Executive Summary

Every semester, BYU students bemoan the fact that they have to stay up for all hours of the night to register for the classes that they want. Many have found that if they don't click the "register" button at exactly the right time, midnight on the dot, their efforts are in vain, and they don't get the classes that they wanted to. For my VBA project, I chose to create a program that will allow a user to automatically register for classes so they don't have to stay up until midnight to register. My program allows a user to select which courses they want to register, and then after the user clicks on "register at midnight", the program will wait until midnight, and the second that their computer clock hits the stroke of 12, the program will register for those classes for the user. It also stores the username and password encrypted with a basic algorithm so that the user does not have to log in over and over again.

Implementation

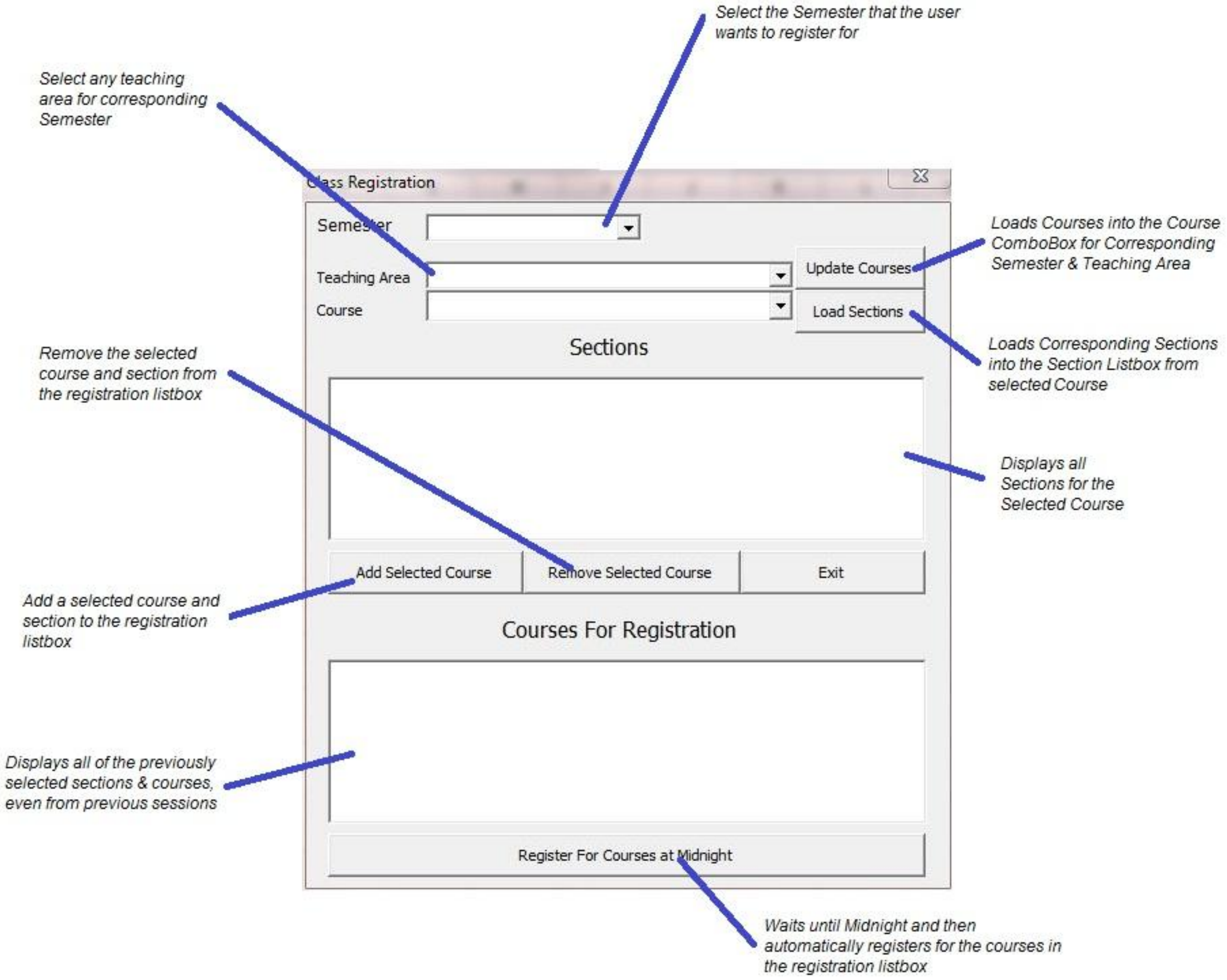
For this documentation, I will walk through the entire process of my program in detail, step by step, and highlight the tools that I used to create my solution. The program utilizes two forms (loginform & registerform), multiple spreadsheets for data storage, and two modules (one of which is the agent class module which I only changed per Dr. Allen's instruction in his office). Upon opening the workbook, the log in form is shown. At initialization, the log in form checks to see if an encrypted password has been stored, and if so it decrypts the username and password and fills in the appropriate textboxes. It is then up to the user to click "submit" to start the log-in process. The following table details the different parts of the log-in form and sub procedures.

Logging In

Subprocedure or Event	What it does
LogIn_Click()	Checks to make sure values have been entered in the fields, and sends values to Login sub after unloading the form
Cancel_Click()	Unloads the form
DecryptPass()	Checks to see if a password/username combo has been stored, decrypts it, and pre-fills the corresponding fields in the login form
Login (netID, netPass)	This is a robust sub that takes many actions: - Calls the storePass subprocedure - Clears the Old Agents and makes new ones - Uses Agent1 & Agent2 to open webpages, logs in, and opens linked pages - Populates Values for the Registration Form, and calls corresponding Subs - Shows registerform
storePass (netID, netPass)	Encrypts and stores current username/password in a spreadsheet
Semester()	Finds which semesters are available for looking up courses, reads the values, and populates the Semester ComboBox of the registerform

TeachingArea()	Finds which teaching areas are available for the selected semester by searching for where the data is located in the page source and reading each course dynamically. Adds the dynamic list to the Teaching Area combo box of the registerform

The Registration Form



Select the Semester that the user wants to register for

Select any teaching area for corresponding Semester

Loads Courses into the Course ComboBox for Corresponding Semester & Teaching Area

Loads Corresponding Sections into the Section Listbox from selected Course

Remove the selected course and section from the registration listbox

Displays all Sections for the Selected Course

Add a selected course and section to the registration listbox

Displays all of the previously selected sections & courses, even from previous sessions

Waits until Midnight and then automatically registers for the courses in the registration listbox

This form is where the bulk of the program takes place. It is the interface for finding, selecting, and registering for selected courses. It pre-loads any courses which were selected in previous sessions, and allows users to edit the list of courses they've selected previously by adding new courses or removing unwanted ones. The following table describes the aspects of the registration form:

Subprocedure or Event	What it does
AddCourseBtn_Click()	Reads the selected course & section from the Course combobox and section listbox, and adds the course and section to the registration listbox. Also stores values in a spreadsheet.
ExitBtn_Click()	Unloads the form

RegisterBtn_Click()	Calls RegistrationTime()
RemoveCourseBtn_Click()	Removes a Course & Section from the Registration Listbox as well as from the spreadsheet where it was stored
SectionBtn_Click()	Calls LoadSections()
UpdateBtn_Click()	Calls UpdateAll()
RegistrationTime()	Waits until 11:58PM of the same day, uses the loginform and login sub to re-log in to BYU's Class Registration page. Then waits until 12:00AM midnight and reads a Javascript string from a spreadsheet where it was stored for the corresponding course and section, and uses an agent to execute it in order to register for the class
LoadSections()	This is a fairly robust subprocedure with a lot of agent methods. It reads a course code from a spreadsheet where it was previously stored, and executes Javascript to submit the correct course value to BYU's server, then loads the page for that course. Then it searches through the source for all of the information for each section of the course, and stores it in a spreadsheet. It can handle up to 100 sections. Finally, it adds each section to the registerform.CourseListbox with essential information for the user (like the time and days the class occurs)– it stores the Javascript for adding the course in a spreadsheet, or if it doesn't exist (i.e. can not yet add the course) it creates the string based on if the course is for enrolled students, Day or evening, the credit hours of the course, Teaching Area, Course Number, and Section Number. Independent Study course functionality is not available at this time.
UpdateAll()	This is a core subprocedure that uses agents to update the registration form and executes javascript in BYU's registration page so that Teaching Area values and course values can be read and loaded into the registerform. It uses string functions to trim and manipulate strings which were read from the html source into a convenient format for the user. Calls the Courses() subprocedure
Courses()	Uses the agent to read the values of the names and codes for courses for a corresponding teaching area and stores them in a spreadsheet as well as populating the course combobox of the register form.

Learning and Conceptual Difficulties

I found this project to stretch me significantly. As an undergrad, I took an excel course for Chemical Engineers in which we learned some basic VBA, and over the summer I used macros to create a VBA program to automate some processes. Because of this, I found the beginning of the course something of a review. This project, however, introduced me to concepts far beyond the scope of anything I've ever done with computers. I was very fortunate to visit with Dr. Allen multiple times over the course of the past week as I have struggled to understand how to use the agent class, and how to sift through html source to find the few pieces of a webpage that I needed to read or manipulate to build my program. I feel that the learning I received about sending commands with javascript to be some of the most valuable of the semester. My only regret is that I waited until the last week to work on my project, due to some confusion regarding topic and scope (I had originally been attempting to do a Financial Program manipulating Mint.com, but couldn't access my bank accounts through the website – Mint.com's problem, not my computer's).

The following are a selection of tools I used to code the VBA (*are things I had not encountered before):

- Single and Multi-dimensional static and dynamic arrays
- Private and Public Subprocedures and variables (Integers, Strings, and Objects)
- Nested For.. Next, Do…Loop (including 'while'), If…Then statements
- String Functions and String Manipulation (Trim, Len, InStr*, Left, Right)
- Javascript*
- HTML source* (I attended the lecture on the subject, but agent didn't work on my computer at the time, so I needed to re-learn this aspect of incorporating IE with VBA)
- Application.Wait* and manipulating date and time algebraically* (this took a couple hours to experiment and learn – more than I expected)
- Passing Parameters by Reference
- Form Controls (Labels, Combo boxes, List boxes, Command Buttons)
- Event Procedures (Click, Change, Initialize, Terminate*)

My greatest problem with the program was that the agent gave errors, some of which we solved, and others of which we couldn't solve, even after spending 45 – 50 minutes together in your office attacking the problem. For some reason, we continually get two errors that have something to do with having multiple agents open, and attempting to attach agents to new windows: "Object Variable or With Block Variable not set" and "Object doesn't support this property or method." This has prevented me from completing the project, although I believe that once we resolve these errors, all of the code is in place for it to work. For the principle of the matter I would like to complete this project, and plan on spending extra time over the holiday with my brother (who majored in Computer Science) to have a finished project that my younger brother and sister (who will be attending BYU for the next few years) can use in any department, so that they do not have to wait until midnight to register for Classes.

I plan on adding additional functionality when we fix the error in the agent. For instance, the program could call the user's cell phone if a course is full, or the user could select an option which would make it so the program would continue attempting to add the course if it was full. I would also improve the aesthetics of the forms and spreadsheets. Finally, I would like to incorporate error checking and input checking to make the program as easy for users as possible.