EPTA

eBay Pricing Tool Assistant

Kevin Kern

PROJECT FOR CSC385 SEMINAR IN COMPUTER SCIENCE SPRING 2015

March 30, 2015

Table of Contents

[1. Introduction 4](#_Toc415832880)

[1.1 Summary 4](#_Toc415832881)

[1.2 Requirements 4](#_Toc415832882)

[1.3 Numbers 4](#_Toc415832883)

[1.4 Terminology 4](#_Toc415832884)

[1.5 References 5](#_Toc415832885)

[2. Project Description 5](#_Toc415832886)

[2.1 Solution 5](#_Toc415832887)

[2.2 Task Plan 5](#_Toc415832888)

[2.3 Background 5](#_Toc415832889)

[3. Functional Requirements 6](#_Toc415832890)

[3.1 Technologies 6](#_Toc415832891)

[3.2 Matching Algorithm 6](#_Toc415832892)

[3.2.1 Background 6](#_Toc415832893)

[3.2.2 Solution 7](#_Toc415832894)

[3.2.3 Implementation 7](#_Toc415832895)

[3.2.4 Example 8](#_Toc415832896)

[3.3 Libraries 9](#_Toc415832897)

[3.3.1 Connector 9](#_Toc415832898)

[3.3.2 Linker 9](#_Toc415832899)

[3.3.3 Parser 9](#_Toc415832900)

[3.3.4 Query Builder 10](#_Toc415832901)

[3.3.5 Store 10](#_Toc415832902)

[3.3.6 Fork 10](#_Toc415832903)

[3.4 Controllers 10](#_Toc415832904)

[3.4.1 Search 11](#_Toc415832905)

[3.4.2 Admin 11](#_Toc415832906)

[3.4.3 API 11](#_Toc415832907)

[3.4.4 AJAX 11](#_Toc415832908)

[3.4.5 Main 11](#_Toc415832909)

[3.5 Configuration Files 12](#_Toc415832910)

[3.5.1 Keys 12](#_Toc415832911)

[3.5.2 Application 12](#_Toc415832912)

[3.6 Models 13](#_Toc415832913)

[3.6.1 KAK\_Model 13](#_Toc415832914)

[3.6.2 Other Models 13](#_Toc415832915)

[3.6.3 Entity Relationship Diagram 14](#_Toc415832916)

[3.7 Use Cases 14](#_Toc415832917)

[3.7.1 User Search 14](#_Toc415832918)

[3.7.2 API User Use Case 15](#_Toc415832919)

[4. Predicted Results 16](#_Toc415832920)

[4.1 Information Retrieved 16](#_Toc415832921)

[4.2 Sorting 16](#_Toc415832922)

[4.3 Information Stored 16](#_Toc415832923)

[4.4 Results Screen 16](#_Toc415832924)

[5. Predicted Roadblocks 17](#_Toc415832925)

[5.1 API Item Restrictions 17](#_Toc415832926)

[5.1.1 Description 17](#_Toc415832927)

[5.1.2 Implication 17](#_Toc415832928)

[5.1.3 Solutions 17](#_Toc415832929)

[5.2 EBay calls per day 17](#_Toc415832930)

[5.2.1 Description 17](#_Toc415832931)

[5.2.2 Implication 17](#_Toc415832932)

[5.2.3 Solution 17](#_Toc415832933)

[6. Works Cited 18](#_Toc415832934)

# 1. Introduction

EPTA, EBay Pricing Tool Assistant, serves to assist end users in selecting optimal pricing for their EBay items.

## 1.1 Summary

What ways can a user predict the best selling option for an online auction tool such as eBay? Currently, eBay provides simple estimates for the price your item should be listed at. The eBay Pricing Tool Assistance plans to correct this issue by providing enhanced data about a given item. The data will be based on both live and past auction data collected from eBay. Using this data, EPTA can sort and categorize items in a way that provides useful information to the end user.

## 1.2 Requirements

This system must perform the following tasks:

* Pull data from the EBay Finding API
  + The amount of items pulled should be significant enough to perform calculations on
* Store data from EBay in an internal database
* Employ a strategy for determining an item of interest
* Provide some form of output to the user

## 1.3 Numbers

The estimated number of users expected to use the system is currently unknown. Based on the limited set of data we can assume that at least one user will be using the system per day. Ultimately it should support a number of users based on the EBay API restrictions. Of which, only 100 items can be returned from EBay per API call. The total number of calls is restricted to 5,000 per day. If the system uses 5 calls per search, and a typical users searches 5 times per day. Then the maximum number of users per day can be assumed as 200 without bypassing EBay API restrictions.

## 1.4 Terminology

|  |  |
| --- | --- |
| Term | Definition |
| Tag | A piece of a string that has been deemed by the application to have useful information. This includes various symbols, model numbers, adjectives, and many others. Examples: hot, open, X345y |
| Tag Type | A way to define a given set of attributes for a tag. For example, title tags (tags that come from item title strings) have a specific set of values to determine how title strings are broken down. |
| Code Igniter | An open source framework that provides simple tools for managing database connections, urls, and organization of your application. The underlying framework allows a user to divide code into a MVC (Model, View, Controller) design. This is how EPTA is organization. |
| eBay | Refers to eBay.com the online auction website |
| Local Data | Refers to any data that is already stored on the application's internal database. For example, a search can pull local data to add to the list of compared objects. This becomes increasingly more valuable as more data is stored. |

## 1.5 References

Included as an MLA formatted page located at the end of the document.

# 2. Project Description

The following sub sections provide an brief overview of the proposed project solution and task plan for implementation of the project.

## 2.1 Solution

This system will use a database to store EBay and local data. This problem will be solved by implementing a series of libraries that are capable of parsing, organizing, and storing various pieces of EBay data. This data will be used to perform calculations for predicting the optimal price for the item.

## 2.2 Task Plan

The aforementioned solution will be carried through the following steps:

* Using a Database
  + Create an ERD
  + Implement the ERD
  + Create back end models for manipulating the data
* Implementing libraries
  + Create a class for each of the following:
    - Building query strings according to the EBay specifications
    - Creating GET requests concurrently to reduce load time
    - Parsing and sorting incoming EBay data
    - Storing of advanced objects such as tags
  + Use the libraries through controllers to achieve optimal results
* Perform calculation for predicting optimal price
  + Libraries will provide data ready for calculations
  + Search controller will modify and output the data as needed
* Output
  + Will be passed into a view for displaying to the end user

## 2.3 Background

This project originates from a project thought up by Dr. Kaplan. Originally, the project was incapable of storing data. The first solution made by a previous student involved using JavaScript Ajax calls to retrieve and output data. It was riddled with errors and did not conform to the goals of this project.

To achieve this solution, I have researched the EBay Finding API along with their other API options. The goal of the research was to discover the formatting of their return data for any given API call. Additionally, I researched into the API restrictions which include the maximum items per call and maximum calls per day (see 1.3).

My research also led me to a series of API call examples, which allowed me to note that the previously submitted project was simply a copy and paste job from the API examples. This forced me to drop all of the content, excluding color scheme and images, from the previous student.

# 3. Functional Requirements

The backend of the system is based on the Code Igniter framework. It employs multiple custom classes declared as libraries to achieve the desired results.

## 3.1 Technologies

A list of technologies to be used in the project and what they are used for.

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| PHP | Backend Scripting |  |
| MySQL | Database Scripting |  |
| Code Igniter | PHP Framework | Open Source framework written in PHP. Providing simple adapters for commonly used concepts such as database, session, and routing. This is primary used to provide a simple MVC organization for the codebase |
| JavaScript | Frontend Scripting | Basic front end scripting language used to manipulate the DOM and fire off AJAX requests |
| jQuery | JavaScript Framework | A JavaScript framework built to speed up application development and provide handy tools. It is used to send simple AJAX requests and modify the DOM with ease. |
| Bootstrap | CSS Template | Provides simple base styling for the user interface |

## 3.2 Matching Algorithm

One of the most critical components of this application is its ability to dictate which items are truly part of a price grouping. If a listing of items contains both "iron man red gloves" and "iron man weight set", it's results can be significantly inaccurate and meaningless. It is absolutely essential to ensure all items considered are indeed representing the same physical item.

### 3.2.1 Background

EBAY does provide similar search results given your keywords. However, more often than not, the search results will contain a large number of outliners. This is due to the loose item definitional nature that allows EBAY users to sell just about anything. EBAY does provide their own internal product id which are based in some logic. For many searches, these can be used to group similar items. However, even in instances of very common items, some items are listed with no product id.

### 3.2.2 Solution

The solution to this, is to implement a "Matching Algorithm." This algorithm focuses on the item's title string, as it is one of the limited ways to get inside the head of a EBAY seller. The theory is rooted in the assumption that an EBAY user is very likely to describe their item in high detail within the title itself. EBAY provides an initial interface for their users to sell items (EBAY, 2015). The critical piece of information is the title string which serves as a literal representation of your real world object (EBAY, 2015). This fact will be used as the cornerstone of the matching algorithm.

First, title strings are made up of a series of words. Each word either alone or in conjunction with another word will describe a given item along with generally a list of other items. By linking together words we can make descriptive statements about physical objects. In our case, a word will be included in a fundamental object called a tag. A tag describes an object in the same way a word would. However tags can include a largest arrangement of symbols, numbers, and letters.

### 3.2.3 Implementation

The first step to comparing tags is breaking down a title string into relevant components. The primary slicing occurs by slicing a string anywhere a space occurs. We can assume spaces separate words because a spacebar is generally the most predominate method of separating ideas when using a keyboard. The next step is to strip out any unnecessary symbols. We must now make another assumption in determining which symbols are meaningless. Based on sampling and trials, we will use a simple list including parenthesis, brackets, quotes, commas, semi-colons , colons, double dashes, and a few others. Basically we need to remove any symbol that doesn't make sense when taken out of a sentence.

The next step is to count the number of occurrences for each tag over a list of items. This is achieved by grouping exact match tags and counting the number that exist. Each group of same tags receives a weighted tag score based on the following equation:

**Weighted Tag Score**   
 = (Number of Single Tag Occurrences / Number Of Total Non Unique Tags)

Now, by looking at the tags within a single item's title, we can sum up all the tags weighted tag scores (WTS) to achieve a Total Item Rating.

**Total Item Rating** = ( WTSTag1 + WTSTag2 + ... + WTSTagN )  
 Where n = Total number of tags linked to item

This total item rating will tell us how closely a given item matches the rest of the items within a group of items.

### 3.2.4 Example

Given The Following Items Are Returned:

* bat man gloves red
* bat man gloves red Christmas
* bat man gloves red
* bat man gloves
* bat man hat
* dog man oven mitt

**The Matching Index**

{bat, 5}, {man, 6}, {gloves, 4}, {red, 3}, {hat, 1}, {Christmas, 1}, {dog, 1}, {oven, 1}, {mitt, 1}

**Total Non unique Tags:** 23

5+6+4+3+1+1+1+1+1

**Applying The Matching Algorithm**

bat: 5/23, man: 5/23, gloves: 4/23, red: 3/23, hat: 1/23, Christmas: 1/23, oven: 1/23,

mitt: 1/23

**Total Item Ratings**  
bat man gloves red 5/23+6/23+4/23+3/23 = **18/23**

bat man gloves red Christmas  5/23+6/23+4/23+3/23 = **18/23**

bat man gloves red 5/23+6/23+4/23+3/23 = **18/23**

bat man gloves  5/23+6/23+4/23 = **15/23**

bat man hat  5/23+6/23 = **11/23**

dog man oven mitt 6/23 = **6/23**

The highest scores are the ones that are the most relevant where anything lower is less relevant. As you can see, the first three items are very similar excluding one outliner word that doesn't appear in any other item. These are simply discarded, acting as if the user added an extra symbol or name that doesn't refer to the object whatsoever.

What does this really do? Well say EPTA gets 500 items from a search. That list happens to contain 5 different physical items. If one person were to look at each item they would say each of the 5 are completely different items. Since a computer does not have this ability, we can use this matching algorithm to determine which items are the most likely based on occurrences.

## 3.3 Libraries

As outlined in the task plan, libraries are a critical component of this application. The following are the abstract concepts and functionality that formula each individual library required for this application.

### 3.3.1 Connector

The core library class which many other libraries derive from. The connector simply provides a reference to the Code Igniter framework, thus allowing them to access the various database models that have been included in the application.

### 3.3.2 Linker

The linker extends the connecter and is used to build relationships between pre determined objects through a relational database. The goal is simply to connect two or more primary ids so that they can be accessed as required by the application. One of the key relationships is between tags and items.

### 3.3.3 Parser

The customer data parser is the pivotal component of the application. Its primary goal is to take in queried data and turn it into something tangible for the application.

#### 3.3.3.1 Scan Method

The scan method takes in a array of items with no preface of where it comes from (locally or eBay API). This method invokes various private methods for manipulating the data. It also stored various pieces of information into logical arrangements that can be obtained by a controller for either other calculations or output to a user.

#### 3.3.3.2 Add Common Method

The add common method simply counts common bits of information about an item being scanned for future comparisons. This is done by locating a given common parameter type (title, product\_id, condition, subtitle, etc) with a privately stored variable. Based on its definition, the method manipulates the data and storing it accordingly.

Some of the factors linked to a given common type are:

* Number of items having the same value for the common type
* The summation of all current price values (used for averages)
* The minimum price found
* The maximum price found
* A list of item ids within this common grouping

#### 3.3.3.3 Build Common Method

The build common method is used to compile a list of all the most common values within a common type grouping based on the number of items within the group. This will result in a list of top matching common factors, such as (Title: Charlie Brown Blankets or Image: http://www.google.com/google\_image.png).

#### 3.3.3.4 Explode Title Method

The explode title method is used to slice up an item's title into logical arrangements called tags. Based on the parameters, the tags can either be stored in the parser or returned back to the caller. The slice is based on values contained within a given tag type. In this case, only the title tag type is considered.

#### 3.3.3.5 Get Match Rating Method

This method is responsible for determining an item's total matching rating (defined in 3.2.3: Matching Implementation). This is done by adding up all of the weighted tag scores for each tag linked to a given item. If the \* symbol was included before any search keyword, that tag will be double weighted. For this case, searching "\*Police Donuts" will put more emphasis on police over donuts. This is not necessarily a end user feature, but more of a testing feature.

Get match rating works by taking in a title string and exploding it with explode title (Defined in 3.3.3.4). It then compares each tag against the full set of tags found over the search set.

### 3.3.4 Query Builder

The query builder library is responsible for crafting well structured query strings. Each query string is consistent with the EBAY API guidelines. The major factors to consider are: EBAY API method name, keywords, page number, and item filters. Item filters are specified by the EBAY finding API (EBAY Developers Program, 2015).

### 3.3.5 Store

The store library is responsible for storing data that requires advanced manipulation to store. It will consist of a single function that will be responsible for conforming to the rules of tag storage and manipulating tags so that they can be stored in the database. Outside of this project scope, this library can be used to handle more advanced storage options.

### 3.3.6 Fork

Provides the ability to fire off HTTP requests in ways other than in series. The fork can fire requests as both background processes or concurrent (parallel) processes. The core usage is for obtaining a large number of resources from the eBay API without tying down the user experience.

## 3.4 Controllers

Controllers do exactly as their name implies, they control data and output it to views. The previously mentioned libraries are responsible for passing data down to each controller.

3.4.1 Search  
This is where all of the magic happens. The search control sets up the eBay API calls and deals with the libraries as they process the data. The controller is responsible for invoking the models to store database information.

3.4.2 Admin  
The admin controller is the starting point for an admin panel for the application. The goal is to provide a top level administrator of the application with the ability to quickly update and change the information contained within the application.

3.4.3 API  
This controller will control the output of the API documentation as well as the provide simple endpoints for manipulating data in the database.

3.4.3.1 getTags method  
Endpoint used to retrieve a list of all tags in the system

3.4.3.2 getTagsByItemId method  
Endpoint used to retrieve a list of all tags linked to a given internal item id

3.4.3.3 getTagsByEbayId method  
Endpoint used to retrieve a list of all tags linked to a given external eBay item id

3.4.3.4 getItems method  
Endpoint used to retrieve a list of items in the system

3.4.3.5 getCategories method  
Endpoint used to retrieve a list of categories in the system

3.4.4 AJAX  
This controller will provide additional HTTP GET endpoints specifically designed to feed additional data into the views on demand. The use of this will permit some of the options on the search screen to remain asynchronous, thus preventing additional wait time for the user. Currently the controller will only require one method.

3.4.4.1 getItemsById method  
This method will return JSON data given a specific item id fed through HTTP Post. This will be used to expand any sub grouping of items within the search pages. This will dramatically reduce the amount of data the application has to pass down on a single search.

3.4.5 Main  
This controller simply outputs the main home page. Out of scope for this project would include adding user session controls to this controller.

## 3.5 Configuration Files

Code Igniter itself does come with a whole mess of configuration files, however there are some custom built one that are needed to quickly change settings within the application.

3.5.1 Keys  
The keys configuration file provides all of the eBay connection information for performing eBay API calls.

The following table represents the required values

|  |  |
| --- | --- |
| Configuration | Definition |
| endpoint | The endpoint for the Finding Service, which should never have to change unless eBay changes their API |
| version | Once again, this is corrected by eBay and using the standard value will not make a difference |
| appid | This is the application id provided by the eBay developers program |
| devid | This is the developer id provided by the eBay developers program |
| certid | This is the certificate id provided by the eBay developers program |
| globalid | Is default to United states but can be changed to get results from other countries |

3.5.2 Application  
The application configuration file provides settings for the application itself. This is where an administrator can play around to get better results.

The following are some of the values required in it.

|  |  |
| --- | --- |
| Configuration | Definition |
| Entries Per Page | The number of items to get from eBay per API call where 100 is the maximum number |
| Pages Per Search | The number of calls to make to eBay during one search, as search the total results would be (entries per page \* pages per search) |
| Enable Recent Search | Simply determines if the application should remember and store the search keywords |
| Enable Local Results | Determines if the application should pull any related items from its local database and add it to your price consideration |
| Enable Tag Collection | Allows the application to store tags that it considers as a background process that runs with the search functionality. |
| Enable Title Matching | Allows the search to remove returned eBay items using the results from the matching algorithm. |
| Title Matching Tolerance | Determines what level of match an item must make in order to be considered in the results. |

# 3.6 Models

Models are used to collect information from a database. The structure of this application has each database table created as its own database model. Each database model extends a base model called KAK\_Model.

## 3.6.1 KAK\_Model

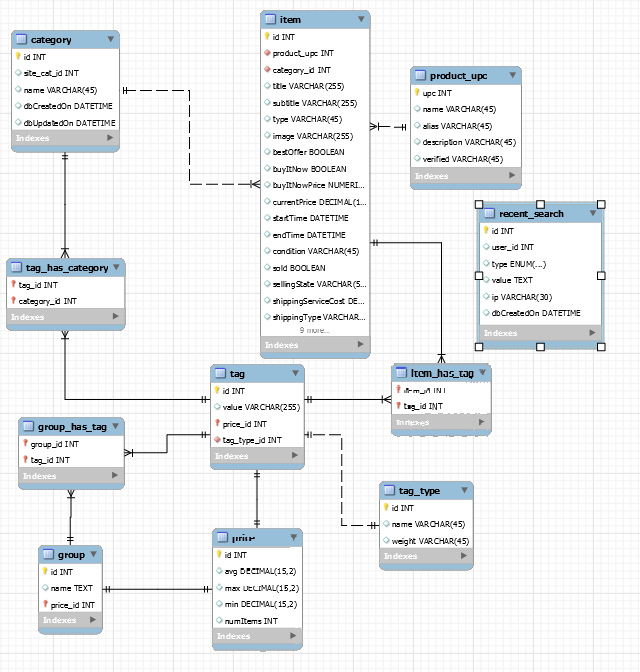
This is the core of all models. Each database model extends this classes and in turns is granted all of its CRUD operations. The KAK\_Model contains all of the basic methods for creating, reading, updating, and deleting information from the database. This allows universal CRUD operations to be performed on all datasets, preventing the need for a complex, non-uniform structure of manipulating data.

The main functions are insert, delete, get, update, and exists. Where each matches with its CRUD counterpart except for the exception of exists, which simply checks if a given object exists in the database.

3.6.2 Other Models  
As previously mentioned, all other models are based on the tables in the database and simply extend the KAK\_Model. Below is a full listing of models with a postfix of \_m used to avoid confusion.

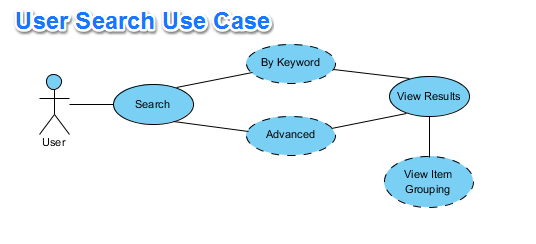
* Category\_m
* Group\_has\_tag\_m
* Group\_m
* Item\_has\_tag\_m
* Item\_m
* Price\_m
* Product\_upc\_m
* Recent\_search\_m
* Tag\_has\_category\_m
* Tag\_m
* Tag\_type\_m

3.6.3 Entity Relationship Diagram  
The following is an ERD used to represent the structure of the MySQL database and a representation of the models in the system.

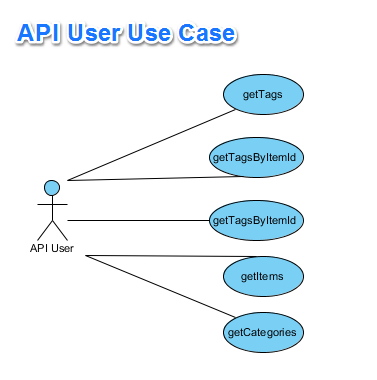


## 3.7 Use Cases

3.7.1 User Search  
The following is a use case showing the action of user proceeding through the search process.



3.7.2 API User Use Case  
The following is a use case showing the methods available to a user accessing the EPTA API.



# 4. Predicted Results

The following are some of the expect results.

## 4.1 Information Retrieved

The application will be able to retrieve at least 100 items per search on the assumption that eBay can return that number of items. A more optimal goal would be to retrieve 500 item per search using the capabilities of the fork library.

4.2 Sorting  
The application will be able to sort the returned results into logical grouping of items. It will be able to find the most commonly occurring pieces of data within the set of items including: product id, title, image url, shipping type, listing type, etc.

4.3 Information Stored  
The application will store all of the eBay items into its internal database given a table structure resembling the XML data retrieved by eBay item. It will also store and related tags from the matching algorithm process, thus linking them to an item. Tags will also be linked to pricing groups, providing the average, minimum, and maximum prices within a grouping.

4.4 Results Screen  
The application will provide a results screen based on a user's search. This screen will include but not be limited to:

* Stats about the item of interest
* List of common factors linking items
  + Including the average, minimum, and maximum price of each sub grouping
* A listing of the most occurring tags in the search results
* The ability to refine your search based on the resulting tags
* Some form of suggestion based on the results in which the user should price their item

# 5. Predicted Roadblocks

Majority of the project roadblocks faced in this project come from either the eBay API or the hardware in which the application is running on.

## 5.1 API Item Restrictions

5.1.1 Description  
EBay API restricts item results to 100 items per API call (EBAY, 2015).

5.1.2 Implication  
The maximum live results considered for a given search would be subject to this limit thus relying heavily on local resources to build an item list significant enough to pull relevant data from.

### 5.1.3 Solutions

* Solution One
  + Bypass the API limit by performing multiple calls in series for one search.
  + Downside
    - Will require additional resources and may blog down the server causing extended delay times for the end user.
* Solution Two
  + Provide the user with an option on the search results page to resend the search starting at the next available page of items from eBay.
  + Downside
    - Requires user to do additional actions.

## 5.2 EBay calls per day

### 5.2.1 Description

The eBay Finding API restricts applications to a limit of 5000 API calls per day (EBAY, 2015).

5.2.2 Implication  
Given that multiple users were using the application, and say five API calls are used per search, the limit would be achieved very quickly.

5.2.3 Solution  
Create multiple eBay applications. Use a custom router to direct the search requests on various API keys using some sort of round robin distribution system.

* Downside
  + Against the eBay API terms of service

# 6. Works Cited

EBAY. (2015). *API Call Limits.* Retrieved March 24, 2015, from Ebay Developers Program: https://go.developer.ebay.com/api-call-limits

EBAY. (2015). *Create Listing*. Retrieved March 24, 2015, from EBAY: http://csr.ebay.com/cse/start.jsf

EBAY Developers Program. (2015). *API Reference*. Retrieved March 26, 2015, from eBay Finding API: http://developer.ebay.com/DevZone/finding/CallRef/index.html

EBAY. (2015). *eBay Developers Forum*. Retrieved March 26, 2015, from EBAY Developers Program: https://forums.developer.ebay.com/

EBAY. (2015). *Users Guide*. Retrieved March 26, 2015, from eBay Finding API: http://developer.ebay.com/DevZone/finding/Concepts/FindingAPIGuide.html

ProgrammableWeb. (2014). *eBay API*. Retrieved March 27, 2015, from ProgrammableWeb: http://www.programmableweb.com/api/ebay