2014

# Movie Manager

Zakary Kubicek
*Grand Valley State University*

# Movie Manager

By
Zakary A. Kubicek
April, 2014

# Movie Manager

By
Zakary A. Kubicek

A project submitted in partial fulfillment of the requirements for the degree of
Master of Science in
Computer Information Systems

at
Grand Valley State University
April, 2014

---

**Jonathan Engelsma** **Date**

## Table of Contents

# Abstract

Despite the increase sales in entertainment options such as Video On Demand, digital purchases, and subscription streaming, physical disc media sales is still a multi-billion dollar industry. The purchase of physical media discs (such as Blu-rays and DVDs) continues to be a popular option for home entertainment [1]. Many consumers have created a vast collection of movies and it can be difficult to manage them all. These consumers may be asking the following questions. How many movies do I own? Do I already own that movie? Have I upgraded my DVD to a Blu-ray yet? If you own a collection of 30 or less movies these questions are easy to answer. But if you have a collection of several hundred these questions get more difficult. This is where the Movie Manager Android app comes in. It allows users to upload their movie collection to the cloud. Users can then rate movies, track whom they have lent a movie out to and manage what formats they own (Blu-ray, DVD, etc.). The app features a rich user interface that will allow users to see key information about a movie such as the release date, cast, MPAA rating, duration, synopsis, and more. This project involved the reimplementation of an Android app created for CIS 680 *Mobile Application Development*. The user interface was given a complete redesign and new features were added such as barcode scanning, cloud storage provided by https://parse.com/, user rating, and loan management. This study will also examine some of the analytics gathered after it was published to the Google Play Store.


# Introduction

I am one of many people who have a large movie collection. You will see me hovering over the 5 dollar bins at electronic stores like Best Buy, or getting up early to add the Black Friday discounted movies in my shopping cart on Amazon. Over the past decade I have acquired a large set of movies. Once you are known to have such a vast collection you become your friends and family's personal rental store. While I love to share the joy of movies with people I know, managing which movies I have given out to whom can be very difficult to handle. My sister-in-law who is notorious for borrowing a movie every time she visits finally told me I should make an app to track my movies. I was taking a course on Android development at Grand Valley State University back in the summer of 2010 so I thought I would give it a shot.

I was able to create a prototype of the Movie Manager app and publish it to the Google Play Store back in 2010. After the class was over I discontinued any extra development. To prevent the app from being unusable in the fall of 2013 I was forced to update a few APIs because they were being deprecated. That small amount of work got me interested in resurrecting this app and creating something that had a lot of great features. Throughout this three year hiatus I had been given dozens of enhancement ideas from user feedback. So it seemed perfect to learn from what I had done before and deliver something excellent that users would really enjoy. This project is a continuation and recreation of the Movie Manager app that I created back in 2010.

The goal of this project is to create a movie collection management Android app that is fun and easy to use. The user's data will be stored in the cloud using a third party data source. That way if the Android device is lost, stolen, or broken, the data will not be lost. Movie information such as release date, cast, MPAA rating, duration, and synopsis will be displayed to users. The movie information data will be provided by a third party open data source so it does not need to be stored on the user's device. The app will be able to find movies by either searching by title or by scanning barcodes on the back of a movie's case. The barcode data will also be provided by a third party open data source. To allow users to correct any incorrectly assigned barcodes, users will be able to add their own barcodes to their movies. To fully utilize that user driven barcode data, a crowd sourced data store will be populated with the correct barcodes to help facilitate future searches. The app will also allow users to rate their movies and record if they have lent them out to someone.

# Background and Related Work

As of April of 2014 there are several Android apps that allow you to manage your movies. Unfortunately all of the good ones cost money, have ads, or limit the number of movies in your collection. For the ones that are completely free, do not have ads, and do not have a limit on the number of movies in your collection they suffer from minimal features and a bad user interface.

The Good – My Movies Free

My Movies is an excellent Android app that allows users to catalog and manage their movie collection. It has lots of features such as a rich UI with multiple views (list, grid, and cover flow), synchronization across multiple devices, social integration, sorting, and filtering. They support Android, iPhone, Windows phone and also have a MAC and Windows application. But with such a good app comes a price. You can only add 50 movies to your collection on the free version. And if you want the Pro version you need to pay $5.99. This is one of the more expensive movie collection apps in the Google Play Store.

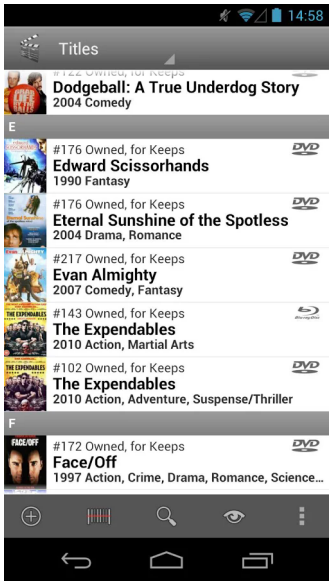


Figure 1 - My Movies List View


Figure 2 - My Movies Movie View

The Bad – Film Library

Film library is another example of a movie collection app that supports both a free and paid version. While the free version does not limit you to a maximum set of movies it does hide certain features such as different views for the movie collection and movie information like budget, revenue, homepage, and trailers. Its user interface is simple and plain. The paid version will cost you only $0.99 which is a five dollar deal compared to the last app.

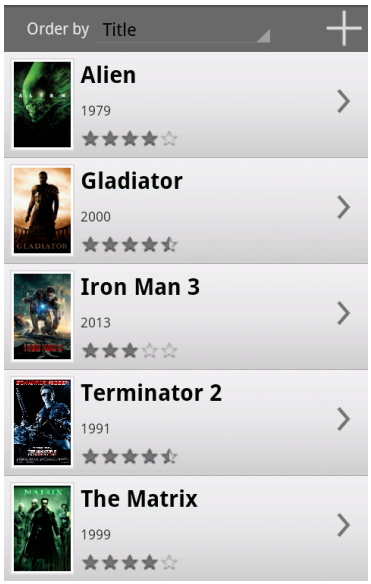


Figure 3 - Film Library List View

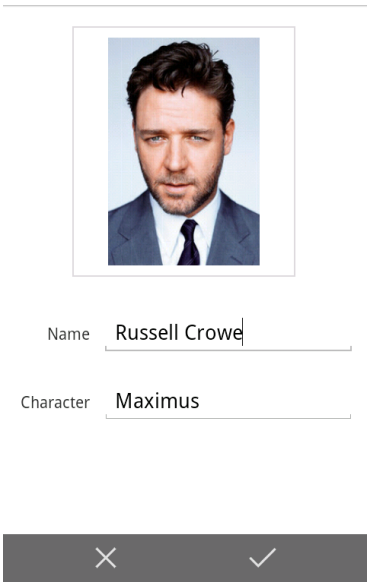


Figure 4 - Film Library Actor View

The Ugly – Movie Manager (no relation)

And finally there are some apps that are missing key features and look awful. The Movie Manager app created by Roni Isserles allows users to add movies manually, give them ratings, and check whether the user has seen them or not. None of the movie information is retrieved from an external source so everything must be found on the device. You cannot search for movies or scan barcodes. The app also sports a color array that is not easy on the eyes. This app while less desirable is completely free with no movie size limit.

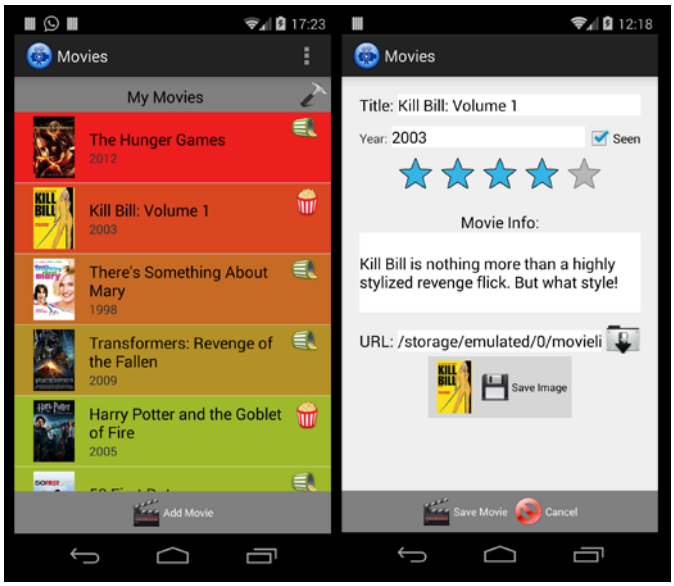


Figure 5 – Other Movie Manager

The goal of this project is to create a movie collection app that has a rich user interface and enough features to send it to the top of the free apps in its category.

# Program Requirements

This project is unique because it not only involves the creation of an app with exciting features but it must also sustain features found in the previous version. In order to keep the current users happy, special care was needed to prevent any old features from being removed. Therefore the following list of features needed to remain in the app:

- Users must be able to browse through their entire movie collection.
- The movie collection must be sorted by title and separated into sections corresponding to the first letter of the title. This sorting must ignore articles such as a, an, and the.
- Users must be able to search for movies by searching via a movie's title.
- Users must be able to add movies to the movie collection from search results.
- Users must be able to remove movies from the movie collection.

- Users must be able to view detailed information about a movie such as title, release date, budget, revenue, tagline, synopsis, and director.

By completing these requirements users currently using the old version of the app will be able to transition to the new app without losing any functionality. In summary these features allow users to perform CRUD (Create, Retrieve, Update, and Delete) operations on their movie collection.

The next set of features was added on top of the previous core features to help the usability of the app. These features along with the core features would complete the first version of the Movie Manager redesign. This version was published to the Google Play Store on March 27, 2014.

- Users must be able to save their data to an offsite data source to protect data in the event of their device being wiped, upgraded, lost, stolen, or broken.
- Users must be able to have a user account so that their movie collection can be associated to them.
- Since multiple devices could interact with a single user's account, we must provide a way to refresh the movie collection in order to synchronize them.
- Users must be able to add and remove multiple movies from their collection at a time.
- Include an updated version of Google Analytics for user behavior monitoring and error reporting
- Update the user interface from a plain Android 2.2 look to a standard Android 4.0 look.
- Add proper user interface scaling to support larger displays such as tablets.

These new features would allow the app to compete with newer apps while keeping the same functionality of the old version of the app. The analytics are used to identify where future improvements to the app should go, what is a typical user's workflow and show any errors that have occurred. The previous version of the app stored all of the movie collection data on the device in a local SQLite database. So if the app's data was cleared or the user upgraded to a new device they could not take their data with them. To avoid that problem a third party data source was used to store the data.

The final set of features was added and published to the app on April 23, 2014. These features were new enhancements to the app that both increased the value of the app and allowed it to compete with existing apps in the movie collection category.

- Users must be able to add movies by scanning the barcode on the back of a movie case.
- Users must be able to view the current rating of a movie and provide a rate of their own. The rate will be a score from 1 – 5.
- Users must be able to assign a person's name to a movie to indicate that it has been lent out to that person. There must also be a way to record that the movie was returned.

# Implementation

Cloud Storage

As mentioned earlier, in a previous version of the app the data was stored locally on the device that ran the app. Since this caused a lot of frustration for users who upgraded their devices and did not want to enter in their movie collection again it was a high priority to be able to upload the data off of the device and into the cloud. My initial thought was to use Google App Engine (GAE) to host a server and populate a NoSQL data store with all the data. I would have created REST endpoints to perform CRUD operations on the movie collection data. Having previously worked with GAE and knowing very little about the new REST endpoints API I chose to avoid weeks of configuration setup and use Parse instead.

Parse, which can be found at https://parse.com is a mobile app platform that stores data for you in a NoSQL data store, provides push notifications, and supports analytics. One large advantage of using Parse is that they provide you with a library to interact with their products in many different programming languages. There are libraries for Android, iOS, OS X, .NET, and JavaScript. All of these libraries use OAuth, HTTPS and SSL to securely make connections from the device and a REST endpoint to interact with your data store. You can also use their REST API directly to program in another language. So for Android, they provide you with a java library that will get you up and running in minutes. There is no server configuration necessary which is why this platform is very appealing. They scale very well which is why big customers such as cisco, the travel channel, the food network, and Sesame Street use them. They provide a very fair package of 1 million requests per month for free. The free package also allows you to write JavaScript server code (called cloud code) that can be run manually or on a schedule to interact with your data store. Parse was bought out by Facebook in 2013 so as an added bonus there is simple Facebook and Twitter integration when creating user accounts built into the API.

So after I replaced all of the interactions with the user's local SQLite database with Parse I also needed to allow users to upload their current movie collection to Parse. So upon launch of the app if they hadn't converted their movies over to Parse yet I prompted them to do so. Once all of the movies had been successfully uploaded to Parse I added an entry in the device's SQLite database indicating the date in which they performed the upload. The existence of that date allowed me to know whether to keep prompting the next time they opened the app. And for the users who did not want to upload their movies they could select a checkbox to never be prompted again.

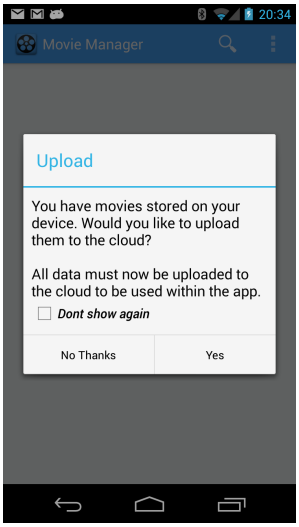


**Figure 6 - Upload Prompt**

Parse has a nice feature set when it comes to creating user accounts. You need at minimum to provide a username and password. Through the Parse API you can allow users to reset their passwords by sending an email with a reset link similar to many applications today. And as I mentioned before users can attach their Facebook and Twitter accounts to their Parse user. While these features are nice I did not take advantage of them. I only needed to associate movies to their users so I created a fake user account for each user identifying them by an authenticated Gmail account on the Android device. I could have just added the Gmail address as a field to the movie class but I wanted to create a Parse user account object for future Facebook integration.

Movie Data Source

My app leverages the free API from The Movie Database (TMDB http://themoviedb.org) to get its movie information. It not only provides text details such as a movie's title, release date, director, and cast information but it provides me with URLs to retrieve images to use in my app. All of the movie thumbnails, posters, and banners are provided by TMDB. By using TMDB I don't have to store any of the movie information the app displays. The TMDB API is a simple REST service that I call manually by performing HTTPS requests from my app. I also use TMDB to perform any searches for the app. This means if the movie cannot be found in TMDB then it cannot be added to a user's movie collection. To work around this a user can create a free TMDB account and create the movie themselves in TMDB. To help bring awareness of this work around I publicly advertise the use of TMDB's data in the description on my Google Play Store listing.

Movie Collection List View

The first view you see when you login is your movie collection displayed in a list. I replaced the old custom gradient header with a standard Android 3.0 action bar which is used to bring you to the search view, bring you to the settings view, and show the about dialog. The action bar also displays the number of movies in

your collection. That feature was removed upon the initial re-design but added back during the 2.2.0 release due to a request from a few users. The action bar also displays the delete option when movies are selected in the list. To select a movie in the list you can either perform the standard long press or by selecting the poster of the movie.

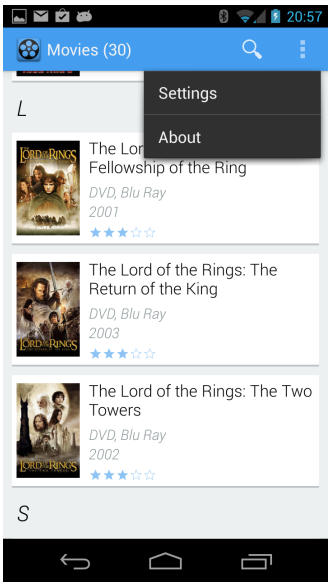


Figure 7 - List View

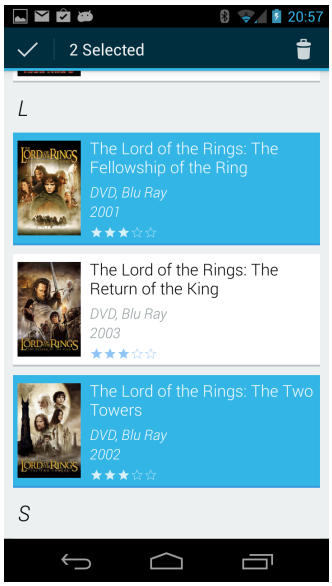


Figure 8 - Multi-Select

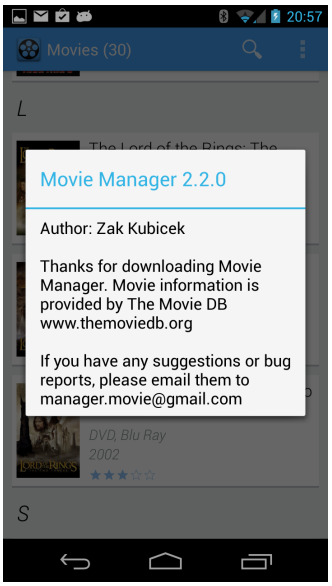


Figure 9 - About Dialog

The fonts used throughout the app are variations of the Roboto font which is now the default Android font used in Google's Android apps. The Roboto fonts can be downloaded from the Android Developers website [2]. An advantage of using the font is that it scales very well with different screen densities. All of the images loaded by the app use an open source library called ImageLoader. ImageLoader can be customized to provide you with multiple ways to cache images that you need to retrieve. The ImageLoader in the Movie Manager app is configured to use 2MB of in memory cache and 50MB of disc cache. Once an image is retrieved from a URL it is placed in both caches. Future fetches of the same image will check the cache first before loading the image from the URL. As the caches fill up the images are removed based on the least recently used method. One advanced feature that was added was to mimic the pull to refresh feature found in several Google Android apps such as Google+, Gmail, and the Email app. The pull to refresh feature works by scrolling to the top of a list and pulling down long enough to signify that you want to refresh the list. This feature was implemented using an open source library called ActionBar-PullToRefresh.

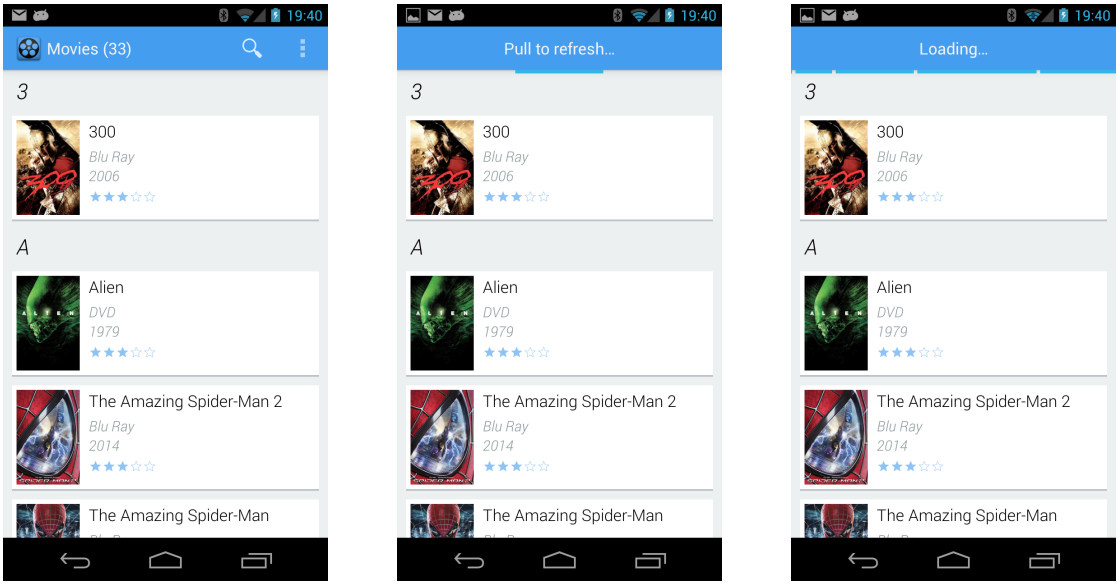


Figure 10 - Pull To Refresh

Search for Movies

By selecting the magnifying glass icon in the list view's action bar the user is brought to the search view. From here users can search for movies and add them to their collection. By selecting the magnifying glass icon a text view is displayed in the action bar so a user can type in a title and search for it. The search title is sent to TMDB using their REST search API in which I collect the results. The results display below the action bar in a similar list view as the movie collection.

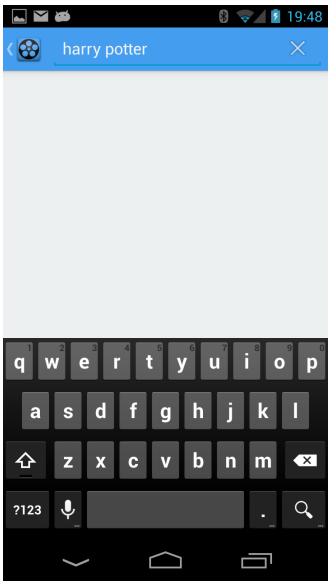


Figure 11 - Title Search

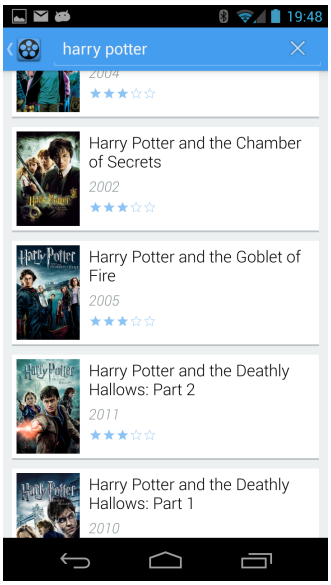


Figure 12 - Search Results

Users are allowed to select multiple results and add them to their library. Once the user selects the add icon a prompt is displayed asking what formats the movies should be in. Movies in the collection are allowed to have multiple formats. Therefore a user can have two Blu-ray copies and three DVD copies of the same movie. Once the movie is added an icon and text string are displayed indicating the movie was successfully added to the movie collection.

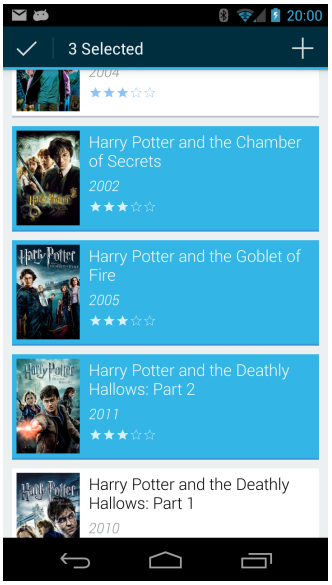


Figure 13 - Add Multiple Movies

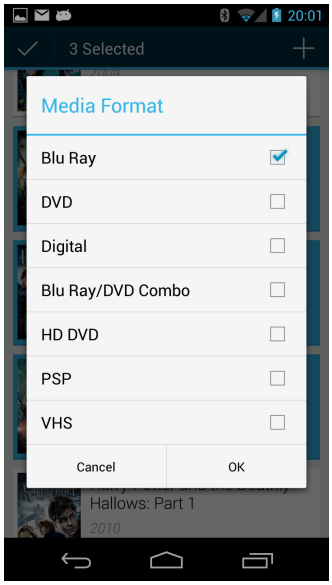


Figure 14 - Choose Media Format

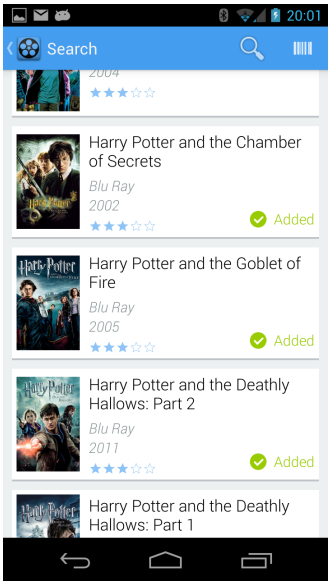


Figure 15 - Added Movies

With version 2.2.0 users were given the ability to search for movies by scanning a barcode typically found on the back of a movie case. I used a free scanning library known as ZXing (Zebra Crossing) to perform the actual barcode scan. Using this library you just need to send off the barcode scan intent and one of two things will occur. If the ZXing Barcode Scanning app has not been installed on the device, the user is brought to the Google Play Store where they can download it. If they already have the app installed they are directed to it and given the opportunity to scan a barcode. After there is a successful scan, the UPC text string is given back to the app. From here a three step process occurs which can be seen in Figure 16 to find the movie that matches the UPC.


Figure 16 - UPC Search Workflow

1. The first step is to check if any of the movies in the user's collection already have that barcode. Since users can manually add barcodes to their movies it was important to check this first so that a user can override any barcode they wish. If the UPC could not be found in the user's collection we proceed to step two.
2. The second step is to check the crowd sourced barcode data store. After users were given the ability to add their own barcodes manually I decided to use that data to create my own data store of UPCs. I created some cloud code on Parse to check for movies created/modified within the last 24 hours and determined their UPC code based on manually entered values from the users. To prevent incorrect values from being entered there must be a minimum of 10 identical UPCs for the same movie. I do allow for multiple UPCs for each movie and type (Blu-ray, DVD, etc.) since movies may be re-released with different UPCs. If the UPC could not be found in the crowd source data store we proceed to step three.
3. The third and final step involves asking the third party UPC search provider Semantics3 (S3) https://www.semantics3.com for the item that is associated with the UPC. S3 has a very extensive list of products so most UPC searches come back with an answer. A result from S3 provides a name of the product which must be parsed to try and determine the title of the movie, the format of the movie, and optionally the year the movie was released. Using the result's meta-data we may come up with multiple title options. After we have a list of titles we must query TMDB for each of them. To help with accuracy the year the movie was created is added to the query if it was extracted from the S3 result. If there are no results then a new TMDB query is run without the year. After checking each title, whichever query gave the least number of results would be the winner of the UPC search.

Now that there are results from the UPC search, they must be displayed and dealt with accordingly. If the result coming back is from the user's movie collection, then the result is shown as the only search result and the movie information is brought up as if the user had selected it from the list. If the result is not in the user's movie collection then it is assumed that the user wants to add the movie to their collection. If there is only one result and the TMDB search result's movie title matched the search query exactly then we assume that we found a perfect match and add the movie to the user's collection automatically. The result is still shown in the search results indicating that it was added to the collection. Otherwise we display the results in the search view's list. To aid in adding movies from the results, if we detected a media format (Blu-ray, DVD, etc.) that value is pre-populated when the user goes to add the movie.

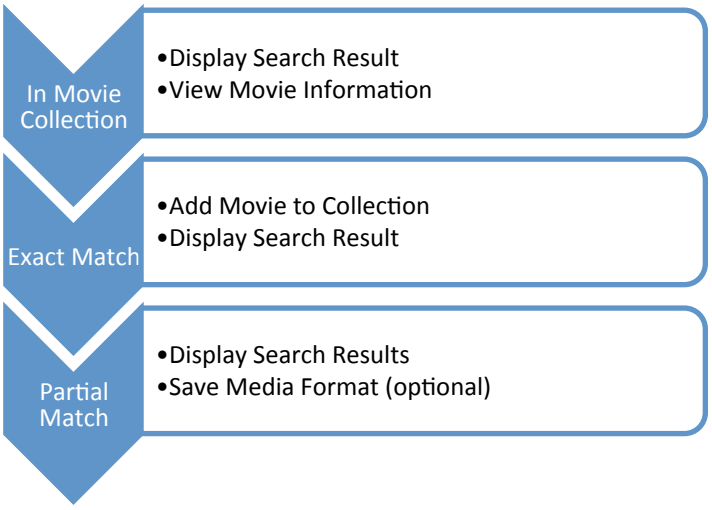


**Figure 17 - UPC Search Results Workflow**

Movie Information View

By selecting a movie list item you can view more details about the movie. The app currently displays, title, release date, MPAA rating, user rating, format, the number of copies if there are more than one, tagline, synopsis, budget, revenue, director and cast. Each cast member has an image provided by TMDB. This view also displays an image of the movie's poster and a banner at the top of the view. The banner image is separated from the rest of the content so that scrolling can be performed at different rates creating a parallax effect. This effect was popular in video games during the 80s and 90s. I was able to leverage an open source library called Paralloid to create this effect.

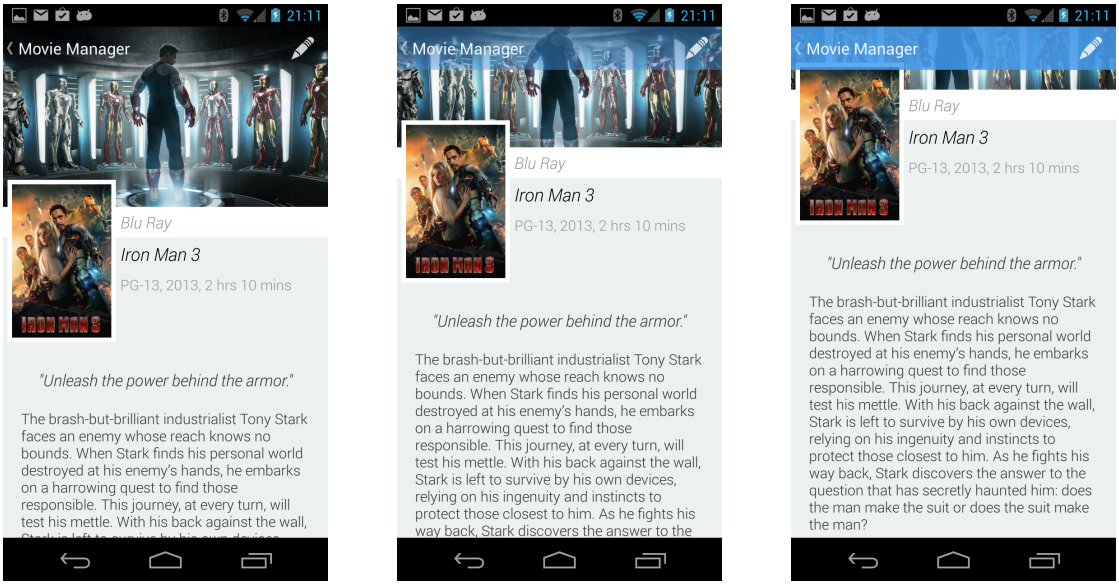


Figure 18 - Parallax Effect

Initially the action bar is transparent and as you scroll down it becomes opaque once you pass the banner. This allows the banner to take up the entire top of the view without a visual interruption from the action bar. To help display the white icons on a light colored banner, a small shadowing effect is placed at the top of the banner.

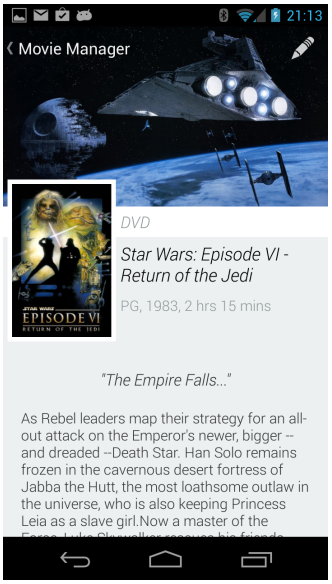


Figure 19 - Transparent Action Bar

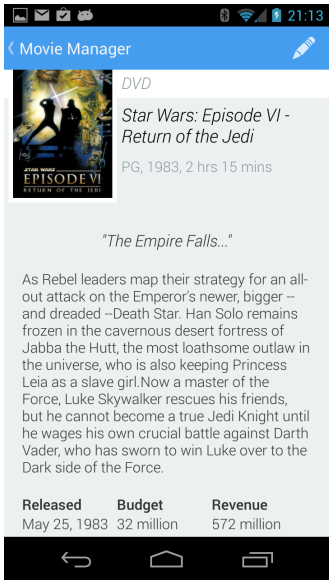


Figure 20 - Opaque Action Bar

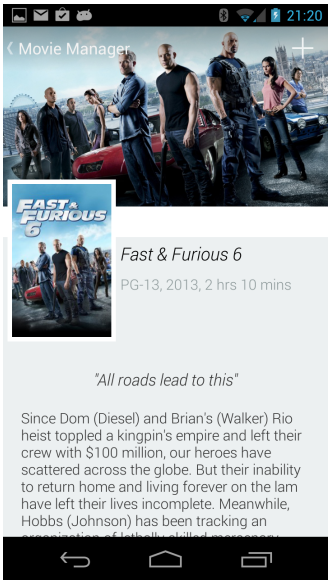


Figure 21 - Banner Shadowing

There is only one of two options shown in the action bar at one time. If the movie being viewed is not in user's movie collection then the option is to add the movie. Otherwise there is an edit option which will display the edit view. The edit view allows users to manage what formats of the movie they own, what their UPCs are and how many copies they own. To add a barcode to a movie format the user must select the barcode input box. The user will then be directed to the barcode scanning app mentioned earlier which can scan a barcode and populate it back to the edit view.

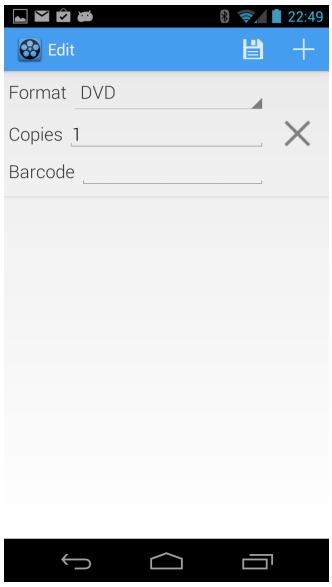


Figure 22 - Single Format

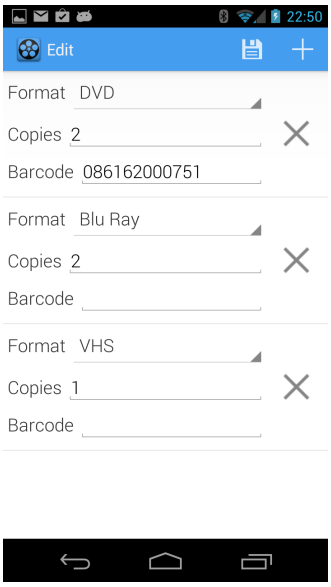


Figure 23 - Multiple Formats

In summary the use of the Parse data source proved to cut down a lot of time of implementing a server side data store. The app leverages several open source projects to help provide rich user interface features such as pull to refresh, image loading, and a parallax effect. References to the open source libraries can be found in the Appendices section of this report. Despite the recent work Android developers have put into Android Studio I used Eclipse as my integrated development environment to create my project. I was comfortable using it several years ago and thought it would be an easier transition getting back into Android development. I also used several sites to generate my icons for me for free. They are also listed in the Appendices section of this report. I used Bitbucket for my version control Mercurial repository. Not only was it used as a way to back up the code to an external system but it was helpful in creating versions of the code so that patch code can be isolated from the next release's code.

# Results, Evaluation, and Reflection

Deployments

*March 23, 2014 – Release 2.0.0 BETA*

The first release of the new app was distributed to a small testing BETA group. This group consisted of a small number of Android users I knew from work. I was able to leverage the ALPHA and BETA workflows provided by the Android developer platform to limit the release to only the BETA group. I created a Movie Manager BETA Testers Google+ community which was assigned to my BETA releases. Any new code version published as a BETA release became available to them in the Google Play Store within a few hours. I recommended that they all download the old version of the app so that the upload process could be tested as well.

*March 27, 2014 – Release 2.1.0*

After a few suggestions from the BETA group I released the new version of the app to the public a few days later. A nice feature with the Android developer platform is that they allow you to create a staged release. So on Thursday March 27th I only released the new version of the app to 20% of my user base. By the middle of the weekend I had not seen any crashes so I increased the rollout percentage to 50%. At the end of the weekend I decided to release the app to the rest of the public. I did get ahead of myself because as soon as Monday morning came around I did have several bugs reported that I needed to fix. Most of the issues were around users being unable to upload all of their movies to the cloud data store. Just like any other real world application the next week and a half was a complete detour from my initial plans. I was busy investigating and fixing bugs as well as adding enhancements for tracking errors. I upgraded my old Google Analytics code to version 3 to record any errors that occurred. The feedback I received ranged from "Great Job, I love the app" to "My movies are gone and I hate you". It does not take much to get a bad review. One user gave the app a two star out of five stars because the count of how many movies are in the collection was removed from the action bar. But along with a few bad reviews there was plenty of positive feedback.

Now that I am storing user information in my own data store I created a privacy policy for any interested parties. The Android developer platform provides the ability to add a privacy policy to the app's webpage within the Google Play Store. The privacy policy was created as a public Google drive document. Inspiration for the privacy policy was provided by the GVSU Mobile Application privacy policy and the MyMovies web application privacy policy.

*April 23, 2014 – Release 2.4.0*

This was the final release for the project that added several of the advanced features. These features include barcode scanning, user ratings, loan management, and the much requested feature the movie count. In fact each feature was in a different version of the app 2.x.0 which is why the public release skips from 2.1.0 to 2.4.0. The versions in between were released solely to BETA testers. This deployment was not a staged rollout. Because I needed all of the code out to the public for the presentation on this project, I needed the new code available in the Google Play Store by the 24th of April.

Analytics

Prior to the new release of the app the total install count was at 1,416 devices. Note that the previous version of the app was released in 2010 which means many of these installs are on devices that are no longer operational. The current install count as of writing this report is down to 1,378 showing a loss of 38 devices. While this decline in users may seem detrimental it is not worrisome. There has been a constant growth in new users every day. The prompt of updating the app to a new version may have notified users of an app they downloaded a long time ago that they no longer use which could have encouraged them to uninstall it. As for user activity, prior to the new release Google Analytics recorded roughly 1 – 2 sessions per day but with the new version of the app Google Analytics has recorded roughly 10 sessions per day. The app is getting a lot more user interaction than it used to. Active users have jumped from 2 prior to April 1st to 20 afterwards. The movie manager app is available in all countries supported by the Google Play Store but is only translated into US English. The top five countries for the app are the United States, Canada, the United Kingdom, Australia, and Mexico with the United States dominating the installs with 77.43%. As of writing this report I am up to 249 unique parse users and over 41.5 thousand movies. That indicates that the average user has roughly 167 movies in their collection. Seeing as those users would have exceeded the quota on many free movie collection apps I would say this app was successful in creating a useful free movie collection app that provides sufficient features and does not suffer from a bad user interface.

# Conclusions and Future Work

An app that has been on the Google Play Store for over three years does not have a short list of future improvements. Over the last three years I have gathered requests for enhancements to the app. Many of these enhancement requests tend to be features commonly found in paid movie collection apps. Based on the most popular requests I would like to work on the following features in the future.

Support for TV – As of the fall of 2013 TMDB offers TV information. This new API can be integrated into my app so that seasons of TV shows can be added to a user's media collection.

Wish list – Provide a separate list of movies that you wish to own. This would be helpful when walking through an electronics store such as Best Buy and being able to add movies to the list for future reference. Users can then watch for movie sales and see if anything on their wish list is for sale.

Social Integration – Add Facebook integration into a user's account. An activity stream can be created based on what activities a user's Facebook friends are doing with the app. Adding social integration could open the doors to creating a Goodreads environment for movies.

Alternate Movie Collection Views – Allow users to view their movies in a grid or cover flow layout.

In addition to these enhancements there will be an investigation into some of the current features similar apps have that this app does not. Filling in those gaps will also lead to an increased interest in this app. As of writing this report I have not seen an Android app that delivers these features with a rich user interface at such a cheap price. I hope its popularity continues to grow.

# Bibliography


[1] L. Schaefer, "DEG Year End 2013 Home Entertainment Report," The Digital Entertainment Group, 2014.

[2] "Typography," [Online]. Available: http://developer.android.com/design/style/typography.html. [Accessed 17 April 2014].


# Appendices

Movie Manager App

https://play.google.com/store/apps/details?id=org.kubicek.mediamanager

Source Control

*Bitbucket* [ https://bitbucket.org/ ]
Mercurial code repository

Open Source Libraries

*ActionBar – Pull To Refresh* [ https://github.com/chrisbanes/ActionBar-PullToRefresh ]
Used to refresh an Android ListView by pulling down at the top of the list

*Android Universal Image Loader* [ https://github.com/nostra13/Android-Universal-Image-Loader ]
Used to load and cache images for the app

*Paralloid* [ https://github.com/chrisjenx/Paralloid ]
Used to create parallax effect

*Smooth Progress Bar* [ https://github.com/castorflex/SmoothProgressBar ]
Used to show progress/activity is occurring in the background. While I used this directly, it was also a dependent of the Pull To Refresh library

*ZXing Barcode Scanning* [ https://github.com/LivotovLabs/zxscanlib ]
Used to scan barcodes

Android Icons

*Android Asset Studio* [ https://android-ui-utils.googlecode.com/hg/asset-studio/dist/index.html ]
Used to create action bar icons

*Icons4Android* [ http://www.icons4android.com/ ]
Used to get free icons