Car Maintenance for Dummies

Morgan Fillmore

12/8/2011

Executive Summary

The Problem

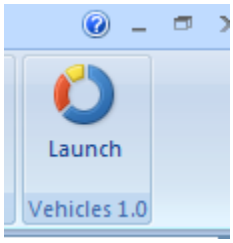
For me, vehicular maintenance has never been my forte. I never seem to take very good care of my cars. I never get oil changes when I need them, my tires stay bald for months, and I never seem to do any other preventative maintenance on my cars. Because I am far from being a car guy, it seems that car-care is always the last thing I ever think about (if at all). There is simply too much to think about, and I never seem to remember to take care of my car unless I'm embarking on a lengthy road trip. This needs to change.

Proposed Solution

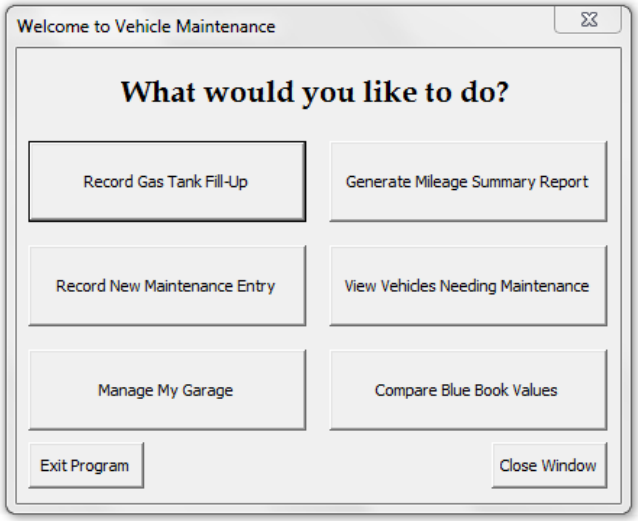
Presenting *VBA Vehicle Service, ver. 1.0*. I endeavored to overcome my weaknesses by creating a car maintenance solution for dummies. This program provides a user-friendly reporting system that will help the average dummy track their gas mileage, oil changes, tires, tag renewal, and other routine maintenance items for all of their vehicles. My wife and I recently made a decision to sell one of our previous cars and purchase a new one. This was quite an experience for us. One thing that I wish I would have had at the time was a simple tool that would help me analyze & compare different cars on the cost criteria while also factoring the value of a car we decided to trade-in.

This program offers a simple solution for a user to easily keep track of their vehicular maintenance. The successful implementation of this tool is dependent on the user's ability to routinely make an entry whenever they fill up their tank. As long as the user makes these simple entries, the program will do the rest. The program offers a number of user forms, web queries, reminders, and even the ability to post results to a blog. Below you'll find a detailed description of the program's core functionality and features that will assist in the implementation of VBA Vehicle Service 1.0

Project Overview



To start the program, the user need only click the button labeled "Launch" under the Vehicles 1.0 section of the Home tab. This brings up the main Home Screen or launch pad to access the functionality of the program.



In order to describe the features in a logical flow, I will describe the process as if someone is new to the program and adding a vehicle to manage for the first time. Below is an outline of this flow that will guide my detailed summary in the remainder of the report.

1. **Manage My Garage**
    a. Add a Vehicle
    b. Remove a Vehicle
    c. Maintenance Preferences

2. **Record Gas Tank Fill-Up**
    a. Select a Vehicle, enter data

3. **View Vehicles Needing Maintenance**
    a. Send a reminder

4. **Record New Maintenance Entry**
    a. Select a vehicle, enter data

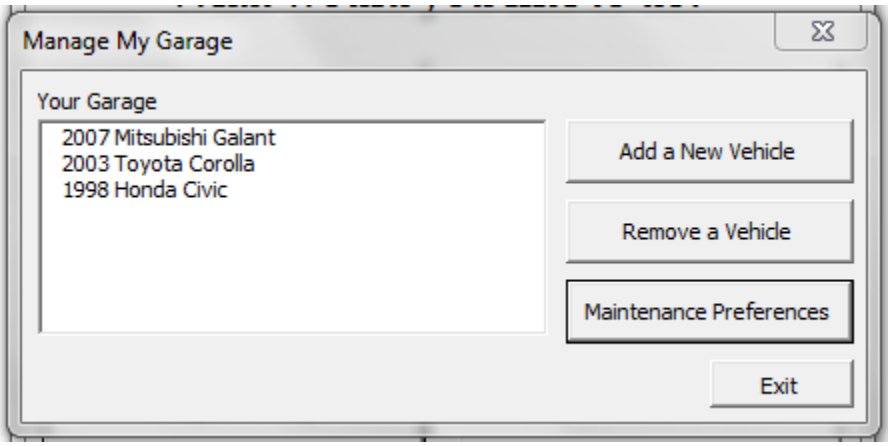
5. **Generate Mileage Summary Report**
    a. Select a vehicle to summarize
    b. Post to Blog & View the Blog

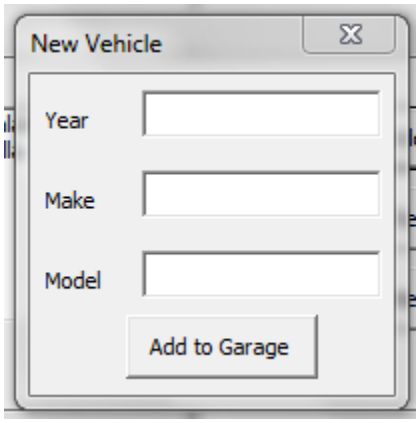
6. **Compare Blue Book Value**
    a. Select a car from your garage
    b. Enter a different vehicle of your choice

At the end of this report, you'll find an analysis of the learning and difficulties that were encountered in the design and programming of this VBA application.

Manage My Garage

This section gives the user the ability to add all the vehicles you wish to monitor. The user form allows you to add a vehicle, remove a vehicle, and edit your vehicles' maintenance preferences. All data entered into these forms is stored on the worksheet titled "My Garage."

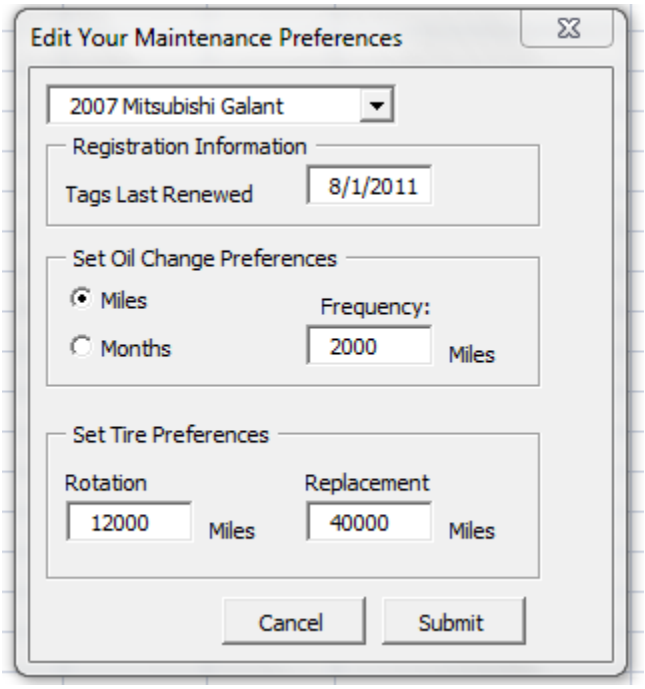




Add a Vehicle – When the user chooses to add a vehicle to the system, the following user form pops up and asks that the user enter in the year, make, & model of the new vehicle.

Remove a Vehicle – Should the user want to remove a vehicle from the garage, the user can select an item from the list box and simply hit the Remove a Vehicle button. This procedure removes the vehicle from the list, the My Garage worksheet, and all other sources within the sheet where this vehicle's information was stored.

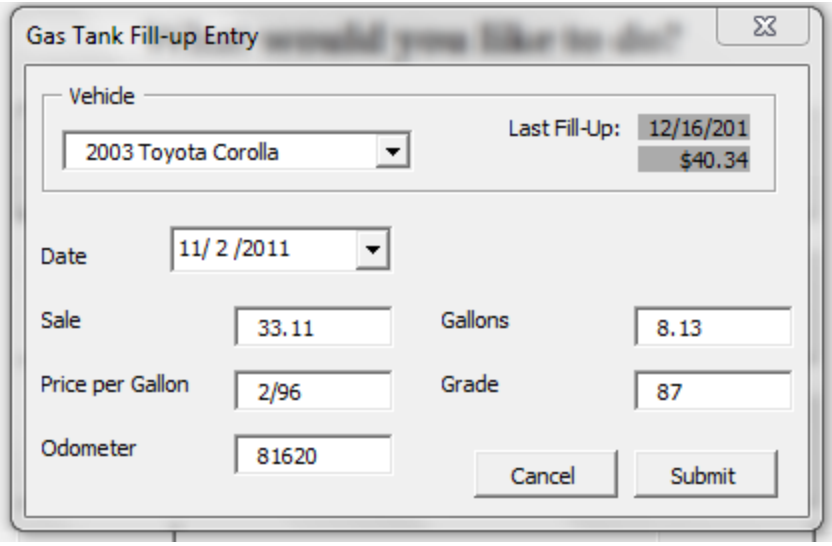
Maintenance Preferences – This program is designed to give the user the ability to set the frequency of routine maintenance per vehicle. This user form is quite simple. Upon selecting a vehicle, from the drop down list, the form automatically populates itself based on the current preferences that have been selected for the vehicle.



By clicking on the 'submit' button, the changes are saved and stored in the My Garage sheet. The form does not close upon submission, so it allows the user to make any changes to other vehicles as well. The information selected in the preferences section is applied in the program to help determine when the user should bring the car in for service.

Record Gas Tank Fill-Up

This is the key step to successfully utilizing this program. Every time the user fills up the gas tank, the information about the fill-up is entered and the odometer is recorded. By doing so, the program keeps track of the vehicle's most recent odometer reading in order to use it for maintenance & analysis.

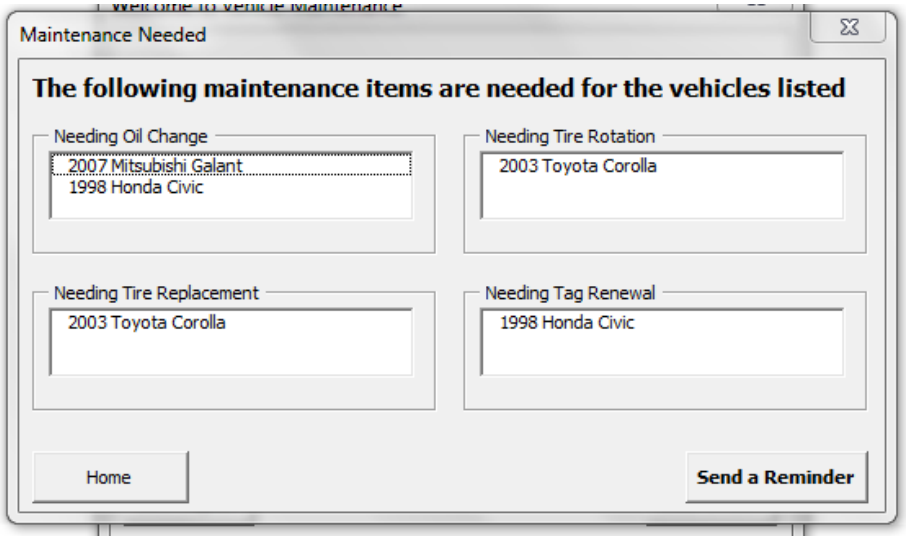


Upon submission, the information is stored on the respective vehicle's sheet as a new line item in the data table, (Row 12 below).

	A	B	C	D	E	F
1	Date	Sale	Price/Gal	Gallons	Odometer	Grade
2	11/22/2011	36.35	3.14	10.78	78987	87
3	11/28/2011	38.71	3.1	11.23	79269	87
4	12/2/2011	37.29	3.1	10.67	79550	87
5	12/5/2011	36.31	3.02	10.8	79809	87
6	11/2/2011	36.67	3.03	10.12	80047	87
7	12/7/2011	35.47	2.98	9.98	80311	87
8	12/9/2011	36.66	3.01	10.56	80596	87
9	12/12/2011	34.33	2.99	9.67	80765	87
10	12/13/2011	38.9	3.1	10.89	81034	87
11	12/16/2011	40.34	3.01	10.1	81322	87
12	11/2/2011	33.11	2.96	8.13	81620	87
13						

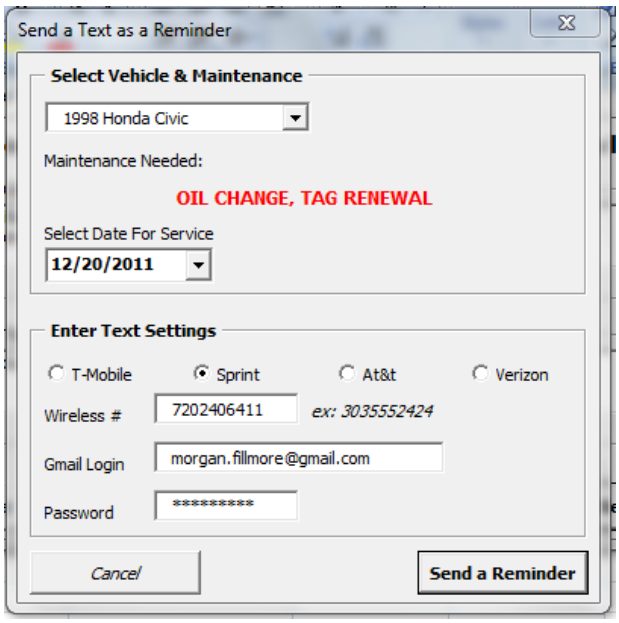
View Vehicles Needing Maintenance

Every time the odometer is logged in the system, the program analyzes the data to inform the user if any maintenance is needed. If nothing is currently needed, the program does nothing. If, however, the 2003 Toyota Corolla is now past the "odometer due date" for an oil change, the program would automatically open the following form as a notification. This form will show the user ALL of the current maintenance needed for ALL the



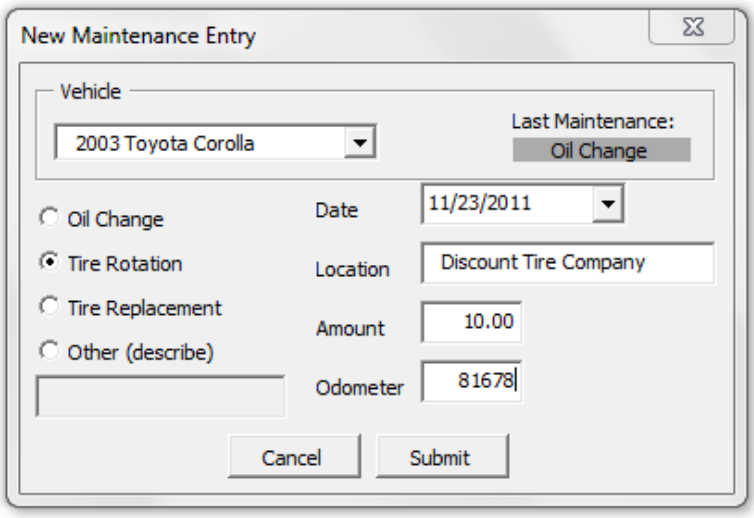
vehicles in the garage. Additionally, if there is any maintenance needed, the form will automatically open every time the workbook is opened to serve as an added reminder.

The maintenance data that populates into the user form is taken from the 'Summary Maintenance' sheet, which is calculated based on the user-defined preferences and the odometer readings based on the last time the user did a repair. Finally, as an added feature for ease of use, the program allows the user to send himself a text message as a reminder to schedule a time to get the needed maintenance completed. Below are the screen shots for the message window, and an example of a message that would be sent. As seen below, the form allows an option for 4 different cell-phone providers.



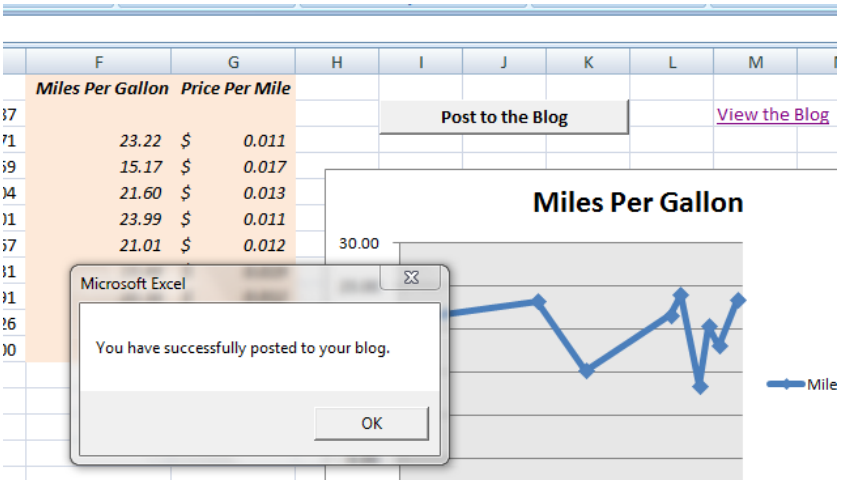
Record New Maintenance Entry

Whenever the user performs a maintenance item, there is a specific form to use to enter this data. By doing so, it provides accurate data as to when (and at what mileage) the event took place, which in turn helps project and track the next due date. The form has simple code written into it to prevent simple mistakes from taking place, such as entering an incorrect data type, or submitting the form with empty values.

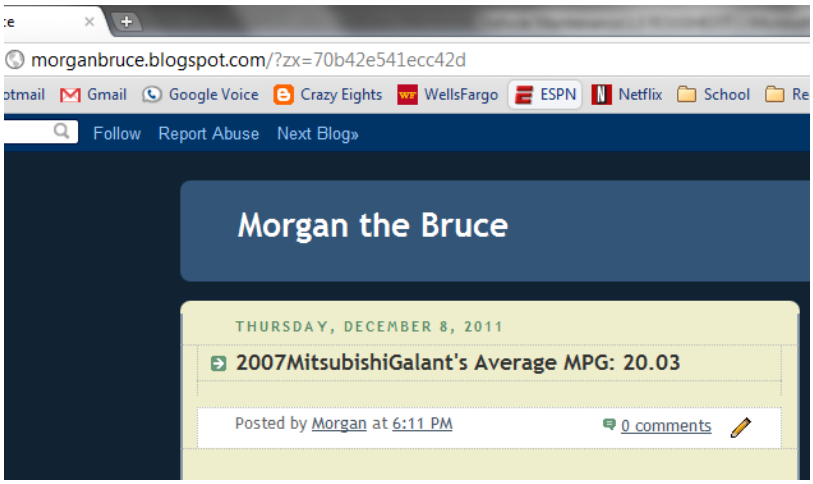


Generate Mileage Summary Report

The simple user form allows the user to select a vehicle, and then can click generate a report. The report pulls all the data for that vehicle, and calculates average Miles Per Gallon and average Price Per Mile and plots both sets of data into a chart. Below is a screenshot of a sample summary.

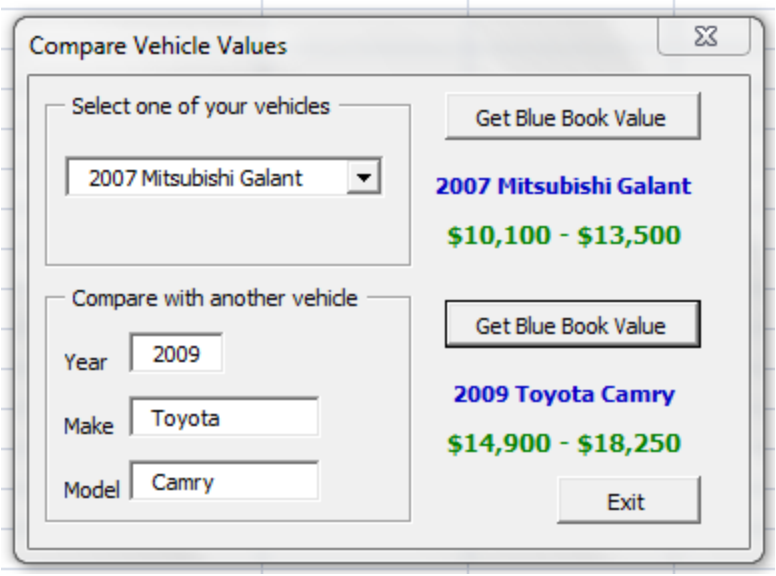


If the user so desires, they are now able to post the results to a blog. In this instance, I have only set it up to post to my own personal blog (through Blogger), and not the blog of choice for the user. It should also be noted that the functionality that posts data to a blog can take up to 30 seconds for the task to be completed. Therefore, after the blog has been posted, a message box appears to notify the user that the post was created successfully, and a link to view the blog then appears. This summary sheet is the key data output from this program and provides the user with a quick & simple overview of the type of mileage and costs that their vehicles are incurring. The data is not complex, and is designed to be easily understood.



Compare Blue Book Values

The last option available on the home screen allows the user to compare the value of one of their cars with the value of another car. This portion of the procedure performs a simple web query on www.autos.msn.com , and reports the information back into the original user form. Because the data for car values can change very frequently, this comparison is not saved as data in the workbook. This can be a useful tool when evaluating whether or not to buy a used car and to get an idea of what your current vehicle is worth (should you decide to sell it as well). The form shown below is the result of having performed two searches.



The process to retrieve the Kelley Blue Book information can also take up to 20 seconds, and the spinning of the cursor serves as the only indication that the query is working properly.

Learning & Conceptual Difficulties Encountered

There were many things that I learned through the completion of this project. The following list is not a comprehensive list of things I learned or challenges encountered, but it was the list of the most important items I gleaned from completing my VBA application.

Planning – I learned that VBA programming requires a great amount of planning and an 'end vision.' When I first started writing the code, I knew most of the features and functionality that I wanted to include, but I didn't have a complete vision of how I wanted things to appear. As a result, I found that I frequently had to re-write my code and change my formatting to have it appear the way I wanted. I realized early on that I at least needed to make my code be flexible enough in case I needed to make changes later. For example, I tried to get in the habit of creating a worksheet object variable and setting the variable to whatever sheet I wanted. In this way, if I decided later on that I wanted it to reference a different sheet all I had to do was change the single line of code that set the variable, rather than find every reference to the sheet in my code.

Dynamic Functionality – Another big thing that I learned was that my code needed to be dynamic. By this I mean that I wanted it to behave in the same manner no matter who was using the program. At first, I began writing the code specifically for me, and not for anyone else. Initially I was only referencing data that was specific to me: only my cars, my information, etc. I figured that if I needed to make a change, I could always go back into my VBA code and change whatever I needed. Instead, I decided my program would be more functional if I allowed it to be catered to different users. I was able to allow the user to add and remove vehicles, change preferences, and so forth. One of the biggest difficulties I encountered in this process was in creating specifically tailored sheets every time a new vehicle was created. I found that it was often hard for me to write code for objects that weren't in existence yet. As a result, I learned the benefit of using object libraries and indexes to reference items (sheets, objects, charts, etc.). This particular item was a valuable learning experience for me.

Automation – I learned a lot about automation in this process. In general, I found that the more the program can do for you, the better. One of the features that I really wanted to have in my program was that a user form would automatically open when a certain event had happened. Specifically, I wanted my 'Maintenance Needed' form to open if the odometer value became greater than the predetermined cut-off point that indicated it was time for maintenance. After meeting with Dr. Allen, he taught me how to work with sub procedures that execute specifically on a given sheet, and how to get them to function on a certain command. I was able to incorporate this part into my code, but I experienced a hiccup because I needed this event to occur based on the output result of a cell formula. Having gotten to the point I was at, I discovered that my code was too convoluted at that point to lend itself to a simple fix for this feature. Eventually, I was able to provide a work-around that would provide an adequate result. Learning how to execute procedures, or functions when an event in excel happens was an awesome tool to learn.

Internet Automation & Blog Posting – When we covered the unit on manipulating internet explorer, I felt like I knew how to do everything. But when it came down to figuring out how to post content from my spreadsheet to a blog, I learned that I still had a lot to understand. I learned a lot about how to parse through the html source of a webpage to find the right objects or fields to work with. One of the hardest things I encountered in this regard was finding out how to manipulate the different buttons and menus/submenus that appeared on the actual website. My first attempt was using wordpress.com as my blogging site, but that turned out to contain far too many menus and 'hovering' sub-menus that proved to be difficult to find in the source code. I then tried using blogger (via Google), and that was more simplistic, but the code was harder to understand. After getting some help from Dr. Allen, I was able to fully grasp some efficient ways to reference points in the source code and how to manipulate it the way I wanted. Another great challenge I faced was in inserting an image file to be posted on the blog. My desire was to be able to take my charts from my spreadsheet and post them as the body of the blog entry. Ultimately, this proved too difficult a task so I resorted to summarizing my blog data in a different manner.

Grab-Bag – In a more general, all-encompassing sense, I learned many other smaller items/habits that have greatly helped the way I write code. I learned how to write code more efficiently, such as using the '.end(xlup)' command instead of '.end(xldown)' in order to select a range. I learned that the 'xldown' method often caused an overflow error. I learned how to better use do-loops, for-next loops, for-each loops, and the select case construct. I learned that code for manipulating text strings is extremely beneficial. I learned how to use the 'instr' command like never before as I attempted to assign names to different fields, or populate different cells or user forms. I now feel extremely comfortable working with user forms. But as a caveat to that, I also learned that forms have their limitations as well. I encountered a couple of runtime errors that were hard to investigate, but through that I learned some more efficient ways to write code.

I ran into a lot of issues as I designed this program, but one thing that I am most proud of is how I learned many different methods of accomplishing something as a result. I discovered many alternative solutions to various tasks when I ran into a particular block of code that didn't function how I had hoped. I was proud of my ability (even though it often took hours) to solve a problem in a different method. It was very rewarding.

Conclusion

In summary, I am very pleased with the end result of my VBA application. I am pleased with its ability to be tailored to other people. I kept this firmly in my mind as I programmed, and I feel that it is a product that others can easily use. I have told my brother about this program and he is already excited to use it. Obviously, there are things that I will continue to add to the program to enhance the functionality, but I am pleased that my program accomplishes everything I initially set out for it to do, and then some. Version 1.0 has just barely rolled out, but my mind is already filled with ideas for expansion and improvement in version 2.0.