# Luke’s Automated SWA Boarding Program

## Executive Summary

Southwest airlines does not offer assigned seats but they do give seating priority based on when the customer prints up their boarding pass, which customers are allowed to do 24 hours before their flight. Luke’s Automated SWA Boarding Program eliminates the headaches of trying to remember to check in by automatically doing it for you and then sending the customer a confirmation email with their new seating priority number. All that is required by the user is to input their name, confirmation number, and their email address, and then set a simple task in the Scheduler program. After that they can sit back, relax, and enjoy the friendly skies while the system does the rest.

## Process

The following are the outlined steps for how this program works:

#### 1 Download files and select storage location

The following files are needed:

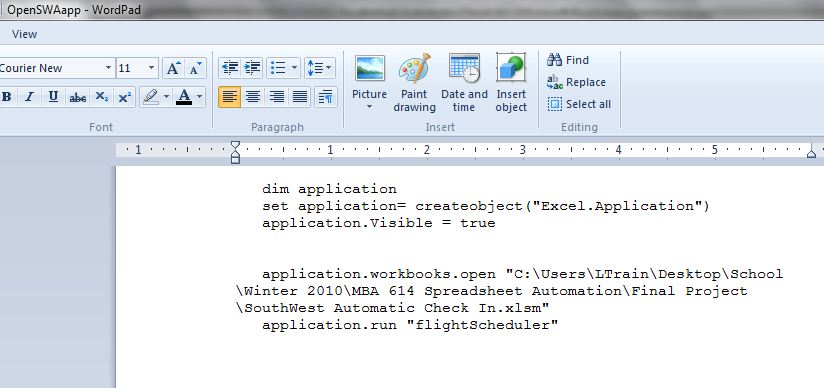
* Excel file: ‘SouthWest Automatic Check In’
* VBScript: ‘OpenSWAapp’

#### 2 Modify the VBScript

This only needs to be done once but is necessary in order for the script to run properly. Part of the code as shown below holds the file path to the excel file that was downloaded. In this example the file path is:

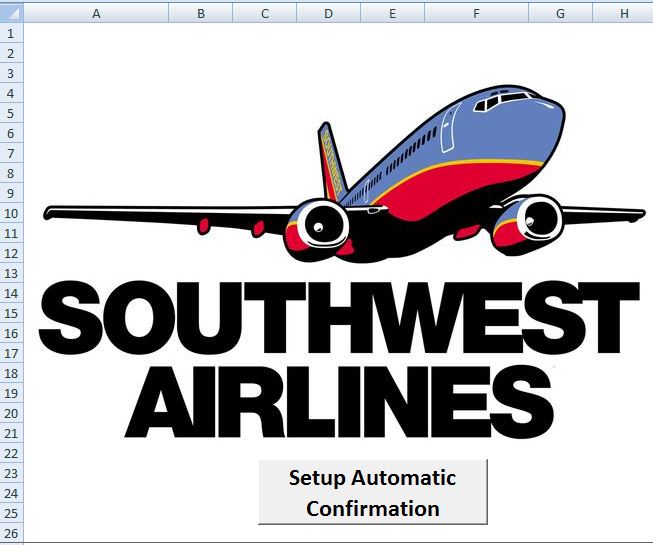
“C:\Users\LTrain\Desktop\School\Winter 2010\MBA 614 Spreadsheet Automation\Final Project\SouthWest Automatic Check In.xlsm”

The user will need to change the file path to match where the excel file is saved on their computer. They can do this by opening the VBScript file in a simple Wordpad.

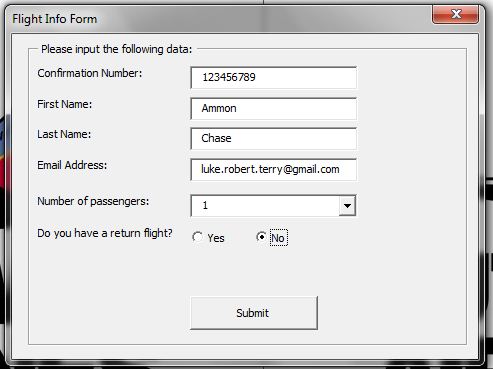


#### 3 Open the Excel File and run the application

When the user opens the Excel File the first sheet is called ‘SW Confirmation’ and is displayed below. After purchasing a Southwest ticket online, simply push the ‘Setup Automatic Confirmation’ button to get started.

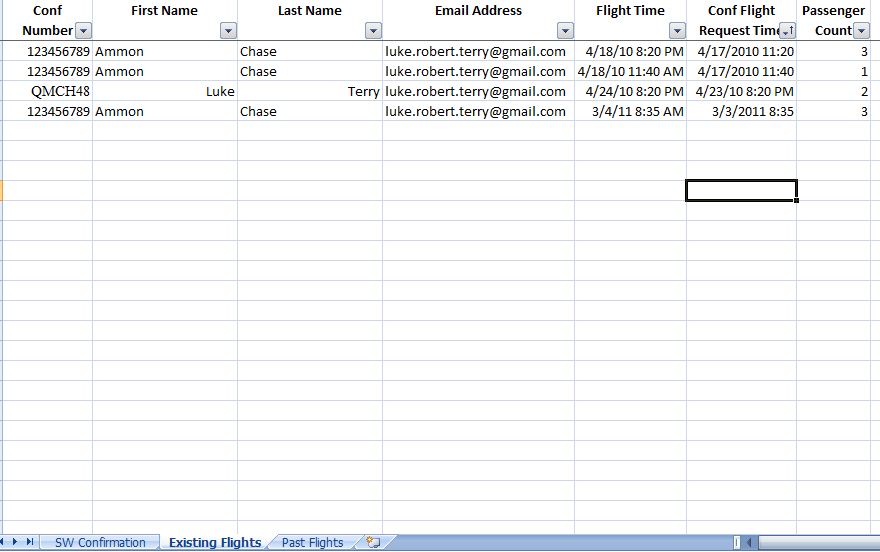
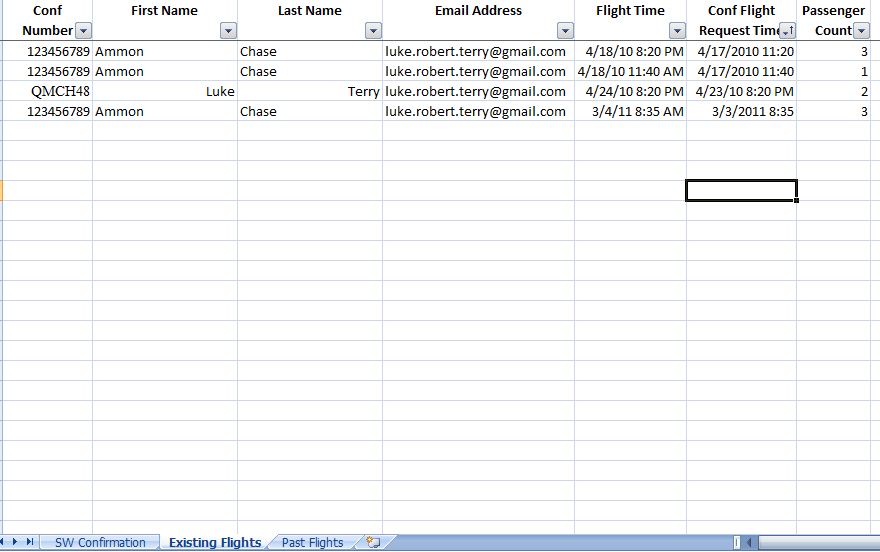


The button will bring up the following user form. Simply put in the flight Confirmation Number, name and email address, and specify the number of passengers using this confirmation number and if there is a return flight. Southwest airlines will allow you to book up to 8 passengers using one confirmation number.

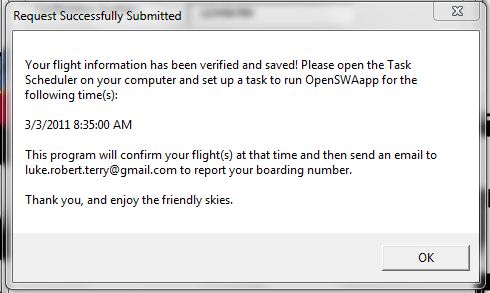


#### 4 Behind the scenes operation to validate and save flight info

As soon as the user hits the submit button, code will run to immediately try and check the passenger in. The user will not see this. The purpose of this is to make sure the user has put in a name and confirmation number that matches with what’s in Southwest’s system. If there is inaccurate information then a pop up message will appear to the user and they can then re-enter their flight information. After the correct information is validated, the program will pull the scheduled flight time from the website and save that with the user’s information in the 2nd sheet of the excel file entitled, ‘Existing Flights.’ If there are multiple flights already saved in the system, then excel will sort them on this sheet with the next upcoming flight appearing at the top of the list, as shown below:

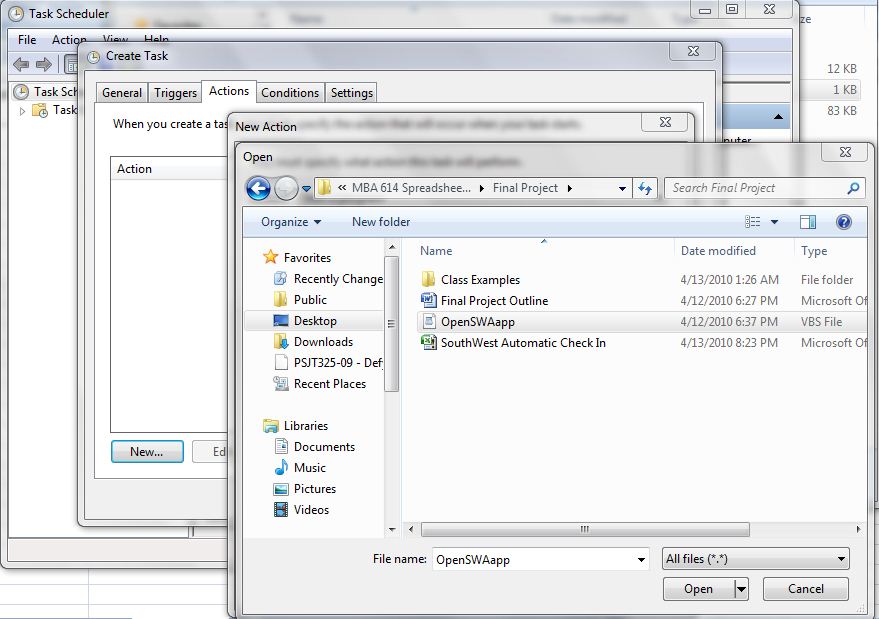


After the flight information has been saved a reminder message will pop up to prompt the user to open the task scheduler on their computer and set a task to run the VBScript at the flight confirmation time, 24 hours before the flight. This message will look like this:

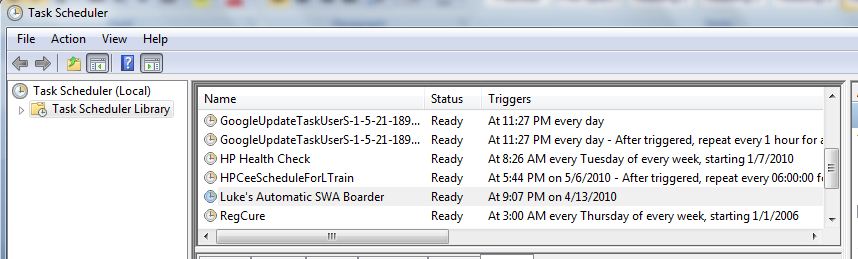


#### 5 User must create a task with the task scheduler

When creating this task there are two simple requirements. First, within the ‘Triggers’ tab the user needs to set the time that the program should run. This will be the time that is posted on the above message. If the user has a return flight, they’ll need to mark that time as well or create a new task with the 2nd trigger time. They should be sure when setting the trigger times that they mark the task as a one-time event (this will keep the task from running every day or week, etc.). The second requirement is to set the ‘Actions’ tab. Here the user must call the VBScript. Simply click the browse button and go to where the ‘OpenSWAapp’ file is stored (as demonstrated below). The user can then save the task and exit. They are officially done with their involvement in the process.



Here is what a typical task will look like in the Task Scheduler

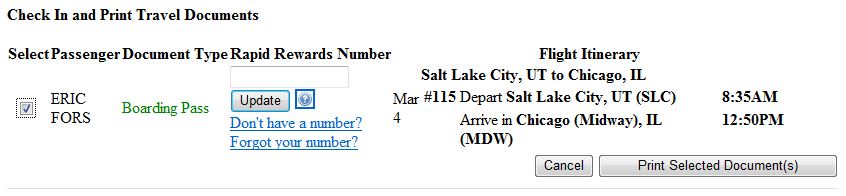


#### 6 System confirms flight

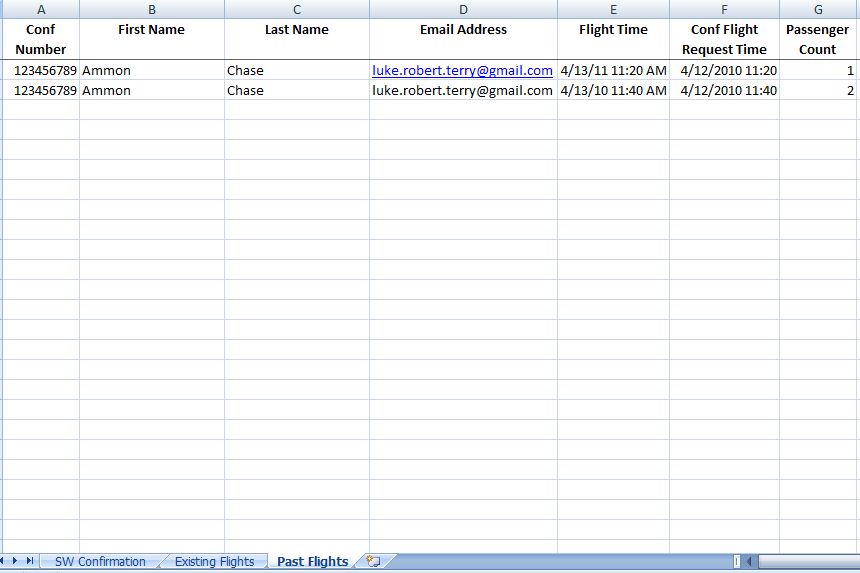
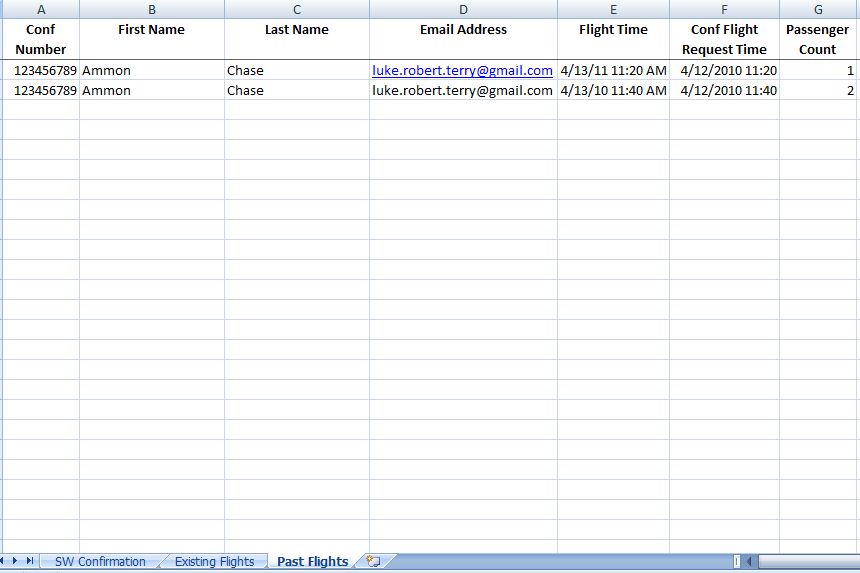
When the task scheduler finally runs the VBScript 24 hours previous to their departure, the VBScript will call the code from the excel file to log into Southwest’s website. Although the user won’t see it, the system will in effect fill out this form on southwest’s homepage:

## Capture3.JPG

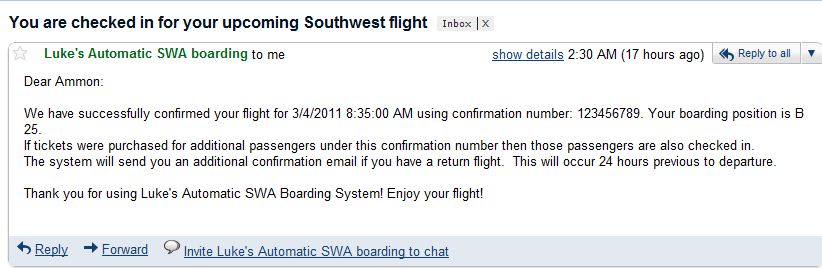
After which it will check in all passengers who are under the given confirmation number, as demonstrated in the screen below.



Additionally, the program will then move the recently checked in flights from the ‘Existing Flights’ tab of the spreadsheet to the ‘Past Flights’ tab.



Finally, the program will send the user a confirmation email and give the passenger’s boarding position as shown.



Challenges

There are two pieces to this program that add an element of potential user error and extra setup that the developer would have preferred to avoid. They are as follows:

#### User must customize VBScript

The VBScript that is called by the Task Scheduler has a line of code that seeks for the path to this excel file. This path will vary between users and must therefore be modified in the code. Although the file path only needs to be changed once, it can be mildly complicated for end users who aren’t familiar with VB.

#### User has to manually create system task to set time for program to run

This restriction somewhat limits a key benefit that the developer hoped to offer their end users. What this means is that the user will have to manually enter the Task Scheduler to set the time to run the program (24 hours before their flight) instead of the excel program setting up the program itself. Although simple to use, most end users are most likely not familiar with using the task scheduler.