2011

Media Maven

Physical Media Inventory Tracker

Paul Brems
Dr. Allen
4/13/2011

Executive Summary

Although nearly all forms of media are being consumed by the “digital revolution,” there are still remnants of times past that we all own – physical CDs, DVDs and books. Digital media managers are commonplace and can be found on every personal computer system. iTunes, Windows Media Player, WinAmp, MediaMonkey, etc. are sophisticated applications and are free to boot. Similar applications to keep track of physical media are more difficult to come by and usually come with a fee attached.

Media Maven is a VBA application that looks to fill that need. It easily and intuitively tracks physical books, CDs, and DVDs. Once entered into the excel “database,” the inventory is easily searched and editable, all through a user form-driven interface. In addition to tracking the inventory that you own, you can also specify a location if you so desire to have more fine-grain control over your inventory. In addition, Media Maven allows you to keep track of the physical media that others have borrowed.

This program was created with my needs in mind but I believe that anyone who wants a capable, portable physical media inventory application will find Media Maven extremely useful and easy to use.

Implementation Documentation

Media Maven consists of a number of different functions. I will first discuss that application as a whole and then each of its function in detail.

Overview and System Requirements

For full functionality, the following is a list of requirements for the application to run.

- Excel 2007 or higher
- CueCat barcode scanner
- ImageMagick (download and install this one)
- Two directories, one named “tmp” and the other “img,” in the same directory where the workbook resides.

As stated previously, Media Maven is a user form-driven application. I debated whether or not to allow user input. I decided on a hybrid solution. Upon opening the Excel workbook, the application runs automatically (after enabling macros, of course). All of the processing can be done within the user form interface. If the user wants to enter data the user can simply exit the user form interface. This is, however, highly discouraged because if data is entered incorrectly, it could prevent the application from functioning properly. Also, if the application is closed, the only way start it again is to either open up the Visual Basic Editor and run the proper sub procedure or to close and open the workbook. In future iterations, I may decide to bind the “exit” method with the “close” function of Excel.

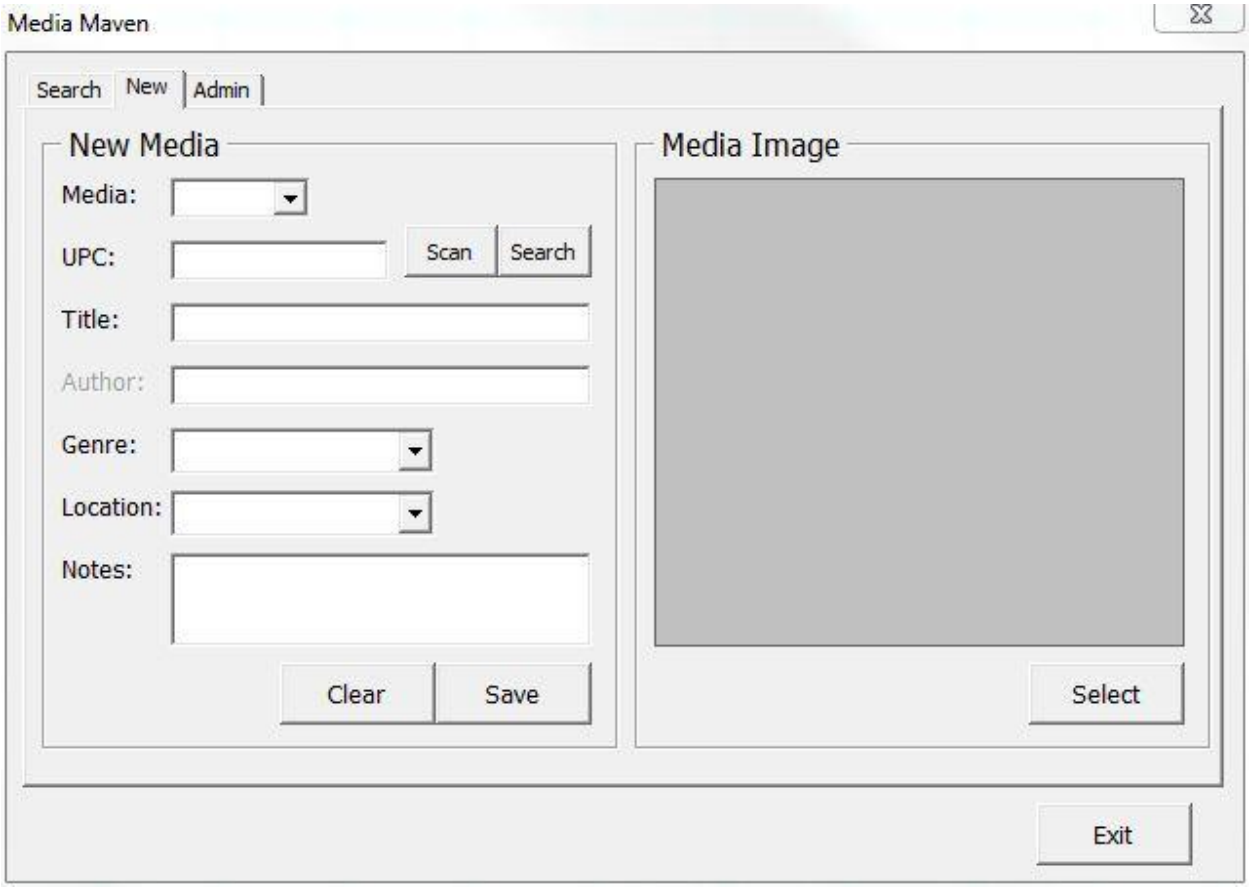

Figure 1 Main Interface

The main interface (Figure 1) contains three tabs – Search, New, and Admin. A majority of the functionality of the application occurs within those three tabs. A few other user forms, which are initiated from one of the three tabs in the main interface, will be covered later.

Adding New Media

Adding new media to the Media Maven database is fairly straightforward. The user has the ability to input all of the data manually, including selecting an image. Changing the value in the "Media" combo box triggers an event that changes the input fields according to the media type selected. If a user decides to their own image by pressing the "Select" button, the application will automatically make a copy of that file and put it in the applications "tmp" directory. Once in the "tmp" directory, ImageMagick is called from the command line to resize it to fit the Media Image area. The copied, resized image is only saved permanently when the "Save" button is pressed, moving the file from the "tmp" directory to the application's "img" directory.

```
Private Sub selectImgBtn_Click()

    Dim userImg As Variant
    Dim userImgName As String
    Dim path As String

    userImg = Application.GetOpenFilename(filefilter:="Picture Files (*.gif;*.jpg;*.bmp),*.gif;*.jpg;*.bmp", title:="Select an Image")

    If userImg = False Then
        Exit Sub
    Else
        If hasImage = True Then Kill ThisWorkbook.path & "\tmp\" & imgName
        imgName = Mid(userImg, InStrRev(userImg, "\") + 1)
        FileCopy userImg, ThisWorkbook.path & "\tmp\" & imgName
        'path = Chr(34) & "C:\Program Files\ImageMagick-6.6.9-Q16\mogrify" & Chr(34) & " -resize 139 " & Chr(34) & ThisWorkbook.path & "\tmp\" & imgName & Chr(34)
        path = Chr(34) & "C:\Program Files (x86)\ImageMagick-6.6.9-Q16\mogrify" & Chr(34) & " -resize 139 " & Chr(34) & ThisWorkbook.path & "\tmp\" & imgName & Chr(34)
        Shell path
        Application.Wait (Now() + TimeValue("00:00:01"))
        mediaImg.Picture = LoadPicture(ThisWorkbook.path & "\tmp\" & imgName)
        hasImage = True
    End If


End Sub
```

Figure 2 The code behind the "Select" button.

A problem that I ran into was that even though the image was copied and resized before being placed in the Media Image area, it would still display the original size of the image. To remedy this, I added a one second wait time to the application (Application.Wait).

Another method of adding new media to Media Maven involves the convenience of a barcode scanner. The CueCat barcode scanner was implemented into the project. To scan the barcode or ISBN of the media item, the user simply pressed the “Scan” button and an Input Box appears. Upon scanning the barcode, Media Maven will decrypt the scan with the CueCat module from class and input the plain text number in the UPC text field.

Without inputting any other data besides the UPC or ISBN, the user can press the “Search” button. Media Maven will scrape Amazon.com or Campusbooks.com and retrieve all of the information (with exception of the Genre and Location combo boxes) and a picture that is saved to the “tmp” directory and resized (Figure 3). That image will only be moved to the permanent “img” directory if the “Save” button is pressed.

The Genre and Location fields are customizable to the user. Any new value inserted here will be saved in the Media Maven database and available to use again after saving media item. These values are administered (deleted or modified) on the Admin page.

ISBN and UPC

Media Maven accepts ISBNs and UPCs of the most common length – 10, 12, and 13 digits. Any other length digit will produce a Message Box and won’t allow the user to continue. The following describes the way Media Maven uses the UPCs and ISBNs.

On some books there are two different barcodes – one for the UPC and the other for the ISBN. No matter the case, the ISBN is the preferred identifier of the book. Even within the ISBN there are two formats that include an ISBN-10 and ISBN-13. In the past decade, most books either have both types of ISBN on the cover or strictly use the ISBN-13. The ISBN-13 format is easy to spot because it usually starts with the digits “978.” This is the preferred ISBN format for Media Maven. If it is not available, the ISBN-10 should work OK.

For CDs or DVDs, the user can simply scan the UPC with the barcode scanner or type it in to the field and press “Search.”

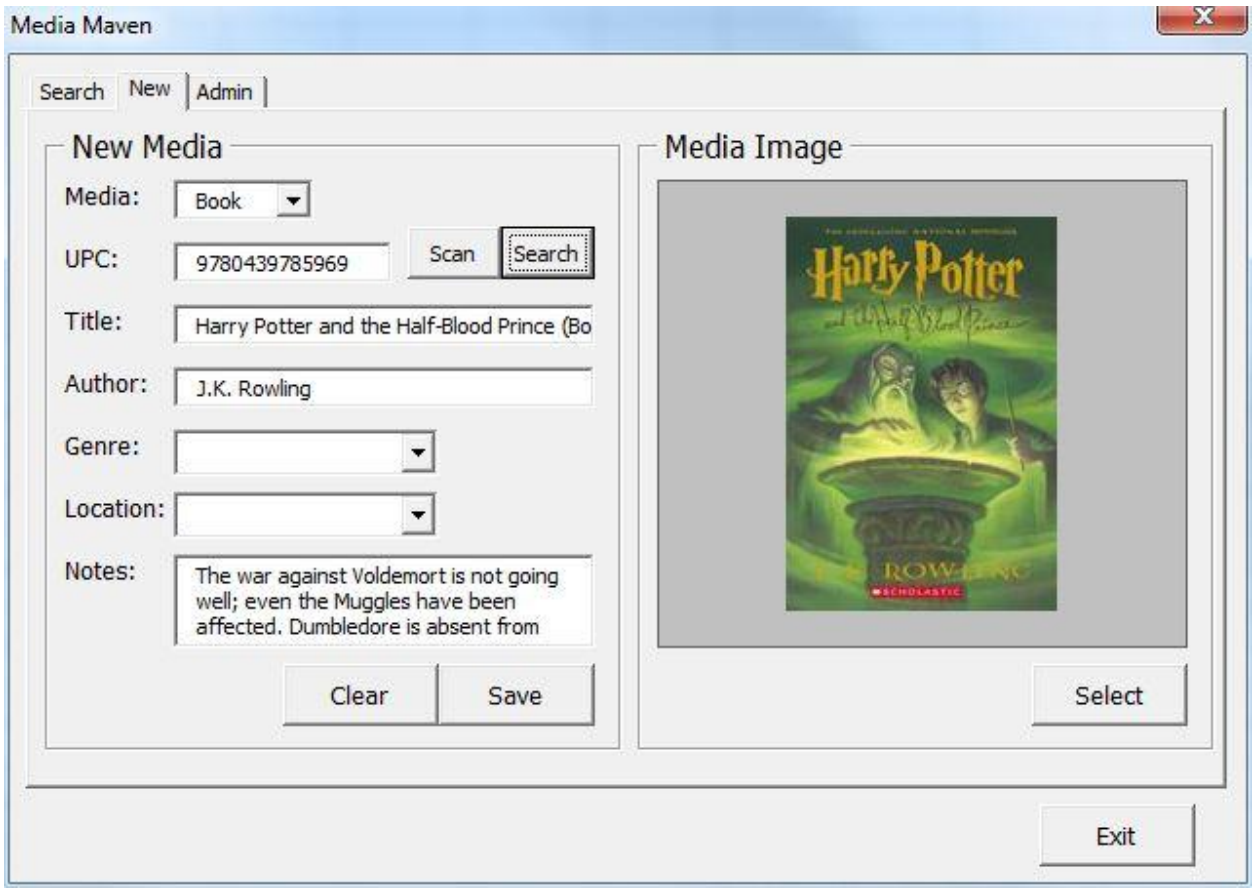

Figure 3 A screenshot of the New Media interface after performing a web search.

Searching Media

Once there are media items in the database, the user can switch to the Search page to search for a specific item or items. Here the media type must be selected. The search can be limited to one type of media (book, CD, DVD) or all of them at the same time (All). The fields will change depending on the value in the “Media” combo box and any combination of the available fields can be used to perform the search.

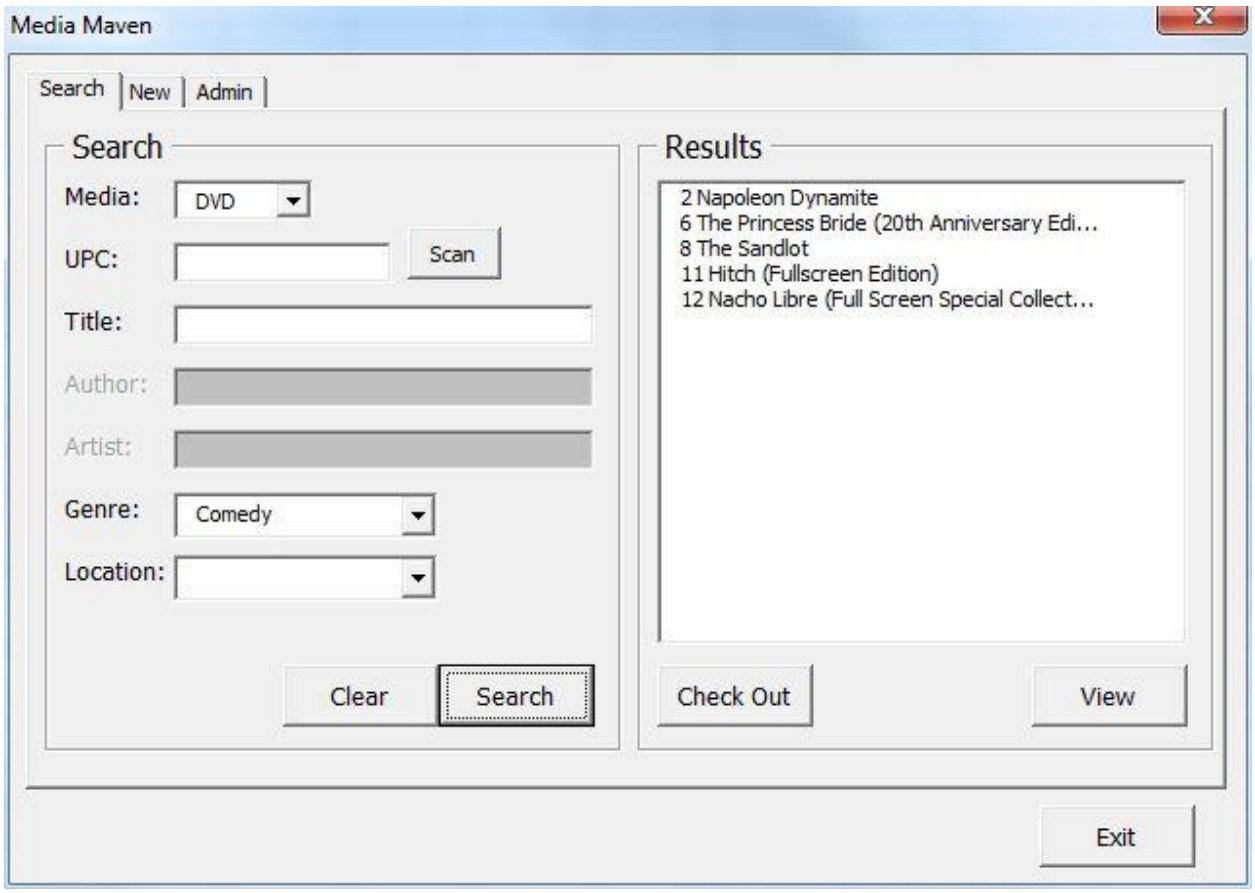

Figure 4 A completed search.

To view an item in the Results list, the user selects an item and either double clicks or pressed the "View" button. This will bring up another window with all of the item's details that looks similar to the "New" page. In this "View" form, the user can edit the record as needed, including selecting a new image (Figure 5).

Just below the Results list on the left is a button called "Check Out." Media Maven provides the ability to keep track of media that has been borrowed by others. Selecting a result from the list and pressing the "Check Out" button will bring up a form to fill out certain details, including contact information of the person who borrowed it, the date it was borrowed, and the date it is expected to be returned. Once checked out, a new record will be entered on the Borrowed worksheet and the Borrowed Media list on the Admin page. It will also show "(Checked Out)" on the Search Results list.

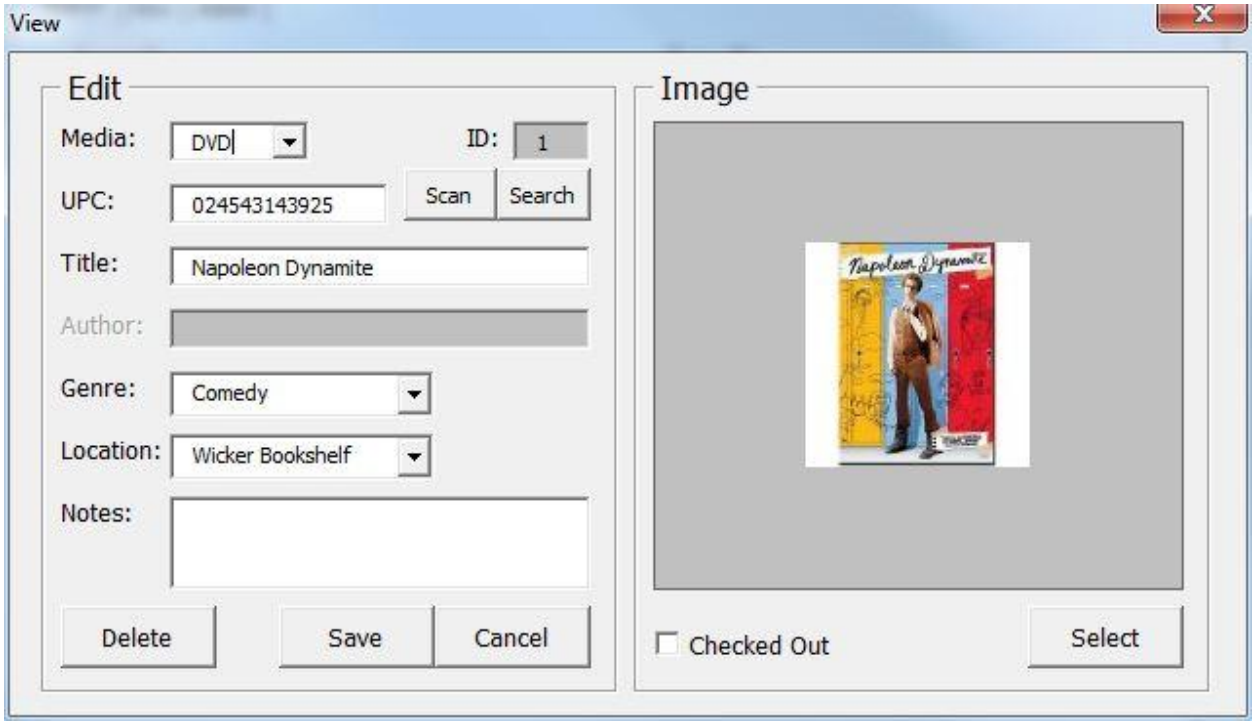

Figure 5 View and edit details.

Admin Page

The Admin page is where a few administrative tasks are performed (Figure 6). Genres for the different media types can either be renamed or deleted. The same option is also available for Locations. The Admin page is also where a running list of borrowed media items is kept. By selecting an item in the list and pressing the "View" button, the check out details appear. Here you can modify those details or check the book back in. If an item is out for 21 days (3 weeks) past the time you expected it to be back in your possession, "(Late)" will appear next to the item in the list. In addition, each time the workbook is opened while there are late items, a notification box listing those items will appear before the main interface is displayed. From the notification list, an item can be viewed and modified. In the future, I would like to implement an email function where you can email the person who has borrowed your item as a reminder.

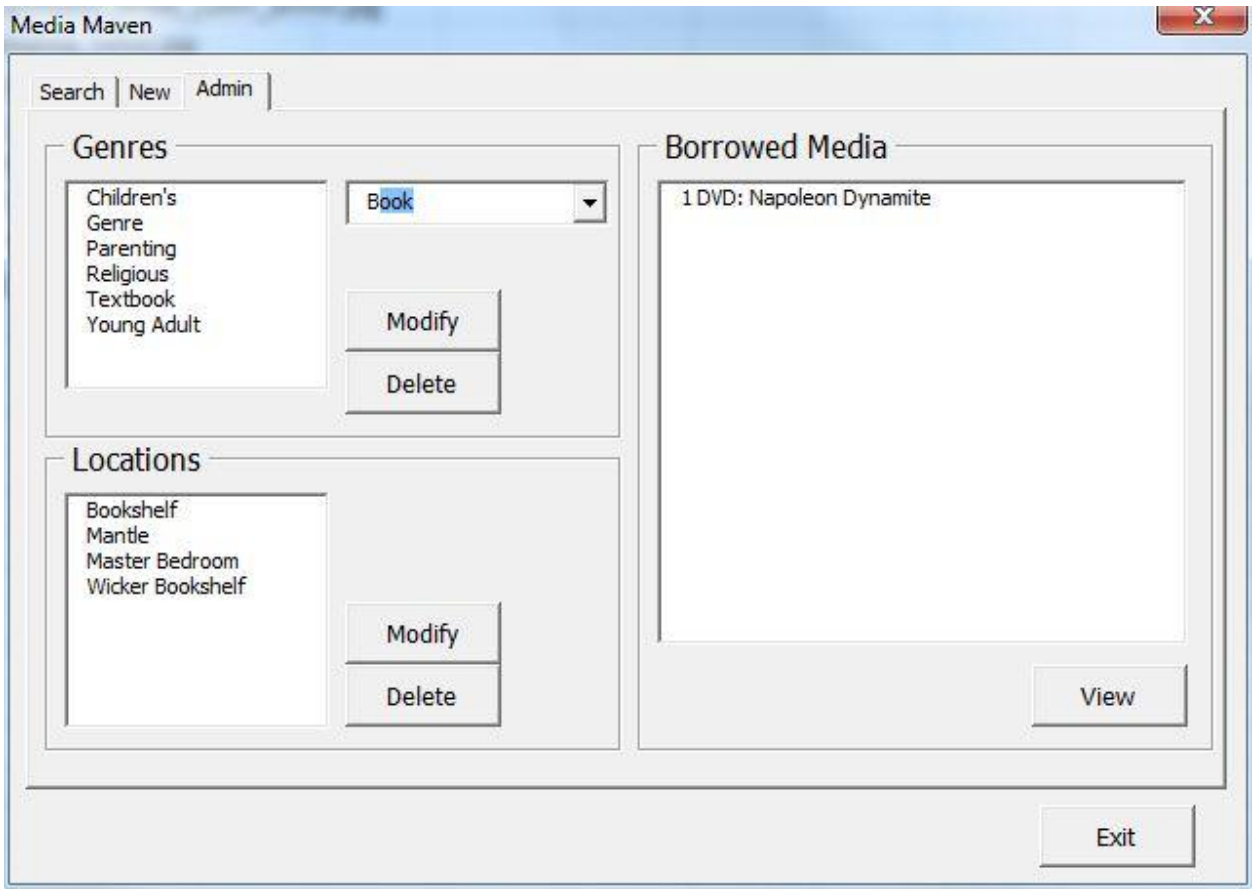

Figure 6 Admin Page

Learning and Conceptual Difficulties

These were many and frequent. One of the main ones had to do with coming up with a solution to retrieve images from the web for a particular media item, resize it on the fly so it would fit in the Image Area on the user forms, and managing the images after that. To retrieve information for a specific media item, I started with the Web Query Wizard built into Excel. I quickly discovered its shortcomings (or my lack of knowledge in how to fully utilize its capabilities), chief among them its inability to retrieve images from a web page. I set about learning Dr. Allen's Agent class and used it to scrape the web for both the media details as well as the image. It took me a little while to understand the class but a small tutorial review from Dr. Allen himself got me pointed in the right direction (key takeaway: use the "savePage" method in the Agent class to save the source of the target webpage and manipulate the page from there). After knowing how to retrieve the image itself, I needed a way to resize the image so that it fit inside the Media Image area of the user form. I tried to figure out a way to do it with the tools in Excel but I could not find anything that worked. Dr. Allen suggested I use a program called ImageMagick that has some slick command line tools to manipulate images. I installed the program and went about figuring out how to use it. Dr. Allen gave helped me with the code to call it from within VBA, which turned out to be one line of code (well, it's actually two lines of code in my program, but it could really be just one) (Figure 7).

```
path = Chr(34) & "C:\Program Files (x86)\ImageMagick-6.6.9-Q16\mogrify" & Chr(34) & " -resize 139 " & Chr(34) & ThisWorkbook.path & "\tmp\" & imgName & Chr(34)
Shell path
```

Figure 7 Calling ImageMagick from the VBA

The above code performs the code in the "path" variable. At first, after this line of code, I was immediately calling the function to assign the picture to the Media Image area. However, for some reason, this would still display the original image that may have been too big for the area. Upon viewing the media item record again, the image would display the correct size. It seemed that the VBA code was moving too fast and not using the recently resized image. To fix this, I used Application.Wait to force the application to wait one second before continuing.

Another issue that came up while using ImageMagick was the fact that, during development, I was switching between 32-bit and 64-bit machines. This changed the path of the ImageMagick installation and if I didn't change it in the code, it would cause the program to error. I resolved this by checking a certain cell on the "Settings" worksheet as the program opens. If that cell is empty, it prompts the user to specify the ImageMagick install directory. If the cell is not empty but it turns out that the directory does not exist, the program will prompt the user to select the correct directory and it will write the directory path to the cell on the "Settings" worksheet.

Another issue I had was trying to make Excel work similar to a database, relating "tables" or different worksheets. I resolved this with auto-incrementing IDs for each media item entered into the database as well as borrowed items. I was able to use these unique IDs to perform look ups and, essentially, joins to combine the results from two different worksheets.

There were many other difficulties I encountered, but most had to do with typical debugging procedures. In fact, I'm still trying to get rid of all the bugs!