

## VBA Final Project Write-up

Rui Wan

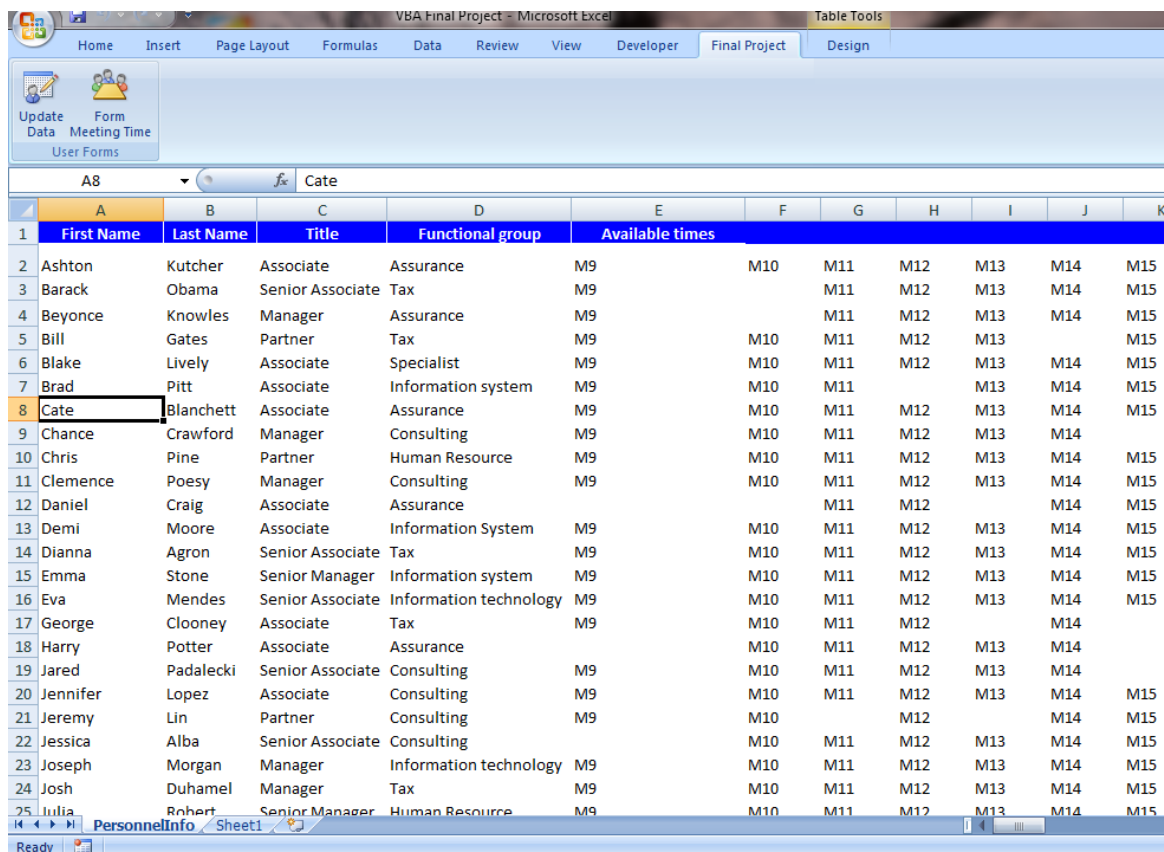
2012.12.07

### Executive Summary

The final project that I did is for professional firms to schedule meeting times. My project contains a spreadsheet with two user forms. One form is for users to edit or update personnel information. And the other form is for users to input attendees of a meeting, and then form the best time to meet. When there are more than one available times to meet, the user can use the “Next” “Prior” button to navigate and determine the best time themselves.

### Implementation documentation

I based my project largely on the user form project. My spreadsheet looks like this:



	A	B	C	D	E	F	G	H	I	J	K
1	First Name	Last Name	Title	Functional group	Available times						
2	Ashton	Kutcher	Associate	Assurance	M9	M10	M11	M12	M13	M14	M15
3	Barack	Obama	Senior Associate	Tax	M9		M11	M12	M13	M14	M15
4	Beyonce	Knowles	Manager	Assurance	M9		M11	M12	M13	M14	M15
5	Bill	Gates	Partner	Tax	M9	M10	M11	M12	M13		M15
6	Blake	Lively	Associate	Specialist	M9	M10	M11	M12	M13	M14	M15
7	Brad	Pitt	Associate	Information system	M9	M10	M11		M13	M14	M15
8	Cate	Blanchett	Associate	Assurance	M9	M10	M11	M12	M13	M14	M15
9	Chance	Crawford	Manager	Consulting	M9	M10	M11	M12	M13	M14	
10	Chris	Pine	Partner	Human Resource	M9	M10	M11	M12	M13	M14	M15
11	Clemence	Poesy	Manager	Consulting	M9	M10	M11	M12	M13	M14	M15
12	Daniel	Craig	Associate	Assurance			M11	M12		M14	M15
13	Demi	Moore	Associate	Information System	M9	M10	M11	M12	M13	M14	M15
14	Dianna	Agron	Senior Associate	Tax	M9	M10	M11	M12	M13	M14	M15
15	Emma	Stone	Senior Manager	Information system	M9	M10	M11	M12	M13	M14	M15
16	Eva	Mendes	Senior Associate	Information technology	M9	M10	M11	M12	M13	M14	M15
17	George	Clooney	Associate	Tax	M9	M10	M11	M12		M14	
18	Harry	Potter	Associate	Assurance		M10	M11	M12	M13	M14	
19	Jared	Padalecki	Senior Associate	Consulting	M9	M10	M11	M12	M13	M14	
20	Jennifer	Lopez	Associate	Consulting	M9	M10	M11	M12	M13	M14	M15
21	Jeremy	Lin	Partner	Consulting	M9	M10		M12		M14	M15
22	Jessica	Alba	Senior Associate	Consulting		M10	M11	M12	M13	M14	M15
23	Joseph	Morgan	Manager	Information technology	M9	M10	M11	M12	M13	M14	M15
24	Josh	Duhamel	Manager	Tax	M9	M10	M11	M12	M13	M14	M15
25	Julia	Robert	Senior Manager	Human Resource	M9	M10	M11	M12	M13	M14	M15

It contains information of employees' names, titles, functional groups, and available times to meet. The symbols under available times are very intuitive to understand. For example, “M9” means Monday at 9 am is available. I have listed all possible workday

hours, and a blank cell means at that time that particular employee is not available. Also on the tab collections, there is a tab named “Final Project”, and by clicking it, there are two buttons for the two user forms.

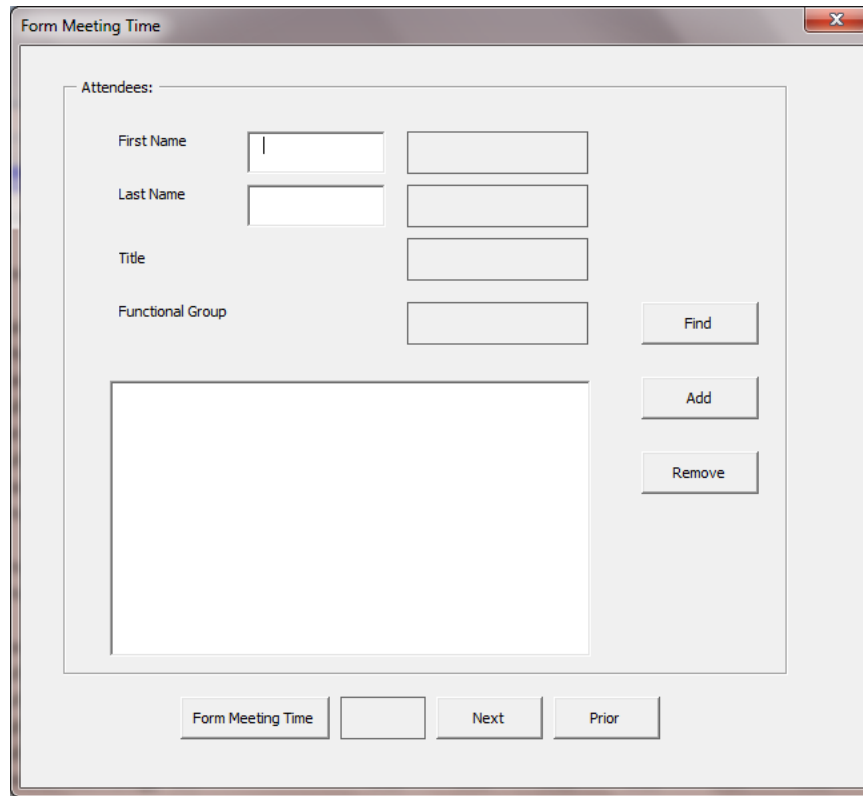
The update data form looks like below:

The screenshot shows a Windows-style dialog box titled "Update Data". It contains the following elements:

- First Name:** A text box containing "Cate".
- Last Name:** A text box containing "Blanchett".
- Title:** A group box containing five radio buttons: "Associate" (selected), "Senior Associate", "Manager", "Senior Manager", and "Partner".
- Functional Group:** A group box containing seven radio buttons: "Assurance" (selected), "Tax", "Information System", "Information Technology", "Human Resource", "Specialist", and "Consulting".
- Available Times:** A text box that is currently empty.
- Buttons:** "Add" and "Cancel" buttons are positioned to the right of the "Available Times" text box.
- Time List:** A list box containing the items "M9", "M10", "M11", "M12", "M13", and "M14".
- Buttons:** "Add Time" and "Remove" buttons are positioned to the right of the list box.
- Save Button:** A "Save" button is located at the bottom right of the form.

Users can update or edit personnel information using this form. If a cell or a range has been selected when the form is initiated, then this form automatically pulls information from the active cell into the output of this form. The user can edit the form by direct typing or selecting from list, or adding to a list box. Once the user is done, he can click the “Save” button and the update from this form should be reflected on the corresponding spreadsheet.

The second form is for users to form meeting time, and it looks like this:

The image shows a software window titled "Form Meeting Time" with a standard Windows-style title bar (minimize, maximize, close buttons). Inside the window, there is a section labeled "Attendees:" which contains four input fields: "First Name", "Last Name", "Title", and "Functional Group". Each of these fields has a corresponding empty text box to its right. To the right of these input fields are three buttons: "Find", "Add", and "Remove", arranged vertically. Below the input fields is a large, empty rectangular box, likely intended for a list of attendees. At the bottom of the window, there is a horizontal row of four buttons: "Form Meeting Time", an empty button, "Next", and "Prior".

The user can input the attendees of a particular meeting by either inputting the first names or the last names, and then hit “Find”. Once the program finds the attendee in the database, his or her title and functional group will automatically appear on the form for the user to verify its validity. After making sure, it is the correct attendee, the user can click “Add”, to add this attendee into the list box. Also the user can remove attendees from the list by clicking the “Remove” button. Once the list box has all the meeting attendees, the user can click “Form Meeting Time” to get the available time to meet. If there are more than one possible solutions, the user can click “Next” or “Prior” to navigate through those options. If there is no possible solution, the user will get a message box alerting him, so that he can rearrange the attendee list and rerun the program again.

The main logic behind “form meeting time” is first to list all available times among the all the attendees into “sheet1”. Second, we add one point to the times that are shared by more than one person. The more person share this particular time, the larger value the available time gets. For example, if there are five people attending this meeting, then the highest possible value for a particular available time is five, indicating all five people

have this time available. Then I rank those available times according to the assigned values. If the highest value equal to the number of attendees, then we have a possible meeting time. If there are multiple of such times, the users can choose the time that best accommodates their circumstance. If the highest value is less than the number of attendees, then no meeting time is possible. The user need to go back and reforming the list.

## **Discussion of learning and conceptual difficulties encountered**

When I formed the proposal, I thought it is an easy project and shouldn't take me much time. It was not until I actually started the project that I realized how much detail it required me to take care of. These past few weeks, all I could think of was this project. I ate lunch thinking about how to solve one of the bugs, I took naps thinking about how to solve a problem with a do loop...

The whole process was full of my happiness and anger and freak outs. Every time my roommate saw me laughed for a while and then started to freak out, she knew I was working on the VBA final project again. But I must say, I learned the most by struggling through the most difficult task. The knowledge is not mine if I just listen to the Professor teaching it in class; I have to actually do it by myself to fully learn it. And I feel like I learned the most through the debugging process, and trying to figure out the problem by myself. Even though it means hours of hours seemingly meaningless searching for error, I feel super relieved once I successfully remove the bug on my own.

After doing this process, I am more competent at building user forms, and I am also getting better at building all kinds of if-statement to solve my logic needs. However, the most valuable thing I learned is the need for detail orientation. The user forms can be polished better if I spend more time on it. And we must put ourselves into the shoe of actual users to make improvement of the user forms. I revised my user forms at least three times during the whole process, every time I added an extra more features to make it more user-friendly. And I believe they still have plenty of room for improvement.

During the process, I also found out one of my big problem at programming is that I tend to overcomplicate the things I am trying to solve. Sometimes, I spend a whole night working on a multi-dimensional array to store my data, but couldn't get it figured out, then I asked the professor for help the next day, and his approach is actually much easier compared to mine. So the code you see for now, is just a portion of the vast amount of code that I wrote, since most of them don't work as they supposed to, I discarded them before you can see them.

However, for some reason, the main logic to form the meeting times doesn't work that well now. It worked just fine back at the Professor's office, but it didn't now. I couldn't get it figured out. Instead of returning the available times for all the attendees, it actually returns almost all times. The code is under the button "Form Meeting Time".

## **Assistance**

During the whole process, the professor helped me a lot. The main logic in forming a meeting is by the help of him. I also get some help from my Isys major friends, but they aren't familiar with the VBA language, so they mostly just help me form up the logic, and I need to write the code myself.

## **Write-up detail**

The basic idea I have for my project is to build a program that can help facilitate the process of forming a meeting time that accommodate everyone's schedule. The first step I took is to design the outlook of my user form. I knew I need two forms, one to update data, and one to use the data to form meeting times.

After I have sketched out a rough draft for my form, I started to change names and captions of those labels, buttons, list boxes and so forth on the form. Then I use the user form project as a reference to build up my first user form's code.

The second user form took me substantially more time to complete. I tried to come up with my own logic but failed. I have tried to use find duplicates to find the same available times, but the duplicates exists among two things, not necessarily among more than two, so it didn't work. I also tried to use one array to look across columns to look at each available time, and one array to look across the rows to look at different attendees. Again, that is just too complicated for me to handle. After timeless freak outs, I finally decided to seek help from the Professor. And he kindly walked me through his logic on how to approach this problem. I think the logic is beautiful. Here is the main idea of how the logic flows:

First, I make sure that every time an attendee is entered into the list box, the attendee row is selected. Once all the attendees have been entered, I have a selection of rows from all the attendees, from which I can build a "For each cell in selection" loop to process my logic. I also need to take care of the "Remove" button to make sure every time I click it; the removed attendee is no longer selected. It is actually very tricky to get it work. Then we can use the logic of assigning each available time a value based on its frequency of appearance among all the attendees to solve the scheduling problem.

Like I mentioned before, the logic sounds perfect for me, and I have watched it run perfectly in the Professor's office, but when I get home and rerun it, it doesn't work the way it did. ☹

Anyhow, let's assume it works just fine like it did in the Professor's office, once there are multiple solutions, I build the "Next" and "Prior" button for users to navigate back and forth. The code I wrote for the "Next" and "Prior" button is also based on one of the Professor's lesson. I modified the code to best accommodate my program and it worked fine.

In summary, I think it is a very neat program if we can get the dysfunctional part working. The users from the Professional firms, such as Public Accounting firms, can actually use it to schedule meetings quickly and efficiently. I am grateful for all the help I receive for this project!