



Event Check-in Solution

*Made for the BYU Women in Business Club to
assist in member check-ins at events*

I Sys 520 Final Project, Courtenay Maynes

12/8/2012

Table of Contents

Executive Summary	2
Background	2
Overview	2
Conclusion	2
Implementation	3
How to Use It	3
How It Works	8
Keep Track of Members and Club Events	8
Adding New Events	8
Allowing Check-ins via Magstripe Reader or Direct Keyboard Input	9
Learning & Conceptual Difficulties	10
Assistance	11

Executive Summary

Background

The BYU Women in Business Club has been around for several years, but in the past school year, its membership has grown exponentially. While the leadership is delighted to see this kind of growth, the president of BYU's Women in Business club, Katherine Poulter, expressed some concerns she had concerning the current method of attendee check-ins at events. For each event, the leadership had to print off a list of members and cross off names as each person came up to the doorway. For the first event of the semester, the inefficiency of this checking-in process caused the event to start 20 minutes later than it was supposed to. Katherine approached me and asked if I would be able to create some sort of program that could use the BYU-issued student ID cards to log members into events. I created a program using Excel UserForms and Macros along with the hardware of a keyboard wedge magstripe reader to improve efficiency for the club and simplify the log-in process.

Overview

I built a system to meet the current need of the Women in Business club of a new way to manage check-ins. The user just needs to click on the check-in button, select which event he or she is checking people in to, and swipe each student's card as the student walks in to the event. The first of the two sheets in the workbook is the "Members" sheet, which holds each student's first name, last name, and BYU ID number, as well as a record of which events each student has attended. The second sheet is the "Events" sheet, which keeps track of each event with an ID, title, date, start time, end time, and number of attendees. I've added a tab to the ribbon titled "WIB Functions" (WIB stands for Women in Business) and it has four buttons. The buttons include a "Check in" button, "Create New Event" button, "View Members" button, and "View Event Reports" button. The interface is fairly simple to use and is pretty user friendly.

Conclusion

This solution should improve efficiency for the club. Checking one person in takes approximately 3 seconds (due to the length of time it takes to swipe a card—the system itself takes less than a second to process each check-in). Since the club usually has about one hundred fifty people at each event, checking all the attendees in should take less than ten minutes using only one station. I recommend that the club purchases two keyboard wedge magstripe readers and sets up two stations performing check-ins, cutting check-in time down to less than five minutes. Additionally, the system I've created provides helpful member attendance information for the leadership to determine how successful each of their events are and where there is room for improvement. Finally, this program is very flexible and I could add RSVP information or other metrics if that's what the club desires. So far, I have been only asked to manage check-ins for events, so I've tried to limit my scope to that.

Implementation

First, I will describe how to use the system and then explain how it works. The “How to Use It” section is meant to be a sort of user guide, while the “How it Works” section describes my thought process and the logic behind the system.

The system provides the following functionalities:

- Keeps track of members of the club and their ID numbers
- Keeps track of club events (including each event’s date, start time, and end time)
- Allows the user to add a new event
- Allows the user to check in attendees to specific events
- Accepts magstripe BYU ID cards swiped through a card reader as input
- Allows the user to manually type in a BYU ID number to check a club member in
- Notifies the user of specific errors in check-in inputs
- Runs until every attendee is logged in and the user quits
- Allows for check-ins even after the user has quit by opening the program back up
- Keeps track of summary data, mainly the number of attendees per event

How to Use It

When a user wants to start checking members into the event, they must click on the “WIB Functions” tab and then the “Check in” button, as seen in Figure 1.

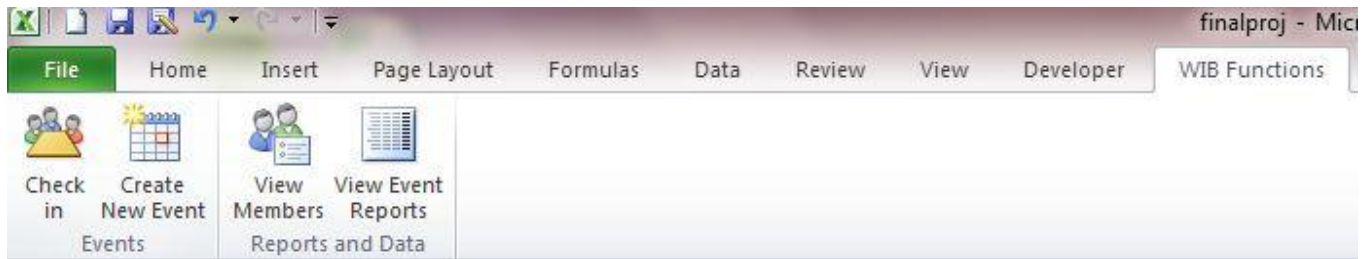


Figure 1 - WIB Functions Menu

When the user clicks on the “Check in” button, the following figure (Figure 2) appears. Note: some of my figures appear blurry due to the snipping tool I used, but in Excel they are not blurry at all.

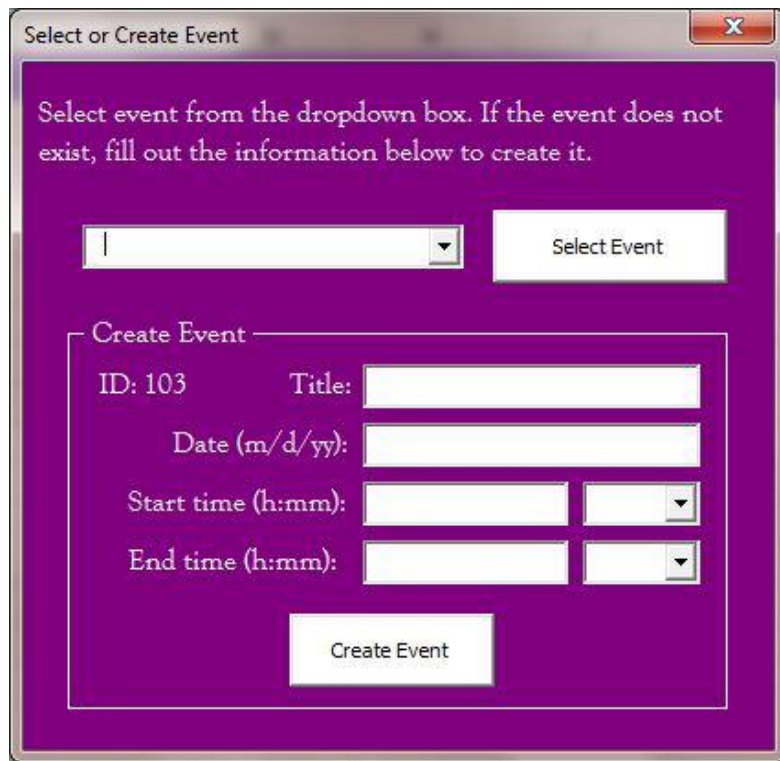


Figure 2 - Check-in Start Screen

Before any members can be checked in to an event, we must first know what event they are checking into. The user has two options here. They can either select an event from the events already contained in the workbook, or, if the event they want to check people in to has not been created, they can create that event. Each event has an ID (provided by the system), a title, date, start time, and end time.

If the user wants to choose an existing event, they can just click on the dropdown, click on the event they want, and then click the “Select Event” button (See Figure 3). If the user wants to create a new event, they can fill in the information in the boxes provided and click on “Create Event”. Dropdowns are provided for the AM/PM part of the date (See Figure 4).

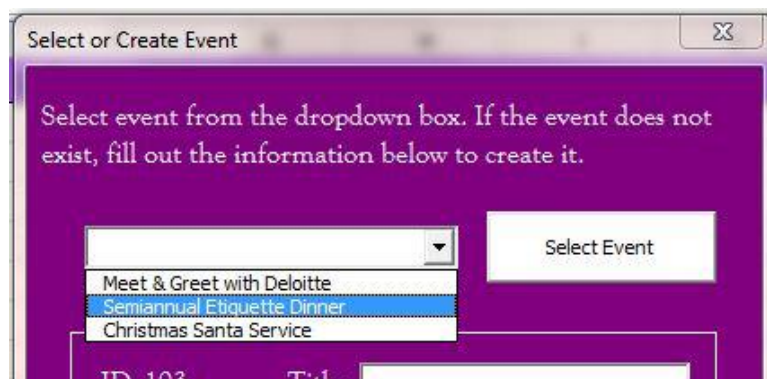


Figure 3 - Selecting an Event

Create Event

ID: 103 Title: Gove's Birthday Party

Date (m/d/yy): 12/13/12

Start time (h:mm): 7:00 PM

End time (h:mm): 9:00 AM

Create Event

Figure 4 - Creating an Event

Once the user clicks “Select Event” or “Create Event,” a new form appears and waits for the user to either swipe a card or enter the ID number (see Figure 5).

Women in Business Event Check-In

Slide card

If the member doesn't have their card, just type the member's BYU ID number followed by the enter key.

Figure 5 - Checking in Attendees Screen

The program will wait on this screen for input from the users. Input from the keyboard will appear in the light green box. If there's a syntactical input error, such as a card being swiped too quickly for the reader to get the data or an ID is entered that is too short, an error message will appear as seen in Figure 6 and 7. This error message will remain for 2 seconds and then the original slide card screen (Figure 5) will return. The message is based on the type of error received. The error screen is the same screen as the input screen, just with a red background and the text of the message changed.

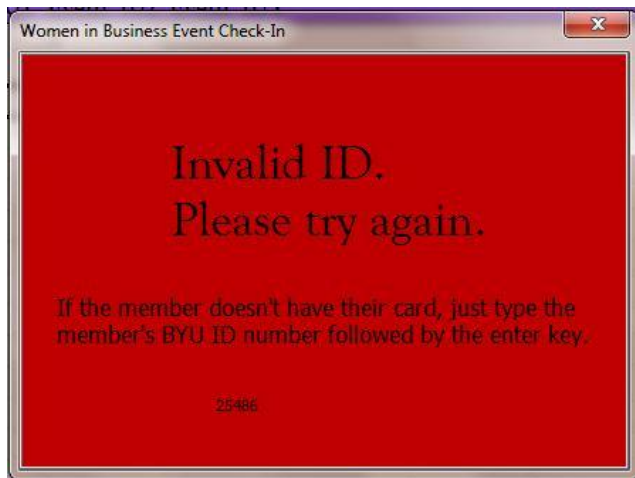


Figure 6 - Error Message for a Manually Entered ID

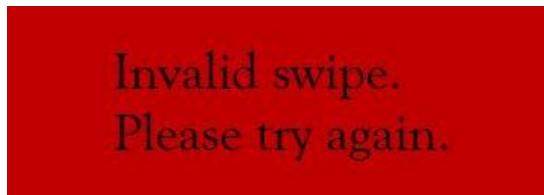


Figure 7 - Error Message from an Invalid Card Swipe

As each card is swiped, the system takes the ID, finds it in the list of members, and puts an “X” for that specific event to check in that user (See Figure 8). When the user is finished checking people in, they just click the close button in the top right corner of the program. If someone comes up and wants to check in after the user has already closed out, all the user has to do is click the “Check in” button once more, select the event from the dropdown box, and slide that person’s card.

	A	B	C	D	E	F	G
1	First Name	Last Name	Student ID	Event 100	Event 101	Event 102	Event 103
2	Gove	Allen	11111111	X	X	X	
3	Christi	Allen	22222222	X	X		
4	Dolores	Atkins	33333333	X	X		
5	Jan	Baker	123456789	X			
6	Dolores	Barr	987654321	X			
7	Sherri	Barton	651756505				
8	Angela	Barton	732465638	X	X		
9	Patricia	Barton	634110351				
10	Meredith	Beasley	232182390	X	X		
11	Priscilla	Becker	858106633				

Figure 8 - Member and Attendance Information

Besides checking students into events, users can add an event completely independent from whether or not they are checking students in right then or not. The user can click the “Create New Event” button in the “Events” group, and the form shown in Figure 9 will appear.

Select or Create Event

Create an event below.

Select Event

Create Event

ID: 104 Title: Gove's Christmas Party

Date (m/d/yy): 12/22/12

Start time (h:mm): 9:00 AM

End time (h:mm): 10:00 PM

Create Event

Figure 9 - Create an Event

This form is very similar to the form that appears when “Check in” is clicked. However, the dropdown box and select event button are disabled, and the instructions state “Create an event below.” All the user needs to do is fill in the information and click the “Create Event” button.

The final two buttons are “View Members” and “View Event Reports.” These buttons have very simple functions. When a user clicks “View Members,” the “Members” sheet is selected. When a user clicks “View Event Reports,” the “Events” sheet is selected. Figure 10 is an example of what the events sheet will look like.

	A	B	C	D	E	F
1	ID	Title	Date	Start Time	End Time	Number Attended
2	100	Meet & Greet with Deloitte	9/12/2012	4:00 PM	6:00 PM	105
3	101	Semiannual Etiquette Dinner	11/17/2012	6:30 PM	9:00 PM	110
4	102	Christmas Santa Service	12/12/2012	12:00 PM	2:00 PM	2
5	103	Gove's Birthday Party	12/13/2012	7:00 PM	9:00 PM	1
6	104	Gove's Christmas Party	12/22/2012	9:00 AM	10:00 PM	0

Figure 10 - View Event Reports

One thing to note about the “Events” sheet is the “Number Attended” column. This number counts the number of people that attended each activity based on the X’s on the membership sheet. This information can be very useful to members of the club leadership that work on planning events because it will help them gauge the success of various events and maybe give them an idea of what works well and what doesn’t.

How It Works

I divided this section into the following parts:

- Keep Track of Members and Club Events
- Adding New Events
- Handling Check-ins via Magstripe Reader or Direct Keyboard Input

In each part, I'll go into more detail about how each function of my system works technically. I may include snippets of my VBA code to aid understanding.

Keep Track of Members and Club Events

To keep track of the members of the club and club events, the system just uses simple worksheets. The only formula in any of the cells is in the "F" column of the "Events" worksheet (See Figure 10), and this formula is a COUNT formula to determine how many people attended the corresponding event. This formula is automatically generated when an event is created by finding the first empty column on the "Members" worksheet. The formula is as follows: =COUNTA(Members!D:D)-1. But when the event is actually created in the system, the following line of code is what puts the formula in that cell:

```
Sheets("Events").Range("A1").Offset(count, 5).Value = "=COUNTA(Members!" & myColumnLetter  
& ":" & myColumnLetter & ")-1"
```

"Count" is the number of rows that are already filled + 1. myColumnLetter is the column letter on the "Members" sheet that corresponds to this event.

Adding New Events

Adding a new event can happen in one of two ways. The first way is when the user clicks the "Create New Event" button. This brings up the figure shown in Figure 9. In order to determine the ID number, my code goes to the "Events" sheet, finds the last populated event row, takes that ID number, and adds one to it to create the new ID. For example, if I were to click "Create New Event," and my "Events" sheet looked like Figure 10, the program would find the last row, grab that ID (the one with a value of 104), add one to that value (so the new value is 105), and set the ID to 105. Other special code includes the combo boxes—I programmed them to hold only two values, "AM" or "PM," which makes sure times are entered in the correct format. When the user clicks "Create Event," the system goes to the "Events" sheet, finds the first empty row, and populates that row with the event data. It concatenates the value from the time text box with the value from the combo box, putting a space in between. Finally, the code goes to the "Members" page, iterates through the first row until it finds a blank cell, and enters the words "Event" plus the ID of the event. This is in preparation for checking members into events. When a member is checked into an event, an X will show up in this column on the same row as his or her name.

The second way to add an event is when the user clicks "Check in." Figure 2 appears. The combo box is pre-populated with the events from the "Events" sheet, but the bottom part allows the user to add a new event rather than selecting an existing event. The formatting is all the same as the "Create New Event" screen, and so is the code. In fact, I used the same UserForm for both of these functions, but I changed the caption of the label and enabled the selecting functions if the user is creating an event through the "Check in" button.

Allowing Check-ins via Magstripe Reader or Direct Keyboard Input

Check-in occurs when the user clicks the “Check in” button. They must either select an existing event or create a new event as seen on Figure 2. The combo box of existing events pre-populates based on the “Events” sheet. My code iterates through each row and snags the title of each event, adding the title to the combo box, until a blank cell is reached. Then it quits out of that do loop. When the user clicks “Create Event” or “Select Event”, the column of that event on the “Members” sheet is passed from the current userform (UserForm2) to the next userform (UserForm1) with the following code:

```
Load UserForm1  
Call UserForm1.fillVars(eventColumn)  
Unload Me  
UserForm1.Show
```

The “Call” line gives UserForm1 the column of the selected/created event on the “Members” sheet. Then, the next userform is shown (Figure 5). Nothing happens with this screen in the code until a key is pressed (or until a card is swiped—the card swipe is treated as keyboard input). When a key is pressed, the system stores the key in a variable called “keys” which strings together each key pressed until certain things happen. (keys = keys & Chr(KeyAscii))

The magstripe input from a BYU ID looks like this: ;256518817=02133?

When a card is swiped and this input comes in correctly, the code will log the member in to the event. If the user doesn’t want to swipe the member’s card, the user can type in the member’s BYU ID and press the enter button. If an error occurs when swiping a BYU ID, that input looks like this: ;E?

There are three different things that happen based on the variable “keys.” First, if “keys” is longer than 17 characters or if it is “;E?” then the error message shown in Figure 7 will appear for two seconds. The background is changed to red to show the user that an error has occurred, since red is usually associated with errors and used to show that something is not as it should be. I used the following code to set the error to a timer:

```
timeSet = Now + TimeValue("00:00:02")  
Application.OnTime timeSet, "BackToNormal"
```

The “BackToNormal” subprocedure sets the userform back to its original green and restores the labels to their original captions.

The second thing that can happen based on the variable “keys” happens if the “enter” key (KeyAscii = 13) is pressed and the length of “keys” is less than ten. If a user pushes “enter” before they’ve completed typing in a BYU ID (which should be 10 keys), then the error message in Figure 6 appears. Again, the screen turns red for two seconds before returning to normal.

The third thing that can happen based on the “keys” variable is if either the enter key is pushed and the length of “keys” is equal to 10 or the “=” key is input and the length of keys is equal to 10. As you can see in the magstripe input, the student’s BYU ID comes between the “;” and “=”. The rest of the numbers are irrelevant in this case. So, if a valid ID was entered either manually or via card swiping, I grab the input

(the input stops at the "=", so I don't need to worry about the rest of the numbers), replace ";" with "", replace "=" with "", and search the "Members" sheet, column "C", for the student ID.

If something is found, then we know the member exists and has checked into the event. So, we put an "X" into the corresponding cell to check that member into the event. I used the following line of code:

```
Sheets("Members").Range("A1").Offset(rng.row - 1, col - 1).Value = "X"
```

The variable "rng" is the found cell, and the "col" is the column number of the event that I pulled from the previous userform. I subtract 1 from each, because in offsets, 0 is the first element.

Once I put an "X" to mark that member present at the event, I set a message to appear for two seconds, showing the user that the member has been successfully checked in (see Figure 11). Then, I run the "BackToNormal" subprocedure that I referenced earlier to restore the original "Slide Card" message (Figure 5).

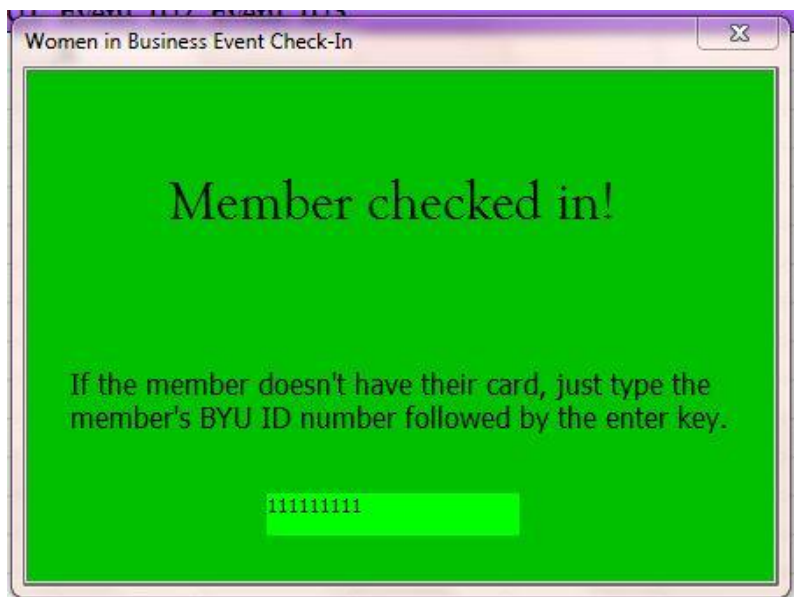


Figure 11 - Member Successfully Checked in

If nothing is found in the column of ID's, then a message box appears saying "Nothing found." This means that the ID entered or swiped in is valid, but the person trying to check in is not a member of the club. The user can then go in and add the member to the bottom of the "Member" sheet along with their ID and re-check them in.

Learning & Conceptual Difficulties

The first stumbling block I came across in this project was trying to figure out how to allow data to input straight from the keyboard into the program. I tried to use the keypress function, but I didn't want the card swipe input to go into a textbox. I wanted a screen where the program would wait until a card was scanned, and for the userform to directly accept the keyboard data. At the same time, I wanted a textbox for a user to be able to manually input an ID number and a button to submit it. I tried to say UserForm1.SetFocus so that the focus didn't go to the textbox and button on my userform, but that

didn't work, and I couldn't find a solution online. The sites I read stated that focus cannot shift back to the userform if you have any buttons or textboxes. I solved this problem by deleting the button and textbox. The userform then could do the keypress function without a problem. To solve the problem of not having a textbox or button, I created a label that acted as a textbox. The label caption changed every time a key was pressed to the current value of "keys." Instead of a button, I wrote instructions to the user telling them to press the enter key.

I had a bit of difficulty passing a variable between userforms, especially because I was hesitant to make it public, but I was able to pass variables back and forth using lines of code such as "Call UserForm1.FillVars(column)," which ended up working very well. That took me a while to figure out though.

I also wanted this program to be extremely user friendly and very easy to use. Therefore, I spent a lot of time learning about how to use fonts and colors on userforms. I changed labels or made certain ones appear or reappear based on whether or not there was an error. I learned how to use a timer to display messages for short periods of time to give users feedback on errors and successful check-ins.

Assistance

The only assistance I received on this project was assistance through my research on help sites. I wrote my solution and didn't copy it from an outside source.