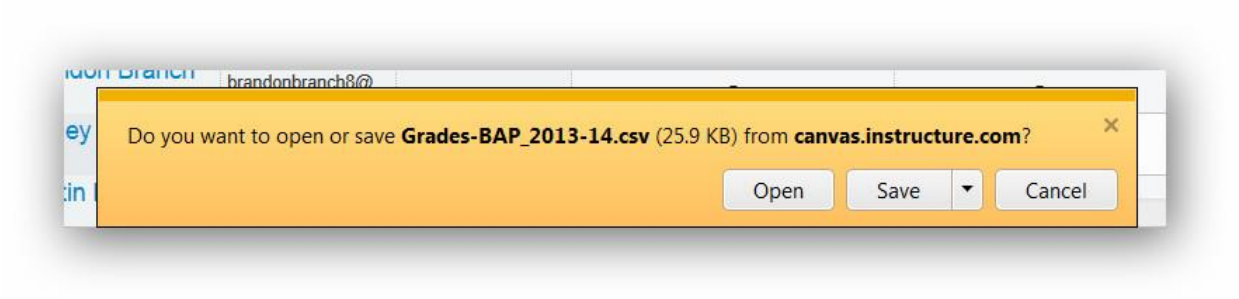
The newUsers string has the correct formation for this textbook, so it is copied directly into the textbox. This was thankfully easy since the textbox was given an id in the html [user_list_textarea]. The add button is also identified and easily clicked. The next window, however, has a next button that is not identified. A simple For loop with two nested If statements finds it in the code, but it is frequently clicked long before the code finishes the loop (through the document's length). Thus I put an On error block around the For Next loop.

Now that the names have all been updated the sheets of each event need to be updated. (The upload will not write in "unknown" names for upload to Canvas) The updateAll sub simply cycles through each of the event tabs and updates the active sheet.

Then the preparation for uploading is done. A CSV file will be needed for the next step, so it is created next. To easily manipulate it later it is given a name, so before it is created any CSV with that name is "Killed." (I was really excited to get to use this command.) The kill command is nested in an "on error resume next" block so that the first time this is run it won't crash.

The next sub will pull the gradebook down from Canvas. Each student is given a unique number by Canvas which must be used when uploading. Since the ie window brings up a

security window asking whether we want to save the reporter will need to click save. This window is invisible in the sense that it is not part of the html. There appears to be a workaround, but it involved as much code as I had written to this point. Thus I instead had the sub start counting from 1 to 400,000,000 which gives the reporter about 6 seconds to click save.



The importData sub then opens the file picker in the downloads folder for the computer. (This might not work on an apple, it uses the UserName with the normal path for downloads on Microsoft computers of C:\Users\Username\Downloads and adds "\Grades*.csv" so that only the csv files with the word grades at the start will show. Unless the reporter is also a TA on Canvas for some other class this should bring up only grades from Canvas to choose from. The grades data is then imported to the CSV file that was created and named earlier.

The uploadMinutes procedure was easier to write, but was really the reason I wanted to do this project. The greatest amount of my time as reporter went to meticulously putting in the minutes for each person, noting who wasn't yet in Canvas with highlights, and returning later for them. Since everything has been updated by the import data button none of the extra work needs to be done, the data can just be written over to the CSV and then uploaded.

First the CSV file is opened and the unnecessary data is deleted. Only the student names and identifiers are needed, so all columns to the right of that are selected and cleared of contents. Then the sub writes the data for each event as a separate column in the CSV file. There is a lot of bouncing between windows, but that will be intuitive when I explain what I'm doing so I won't enumerate the many times I had to put it into the code.

The code uses a do until loop inside of a for next loop to do the writing. The For next loop writes the name of the event (assignment to Canvas) at the top of the next column, and the total minutes (points to Canvas) under the name. Then the Do until loop goes down the sheet and checks each person's name. If the name is not "unknown" it is found in the CSV file and copies the minutes they have over to the appropriate cell in the CSV file.

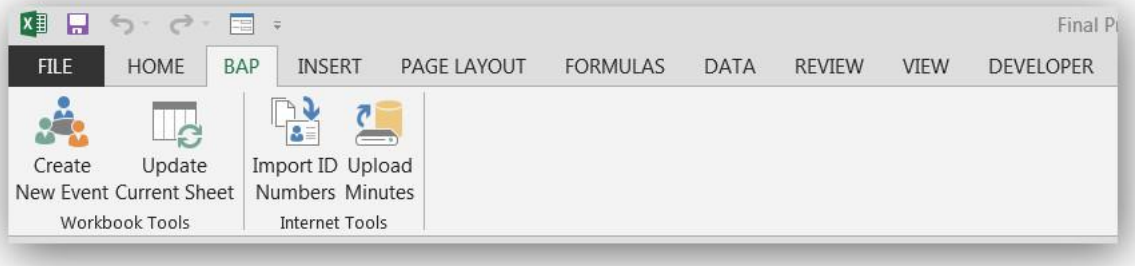
Student	ID	Section	Opening Meeting	Meet the Firms	
Points Possible			50	50	
Alberson, Daniel	1059611	BAP 2013-14	50		
Aldridge, Jason	246068	BAP 2013-14			
Allred, Jonathan	3430144	BAP 2013-14			
Anderson, Michael Gifford-Lee	3261789	BAP 2013-14		50	
Andrew Taylor	3430210	BAP 2013-14			
Barton, Scott	3719677	BAP 2013-14		50	

The loop continues until it reaches the bottom (a blank cell). Then it returns to the For next loop, goes to the next event sheet and repeats to the end of the workbook. Once all data has been uploaded the CSV file is closed.

Once all the data has been put into the CSV file the SendData sub opens Internet Explorer and logs into Canvas. It navigates to the grades page and selects the upload data option. At this point the macro will pause for the reporter to input things (I describe how in the next paragraph).

The next step is to check all the new events and select from a dropdown menu that it is a new assignment. This will vary in size since it will register a change for every assignment that was added and for every person who is now added to a previously created assignment. In order to give the reporter ample time to correctly select each option I created a "Wait" form. It simply asks if the reporter is "done yet." Once the reporter has finished the Canvas wizard for updating data the "yes" button is selected and the sub finishes by logging out of Canvas and closing Internet Explorer.

Aside from these main functionality macros I also changed the location of the BAP ribbon grouping (to be after home). I deleted all the unnecessary buttons, created the new buttons (also picked icons for the new buttons), and grouped the new buttons into two groups (Workbook Tools and Internet Tools). I also did a lot of cleanup, as I further explain in the Assistance section.



Learning & Conceptual difficulties:

While in class I copied down how to search for tags of buttons for the agent, but I didn't catch how to just click ids. I learned how to do this through experimentation on this project.

As I mentioned before I ran into several problems that would require learning a lot more than is in the scope of our course, or doubling the code I had to do a simple action. This included the invisible security window of Internet Explorer, learning how to manipulate the API that Canvas uses, and manipulating Google forms data. I feel that the workarounds I used simplified the code (increasing readability), and still made the process as easy as I hoped. The Google forms data turned out to be easily accessible using the publish hyperlink, though I didn't discover that workaround until I had spent a few hours trying to use the agent to navigate to the page of the results so that I could either copy it directly or download it as an excel sheet.

We did not go over how to create or manipulate other files. The recorder helped with creating the CSV file, but manipulating windows was a new concept. The process was similar to other objects we worked with in class, but sometimes it seems I wanted to use the workbooks class of objects and other times I just wanted to activate windows. I learned mostly by trial and error which was the one I wanted.

For uploading data I couldn't seem to get Canvas to accept the directory string that would lead to the CSV file. So the reporter will have to navigate to the CSV file and open it. Since I was planning on the reporter taking control on the next step anyways I don't see this as a big problem. As I played around with this upload process I saw lots of room for errors, so I think automating it would cause bigger problems than the little amount of work the reporter would have to do in selecting a few drop down options.

Assistance:

I used the old BAP tool as a starting point. The code was not well formatted, and there were no notes. Many variables were declared, but never used. I salvaged the update current sheet and create event items. I used the same buttons for those two, but updated the code to use names instead of the generic names that were originally used.

The sub for updating the current sheet used the "find" command to determine if an ID number was in the student info page. If it was not it would update the name to "unknown." If the ID were not there an error would occur, and an "on error go to" statement was used to skip code for writing names and go to the code for writing unknown. I used this same method to determine if imported student ID numbers were completely new (not in the workbook at all). This is much more efficient than running a loop to search through all the hundred ID numbers for every ID number that needs to be checked.

Much of the code within the subs used by the 'update current sheet' button pertained to prior functionality that was obsolete. I read through all the code and deleted anything that was not necessary. The only sub that was initially untouched was the one that transforms the card swipe into just the student ID number. However, I later discovered that some students entered

their ID number in the Google form with hyphens. I had to add to the sub in order to remove any hyphens.

I kept the “create new event” button/form the same. It was doing everything it was supposed to.