**1. Executive Summary**

Registering for classes at BYU can be a pain, especially if you are trying to get into a class that everyone else wants. There are two aspects to this, signing up for those classes at midnight on the day of your priority registration before it fills up and if a class is already full trying to be the first to get the open spot when someone drops the class! This program will allow you to display all of your current classes for a given semester and give you the option to sign up the classes at the click of a button or at a given date. You can also tell it to be persistent and keep trying to add your class every 15 minutes. You simply select the classes you want and the date you want it to register for classes, and the program will sign you up!

**2. Implementation documentation.**

For this project I had to learn quite a few different concepts including:

* Executing javascipt
* Navigate through multiple webpages
* Parse through HTML
* Creating navigable userforms

The overall concept is really straight forward and I thought it was going to be a pretty simple project, but there were some things that really made this a difficult project. My development process went as follows:

* Open Internet Explorer and login to the website
* Get currently enrolled classes
* Be able to change displayed enrolled classes
* Show classes that could be added
* Select class/classes to add
* Add selected class/classes manually
* Add class/classes on a schedule

Opening Internet Explorer and logging in were pretty easy to implement because those concepts were taught in class. I wanted to change the login module a bit though – what if the user types in the wrong username and password? So the challenge is what the user first sees. I decided to have a main form (Figure 1) and just have the user work off of that instead of the work sheets.



**Figure 1 - MainForm**

The main form allows the user to login. Clicking the “Login” button brings up the login form (Figure 2).



**Figure 2 – Login Form**

If the user enters incorrect credentials, an indication comes up on a different form that looks the same (Figure 3).



**Figure 3 – Login Form 2**

Once the user enters the correct credentials, the main form becomes enabled (Figure 4).



**Figure 4 – Main Form**

Note that Figure 4 also shows a list of semesters. The only items that become enabled at first are the “Show” buttons in each section. The idea is that the other 5 buttons only apply to new classes to be added, so they won’t show up until that “Show” button is clicked. Figure 5 below shows the results when “Enrolled Schedule” “Show” button is clicked. Figure 6 shows the results when “Classes to Add” “Show” button is clicked and there is one class that has been selected to add.



**Figure 5 – Main Form**



**Figure 6 – Main From**

If the user wants to have more classes to add, they need to click on the “Choose New” button and that will display the “Choose Semester/Department/Class” form (Figure 7).



**Figure 7 - Choose Semester/Department/Class**

The user then selects the Semester/Term, Department, and Class of their choosing and also selects a section from the list of selections relevant to their semester, department, and class selections. If they choose “Select” then the class is added to the “Classes to Add” list. If they choose cancel, then they are returned to the main form.

On the main form, the user also has the option to select “Add Now”. When clicked, the “Add Now” button calls a sub to attempt to add each of the classes selected to the user’s schedule. If successful, a dialog will appear that indicates which classes were added successfully (Figure 8). If unsuccessful, a dialog appears indicating which classes were unsuccessfully added (very similar to Figure 8); however, if the check box labeled “Persistent” is checked, then there will be no error message. Instead, if the class wasn’t added, it will try 15 minutes later and continue to do so until the user deselects the “Persistent” box. Note that if a class was added successfully, the class is removed from the “Classes to Add” list. Likewise, if a class cannot be added it is not removed from the “Classes to Add” list.



**Figure 8 – Classes Added**

The “Remove Selected” button merely removes the selected class from the “Classes to Add” list. If the user decides that the selected class is no longer desired or offered, then the class can be removed using this button.

The “Add Later” button opens a new dialog (Figure 9) that allows the user to set a time that the program will go and make an attempt to add the class.



**Figure 9 – Set Date**

The user selects a date from the Date Time picker control and then clicks “Set”. The dialog disappears and the Main From now displays the date that the classes will be added (Figure 10). “Unset” on Figure 9 merely removes that function from happening.



**Figure 10 – Main Form**

When the user is finished, they can exit by clicking the button labeled “Exit”.

**3. Learning and Conceptual Difficulties**

This project was particularly challenging because of the great amount of interaction with the web browser. Not only that, I had a very hard time determining the best interface for the user – userforms or the spreadsheets. I actually changed the presentation halfway through the project which let me clean up some of my code a little bit, but it also introduced some more errors into my project that I had to work out.

I am most proud of the parsers I built (with the help of the homework assignment, but I actually learned how to use them during this project) and the way I used the parsers to get information off of the webpages. I ended up using 6 different parsers to get this program working. This included using multiple objects to pull information about each semester, department, class, and section. Because the webpages had values to indicate the value *and* the display value, I had to pull both of those out and I put them in a multi-dimensional array so the user could see the human readable value and the program could return the actual value back to the website.

One function that I was particularly pleased with was the one that actually registers the class on through the javascript on the webpage. However, it is hard for me to really do it justice by pasting the code here because the underlying subs/functions that I call in that function also are part of the brilliance of this code.



EnterSemester, EnterDepartment, and EnterClass, are all support methods that I use throughout the program and it made it so easy to just make them universally useful. The params variable was pretty much given to me on the webpage as was the javascript function “aeSubmitLink”. If the aeSubmitLink throws an error for any reason, then the Function returns false. The ClassAdded Function checks whether or not the class was added and returns true if it was or false if it wasn’t.

I like this function because it is a good representation of my work on this project – making subs universally useful (can be called anywhere in the program without failing of halting the program), used the coding on the webpage to my advantage (even when code was a bear to decode), and using functions to determine the status of the program (the ClassAdded function uses a parser to determine if the webpage says “Class Added” or not).