# Learning Suite Schedule Importer

Jacob Durham

**Executive Summary**

As a student at BYU you keep track of all of your class assignments, quizzes and exams through Learning Suite. Learning Suite is a great website, however when you have 5 different classes, getting a grip on everything that you have to get done over the course of a week or a day can get frustrating. Learning Suite does offer an iCalendar Feed so that you can easily upload your schedule into your Gmail account calendar however, this functionality still left me frustrated because when you upload your iCalendar Feeds it is very difficult to see everything that you have to do because each date box is so small that only about two or three schedule items can be seen at once. So in order to make this process easier, I created a macro that enables the user to import their schedule for a given date range that they set via a user-form for all of their classes. Then if the user wants to, they can easily manually create to do lists complete with due dates and assignment names extremely easily because all of their class schedules are in a format that can easily be edited and manipulated in one excel spreadsheet.

**Implementation Documentation**

A user will find that using this program is extremely easy. The steps are as follows:

1. When the User opens the file they can start the program by clicking on the “GetSchedule” tab located at the end of the tabs in the ribbon. Then the user will click the “Get Schedule” button that will appear after the “GetSchedule” tab is opened.





1. Once the “Get Schedule” button is clicked a user-form will appear. The user-form asks for the user’s BYU Net ID and Password and then allows the user to select the schedule date range through a graphical calendar. The Net ID and Password will be stored in a place in memory then used later to log into their Learning Suite account. As you can see the user’s password is protected by replacing every letter that is typed, with an asterisk. Once the user fills in the fields and clicks “OK” the user can sit back relax and watch their schedule for all of their classes import into the excel spreadsheet.

The finished product is displayed in the image below. As you can see the date range is placed at the top for the user to reference the date range that they entered. Also, the schedule has been displayed in a manner that allows the user to easily understand which class schedule and day they are looking at through the use of bold letters, colored cells, and borders. All of the classes are bolded and put above their respective schedule and along with the headers they are highlighted in gray for the user to easily see the divide between classes. Then in order to allow the user to easily see what schedule item relates to what day, there are borders surrounding each individual day.

The process of running this program is extremely easy for any user and will enable students to get a grasp on their workload with very little effort.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **NetID:** | **jacobpd** |  |  |  |
| **From Date:** | **4/1/2013** |  |  |  |
| **To Date:** | **4/2/2013** |  |  |  |
| **ACC 540 - Advanced Financial Accounting** |   |   |   |
| **Date** | **Session** | **Reading Assignment** | **Homework Problems** | **Subject** |
| 4/1/2013 | 22 | Ch. 13, pp. 686-697 | E13-1, 2, 3, 11 | Government Wide Reporting |
|   |  |  |  |   |
|   |  |  | Chapter 10 Homework (2) Opens | Governmental Case Part 3 |
|   |  |  |  |   |
|   |  |  | 7:30 AM | Individual Quiz 7 |
|   |  |  | Chapter 13 Homework Closes |   |
|   |  |  |  |   |
|   |  |  | 11:59 PM |   |
| 4/3/2013 | 23 | Ch. 10, pp. 506-521 | Chapter 11 Homework Opens |   |
| **ACC 550 - Fraud Prevention + Detection** |   |  |  |
| **Date** | **Topic** | **Assignments** |  |  |
| 4/1/2013 | Student Team Reports--Being an Expert Witness | Expert Witnessing in a Fraud Case |  |  |
|   |  |   |  |  |
|   |  |   |  |  |
|   |  |   |  |  |
|   |  | 11:59 PM |  |  |
| 4/3/2013 | Student Team Reports--Being an Expert Witness |   |  |  |
| **I SYS 515 - Section 002** |   |  |  |
| **Date** | **In-Class Files** | **HW Due** |  |  |
| 4/2/2013 | Business Intelligence |   |  |  |
| **M B A 614 - ISys 520: Spreadsheet Automation** |   |   |  |
| **Date** | **Content** | **Assignments** | **Class Examples** |  |
| 4/1/2013 | Ch. 17: Automating Solver |   | portfolioOptimizer\_w13\_a.xls Download |  |
|   |  |  |   |  |
|   | portfolioOptimizer.xls Download |  |   |  |
|   |  |  |   |  |
| 4/3/2013 | Manipulating the Excel Ribbon |   | ribbon\_w13\_a.xlsm Download |  |
| **M B A 621 - Advanced Corporate Finance** |  |  |  |
| **Date** | **Column Title** |  |  |  |
| 4/1/2013 |   |  |  |  |
| 4/2/2013 |   |  |  |  |

Now I would like to explain what the code is doing in order to make all of this work so easily for the user.

1. The Ribbon – I added the tab and the button in the ribbon using the “Customer UI Editor” that we learned about in class. When the “Get Schedule” button was clicked it started a macro that would show the user-form.
2. The User-Form – As stated above, the NetID and Password were stored in memory to be used to login to the students Learning Suite account. The Date range values were also stored in memory to be used later. Then clicking the “OK” button runs an executive sub which calls the main Sub Procedure to run.
3. The Main Sub Procedure – This sub procedures has multiple parts to it that I will explain below:
	1. First, all of the cells in the spreadsheet are cleared in order to prep the spreadsheet to get a new schedule.
	2. Internet Explorer is then opened, using the agent that Professor Allen provided in class, and it navigates to the Learning Suite login page. The username and password from the user-form are now used to login.
	3. After logging in, Learning Suite lists out the student’s classes. I imported this webpage into excel and loaded the class list into an array.
	4. Then using a for loop I did the following activities to each class in the array from step C.
		1. Using the agent, VBA follows links by text to the class schedule, then imports that schedule into a new excel spreadsheet.
		2. With this imported schedule, VBA grabs the headers of the schedule pastes it into the finished schedule worksheet, then using the date range given in the user-form, VBA copies the schedule for that date range and pastes it into the finished schedule spreadsheet that is shown above .
		3. Once this data is pasted onto this final schedule spreadsheet, VBA starts formatting all of the data to look more organized by adding borders, highlighted cells, and bolded font.
		4. With the conclusion of the formatting, the for loop has completed one loop for the first class in the array and will repeat that same process for each class in the array described in step C.
	5. Once the for loop has run through the full array, VBA then auto-fits the columns and ends the sub, at which time VBA goes back to the executive sub and the user-form is unloaded and the user is able to see their schedule.

**Learning and Conceptual Difficulties**

There were a few places along the way that I faced difficulties. The first which surprisingly seems to be the hardest part of any VBA project was simply coming up with a logical way to work through the problem. Knowing ahead of time what approach to take, which tools to use and when to use those tools is a very crucial step. I have found that if I take the time to really think through the entire project before actually writing any code, then I am able to save lots of time in the long run. However because I am new to VBA it was difficult to choose beforehand which tools I should use when I am so unaware of all of the tools that are available to me. I think that this will become easier as I use VBA more. Another task that was difficult but a very beneficial thing to learn was working with the Agent that Professor Allen provided in class. Understanding how it worked and how to interact with the agent took some time but once I understood it, it was extremely helpful. The borders were another part of the VBA code that made this project difficult. It wasn’t so much a matter of learning how to put a border on a cell that was difficult, but learning how to define a range that I want to have a border around. This exercise really taught me how to work with the more basic parts of VBA more competently. Finally the last struggle that I had was with the Calendar in the user-form. I was able to set up the calendar and make it work just fine in my 2007 version of excel, however it turns out that this user-form control is not available in Excel 2010. So when I emailed my macro to others to test out, it wouldn’t work. After spending some time looking for Date-Picker controls that would work in 2007 and 2010 for a while I discovered that there was no type of date-picker available in 2010. So I ended up making two versions of my project. To make it work in the 2010 version I ended up asking the user to enter the desired date range in a specified cell on the actual finished schedule spreadsheet. Once I did this and re-wrote the code to reference those cells the program ran perfectly on 2010 versions as well. This was disappointing because I think the graphic calendar in the 2007 version looks much better, but after researching online, it looks like the only way around this is to create my own date picking user-form control which was outside of my skill set.

**Assistance**

Throughout my project I would say that there were 3 people who helped me the most

* Professor Gove Allen – Basically helped me with everything in my project because I learned everything in class but in particular through one on one sessions, Professor Allen helped me to figure out the best way to approach my project by telling me to import the schedule webpage rather than working with the webpage source information. I believe that this probably saved me a few hours of time. Also Professor Allen helped me become more comfortable with the web agent.
* Dave Douma – Also helped me to understand how to work with the web agent and use certain methods like followlinkbytext and waitforload.
* Austin Dunn – Helped me figure out how to get my ribbon to work. I couldn’t get the new tab to show up in my ribbon, and he walked through the code with me to help me understand what each line of code meant and what it was doing.

**Conclusion**

Overall I would say that this project was a great experience for me because it caused me to work through a lot of the skills that we had learned in class. If someone had told me at the beginning of the semester that I would be able to create this program I wouldn’t have believed it. It is incredible how much can be done with VBA.